XGBoost algorithm

อ.ดร.สิวะโชติ ศรีสุทธิยากร
ภาควิชาวิจัยและจิตวิทยาการศึกษา คณะครุศาสตร์
จุฬาลงกรณ์มหาวิทยาลัย

What is the difference between the R gbm (gradient boosting machine) and xgboost (extreme gradient boosting)?

Tianqi Chen, PhD on Large-scale Machine Learning

Answered September 4, 2015

I am the author of xgboost. Both xgboost and gbm follows the principle of gradient boosting. There are however, the difference in modeling details. Specifically, xgboost used a more regularized model formalization to control over-fitting, which gives it better performance.

We have updated a comprehensive tutorial on introduction to the model, whhich you might want to take a look at. Introduction to Boosted Trees

The name xgboost, though, actually refers to the engineering goal to push the limit of computations resources for boosted tree algorithms. Which is the reason why many people use xgboost. For model, it might be more suitable to be called as regularized gradient boosting.

XGBoost

Let $D = \{(\mathbf{x_i}, y_i)\}_{i=1}^{n}$ be a training dataset, a tree ensemble model uses K additive functions to predict the output.

$$\hat{y}_i = \phi(\mathbf{x}_i) = \sum_{m=1}^{M} f_m(\mathbf{x}_i),\ f_m \in F$$

- where $F = f(x) = \omega_{q(x)}$ ($q : \mathbb{R} \to \mathbb{T},\ \omega \in \mathbb{R}^{\mathbb{T}}$) is a regression tree space, q is tree structure, and T is number of leaves in the tree.
- Each f_m corresponds to an independent tree structure q and (vector of) leaf score ω (ω_i represent score on ith leaf.)

Gradient Boosting

Let $(X, y) = \{(\mathbf{x_i}, y_i)\}_{i=1}^{n}$ be a training dataset, and $L(y_i, F(\mathbf{x_i}))$ be differentiable Loss function.

Step 1: fit the null model (model with only constant term) $\implies$ $F_0(\mathbf{X}) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$

Step2: for m in $1 : M$

1. Compute $e_{im} = -[\frac{\partial L(y_i, F(\mathbf{x_i}))}{\partial F(\mathbf{x_i})}]_{F(x)=F_{m-1}(x)}$ for $i = 1, 2, ..., n$
   Pseudo residual
2. Fit a regression tree to the e_{im} values and create terminal region R_{jm} for $j = 1, ..., J_m$
3. For $j = 1, 2, ..., J_m$ compute $\gamma_{jm} = \underset{x}{\operatorname{argmin}} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$
4. Update $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \nu \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$

Output: $F_M(\mathbf{X})$

XGBoost

Let $(X, y) = \{(\mathbf{x_i}, y_i)\}_{i=1}^{n}$ be a training dataset, $l(y_i, \hat{y}_i)$ be differentiable Loss function,

$$L(\phi) = \sum_i l(y_i, \hat{y}_i) + \sum_m \Omega(f_m)$$

be regularized objective function, where $\Omega(f) = \gamma T + \frac{1}{2}\lambda \parallel \omega \parallel^2$.

Step 1: fit the null model (make initial prediction) ——> $\hat{y}_i^{(m=0)} = c$

Step2: for m in $1 : M$

1. Compute $e_{im} = -[\frac{\partial L^{(m-1)}}{\partial \hat{y}^{(m-1)}}]$ for $i = 1, 2, ..., n$
2. Fit a **XGBoost tree** to the e_{im} values and create terminal region R_{jm} for $j = 1, ..., T_m$
3. For $j = 1, ..., T_m$ compute $f_{jm} = \underset{f}{\operatorname{argmin}} \sum_{x_i \in R_{ij}} l(y_i, \hat{y}_i^{(m-1)} + f_m(\mathbf{x}_i)) + \Omega(f_m)$
4. Update $\hat{y}^m = \hat{y}^{(m-1)} + \epsilon \sum_{j=1}^{J_m} f_{jm} I(x \in R_{jm})$

Eta

Output: $\hat{y}^m$

XGBoost

- By using **Taylor serie approximation**, the second-order approximation of $L^{(m)}$ is

$$L^{(m)} \approx \sum_i [l(y_i, \hat{y}_i^{(m-1)}) + g_j f_m(\mathbf{x}_i) + \frac{1}{2} h_i f_m^2(\mathbf{x}_i)] + \Omega(f_m)$$

where $g_i = \partial_{\hat{y}^{(m-1)}} l(y_i, \hat{y}_i^{(m-1)})$ and $h_i = \partial^2_{\hat{y}^{(m-1)}} l(y_i, \hat{y}^{(m-1)})$ are first and second order derivative of loss function.

- We can remove constant term $l(y_i, \hat{y}_i^{(m-1)})$ to simplify the objective function in step 2.3

$$\begin{aligned}
\bar{L}^{(m)} &= \sum_i [g_i f_m(\mathbf{x}_i) + \frac{1}{2} h_i f_m^2(\mathbf{x}_i)] + \Omega(f_m) \\
&= \sum_i [g_i f_m(\mathbf{x}_i) + \frac{1}{2} h_i f_m^2(\mathbf{x}_i)] + \gamma T + \frac{1}{2} \sum_{j=1}^{T} \omega_j^2 \\
&= \sum_{j=1}^{T} [(\sum_{i \in R_j} g_i)\omega_j + \frac{1}{2}(\sum_{i \in R_j} h_i + \lambda)\omega_j^2] + \gamma T \\
&= \sum_{j=1}^{T} [G_j \omega_j + \frac{1}{2}(H_j + \lambda)\omega_j^2] + \gamma T
\end{aligned}$$

Loss = 0.5(y-y.hat)^2
L'=-(y-y.hat)
L" = 1

$\partial L / \partial \omega$ =G + (H+λ)w =0
L"= (sum(1)+λ)

- Given leaf j, the **optimal output value (leaf score)** is equal to

$$\omega_j^* = -\frac{G_j}{H_j + \lambda}$$

and **tree quality** calculate by plug the optimal output values into objective function

$$L^* = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j + \lambda} + \gamma T$$

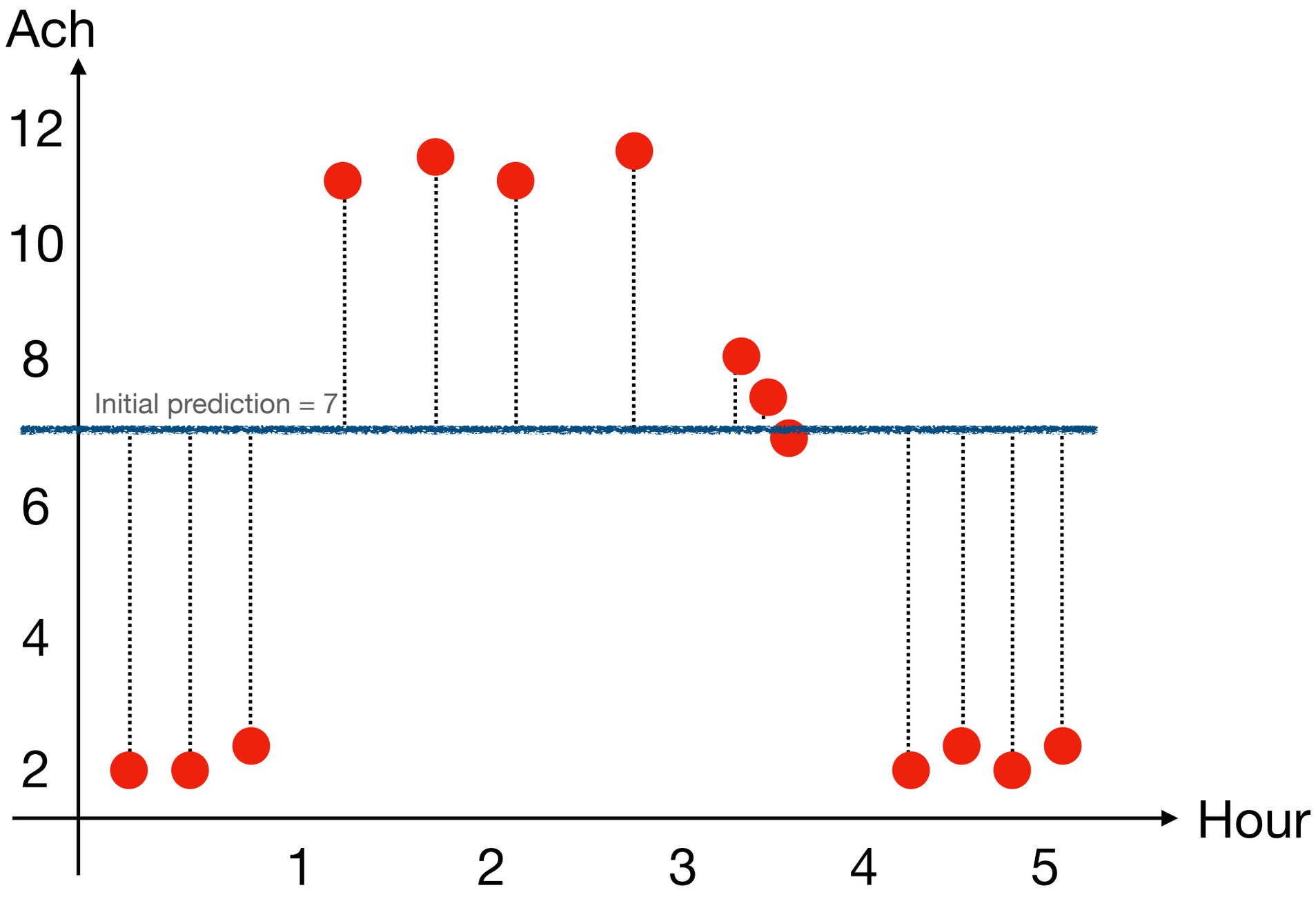
and define **Gain Score** that used to evaluate the splitting threshold

$$\text{Gain} = [\underbrace{\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda}}_{\text{Branch quality}} - \underbrace{\frac{(G_L + G_R)^2}{H_L + H_R + \lambda}}_{\text{Parent node quality}}] - \gamma$$

