XGBoost Regressor

Machine Learning Algorithm for Regression Problems

1 Core Concept

XGBoost Regressor applies gradient boosting with sequential decision trees to solve **regression problems**. Unlike classification, it predicts continuous numerical values by iteratively reducing residuals.

Definition

XGBoost Regressor: An ensemble learning algorithm that builds sequential decision trees where each tree corrects the errors (residuals) of the previous trees, optimized using gradient descent.

2 Dataset Structure

Experience (Years)	Career Gap	Salary (K)
2	Yes	40
2	No	42
3	No	52
4	No	60
5	Yes	62

- **Input Features (X):** Experience, Career Gap
- **Output Feature (Y):** Salary (Continuous → Regression)

3 XGBoost Regressor: Step-by-Step Process

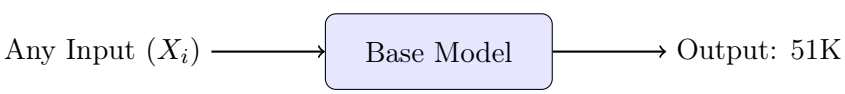
3.1 Step 1: Create Base Model

Important

For regression, the base model outputs the **average** of all target values.

$$\hat{y}_{\text{base}} = \frac{\sum_{i=1}^{n} y_i}{n} = \frac{40 + 42 + 52 + 60 + 62}{5} = \frac{256}{5} = \boxed{51.2 \approx 51\text{K}}$$

- Base model is **unbiased** — gives same prediction for any input
- Every record initially gets $\hat{y} = 51\text{K}$ as prediction



3.2 Step 2: Compute Residuals

Definition

Residual = Actual Value − Predicted Value

$$r_i = y_i - \hat{y}$$

Exp	Gap	Actual (y)	Predicted ($\hat{y}$)	Residual (r_1)
2	Yes	40	51	−11
2.5	No	42	51	−9
3	No	52	51	+1
4	No	60	51	+9
5	Yes	62	51	+11

3.3 Step 3: Construct Decision Tree

Build a decision tree using:

- **Input Features:** X_i (Experience, Gap)
- **Target:** r_1 (Residuals from Step 2)

Note

The decision tree is trained to predict **residuals**, not actual values. This is the key to gradient boosting!

3.4 Step 4: Calculate Similarity Weight

Important

Similarity Weight Formula for Regression:

$$\boxed{W = \frac{(\sum \text{Residuals})^2}{\text{Number of Residuals} + \lambda}}$$

Remember

Classification vs Regression Similarity Weight:
Classification:

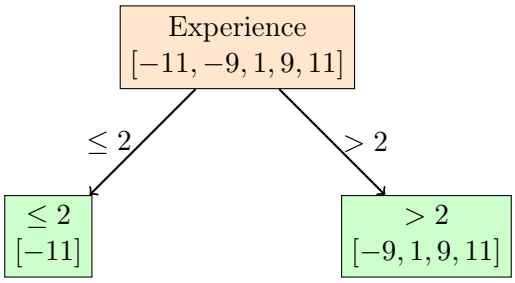
$$W = \frac{(\sum \text{Residuals})^2}{\sum P(1-P) + \lambda}$$

Regression:

$$W = \frac{(\sum \text{Residuals})^2}{n + \lambda}$$

where n = number of residuals in that node

3.4.1 Example: Split on Experience ≤ 2



Calculations (assuming $\lambda = 1$):

Left Child (≤ 2): Contains $[-11]$

$$W_{\text{left}} = \frac{(-11)^2}{1+1} = \frac{121}{2} = \boxed{60.5}$$

Right Child (> 2): Contains $[-9, 1, 9, 11]$

$$W_{\text{right}} = \frac{(-9+1+9+11)^2}{4+1} = \frac{(12)^2}{5} = \frac{144}{5} = \boxed{28.8}$$

Root Node: Contains all $[-11, -9, 1, 9, 11]$

$$W_{\text{root}} = \frac{(-11-9+1+9+11)^2}{5+1} = \frac{(1)^2}{6} = \frac{1}{6} \approx \boxed{0.167}$$

3.5 Step 5: Calculate Gain

Definition

Gain Formula:

$$\boxed{\text{Gain} = W_{\text{left}} + W_{\text{right}} - W_{\text{root}}}$$

Gain measures how much **improvement** a split provides.

