XGBoost Algorithm

eXtreme Gradient Boosting

MUKESH KUMAR

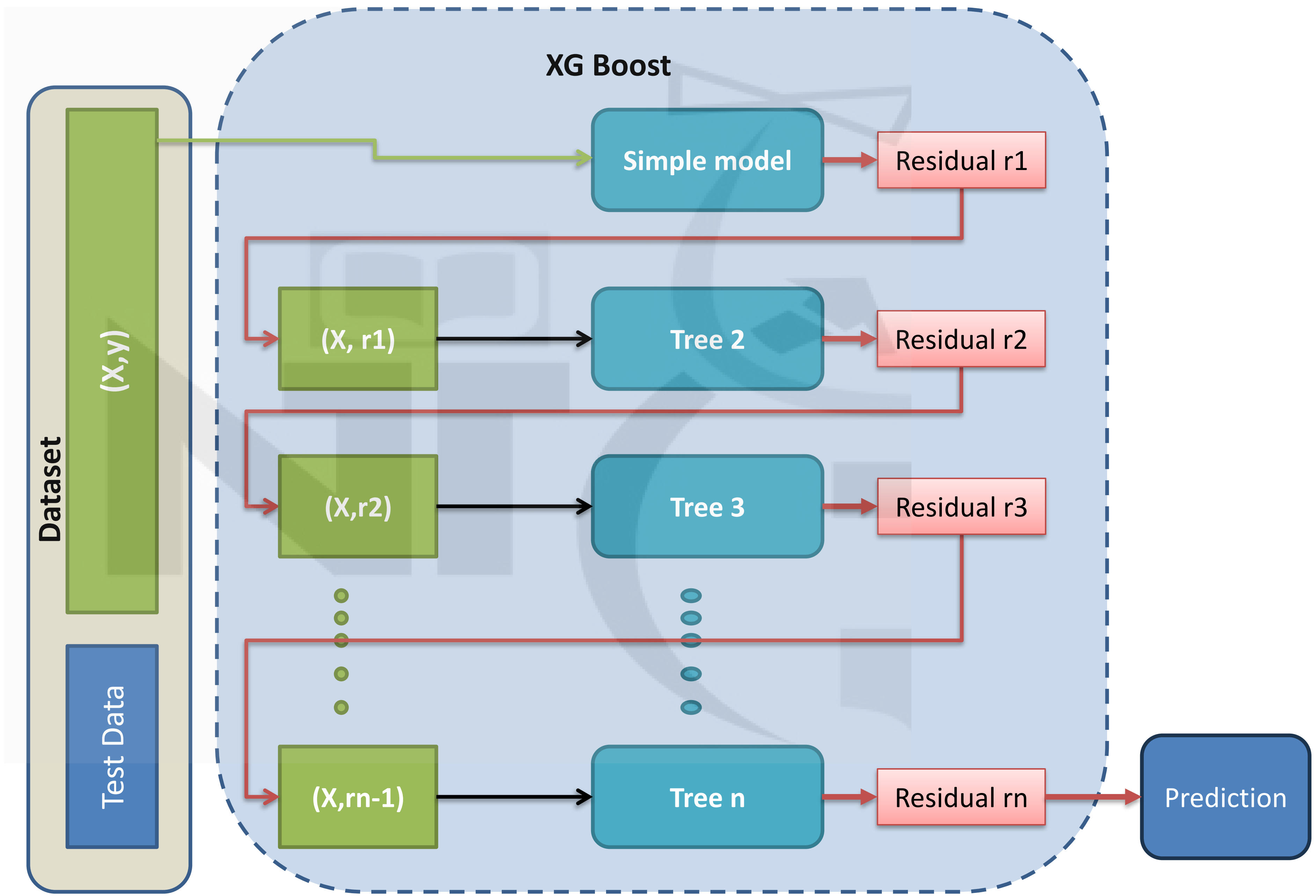
Introduction to XGBoost

- Developed by Tianqi Chen
- Fast, accurate, and widely used gradient boosting implementation
- Popular in data science competitions like Kaggle

- **XGBoost (Extreme Gradient Boosting)** is a highly efficient and scalable machine learning algorithm based on gradient boosting. It builds decision trees sequentially, with each tree aiming to correct the errors of the previous one

Why XGBoost?

- Regularization to reduce overfitting
- High performance and scalability
- Supports parallel and distributed computing
- Handles missing values automatically



XG Boost
Dataset
(X,y)
Test Data
Simple model
Residual r1
(X, r1)
Tree 2
Residual r2
(X,r2)
Tree 3
Residual r3
(X,rn-1)
Tree n
Residual rn
Prediction

For Regression:

$$F(x) = F_0(x) + \eta \cdot h_1(x) + \eta \cdot h_2(x) + \cdots + \eta \cdot h_M(x)$$

Where:

- $F_0(x)$ is the initial guess (like mean of targets),
- Each $h_m(x)$ is a weak learner trained on **residuals** (errors).

How XGBoost Works?

