# Gradient Boosting

Gradient Boosting is a **supervised learning** algorithm used for **classification and regression**. It is an **ensemble technique** that builds multiple weak learners (usually decision trees) in a sequential manner, where each new tree **corrects the errors** of the previous ones using gradient descent.

---

## 🔍 How Gradient Boosting Works

1. **Train an initial weak learner** (usually a decision tree).
2. **Calculate residuals (errors)** between actual and predicted values.
3. **Fit a new weak learner** to predict the residuals.
4. **Update the model** by adding this new learner to minimize error.
5. **Repeat** steps 2–4 for a fixed number of iterations or until errors are minimized.

## 🔥 Popular Variants of Gradient Boosting

1. **XGBoost (Extreme Gradient Boosting)** → Faster and more regularized version.
2. **LightGBM (Light Gradient Boosting Machine)** → Optimized for large datasets.
3. **CatBoost (Categorical Boosting)** → Designed for handling categorical features efficiently.

---

## 🎯 When to Use Gradient Boosting?

✅ When you need **high accuracy** and are okay with longer training times.

✅ When working with **structured/tabular data** (like Kaggle competitions).

✅ When **feature importance** is useful for interpretability.

Day-8 Gradient Boosting

Gradient boosting is a type of ensemble supervised machine learning algorithm that combines multiple weak learners to create a final model. It can be used for both classification and regression problems.

Gradient Boosting:

idea: • Predictor $f_k$ at stage $k$ incurs loss $L(f_k(x), y)$

• Train $h_{k+1}$ to approximate negative gradient:

$$h_{k+1}(x) \approx -\frac{\partial L(f_k(x), y)}{\partial f_k(x)}$$

• Update predictor by adding a multiple $\eta_{k+1}$ of $h_{k+1}$:

$$f_{k+1}(x) \leftarrow f_k(x) + \eta_{k+1} h_{k+1}(x)$$

⇒ • Consider squared loss

$$L(f_k(x_n), y_n) = \frac{1}{2}(f_k(x_n) - y_n)^2$$

• Negative gradient corresponds to residual $r$

$$-\frac{\partial L(f_k(x_n), y_n)}{\partial f_k(x_n)} = y_n - f_k(x_n) = r_n$$

• Train base learner $h_{k+1}$ with residual dataset $\{(x_n, r_n)_{r_n}\}$

• Base learner $h_{k+1}$ can be any non-linear predictor (often a small decision tree)

⇒ Pseudo Code: GBA:

• Initialize predictor with a constant $c$:

$$f_0(x_n) = \arg\min_c \sum_n L(c, y_n)$$

• for $k = 1$ to $K$ do

→ compute pseudo residual $r_n = -\frac{\partial L(f_{k-1}(x_n), y_n)}{\partial f_{k-1}(x_n)}$

(residual: difference between prediction and target)

- Train a base learner $h_k$ with residual dataset $\{(x_n, r_n)\}$
- Optimize step length

$$\eta_k = \text{argmin}_\eta \sum_n L(f_k(x_n) + \eta h_k(x_n), y_n)$$

- Update predictor : $f_k(x) \leftarrow f_{k-1}(x) + \eta_k h_k(x)$