XGBoost Regressor

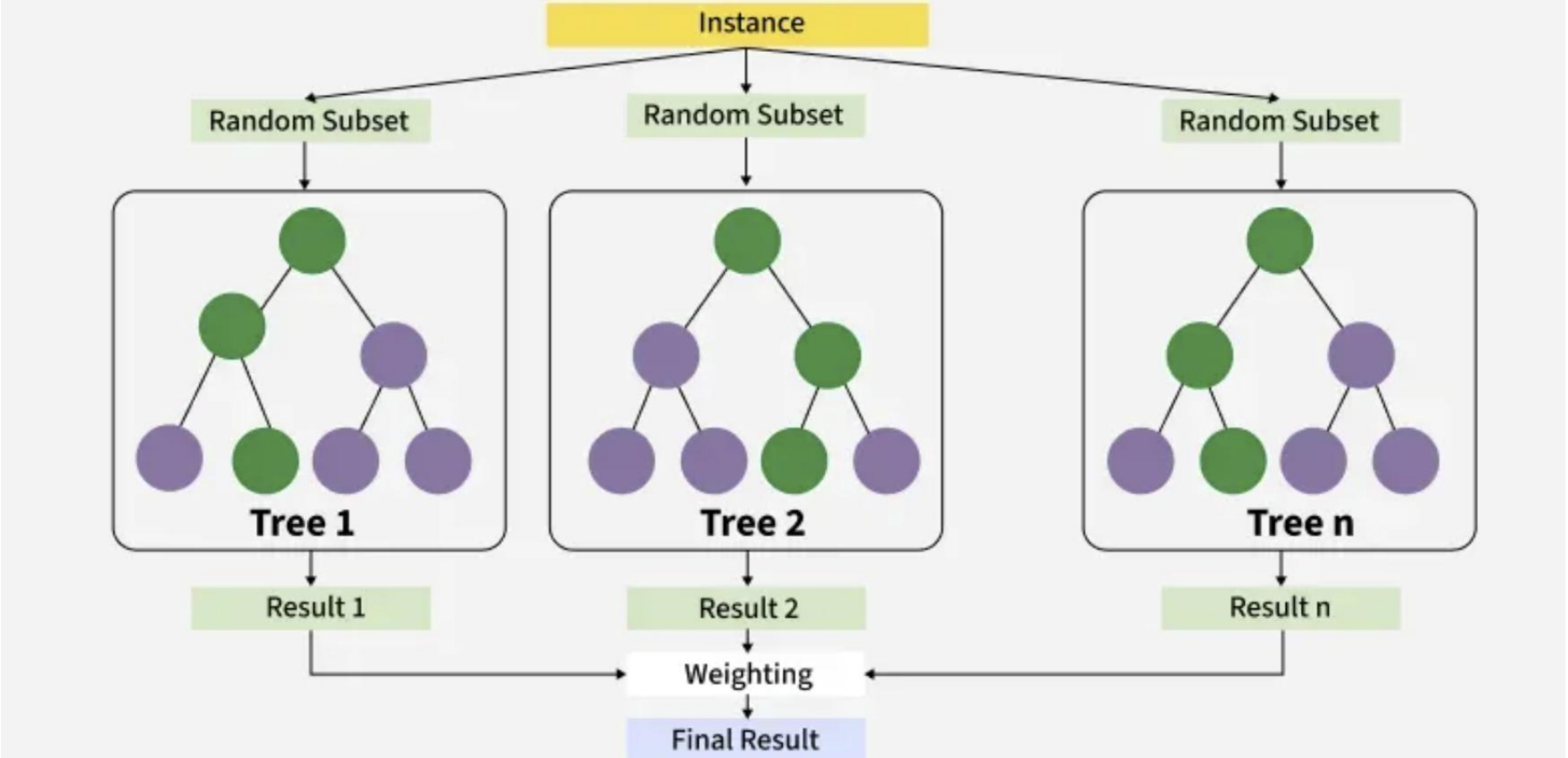
🚀 What is XGBoost Regression?

- **XGBoost (Extreme Gradient Boosting)** is a powerful **machine learning algorithm** based on decision trees.
- It works by combining many small “weak” trees to build a **strong prediction model**.
- It is widely used because it is **fast, accurate, and works well on large datasets**.

Why XGBoost is Popular?

✅ Very fast (optimized for speed)
✅ Handles missing values automatically
✅ Prevents overfitting (regularization built-in)
✅ Works for **Regression & Classification**

XGBoost Regression



How It Works (Step-by-Step)

1.Start with a simple prediction (like predicting the average value for all houses).

