Machine Learning

Module 2

MODULE-2	SupervisedLearningRegressionand Classification Algorithms	22AIM52.3	8Hours
Regression: IntroductiontoRegression-RegressionModels-LinearRegression,PolynomialRegression-. **DecisionTrees:**IntroductiontoDecisionTrees-TreeConstruction,SplittingCriteria,andPruning-Handling Missing Values and Categorical Features- Gini Index-ID3-CART.			

---

1) Regression in Machine Learning

1. Definition

Regression in Machine Learning is a **supervised learning technique** used to **predict a continuous numerical value** based on one or more input (independent) features.
It helps in finding **relationships between variables** so that predictions can be made.

2. Variables in Regression

- **Dependent Variable (Target):**
  The variable to be predicted.
  Example: House Price.
- **Independent Variables (Features):**
  The input variables that influence the prediction.
  Example: Locality, Number of Rooms, Size.

3. Nature of Output

Regression problems deal with **continuous output variables** such as:

- Salary
- Weight
- House Price

The simplest regression model is **Linear Regression**.

4. Applications of Regression

1. **Predicting Prices:**
   Used to predict house prices based on size, location, and features.
2. **Forecasting Trends:**
   Forecast sales or demand using historical data.
3. **Identifying Risk Factors:**
   Analyze patient data to find heart disease risk.

4. **Decision Making:**
   Recommend which stock to buy based on market trends.

5. Advantages

- Simple and easy to understand.
- Interpretable results.
- Can handle linear relationships effectively.
- Useful for prediction and forecasting.

6. Disadvantages

- Assumes linearity between variables.
- Sensitive to **multicollinearity** (high correlation between features).
- Not suitable for complex non-linear data.
- May underperform when relationships are non-linear.

7. Types of Regression Models

Type	Description	Example / Use Case
1. Simple Linear Regression	Predicts target using one independent variable; assumes linear relation.	Predicting house price using size only.
2. Multiple Linear Regression	Extends simple regression using multiple independent variables.	Predicting price using size, location, and rooms.
3. Polynomial Regression	Models non-linear relationships by adding polynomial terms.	Predicting population growth over time.
4. Ridge & Lasso Regression	Regularized models that penalize large coefficients to prevent overfitting.	Used when dataset has many features.
5. Support Vector Regression (SVR)	Based on SVM; fits best hyperplane for continuous values.	Predicting salary or temperature.
6. Decision Tree Regression	Uses tree structure to split data into decisions and outcomes.	Predicting customer behavior using age, income.
7. Random Forest Regression	Ensemble of multiple decision trees; final prediction is average of all trees.	Predicting sales, customer churn, etc.

2) Linear Regression

1. Definition

Linear Regression is a **supervised machine learning algorithm** used to **predict a continuous output variable** based on one or more input features.

It assumes a **linear relationship** between the input (independent variable) and output (dependent variable), meaning the output changes at a **constant rate** as the input changes.

2. Concept

Linear regression finds the **best-fit straight line** that represents the relationship between variables.
This line is then used to make predictions on new data.

- **Independent Variable (Input):** The factor that influences the output.
  Example: Hours studied
- **Dependent Variable (Output):** The factor we want to predict.
  Example: Exam score

📘 **Example:**
If students who study more hours tend to score higher marks, linear regression can model this relationship and predict exam scores based on study hours.

3. Equation of the Best-Fit Line

For **Simple Linear Regression (one independent variable)**:

$$y = mx + b$$

Where:

- **y** → Predicted value (Dependent variable)
- **x** → Input value (Independent variable)
- **m** → Slope of the line (change in y for a unit change in x)
- **b** → Intercept (value of y when x = 0)

The algorithm finds the best values of **m** and **b** such that the **error (difference between actual and predicted values)** is minimized.

