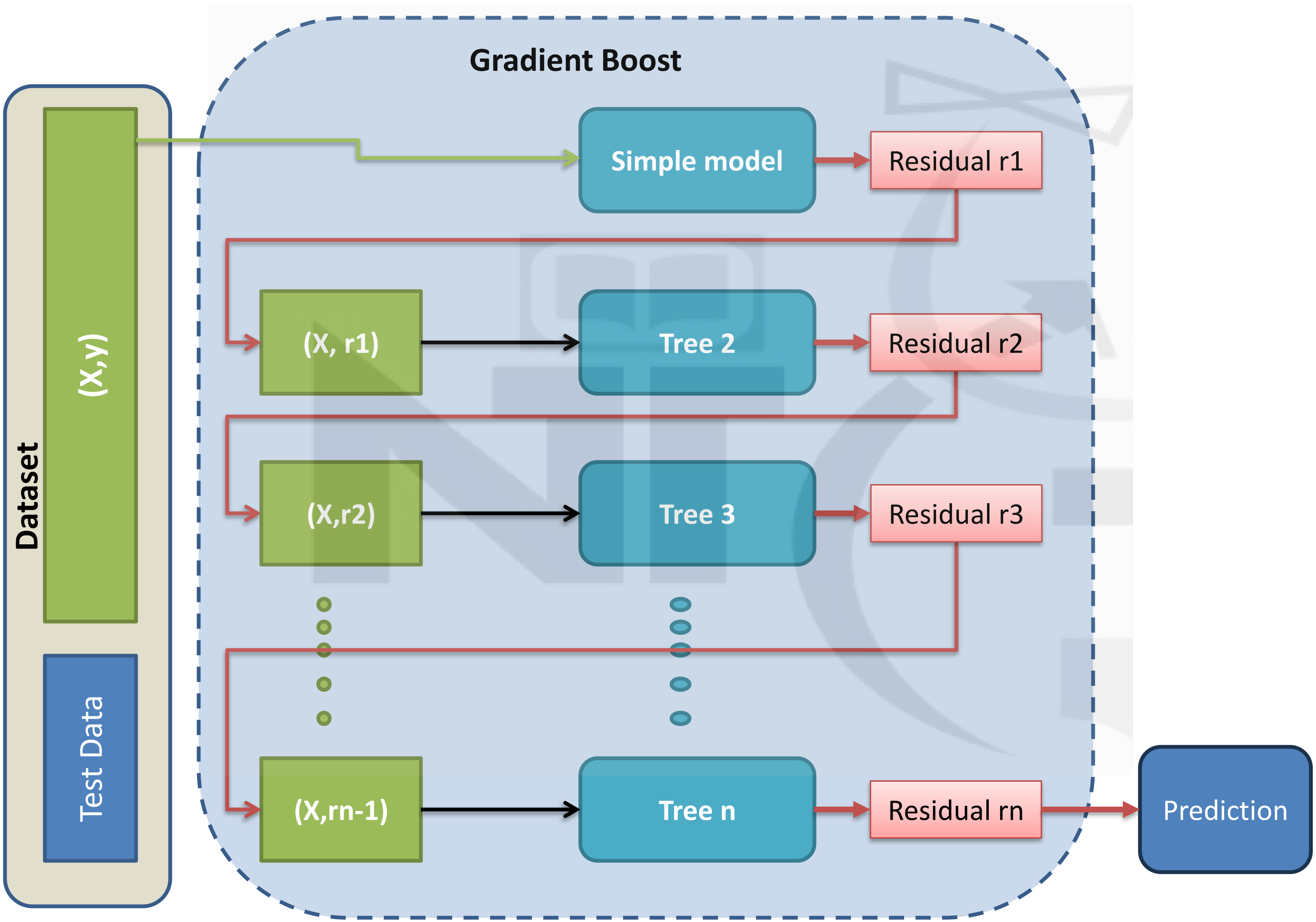
Gradient Boosting

-MUKESH KUMAR

Boosting

- Boosting reduces both bias and variance by **sequentially** training models, where each new model focuses on the mistakes made by the previous ones.

- Models are trained one after the other, and each tries to correct the errors of its predecessor.


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

How Gradient Boosting works?

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).

GB Algorithm

GRADIENT BOOSTING FOR REGRESSSION

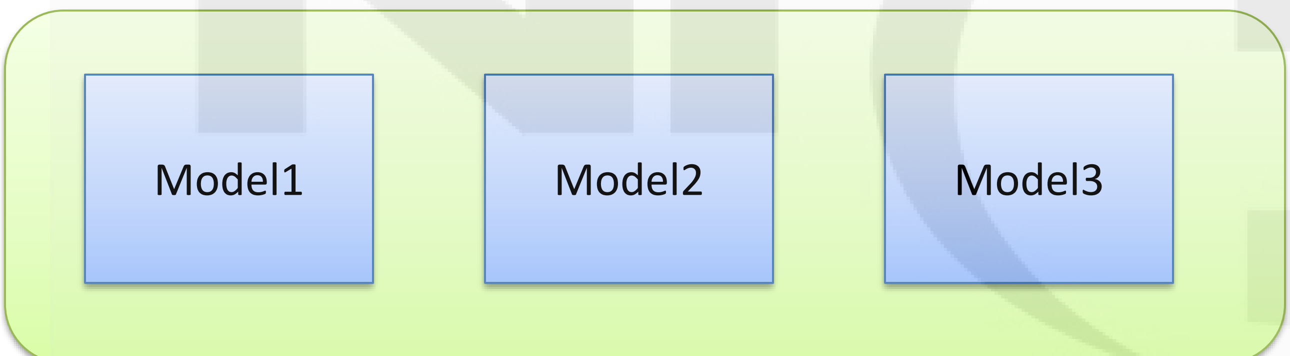
Sample data

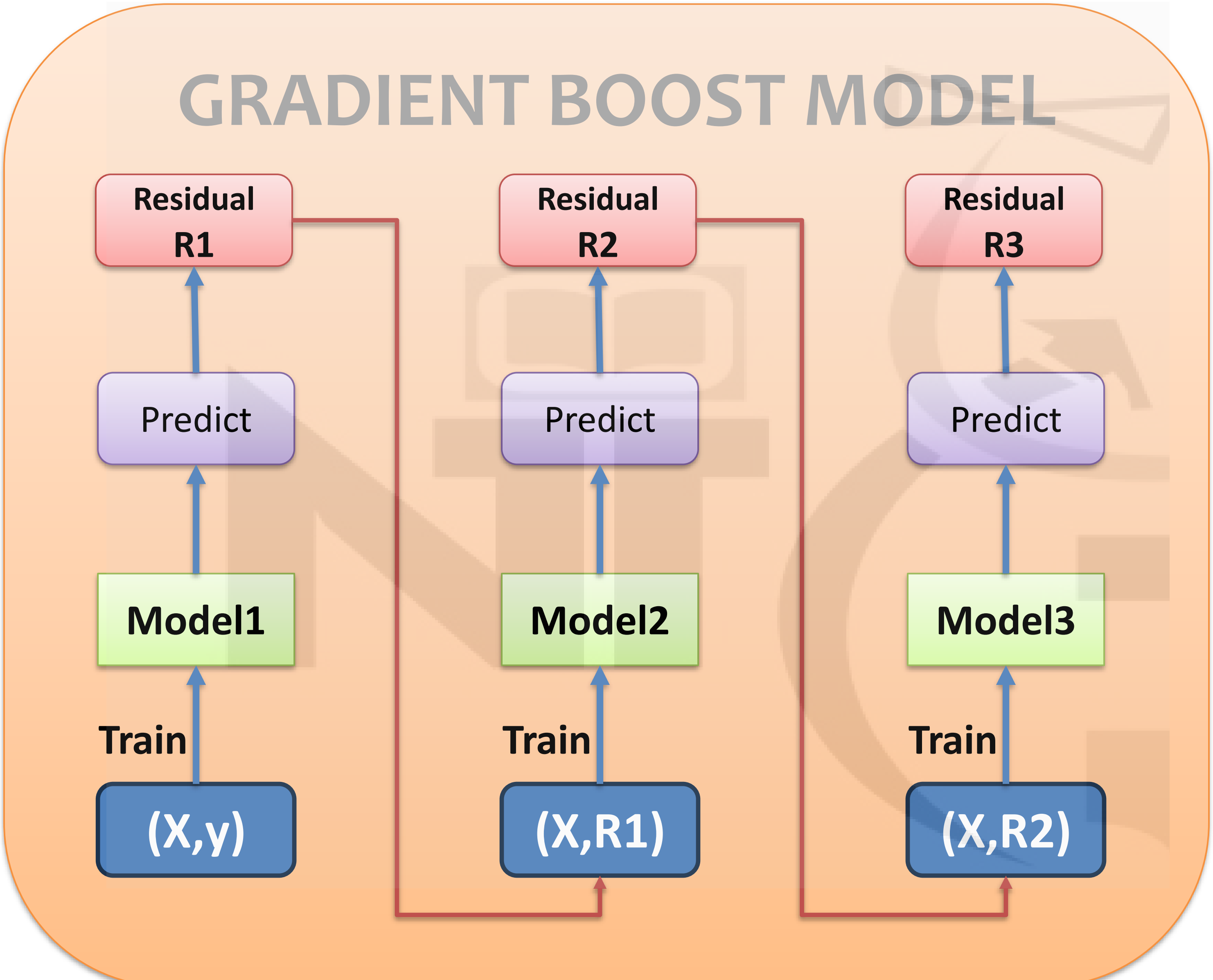
We are solving a Regression Problem

House Size	NumberOf Bedrooms	HousePrice	Model1 Prediction
1200	3	250000	350000
1500	4	300000	350000
1800	3	350000	350000
2000	4	400000	350000
2200	5	450000	350000