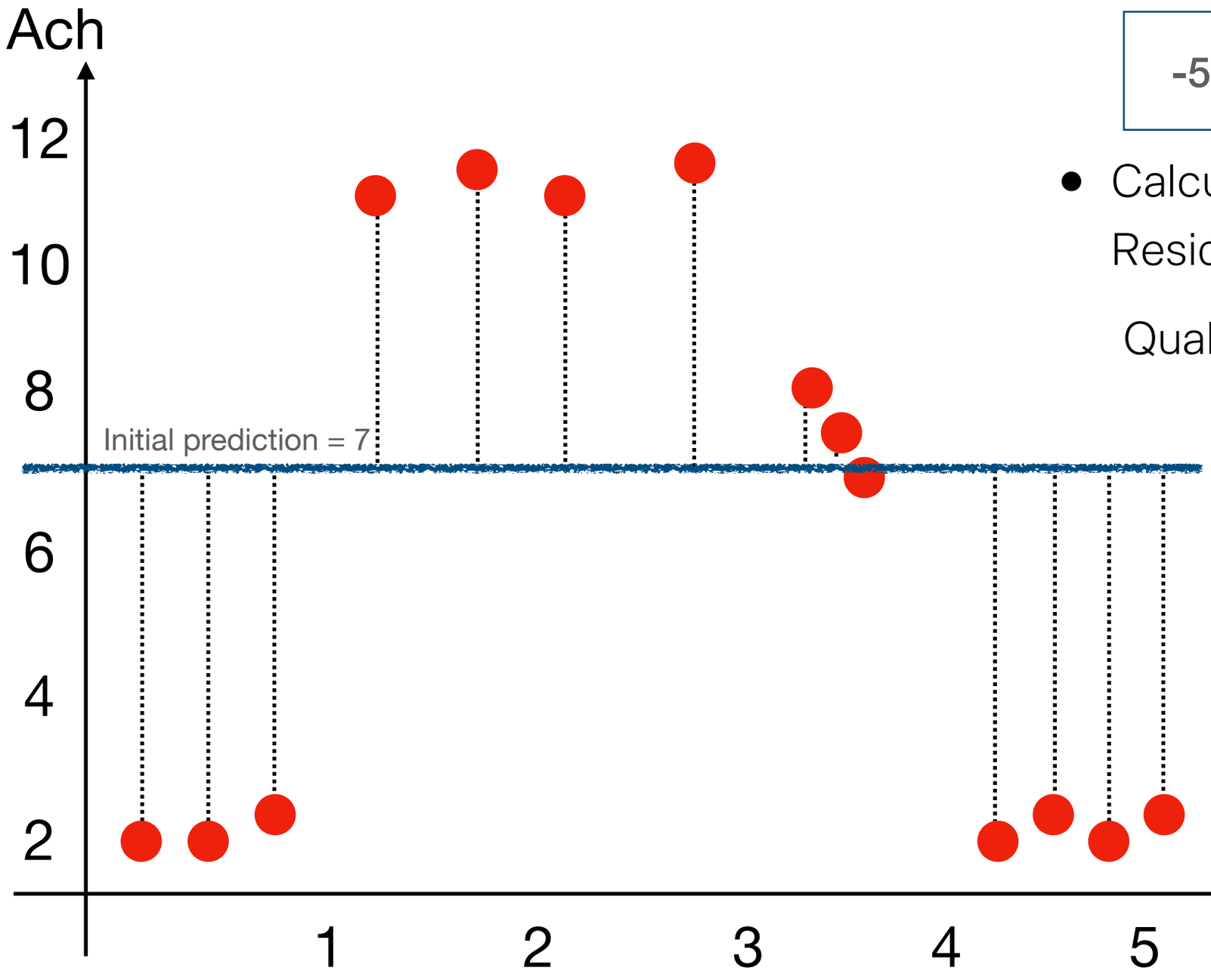
If gain < 0, we would better not to add that branch.

XGBoost for Regression

1. Make an initial prediction.
2. Calculate residuals
3. Fit a **regression Tree (called XGBoost Tree)** to the residuals.



XGBoost for Regression



1. Make an initial prediction.
2. Calculate residuals (e_i)
3. Fit a **regression Tree (called XGBoost Tree)** to the residuals.
    - Start with single leaf tree.

      -5.0, -5.0, -4.5, …

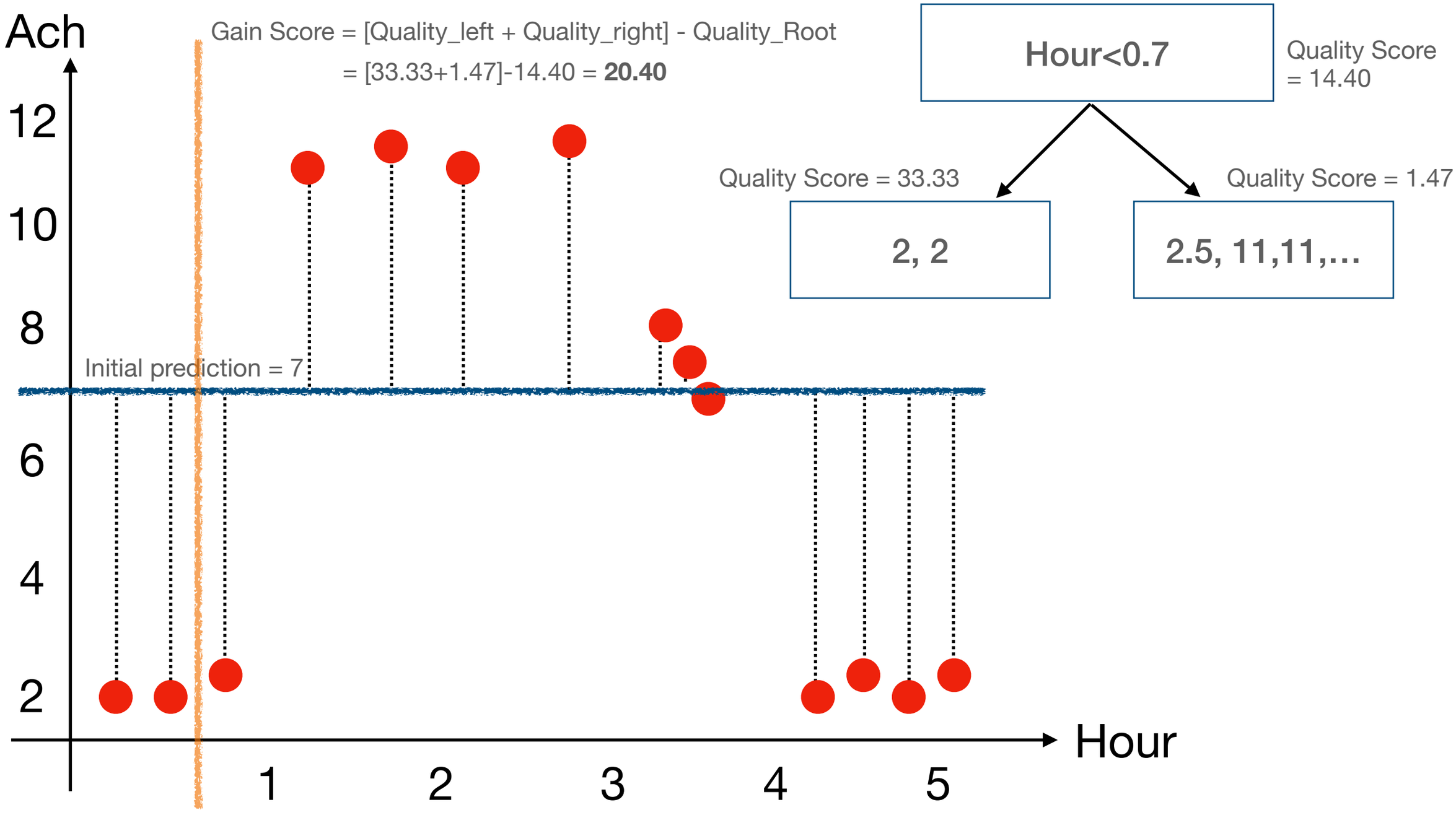
    - Calculate Quality Score for the Residuals

      Quality= $l_j^* = \dfrac{(\sum e)^2}{n_j + \lambda}$

      - **Regularization parameter**
      - If we let $\lambda = 0$, and $\gamma = 0$ the Quality score of the root node is **14.40**

