AI VIET NAM
@aivietnam.edu.vn

Module 03 – Exercise Class

Random Forest
AdaBoost - Gradient Boosting

Nguyen Quoc Thai

Objectives

Ensemble Learning

- ❖ Introduction
- ❖ Ensemble Methods
- ❖ Learning Ensembles
- ❖ Constructing Ensembles

Boosting

- ❖ Boosting Methods
- ❖ AdaBoost
- ❖ Gradient Boosting
- ❖ Calculate Weight

Bagging

- ❖ Bootstrapping
- ❖ Decision Tree
- ❖ Random Forest
- ❖ Extract Subset Training Data

Implementation

- ❖ Housing Dataset
- ❖ Random Forest
- ❖ AdaBoost
- ❖ Gradient Boosting
- ❖ Sklearn

Outline

SECTION 1
Ensemble Learning

SECTION 2
Bagging Methods

SECTION 3
Boosting Methods

SECTION 4
Implementation

Ensemble Learning

Decision Tree

If the result from 1 tree is not good…

Why don’t we just use more trees?

Ensemble Learning

Example

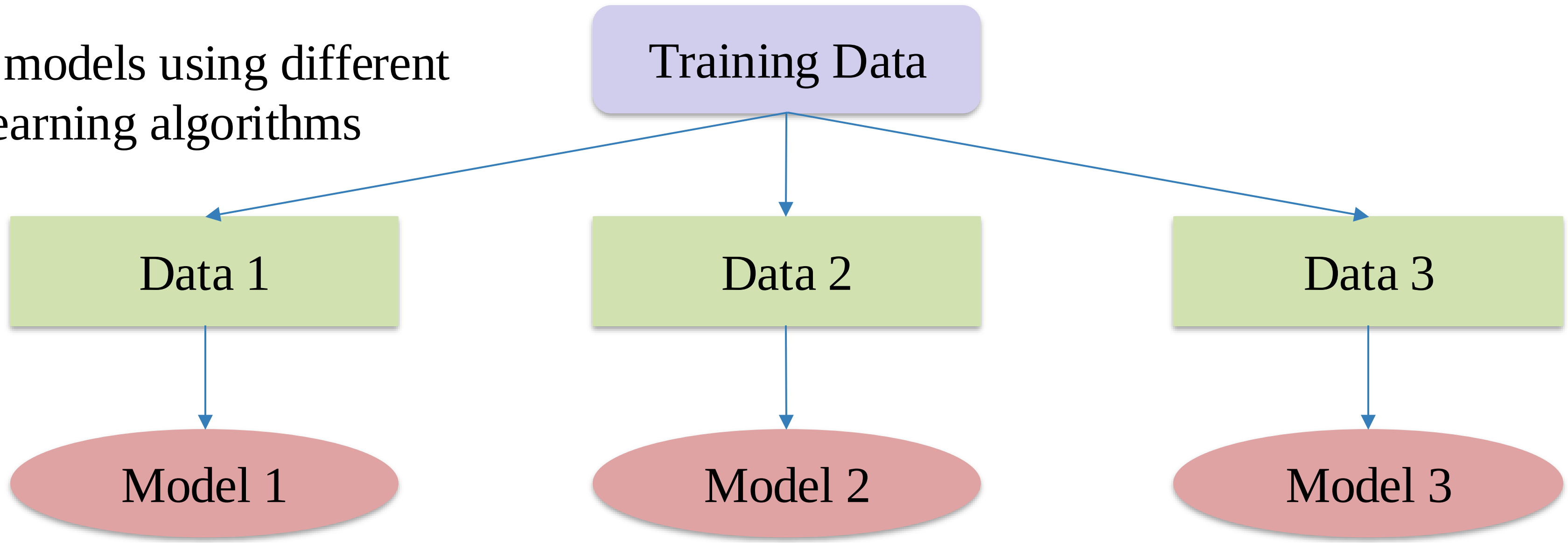
Accuracy: 100%

Ground Truth	Predict 1	Predict 2	Predict 3	Combine
0	0	0	0	0
0	0	1	0	0
0	0	0	1	0
0	1	0	0	0
1	1	0	1	1
1	1	1	1	1
1	1	1	1	1
1	0	1	1	1

Ensemble Learning

Learning Ensembles

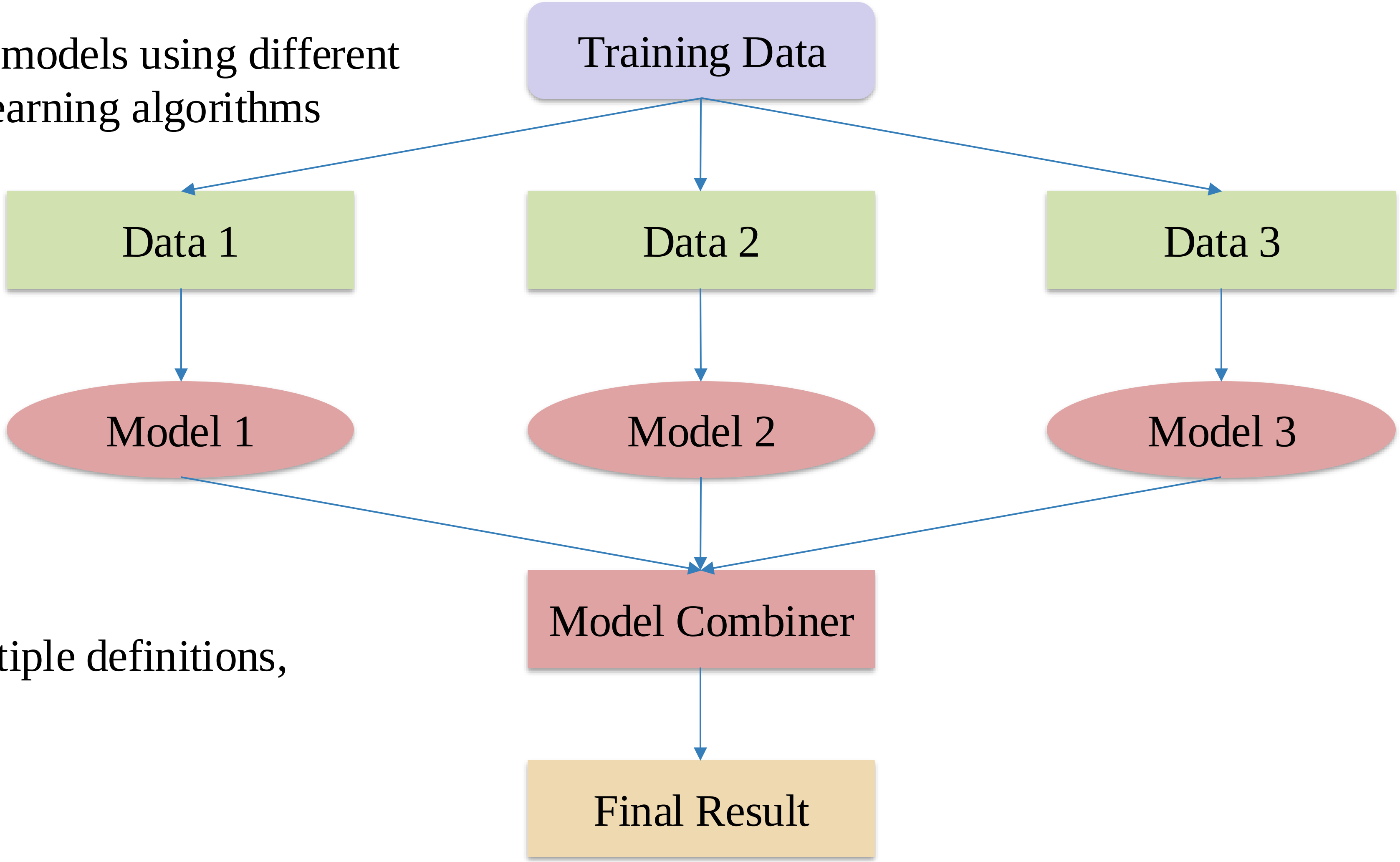
❖ Learn multiple alternative models using different training data or different learning algorithms



Ensemble Learning

Learning Ensembles

- Learn multiple alternative models using different training data or different learning algorithms



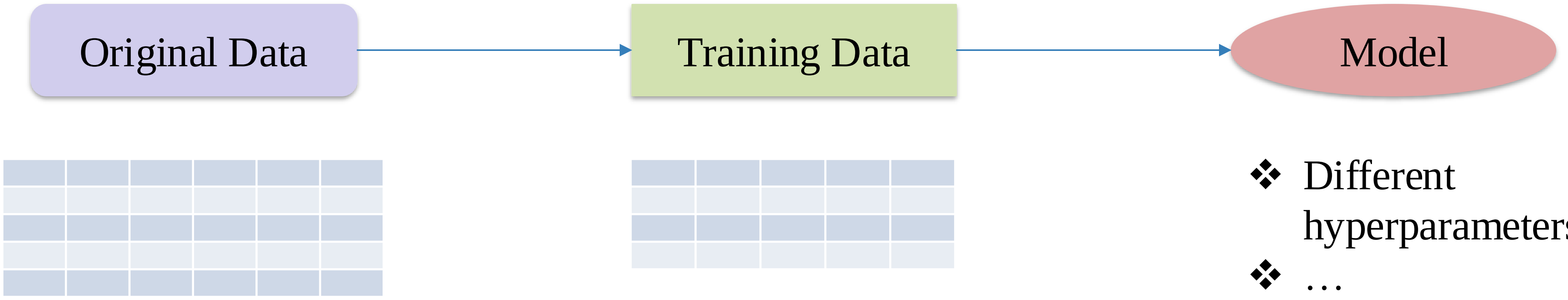
- Combine decisions of multiple definitions,
  e.g weighted voting

Ensemble Learning

Methods for Constructing Ensembles

- By manipulating the training set
- By manipulating the input features
- By manipulating the class labels
- By manipulating the learning algorithm

- KNN
- Decision Tree
- …



- Different hyperparameters
- …

Ensemble Learning

Homogeneous Ensembles

❖ Use a single, arbitrary learning algorithm but manipulate training data to make it learn multiple models

Ensemble Learning

- Bagging
  - ❖ Resample data
  - ❖ Useful to reduce the variance
  - ❖ Random Forest
- Boosting
  - ❖ Re-weight data
  - ❖ Useful to reduce the bias
  - ❖ AdaBoost
  - ❖ Gradient Boosting

AI VIET NAM
@aivietnam.edu.vn

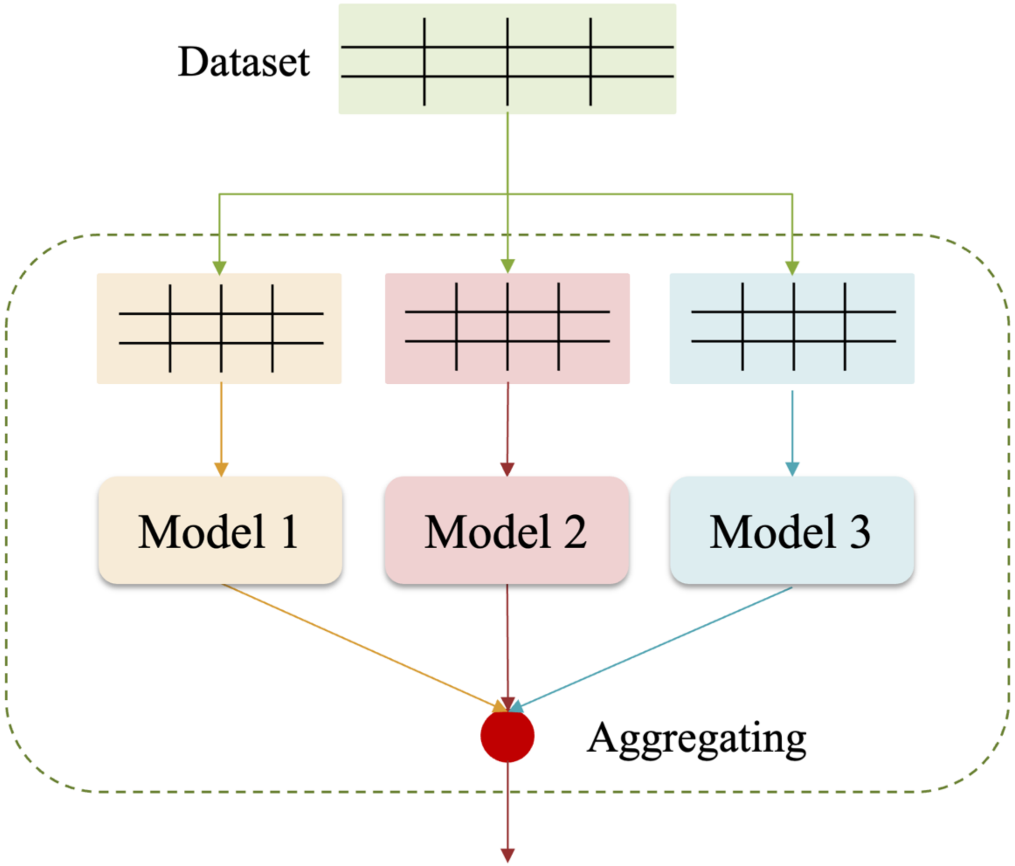
Outline

SECTION 1
Ensemble Learning

SECTION 2
Bagging Methods

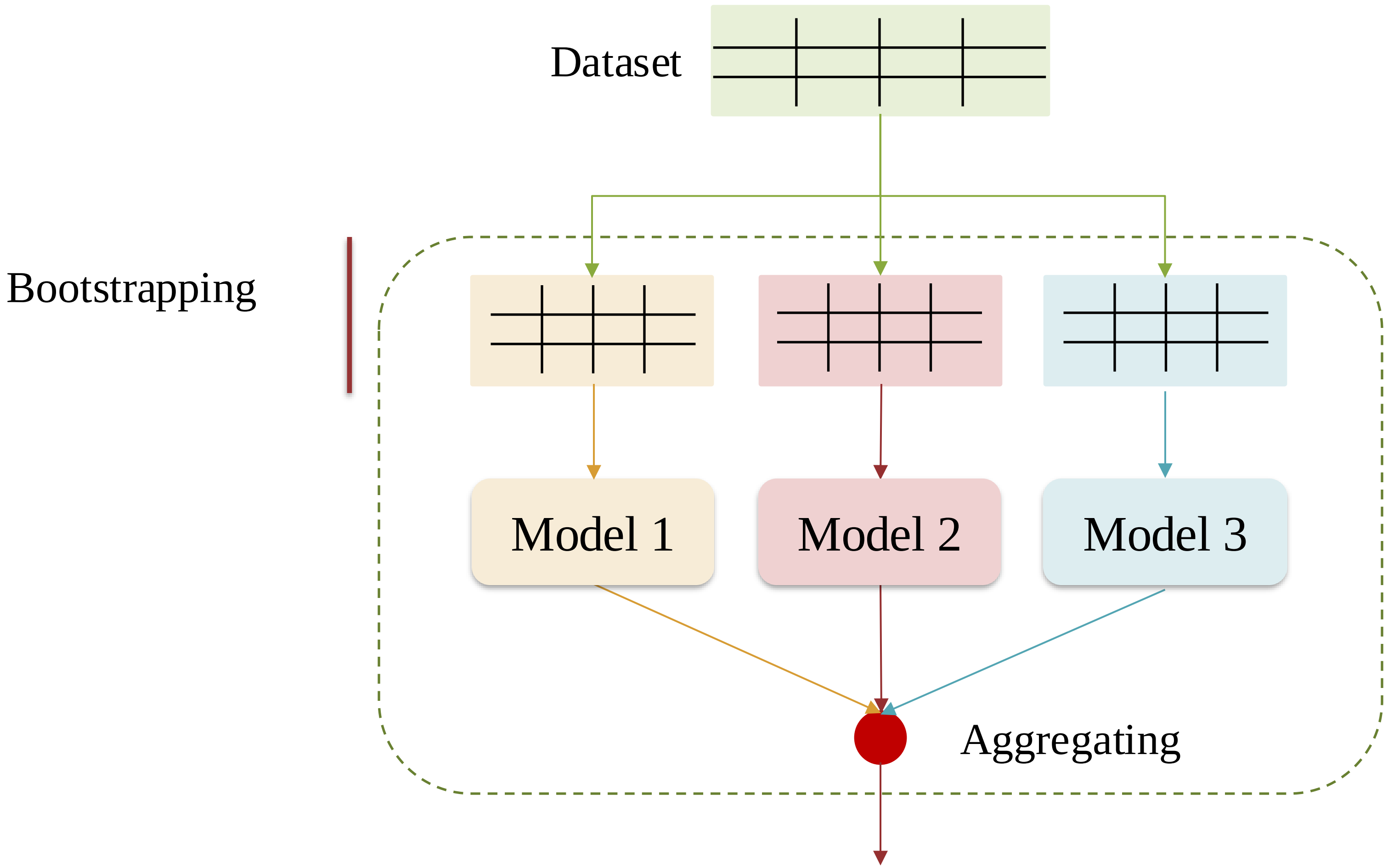
SECTION 3
Boosting Methods

SECTION 4
Implementation



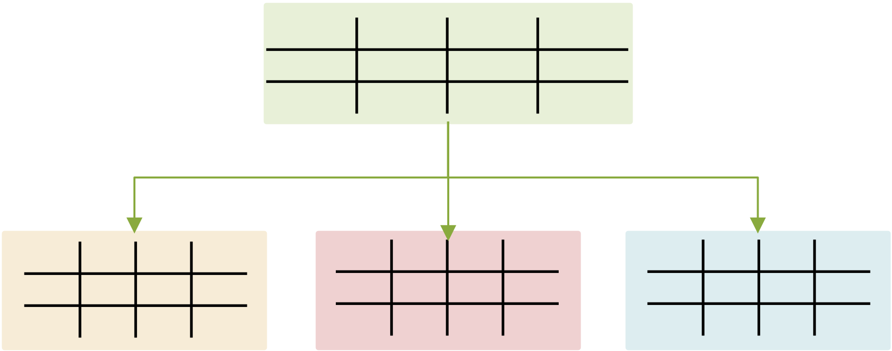
Random Forest

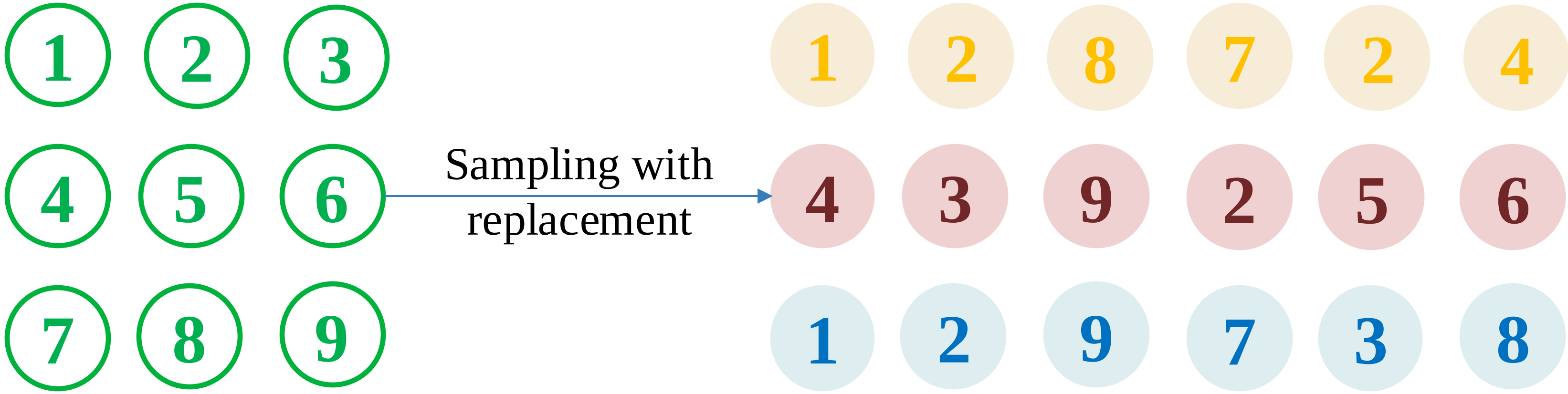
Bagging (Bootstrapping Aggregating)