Build a simple Gradient Boosting

- Let's say we want to build a GB with 3 models





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

In Regression Problem:

- Model 1 Pred will simply be an average of all the house prices
- This is just a starting point for the subsequent models

House Size	NumberOf Bedrooms	HousePrice	Model1 Prediction	Residual Model1
1200	3	250000	350000	-100000
1500	4	300000	350000	-50000
1800	3	350000	350000	0
2000	4	400000	350000	50000
2200	5	450000	350000	100000

Residual for Model1

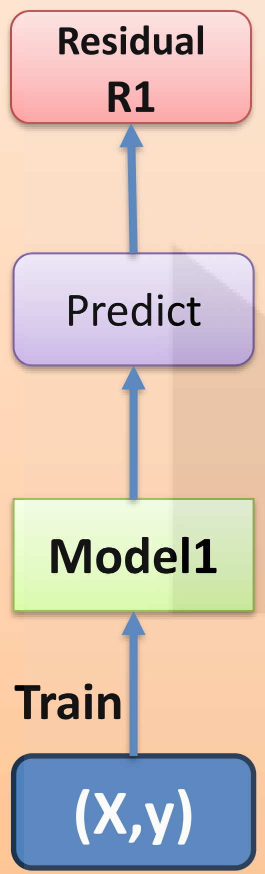
Calculate the Residual for model 1:

- Actual minus predicted

$r^{(1)} = y - \hat{y}^{(0)}$: residuals after Model 0

House Size	NumberOf Bedrooms	HousePrice	Model1 Prediction	Residual Model1
1200	3	250000	350000	-100000
1500	4	300000	350000	-50000
1800	3	350000	350000	0
2000	4	400000	350000	50000
2200	5	450000	350000	100000

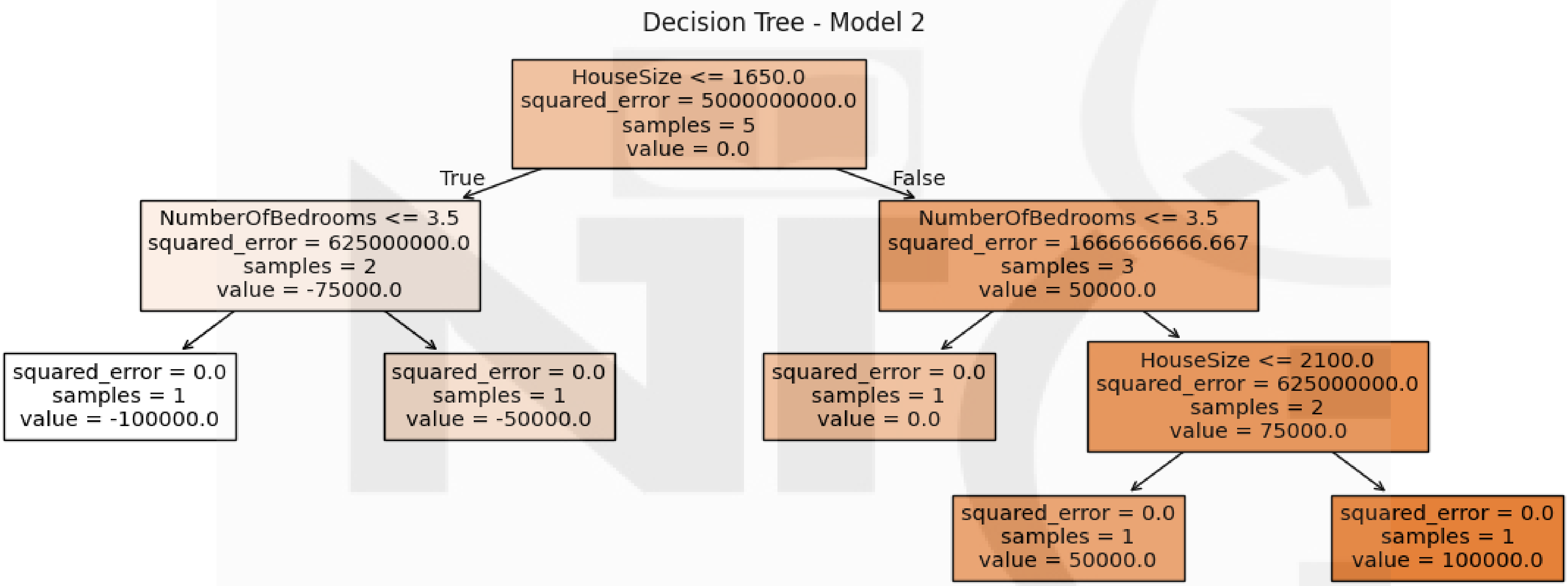
GRADIENT BOOST MODEL



Model2: Build a Decision Tree

- Model 2 is built on X and residual of model 1
- Features = [HouseSize, NoOfBedrooms]
- Labels = Residuals from model 1

