Korea Aerospace University

2AI4203: Machine Learning

Lec05: Tree-Based Machine Learning Models

Jiyoon Shin

KAU College of AI Convergence

Sep. 30, 2026

Recap:

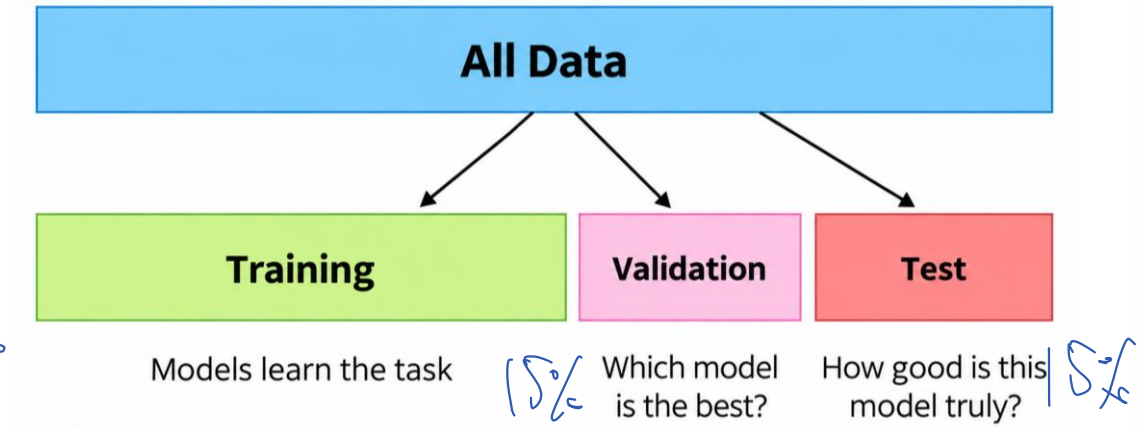
Model Evaluation and Generalization

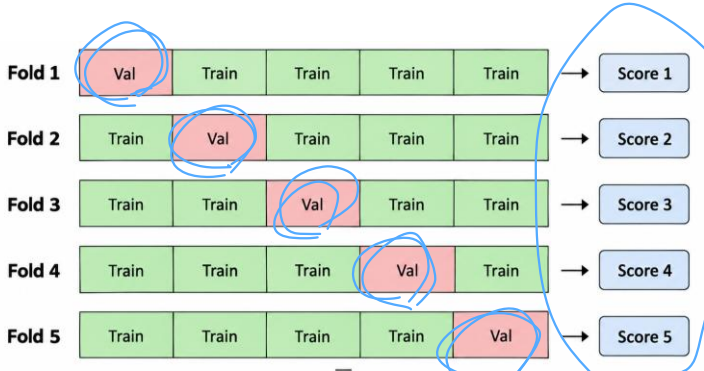
검증

일반화

Model Evaluation (검증)

- A model should perform well on unseen data, not only on training data
- Data splits
  - Training set: learn model parameters
  - Validation set: select models and hyperparameters
  - Test set: final evaluation
- K-fold cross-validation: evaluate across multiple train-validation splits





Performance Metrics (계속)

- Regression
  - MAE: average absolute error
  - MSE: average squared error
  - RMSE: square root of MSE

$y - \hat{y}$ → $|\cdot|$ → $\frac{1}{N}\Sigma$

error → absolute → mean

error → squared → mean

$$\sqrt{\frac{1}{N}\Sigma(y-\hat{y})^2}$$

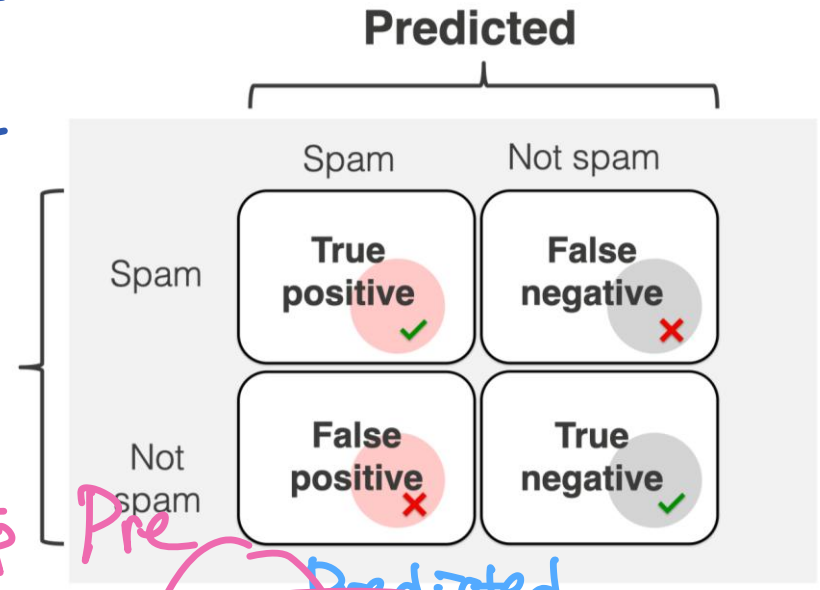
- Classification
  - Accuracy: overall correctness
  - Precision: correctness among predicted positives
  - Recall: proportion of actual positives detected
  - F1 Score: balance precision and recall

