LightGBM Regression

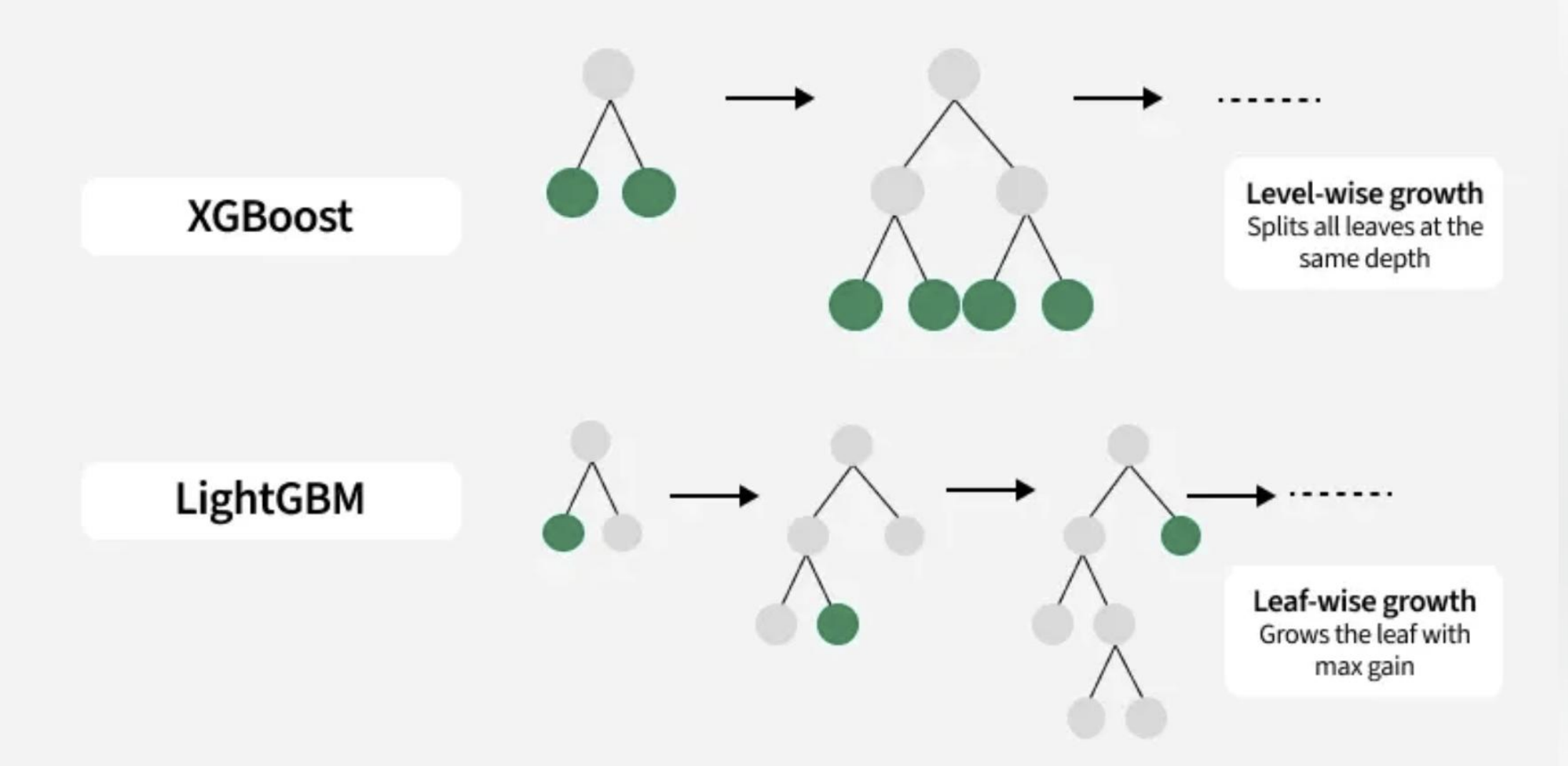
What is LightGBM?

- **LightGBM** = **Light Gradient Boosting Machine**
- It is a **boosting algorithm** (like XGBoost, AdaBoost) but **faster and more memory-efficient**.
- Specially designed for **large datasets** and **high-performance ML tasks**.