4. Best-Fit Line

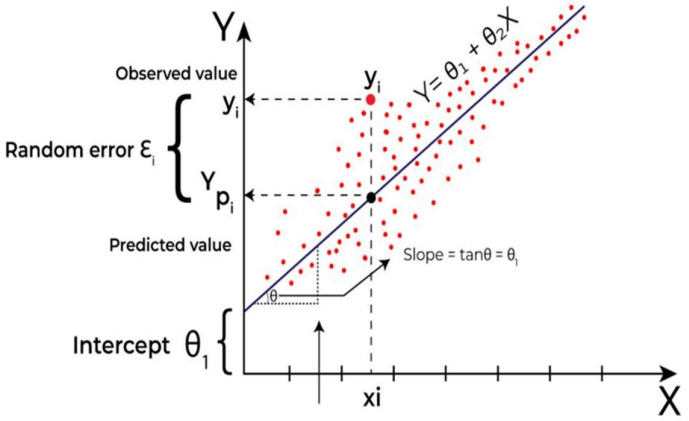
- The **best-fit line** is the straight line that most accurately represents the relationship between **X** and **Y**.
- It is determined using a mathematical approach called **Least Squares Method**, which minimizes the **sum of squared errors** between actual and predicted points.

Goal:
To minimize the prediction error and accurately estimate future values.

5. Advantages of Linear Regression

1. **Simplicity and Interpretability:**
   Easy to understand and implement; provides clear relationships between variables.
2. **Predictive Ability:**
   Can forecast future outcomes using past data.
3. **Efficiency:**
   Computationally fast and works well for linearly related problems.
4. **Analytical Insight:**
   Reveals how strongly input variables influence the target variable.

6. Graphical Representation



The line on the graph shows how the dependent variable (Y) changes with the independent variable (X).
The slope indicates the rate of change, and the intercept shows the starting value of Y when X = 0.

3) Linear Regression – Detailed Notes

---

1. Minimizing the Error: Least Squares Method

To find the **best-fit line**, Linear Regression uses the **Least Squares Method**.
This method minimizes the **sum of squared differences (residuals)** between the actual values and predicted values.

Residual Formula:

$$\text{Residual} = (Y_{\text{actual}} - Y_{\text{predicted}})$$

Objective (SSE – Sum of Squared Errors):

$$SSE = \sum(Y_{\text{actual}} - Y_{\text{predicted}})^2$$

The algorithm finds the line for which **SSE is minimum**, ensuring that the predictions are as close as possible to the actual data points.

---

2. Interpretation of the Best-Fit Line

$$y = mx + b$$

- **Slope (m):**
  Indicates how much *Y* changes for each 1-unit change in *X*.
  Example: If slope = 5, then for every 1-unit increase in *X*, *Y* increases by 5.
- **Intercept (b):**
  Value of *Y* when *X = 0* (point where the line crosses the Y-axis).

---

3. Assumptions of Linear Regression

No.	Assumption	Explanation
1	**Linearity**	Relationship between X and Y is a straight line.
2	**Independence of Errors**	Errors should not influence each other.
3	**Normality of Errors**	Errors follow a normal (bell-shaped) distribution.
4	**No Multicollinearity**	Independent variables are not highly correlated.
5	**Constant Variance (Homoscedasticity)**	Errors have equal spread across all input values. Unequal spread = Heteroscedasticity.
6	**No Autocorrelation**	Errors should not show repeating patterns (esp. in time-series).
7	**Additivity**	Combined effect on Y is the sum of individual effects of each X.

---

4. Multicollinearity in Regression

- Occurs when **two or more independent variables are highly correlated**.
- Violates the assumption of independence among predictors.
- Leads to unstable coefficient estimates and unreliable predictions.

---

5. Types of Linear Regression

Type	Description	Formula	Example
1. Simple Linear Regression	Uses one independent variable (X) to predict a dependent variable (Y).	$y = mx + b$	Predicting salary based on experience.
2. Multiple Linear Regression (MLR)	Uses two or more independent variables to predict one dependent variable.	$y = b_0 + b_1x_1 + b_2x_2 + \ldots + b_nx_n$	Predicting house price using area, rooms, and location.

---

6. Use Cases of Multiple Linear Regression

1. **Real Estate Pricing:** Predict property prices using size, location, and bedrooms.
2. **Financial Forecasting:** Predict stock prices using interest rate, inflation, and trends.
3. **Agricultural Yield:** Estimate crop yield using rainfall, soil, and fertilizer data.
4. **E-commerce Sales:** Study how price, marketing, and season affect sales.

---

7. Evaluation Metrics for Linear Regression

Used to measure how well the model predicts outcomes.

(i) Mean Squared Error (MSE)

$$MSE = \frac{1}{n}\sum(Y_{\text{actual}} - Y_{\text{predicted}})^2$$

- Measures average squared difference.
- **Lower MSE → better model**.
- Sensitive to outliers.