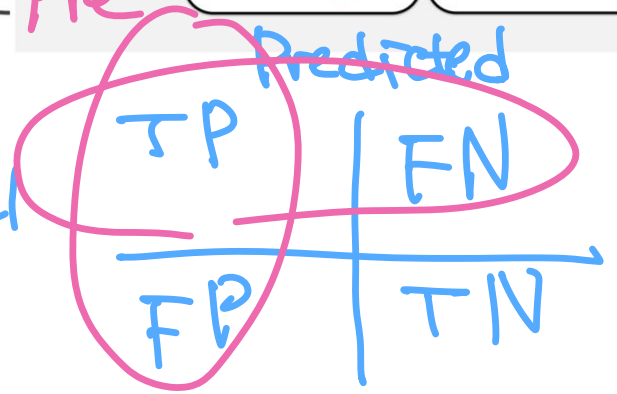
$$\frac{TP+TN}{TP+TN+FP+FN}$$

$$\frac{TP}{TP+FP}$$

$$\frac{TP}{TP+FN}$$

$$2 \cdot \frac{Pre \cdot Re}{Pre+Re}$$

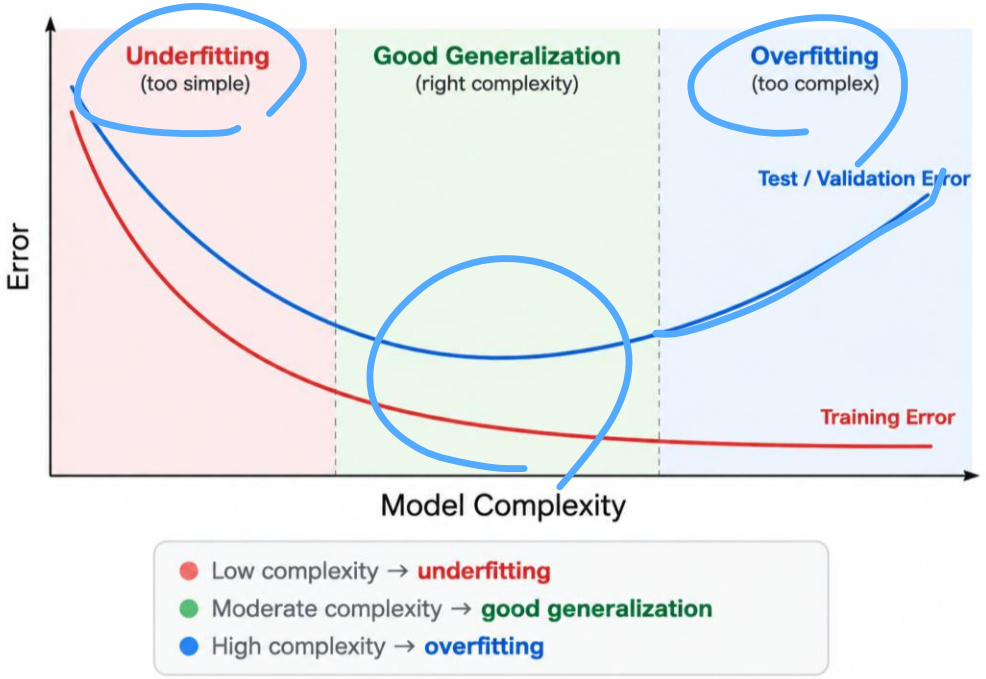




Generalization 일반화

- Generalization: performance on unseen data

- Underfitting: model is too simple
  - → high training and validation errors
- Overfitting: model fits training data too closely
  - → low training error, high validation error



- Model complexity affects the balance between underfitting and overfitting
  - High bias → often underfitting
  - High variance → often overfitting