Basic Concepts Recap

- **Gradient Boosting:** Ensemble method that builds models sequentially
- Each new model corrects the errors of the previous one
- Final prediction = sum of all weak learners

XGBoost Regression Tree

Sample Data

housesize	houseprice
1200	270000
1500	310000
1800	330000
2000	420000
2200	470000

Let’s build Model 1

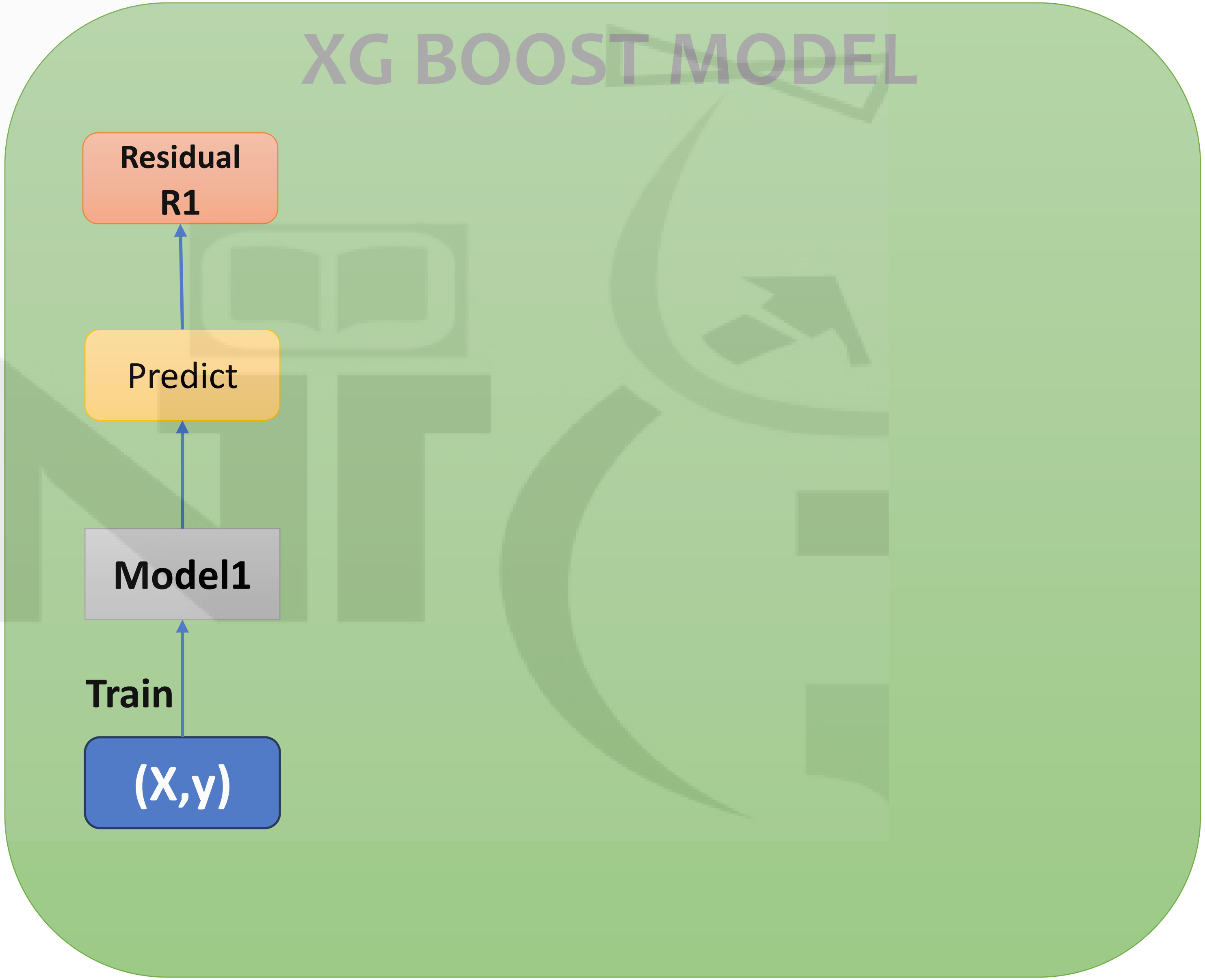
- We start with a mean like Gradient Boost

housesize	houseprice	Pred1
1200	270000	360000
1500	310000	360000
1800	330000	360000
2000	420000	360000
2200	470000	360000

Calculate Residual

- Residual formula is Actual minus predicted value

housesize	houseprice	Pred1	residual_1
1200	270000	360000	-90000
1500	310000	360000	-50000
1800	330000	360000	-30000
2000	420000	360000	60000
2200	470000	360000	110000


XG BOOST MODEL
Residual
R1
Predict
Model1
Train
(X,y)

Model 2 : Build a tree

- In XGBoost the way we build tree is different
- Gradient boost uses Gini index/info gain to find the best split
- XGBoost uses Greedy algorithm to find the best split

Similarity Score (Squared Error Loss)

- XGBoost uses similarity score to decide best split

Formula for Similarity Score

The similarity score in XGBoost for a given node is calculated using the formula:

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

Where:

- $\sum Residuals$ is the sum of all residual values in that node.
- Number of Residuals is the count of residual values in that node.
- λ (lambda) is a regularization parameter that helps to prevent overfitting. A common default value for λ is 1.

Similarity score for “Residual_1”

1. Calculate the sum of the residuals:

$$\sum Residuals = -90000 + (-50000) + (-30000) + 60000 + 110000 = 0$$

2. Count the number of residuals:

$$Number of Residuals = 5$$

3. Assume a default value for lambda:

   Let's assume the regularization parameter $\lambda = 1$.