(ii) Mean Absolute Error (MAE)

$$MAE = \frac{1}{n}\sum | Y_{\text{actual}} - Y_{\text{predicted}} |$$

- Measures average absolute difference.
- **Less sensitive to outliers.**

(iii) Root Mean Squared Error (RMSE)

$$RMSE = \sqrt{MSE}$$

- Gives error in the same units as the output.
- Indicates how close predictions are to actual values.

(iv) Coefficient of Determination (R^2)

$$R^2 = 1 - \frac{RSS}{TSS}$$

Where:

- **RSS (Residual Sum of Squares):** $\sum(Y_{\text{actual}} - Y_{\text{predicted}})^2$
- **TSS (Total Sum of Squares):** $\sum(Y_{\text{actual}} - \bar{Y})^2$

Interpretation:

- R^2 ranges from **0 to 1**.
- Closer to **1 → better fit** (model explains more variance).

(v) Adjusted R^2

$$Adjusted \qquad R^2 = 1 - [\frac{(1 - R^2)(n - 1)}{n - p - 1}]$$

Where **p** = number of predictors.

- Adjusts R^2 by penalizing irrelevant predictors.
- Prevents overfitting.

---

8. Summary Table

Metric	Measures	Range	Ideal Value	Notes
MSE	Average squared error	≥ 0	Lower is better	Sensitive to outliers
MAE	Average absolute error	≥ 0	Lower is better	Robust to outliers
RMSE	Root of MSE	≥ 0	Lower is better	Same units as output
R^2	Variance explained	0–1	Closer to 1	Measures goodness of fit

Metric	Measures	Range	Ideal Value	Notes
Adjusted R^2	Adjusted variance explained	0–1	Closer to 1	Penalizes extra variables

4) Regularization Techniques for Linear Models

Regularization is a technique used to prevent **overfitting** in linear regression models by adding a **penalty term** to the cost function. It discourages the model from fitting noise in the training data and keeps the regression coefficients small.

---

1 Lasso Regression (L1 Regularization)

- **Concept:** Adds a penalty equal to the **absolute value** of the magnitude of coefficients.
- **Objective Function:**

$$\text{Minimize } \sum(y_i - \hat{y}_i)^2 + \lambda \sum | \theta_j |$$

- **Effect:**
  - Can shrink some coefficients to **zero**, effectively performing **feature selection**.
  - Helps identify the most important variables.
- **Use Case:** Useful when we suspect that only a few features are truly important.

---

2 Ridge Regression (L2 Regularization)

- **Concept:** Adds a penalty equal to the **square** of the magnitude of coefficients.
- **Objective Function:**

$$\text{Minimize } \sum(y_i - \hat{y}_i)^2 + \lambda \sum \theta_j^2$$

- **Effect:**
  - Reduces the impact of multicollinearity.
  - Coefficients are shrunk towards zero but **never exactly zero**.
- **Use Case:** Works well when all features are important but highly correlated.

---

3 Elastic Net Regression

- **Concept:** Combines both **L1 (Lasso)** and **L2 (Ridge)** penalties.
- **Objective Function:**

$$\text{Minimize } \sum(y_i - \hat{y}_i)^2 + \lambda[\alpha \sum \mid \theta_j \mid + (1-\alpha)\sum \theta_j^2]$$

- **Effect:**
  - Balances feature selection (L1) and coefficient shrinkage (L2).
  - Useful when there are **multiple correlated predictors**.

Advantages of Linear Regression

1. Simple and easy to interpret.
2. Computationally efficient for large datasets.
3. Provides insights into variable relationships.
4. Can serve as a **baseline model**.
5. Well-supported across libraries and tools.

Disadvantages of Linear Regression

1. Assumes a **linear** relationship between variables.
2. Sensitive to **multicollinearity**.
3. Requires careful **feature engineering**.
4. Can **overfit** or **underfit** data.
5. Limited ability to model **complex relationships**.

5) Polynomial Regression

Definition:

Polynomial Regression is an extension of **Linear Regression** that models the relationship between the independent variable (**x**) and the dependent variable (**y**) as an **nth-degree polynomial**.
It helps capture **non-linear patterns** in the data by fitting a **curved line** rather than a straight one.