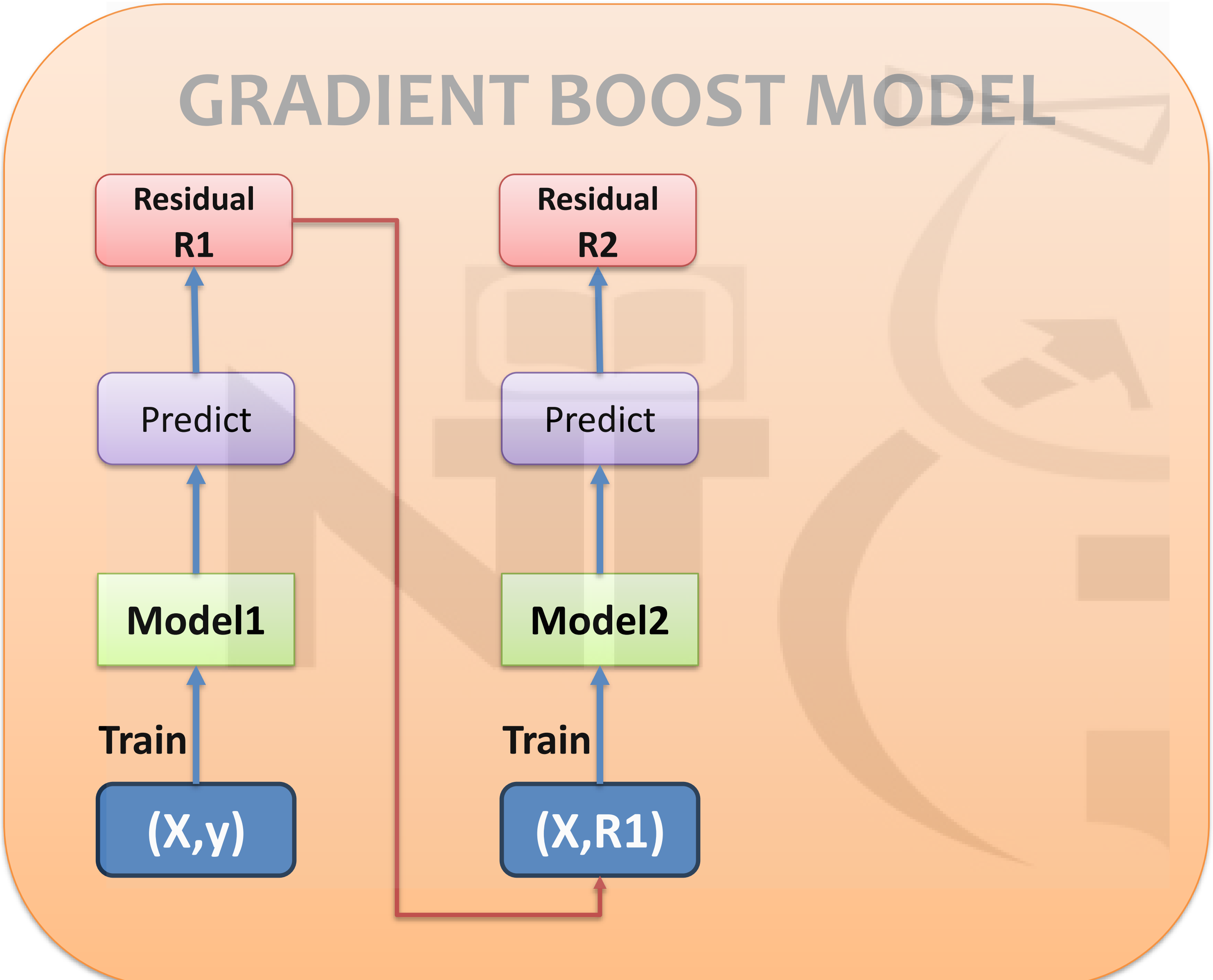
Model 2


Decision Tree - Model 2
HouseSize <= 1650.0
squared_error = 5000000000.0
samples = 5
value = 0.0
True
False
NumberOfBedrooms <= 3.5
squared_error = 625000000.0
samples = 2
value = -75000.0
NumberOfBedrooms <= 3.5
squared_error = 1666666666.667
samples = 3
value = 50000.0
squared_error = 0.0
samples = 1
value = -100000.0
squared_error = 0.0
samples = 1
value = -50000.0
squared_error = 0.0
samples = 1
value = 0.0
HouseSize <= 2100.0
squared_error = 625000000.0
samples = 2
value = 75000.0
squared_error = 0.0
samples = 1
value = 50000.0
squared_error = 0.0
samples = 1
value = 100000.0

Model 2 predictions

Model 2 will predict residuals

House Size	NumberOf Bedrooms	HousePrice	Model1 Prediction	Residual Model1	Model2 Prediction
1200	3	250000	350000	-100000	-100000
1500	4	300000	350000	-50000	-50000
1800	3	350000	350000	0	0
2000	4	400000	350000	50000	50000
2200	5	450000	350000	100000	100000


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

If there were only two models in our Gradient boosting then the final prediction at this point will be :

Final Pred = (M1 Pred + M2 Pred)

House Size	NumberOf Bedrooms	HousePrice	Model1 Prediction	Residual Model1	Model2 Prediction	Updated Prediction Model2
1200	3	250000	350000	-100000	-100000	250000
1500	4	300000	350000	-50000	-50000	300000
1800	3	350000	350000	0	0	350000
2000	4	400000	350000	50000	50000	400000
2200	5	450000	350000	100000	100000	450000

Calculate Model2 Residuals

M2 Residuals = actual price – (M1 Pred + M2 Pred)

HouseSize	NumberOf Bedrooms	HousePrice	Model1 Prediction	Residual Model1	Model2 Prediction	Updated_Prediction_Model2	Residual Model2
1200	3	250000	350000	-100000	-100000	250000	0
1500	4	300000	350000	-50000	-50000	300000	0
1800	3	350000	350000	0	0	350000	0
2000	4	400000	350000	50000	50000	400000	0
2200	5	450000	350000	100000	100000	450000	0

- To avoid overfitting, we use learning rate
- **M2 Residuals = actual price – (M1 Pred + (LR*M2 Pred))**

Residuals with LearningRate

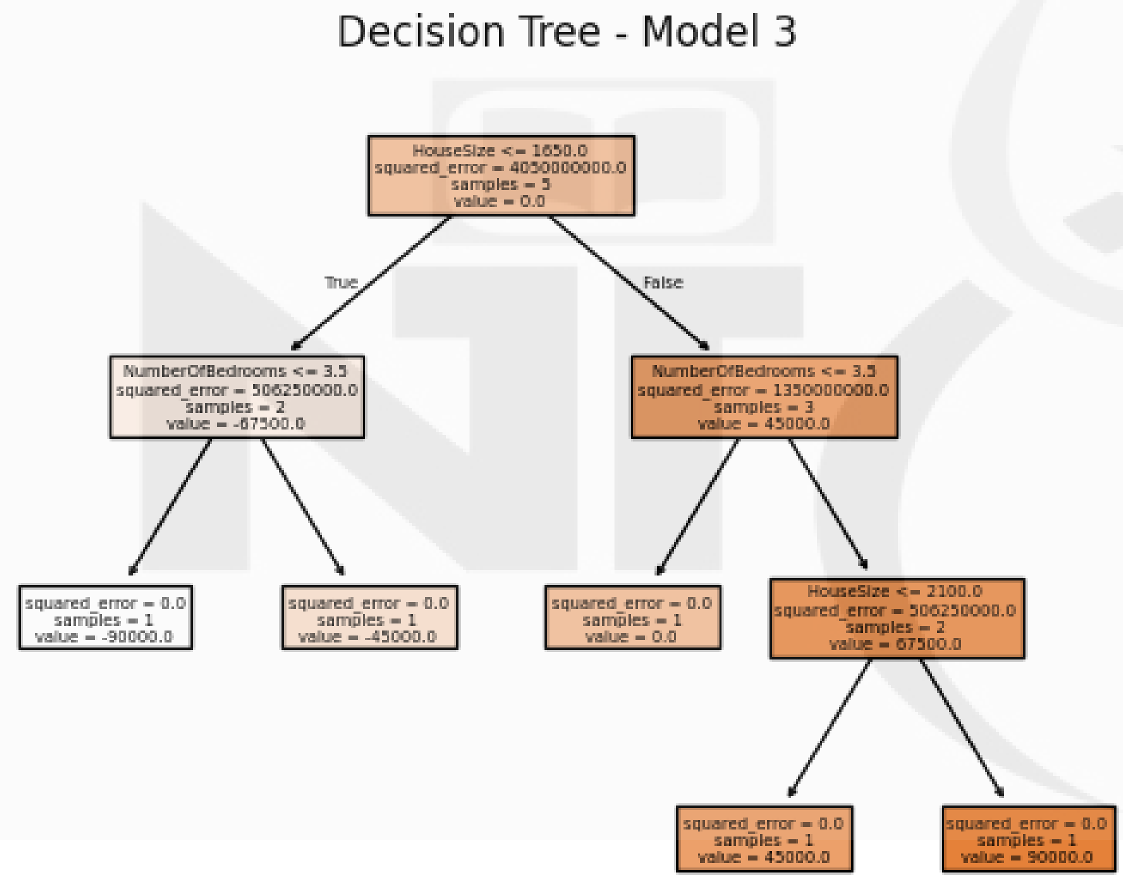
- So, after using learning rate residuals are not zero but they are smaller than previous model's residuals

HouseSize	NumberOfBedrooms	HousePrice	Model1_Prediction	Residual_Model1	Model2_Prediction	Updated_Prediction_Model2	Residual_Model2_overfit	Residual Model2
1200	3	250000	350000	-100000	-100000	340000	0	-90000
1500	4	300000	350000	-50000	-50000	345000	0	-45000
1800	3	350000	350000	0	0	350000	0	0
2000	4	400000	350000	50000	50000	355000	0	45000
2200	5	450000	350000	100000	100000	360000	0	90000