4. Calculate the similarity score:

$$Similarity Score = \frac{(0)^2}{5+1} = \frac{0}{6} = 0$$

housesize	houseprice	Pred1	residual_1
1200	270000	360000	-90000
1500	310000	360000	-50000
1800	330000	360000	-30000
2000	420000	360000	60000
2200	470000	360000	110000

Find Midpoints of Subsequent 'housesize' Values:

Next, we calculate the midpoints between each subsequent pair of 'housesize' values. These midpoints will serve as our potential split points:

- Midpoint 1: $(1200 + 1500)/2 = 1350$
- Midpoint 2: $(1500 + 1800)/2 = 1650$
- Midpoint 3: $(1800 + 2000)/2 = 1900$
- Midpoint 4: $(2000 + 2200)/2 = 2100$

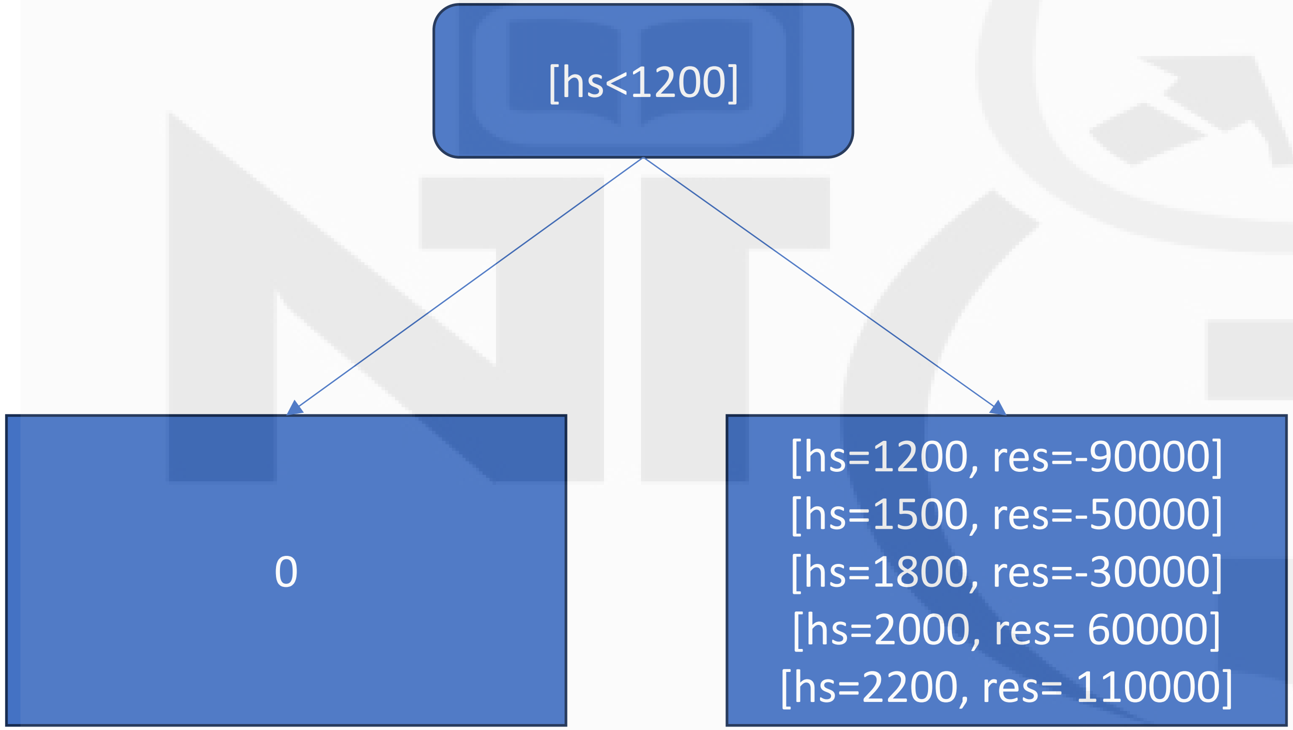
So, our potential split points for the 'housesize' feature are 1350, 1650, 1900, and 2100.

Create Splits and Calculate Similarity Score for Each Split:

- For each potential split point, we will divide the data into two nodes (left and right) based on whether the 'housesize' is less than or greater than the split point.

- Then, we calculate the similarity score for each of these resulting nodes. We'll use the same similarity score formula as before (assuming $\lambda=1$)