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \ldots + \beta_n x^n + \epsilon$$

Need for Polynomial Regression

1. **Non-linear Relationships:**
   When data shows a curved trend, polynomial regression can model it better than a straight line.
2. **Better Fit for Curved Data:**
   Residuals (errors) from a linear model may show patterns if the true relationship is curved — polynomial terms fix this.
3. **Flexibility and Complexity:**
   By adding higher-degree terms, the model gains flexibility and can represent more complex relationships.

How Polynomial Regression Works

- Polynomial regression adds higher-degree powers of the independent variable to the regression model.
- The coefficients ($\beta_0, \beta_1, \ldots \beta_n$) are found using the **Least Squares Method**, minimizing the **sum of squared errors (SSE)** between predicted and actual values.
- The challenge is to choose the **right degree (n)**:
  - A **low degree** may **underfit** (too simple).
  - A **high degree** may **overfit** (too complex).

Example:

In **finance**, suppose we analyze the relationship between an employee's **years of experience (x)** and **salary (y)**.
A straight line (linear regression) may not fit well, but a **quadratic polynomial regression** can capture the curve more accurately.

$$y = \beta_0 + \beta_1 x + \beta_2 x^2$$

Here, $\beta_2 x^2$accounts for the curvature in the data, giving a better salary prediction model.

Choosing the Polynomial Degree

- **Degree 1:** Linear (straight line)
- **Degree 2:** Quadratic (U or inverted U shape)
- **Degree 3+:** More complex curves
  Selecting the right degree depends on data behavior and avoiding overfitting.

Key Differences Between Linear and Polynomial Regression

Aspect	Linear Regression	Polynomial Regression
Relationship	Models a **straight-line** relationship	Models a **curved/non-linear** relationship
Equation	$y = b_0 + b_1 x$	$y = b_0 + b_1 x + b_2 x^2 + \ldots + b_n x^n$
Flexibility	Limited	More flexible due to higher-degree terms
Fit Type	Straight line	Curved line
Overfitting Risk	Low	Higher if degree is too large

6) Decision Tree in Machine Learning

Definition:

A **Decision Tree** is a **supervised learning algorithm** used for both **classification** and **regression** tasks.
It works like a **flowchart** that makes decisions step by step.

Each component of a Decision Tree represents:

- **Root Node:** Represents the entire dataset and the first feature split.
- **Internal Nodes:** Represent attribute-based tests.
- **Branches:** Represent outcomes of the tests.
- **Leaf Nodes:** Represent the final output (class label or predicted value).

How a Decision Tree Works

A Decision Tree splits the dataset based on **feature values** to form **pure subsets**, i.e., groups containing data points of the same class.
Each split is selected to maximize the **information gained** or minimize **impurity**.

Example: Predicting if a customer will buy a product:

1. **Root Node (Income):**
   If income > ₹50,000 → go to next test, else → "No Purchase".
2. **Internal Node (Age):**
   If age > 30 → next test, else → "No Purchase".
3. **Internal Node (Previous Purchases):**
   If previous purchases > 0 → "Purchase", else → "No Purchase".

Key Terms

1. Information Gain

- Measures how well a feature separates data into target classes.
- Calculated using **Entropy**, which measures uncertainty.

$$Entropy(S) = -p_1 \log_2(p_1) - p_2 \log_2(p_2)$$
$$Information\ Gain = Entropy(parent) - \sum \frac{|S_v|}{|S|} Entropy(S_v)$$

- **High Information Gain** → better feature for splitting.

Example:
If splitting by "Age" divides the data into all "Yes" and all "No," then Information Gain is **maximum**.

---

2. Gini Index

- Measures **impurity** or **inequality** in a dataset.
- **Lower Gini Index = purer node.**

$$Gini = 1 - \sum (p_i)^2$$

Where p_i= probability of each class.

- **Gini = 0:** Pure node (only one class).
- **Gini = 0.5:** Maximum impurity (equal mix of classes).

Advantages of Gini Index:

- Simple and fast to compute.
- Works well for binary splits.

Steps to Build a Decision Tree

1. Start with all training data at the root node.
2. Calculate **Information Gain** or **Gini Index** for each attribute.
3. Choose the **best attribute** for splitting.
4. Recursively repeat the process for child nodes.
5. Stop when:
    - All samples belong to one class, or
    - No attributes remain.