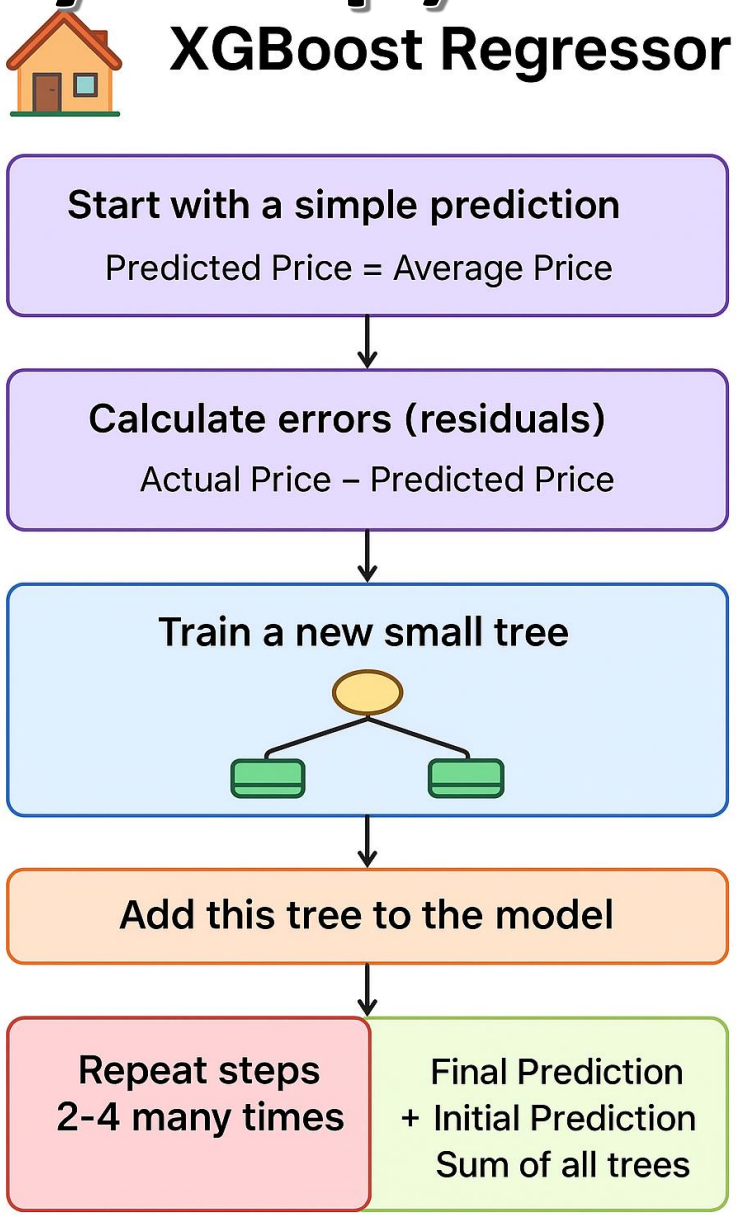
2.Calculate errors (residuals) = Actual value – Predicted value.

3.Train a new small tree to fix those errors.

4.Add this tree to the model (with some weight).

5.Repeat steps 2–4 many times, each new tree fixing the mistakes of the previous ones.

6.Finally, combine all trees → gives a strong regressor.



Example: Predicting House Prices

Example: Predicting House Prices

Suppose we want to predict house price from **size (sqft), location, and number of rooms**.

- Step 1: Model starts with an average prediction (e.g., ₹50 Lakhs).
- Step 2: Some houses are actually higher/lower → errors are calculated.
- Step 3: A new tree learns these errors (like: if size > 2000 sqft, add ₹20 Lakhs).
- Step 4: More trees are added, each fixing remaining mistakes.
- Step 5: After many rounds, the model gives very accurate house price predictions

Python

```
import xgboost as xg

model = xgb.XGBRegressor(
    n_estimators=100,     # number of boosting rounds
    learning_rate=0.1,    # step size shrinkage
    max_depth=3,          # depth of trees
    random_state=42
)
```

Parameters:

.

n_estimators ***int, default=50***

The maximum number of estimators at which boosting is terminated. In case of perfect fit, the learning procedure is stopped early. Values mst be in the range [1, inf).

random_state ***int, RandomState instance or None, default=None***

Controls the random seed given at each estimator at each boosting iteration. Thus, it is only used when estimator exposes a random_state. In addition, it controls the bootstrap of the weights used to train the estimator at each boosting iteration. Pass an int for reproducible output across multiple function calls. See Glossary.