Potential Split 1: housesize < 1200



housesize	residual_1
1200	-90000
1500	-50000
1800	-30000
2000	60000
2200	110000

1350
1650
1900
2100

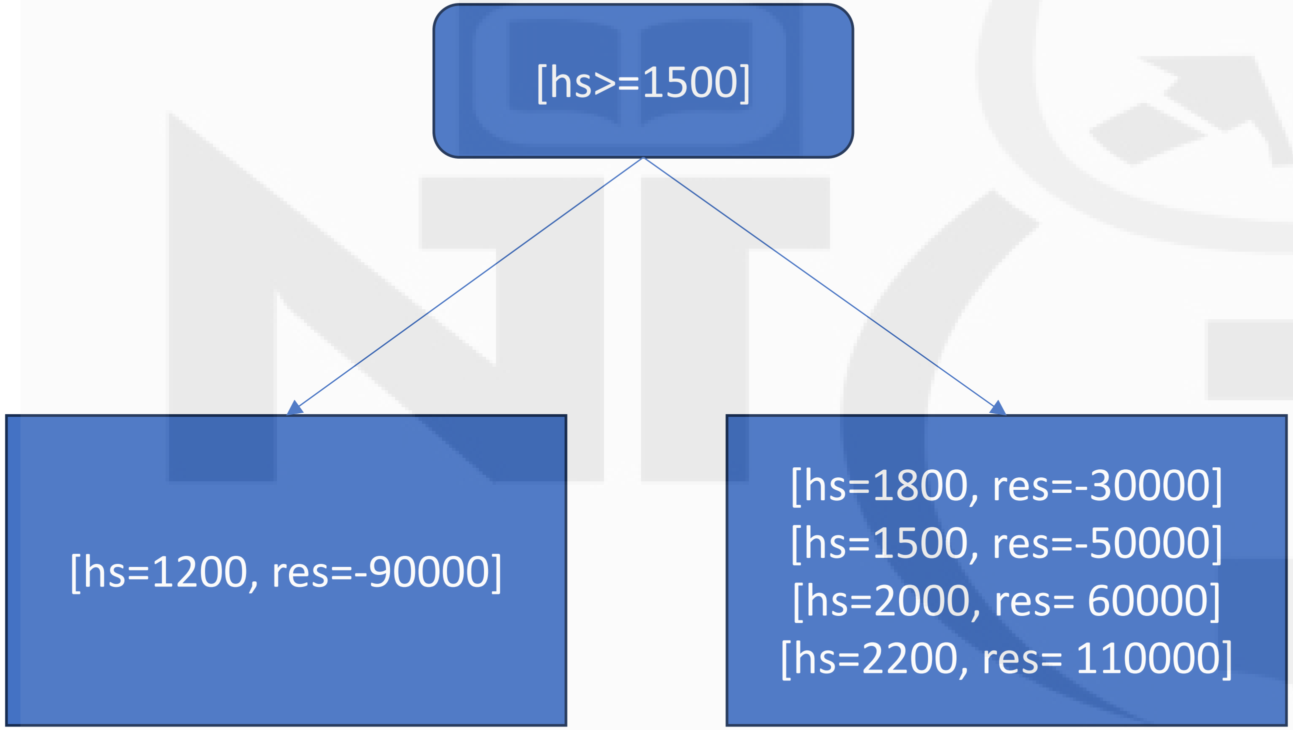
First split : similarity score

- **Left Node (housesize < 1200):** Empty (0 data points)
  - $\sum Residuals_{left} = 0$
  - $NumberofResiduals_{left} = 0$
  - $SimilarityScore_{left} = \frac{(0)^2}{0} = 0$ (We'll treat division by zero as 0 gain contribution)
- **Right Node (housesize $\geq$ 1200):** All data points
  - $\sum Residuals_{right} = 0$
  - $NumberofResiduals_{right} = 5$
  - $SimilarityScore_{right} = \frac{(0)^2}{5} = 0$
- **Gain at split < 1200:** $0 + 0 - 0 = 0$

Potential Split 2: housesi

e in XGBoost for a given node is calculated using the formul

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



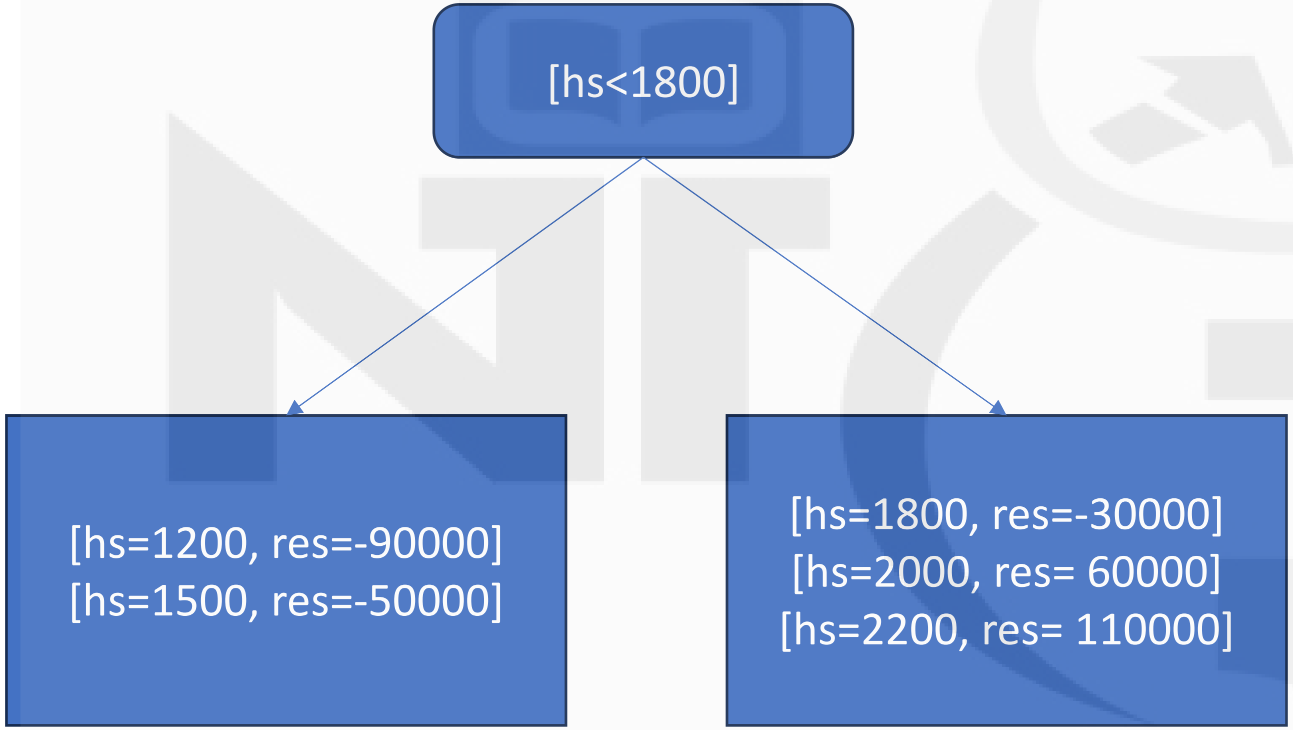
housesize	residual_1
1200	-90000
1500	-50000
1800	-30000
2000	60000
2200	110000