Advantages

- Easy to interpret and visualize.
- Handles both **numerical** and **categorical** data.
- Requires little data preprocessing (no feature scaling).
- Works well for both classification and regression problems.

Disadvantages

- Prone to **overfitting** (can fit noise in data).
- Small changes in data can change the tree structure (instability).
- Biased towards attributes with more levels.
- Less effective for continuous variables without careful tuning.

7) eal-Life Use Case: Activity Prediction Based on Weather

Let's take an example where we predict **which outdoor activity to do** based on weather conditions.

Step-by-Step Working:

1. **Start with the Whole Dataset (Root Node):**
   - Begin with all available weather data (Outlook, Humidity, etc.).
   - This is the **root node** of the tree.
2. **Choose the Best Attribute (Question):**
   - The tree chooses the best feature to split the data using **Information Gain** or **Gini Index**.
   - Example: The first question is "What is the Outlook?"
     - Possible answers: **Sunny**, **Cloudy**, **Rainy**
3. **Split the Data into Subsets:**
   - Create branches for each condition:
     - Sunny → Subset 1
     - Cloudy → Subset 2
     - Rainy → Subset 3
4. **Recursive Splitting (Further Division):**
   - If a subset is still mixed, ask another question.
   - Example: For **Sunny**, ask "Is Humidity High or Normal?"
     - High → "Swimming"
     - Normal → "Hiking"
5. **Assign Final Decisions (Leaf Nodes):**
   - When a subset is pure (all samples have the same label), stop splitting.
   - Final outcomes:
     - Cloudy → Hiking
     - Rainy → Stay Inside
     - Sunny + High Humidity → Swimming

        - Sunny + Normal Humidity → Hiking
6. **Use the Tree for Prediction:**
    - To predict, follow the branches:
        - Example: Outlook = Sunny → Humidity = High → Result = **Swimming**

8) Decision Tree Pruning

Definition:

Pruning is a technique used to **simplify** a decision tree by removing unnecessary branches or nodes that do not contribute significantly to prediction accuracy.

The goal is to **reduce overfitting** and **improve generalization** on new data.

Types of Pruning:

1. Pre-Pruning (Early Stopping):

The tree growth is **stopped early** before it becomes too complex.

Common techniques:

- **Maximum Depth:** Limit the number of levels in the tree.
- **Minimum Samples per Leaf:** Set minimum data points in a leaf node.
- **Minimum Samples per Split:** Require a minimum number of samples to split a node.
- **Maximum Features:** Restrict number of features considered per split.

✅ **Goal:** Avoid overfitting by keeping the tree small and general.

---

2. Post-Pruning (Reduced Nodes):

The tree is **fully grown first**, and then unnecessary nodes are **removed** afterward.

Common techniques:

- **Cost-Complexity Pruning (CCP):** Balances accuracy and simplicity.
- **Reduced Error Pruning:** Removes branches that don't reduce accuracy.
- **Minimum Impurity Decrease:** Prunes nodes with low improvement in impurity.
- **Minimum Leaf Size:** Removes small leaf nodes.

✅ **Goal:** Simplify the model while maintaining or improving accuracy.

Importance of Pruning:

1. **Prevents Overfitting:**
   Removes branches that capture noise or outliers in the data.
2. **Improves Generalization:**
   Makes the model perform better on unseen data.
3. **Reduces Model Complexity:**
   Produces a simpler, faster, and more efficient tree.
4. **Enhances Interpretability:**
   A smaller tree is easier to read and explain.
5. **Speeds Up Training and Prediction:**
   Fewer nodes mean faster computation.
6. **Simplifies Maintenance:**
   Pruned trees are easier to update and adapt to new data.

9) Handling Missing Data in Decision Tree Models

Introduction:

Decision Trees are widely used for **classification and regression** tasks due to their **simplicity and interpretability**.
However, real-world datasets often contain **missing or incomplete data**, which can reduce model accuracy.
Decision trees handle such missing values intelligently through **imputation, weighting, and surrogate splits**, allowing them to remain effective even with incomplete information.

Types of Missing Data

1. **Missing Completely at Random (MCAR):**
   - Missing data occurs randomly without any pattern.
   - Example: Random sensor malfunction causes missing readings.
2. **Missing at Random (MAR):**
   - Missingness depends on **other observed variables**, not on the missing value itself.
   - Example: Income data missing more often for younger people (depends on age, not income).
3. **Missing Not at Random (MNAR):**
   - The missingness is related to the **missing value itself**.
   - Example: People with higher incomes not reporting their income.