편향

다양성

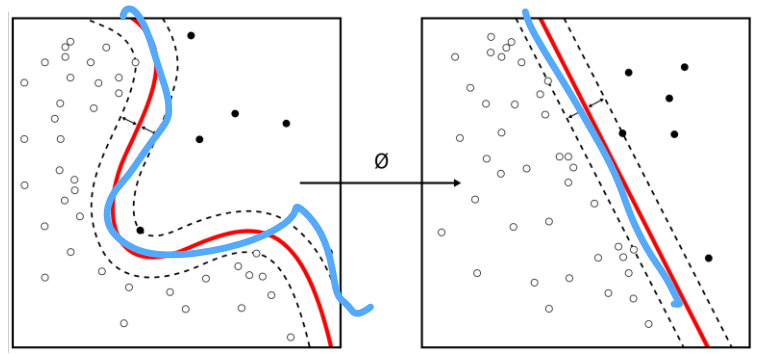
Improving Generalization

- Control model complexity

- Regularization: penalize overly complex solutions

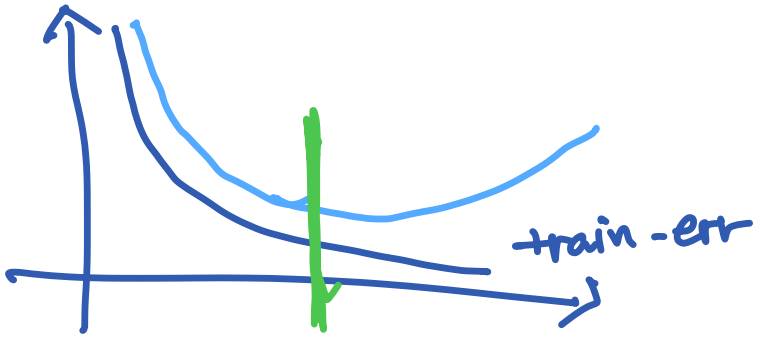
$$L_{total} = L_{data} + \lambda R(\theta)$$

  - L1: encourages sparse weights
  - L2: encourages small weights



- Early stopping: stop when validation performance stops improving

- Use more training data when possible



Korea Aerospace University

Part 1. Decision Trees

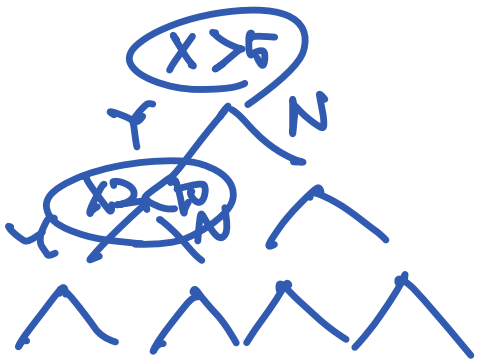
(기본 classic ML
tabular data

Linear Models vs. Decision Trees

- Linear models
  - Weighted combination of input features
  - Use a single global linear function
  - Produce linear relationships / decision boundaries

$$z = \mathbf{w}^T \mathbf{x} + b$$

- Decision trees
  - Sequence of simple decision rules
  - Capture nonlinear patterns
  - Create flexible decision boundaries