1350
1650
1900
2100

Potential Split 2: Similarity Score

- **Left Node (housesize < 1500):** [hs=1200, res=-90000]
  - $\sum Residuals_{left} = -90000$
  - $NumberofResiduals_{left} = 1$
  - $SimilarityScore_{left} = \frac{(-90000)^2}{1} = 8100000000$
- **Right Node (housesize $\geq$ 1500):** [hs=1500, res=-50000], [hs=1800, res=-30000], [hs=2000, res=60000], [hs=2200, res=110000]
  - $\sum Residuals_{right} = -50000 - 30000 + 60000 + 110000 = 90000$
  - $NumberofResiduals_{right} = 4$
  - $SimilarityScore_{right} = \frac{(90000)^2}{4} = 2025000000$
- **Gain at split < 1500:** $8100000000 + 2025000000 - 0 = 10125000000$

Potential Split 3: housesize < 1800



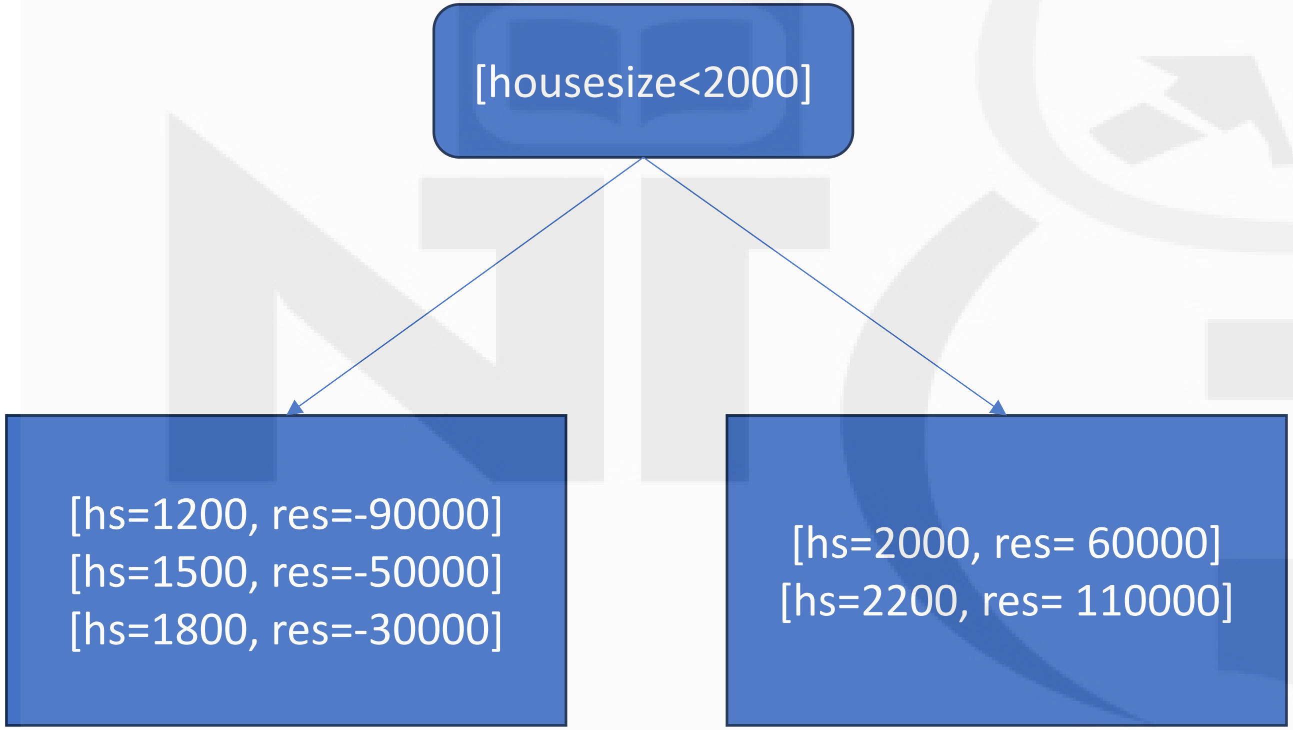
housesize	residual_1
1200	-90000
1500	-50000
1800	-30000
2000	60000
2200	110000

1350
1650
1900
2100

Potential Split 3: Similarity Score

- **Left Node (housesize < 1800):** [hs=1200, res=-90000], [hs=1500, res=-50000]
  - $\sum Residuals_{left} = -90000 - 50000 = -140000$
  - $NumberofResiduals_{left} = 2$
  - $SimilarityScore_{left} = \frac{(-140000)^2}{2} = 9800000000$
- **Right Node (housesize $\geq$ 1800):** [hs=1800, res=-30000], [hs=2000, res=60000], [hs=2200, res=110000]
  - $\sum Residuals_{right} = -30000 + 60000 + 110000 = 140000$
  - $NumberofResiduals_{right} = 3$
  - $SimilarityScore_{right} = \frac{(140000)^2}{3} \approx 6533333333.33$
- **Gain at split < 1800:** $9800000000 + 6533333333.33 - 0 = 16333333333.33$