How Decision Trees Handle Missing Data

Decision trees manage missing values **during both training and prediction** using three main mechanisms:
attribute splitting, weighted impurity calculation, and surrogate splits.

---

1. Attribute Splitting

- The algorithm selects the **best feature** to split the dataset using metrics like **Gini impurity** or **Information Gain**.
- If a data point has a **missing value** for the chosen feature:
  - The tree uses available data to decide **which branch** to follow.
  - This allows partial data to still contribute to model learning.

✅ **Result:** The model does not discard incomplete records, maintaining data efficiency.

---

2. Weighted Impurity Calculation

- When evaluating a potential split, decision trees **weigh the impurity** of each branch.
- If some data points are missing for the splitting feature:
  - The impurity (e.g., Gini or Entropy) is **weighted** according to how many data points have missing or available values.
  - The final split decision considers the **overall quality**, including missing data effects.

✅ **Result:** Missing data is accounted for in impurity measures, leading to more balanced splits.

---

3. Surrogate Splits

- A **surrogate split** is a **backup rule** used when the primary splitting feature has missing values.
- During training:
  - The decision tree identifies **alternative features** that closely mimic the behavior of the main split.
- During prediction:
  - If a data point is missing a value for the main feature, the **surrogate feature** is used instead.

✅ **Result:** Ensures smooth prediction even when data is incomplete.

Example: Flight Delay Prediction

Let's consider a model predicting **flight delays** using features such as **Time of Day**, **Weather**, and **Airline**.

1. **Optimal Feature Selection:**
   - The tree first splits based on the most informative feature, e.g., *Time of Day*.
2. **Impurity Calculation with Missing Data:**
   - When *Weather* data is missing for some flights, impurity is calculated **with weighted missing data**, ensuring fair consideration.
3. **Surrogate Splits Implementation:**
   - If *Weather* is missing, the model uses *Airline* as a **surrogate feature** to continue prediction.

✅ **Outcome:**
Even if some *Weather* data is missing, the model continues to make accurate predictions for flight delays.

Advantages of Handling Missing Data in Decision Trees

1. **Improves Model Robustness:** Works effectively even with incomplete data.
2. **Prevents Data Loss:** Avoids discarding missing records, using more data for training.
3. **Maintains Prediction Accuracy:** Surrogate splits ensure predictions remain valid.
4. **Enhances Real-world Applicability:** Suitable for real datasets where missing values are common.

10) Iterative Dichotomiser 3 (ID3) Algorithm

Introduction:

The **ID3 (Iterative Dichotomiser 3)** algorithm is a fundamental **decision tree learning algorithm** developed by **Ross Quinlan**.
It is a **supervised learning** technique used for **classification tasks**, which builds a tree by **selecting the best attribute** at each step using **Information Gain** based on **Entropy**.
The goal of ID3 is to create a **tree that classifies data with maximum accuracy** and minimal uncertainty.

How ID3 Works:

1. **Selecting the Best Attribute:**
   - ID3 uses **Entropy** and **Information Gain** to identify the attribute that best separates the dataset.
   - **Entropy** measures the impurity in the dataset.
   - The attribute with the **highest Information Gain** is chosen for splitting.

2. **Creating Tree Nodes:**
    - The selected attribute forms a **decision node**, and the data is divided into subsets based on its values.
    - For each subset, the algorithm **recursively** repeats the process to form new branches.
3. **Stopping Criteria:**
    - The recursion stops when:
        - All samples in a subset belong to the **same class**, or
        - No remaining attributes are left to split.
4. **Handling Missing Values:**
    - Missing data can be handled by substituting **mean/mode** or using the **majority class value**.
5. **Tree Pruning:**
    - Although not part of basic ID3, pruning (used in **C4.5**) is later applied to reduce **overfitting** and improve **generalization**.

Mathematical Concepts in ID3

1. Entropy (H)

Measures the **uncertainty** or **impurity** in a dataset.

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

Where:

- p_i= proportion of samples belonging to class i
- Lower entropy = more homogeneous data.