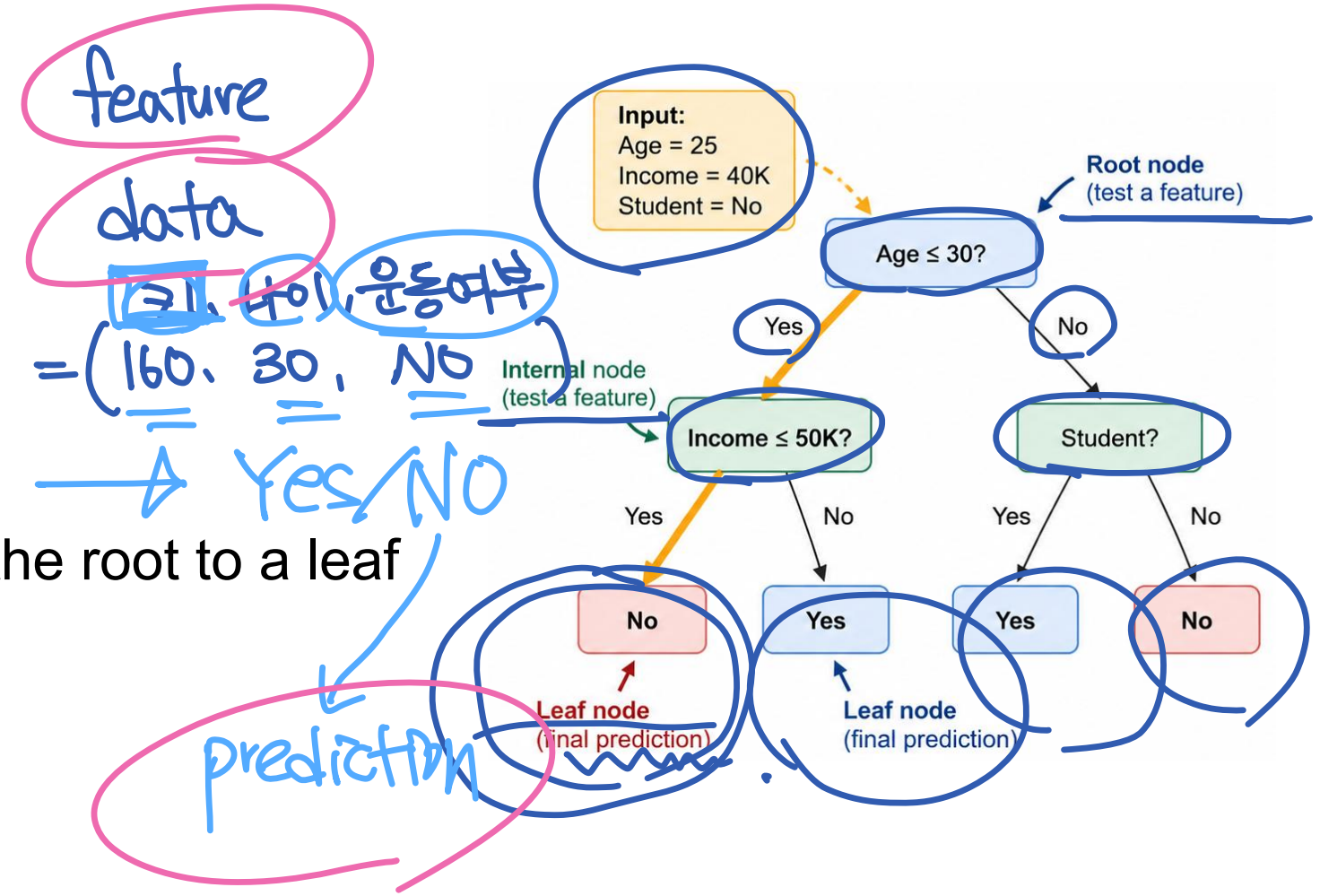


Decision Trees

- A decision tree makes predictions through a sequence of decision rules

- Internal node: tests a feature