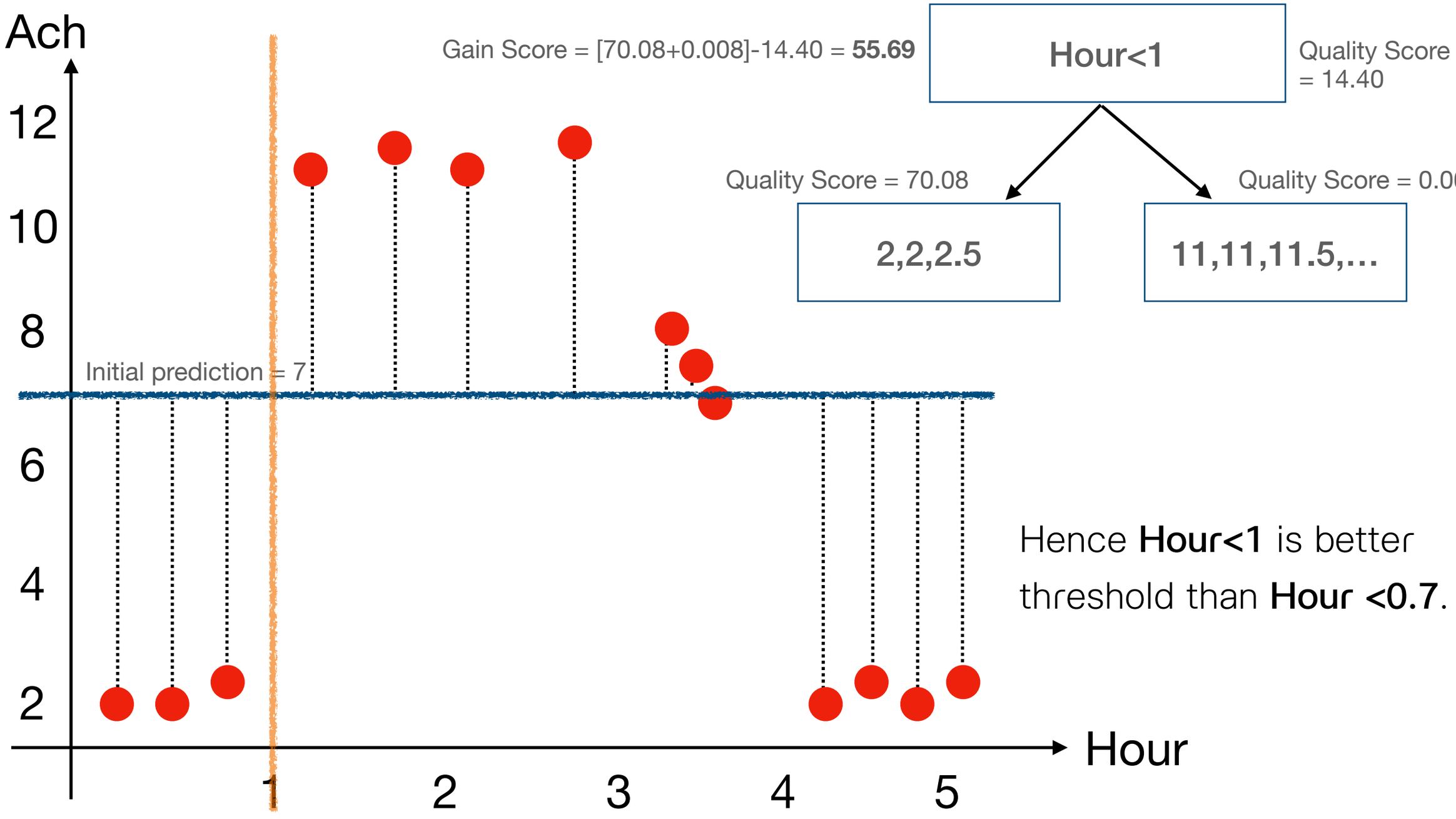
XGBoost for Regression

1. Make an initial prediction.
2. Calculate residuals (e_i)
3. Fit a **regression Tree (called XGBoost Tree)** to the residuals.
    - Start with single leaf tree.



XGBoost for Regression

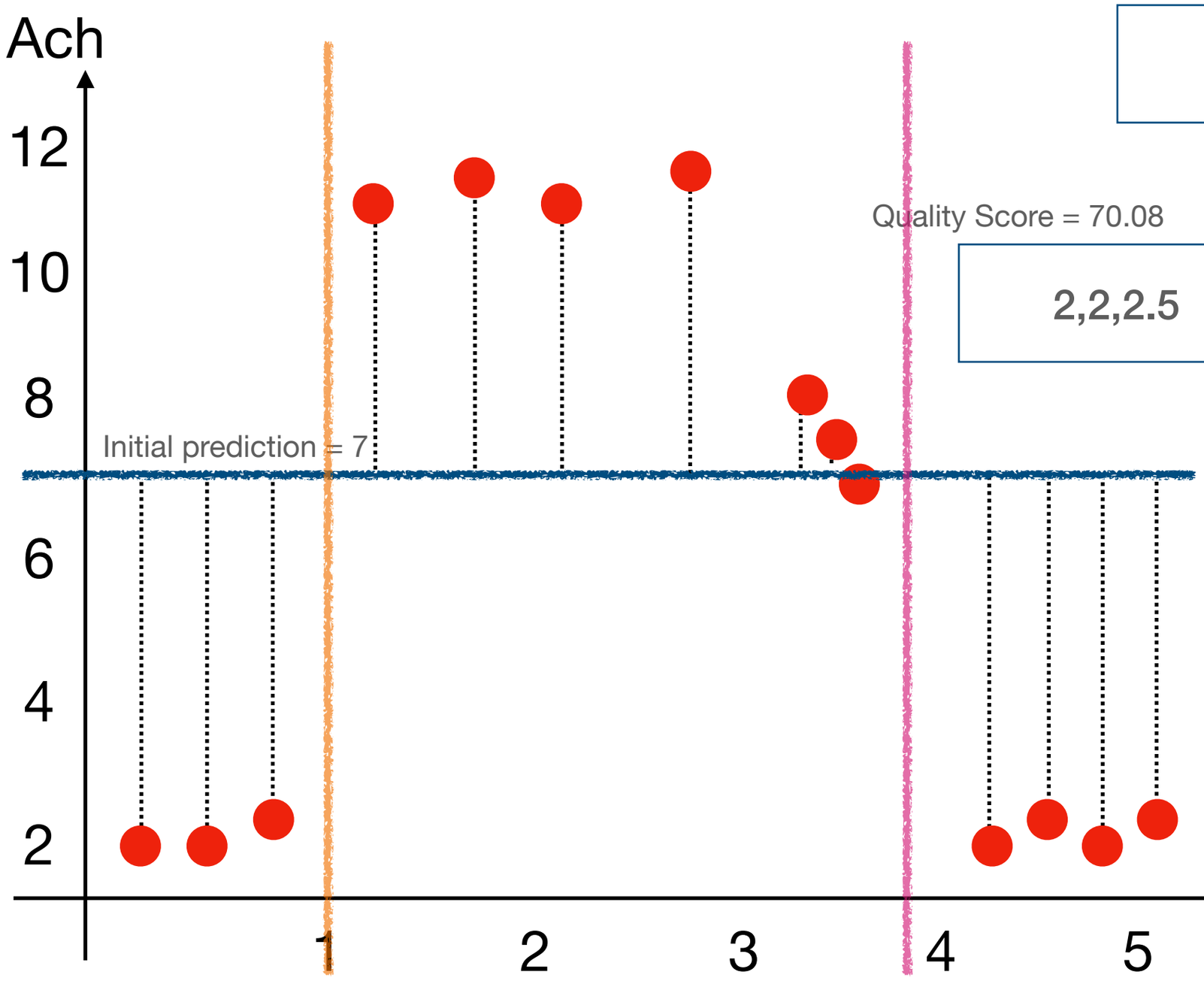
1. Make an initial prediction.
2. Calculate residuals (e_i)
3. Fit a **regression Tree (called XGBoost Tree)** to the residuals.
    - Start with single leaf tree.