Random Forest

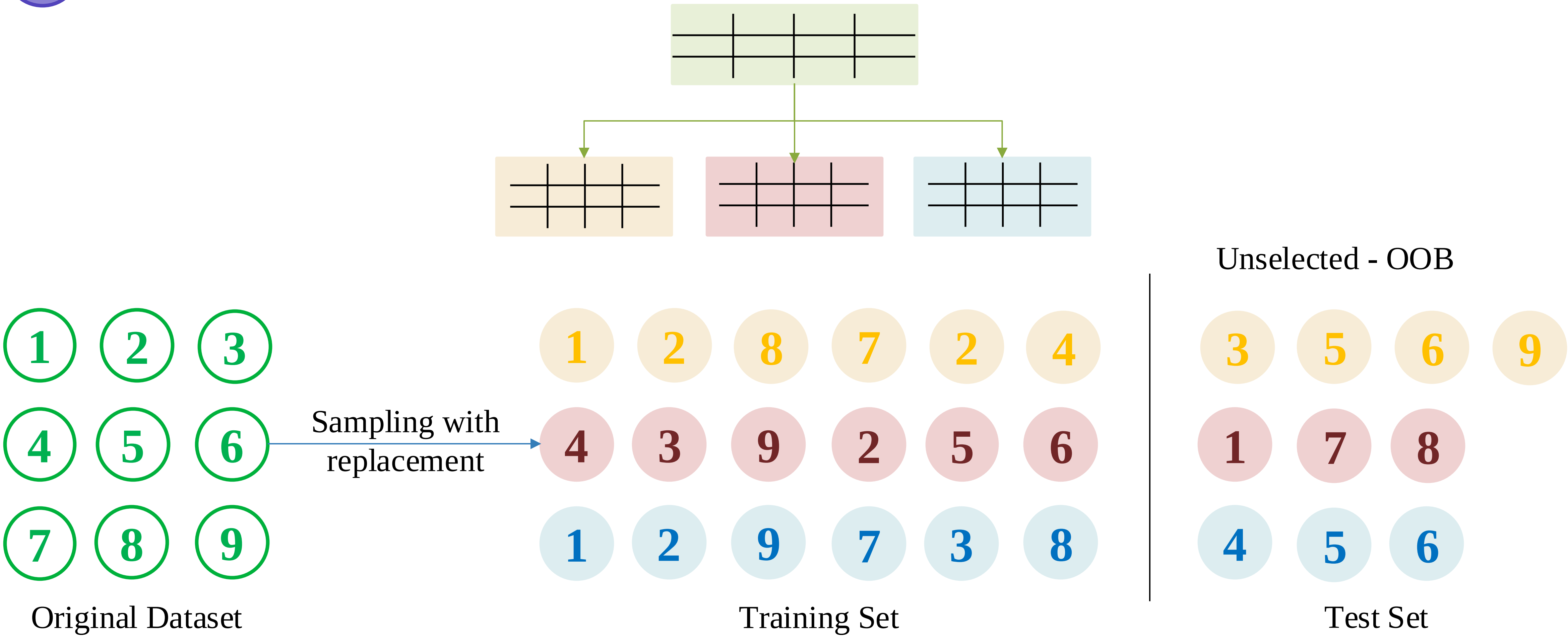
Bootstrapping





Random Forest

Out-of-bag (OOB)

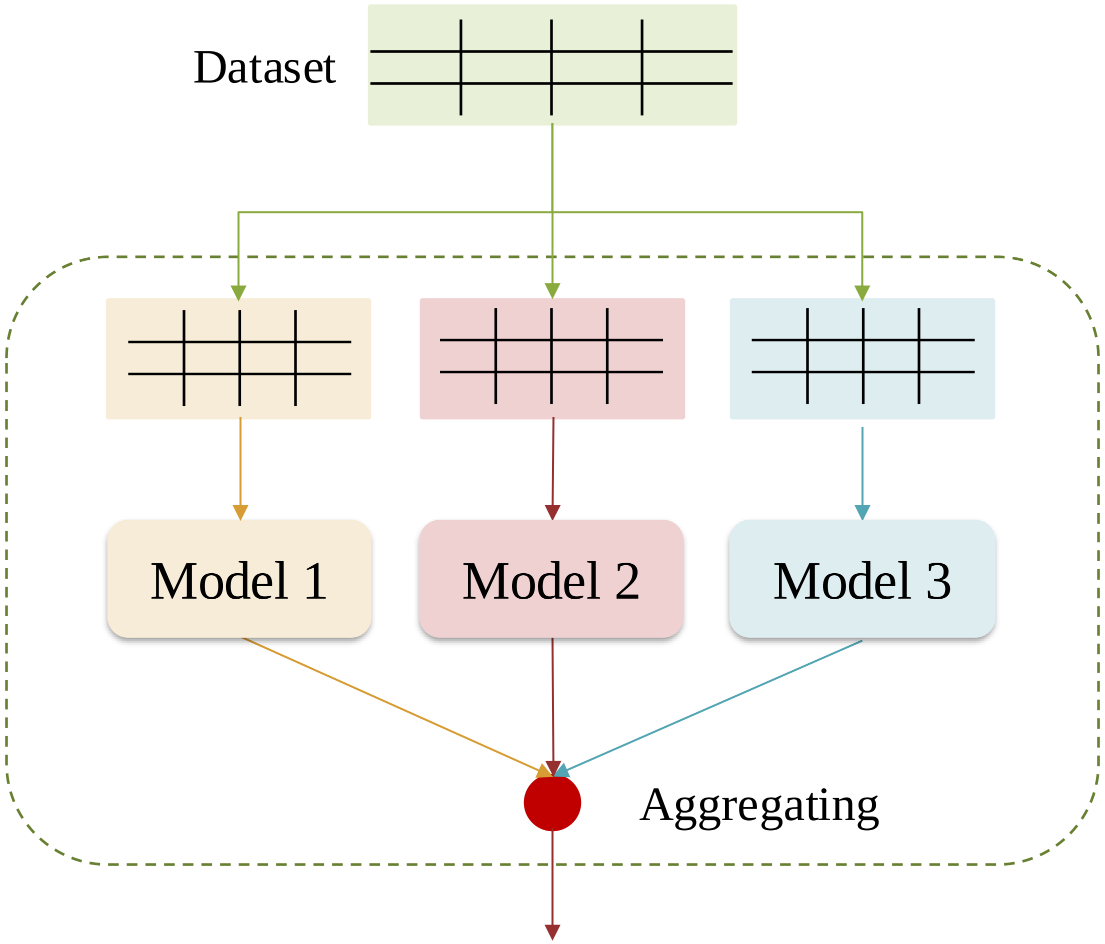


Random Forest

Aggregating

Bootstrap Dataset

Evaluation: Test Set



Voting

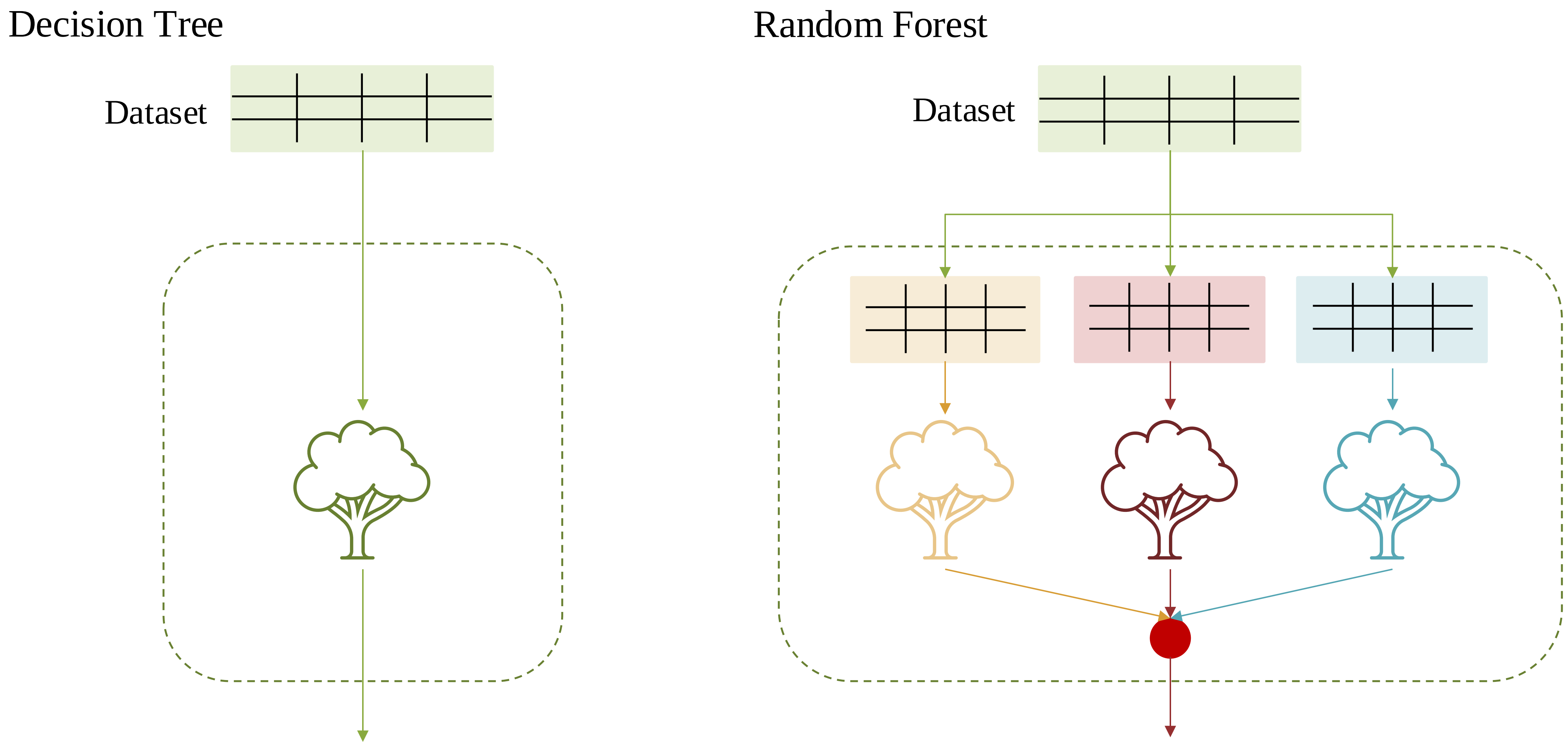
0	1	1	**1**
0	0	1	**0**
1	1	1	**1**

Averaging

1	2	3	**2**
1	3	2	**2**
1	1	1	**1**

Random Forest

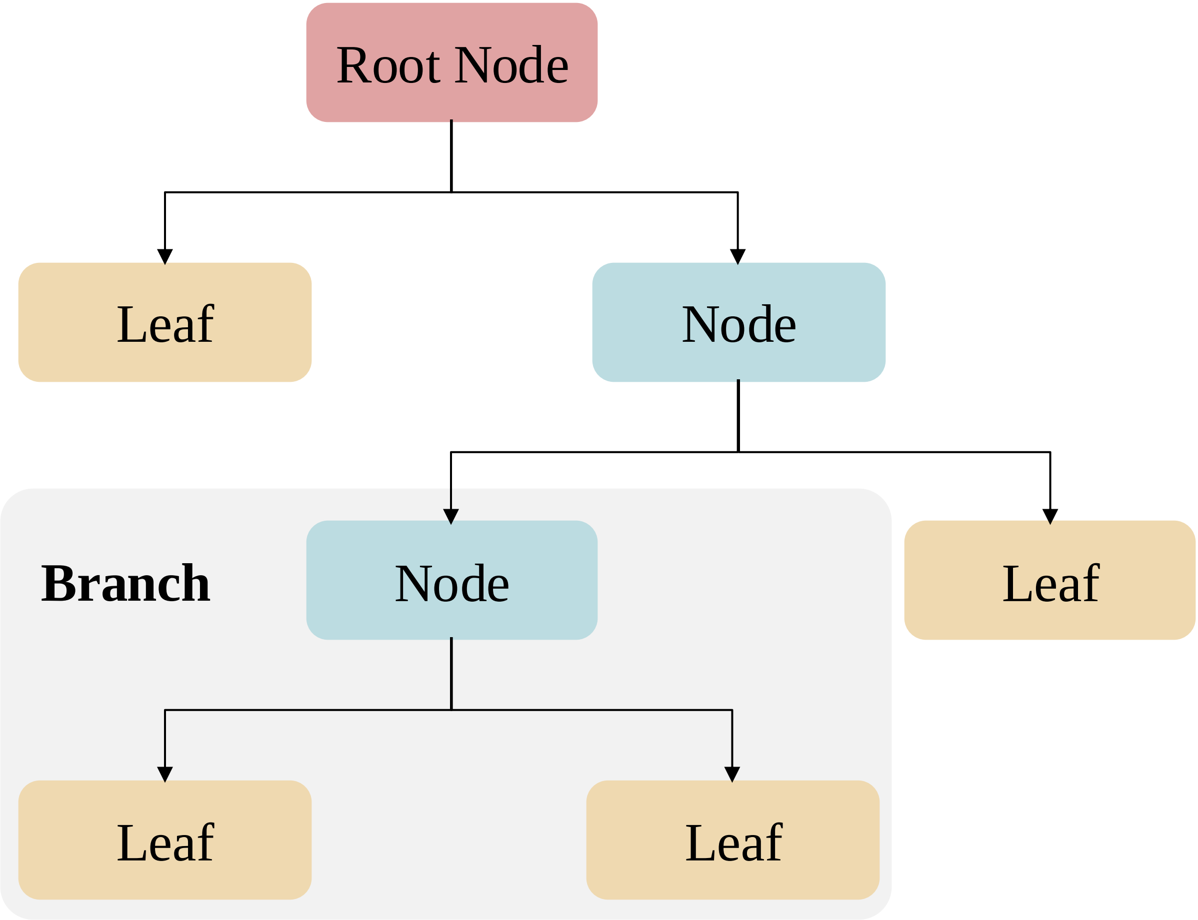
Random Forest



Random Forest

Decision Tree

- **Root Node**: the top-level node
- **Node**: internal node or decision node
- **Parent Node**: a node that precedes a (child) node
- **Leaf**: terminal node – a node at the end of a branch – represents outcome of the tree (label or numerical value)
- **Branches**: a subset of a tree, starting at an (internal) node until the leaves



Random Forest

Decision Tree for Classification

$$Gini(D) = \frac{n_1}{n} Gini(D_1) + \frac{n_2}{n} Gini(D_2)$$

$$Gini(D_i) = 1 - \sum_{j=1}^{c} p_j^2$$

$$D = \{3-, 3+\}$$

$$Gini(D) = 1 - \left(\frac{3}{3+3}\right)^2 - \left(\frac{3}{3+3}\right)^2 = \frac{1}{2}$$

$$D \xrightarrow[\text{Length} \leq 1.1]{\text{True}} D_1 = \{2-, 0+\} \qquad D \xrightarrow[\text{Length} \leq 1.1]{\text{False}} D_2 = \{1-, 3+\}$$

$$Gini(D_1) = 1 - \left(\frac{2}{2+0}\right)^2 - \left(\frac{0}{2+0}\right)^2 = 0$$

$$Gini(D_2) = 1 - \left(\frac{1}{1+3}\right)^2 - \left(\frac{3}{1+3}\right)^2 = \frac{3}{8}$$

$$Gini(Length \leq 1.1) = \frac{2}{6} * 0 + \frac{4}{6} * \frac{3}{8} = \frac{1}{4}$$

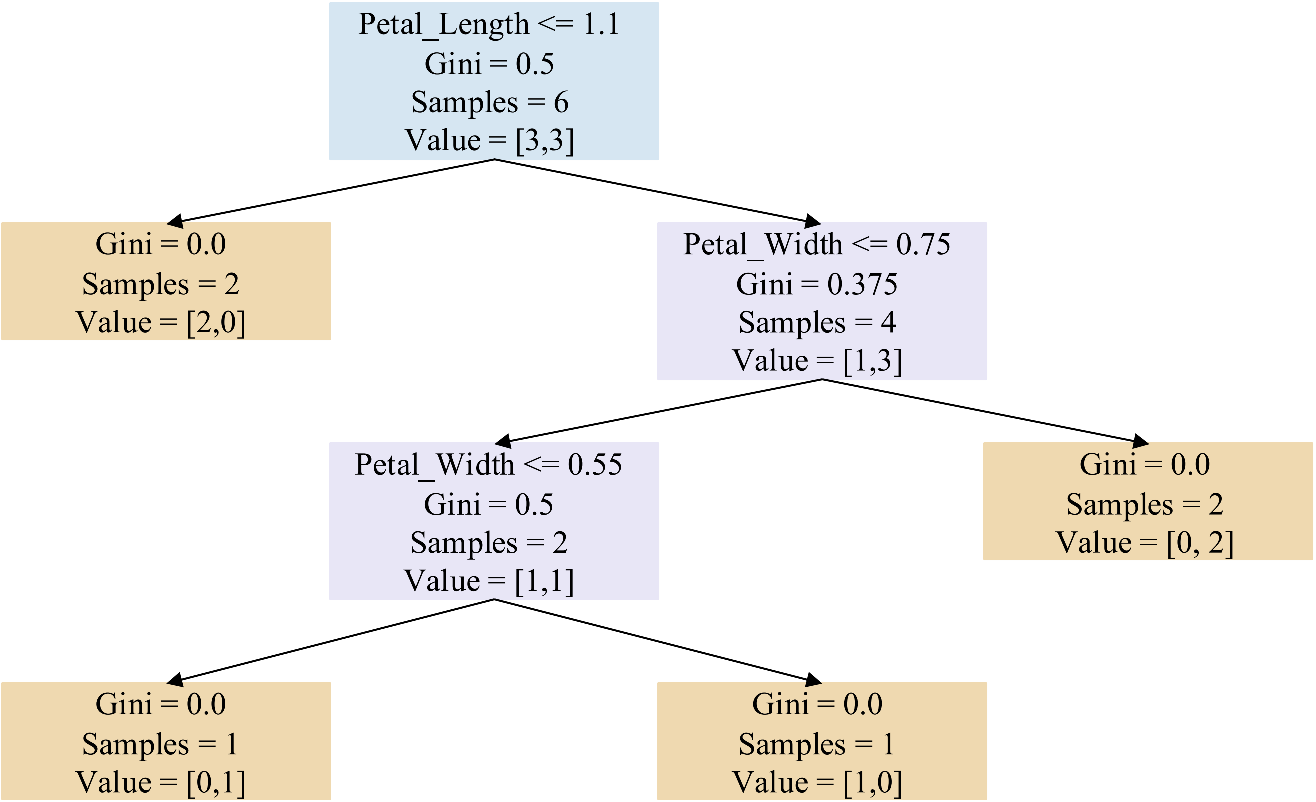
Petal_Length	Petal_Width	Label
1	0.2	0
1.3	0.6	0
0.9	0.7	0
1.7	0.5	1
1.8	0.9	1
1.2	1.3	1

Random Forest

Decision Tree for Classification

$$Gini(D) = \frac{n_1}{n} Gini(D_1) + \frac{n_2}{n} Gini(D_2)$$

$$Gini(D_i) = 1 - \sum_{j=1}^{c} p_j^2$$



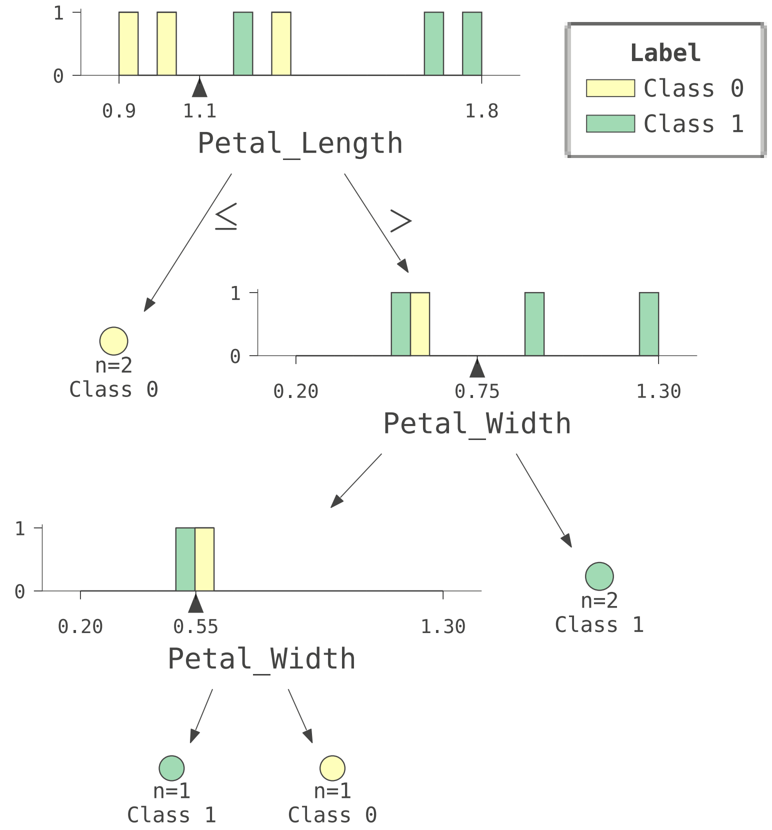
Petal_Length	Petal_Width	Label
1	0.2	0
1.3	0.6	0
0.9	0.7	0
1.7	0.5	1
1.8	0.9	1
1.2	1.3	1

Random Forest

Decision Tree for Classification

$$Gini(D) = \frac{n_1}{n} Gini(D_1) + \frac{n_2}{n} Gini(D_2)$$

$$Gini(D_i) = 1 - \sum_{j=1}^{c} p_j^2$$



Petal_Length	Petal_Width	Label
1	0.2	0
1.3	0.6	0
0.9	0.7	0
1.7	0.5	1
1.8	0.9	1
1.2	1.3	1