Model 3 : Build another Tree

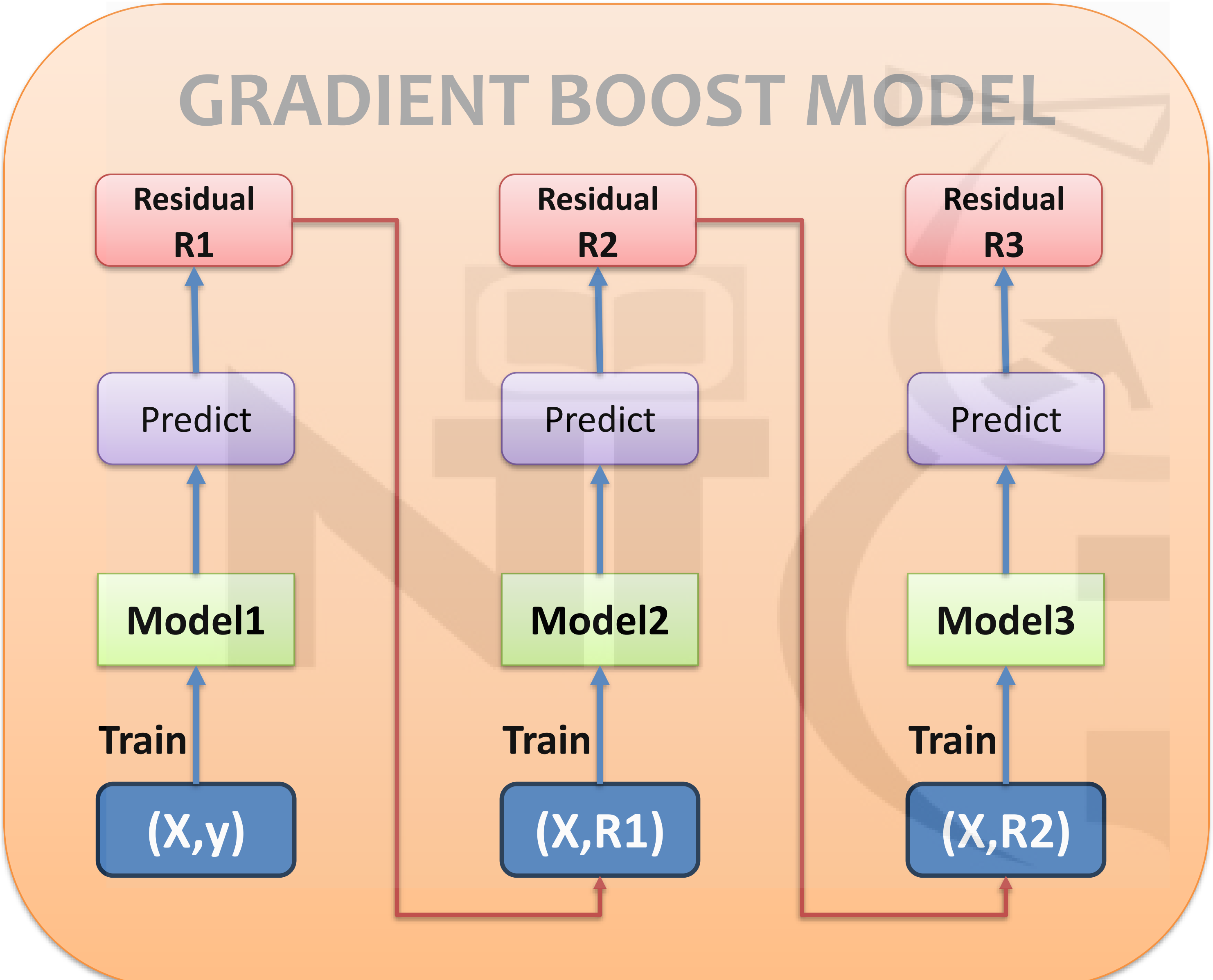
- Features = [HouseSize, NoOfBedrooms]
- Labels = Residuals from model 2

Model 3


Decision Tree - Model 3
HouseSize <= 1650.0
squared_error = 4050000000.0
samples = 5
value = 0.0
True
False
NumberOfBedrooms <= 3.5
squared_error = 506250000.0
samples = 2
value = -67500.0
NumberOfBedrooms <= 3.5
squared_error = 1350000000.0
samples = 3
value = 45000.0
squared_error = 0.0
samples = 1
value = -90000.0
squared_error = 0.0
samples = 1
value = -45000.0
squared_error = 0.0
samples = 1
value = 0.0
HouseSize <= 2100.0
squared_error = 506250000.0
samples = 2
value = 67500.0
squared_error = 0.0
samples = 1
value = 45000.0
squared_error = 0.0
samples = 1
value = 90000.0

Model3 Prediction

HouseSize	NumberOfBedrooms	HousePrice	Model1_Prediction	Residual_Model1	Model2_Prediction	Updated_Prediction_Model2	Residual_Model2_overfit	Residual Model2	Model3 Prediction
1200	3	250000	350000	-100000	-100000	340000	0	-90000	-90000
1500	4	300000	350000	-50000	-50000	345000	0	-45000	-45000
1800	3	350000	350000	0	0	350000	0	0	0
2000	4	400000	350000	50000	50000	355000	0	45000	45000
2200	5	450000	350000	100000	100000	360000	0	90000	90000


GRADIENT 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

- Simply add the output of all the models

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).

Final Prediction of GB

Final Pred = M1 Pred + (LR* M2 Pred) + (LR* M3 Pred)

We can see that errors are reducing with each newly added model

House Size	NumberOf Bedrooms	HousePrice	Model1 Prediction	Residual Model1	Model2_Prediction	Updated_Prediction_Model2	Residual_Model2_overfit	Residual Model2	Model3 Prediction	Final Prediction
1200	3	250000	350000	-100000	-100000	340000	0	-90000	-90000	331000
1500	4	300000	350000	-50000	-50000	345000	0	-45000	-45000	340500
1800	3	350000	350000	0	0	350000	0	0	0	350000
2000	4	400000	350000	50000	50000	355000	0	45000	45000	359500
2200	5	450000	350000	100000	100000	360000	0	90000	90000	369000

Summary

- We started with mean of house prices which is model1 output
- Calculate Residuals for M1
- Train M2 on X, R1
- M2 Predicts residuals, calculate M2 residuals
- Train M3 on X, R2
- M3 predicts Residuals
- Calculate the final prediction from GB