For Split: Experience ≤ 2

$$\text{Gain} = 60.5 + 28.8 - 0.167 = \boxed{89.13}$$

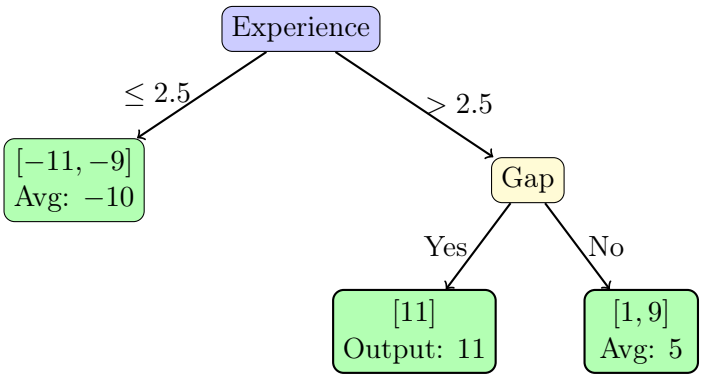
3.5.1 Comparing Different Splits

Split Threshold	Gain	Decision
Experience ≤ 2	89.13	—
Experience ≤ 2.5	143.42	✓**Selected (Higher Gain)**

Tip

Always select the split with the **highest gain**. Compare all possible thresholds for all features.

3.6 Step 6: Build Complete Decision Tree



Note

Leaf node output = **Average of residuals** in that node

4 Final Prediction Formula

Important

XGBoost Regressor Prediction:

$$\boxed{\hat{y}_{\text{final}} = \hat{y}_{\text{base}} + \alpha \cdot DT_1 + \alpha \cdot DT_2 + \ldots + \alpha \cdot DT_n}$$

where:

- $\hat{y}_{\text{base}}$ = Base model output (average)
- α = Learning rate (hyperparameter, e.g., 0.1)
- DT_i = Output of i^{th} decision tree

4.1 Prediction Examples

Example 1: Experience = 3, Gap = No

1. Base learner output: 51
2. Tree traversal: $3 > 2.5 \rightarrow$ Gap = No $\rightarrow$ Leaf: $[1, 9] \rightarrow$ Avg = 5
3. Final: $\hat{y} = 51 + 0.1 \times 5 = 51 + 0.5 = \boxed{51.5}$

Example 2: Experience = 2, Gap = Yes

1. Base learner output: 51
2. Tree traversal: $2 \leq 2.5 \rightarrow$ Leaf: $[-11, -9] \rightarrow$ Avg = -10
3. Final: $\hat{y} = 51 + 0.1 \times (-10) = 51 - 1 = \boxed{50}$

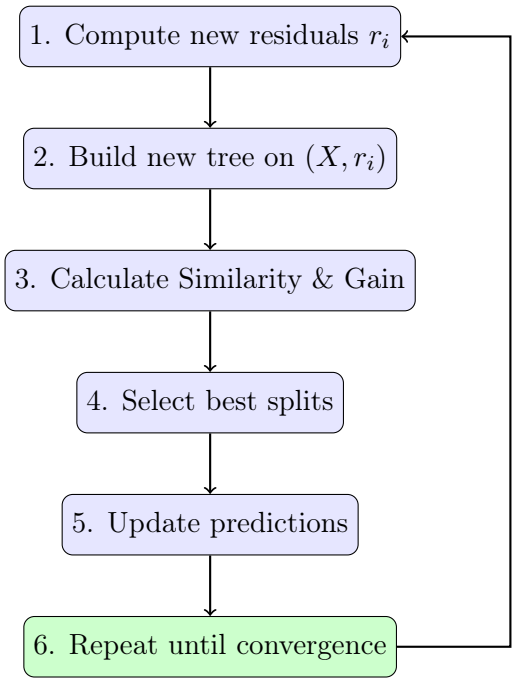
Example 3: Experience = 5, Gap = Yes

1. Base learner output: 51
2. Tree traversal: $5 > 2.5 \rightarrow$ Gap = Yes $\rightarrow$ Leaf: [11] $\rightarrow$ Output = 11
3. Final: $\hat{y} = 51 + 0.1 \times 11 = 51 + 1.1 = \boxed{52.1}$

4.2 Updated Predictions After Decision Tree 1

Exp	Gap	Actual	Old $\hat{y}$	New $\hat{y}$	New Residual (r_2)
2	Yes	40	51	50	-10
2	No	42	51	50	-8
3	No	52	51	51.5	$+0.5$
4	No	60	51	51.5	$+8.5$
5	Yes	62	51	52.1	$+9.9$

5 Iterative Tree Building



6 Role of Lambda (λ)

Important

Lambda (λ) is a **regularization hyperparameter** that prevents overfitting.

$$W = \frac{(\sum \text{Residuals})^2}{n + \lambda}$$

λ Value	Similarity Weight	Effect
Low ($\lambda \approx 0$)	Higher	More complex trees (risk of overfitting)
High ($\lambda >> 0$)	Lower	Simpler trees (more regularization)

Warning

λ must be tuned using **cross-validation**. Too high λ leads to underfitting, too low leads to overfitting.