AI VIET NAM
@aivietnam.edu.vn

Random Forest

Decision Tree for Classification

$$Gain(D) = 1 - Entropy(D)$$

$$Entropy(D) = \frac{n_1}{n} Entropy(D_1) + \frac{n_2}{n} Entropy(D_2)$$

$$Entropy(D_i) = -\sum_{j=1}^{c} p_j \log_2 p_j$$

$D = \{3-, 3+\}$

$$Entropy(D) = -\frac{3}{6}\log_2\frac{3}{3} - \frac{3}{6}\log_2\frac{3}{6} = 1$$

Width ≤ 0.8

True: $D_1 = \{3-, 1+\}$

False: $D_2 = \{0-, 2+\}$

$$Entropy(D_1) = -\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4} = 0.81$$

$$Entropy(D_2) = -\frac{0}{2}\log_2\frac{0}{2} - \frac{2}{2}\log_2\frac{2}{2} = 0.0$$

$$Entropy(Width \leq 0.8) = \frac{4}{6} * 0.81 + \frac{2}{6} * 0.0 = 0.54$$

$$Gain(Width \leq 0.8) = 1 - 0.54 = 0.46$$

Petal_Length	Petal_Width	Label
1	0.2	0
1.3	0.6	0
0.9	0.7	0
1.7	0.5	1
1.8	0.9	1
1.2	1.3	1

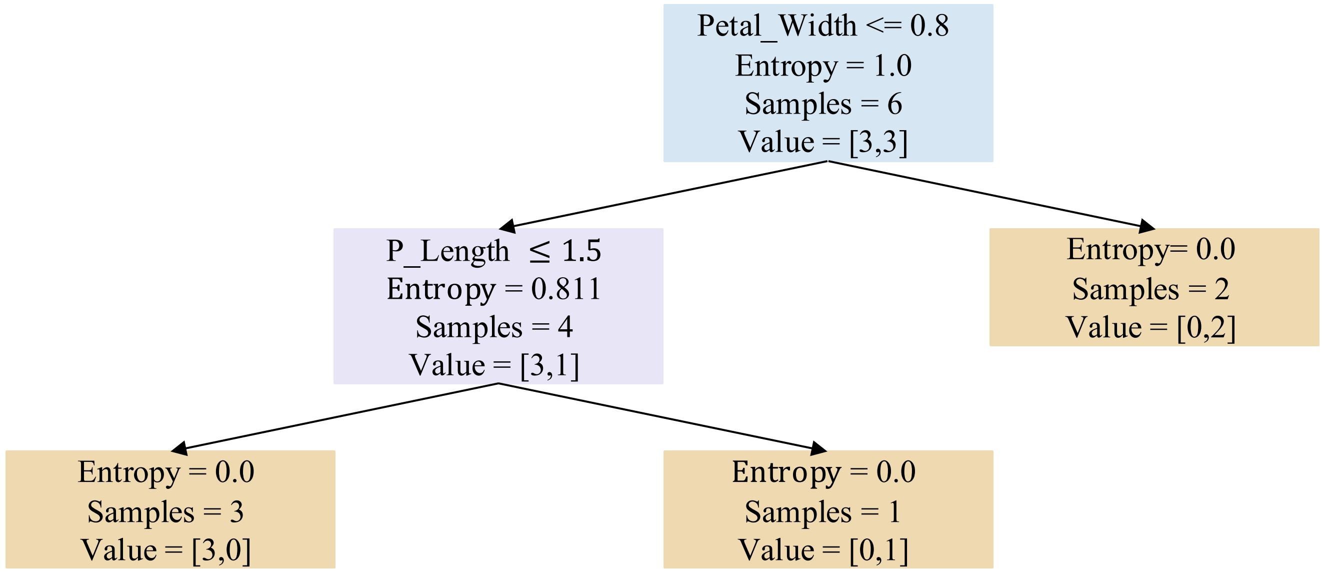
Random Forest

Decision Tree for Classification

$$Gain(D) = 1 - Entropy(D)$$

$$Entropy(D) = \frac{n_1}{n} Entropy(D_1) + \frac{n_2}{n} Entropy(D_2)$$

$$Entropy(D_i) = -\sum_{j=1}^{c} p_j \log_2 p_j$$



Petal_Length	Petal_Width	Label
1	0.2	0
1.3	0.6	0
0.9	0.7	0
1.7	0.5	1
1.8	0.9	1
1.2	1.3	1

AI VIET NAM
@aivietnam.edu.vn

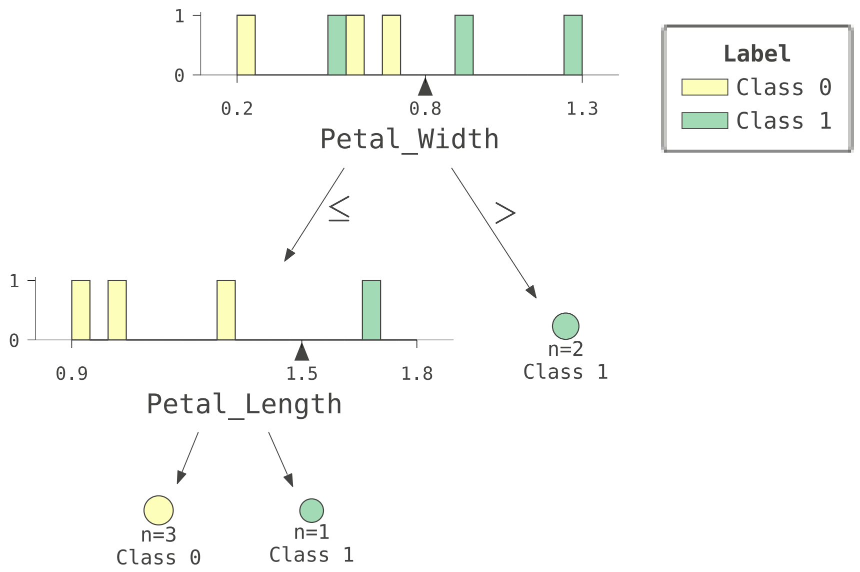
Random Forest

Decision Tree for Classification

$$Gain(D) = 1 - Entropy(D)$$

$$Entropy(D) = \frac{n_1}{n} Entropy(D_1) + \frac{n_2}{n} Entropy(D_2)$$

$$Entropy(D_i) = -\sum_{j=1}^{c} p_j \log_2 p_j$$



Petal_Length	Petal_Width	Label
1	0.2	0
1.3	0.6	0
0.9	0.7	0
1.7	0.5	1
1.8	0.9	1
1.2	1.3	1

Random Forest

Decision Tree for Regression

$$SSE(D) = SSE(D_1) + SSE(D_2)$$
$$MSE(D) = MSE(D_1) + MS(D_2)$$

$$SSE(D_i) = \sum_{j=1}^{n_i} (x_j - \bar{x}_i)^2$$

$$MSE(D_i) = \frac{1}{n_i} \sum_{j=1}^{n_i} (x_j - \bar{x}_i)^2$$

$$D = \{0, 0, 55, 83\}$$

$$mean(D) = 34.5$$

$$SSE(D_1) = (0 - 34.5)^2 + (0 - 34.5)^2 + (55 - 34.5)^2 + (83 - 34.5)^2 = 5153$$

$$MSE(D_1) = \frac{(0 - 34.5)^2 + (0 - 34.5)^2 + (55 - 34.5)^2 + (83 - 34.5)^2}{4} = 1288.25$$

Experience	Salary
1.5	0
2.5	0
4.0	55
5.5	83

Random Forest

Decision Tree for Regression

$$SSE(D) = SSE(D_1) + SSE(D_2)$$
$$MSE(D) = MSE(D_1) + MS(D_2)$$

$$SSE(D_i) = \sum_{j=1}^{n_i} (x_j - \bar{x}_i)^2$$

$$MSE(D_i) = \frac{1}{n_i} \sum_{j=1}^{n_i} (x_j - \bar{x}_i)^2$$

$D = \{0, 0, 55, 83\}$

True | False

$SSE(Experience \leq 2.0)$

$D_1 = \{0\}$

$D_2 = \{0, 55, 83\}$

$mean(D_1) = 0$

$mean(D_2) = 46$

Experience	Salary
1.5	0
2.5	0
4.0	55
5.5	83

$$SSE(D_1) = (0 - 0)^2 = 0$$

$$SSE(Experience \leq 2.0) = 1450$$

$$SSE(D_2) = (0 - 46)^2 + (55 - 46)^2 + (83 - 46)^2 = 1450$$

Random Forest

Decision Tree for Regression

$$SSE(D) = SSE(D_1) + SSE(D_2)$$
$$MSE(D) = MSE(D_1) + MS(D_2)$$

$$SSE(D_i) = \sum_{j=1}^{n_i} (x_j - \bar{x}_i)^2$$

$$MSE(D_i) = \frac{1}{n_i} \sum_{j=1}^{n_i} (x_j - \bar{x}_i)^2$$

$D = \{0, 0, 55, 83\}$

True | False

$MSE(Experience \leq 2.0)$

$D_1 = \{0\}$

$mean(D_1) = 0$

$D_2 = \{0, 55, 83\}$

$mean(D_2) = 46$

Experience	Salary
1.5	0
2.5	0
4.0	55
5.5	83

$$MSE(D_1) = \frac{(0-0)^2}{1} = 0$$

$$MSE(D_2) = \frac{(0-46)^2 + (55-46)^2 + (83-46)^2}{3} = 483$$

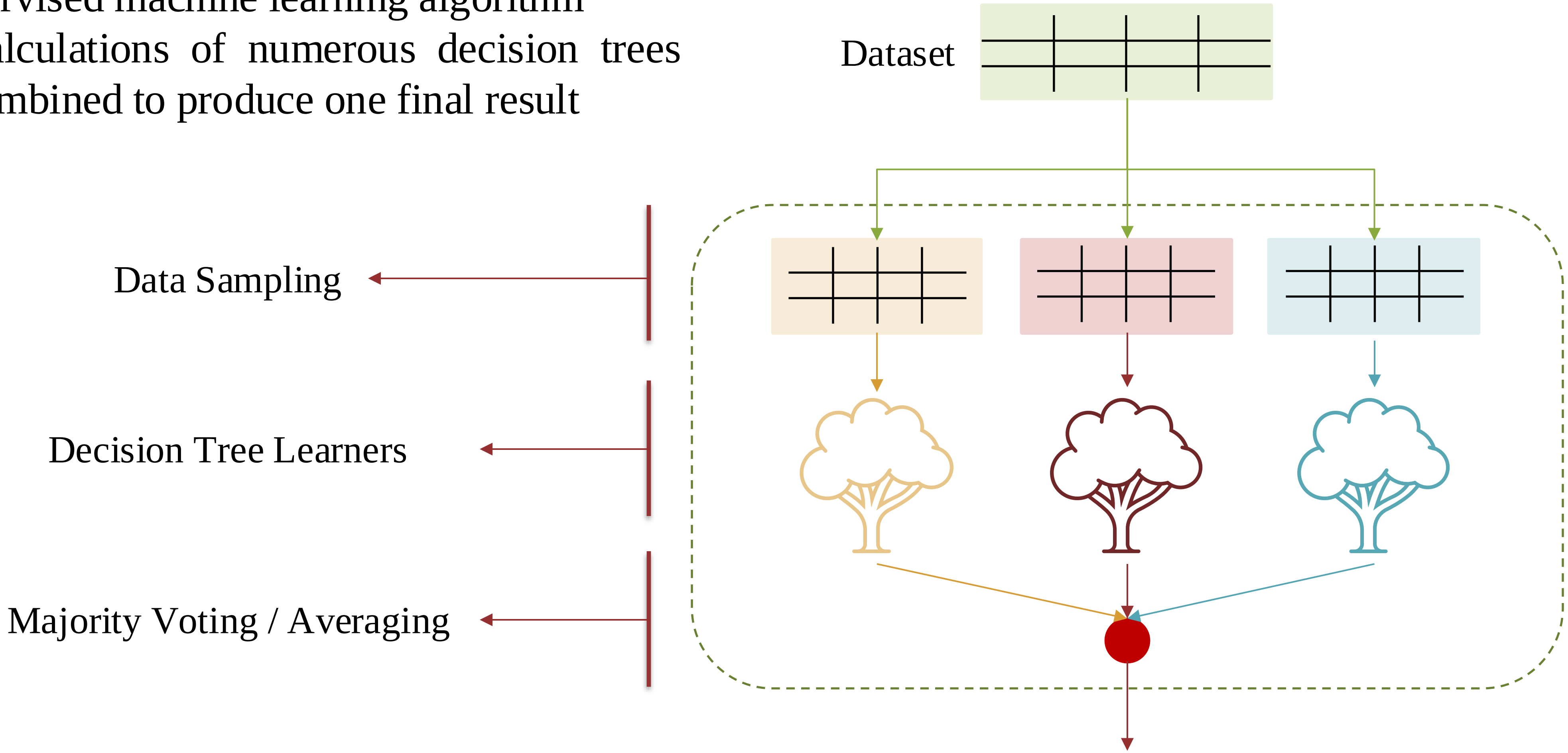
$$MSE(Experience \leq 2.0) = 483$$

Random Forest

Random Forest

A random forest

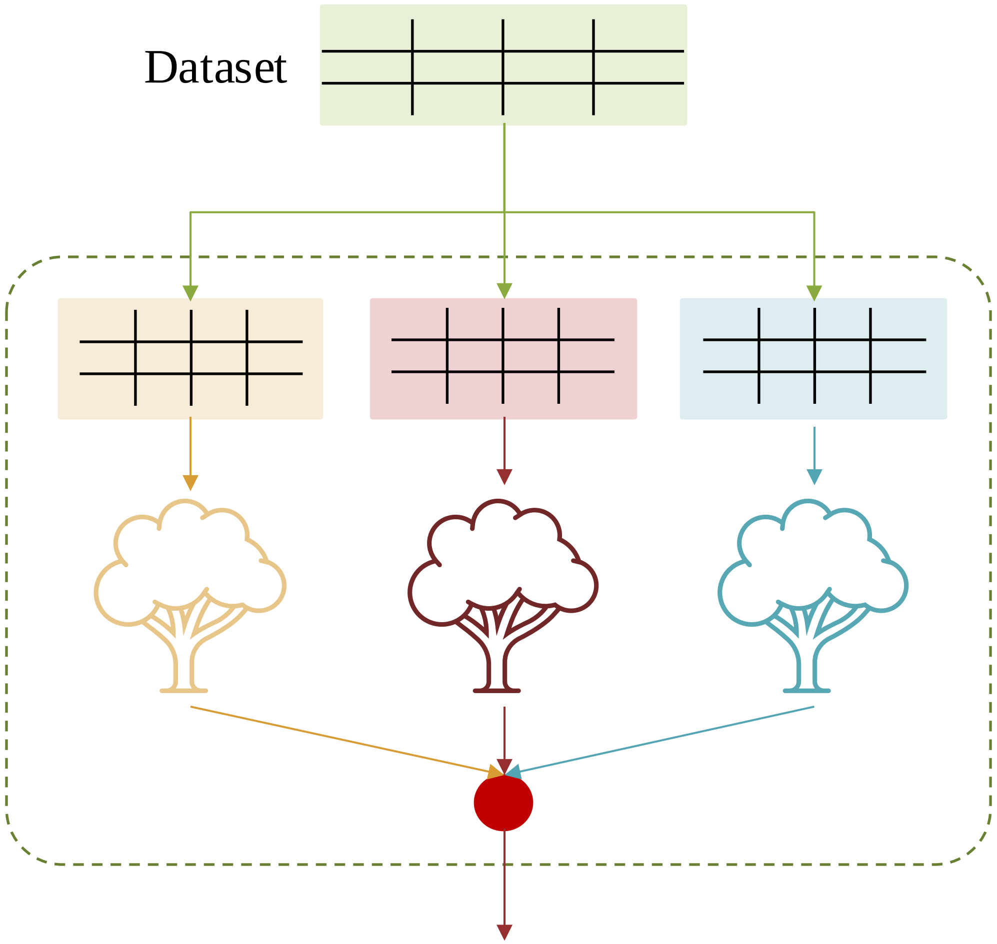
- a supervised machine learning algorithm
- the calculations of numerous decision trees are combined to produce one final result



AI VIET NAM
@aivietnam.edu.vn

Random Forest

Data Sampling



Feature = 1

Randomly sample with replacement

Length	Width	Label
1	0.2	0
1.3	0.6	0
0.9	0.7	0
1.7	0.5	1
1.8	0.9	1
1.2	1.3	1

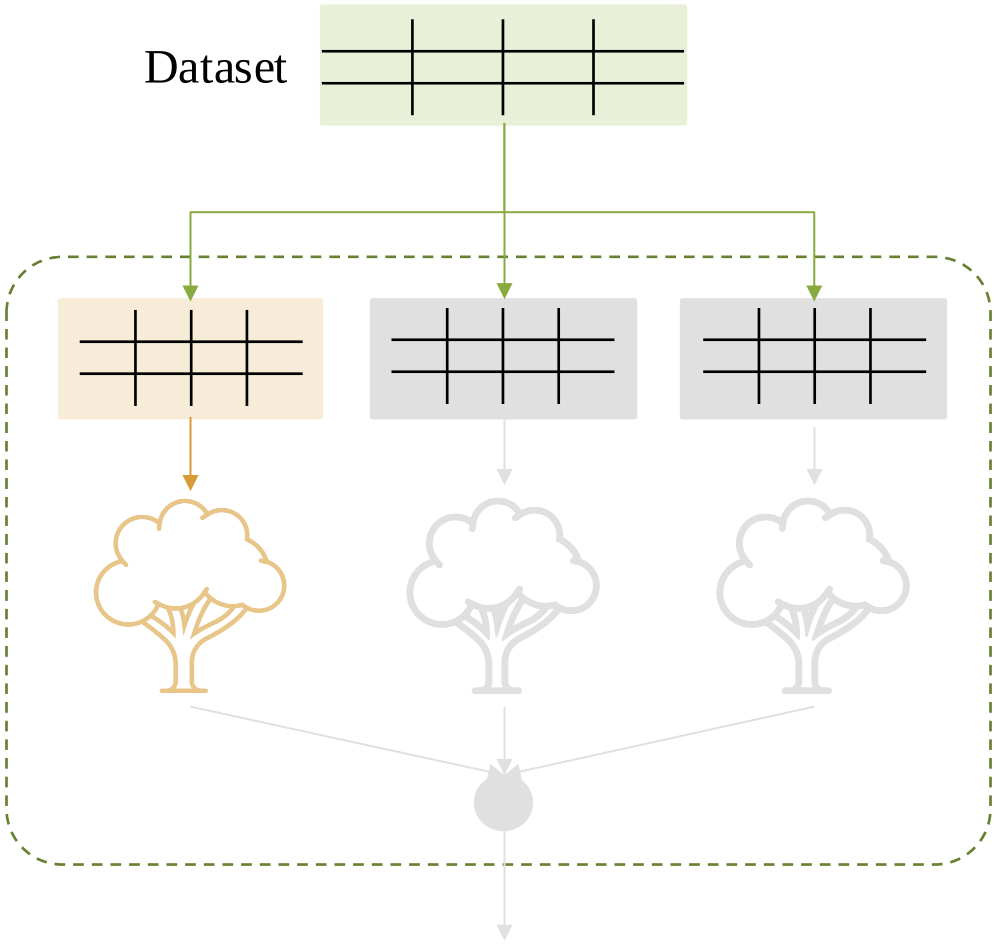
Length	Label
1	0
1.3	0
1	0
1.8	1
1.8	1
1.2	1

Width	Label
0.6	0
0.6	0
0.7	0
0.7	0
0.9	1
1.3	1

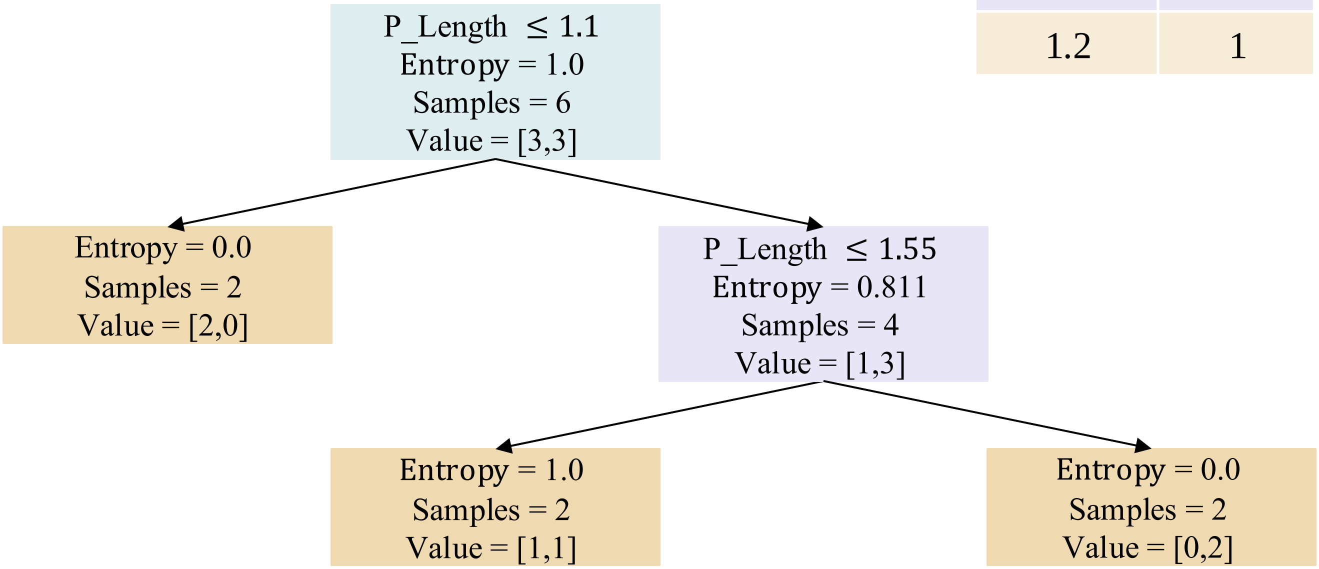
Length	Label
1	0
1.3	0
1.2	1
1.8	1
1.8	1
1.2	1

