# Module 3

# Classification and regression

## 📌 Introduction to Supervised Learning

Supervised learning involves training models on labeled data to predict outcomes for new, unseen data. The key objective is to minimize errors while ensuring generalization to new datasets, develop models that can generalize well to unseen data.

Key Learning Areas:

✔ **Classification Models** – Assign categorical labels to data points.

✔ **Regression Models** – Predict continuous numerical values.

✔ **Decision Trees and Regression Trees** – Rule-based models for decision-making.

✔ **Bias-Variance Tradeoff** – Understanding and balancing model complexity.

✔ **Ensemble Learning Techniques** – Combining multiple models for better performance.

In Module 3, we focus on:

* **Classification** → Assigning categorical labels to data points.
* **Regression** → Predicting continuous numerical values.
* **Decision Trees and Regression Trees** → Structured, rule-based learning models.
* **Understanding Tree-Based Models** → How they learn and make decisions.

## 📌 Classification: Categorizing Data into Groups

### 🔹 What is Classification?

Classification is a **supervised learning approach** where models learn to assign data points to **predefined categories** based on input features.

There are **two main types of classification problems**:

✔ **Binary Classification** → Two possible labels (e.g., spam vs. not spam).  
✔ **Multiclass Classification** → More than two categories (e.g., categorizing medical conditions).

Examples of classification tasks include:

✔ Identifying spam emails vs. non-spam emails.

✔ Detecting fraudulent transactions vs. legitimate transactions.

✔ Categorizing customer reviews as positive, neutral, or negative.

✔ Diagnosing diseases based on medical records.

### 🔹 Multiclass Classification Strategies

Since many classification models are **binary by default**, special strategies are needed for **multiclass classification** (where there are more than two labels).

1. **One-vs-All (OvA) Strategy**

* Also known as **One-vs-Rest (OvR)**.
* Trains **K binary classifiers**, where each classifier learns to distinguish **one class vs. all others**.
* The final class label is assigned based on **which classifier assigns the highest probability score**.

✔ **Simple and efficient**, especially for high-dimensional data.  
✔ Works well when **one class is dominant over others**.

1. **One-vs-One (OvO) Strategy**

* Trains **K(K-1)/2** binary classifiers, each distinguishing between **two classes at a time**.
* A **voting mechanism** determines the final class label, based on which class wins the most pairwise comparisons.

✔ **More robust when classes are similar**, leading to **better decision boundaries**.  
✔ **Computationally expensive**, especially when there are **many classes**.

✔ **Use OvA when the dataset is large and computing power is limited.**  
✔ **Use OvO when classification accuracy is a priority over computation speed.**

## 📌 Decision Trees: Rule-Based Learning

### 🔹 What Are Decision Trees?

A **Decision Tree** is a **structured, rule-based model** that makes sequential decisions by **splitting the dataset** into smaller, more homogeneous groups based on feature values.

It is one of the most **intuitive and interpretable** machine learning algorithms, widely used for both **classification** and **regression** tasks.

✔ **Classification Trees** → Used when the target variable is categorical.  
✔ **Regression Trees** → Used when the target variable is continuous.

### 🔹 How Decision Trees Work

A **Decision Tree** works by **iteratively dividing** the dataset at different feature values to **reduce uncertainty** in classification or prediction.

The structure of a **Decision Tree** consists of:

* **Root Node** → Represents the **entire dataset** and the **first decision point**.
* **Internal Nodes** → Represent **feature-based decision rules**.
* **Branches** → Show the possible **outcomes** of a decision.
* **Leaf Nodes** → Represent the **final classification (category label) or prediction (numerical value)**.

At each step, the model **selects the best feature** to split the dataset, and this process continues recursively **until a stopping criterion is met**.

### 🔹 Spliting Criteria for Decision Trees

**Why Do We Split the Data?**

The main idea behind **Decision Trees** is to repeatedly **split** the dataset in a way that **maximizes purity** in the resulting subsets.

To decide **how to split** the dataset at each node, **Decision Trees use different splitting criteria**:

1. **Entropy & Information Gain**

✔ **Entropy** measures the amount of uncertainty (or disorder) in a dataset. A **pure node** has **entropy = 0**, meaning all instances belong to the same class.  
✔ **Information Gain (IG)** measures how much **entropy decreases** after a split.  
✔ The feature with the **highest Information Gain** is chosen as the **best split**.

✔ **Low entropy** → The data is mostly one class (better split).  
✔ **High entropy** → The data is mixed, making classification harder.

✔ **Information Gain Formula**:

​ is the proportion of instances in each resulting subset.

1. **Gini Impurity**

✔ Measures the **probability** of misclassification at a node.  
✔ **Gini Impurity = 0** when the node contains a **single class** (perfect purity).  
✔ The goal is to **minimize Gini Impurity** when splitting data.

✔ **Gini is computationally faster than entropy**, making it more efficient for large datasets.  
✔ **Entropy is more useful for imbalanced data**, as it considers information content.

**Choosing Between Gini and Entropy**

* **Use Gini** when **computational speed is a priority**.
* Use **Entropy** when working with **imbalanced** class distributions.

### 🔹 Growing a Decision Tree: Recursive Partitioning

The **recursive partitioning** (also called **top-down induction**) process follows these steps:

* 1. **Start with the full dataset.**
  2. **Select the best feature** to split on (based on Entropy, Gini, or other criteria).
  3. **Divide the dataset** into two or more subsets.
  4. **Repeat the process recursively** for each subset until a stopping criterion is met:
* The tree reaches **maximum depth**.
* A node has **too few samples** to split further.
* The data in a node belongs to **a single class (pure node)**.