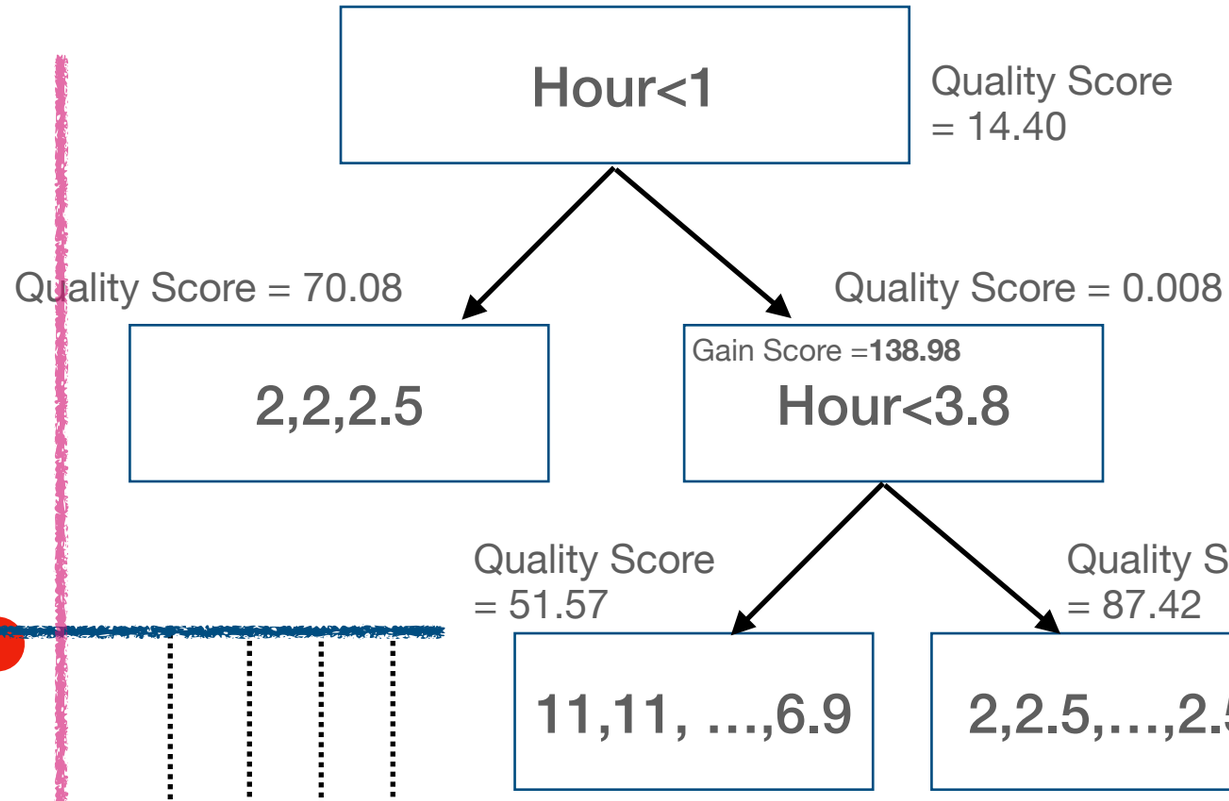


Hence **Hour<1** is better threshold than **Hour <0.7**.

XGBoost for Regression

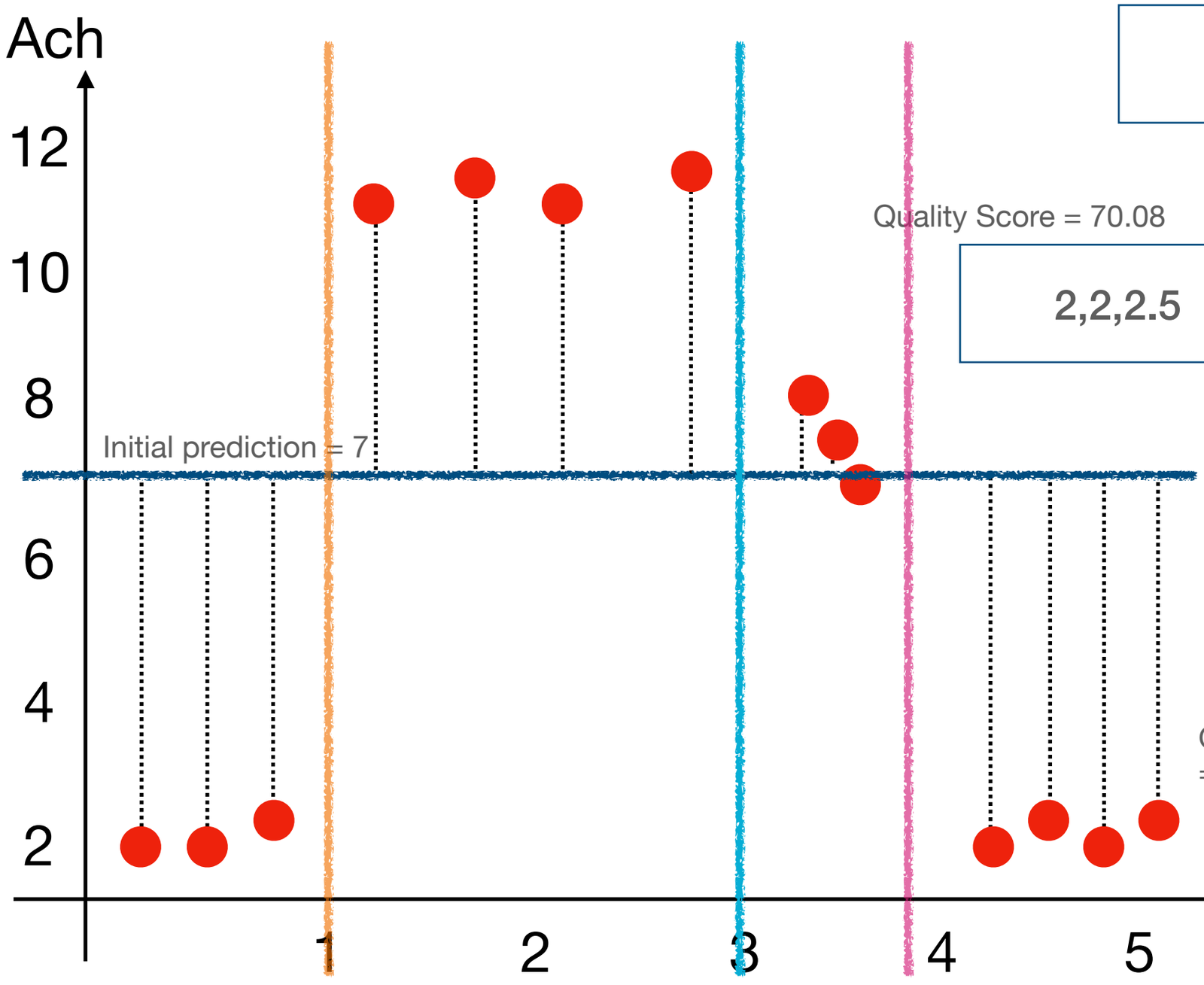
1. Make an initial prediction.
2. Calculate residuals (e_i)
3. Fit a **regression Tree (called XGBoost Tree)** to the residuals.
    - Start with single leaf tree.

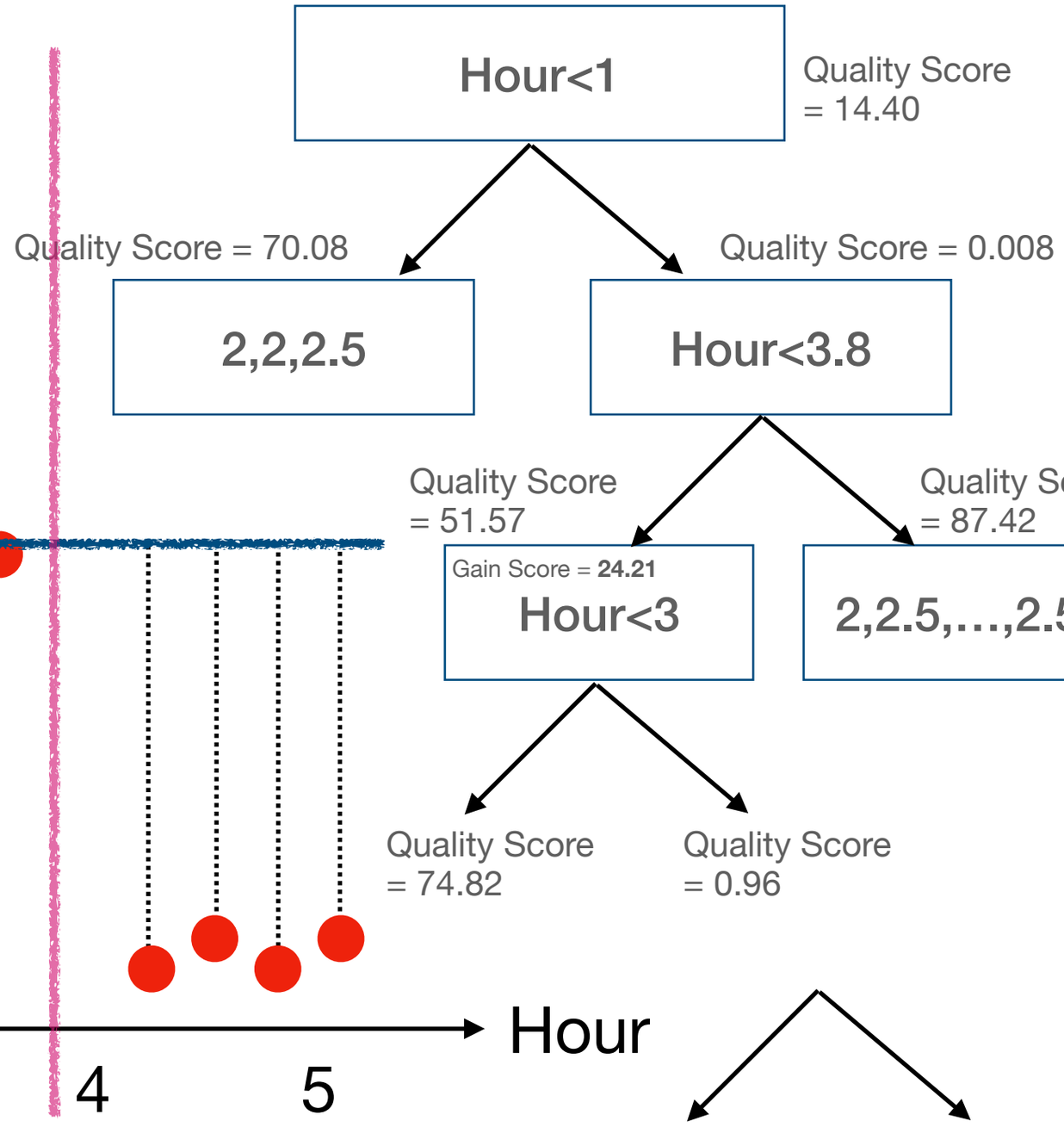




XGBoost for Regression

1. Make an initial prediction.
2. Calculate residuals (e_i)
3. Fit a **regression Tree (called XGBoost Tree)** to the residuals.
    - Start with single leaf tree.