Random Forest

Decision Tree Learners

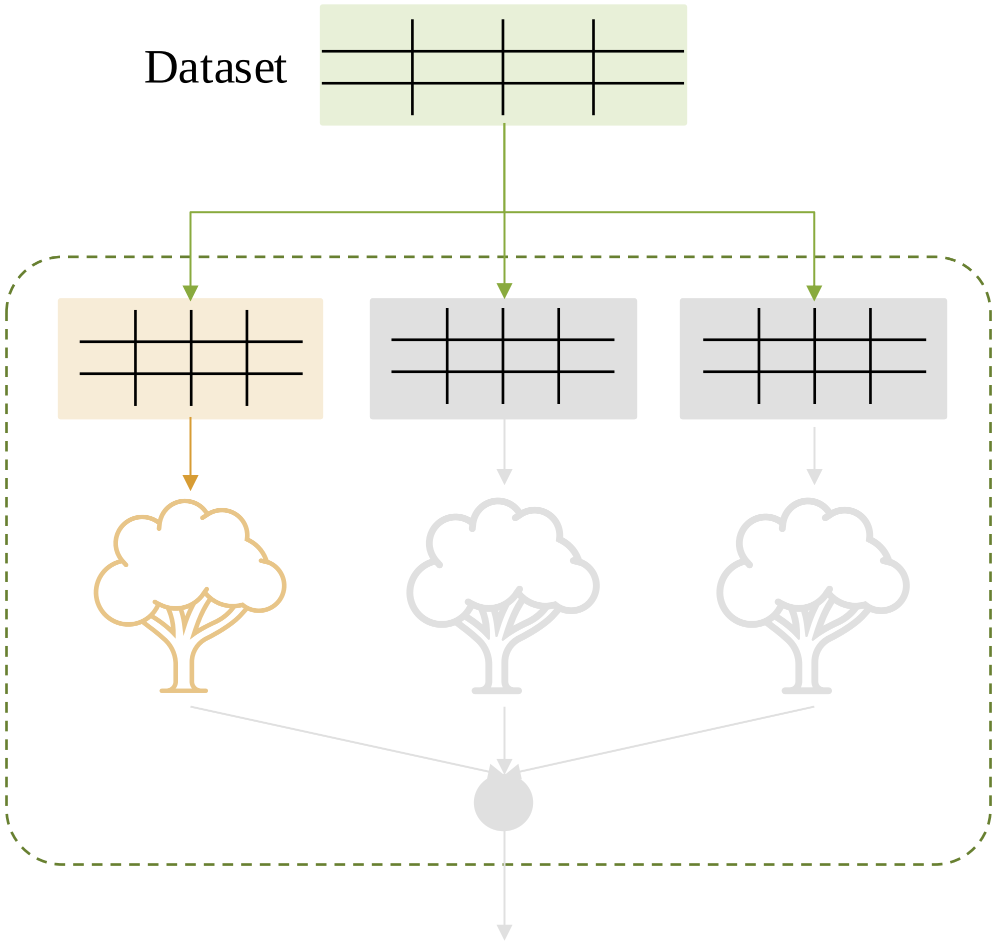


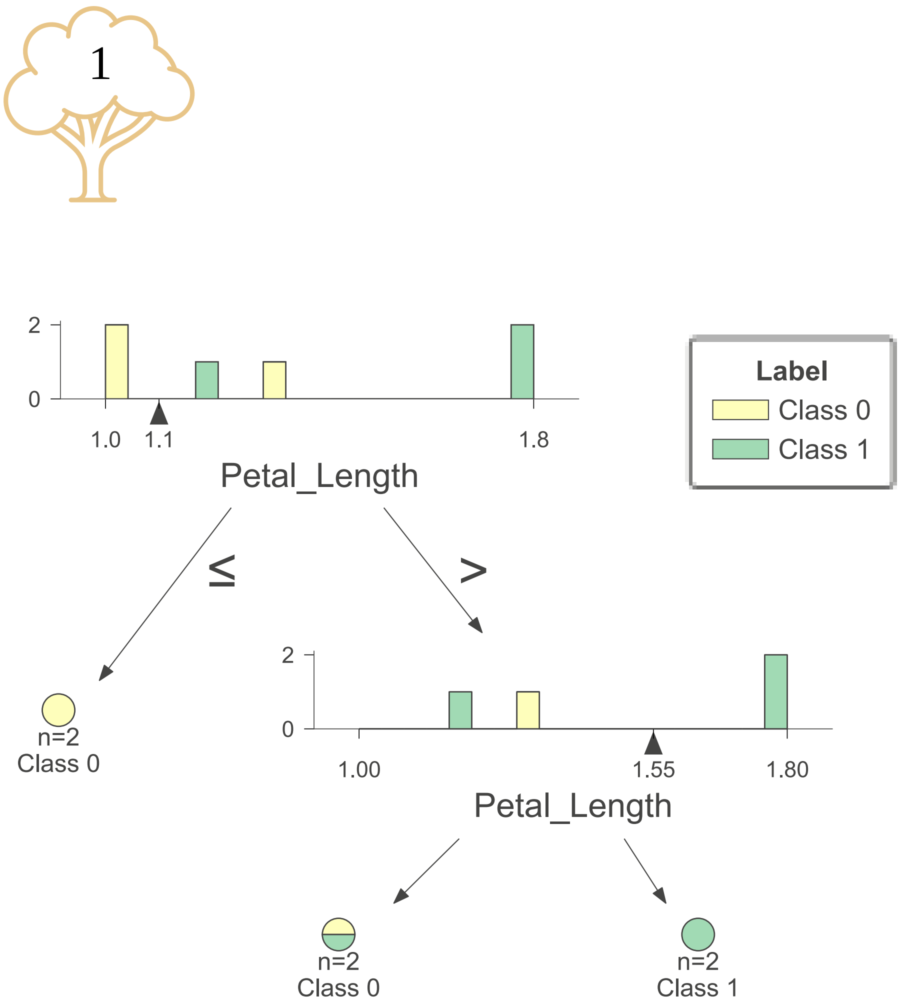
Length	Label
1	0
1.3	0
1	0
1.8	1
1.8	1
1.2	1



