**🧠 What is Decision Tree Regression?**

**Decision Tree Regression** is a type of **Supervised Learning** that predicts a **continuous target variable** by **splitting the data into smaller regions** based on feature values.

Think of it like a flowchart: the model asks a series of questions about the features and navigates down the tree to reach a predicted value.

Unlike linear or polynomial regression, it **does not assume a linear relationship** between features and output.

**⚙️ How It Works**

1. The algorithm starts at the **root node** (all data).
2. It **splits the data** based on a feature that **reduces the error the most** (e.g., variance or MSE).
3. This process repeats recursively, creating **branches** and **leaf nodes**.
4. **Leaf nodes** contain the **predicted output** (average of target values in that region).

**💡 Example**

Predict house price based on **Size** and **Bedrooms**:

| **Size (sqft)** | **Bedrooms** | **Price (₹ lakhs)** |
| --- | --- | --- |
| 1000 | 2 | 50 |
| 1500 | 3 | 75 |
| 2000 | 4 | 100 |
| 2500 | 4 | 120 |
| 3000 | 5 | 150 |

* Root node: All houses
* Split 1: Size ≤ 2000 → Leaf node predicts average price
* Split 2: Size > 2000 → Leaf node predicts average price

Each leaf gives the predicted house price for that segment.

**📊 Visualization Concept**

[Size <= 2000?]

/ \

Yes No

/ \

Avg Price=75 Avg Price=140

For multiple features, the tree splits on **best feature** at each node.

**⚙️ Python Example**

from sklearn.tree import DecisionTreeRegressor

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error

# Example data

X = [[1000], [1500], [2000], [2500], [3000]] # Size

y = [50, 75, 100, 120, 150] # Price

# Split data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train model

model = DecisionTreeRegressor(random\_state=42)

model.fit(X\_train, y\_train)

# Predict

y\_pred = model.predict(X\_test)

# Evaluate

print("Predictions:", y\_pred)

print("Mean Squared Error:", mean\_squared\_error(y\_test, y\_pred))

**📏 Advantages**

✅ Can handle **non-linear relationships**  
✅ Does **not require feature scaling**  
✅ Easy to **visualize and interpret**  
✅ Can handle both **numerical and categorical features**

**⚠️ Limitations**

❌ Can **overfit** easily if tree is too deep  
❌ Sensitive to **small variations in data**  
❌ Less stable (small change in data → big change in tree)  
❌ Cannot extrapolate beyond training data

**🌍 Real-World Applications**

| **Domain** | **Use Case** |
| --- | --- |
| Real Estate | Predict house prices based on size, bedrooms, location |
| Finance | Estimate credit risk score |
| Retail | Predict product sales based on historical trends |
| Healthcare | Predict patient recovery time |
| Manufacturing | Predict equipment failure time |

**🧩 Quick Summary Table**

| **Feature** | **Decision Tree Regression** |
| --- | --- |
| **Goal** | Predict continuous value using a tree structure |
| **Relationship** | Can handle non-linear patterns |
| **Algorithm Type** | Supervised Learning (Regression) |
| **How It Works** | Recursive splitting of features into branches and leaves |
| **Evaluation Metrics** | MSE, RMSE, R² |
| **Advantages** | Non-linear, interpretable, no feature scaling required |
| **Limitations** | Overfitting, instability, cannot extrapolate |