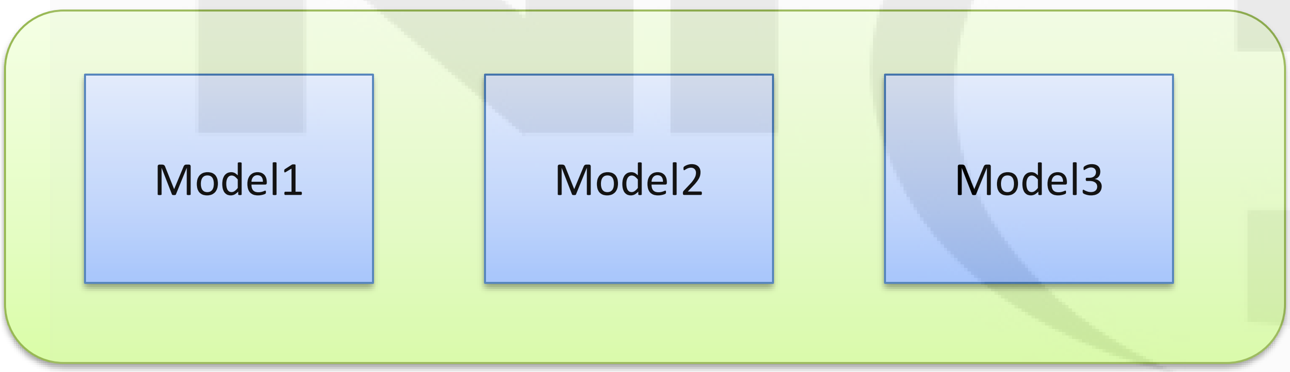
GRADIENT BOOSTING FOR CLASSIFICATION

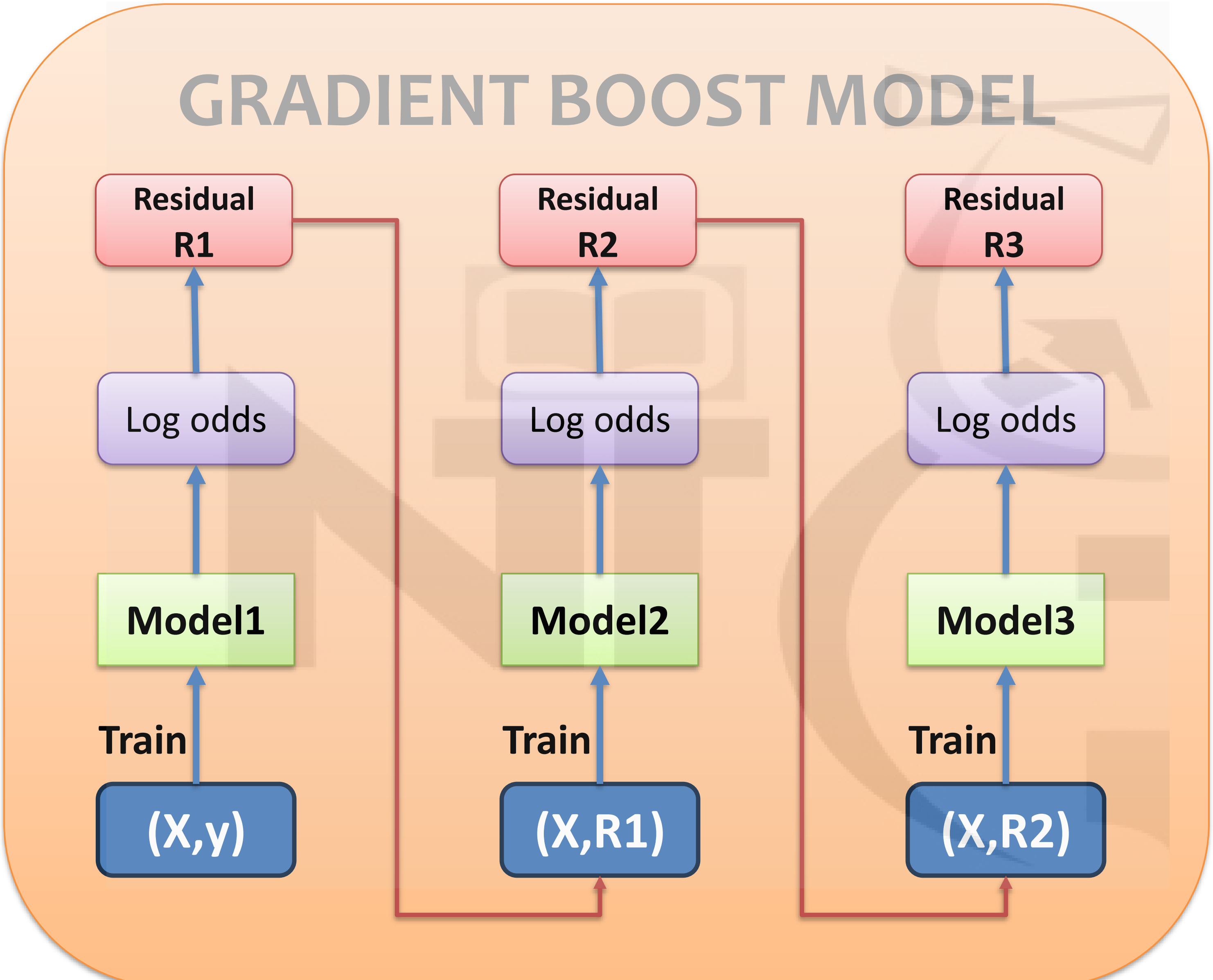
Sample Data

ID	Glucose	Weight	Diabetic
1	185	82	1
2	89	65	0
3	120	70	1
4	130	72	1
5	150	78	1
6	95	60	0
7	140	75	1
8	100	66	0

Build a simple Gradient Boosting

- Let's say we want to build a GB with 3 models





GRADIENT BOOST MODEL
Residual R1
Residual R2
Residual R3
Log odds
Log odds
Log odds
Model1
Model2
Model3
Train
Train
Train
(X,y)
(X,R1)
(X,R2)

Model 1

- In gradient boost we start with a simple model , in regression we took mean

- In classification we calculate log(odds)

- Log(odds) = log(p/1-p)

- Log base e is used

Glucose	Weight	Diabetic	pred1 (log-odds)
185	82	1	0.510826
89	65	0	0.510826
120	70	1	0.510826
130	72	1	0.510826
150	78	1	0.510826
95	60	0	0.510826
140	75	1	0.510826
100	66	0	0.510826

Calculate Residuals

- To calculate residuals we need to do :
  - Actual – predicted value
- But actual value is a prob and we have log(odds) form model 1
- First we need to convert log(odds) to probability using sigmoid function

Looking at the pred1 probablity :

- all the values are above 0.5 which means all the records are classified as class 1 > diabetic

- This is a very basic model hence the predictions are not great but it's a good starting point

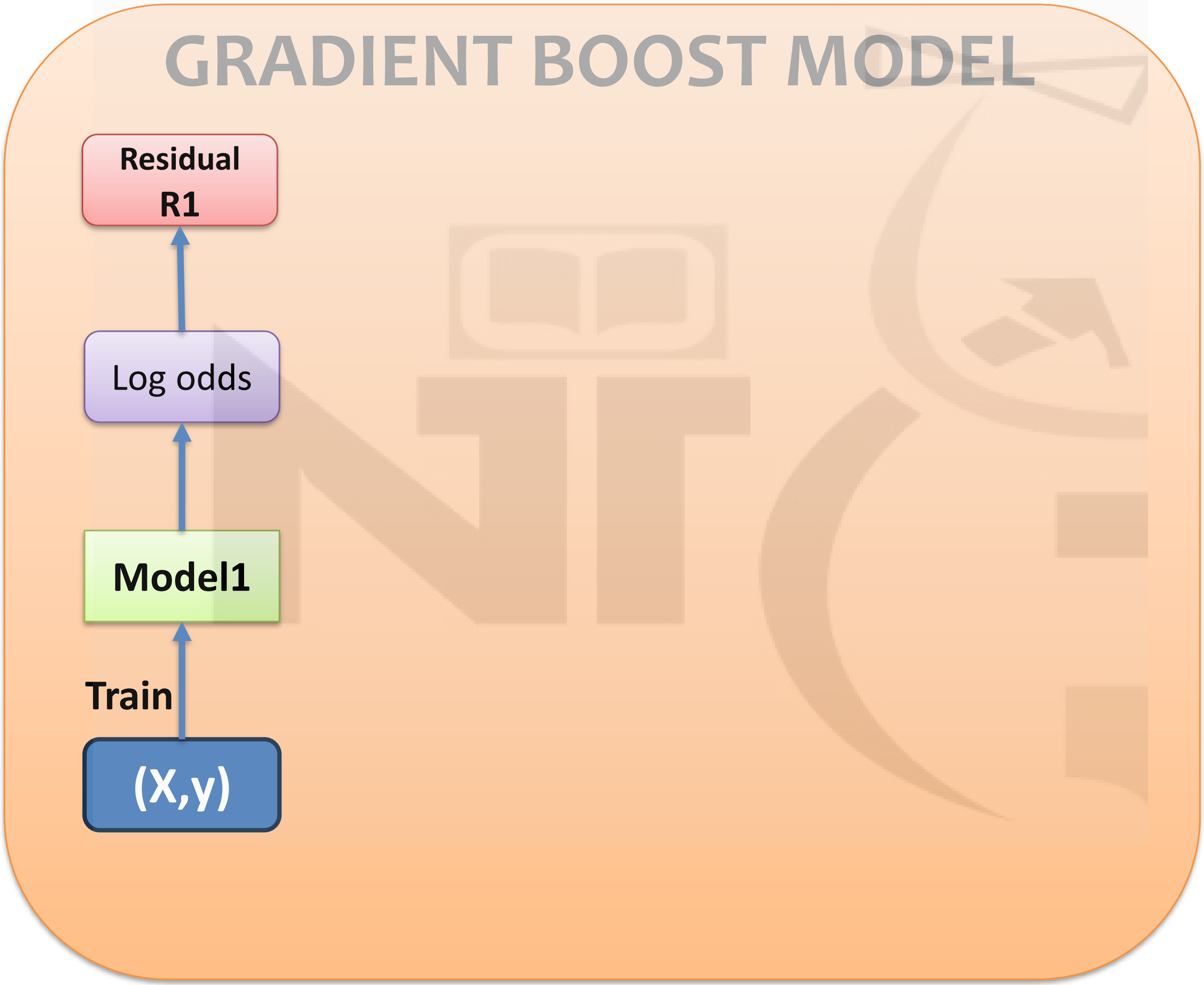
Glucose	Weight	Diabetic	pred1 (log-odds)	pred1 (probability)
185	82	1	0.510826	0.625
89	65	0	0.510826	0.625
120	70	1	0.510826	0.625
130	72	1	0.510826	0.625
150	78	1	0.510826	0.625
95	60	0	0.510826	0.625
140	75	1	0.510826	0.625
100	66	0	0.510826	0.625

Calculate Residuals of M1

Residuals = Actual – Predicted

In our case = Diabetic - Pred1(Prob)

Glucose	Weight	Diabetic	pred1 (log-odds)	pred1 (probability)	res1
185	82	1	0.510826	0.625	0.375
89	65	0	0.510826	0.625	-0.625
120	70	1	0.510826	0.625	0.375
130	72	1	0.510826	0.625	0.375
150	78	1	0.510826	0.625	0.375
95	60	0	0.510826	0.625	-0.625
140	75	1	0.510826	0.625	0.375
100	66	0	0.510826	0.625	-0.625

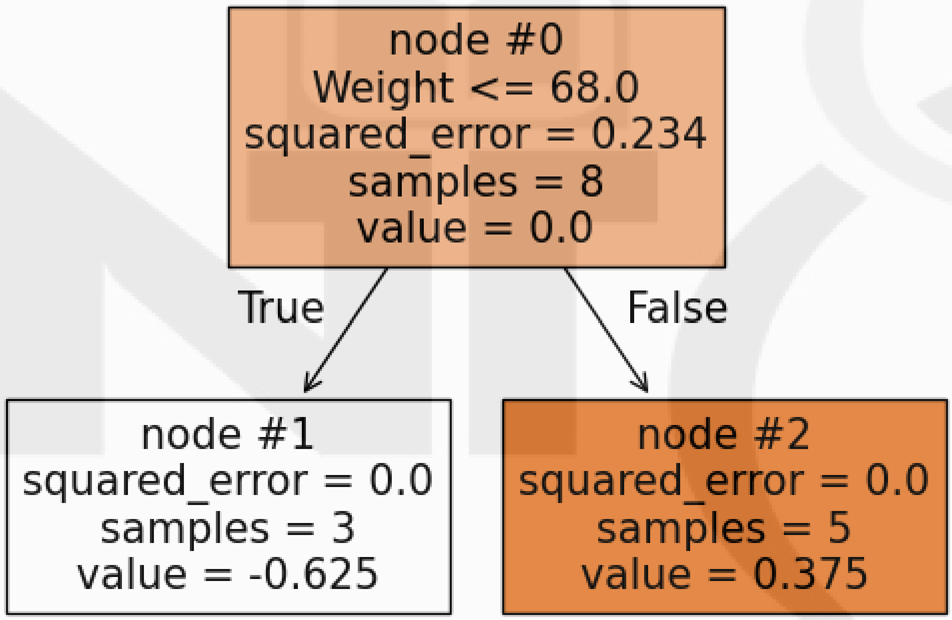

GRADIENT BOOST MODEL
Residual
R1
Log odds
Model1
Train
(X,y)