Potential Split 4: housesize < 2000



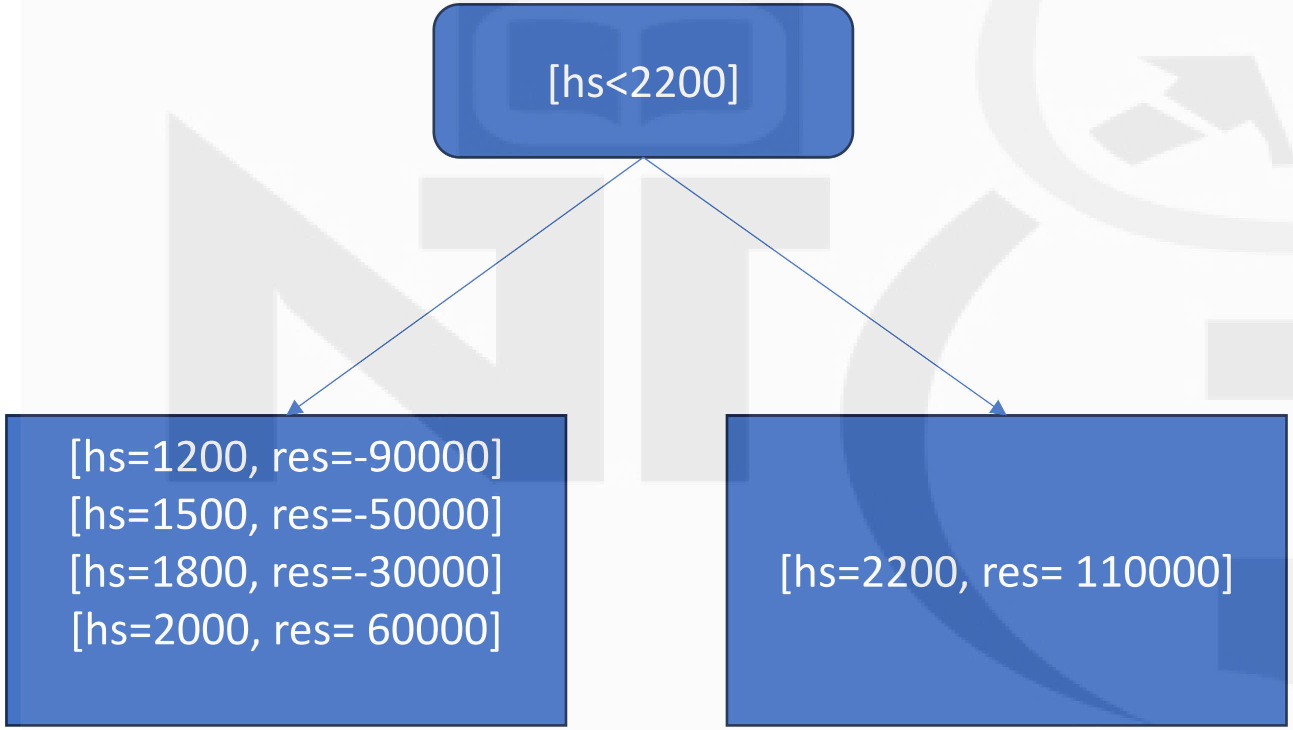
1350
1650
1900
2100

housesize	residual_1
1200	-90000
1500	-50000
1800	-30000
2000	60000
2200	110000

Potential Split 4: Similarity Score

- **Left Node (housesize < 2000):** [hs=1200, res=-90000], [hs=1500, res=-50000], [hs=1800, res=-30000]
  - $\sum Residuals_{left} = -90000 - 50000 - 30000 = -170000$
  - $NumberofResiduals_{left} = 3$
  - $SimilarityScore_{left} = \frac{(-170000)^2}{3} \approx 9633333333.33$
- **Right Node (housesize $\geq$ 2000):** [hs=2000, res=60000], [hs=2200, res=110000]
  - $\sum Residuals_{right} = 60000 + 110000 = 170000$
  - $NumberofResiduals_{right} = 2$
  - $SimilarityScore_{right} = \frac{(170000)^2}{2} = 14450000000$
- Gain at split < 2000: $9633333333.33 + 14450000000 - 0 = 24083333333.33$

Potential Split 5: housesize < 2200



housesize	residual_1
1200	-90000
1500	-50000
1800	-30000
2000	60000
2200	110000

1350
1650
1900
2100

Potential Split 5: Similarity score

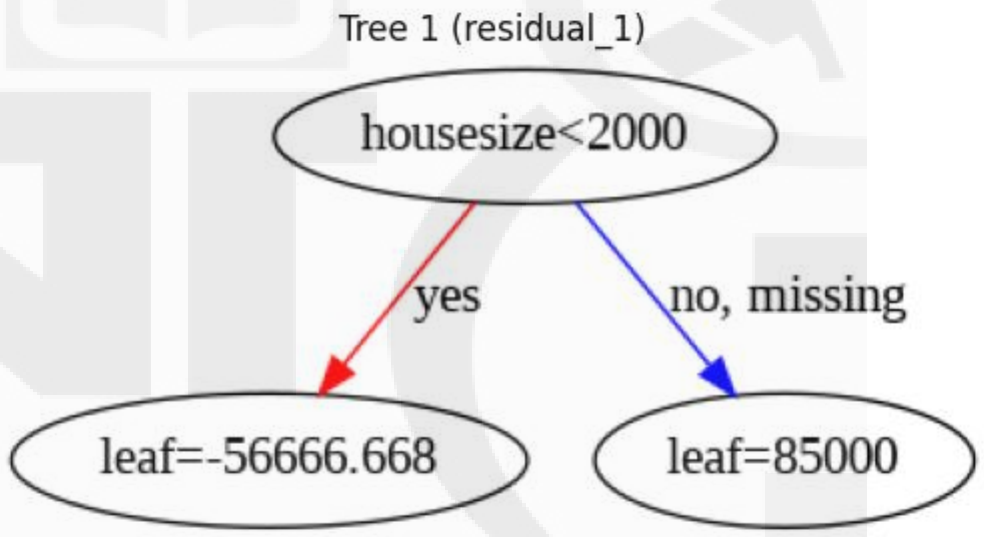
- **Left Node (housesize < 2200):** [hs=1200, res=-90000], [hs=1500, res=-50000], [hs=1800, res=-30000], [hs=2000, res=60000]
  - $\sum Residuals_{left} = -90000 - 50000 - 30000 + 60000 = -110000$
  - $NumberofResiduals_{left} = 4$
  - $SimilarityScore_{left} = \frac{(-110000)^2}{4} = 3025000000$
- **Right Node (housesize $\geq$ 2200):** [hs=2200, res=110000]
  - $\sum Residuals_{right} = 110000$
  - $NumberofResiduals_{right} = 1$
  - $SimilarityScore_{right} = \frac{(110000)^2}{1} = 12100000000$
- **Gain at split < 2200:** $3025000000 + 12100000000 - 0 = 15125000000$