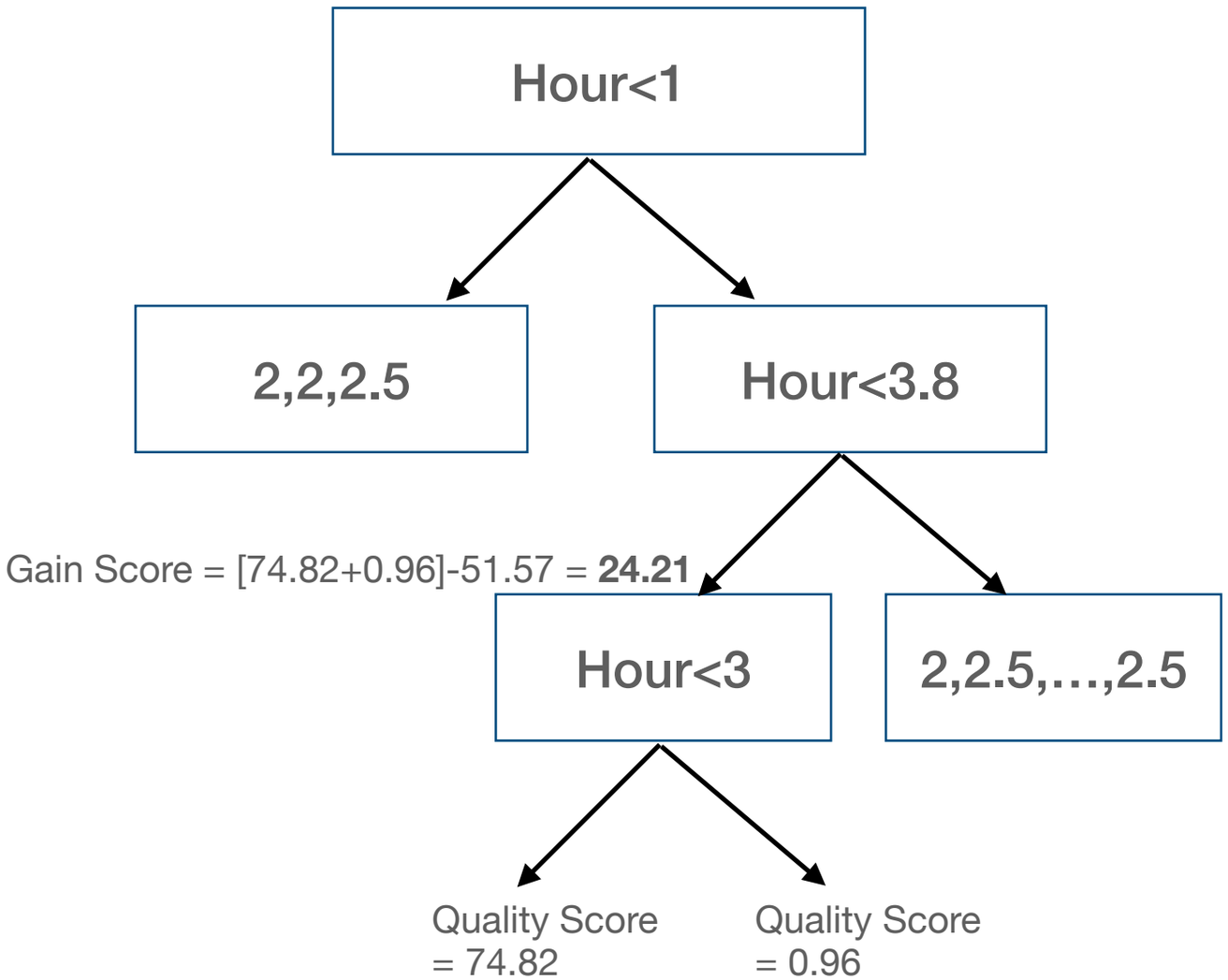




…

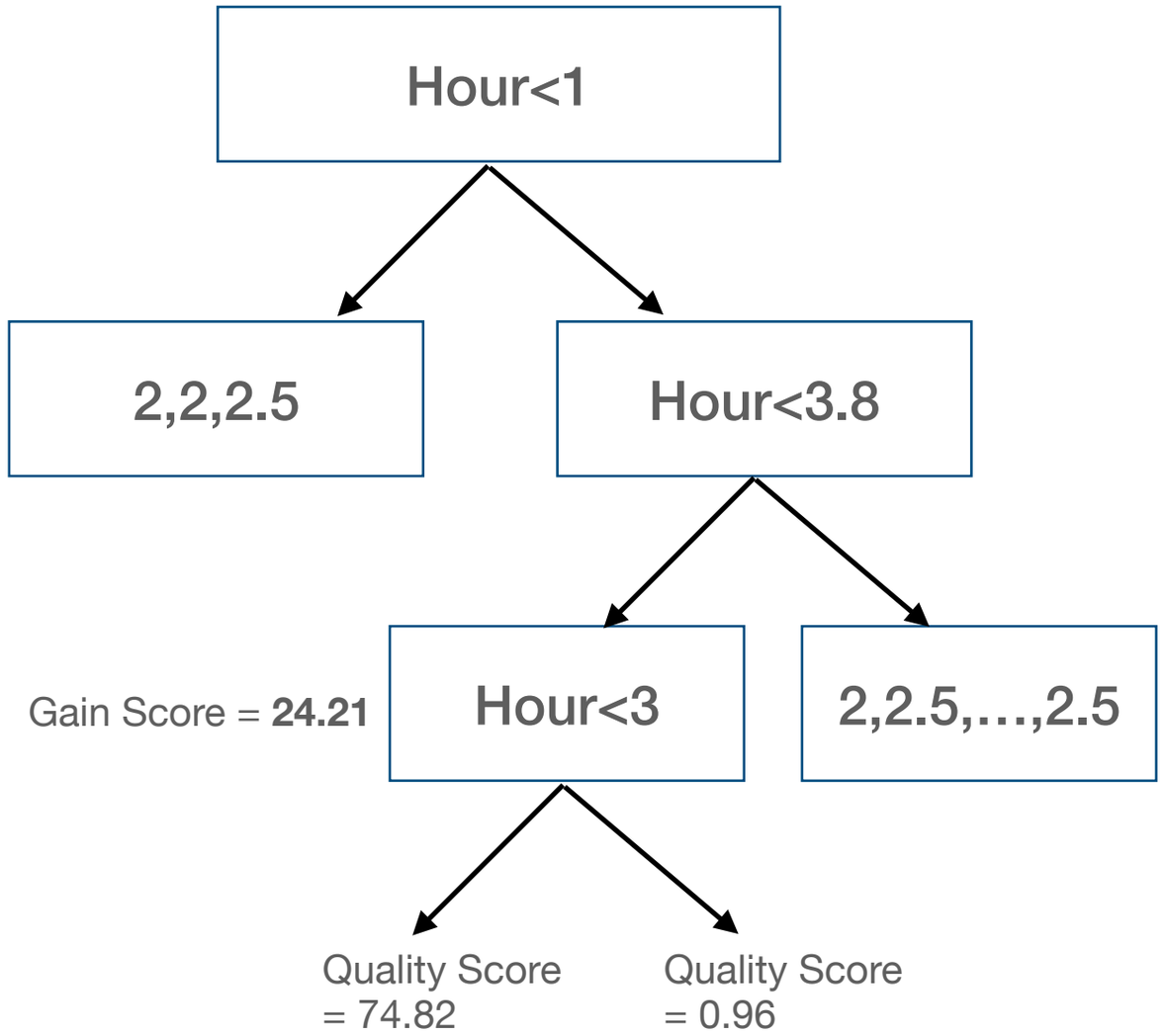
XGBoost for Regression

1. Make an initial prediction.
2. Calculate residuals (e_i)
3. Fit a **regression Tree (called XGBoost Tree)** to the residuals.
    - Start with single leaf tree.

