**🧠 What is Random Forest Regression?**

**Random Forest Regression** is an **ensemble learning method** that builds **multiple decision trees** and combines their predictions to produce a **more accurate and stable output**.

Think of it as a “forest” of decision trees — instead of relying on a single tree, the model **averages predictions** from many trees.

It solves the **overfitting and instability problems** of a single decision tree.

**⚙️ How It Works**

1. **Bootstrap Sampling:** Randomly select subsets of the training data for each tree.
2. **Feature Randomness:** At each split in a tree, only a random subset of features is considered.
3. **Train Trees:** Each decision tree learns independently.
4. **Aggregate Predictions:** For regression, predictions from all trees are **averaged** to get the final output.

This reduces **variance** and improves **accuracy**.

**💡 Example**

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

| **Size (sqft)** | **Bedrooms** | **Age** | **Price (₹ lakhs)** |
| --- | --- | --- | --- |
| 1000 | 2 | 5 | 50 |
| 1500 | 3 | 10 | 75 |
| 2000 | 4 | 2 | 100 |
| 2500 | 4 | 15 | 120 |
| 3000 | 5 | 8 | 150 |

* Each tree sees a **different random subset of data**
* Each tree may focus on **different features**
* Final price prediction = **average of all tree predictions**

**📊 Visualization Concept**

[Random Forest]

/ | \

Tree1 Tree2 Tree3

| | |

Predict Predict Predict

\ | /

Final Prediction (Average)

**⚙️ Python Example**

from sklearn.ensemble import RandomForestRegressor

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

from sklearn.metrics import mean\_squared\_error

# Example data

X = [[1000,2,5],[1500,3,10],[2000,4,2],[2500,4,15],[3000,5,8]] # Features: Size, Bedrooms, Age

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 Random Forest model

model = RandomForestRegressor(n\_estimators=100, 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**

✅ Reduces **overfitting** compared to single decision trees  
✅ Can handle **non-linear relationships**  
✅ Works well with **numerical and categorical features**  
✅ More **stable and accurate predictions**

**⚠️ Limitations**

❌ Harder to interpret than a single decision tree  
❌ Requires more **computational power**  
❌ Can be slower for **large datasets**

**🌍 Real-World Applications**

| **Domain** | **Use Case** |
| --- | --- |
| Real Estate | Predict house prices using multiple features |
| Finance | Stock price prediction, credit risk assessment |
| Retail | Forecast product sales, demand prediction |
| Healthcare | Predict patient recovery time or medical costs |
| Manufacturing | Predict machinery maintenance or failure |

**🧩 Quick Summary Table**

| **Feature** | **Random Forest Regression** |
| --- | --- |
| **Goal** | Predict continuous value using ensemble of trees |
| **Relationship** | Can handle non-linear patterns |
| **Algorithm Type** | Supervised Learning (Regression) |
| **How It Works** | Aggregate predictions from multiple decision trees |
| **Evaluation Metrics** | MSE, RMSE, R² |
| **Advantages** | Accurate, stable, reduces overfitting |
| **Limitations** | Harder to interpret, computationally intensive |