Random Forest Regression

What is Random Forest Regression?

Random Forest Regression is a **supervised machine learning algorithm** used to predict **continuous numerical values**.

Key idea:

- It is an **ensemble of decision trees**.
- Each tree makes a prediction on the input data.
- The **final prediction** is the **average of all tree predictions** (for regression).

Why use Random Forest instead of a single tree?

- Single trees can **overfit** (fit too closely to training data).
- Random Forest reduces overfitting by averaging predictions of many trees.
- Works well with **non-linear data**.

How Random Forest Works

Step 1: Bootstrap Sampling

- Randomly select samples **with replacement** from the original dataset to train each tree.
- Some samples may repeat, and some may be left out.
- Each tree gets a slightly different dataset → this adds **diversity**.

Step 2: Feature Randomness

- At each split of a tree, choose a **random subset of features** instead of all features.
- This further reduces correlation between trees.

Step 3: Build Decision Trees

- Each tree is a **decision tree regressor**:
    - It splits the data into leaves based on feature thresholds.
    - Each leaf contains some training samples.
    - The **prediction of a leaf** = average of target values in that leaf.

Step 4: Make Predictions

- Input goes down **each tree** → lands in a leaf → tree predicts a value.
- **Random Forest prediction** = **average of all tree predictions**.

Manual Example: Predict House Price

Dataset (House Size → Price)

House Size (x)	Price (y in $1000)
1	150
2	200
3	250
4	300
5	350

We'll use **3 trees** for simplicity.

Step 1: Bootstrap Samples for Each Tree

Tree 1 sample: 2, 3, 3, 5, 1 → Prices: 200, 250, 250, 350, 150

Tree 2 sample: 1, 4, 5, 2, 2 → Prices: 150, 300, 350, 200, 200

Tree 3 sample: 3, 4, 5, 3, 1 → Prices: 250, 300, 350, 250, 150

Step 2: Build Trees

We'll use **simple splitting rules** for manual calculation:

Tree 1

- Split: $x \leq 3$ → left leaf, $x > 3$ → right leaf
- **Left leaf values:** x=1,2,3,3 → Prices = 150, 200, 250, 250 → Average = 212.5
- **Right leaf values:** x=5 → Price = 350

Prediction rule:

- $x \leq 3$ → 212.5

- $x > 3 \rightarrow 350$

Tree 2

- Split: $x \leq 2 \rightarrow$ left leaf, $x > 2 \rightarrow$ right leaf
- **Left leaf:** x=1,2,2 → Prices = 150, 200, 200 → Average = 183.33
- **Right leaf:** x=4,5 → Prices = 300, 350 → Average = 325

Prediction rule:

- $x \leq 2 \rightarrow 183.33$
- $x > 2 \rightarrow 325$

Tree 3

- Split: $x \leq 3 \rightarrow$ left leaf, $x > 3 \rightarrow$ right leaf
- **Left leaf:** x=3,3,1 → Prices = 250, 250, 150 → Average = 216.67
- **Right leaf:** x=4,5 → Prices = 300, 350 → Average = 325

Prediction rule:

- $x \leq 3 \rightarrow 216.67$
- $x > 3 \rightarrow 325$

Step 3: Make Prediction for x = 3.5

Check each tree:

- **Tree 1:** x=3.5 → $x > 3$ → Right leaf → predicts **350**
- **Tree 2:** x=3.5 → $x > 2$ → Right leaf → predicts **325**
- **Tree 3:** x=3.5 → $x > 3$ → Right leaf → predicts **325**

Step 4: Aggregate Predictions

Random Forest Regression averages all tree predictions:

$$= (350 + 325 + 325) / 3$$

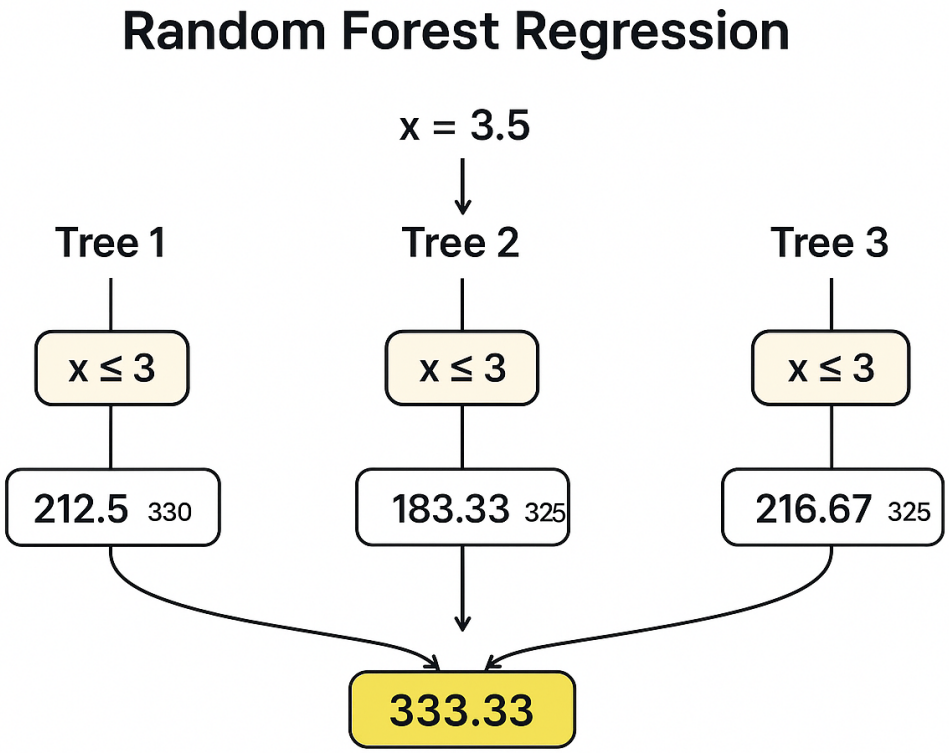
= 1000 / 3

≈ 333.33

So the **predicted price for house size 3.5** = **$333,330**

Step 5: Observations

- Each tree sees **slightly different data** → predictions vary.
- Averaging predictions → reduces variance and overfitting.
- Random Forest can **model non-linear relationships**, unlike linear regression.



Python Implementation

```
from sklearn.ensemble

import RandomForestRegressor

import numpy as np


# Dataset

X = np.array([[1],[2],[3],[4],[5]])

y = np.array([150,200,250,300,350])


# Random Forest Regressor

rf = RandomForestRegressor(n_estimators=3, random_state=42)

rf.fit(X, y)


# Predict house size 3.5

prediction = rf.predict([[3.5]])

print("Predicted Price:", prediction[0])
```

Output:

```
Predicted Price: 333.33
```