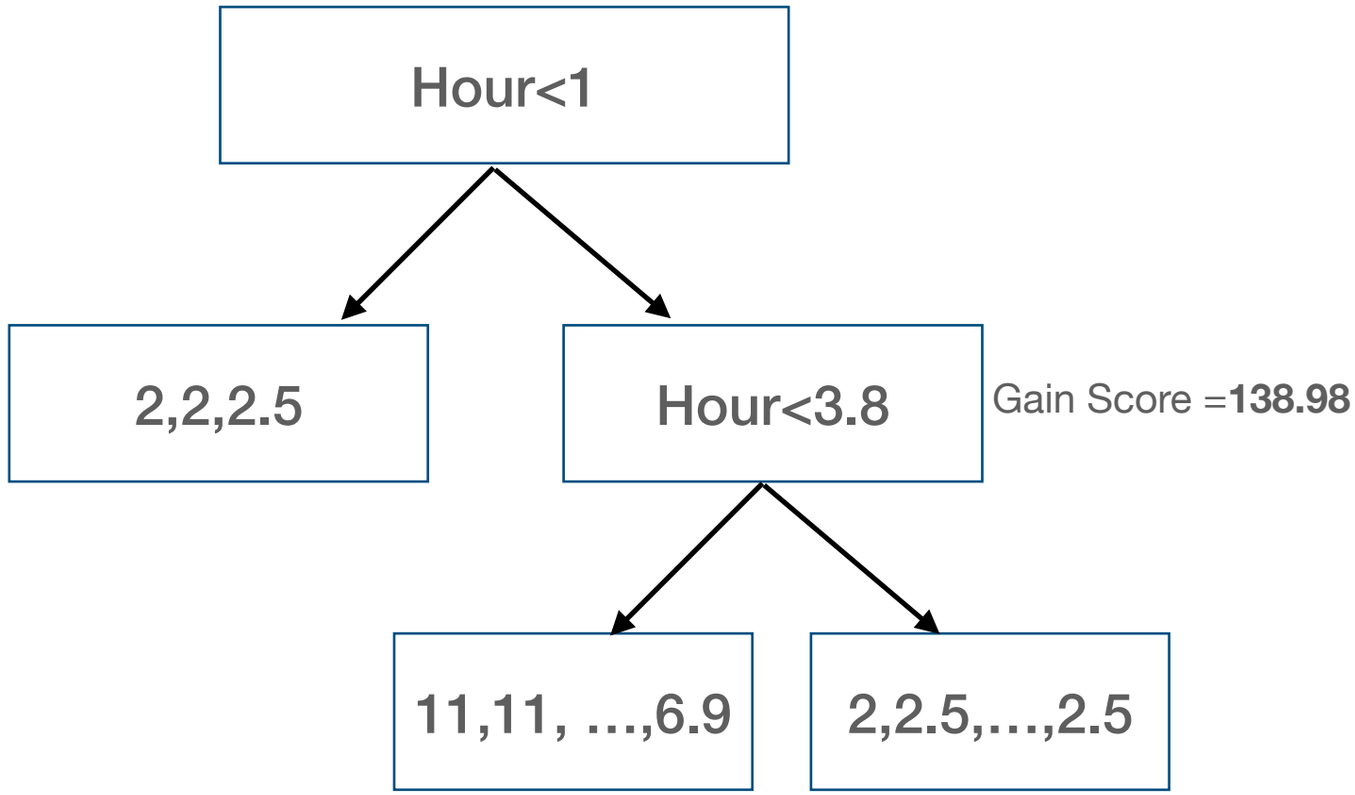


4. **Pruning a tree** - Calculate the difference value: **Gain** - γ
    - If **Gain** - γ < 0 we will remove the branch.
    - EX. Let $\gamma = 100$

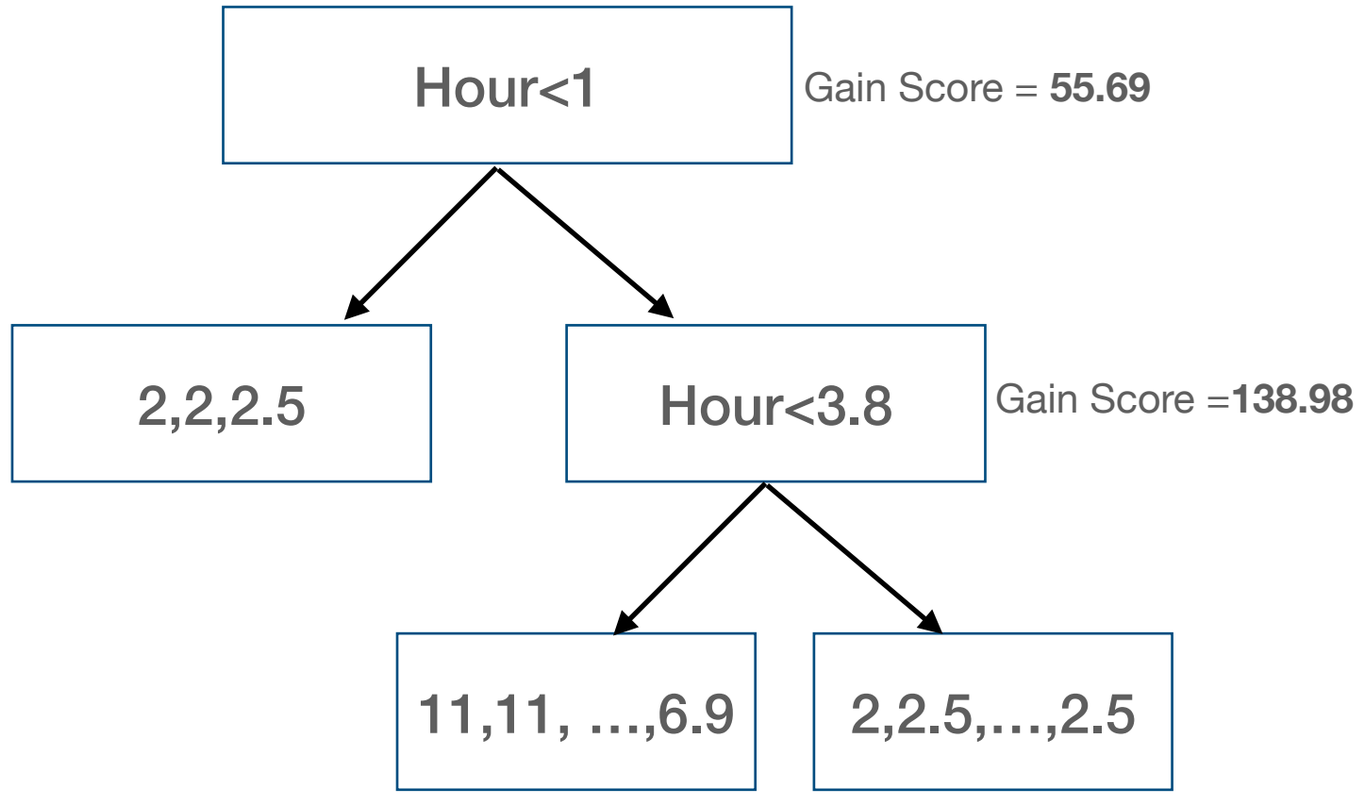
4. **Pruning a tree** - Calculate the difference value: **Gain - γ**
    - If **Gain - γ** < 0 we will remove the branch.
    - EX. Let $\gamma = 100$



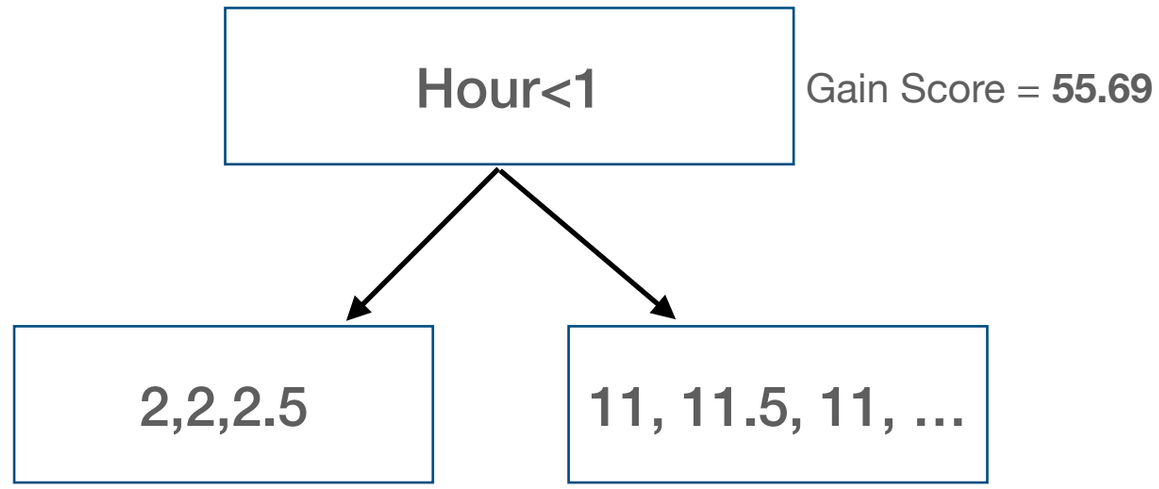
4. **Pruning a tree** - Calculate the difference value: **Gain - γ**
    - If **Gain - γ** < 0 we will remove the branch.
    - EX. Let $\gamma = 100$