- Branch: outcome of a test
- Leaf node: final prediction

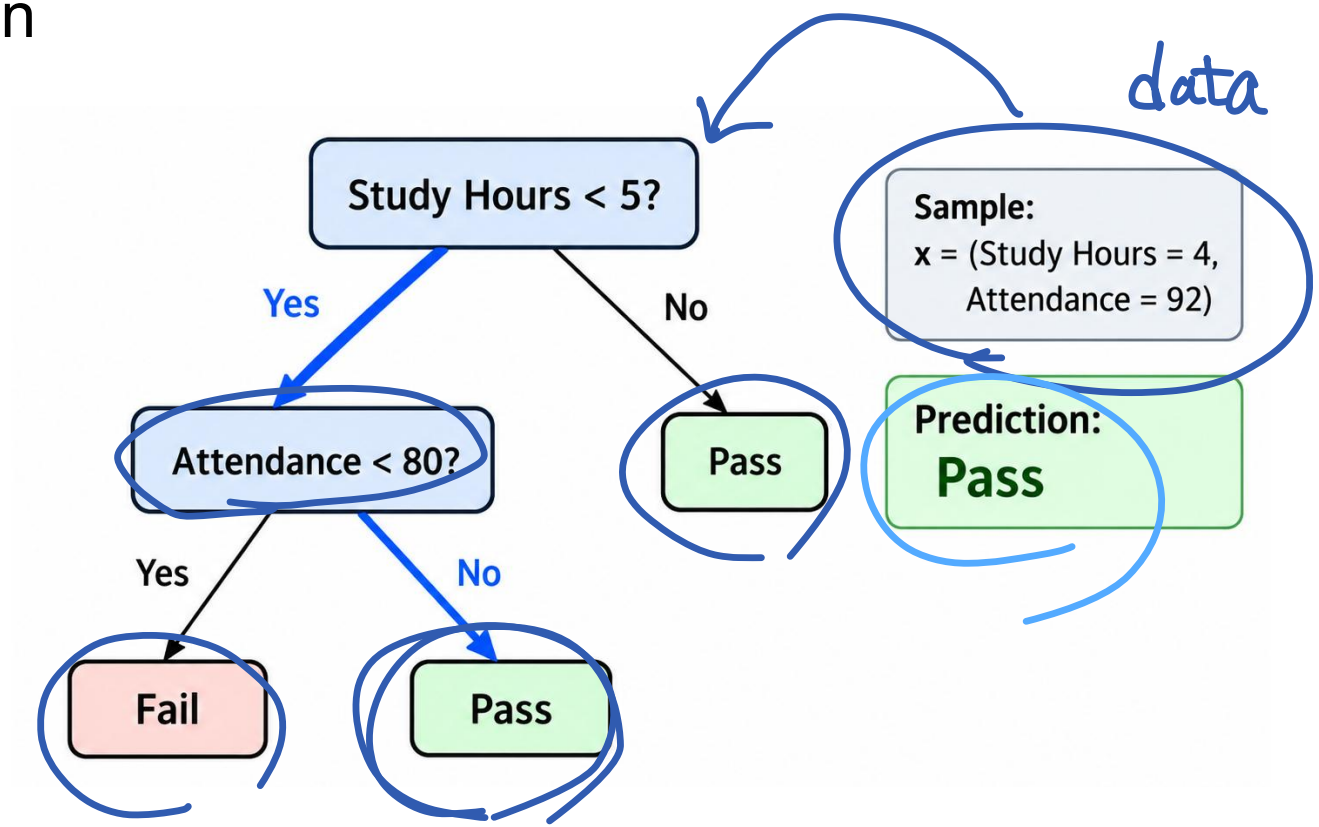
- An input follows one path from the root to a leaf