---

2. Information Gain (IG)

Measures how much an attribute **reduces entropy** or uncertainty.

$$IG(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \times Entropy(S_v)$$

- S_v: subset of data where attribute A = v
- The attribute with the **highest IG** is chosen for splitting.

---

3. Gain Ratio

Gain Ratio corrects Information Gain's bias towards multi-valued attributes.

$$Gain\ Ratio = \frac{Information\ Gain}{Split\ Information}$$

It normalizes IG by the number of distinct values, giving fairer attribute selection.

Advantages of ID3 Algorithm

1. **Simplicity & Interpretability:**
   - Easy to understand; each decision path explains the logic behind classification.
2. **Efficient for Categorical Data:**
   - Works well with discrete (categorical) features without preprocessing.
3. **Fast Greedy Approach:**
   - Builds the tree quickly by selecting attributes with the highest information gain.
4. **Foundation for Advanced Models:**
   - Serves as the basis for **C4.5** and **C5.0** algorithms.
5. **Versatile Applications:**
   - Used in **medical diagnosis**, **finance**, and **marketing segmentation**.

Limitations of ID3 Algorithm

1. **Overfitting:**
   - May create overly complex trees that perform poorly on unseen data.
2. **Handling Continuous Data:**
   - Requires discretization of numeric values, which may lead to information loss.
3. **Bias Toward Multi-Valued Attributes:**
   - Prefers features with many unique values, which may be irrelevant.
4. **No Pruning Mechanism:**
   - Basic ID3 lacks pruning, increasing risk of overfitting.
5. **Scalability Issues:**
   - Computationally expensive for large datasets as entropy and gain must be recalculated for every feature.

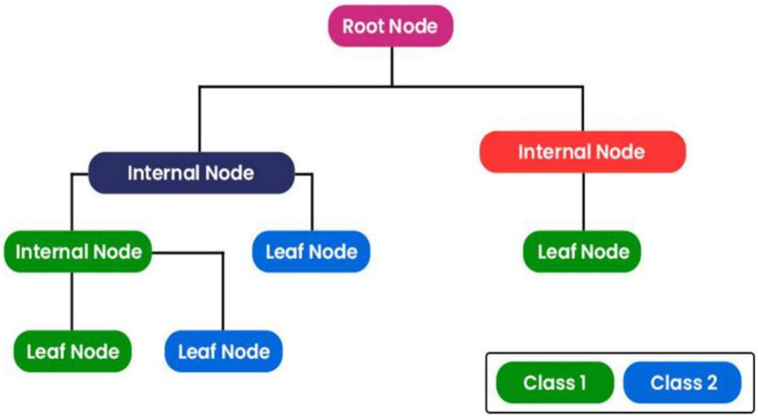
11) CART (Classification and Regression Tree) Algorithm

Definition:
CART (Classification and Regression Trees) is a **supervised machine learning algorithm** used for both **classification** (categorical target) and **regression** (continuous target) problems.

It builds **binary trees** using the concept of **Gini impurity** (for classification) and **residual reduction** (for regression).

How CART Works:

1. **Splitting:**
   - The data is split recursively based on the **best feature and threshold** that reduce impurity the most.
   - For **classification**, CART uses **Gini impurity**.
   - For **regression**, it minimizes the **residual sum of squares**.
2. **Stopping Criteria:**
   - Splitting continues until a condition is met (e.g., max depth reached or minimum samples per leaf).
3. **Pruning:**
   - To prevent **overfitting**, pruning removes branches that provide little predictive power (using cost-complexity pruning).



Mathematical Concept (Gini Impurity):

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

where p_i= probability of class i.

- Gini = 0 → pure node (all samples of one class)
- Gini = 1 → impure node (mixed classes)

Advantages:

- Handles **both classification & regression** tasks.
- **Non-parametric** — no assumptions on data distribution.
- Works with **categorical and numerical** data.
- **Interpretable** and easy to visualize.
- Automatically performs **feature selection** during splitting.

Limitations:

- **Overfitting** if tree is too deep (requires pruning).
- **Sensitive** to small data changes (instability).
- **Bias** toward features with many levels.
- Alone, it performs **worse than ensemble methods** like Random Forest.

Applications:

- **Finance:** Loan approval, credit scoring.
- **Healthcare:** Disease prediction and diagnosis.
- **Marketing:** Customer segmentation, churn prediction.
- **Education:** Student performance prediction.