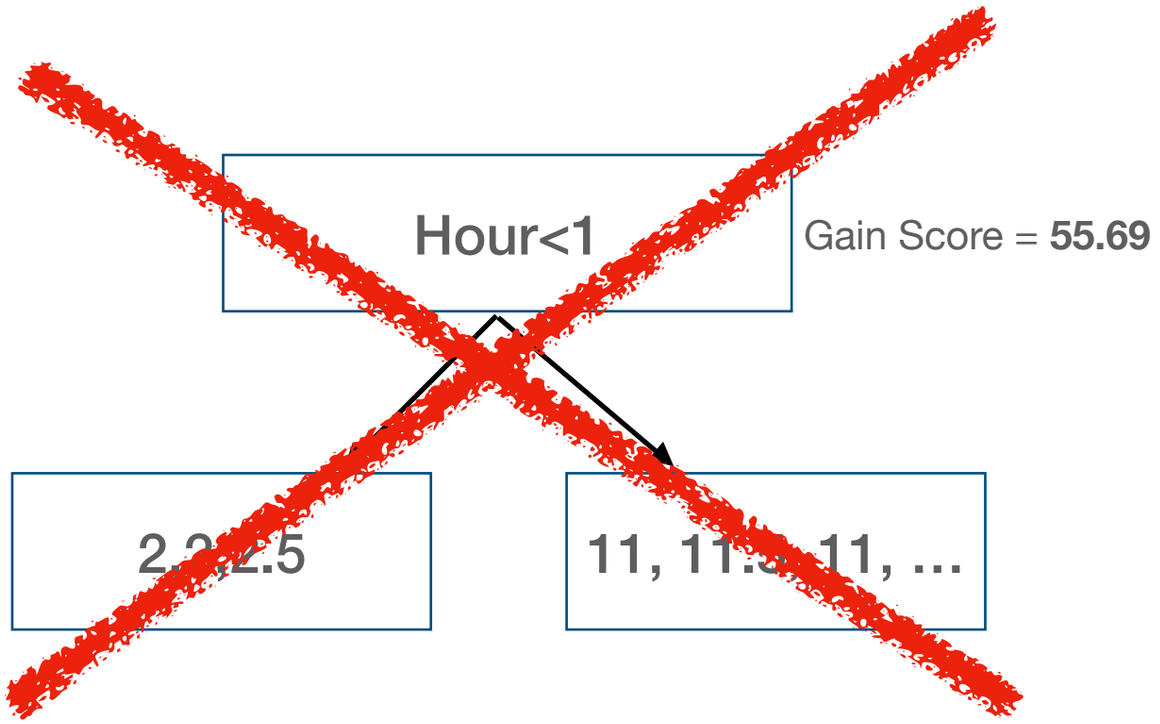
4. **Pruning a tree** - Calculate the difference value: **Gain** - γ
    - If **Gain** - γ < 0 we will remove the branch.
    - EX. Let $\gamma = 100$



4. **Pruning a tree** - Calculate the difference value: **Gain** - γ
    - If **Gain** - γ < 0 we will remove the branch.
    - EX. If we let $\gamma = 140$



4. **Pruning a tree** - Calculate the difference value: **Gain** - γ
    - If **Gain** - γ < 0 we will remove the branch.
    - EX. If we let $\gamma = 140$



4. **Pruning a tree** - Calculate the difference value: **Gain - γ**
    - If **Gain - γ** < 0 we will remove the branch.
    - EX. If we let $\gamma = 140$

Initial predicted value =7

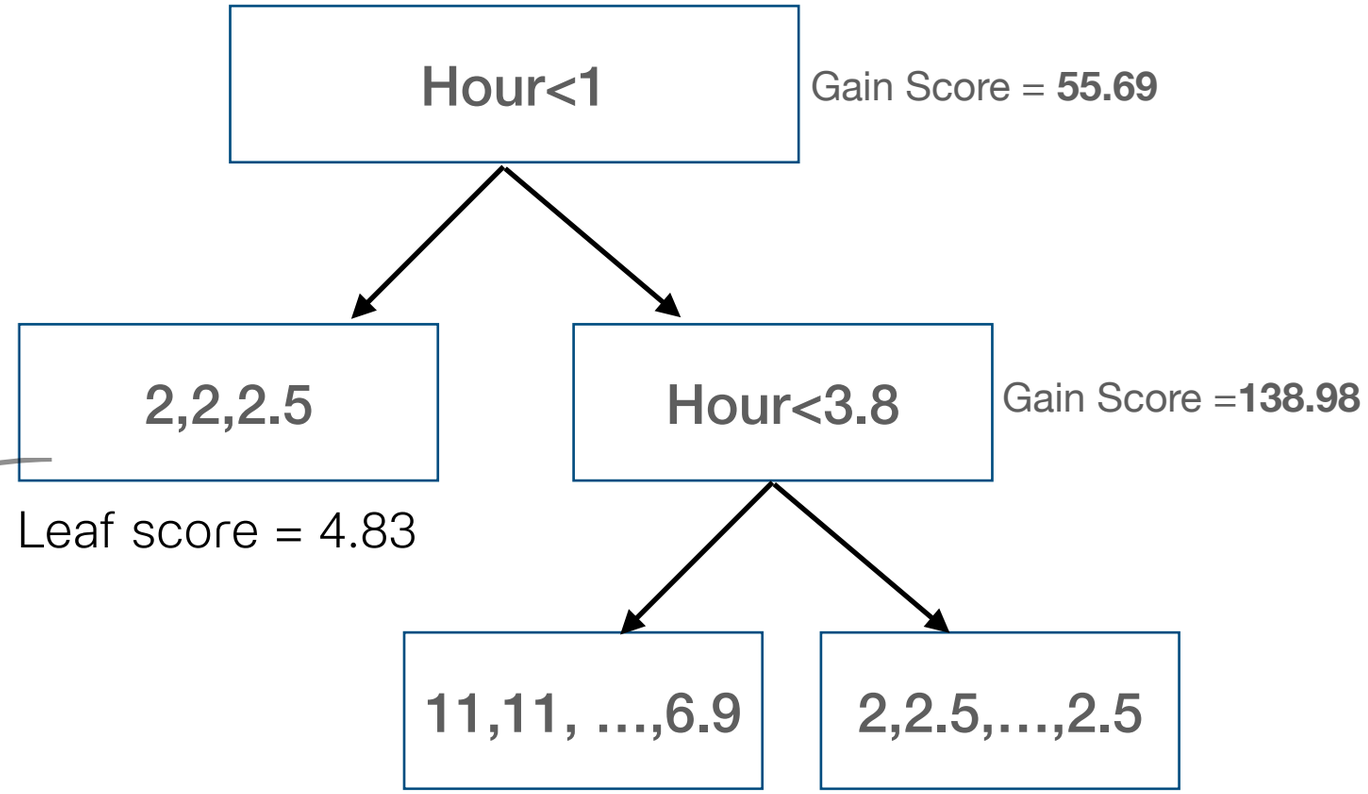
4. **Pruning a tree** - Calculate the difference value: **Gain - γ**
   - If **Gain - γ** < 0 we will remove the branch.
   - EX. Let $\gamma = 100$

5. **Calculate the output values**

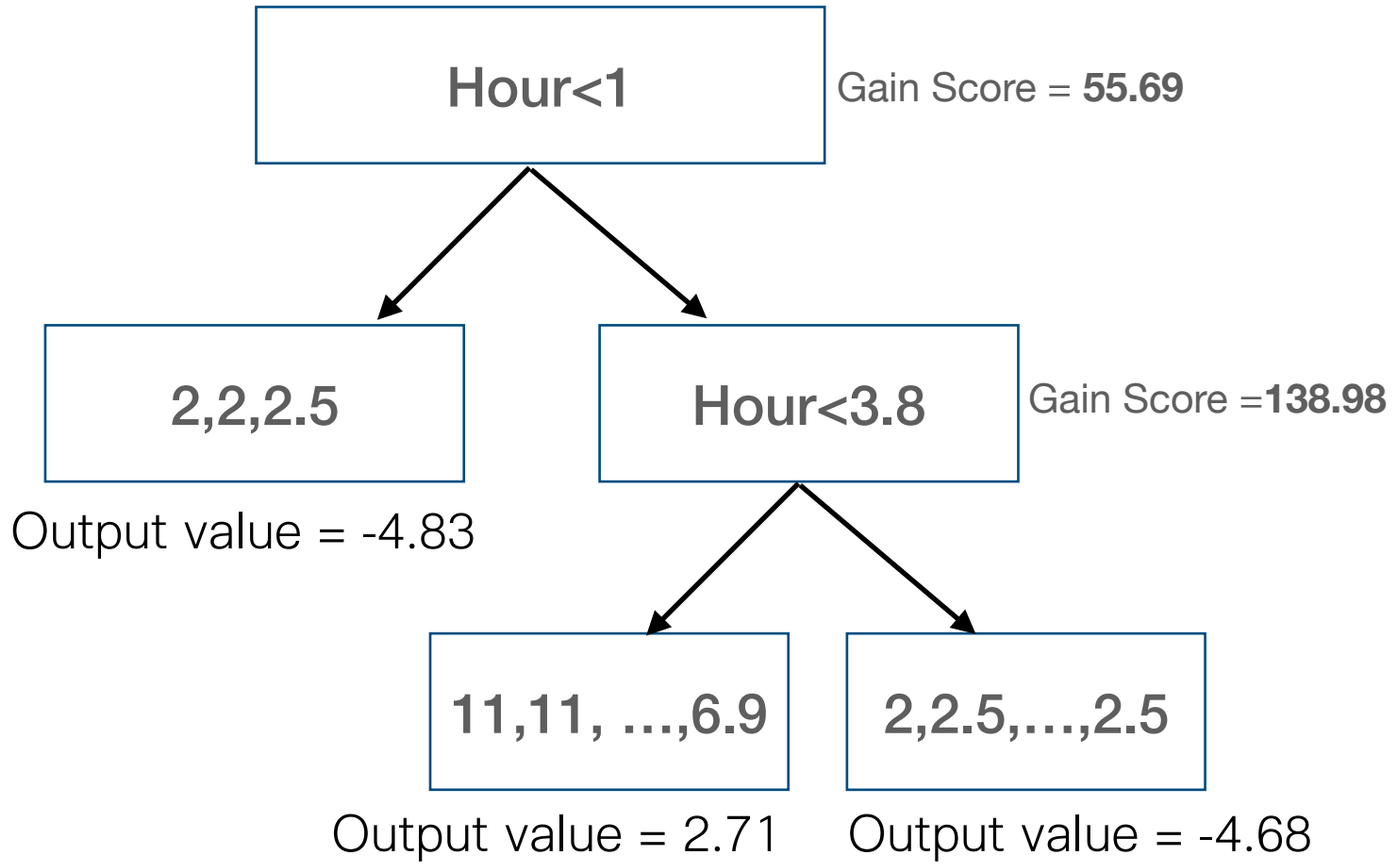
Output value = $\omega_j^* = -\frac{\sum e}{n + \lambda}$