Random Forest

Decision Tree Learners

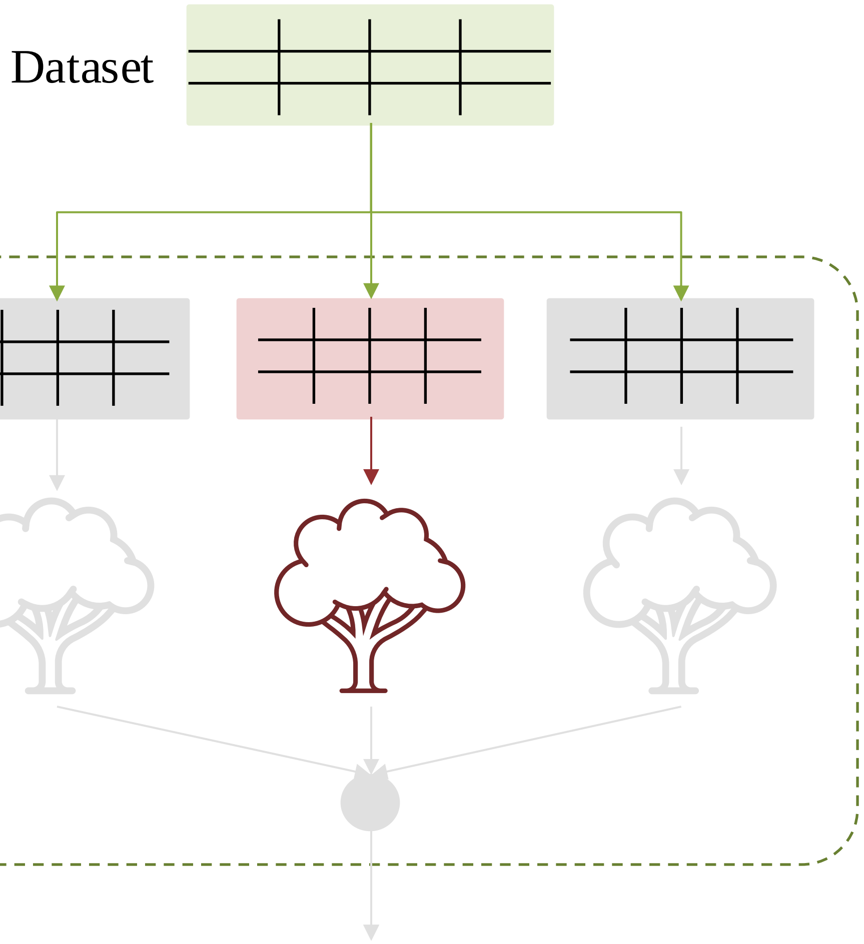


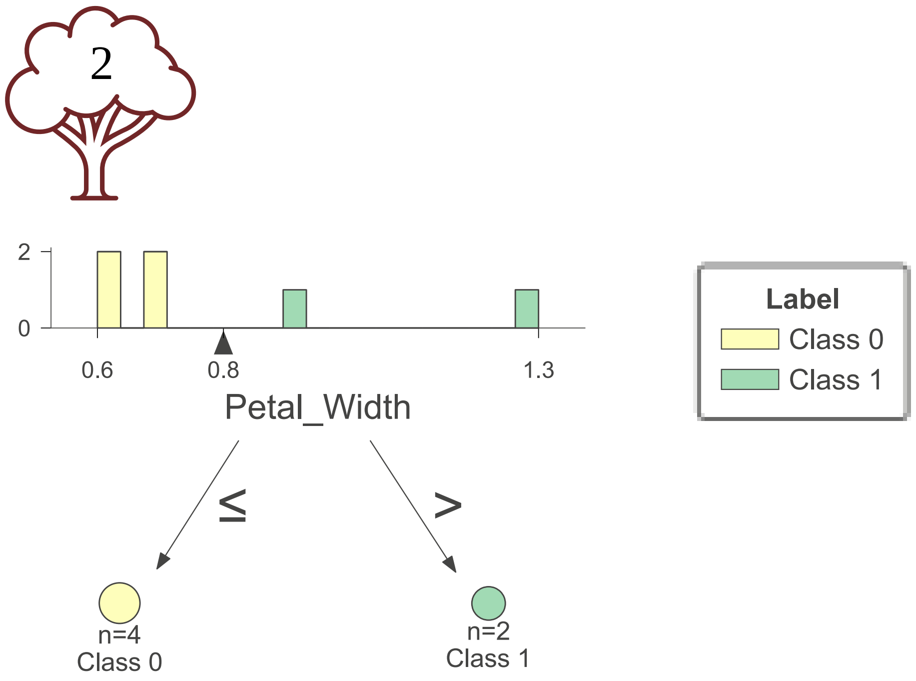


Length	Label
1	0
1.3	0
1	0
1.8	1
1.8	1
1.2	1

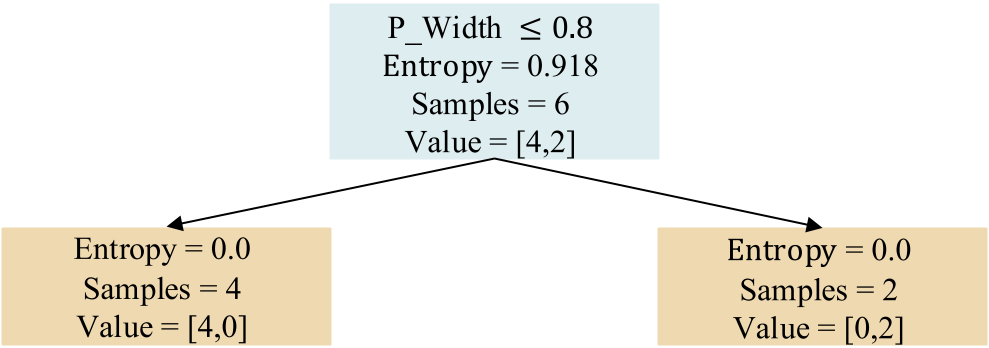
Random Forest

Decision Tree Learners



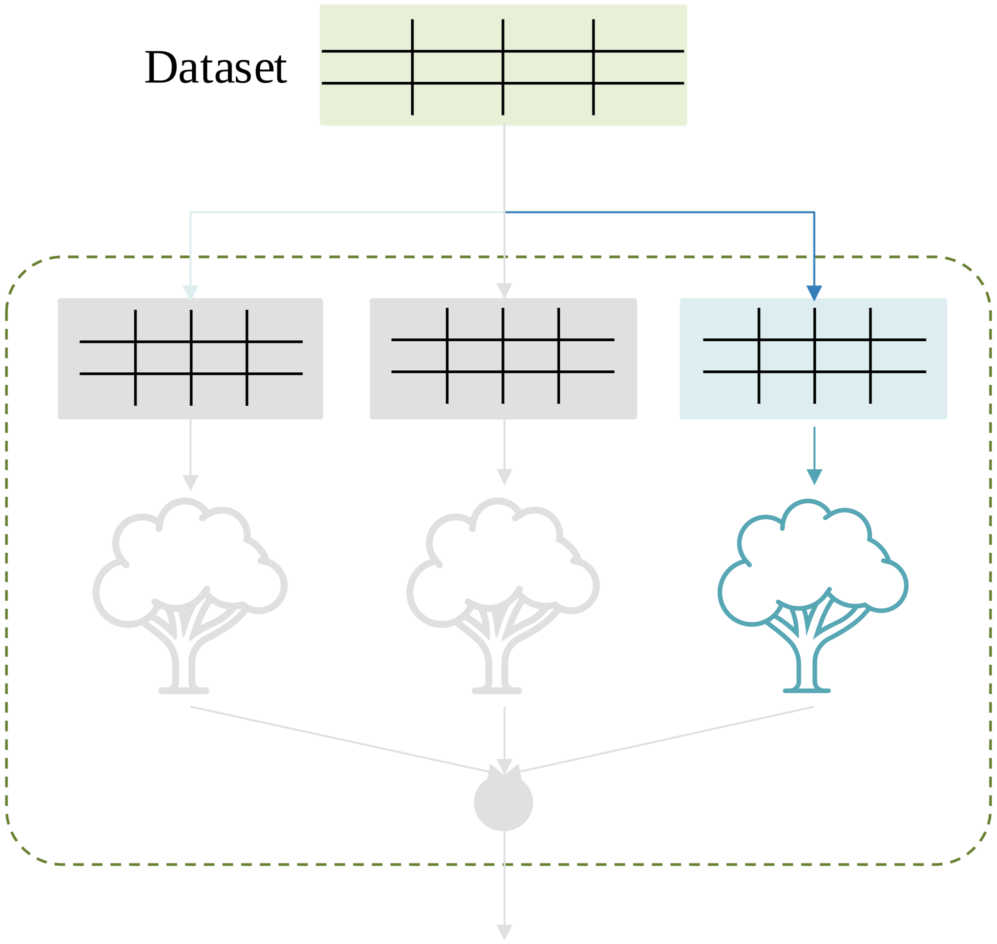


Width	Label
0.6	0
0.6	0
0.7	0
0.7	0
0.9	1
1.3	1

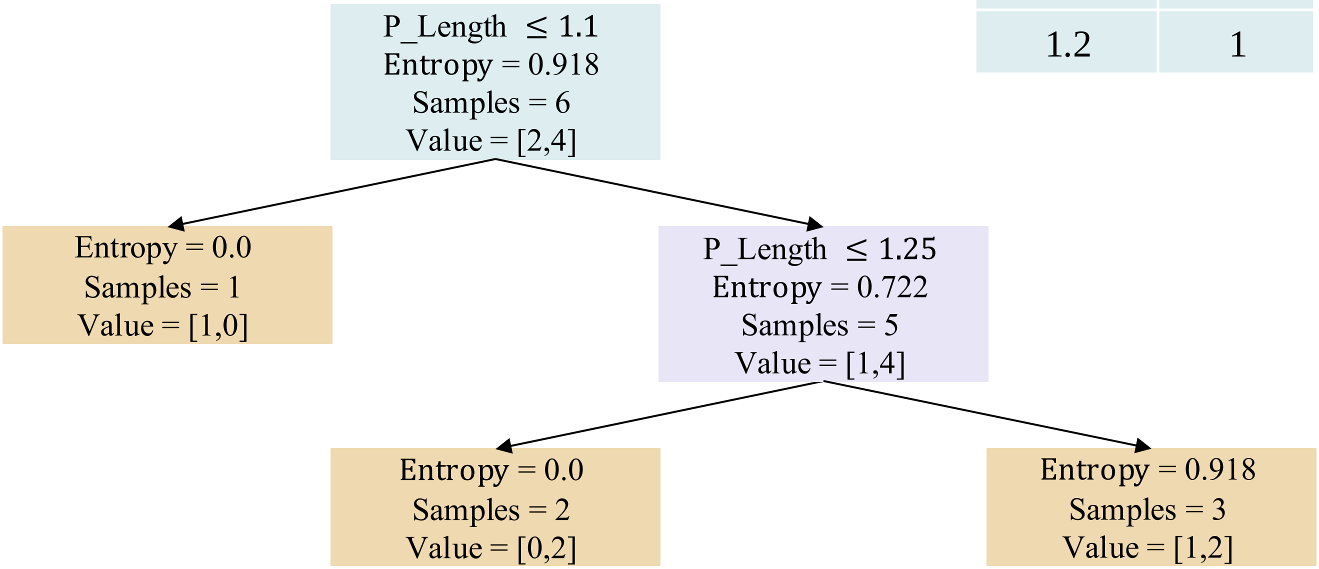


Random Forest

Decision Tree Learners

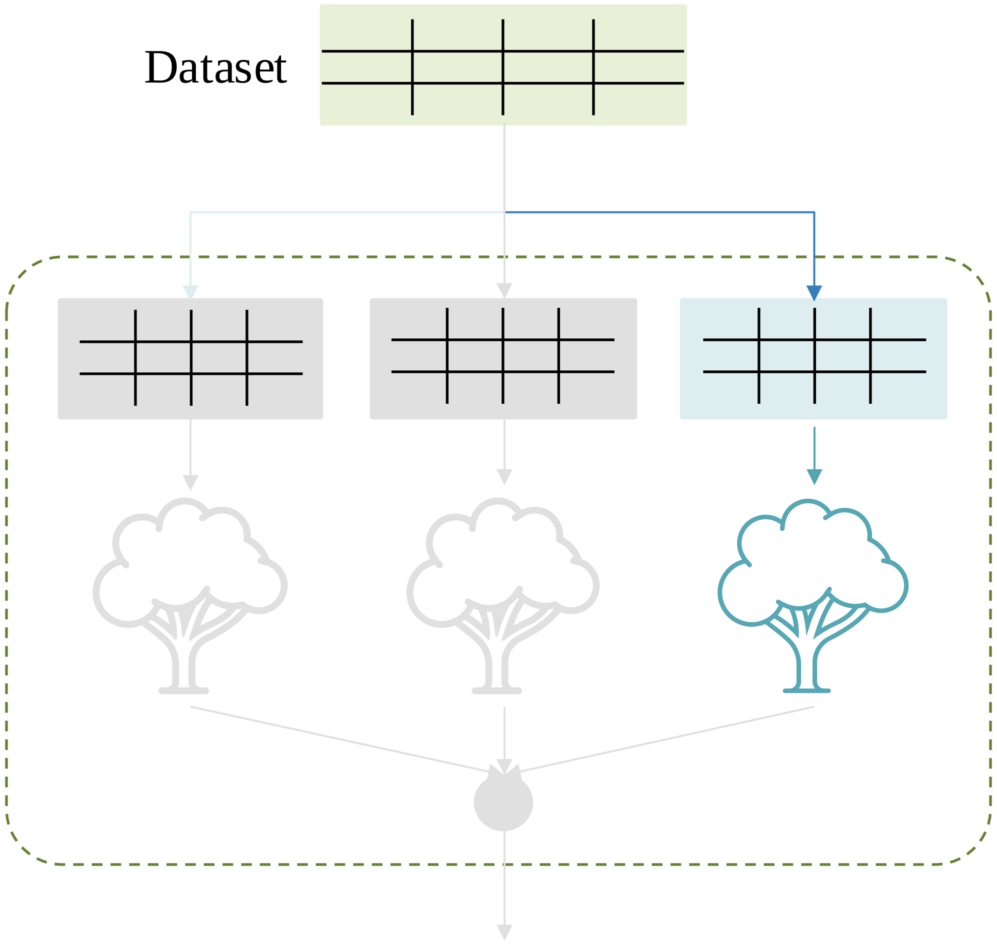


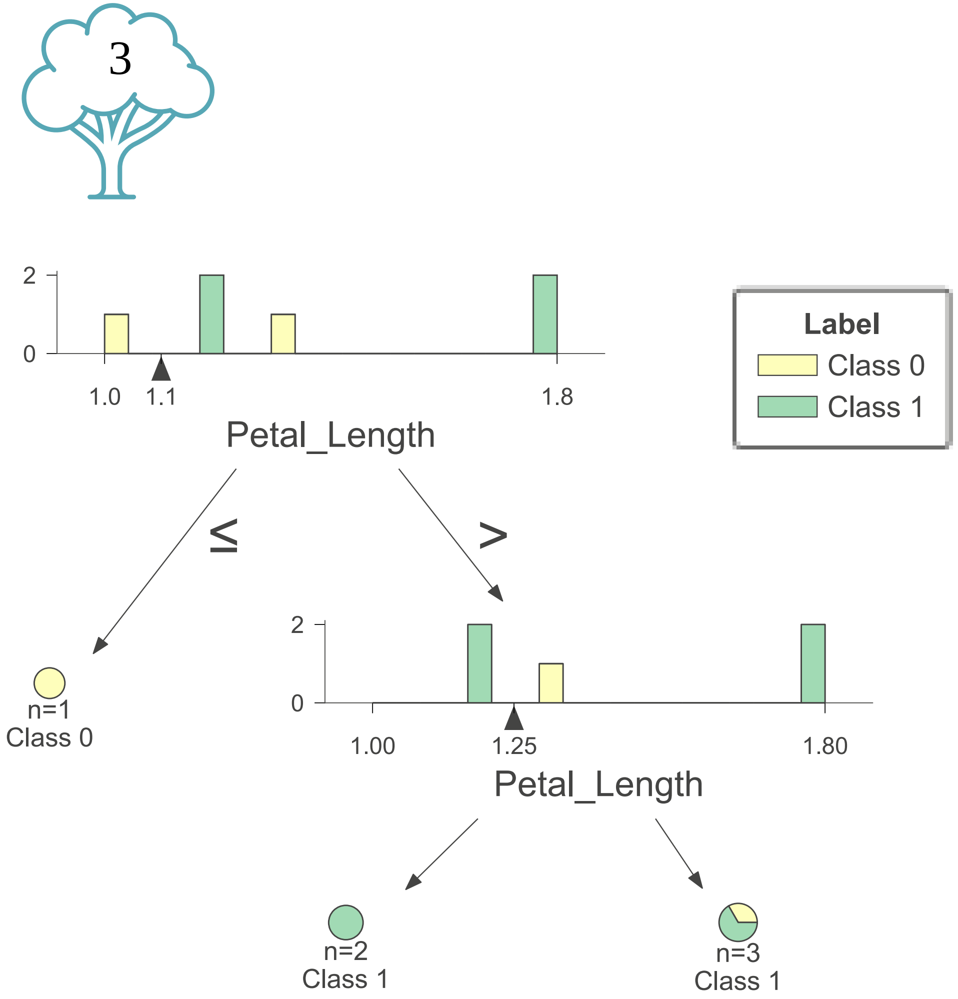
Length	Label
1	0
1.3	0
1.2	1
1.8	1
1.8	1
1.2	1



Random Forest

Decision Tree Learners



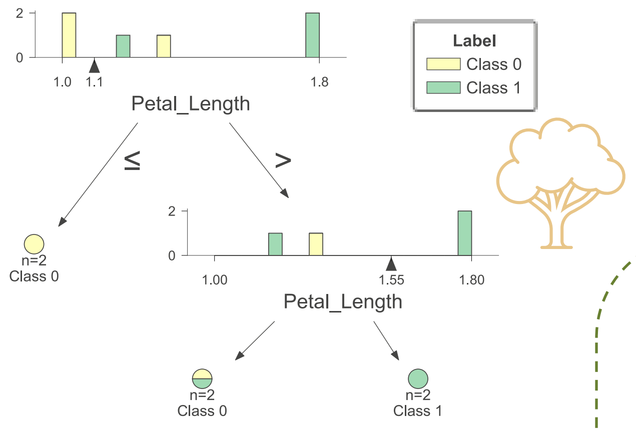


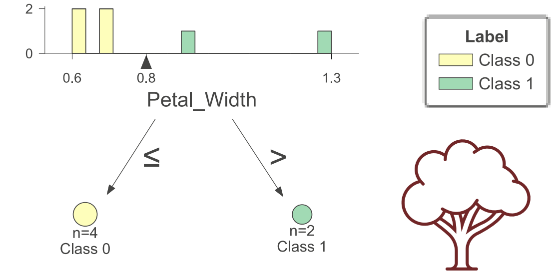
Length	Label
1	0
1.3	0
1.2	1
1.8	1
1.8	1
1.2	1