Model 2 : Build Regression Tree

- Features = [Glucose, Weight]

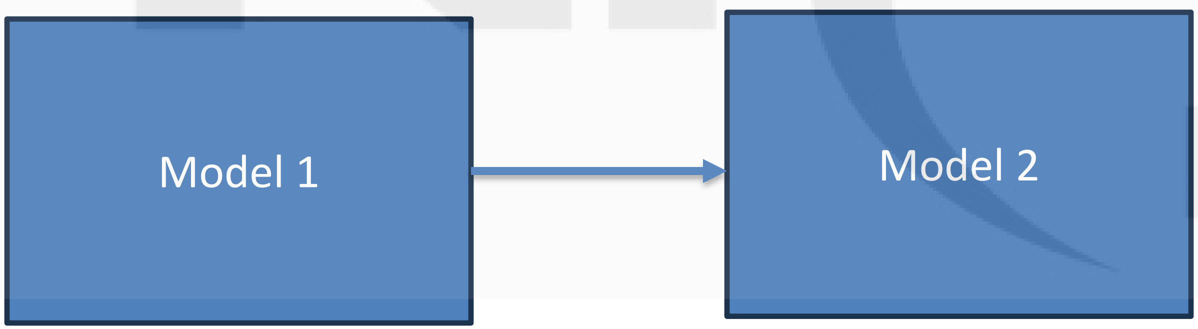
- Label = residual M1

Model 2


node #0
Weight <= 68.0
squared_error = 0.234
samples = 8
value = 0.0
True
False
node #1
squared_error = 0.0
samples = 3
value = -0.625
node #2
squared_error = 0.0
samples = 5
value = 0.375

Model at this point

- We have two models ready
- So final output will be the output of M1 +M2



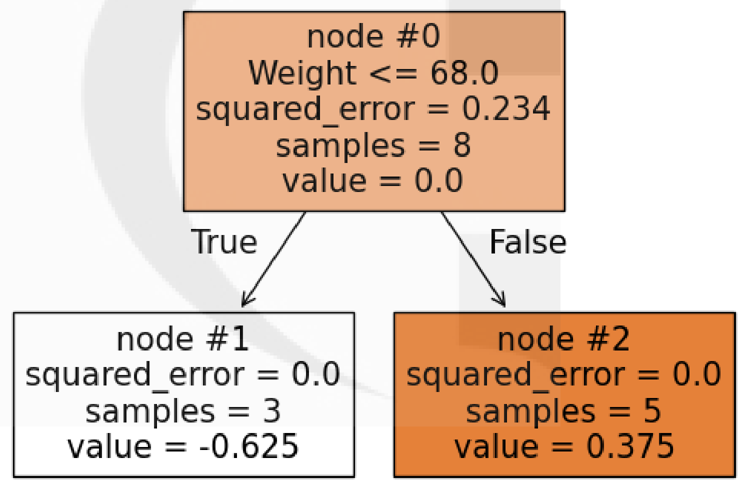
- M1 output is log(odds)
- But the tree we create at each leaf node give us probability , so we need to transform probabilities to log(odds)
- This way we can add both log(odds) from M1 and M2 and obtain the final output

Why logodds over probabilities

- If we have 3 models , final output:
  - **Using logodds**
    - F = 0.4 + 0.3 + 0.5 = 1.2
    - p = sigmoid(1.2) = 0.768
  - **Using Prob:**
    - p = 0.598 + 0.574 + 0.622 = 1.794 ← not valid!

- 3 sample in node1 and 5 samples in node 2
- We need to findout which records fall in which node so that for individual records we can use these prob value to get log (odds)

$$\frac{\sum Residual}{\sum[PreviousProb * (1 - PreviousProb)]}$$

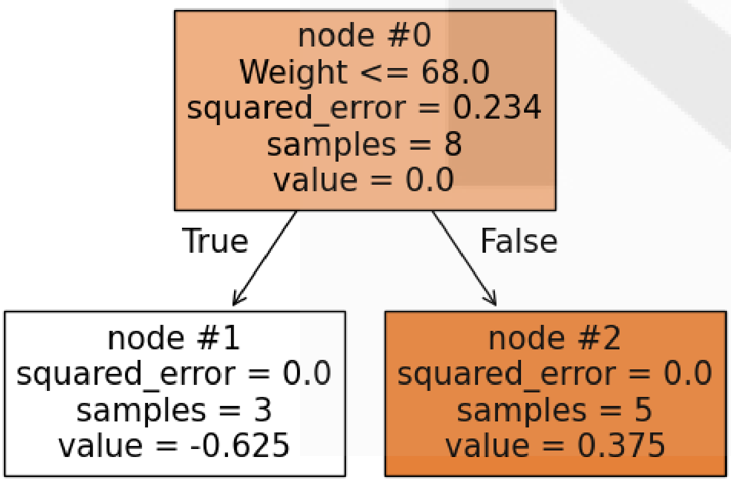


- Sklearn provides a direct formula to find which node each record belongs to
- Leaf_entry1 shows the information

Glucose	Weight	Diabetic	pred1 (log-odds)	pred1 (probability)	res1	leaf_entry1
185	82	1	0.510826	0.625	0.375	2
89	65	0	0.510826	0.625	-0.625	1
120	70	1	0.510826	0.625	0.375	2
130	72	1	0.510826	0.625	0.375	2
150	78	1	0.510826	0.625	0.375	2
95	60	0	0.510826	0.625	-0.625	1
140	75	1	0.510826	0.625	0.375	2
100	66	0	0.510826	0.625	-0.625	1

Calculate of log odds of left node

$$\frac{\sum Residual}{\sum[PreviousProb * (1 - PreviousProb)]}$$



Glucose	Weight	Diabetic	pred1 (log-odds)	pred1 (probability)	res1	leaf_entry 1
185	82	1	0.510826	0.625	0.375	2
89	65	0	0.510826	0.625	-0.625	1
120	70	1	0.510826	0.625	0.375	2
130	72	1	0.510826	0.625	0.375	2
150	78	1	0.510826	0.625	0.375	2
95	60	0	0.510826	0.625	-0.625	1
140	75	1	0.510826	0.625	0.375	2
100	66	0	0.510826	0.625	-0.625	1