Summary of Gains

Split Point (housesize <)	Gain (Change in Similarity Score)
1200	0
1500	10125000000
1800	16333333333.33
2000	24083333333.33
2200	15125000000

As we can see, the split at housesize < 2000 yields the highest gain.

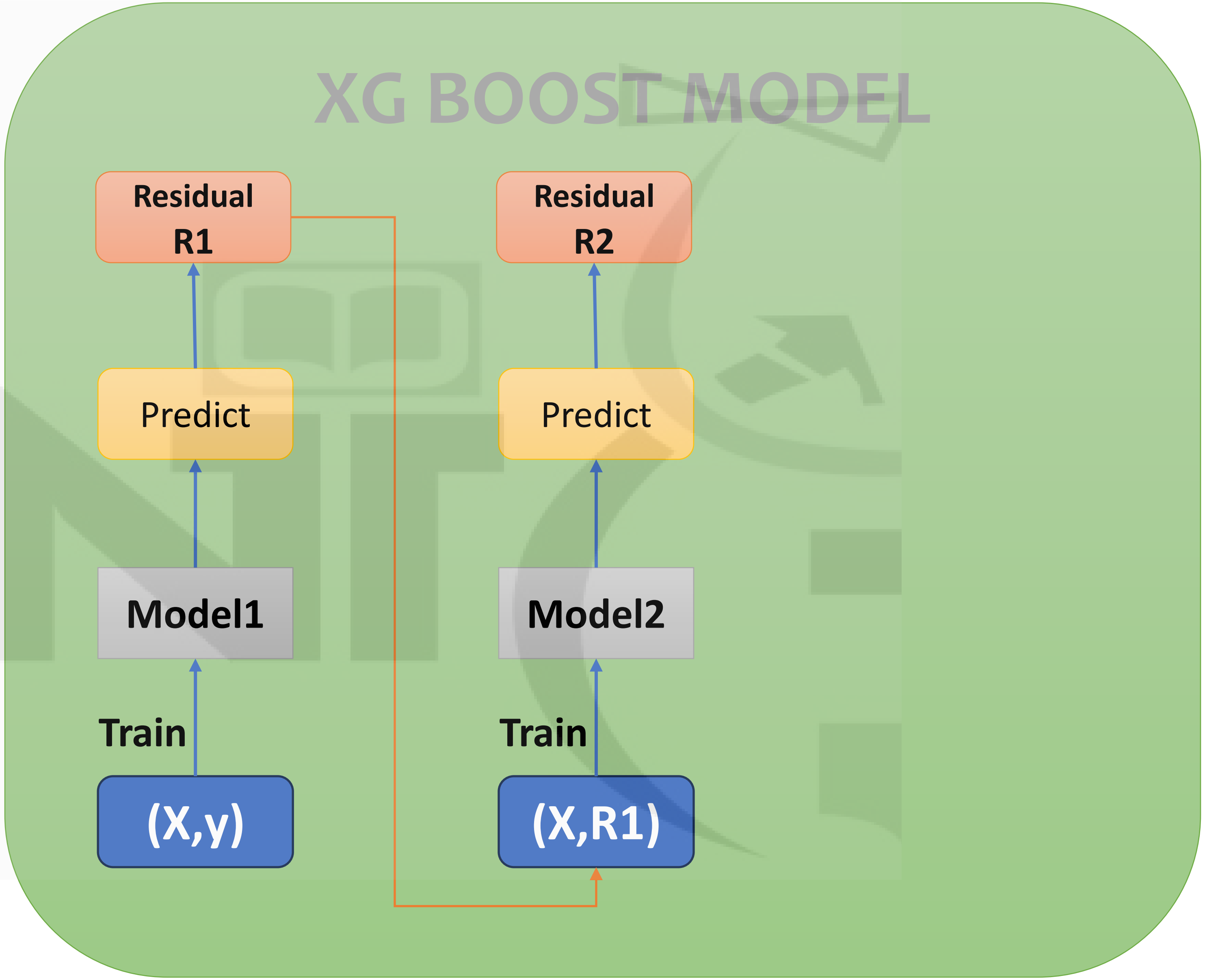
Model 2


Tree 1 (residual_1)
housesize<2000
yes
no, missing
leaf=-56666.668
leaf=85000

Find model 2 Predictions and Residuals

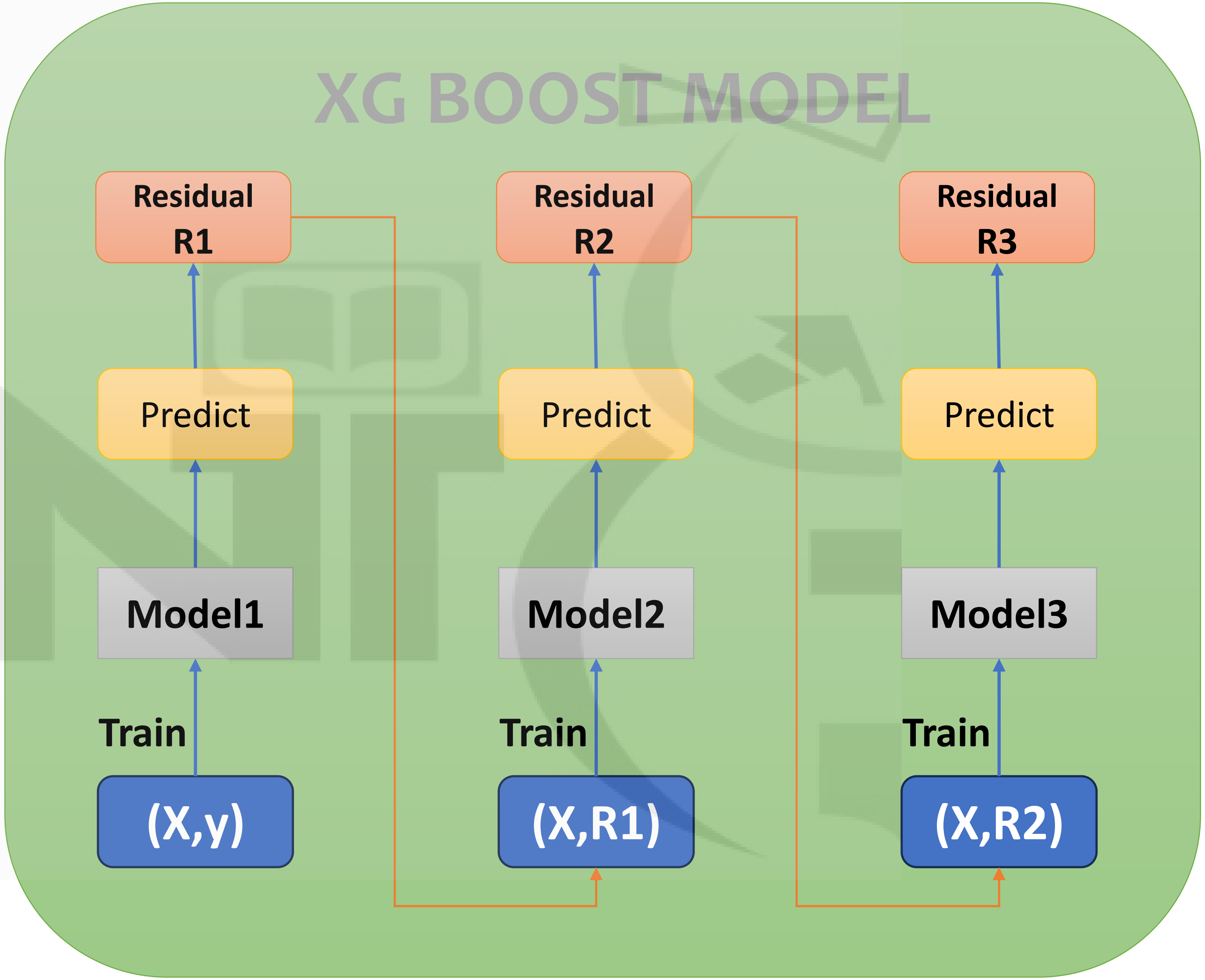
- Residuals = Actual - Predicted

housesize	houseprice	Pred1	residual_1	pred_2	Updated_Model2_pred	residual_2
1200	270000	360000	-90000	-56666.668	303333.332	-33333.332
1500	310000	360000	-50000	-56666.668	303333.332	6666.66797
1800	330000	360000	-30000	-56666.668	303333.332	26666.668
2000	420000	360000	60000	85000	445000	-25000
2200	470000	360000	110000	85000	445000	25000


XG BOOST MODEL
Residual
R1
Residual
R2
Predict
Predict
Model1
Model2
Train
Train
(X,y)
(X,R1)

Model 3

housesize	houseprice	Pred1	residual_1	pred_2	Updated_Model2_pred	residual_2	pred_3	final_pred
1200	270000	360000	-90000	-56666.668	303333.332	-33333.332	-33333.332	270000
1500	310000	360000	-50000	-56666.668	303333.332	6666.66797	8333.33496	311666.667