At the end of this process, **leaf nodes** contain either:

✔ A **class label** (for classification).  
✔ A **predicted value** (for regression).

### 🔹 Preventing Overfitting: Pruning Decision Trees

Overfitting occurs when a Decision Tree **memorizes** training data instead of learning **generalizable patterns**.

✔ Overfitting happens when a tree is too deep, capturing noise instead of patterns.  
✔ Underfitting happens when a tree is too shallow and lacks enough splits to capture relationships in the data.

To **prevent overfitting**, we use **pruning techniques**:

1. **Pre-Pruning (Early Stopping)**

✔ Stops tree growth early by setting a maximum depth or minimum number of samples per node.  
✔ Limits the number of splits, preventing overly complex trees.

1. **Post-Pruning (Reduced Error Pruning)**

✔ The tree is **grown fully** and then pruned **after training**.  
✔ **Branches that do not improve accuracy** are removed.  
✔ Uses **cross-validation** to determine **which branches to prune**.

✔ **Pruning helps simplify trees**, making them more interpretable and improving performance on **new data**.

✔ **Pruning helps simplify trees**, making them more interpretable and improving performance on **new data**.

## 📌 Regression Trees: Predicting Continuous Values

### 🔹 What Are Regression Trees?

A **Regression Tree** is an extension of a **Decision Tree** used for predicting **continuous numerical values** instead of classifying categorical labels. It follows the same recursive partitioning strategy but evaluates **splitting criteria based on numerical error metrics** rather than classification purity.

✔ **Regression Trees break down a dataset into smaller and smaller subsets**, while at the same time, an associated decision tree is incrementally developed.  
✔ The **final predictions** are obtained by taking the **average of values** in the leaf nodes rather than majority voting (as in classification trees).

Regression Trees are **widely used** for:

✔ **Sales forecasting** (e.g., predicting product demand).  
✔ **Real estate pricing** (e.g., estimating house prices based on location and features).  
✔ **Medical prognosis** (e.g., predicting hospital stay durations based on patient data).

### 🔹 How Regression Trees Work

The **main goal** of a Regression Tree is to split the data in a way that **minimizes prediction error**. Unlike classification trees, which maximize class purity using **Entropy or Gini Impurity**, Regression Trees evaluate splits using **variance reduction methods**.

**Step-by-Step Process**

1. **Start with the entire dataset** as the root node.
2. **Identify the best feature and split point** that minimizes prediction error.

When selecting split points, feature values are sorted, for continuous features, potential splits are **midpoints between consecutive unique values.**

1. **Recursively partition the data** into smaller subsets until a stopping criterion is met:

* The tree reaches **maximum depth**.
* The number of data points in a node **falls below a threshold**.
* Further splits **do not improve prediction accuracy significantly**.

1. **Make predictions at the leaf nodes** → Each leaf node contains the **average value of the target variable**.

At the end of this process, a Regression Tree produces a **piecewise constant function**, meaning the dataset is split into **intervals**, and the model predicts a **constant value** (the mean of training samples) for each interval.

### 🔹 Spliting Criteria for Regression Trees

Since Regression Trees predict **continuous values**, they use different **split evaluation criteria** compared to Classification Trees.

1. **Mean Squared Error (MSE) Reduction**

✔ The most **common** method for deciding where to split the data in a Regression Tree.  
✔ **MSE measures the variance of the target values within a node**. The lower the MSE, the **better the split**.

✔ **Formula for MSE**:

* -> Actual target values in the node.
* -> Mean of the target values in the node.
* -> Number of samples in the node.

✔ **At each node, the feature and split value that result in the lowest weighted MSE is chosen**.

1. **Mean Absolute Error (MAE) Reduction**

**✔ Similar to MSE but less sensitive to outliers.  
✔ Uses the absolute difference instead of squared errors.  
✔ Formula for MAE:**

**✔ MAE is useful when data contains extreme outliers, as it does not penalize them as heavily as MSE.**

1. **Variance Reduction**

**✔ The goal of Regression Trees is to reduce variance in the dataset.  
✔ Variance measures how spread out the target values are in a given node.  
✔ The split that results in the greatest variance reduction is selected.**

**✔ Formula for Variance Reduction:**

Where is the mean of the target values in the node.

✔ Lower variance = **better homogeneity** in target values within a node = **better prediction accuracy**.

## 📌Comparing Decision Trees - Classification vs. Regression

|  |  |  |
| --- | --- | --- |
| Feature | Classification Trees | Regression Trees |
| Prediction Output | Discrete Class Labels (e.g., "Dog" or "Cat") | Continuous Numeric Values (e.g., price, temperature) |
| Splitting Criteria | Entropy, Gini Impurity | Mean Squared Error (MSE), Mean Absolute Error (MAE) |
| Use Cases | Email Spam Detection, Disease Diagnosis | Sales Prediction, Stock Price Forecasting |
| Final Decision at Leaf Node | Email Spam Detection, Disease Diagnosis | Average Value of Data Points |

✔ **Classification Trees assign categories**, while **Regression Trees predict continuous values**.  
✔ **Both trees use recursive splitting**, but **their criteria differ** based on whether the target is **categorical or numerical**.