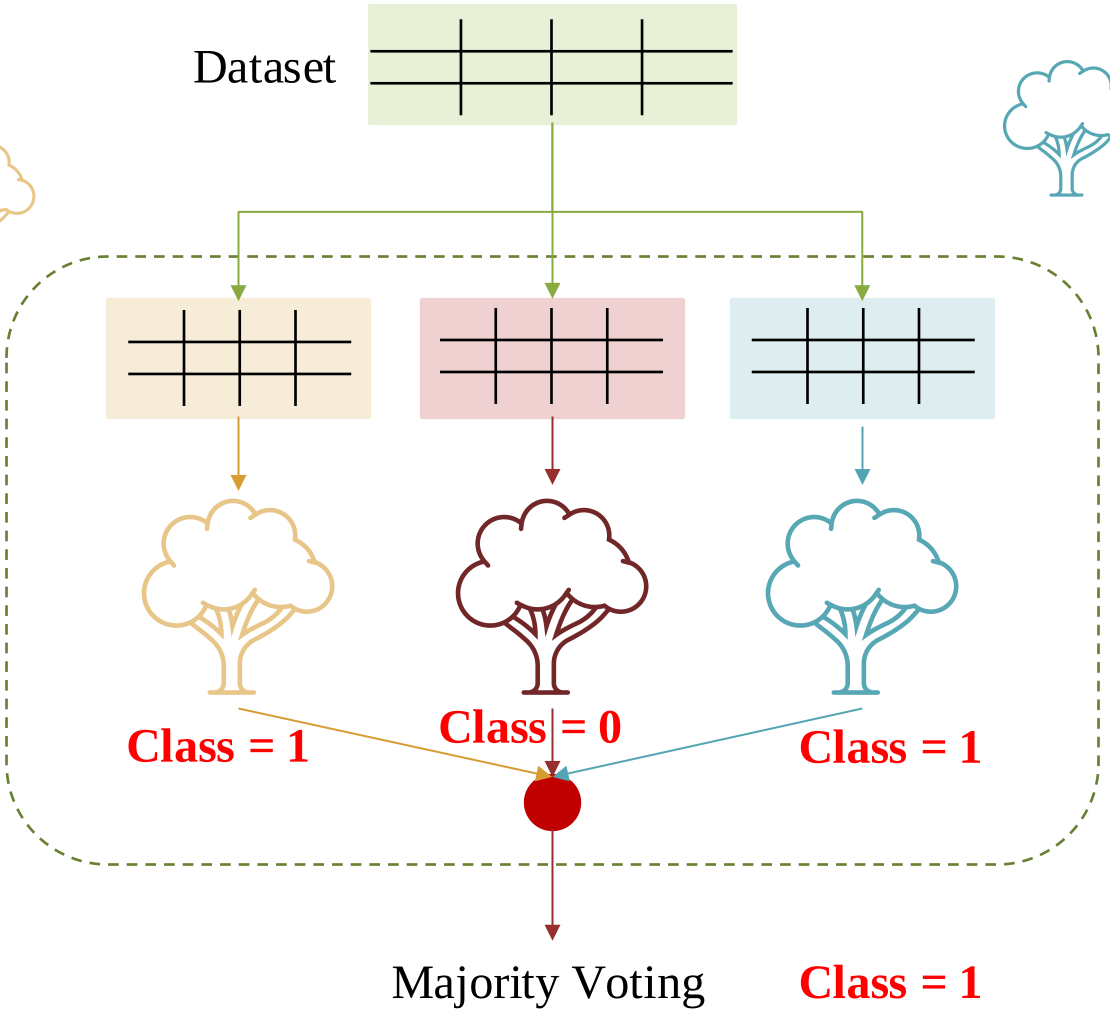
AI VIET NAM
@aivietnam.edu.vn

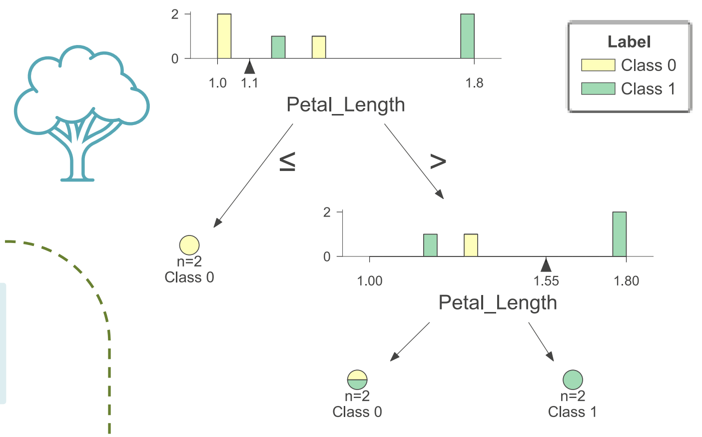
Random Forest

Majority Voting for Classification





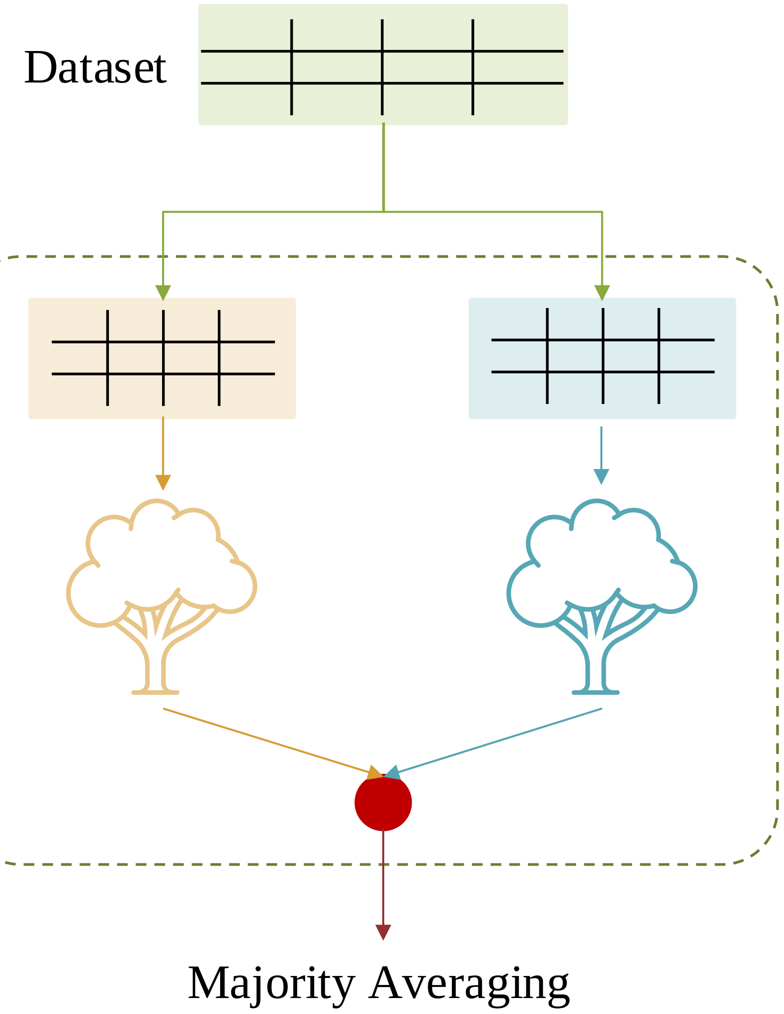




Test sample:
Petal_Length = 1.6
Petal_Width = 0.8

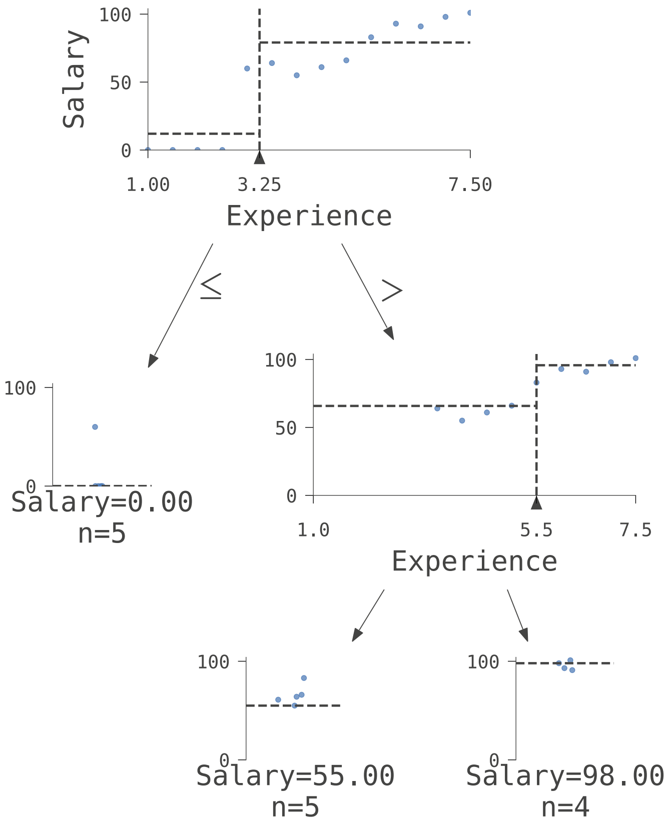
Random Forest

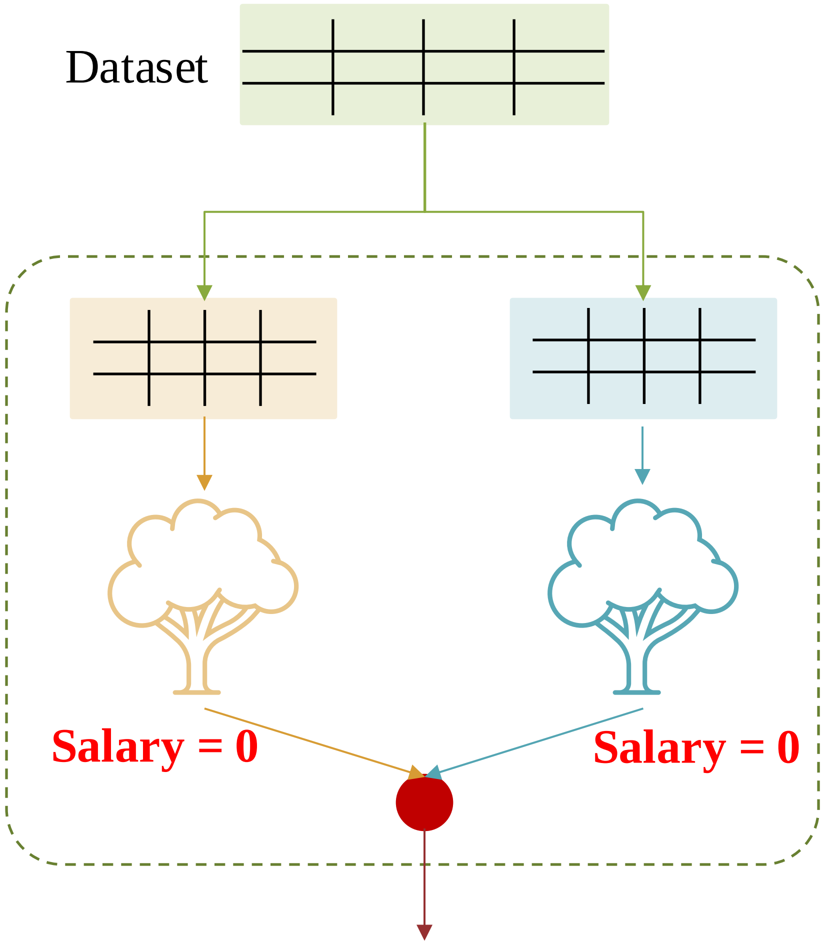
Random Forest for Regression

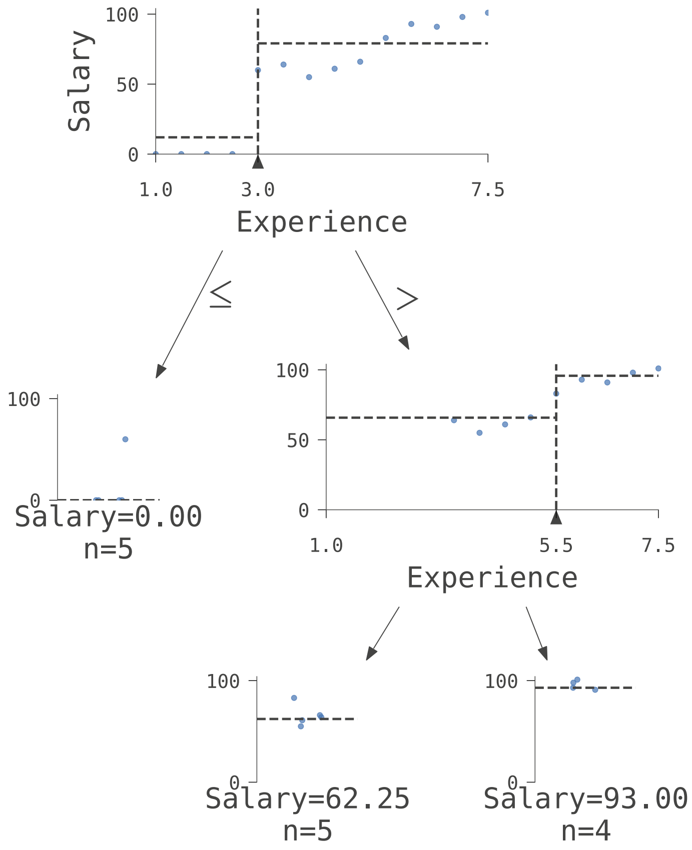


Random Forest

Majority Averaging for Regression







Test sample:
Experience = 3

Majority Averaging **Salary = (0+0)/2=0**

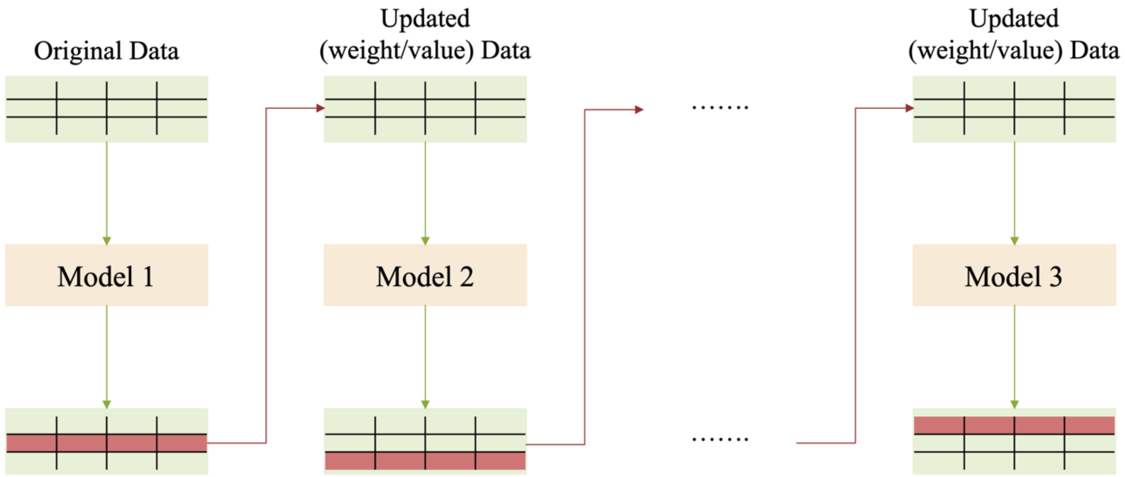
Outline

SECTION 1
Ensemble Learning

SECTION 2
Bagging Methods

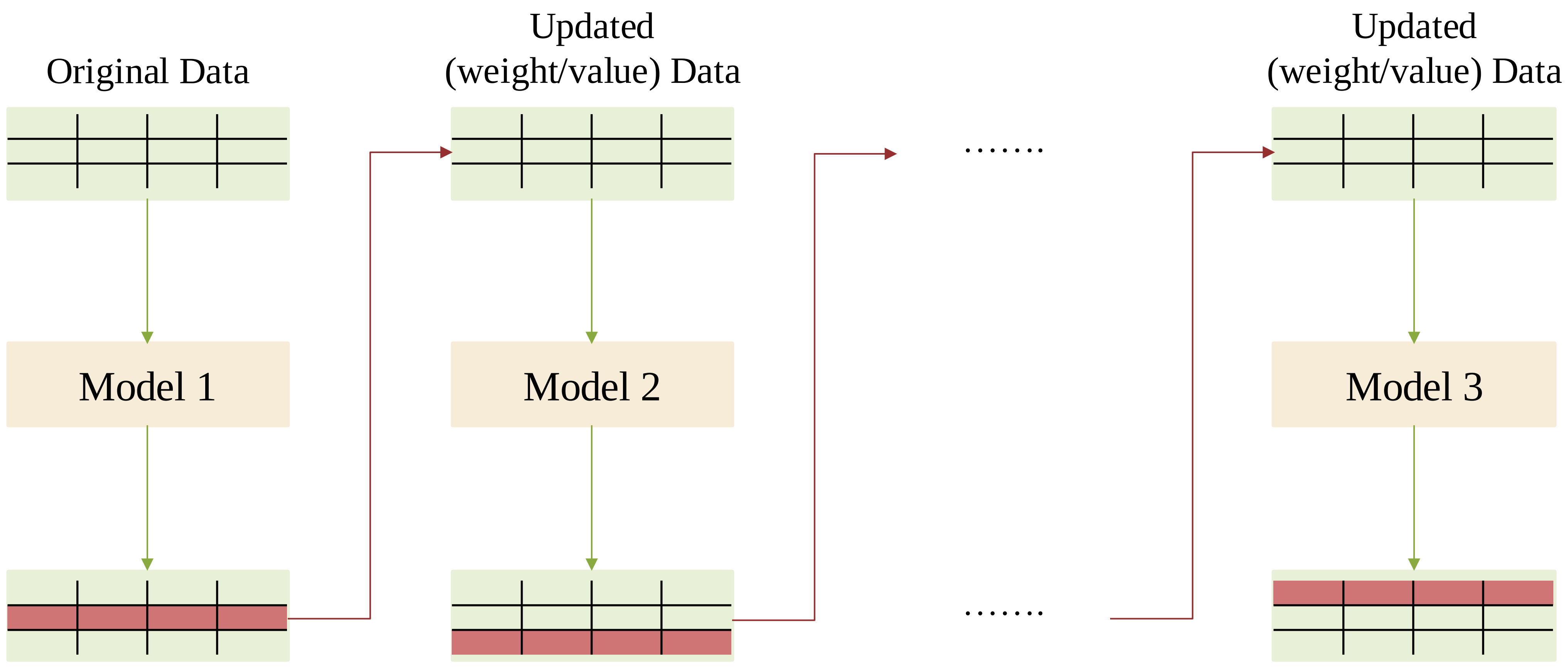
SECTION 3
Boosting Methods

SECTION 4
Implementation



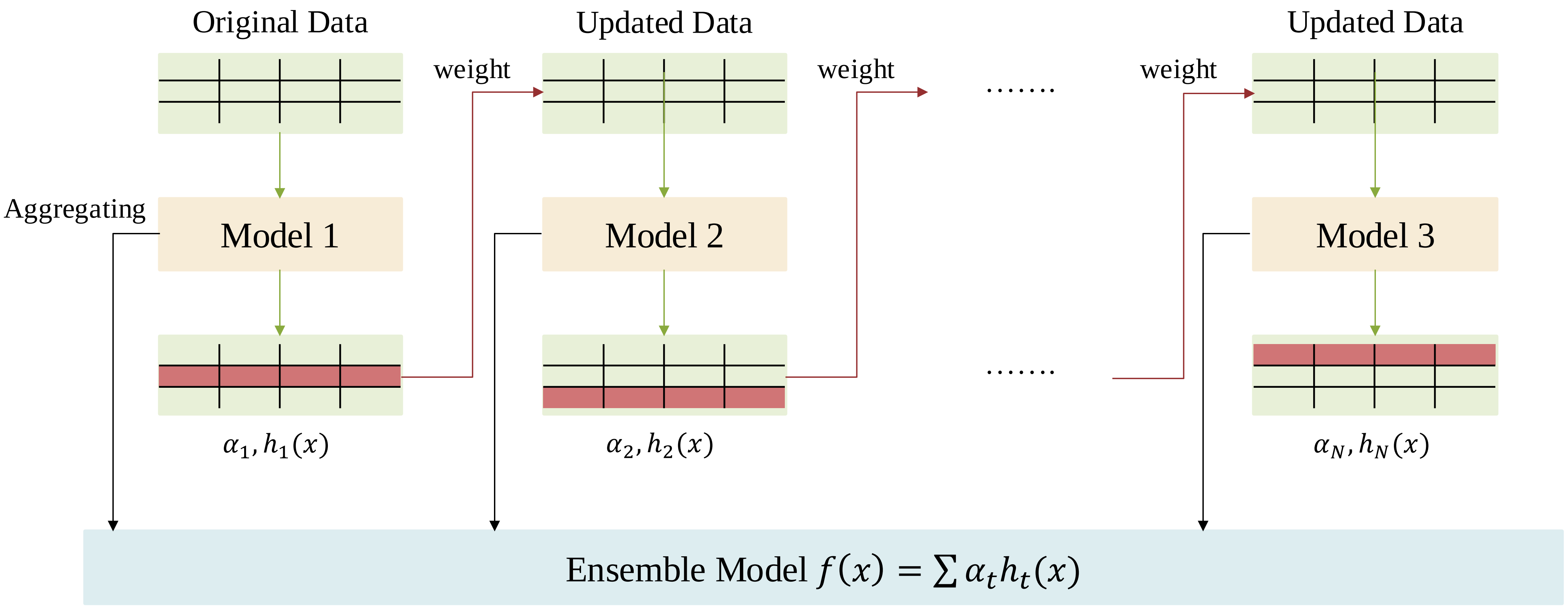
Boosting Methods

Boosting Methods



Boosting Methods

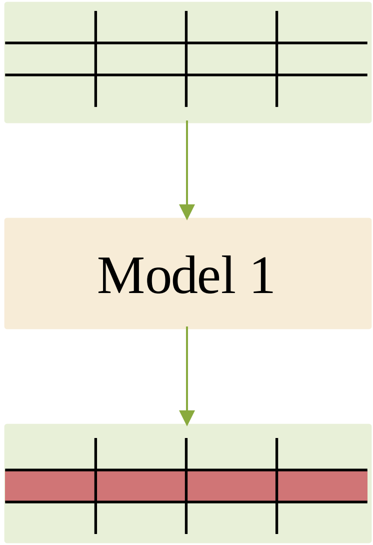
AdaBoost (Adaptive Boosting)



AdaBoost

Calculate weights of samples

Original Data



$\alpha_1, h_1(x)$

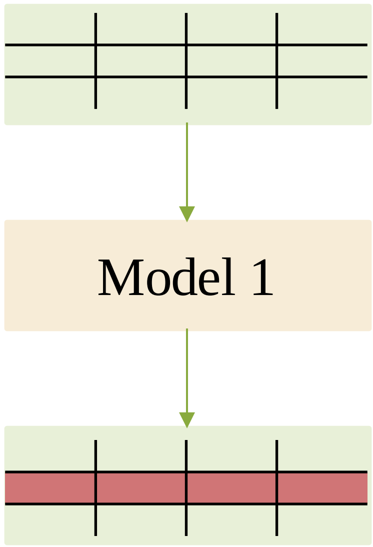
Initial sample weight = 1/N

Length	Width	Label	Weight
1	0.2	0	1/6
1.3	0.6	0	1/6
0.9	0.7	0	1/6
1.7	0.5	1	1/6
1.8	0.9	1	1/6
1.2	1.3	1	1/6

AdaBoost

Fitting model

Original Data

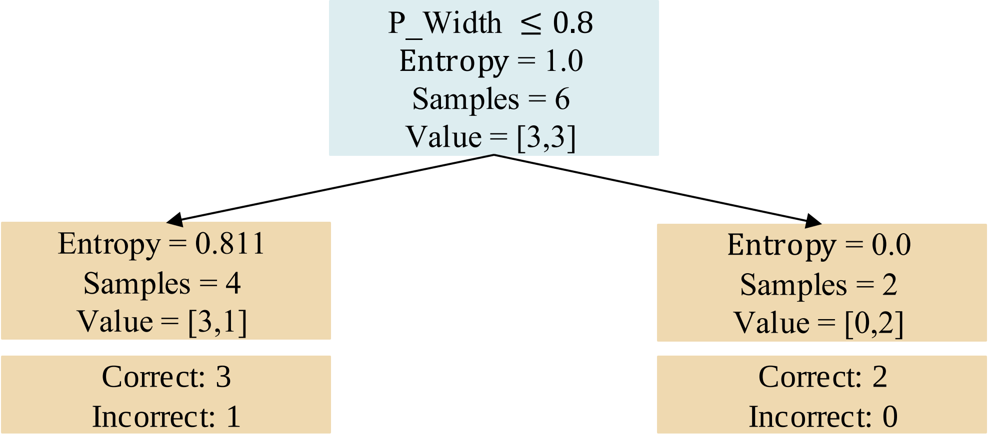


$\alpha_1, h_1(x)$

Initial sample weight = 1/N

Fitting Model 1

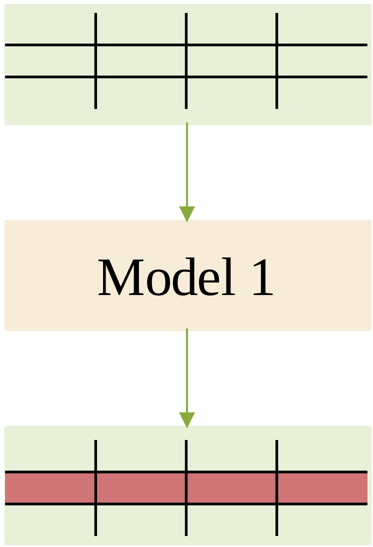
Length	Width	Label	Weight
1	0.2	0	1/6
1.3	0.6	0	1/6
0.9	0.7	0	1/6
1.7	0.5	1	1/6
1.8	0.9	1	1/6
1.2	1.3	1	1/6



AdaBoost

Calculate weights of weak learner

Original Data



$\alpha_1, h_1(x)$

Initial sample weight = 1/N

Fitting Model 1

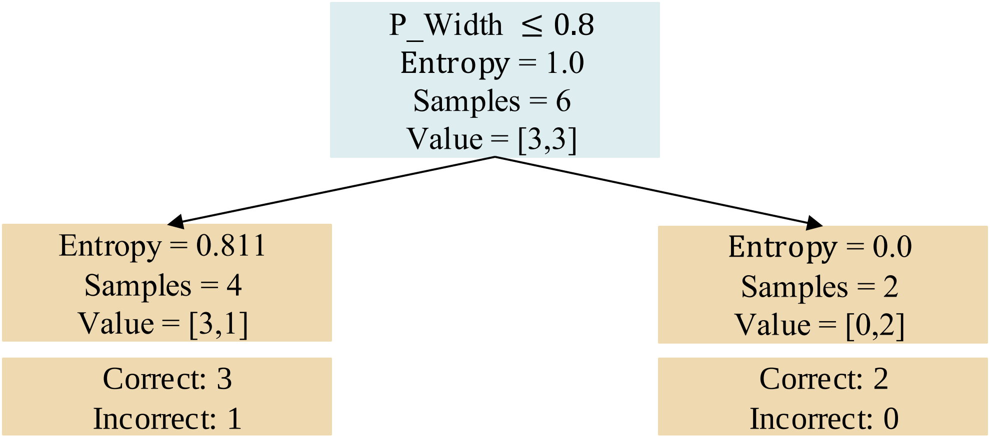
Total error (Error rate)

$$\varepsilon_1 = \sum_{incorrect} w_i = \frac{1}{6}$$

Weight of weak learner

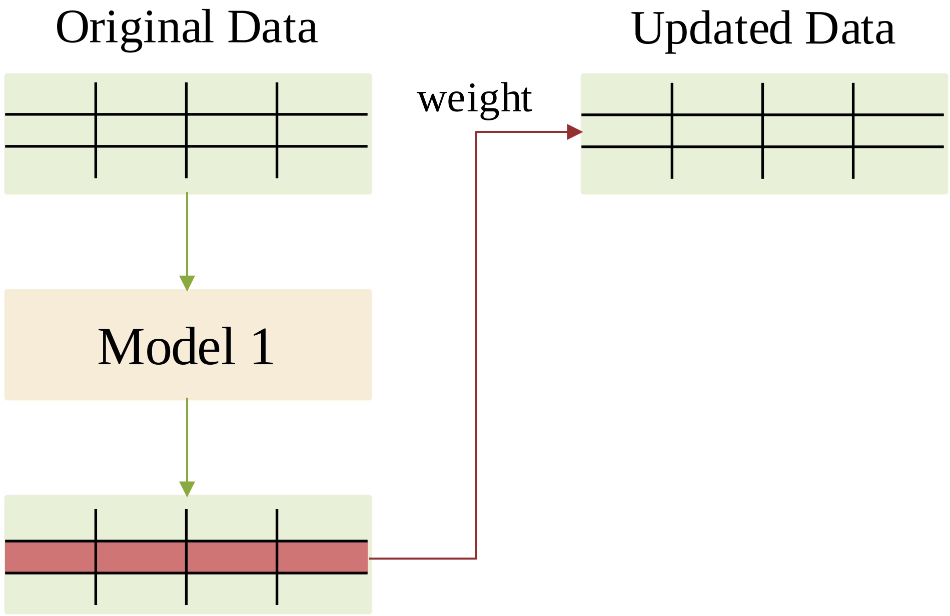
$$\alpha_1 = \frac{1}{2} * ln \frac{(1-\varepsilon_1)}{\varepsilon_1} = 0.8$$

Length	Width	Label	Weight
1	0.2	0	1/6
1.3	0.6	0	1/6
0.9	0.7	0	1/6
1.7	0.5	1	1/6
1.8	0.9	1	1/6
1.2	1.3	1	1/6



AdaBoost

Update weights of samples



$\alpha_1, h_1(x)$

$\alpha_1 = 0.8$

For incorrect predictions

$$w_{new} = w * e^{\alpha} = \frac{1}{6} * e^{0.8} = 0.37$$

For correct predictions

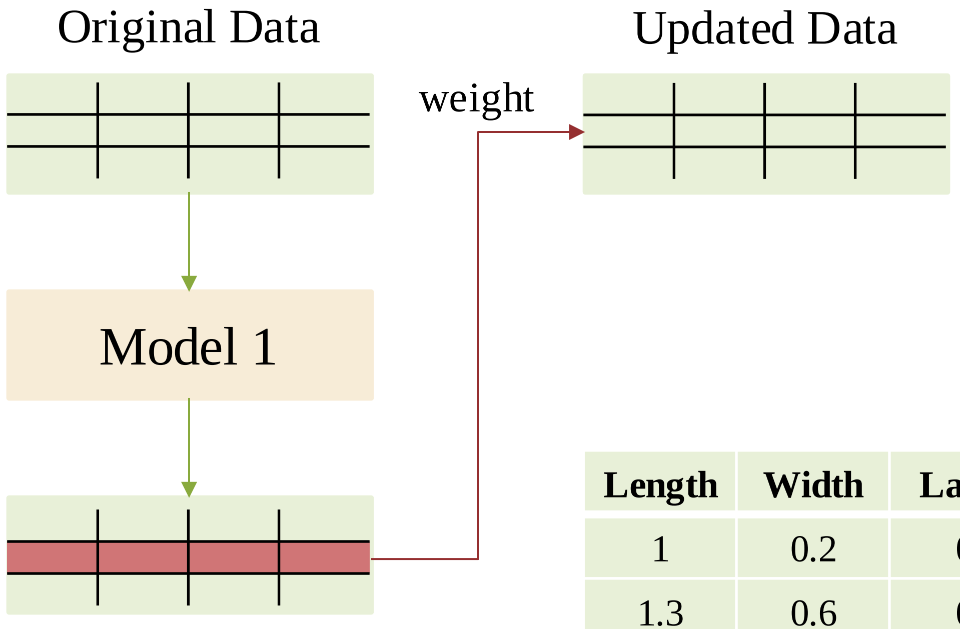
$$w_{new} = w * e^{-\alpha} = \frac{1}{6} * e^{-0.8} = 0.07$$

Length	Width	Label	Weight
1	0.2	0	1/6
1.3	0.6	0	1/6
0.9	0.7	0	1/6
1.7	0.5	1	1/6
1.8	0.9	1	1/6
1.2	1.3	1	1/6

Length	Width	Label	Weight
1	0.2	0	0.07
1.3	0.6	0	0.07
0.9	0.7	0	0.07
1.7	0.5	1	0.37
1.8	0.9	1	0.07
1.2	1.3	1	0.07

AdaBoost

Update weights of samples



$\alpha_1, h_1(x)$

$\alpha_1 = 0.8$

For incorrect predictions

$$w_{new} = w * e^{\alpha} = \frac{1}{6} * e^{0.8} = 0.37$$

For correct predictions

$$w_{new} = w * e^{-\alpha} = \frac{1}{6} * e^{-0.8} = 0.07$$

Length	Width	Label	Weight
1	0.2	0	1/6
1.3	0.6	0	1/6
0.9	0.7	0	1/6
1.7	0.5	1	1/6
1.8	0.9	1	1/6
1.2	1.3	1	1/6

Normalization

Length	Width	Label	Weight
1	0.2	0	0.07
1.3	0.6	0	0.07
0.9	0.7	0	0.07
1.7	0.5	1	0.37
1.8	0.9	1	0.07
1.2	1.3	1	0.07

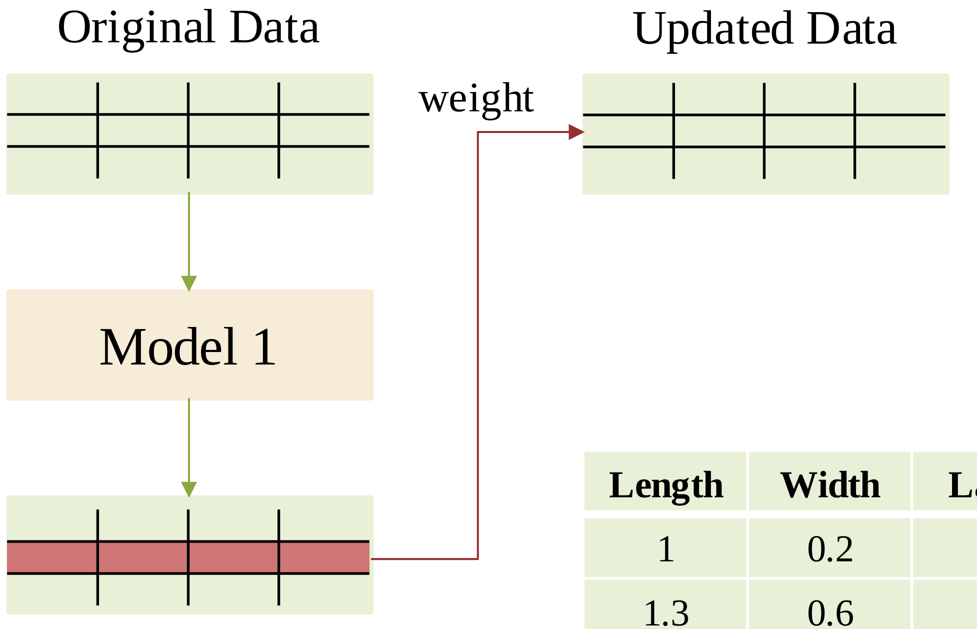
Total new weights = 0.72

0.07*1/0.72 = 0.1
0.37*1/0.72 = 0.5

Length	Width	Label	Weight
1	0.2	0	0.1
1.3	0.6	0	0.1
0.9	0.7	0	0.1
1.7	0.5	1	0.5
1.8	0.9	1	0.1
1.2	1.3	1	0.1