⚡ **Why is LightGBM Special?**

- **Leaf-wise growth** = deeper, more accurate trees (but can overfit small data).
- **Handles large data super fast** 🚀
- Supports **categorical features directly** (no need for one-hot encoding).
- Often **beats XGBoost** in speed and sometimes accuracy.

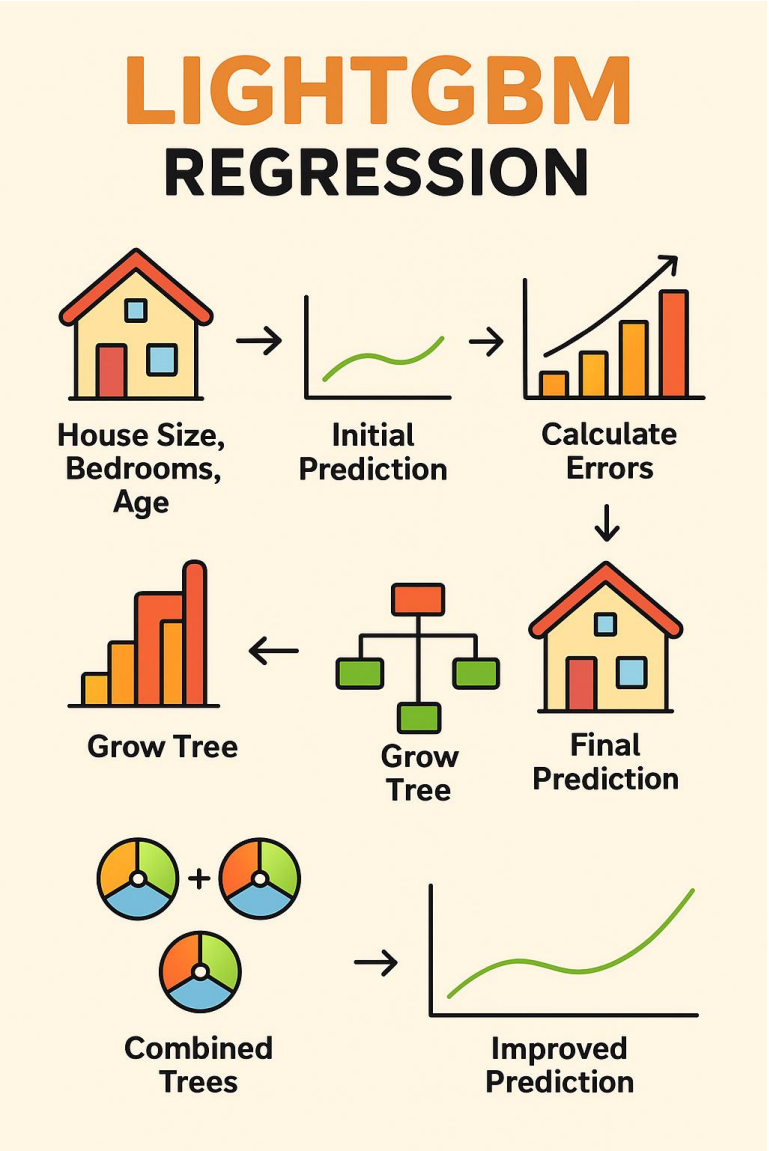
LightGBM Regression



How It Works (Step-by-Step)

🔑 **How it Works (in regression)**

1. **Start with a simple prediction** (like the average of all house prices).
2. **Calculate errors** (how far predictions are from actual).
3. **Grow small decision trees** that try to fix those errors.
    1. LightGBM grows **leaf-wise trees** (unlike XGBoost, which grows level-wise).
    2. This means it focuses on the branch with the **largest error** first.
4. **Add trees together** in a boosting way, so each new tree improves the overall model.
5. Final prediction = **sum of all trees** → much better accuracy!



Example: Predicting House Prices

🏠 **Example (House Price Prediction)**

Let’s say we want to predict **house price** using:

•Size (sqft)

•Bedrooms

•Age of house

👉 LightGBM will:

•Start with an average price (say ₹3,00,000).

•First tree corrects big mistakes (like small houses being overpriced).

•Next trees focus on remaining mistakes (like very new houses being underpriced).

•After 100+ rounds, you get a **highly accurate prediction** curve

Python

```python
import lightgbm as lgb
model = lgb.LGBMRegressor(
    n_estimators=100,     # number of boosting rounds
    learning_rate=0.1,    # step size shrinkage
    max_depth=-1,         # no limit by default
    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.