Classification Trees

- Classification trees predict categorical outputs
- Different inputs may follow different paths
- Each path ends at a class prediction

- Example
  - Goal: pass / fail
  - Features: study hours, attendance



Classification vs. Regression

- Classification

$$\hat{y} \in \{Pass, Fail\}$$

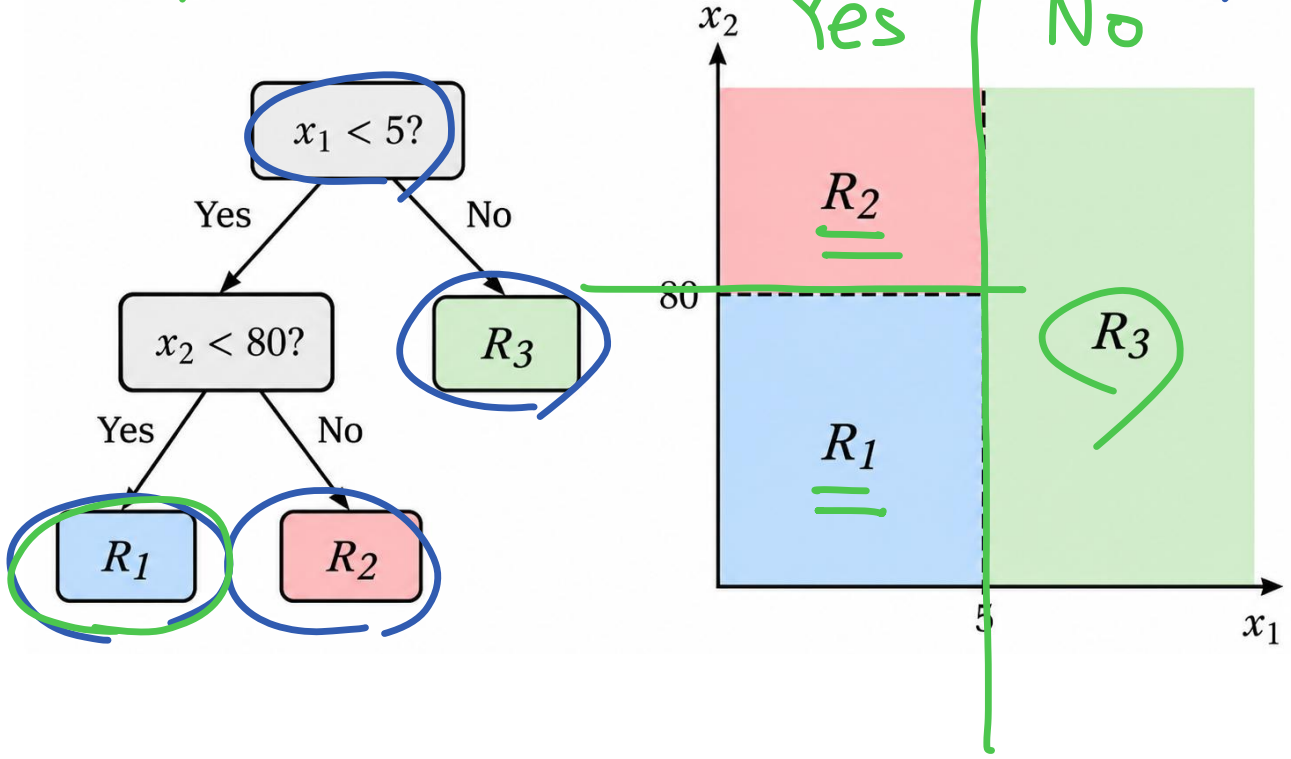
- Regression

$$\hat{y} = \$\,420\text{K}$$

	Classification	Regression
Target	Categorical	Continuous
Leaf prediction	Class / Probability	Numerical value
Example	Pass / Fail	House price

Recursive Partitioning

- Decision trees recursively divide the feature space into smaller regions
- Each leaf corresponds to one region of the feature space
- Axis-aligned splits can form nonlinear decision boundaries

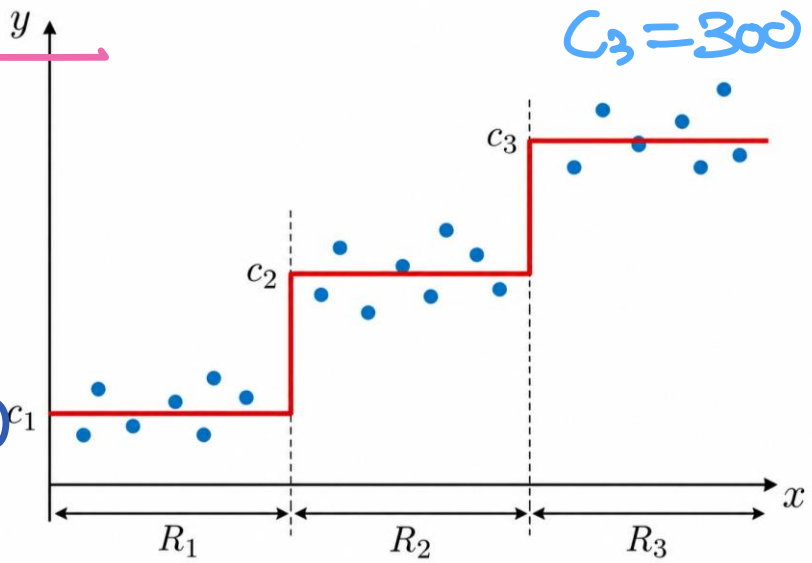


Piecewise-Constant Models

- The feature space is divided into regions $R_1, \ldots, R_M$
- Each leaf assigns one prediction to its corresponding region
- Therefore, a tree represents a piecewise-constant prediction function