Log(odds) calculation for each node

node 1

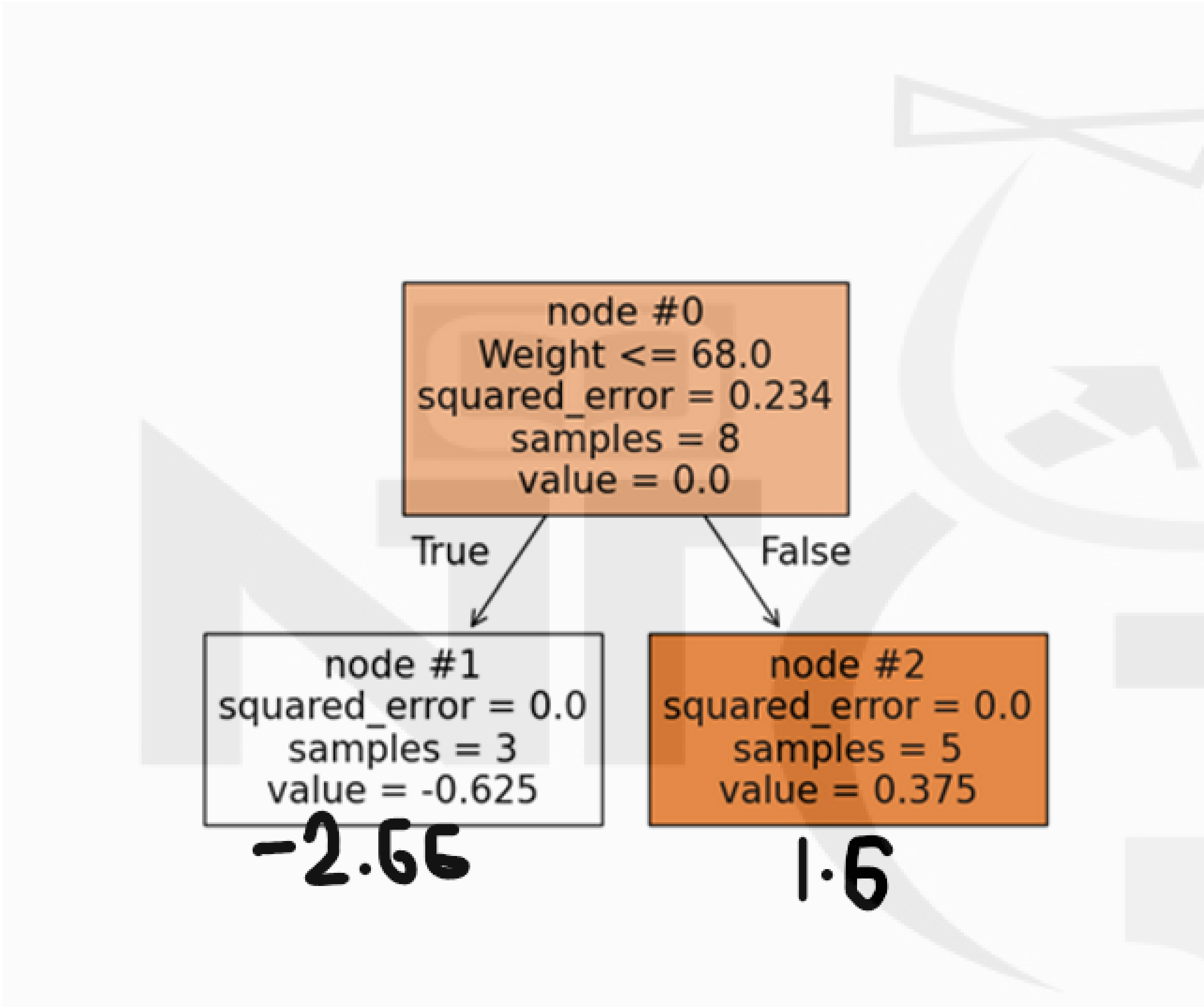
$$= \frac{3\times(-.625)}{3\times 0.625\times(1-0.625)}$$

$$= \frac{-1.875}{0.7031} = -2.66$$

node 2 :

$$= \frac{5\times 0.375}{5\times 0.625(1-0.625)} = \frac{0.375}{0.2343}$$

$$= 1.6$$


node #0
Weight <= 68.0
squared_error = 0.234
samples = 8
value = 0.0
True
False
node #1
squared_error = 0.0
samples = 3
value = -0.625
node #2
squared_error = 0.0
samples = 5
value = 0.375
-2.66
1.6

After log(odds) calculation of combined M1 + M2

Glucose	Weight	Diabetic	pred1 (log-odds)	pred1 (probability)	res1	leaf_entry1	pred2 (log-odds)
185	82	1	0.510826	0.625	0.375	2	2.110826
89	65	0	0.510826	0.625	-0.625	1	-2.159174
120	70	1	0.510826	0.625	0.375	2	2.110826
130	72	1	0.510826	0.625	0.375	2	2.110826
150	78	1	0.510826	0.625	0.375	2	2.110826
95	60	0	0.510826	0.625	-0.625	1	-2.159174
140	75	1	0.510826	0.625	0.375	2	2.110826
100	66	0	0.510826	0.625	-0.625	1	-2.159174

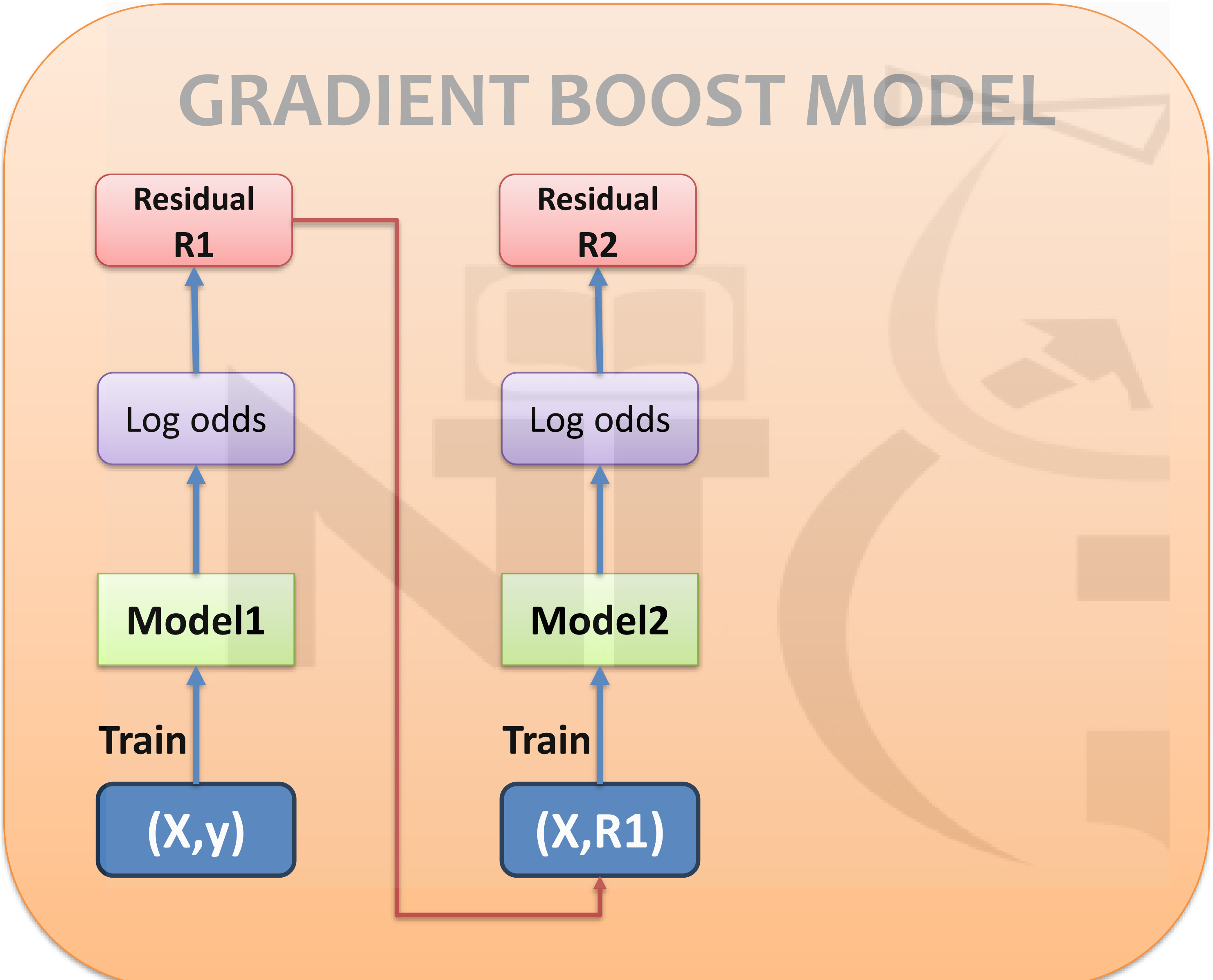
Calculate Residuals M2

- Convert log(odds) to probab and then actual minus predicted
- **Diabetic – Pred2Probab**

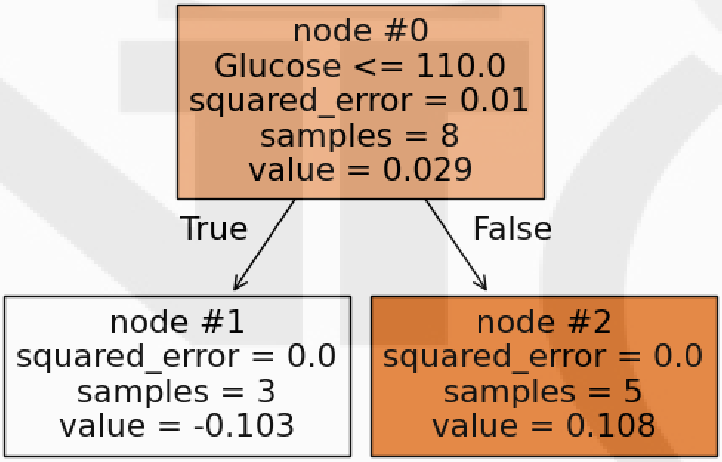
Glucose	Weight	Diabetic	pred1 (log-odds)	pred1 (probability)	res1	leaf_entry 1	pred2 (log-odds)	pred2 (probability)	res2
185	82	1	0.510826	0.625	0.375	2	2.110826	0.891951	0.108049
89	65	0	0.510826	0.625	-0.625	1	-2.159174	0.103477	-0.103477
120	70	1	0.510826	0.625	0.375	2	2.110826	0.891951	0.108049
130	72	1	0.510826	0.625	0.375	2	2.110826	0.891951	0.108049
150	78	1	0.510826	0.625	0.375	2	2.110826	0.891951	0.108049
95	60	0	0.510826	0.625	-0.625	1	-2.159174	0.103477	-0.103477
140	75	1	0.510826	0.625	0.375	2	2.110826	0.891951	0.108049
100	66	0	0.510826	0.625	-0.625	1	-2.159174	0.103477	-0.103477