7 XGBoost: Classifier vs Regressor

Aspect	XGBoost Classifier	XGBoost Regressor
Problem Type	Classification (categorical output)	Regression (continuous output)
Base Model	Log of odds: $\log\left(\frac{p}{1-p}\right)$	Average: $\frac{\sum y_i}{n}$
Similarity Weight	$\frac{(\sum r_i)^2}{\sum P(1-P)+\lambda}$	$\frac{(\sum r_i)^2}{n+\lambda}$
Leaf Output	Probability/Log-odds	Average of residuals
Final Prediction	Apply sigmoid for probability	Direct sum with learning rate

8 Key Hyperparameters

Parameter	Description	Typical Values
α (learning_rate)	Step size for each tree contribution	0.01 – 0.3
λ (reg_lambda)	L2 regularization term	0 – 10
n_estimators	Number of sequential trees	100 – 1000
max_depth	Maximum tree depth	3 – 10

9 Key Formulas Summary

Remember

1. **Base Model (Regression):**

$$\hat{y}_{\text{base}} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

2. **Residual:**

$$r_i = y_i - \hat{y}_i$$

3. **Similarity Weight (Regression):**

$$W = \frac{(\sum \text{Residuals})^2}{\text{Number of Residuals} + \lambda}$$

4. **Gain:**

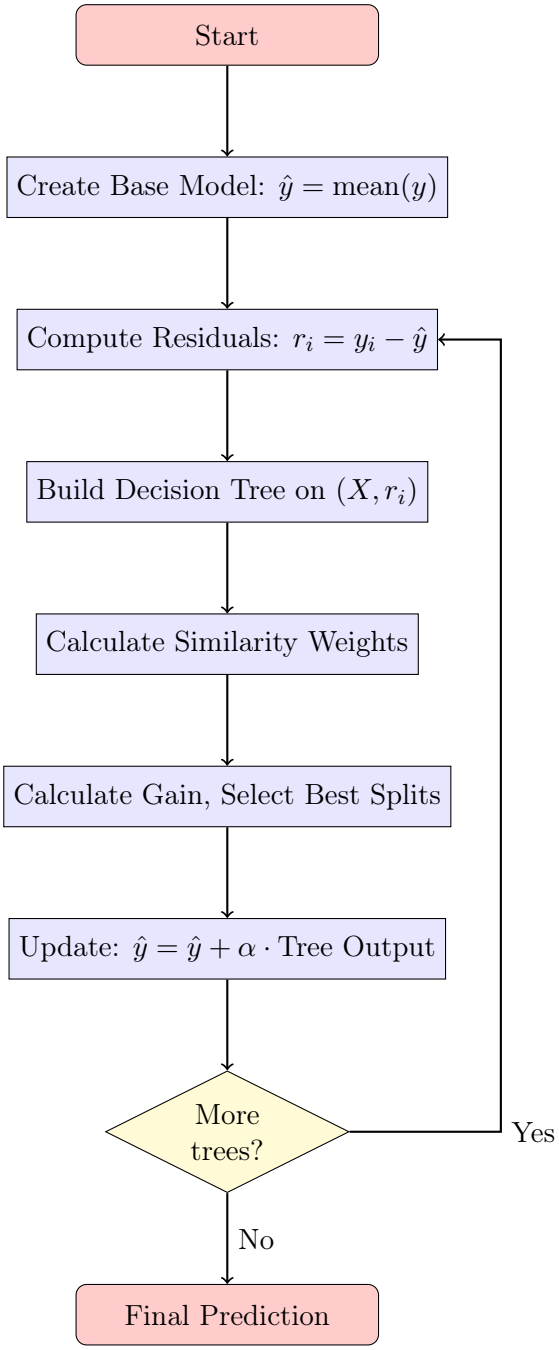
$$\text{Gain} = W_{\text{left}} + W_{\text{right}} - W_{\text{parent}}$$

5. **Final Prediction:**

$$\hat{y} = \hat{y}_{\text{base}} + \sum_{k=1}^{K} \alpha \cdot f_k(X)$$

where $f_k(X)$ = output of k^{th} tree

10 Complete Algorithm Flowchart



11 Common Mistakes

Warning

1. Confusing Formulas:

- Classification uses $\sum P(1-P)$ in denominator
- Regression uses n (count of residuals) in denominator

2. Forgetting Learning Rate:

- Tree output must be multiplied by α before adding
- Missing this leads to completely wrong predictions

3. Lambda Confusion:

- Higher λ = Lower similarity weight = More regularization
- Not the opposite!

12 Quick Revision Points

1. Base model = Average of target values
2. Residual = Actual − Predicted
3. Trees are trained on **residuals**, not actual values
4. Similarity Weight = $\frac{(\sum r)^2}{n+\lambda}$
5. Gain = $W_{\text{left}} + W_{\text{right}} - W_{\text{root}}$
6. Higher Gain $\Rightarrow$ Better Split
7. Final Prediction = Base + $\alpha(\text{Tree}_1)$ + $\alpha(\text{Tree}_2)$ + ...
8. λ controls regularization (prevents overfitting)
9. α (learning rate) controls step size
10. Sequential trees, each correcting previous errors