$$\hat{f}(\mathbf{x}) = \sum_{m=1}^{M} c_m \mathbf{1}(\mathbf{x} \in R_m)$$

- R_m: region corresponding to leaf m
- c_m: prediction assigned to the region

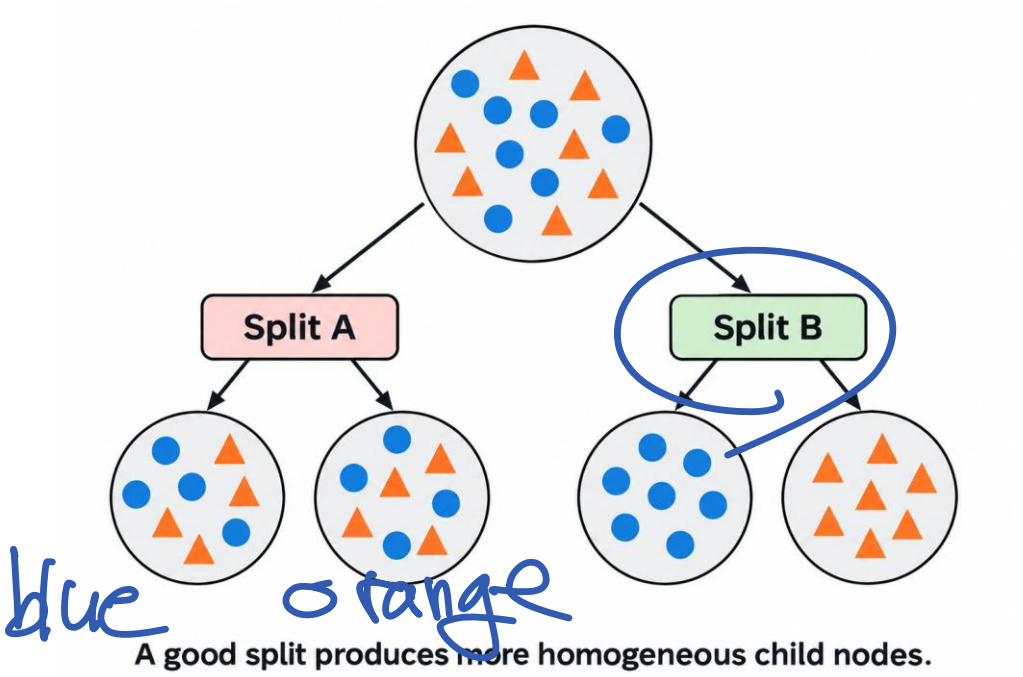


Choosing a Split

- Many features and thresholds may be available
- Different splits produce different groups of samples
- The tree must choose the most useful split

$\hat{y} = W^T X + b$

W, b를 잘 만들?



A good split produces more homogeneous child nodes.

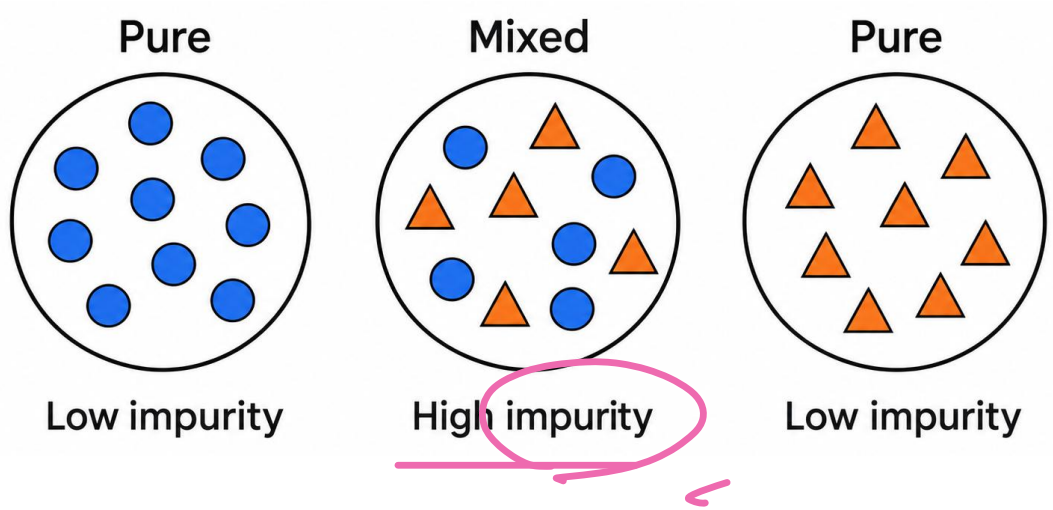
Korea Aerospace University

Part 2. Building Trees

Impurity

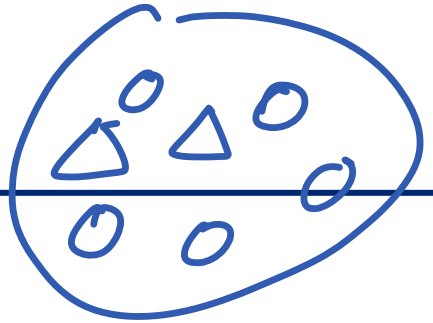
- A good split creates purer child nodes
- Node purity
  - Pure node: all samples belong to one class
  - Impure node: multiple classes are mixed

- We need a numerical measure of node impurity



Gini Impurity

이탈리아 통계학자 ....