Output value = $\frac{\sum e}{n + 0} = 4.83$

Output value = $\frac{\sum e}{n + 1} = -3.63$



4. **Pruning a tree** - Calculate the difference value: **Gain - γ**
   - If **Gain - γ** < 0 we will remove the branch.
   - EX. Let $\gamma = 100$



5. **Calculate the output values**

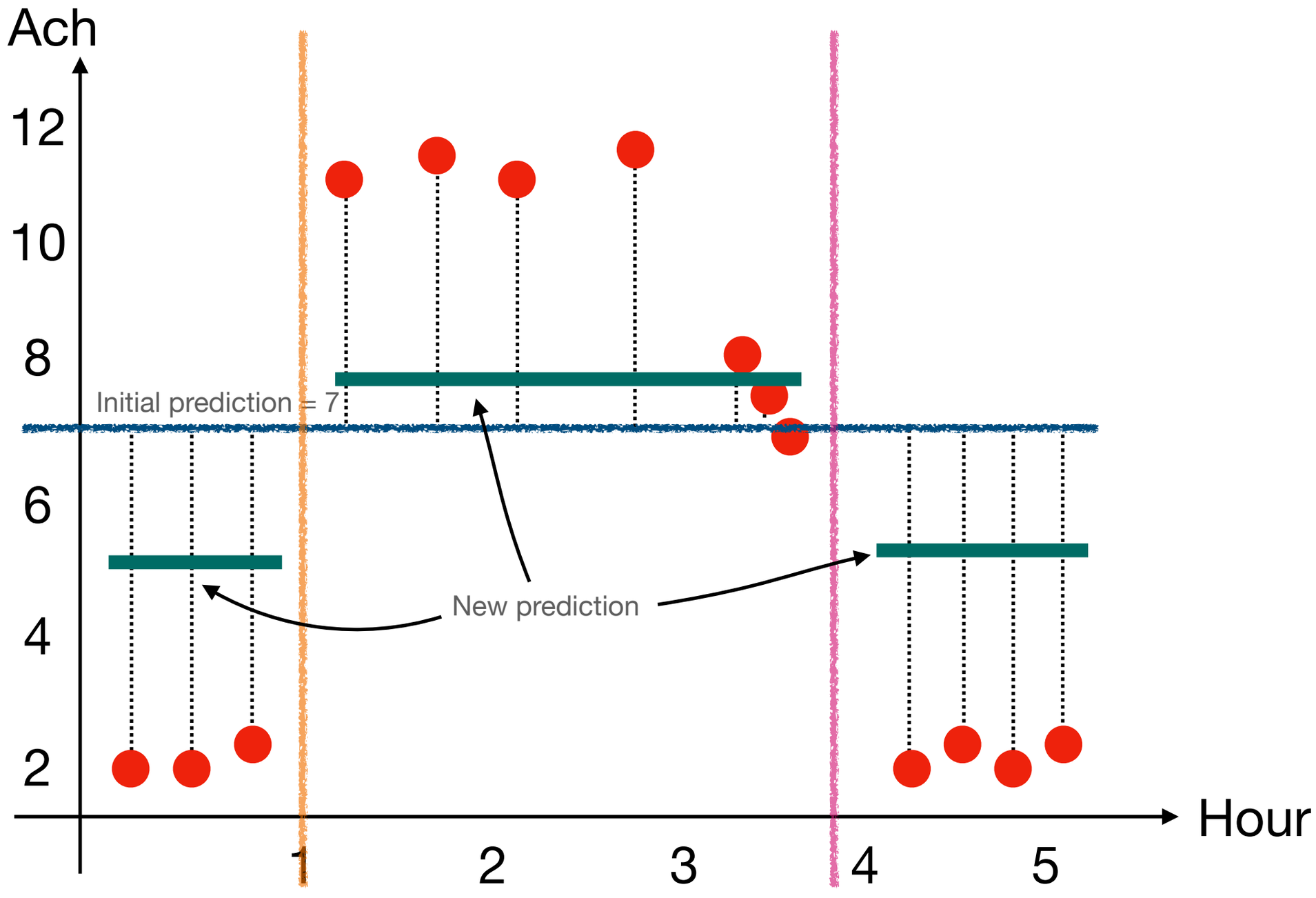
$$\text{Output value} = \frac{\sum e}{n + \lambda}$$

6. **Make a new prediction**

New prediction = initial prediction + ϵ x (output value)

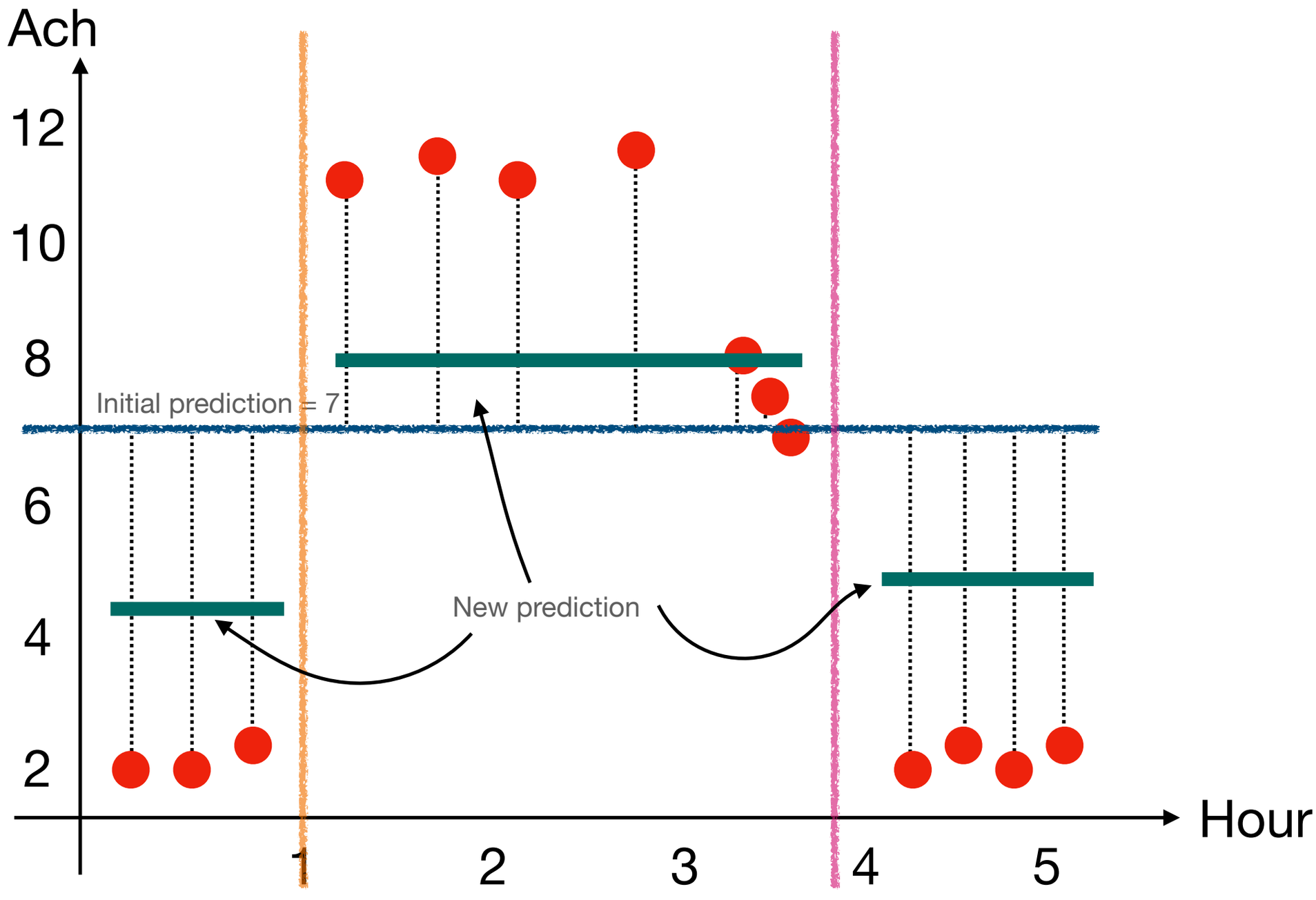
Learning Rate = [0,1] (EX. $\epsilon = 0.3$)

- For Hour = 0.5 ——> the new predicted value = 7 + 0.3 x (4.83) = 5.55
- For Hour = 2 ——> the new predicted value = 7 + 0.3 x (2.71) = 7.813
- For Hour = 5 ——-> the new predicted value = 7 + 0.3 x (4.68) = 5.60


Ach
12
10
8
6
4
2
Initial prediction = 7
New prediction
1
2
3
4
5
Hour

7. **Calculate new residual and repeat 3 to 6**

New prediction = initial prediction + ϵ x (output value1) + ϵ x (output value2)



7. **Calculate new residual and repeat 3 to 6**

New prediction = initial prediction + ϵ x (output value1) + ϵ x (output value2)+ ϵ x (output value3)+….