- Reading pred2Prob column its clear that the probability values are now getting closer to actual values

Glucose	Weight	Diabetic	pred1 (log-odds)	pred1 (probability)	res1	leaf_entry 1	pred2 (log-odds)	pred2 (probability)	res2
185	82	1	0.510826	0.625	0.375	2	2.110826	0.891951	0.108049
89	65	0	0.510826	0.625	-0.625	1	-2.159174	0.103477	-0.103477
120	70	1	0.510826	0.625	0.375	2	2.110826	0.891951	0.108049
130	72	1	0.510826	0.625	0.375	2	2.110826	0.891951	0.108049
150	78	1	0.510826	0.625	0.375	2	2.110826	0.891951	0.108049
95	60	0	0.510826	0.625	-0.625	1	-2.159174	0.103477	-0.103477
140	75	1	0.510826	0.625	0.375	2	2.110826	0.891951	0.108049
100	66	0	0.510826	0.625	-0.625	1	-2.159174	0.103477	-0.103477

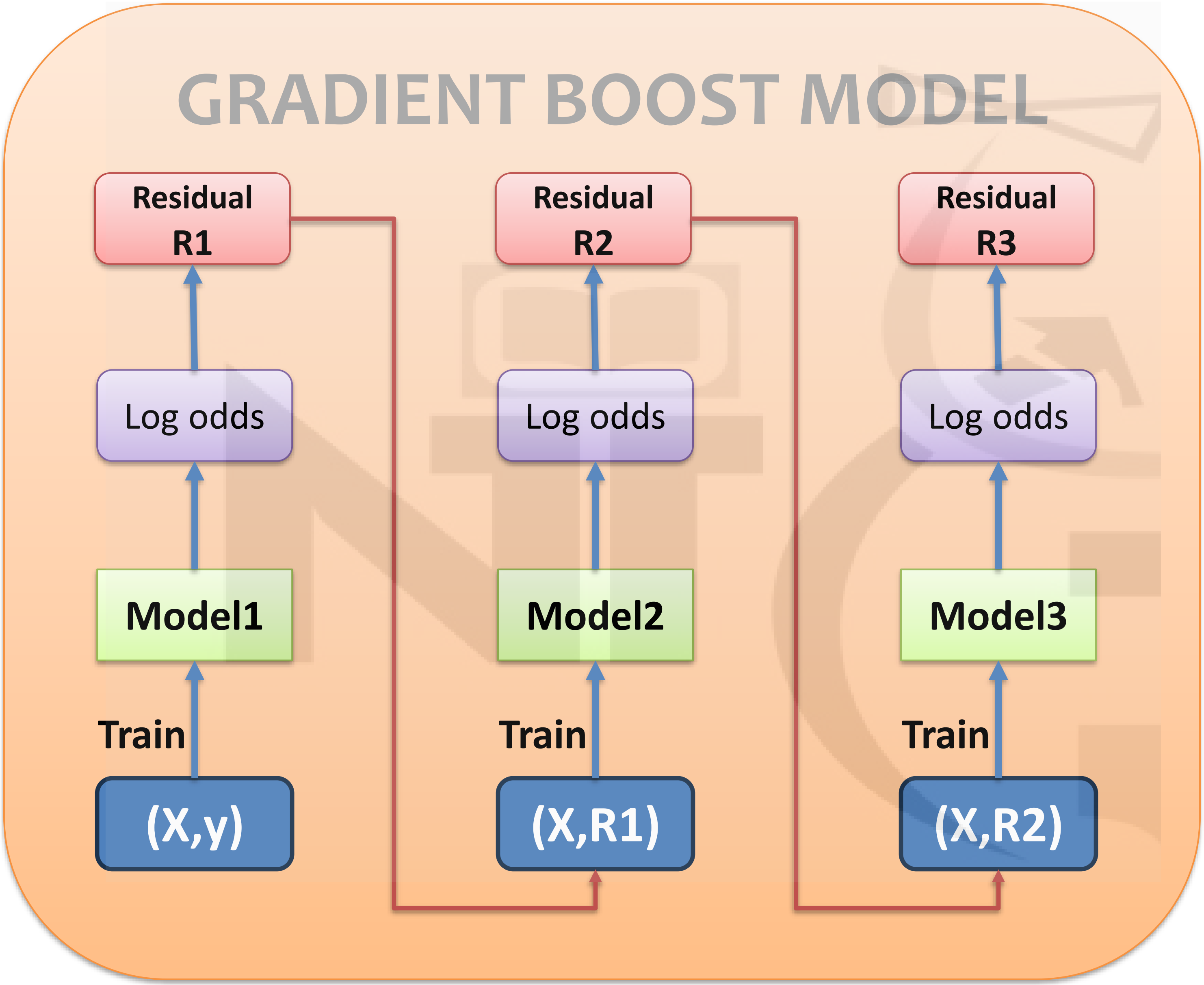

GRADIENT BOOST MODEL
Residual
R1
Residual
R2
Log odds
Log odds
Model1
Model2
Train
Train
(X,y)
(X,R1)

Model3 : Decision Tree

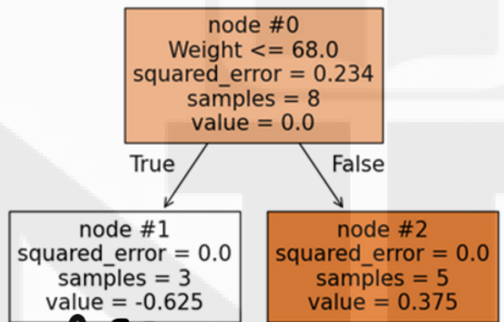


Repeating the same process we get Log(odds)>Prob

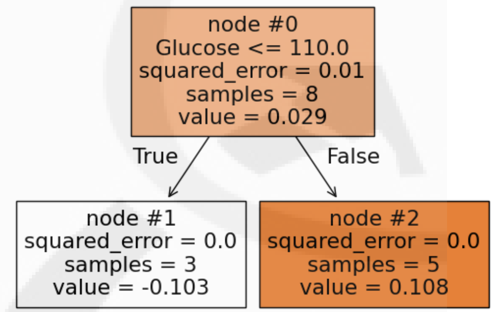
Glucose	Weight	Diabetic	pred1 (log-odds)	pred1 (probability)	res1	leaf_entry 1	pred2 (log-odds)	pred2 (probability)	res2	leaf_entry 2	pred3 (log-odds)	pred3 (probability)
185	82	1	0.510826	0.625	0.375	2	2.110826	0.891951	0.108049	2	3.741651	0.976834
89	65	0	0.510826	0.625	-0.625	1	-2.159174	0.103477	-0.103477	1	-2.768349	0.059059
120	70	1	0.510826	0.625	0.375	2	2.110826	0.891951	0.108049	2	3.741651	0.976834
130	72	1	0.510826	0.625	0.375	2	2.110826	0.891951	0.108049	2	3.741651	0.976834
150	78	1	0.510826	0.625	0.375	2	2.110826	0.891951	0.108049	2	3.741651	0.976834
95	60	0	0.510826	0.625	-0.625	1	-2.159174	0.103477	-0.103477	1	-2.768349	0.059059
140	75	1	0.510826	0.625	0.375	2	2.110826	0.891951	0.108049	2	3.741651	0.976834
100	66	0	0.510826	0.625	-0.625	1	-2.159174	0.103477	-0.103477	1	-2.768349	0.059059


GRADIENT BOOST MODEL
Residual
R1
Residual
R2
Residual
R3
Log odds
Log odds
Log odds
Model1
Model2
Model3
Train
Train
Train
(X,y)
(X,R1)
(X,R2)

.510



-2.65 1.6



-1.116 1.12

Glucose = 95 , weight = 80

.510 + (1.6) + (-1.116)

GB Advantages

- High Predictive Accuracy
- Handles Different Data Types
- **Robust to Outliers and Noise**
  - Boosting builds models sequentially and focuses on **hard-to-predict instances,** making it **more resilient** to noise than bagging methods like Random Forest.

Disadvantages of Gradient Boosting

- Computationally Intensive
- Sensitive to Hyperparameters
- Not Easily Interpretable
- Memory Usage