1800	330000	360000	-30000	-56666.668	303333.332	26666.668	8333.33496	311666.667
2000	420000	360000	60000	85000	445000	-25000	8333.33496	453333.335
2200	470000	360000	110000	85000	445000	25000	8333.33496	453333.335


XG BOOST MODEL
Residual R1
Residual R2
Residual R3
Predict
Predict
Predict
Model1
Model2
Model3
Train
Train
Train
(X,y)
(X,R1)
(X,R2)

Final Prediction Formula

$$\hat{y}(x) = \hat{y}_0 + \eta \cdot f_1(x) + \eta \cdot f_2(x) + \eta \cdot f_3(x)$$

Final Prediction

- Learning rate is called “eta”

$$\hat{y}(x) = \hat{y}_0 + \eta \cdot f_1(x) + \eta \cdot f_2(x) + \eta \cdot f_3(x)$$

housesize	houseprice	Pred1	residual_1	pred_2	Updated_Model2_pred	residual_2	pred_3	final_pred
1200	270000	360000	-90000	-56666.668	303333.332	-33333.332	-33333.332	270000
1500	310000	360000	-50000	-56666.668	303333.332	6666.66797	8333.33496	311666.667
1800	330000	360000	-30000	-56666.668	303333.332	26666.668	8333.33496	311666.667
2000	420000	360000	60000	85000	445000	-25000	8333.33496	453333.335
2200	470000	360000	110000	85000	445000	25000	8333.33496	453333.335

360000 + -56666.668 + -33333.332 = 270000

XGB Advantages and Disadvantages

- Jupyter notebook