AdaBoost

Create new data



$\alpha_1, h_1(x)$

$\alpha_1 = 0.8$

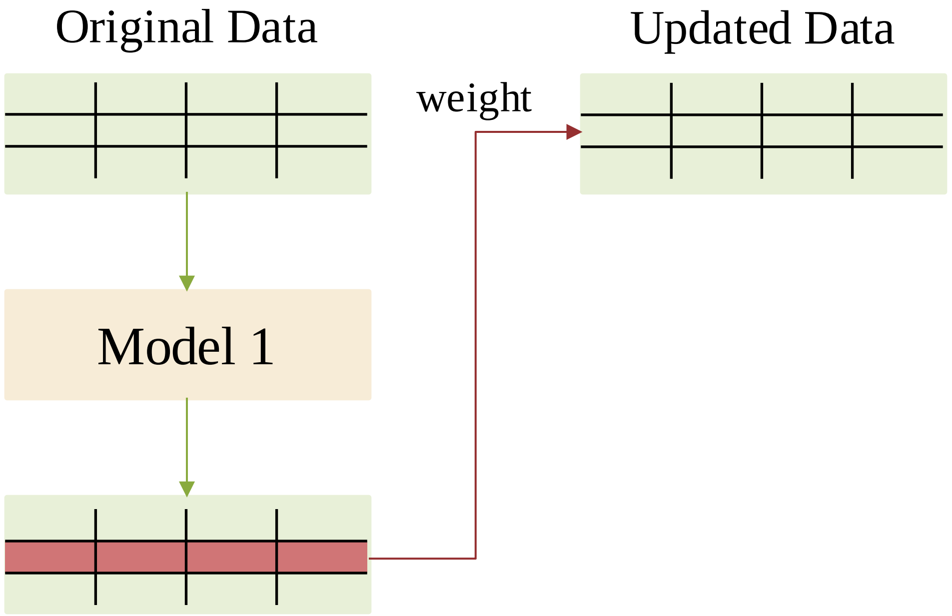
Length	Width	Label	Weight
1	0.2	0	1/6
1.3	0.6	0	1/6
0.9	0.7	0	1/6
1.7	0.5	1	1/6
1.8	0.9	1	1/6
1.2	1.3	1	1/6

Length	Width	Label	Weight	Random
1	0.2	0	0.1	0.0 => 0.1
1.3	0.6	0	0.1	0.1 => 0.2
0.9	0.7	0	0.1	0.2 => 0.3
1.7	0.5	1	0.5	0.3 => 0.8
1.8	0.9	1	0.1	0.8 => 0.9
1.2	1.3	1	0.1	0.9 => 1.0

	Length	Width	Label	Weight
0.1	1	0.2	0	0.1
0.2	1.3	0.6	0	0.1
0.7	1.7	0.5	1	0.5
0.9	1.8	0.9	1	0.1
0.6	1.7	0.5	1	0.5
0.4	1.7	0.5	1	0.5

AdaBoost

Refresh new weights



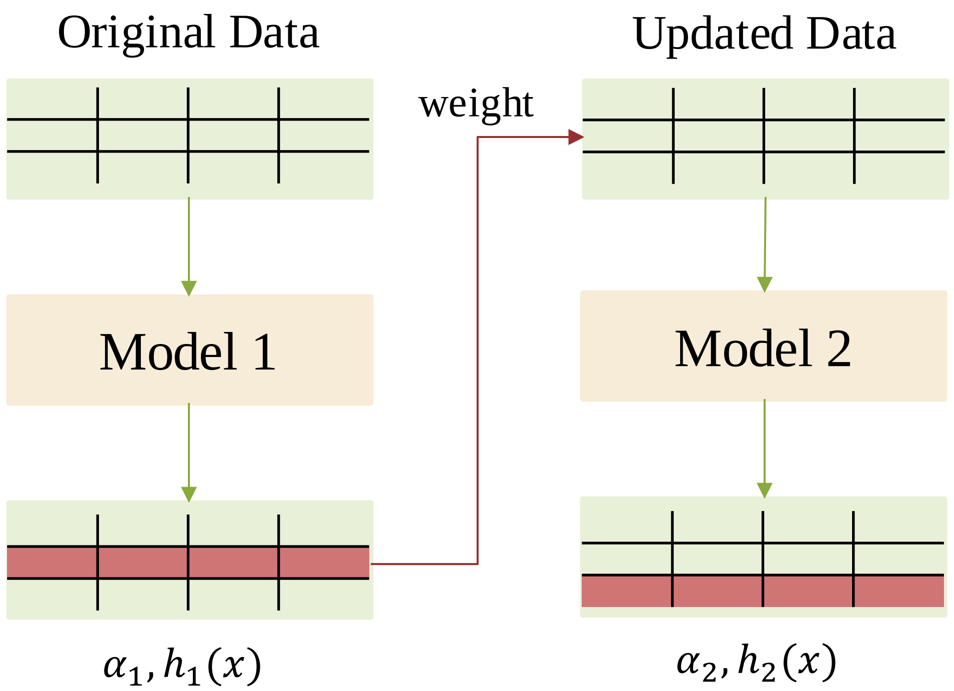
$\alpha_1, h_1(x)$

$\alpha_1 = 0.8$

Length	Width	Label	Weight
1	0.2	0	1/6
1.3	0.6	0	1/6
1.7	0.5	1	1/6
1.8	0.9	1	1/6
1.7	0.5	1	1/6
1.7	0.5	1	1/6

AdaBoost

Fitting model 2 (Round 2)



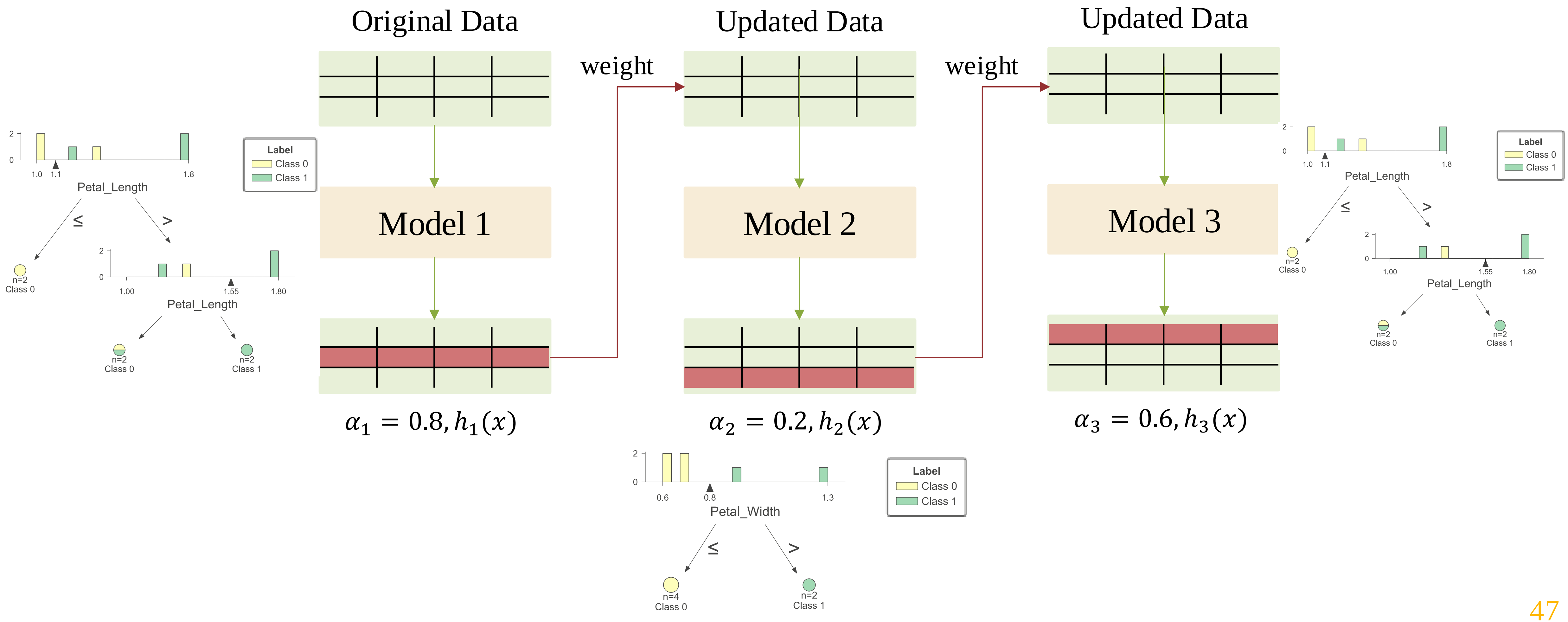
$\alpha_1 = 0.8$

Length	Width	Label	Weight
1	0.2	0	1/6
1.3	0.6	0	1/6
1.7	0.5	1	1/6
1.8	0.9	1	1/6
1.7	0.5	1	1/6
1.7	0.5	1	1/6

AI VIET NAM
@aivietnam.edu.vn

AdaBoost

Training Phase



AdaBoost

Inference Phase

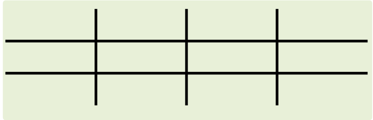
Test sample:
Petal_Length = 1.6
Petal_Width = 0.8

Model 1	Model 2	Model 3
1	0	1
$\alpha_1 = 0.8$	$\alpha_2 = 0.2$	$\alpha_3 = 0.6$

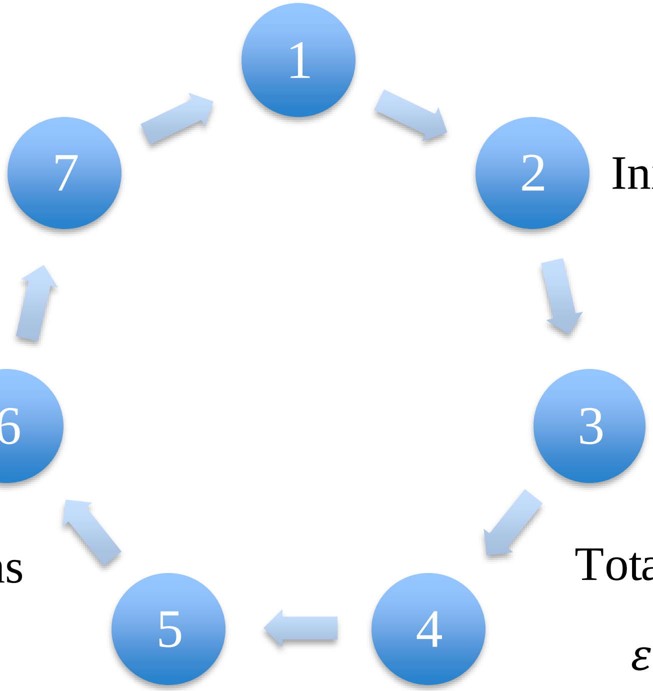
$$f(x) = \sum \alpha_t h_t(x) = 0.8 * 1 + 0.2 * 0 + 0.6 * 1 = 1.4 \Rightarrow \mathbf{1}$$

AdaBoost

Summary



Get subset training data



Initial sample weight = 1/N

Fitting Model

Total error (Error rate)

$$\varepsilon_1 = \sum_{incorrect} w_i$$

Weight of weak learner

$$\alpha_1 = \frac{1}{2} * ln \frac{(1 - \varepsilon_1)}{\varepsilon_1}$$

For incorrect predictions

$$w_{new} = w * e^{\alpha}$$

For correct predictions

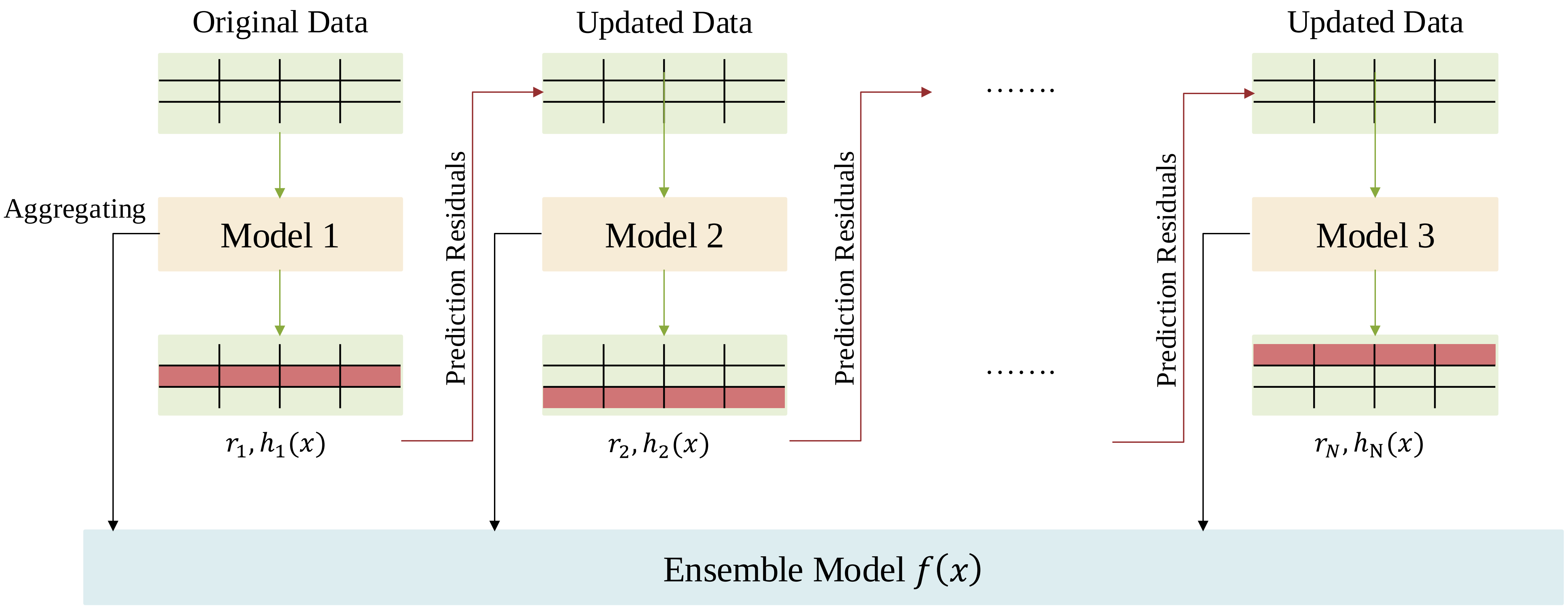
$$w_{new} = w * e^{-\alpha}$$

Normalization

Sampling Probability

Boosting Methods

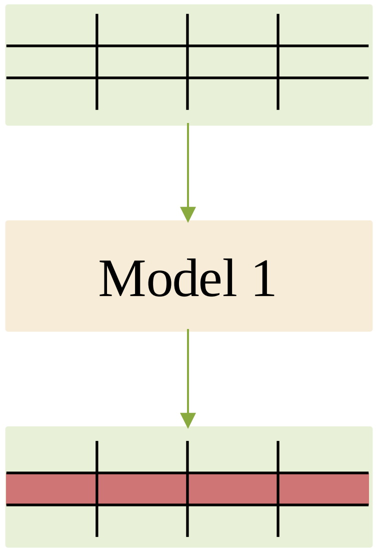
Gradient Boosting



Gradient Boosting

Gradient Boosting

Original Data



$r_1 = y_1 - \hat{y}_1, h_1(x)$

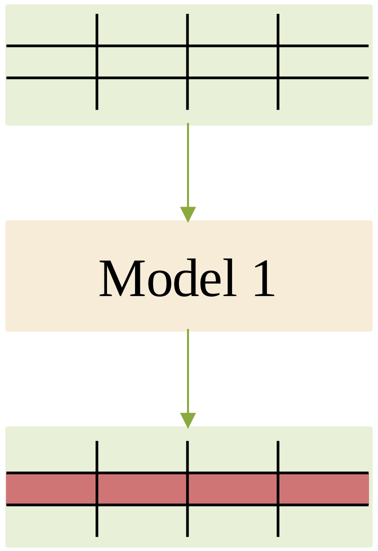
Experience	Salary
1	0
3	20
3.5	30
4	35
5.5	60

Gradient Boosting

Initial Model Prediction

$$\mu = \frac{1}{N}\sum y_i = \frac{0 + 20 + 30 + 35 + 60}{5} = 29$$

Original Data



$r_1 = y_1 - \hat{y}_1, h_1(x)$

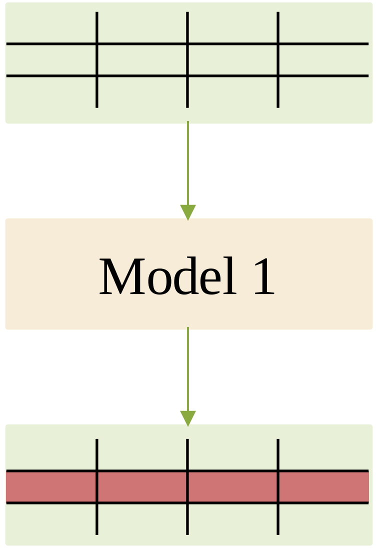
Experience	Salary	Initial Prediction
1	0	29
3	20	29
3.5	30	29
4	35	29
5.5	60	29

Gradient Boosting

Calculating Residuals

$$\mu = \frac{1}{N}\sum y_i \qquad r = y - \hat{y} = y - \mu$$

Original Data



$r_1 = y_1 - \hat{y}_1, h_1(x)$

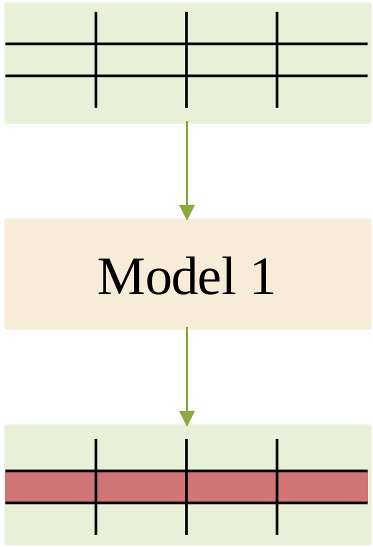
Experience	Salary	Initial Prediction	Residual 1
1	0	29	- 29
3	20	29	- 9
3.5	30	29	1
4	35	29	6
5.5	60	29	31

Gradient Boosting

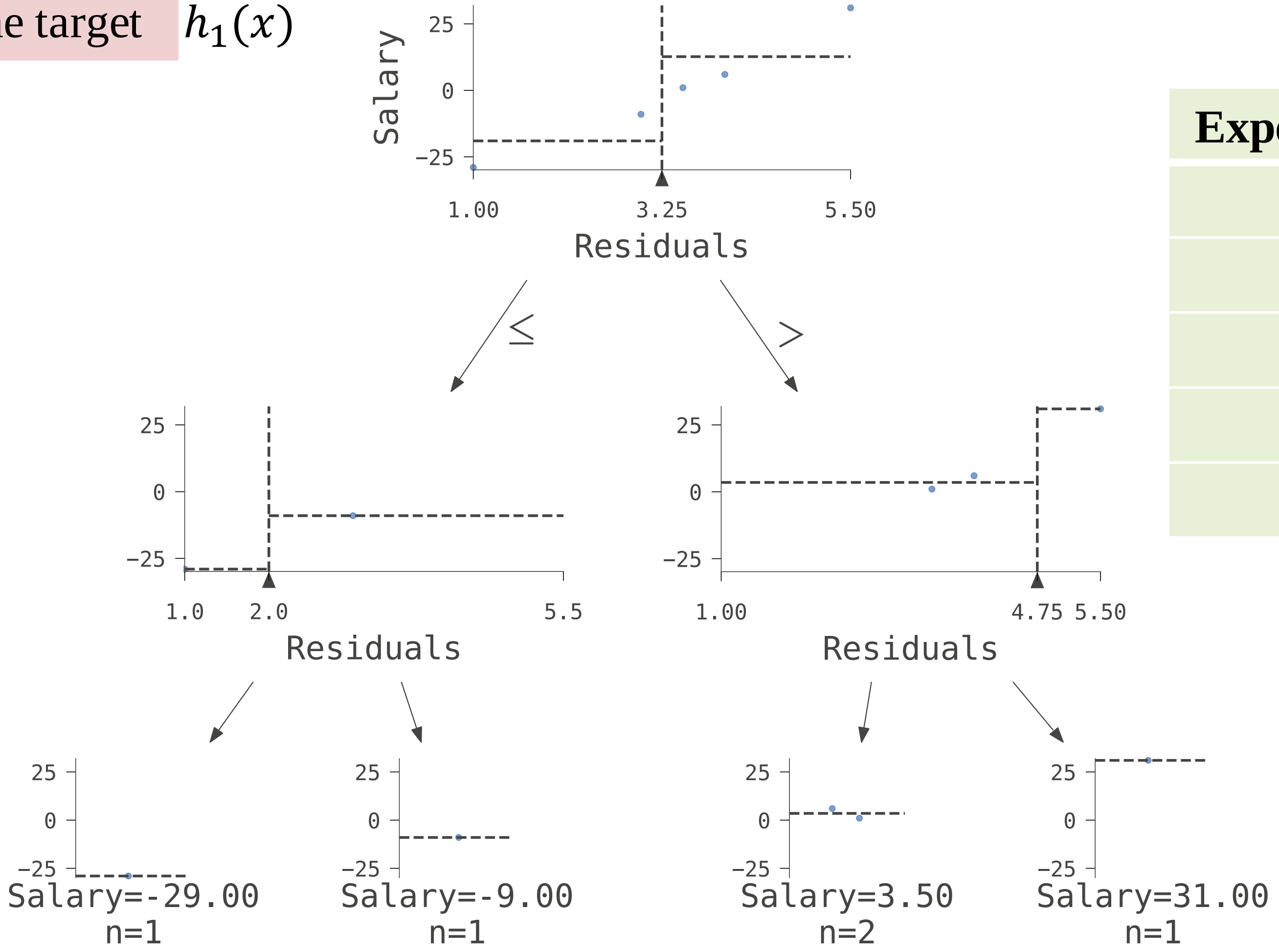
Building a Decision Tree

Using the residuals as the target $h_1(x)$

Original Data



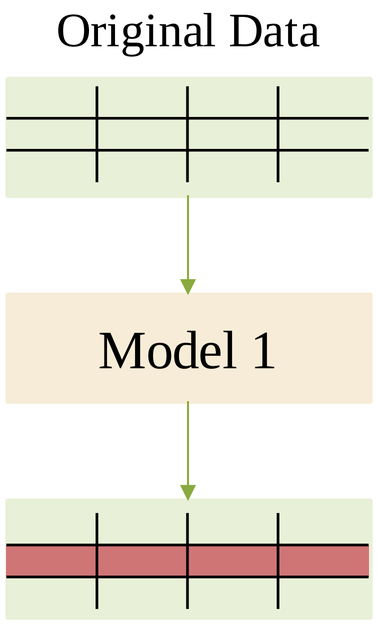
$r_1 = y_1 - \hat{y}_1, h_1(x)$



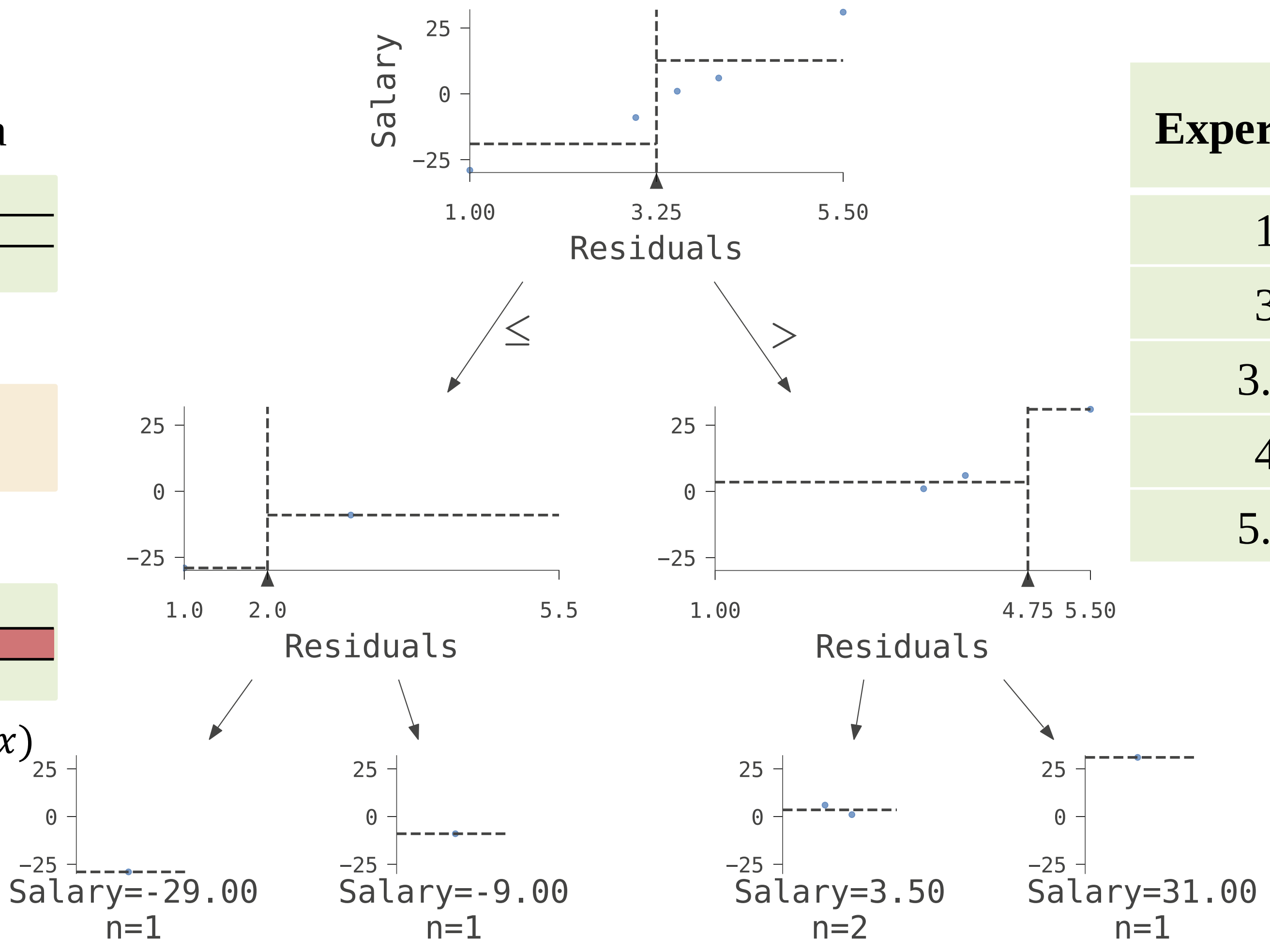
Experience	Residual 1
1	- 29
3	- 9
3.5	1
4	6
5.5	31

Gradient Boosting

Compute Decision Tree Output



$r_1 = y_1 - \hat{y}_1, h_1(x)$



Experience	Residual 1	Predicted Residuals 1
1	- 29	- 29
3	- 9	- 9
3.5	1	3.5
4	6	3.5
5.5	31	31

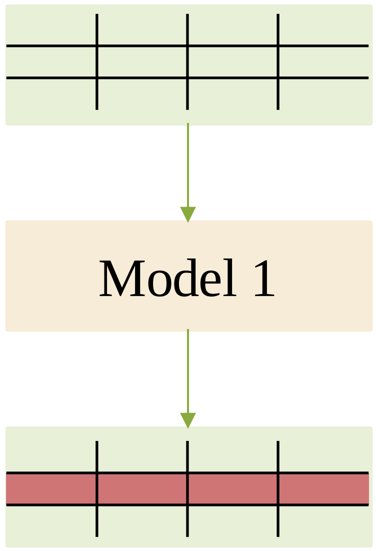
$$\hat{r} = h_1(x)$$

Gradient Boosting

Update Predictions

$$y_{new} = \hat{y} + lr * \hat{r}$$
$$lr = 0.1$$

Original Data



$r_1 = y_1 - \hat{y}_1, h_1(x)$

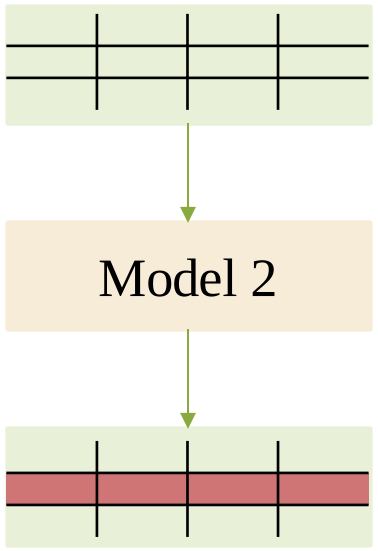
Experience	Salary	Initial Prediction	Residual 1	Predicted Residuals 1	Prediction 1
1	0	29	- 29	- 29	26.1
3	20	29	- 9	- 9	28.1
3.5	30	29	1	3.5	29.35
4	35	29	6	3.5	29.35
5.5	60	29	31	31	32.1

Gradient Boosting

Calculating Residuals for Round 2

$$r_2 = y - y_{new}$$

Updated Data

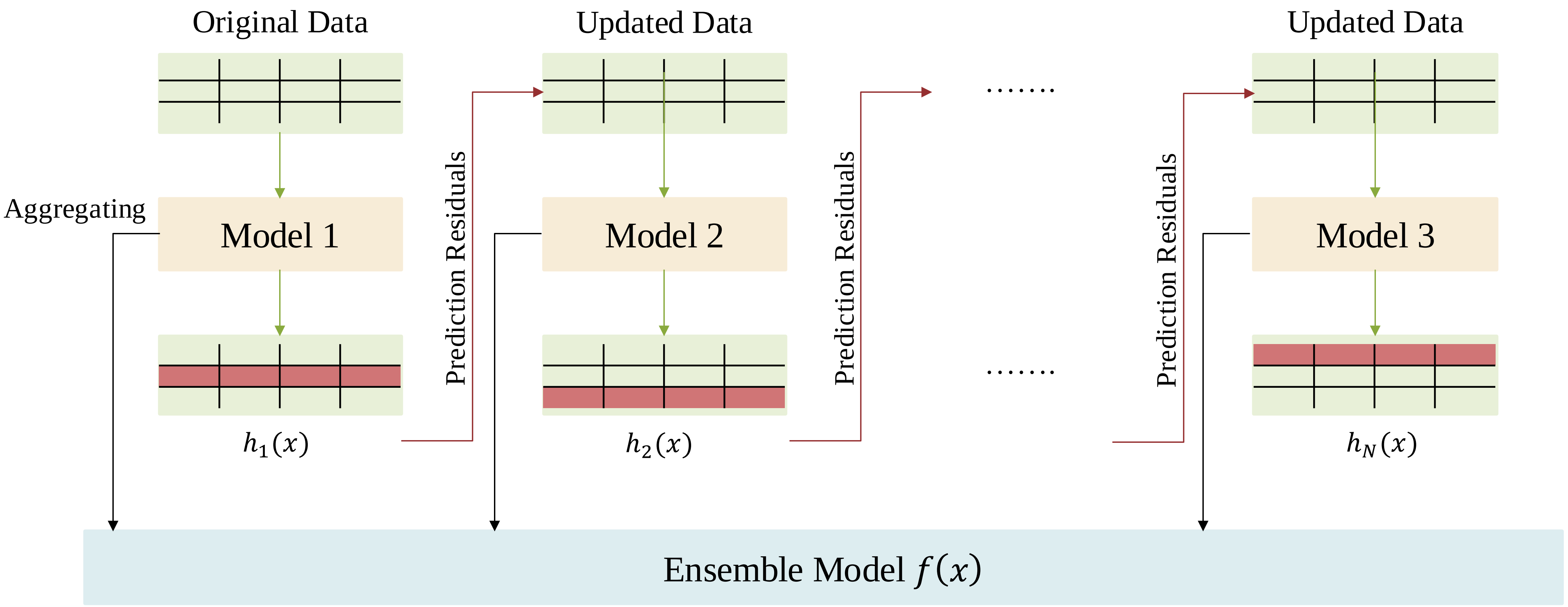


Experience	Salary	Initial Prediction	Residual 1	Predicted Residuals 1	Prediction 1	Residual 2
1	0	29	- 29	- 29	26.1	- 26.1
3	20	29	- 9	- 9	28.1	- 8.1
3.5	30	29	1	3.5	29.35	0.65
4	35	29	6	3.5	29.35	5.65
5.5	60	29	31	31	32.1	27.9

Gradient Boosting

Training Pharse

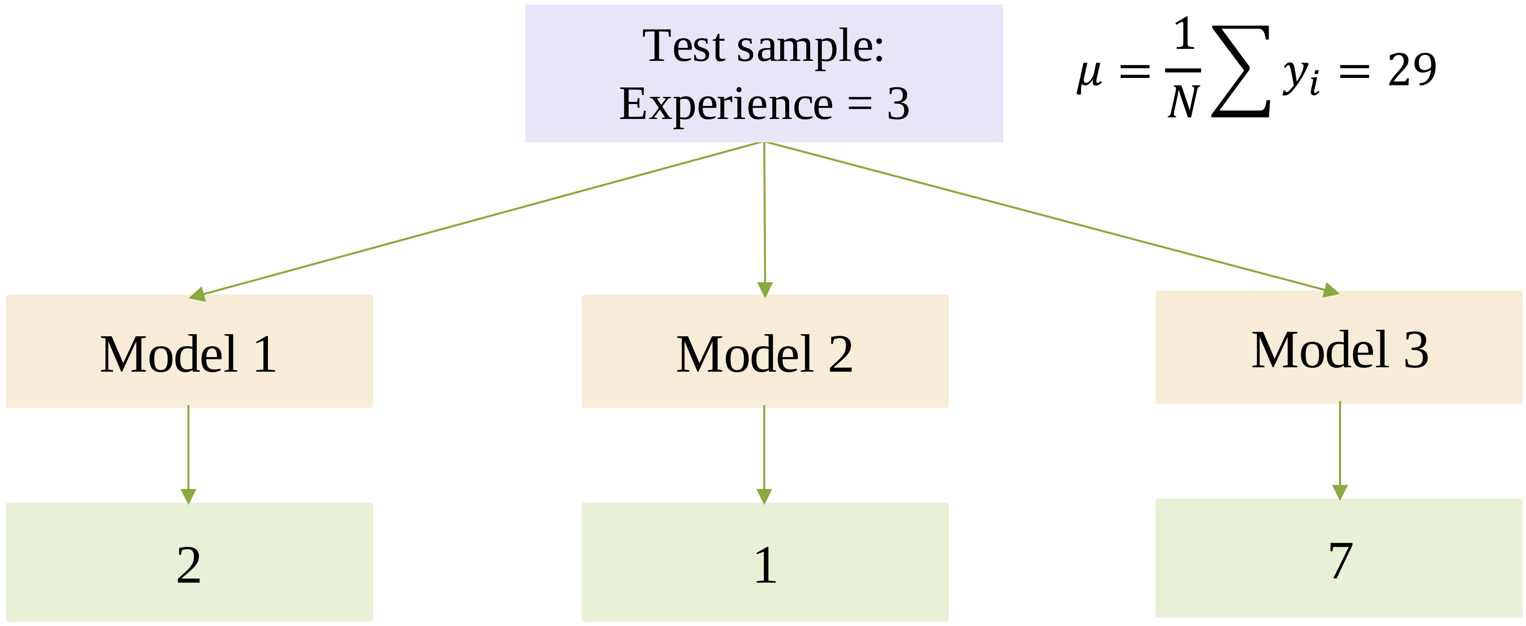
$$\mu = \frac{1}{N}\sum y_i$$



Gradient Boosting

Inference Phase

$$\mu = \frac{1}{N}\sum y_i = 29$$

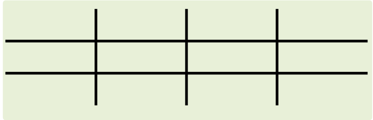


$$f(x) = \mu + lr * \sum h_t(x) = 29 + 0.1 * (2 + 1 + 7) = \mathbf{30}$$

$lr = 0.1$

Gradient Boosting

Summary



Initial Model Prediction $\mu = \frac{1}{N}\sum y_i$

Calculate Residuals

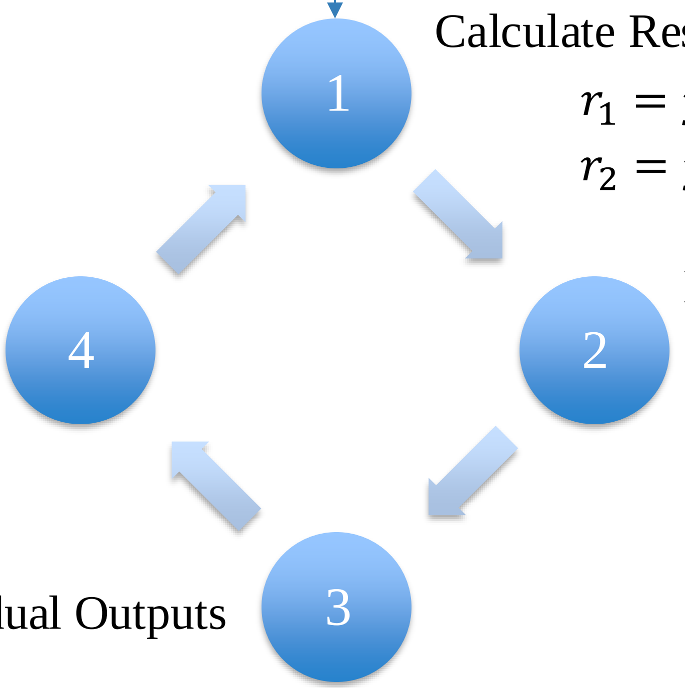
$$r_1 = y - \hat{y} = y - \mu$$
$$r_2 = y - y_{new}$$

Build a Decision Tree $h(x)$

Compute Residual Outputs

Update Predictions

$$y_{new} = \hat{y} + lr * \hat{r}$$
$$lr = 0.1$$



QUIZ TIME

AI VIET NAM
@aivietnam.edu.vn

Outline

SECTION 1
Ensemble Learning

SECTION 2
Bagging Methods

SECTION 3
Boosting Methods

SECTION 4
Implementation

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   price             545 non-null    int64
 1   area              545 non-null    int64
 2   bedrooms          545 non-null    int64
 3   bathrooms         545 non-null    int64
 4   stories           545 non-null    int64
 5   mainroad          545 non-null    object
 6   guestroom         545 non-null    object
 7   basement          545 non-null    object
 8   hotwaterheating   545 non-null    object
 9   airconditioning   545 non-null    object
 10  parking           545 non-null    int64
 11  prefarea          545 non-null    object
 12  furnishingstatus  545 non-null    object
dtypes: int64(6), object(7)
memory usage: 55.5+ KB
```

Implementation

Housing Dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   price             545 non-null    int64
 1   area              545 non-null    int64
 2   bedrooms          545 non-null    int64
 3   bathrooms         545 non-null    int64
 4   stories           545 non-null    int64
 5   mainroad          545 non-null    object
 6   guestroom         545 non-null    object
 7   basement          545 non-null    object
 8   hotwaterheating   545 non-null    object
 9   airconditioning   545 non-null    object
 10  parking           545 non-null    int64
 11  prefarea          545 non-null    object
 12  furnishingstatus  545 non-null    object
dtypes: int64(6), object(7)
memory usage: 55.5+ KB
```

Implementation

Categorical Label Encoding

```
['mainroad',
 'guestroom',
 'basement',
 'hotwaterheating',
 'airconditioning',
 'prefarea',
 'furnishingstatus']
```

```python
categorical_cols = df.select_dtypes(include=['object']).columns.to_list()

ordinal_encoder = OrdinalEncoder()
encoded_categorical_cols = ordinal_encoder.fit_transform(
    df[categorical_cols]
)
encoded_categorical_df = pd.DataFrame(
    encoded_categorical_cols,
    columns=categorical_cols
)
numerical_df = df.drop(categorical_cols, axis=1)
encoded_df = pd.concat(
    [numerical_df, encoded_categorical_df], axis=1
)
```

AI VIET NAM
@aivietnam.edu.vn

Implementation

Train Test Split

```
X, y = dataset_arr[:, 1:], dataset_arr[:, 0]
```

```
test_size = 0.3
random_state = 1
is_shuffle = True
X_train, X_val, y_train, y_val = train_test_split(
    X, y,
    test_size=test_size,
    random_state=random_state,
    shuffle=is_shuffle
)
```

Implementation

Training & Evaluation

```python
regressor = RandomForestRegressor(
    random_state=random_state
)
regressor.fit(X_train, y_train)
```

```
▾ RandomForestRegressor
RandomForestRegressor(random_state=1)
```

```python
y_pred = regressor.predict(X_val)
```

```python
mae = mean_absolute_error(y_val, y_pred)
mse = mean_squared_error(y_val, y_pred)

print('Evaluation results on validation set:')
print(f'Mean Absolute Error: {mae}')
print(f'Mean Squared Error: {mse}')
```

```
Evaluation results on validation set:
Mean Absolute Error: 0.46093873321571177
Mean Squared Error: 0.37944418523089524
```

```python
regressor = AdaBoostRegressor(
    random_state=random_state
)
regressor.fit(X_train, y_train)
```

```
▾ AdaBoostRegressor
AdaBoostRegressor(random_state=1)
```

```python
y_pred = regressor.predict(X_val)
```

```python
mae = mean_absolute_error(y_val, y_pred)
mse = mean_squared_error(y_val, y_pred)

print('Evaluation results on validation set:')
print(f'Mean Absolute Error: {mae}')
print(f'Mean Squared Error: {mse}')
```

```
Evaluation results on validation set:
Mean Absolute Error: 0.567680019897059
Mean Squared Error: 0.5739244030038942
```

```python
regressor = GradientBoostingRegressor(
    random_state=random_state
)
regressor.fit(X_train, y_train)
```

```
▾ GradientBoostingRegressor
GradientBoostingRegressor(random_state=1)
```

```python
y_pred = regressor.predict(X_val)
```

```python
mae = mean_absolute_error(y_val, y_pred)
mse = mean_squared_error(y_val, y_pred)

print('Evaluation results on validation set:')
print(f'Mean Absolute Error: {mae}')
print(f'Mean Squared Error: {mse}')
```

```
Evaluation results on validation set:
Mean Absolute Error: 0.4516626127750995
Mean Squared Error: 0.39610445936979427
```

Summary

Ensemble Learning

- Introduction
- Ensemble Methods
- Learning Ensembles
- Constructing Ensembles

Boosting

- Boosting Methods
- AdaBoost
- Gradient Boosting
- Calculate Weight

Bagging

- Bootstrapping
- Decision Tree
- Random Forest
- Extract Subset Training Data

Implementation

- Housing Dataset
- Random Forest
- AdaBoost
- Gradient Boosting
- Sklearn

Thanks!

Any questions?