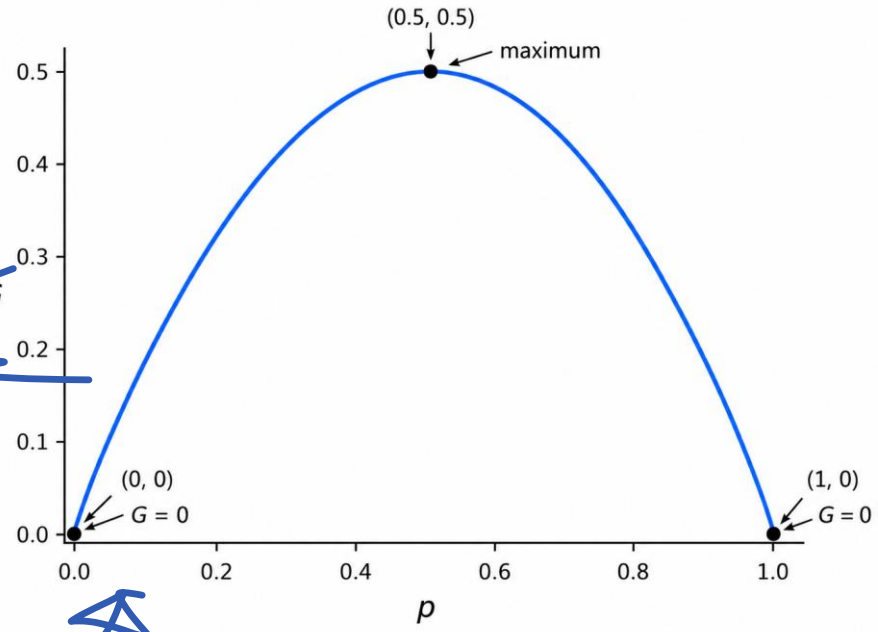


- General case

$$G = 1 - \sum_{k=1}^{K} p_k^2$$

= 1 − 0.8² − 0.2² = G

  - p_k: proportion of class k
  - $G = 0$: pure node
  - Larger G: greater class mixing



- Binary case

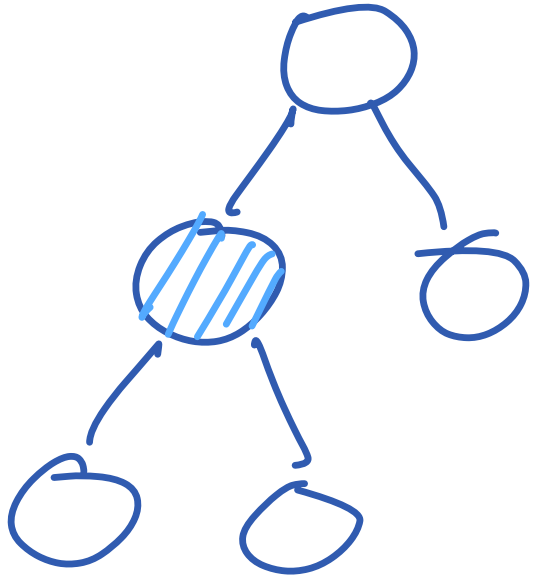
$$G = 1 - p^2 - (1 - p)^2$$

  - $p = 0$ or $1 \rightarrow G = 0$ pure
  - $p = 0.5 \rightarrow \mathrm{G} = 0.5$ (maximum)

Gini Impurity

- Example
  - 8 pass / 2 fail

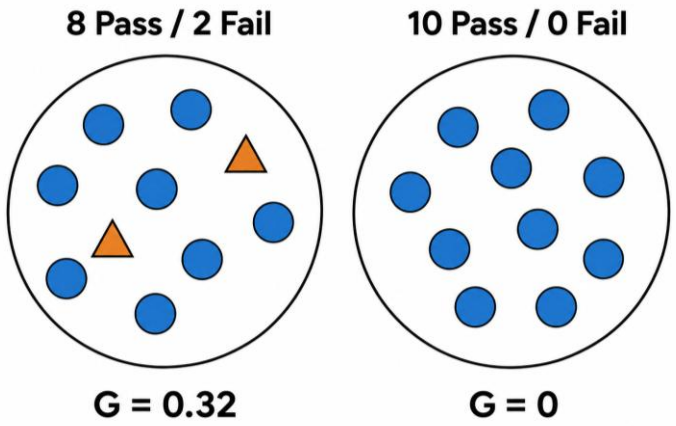
$$p_{Pass} = 0.8, \qquad p_{Fail} = 0.2$$
$$G = 1 - 0.8^2 - 0.2^2 = 0.32$$



  - 10 pass / 0 fail

$$G = 0$$

$1 - 1^2 - 0^2 = 0$



Entropy

- General case

$$H = -\sum_{k=1}^{K} p_k \log_2 p_k$$

  - $H = 0$: pure node
  - Larger H: greater uncertainty

- Binary case

$$H = -p\log_2 p - (1-p)\log_2(1-p)$$

  - $p = 0.5 \implies H = 1$
  - $p = 0 \text{ or } 1 \implies H = 0$

pure.

- Gini and entropy usually produce similar splits

p=0 or 1 G=0 H=0

p=0.5 G, H ↑.

Split Score

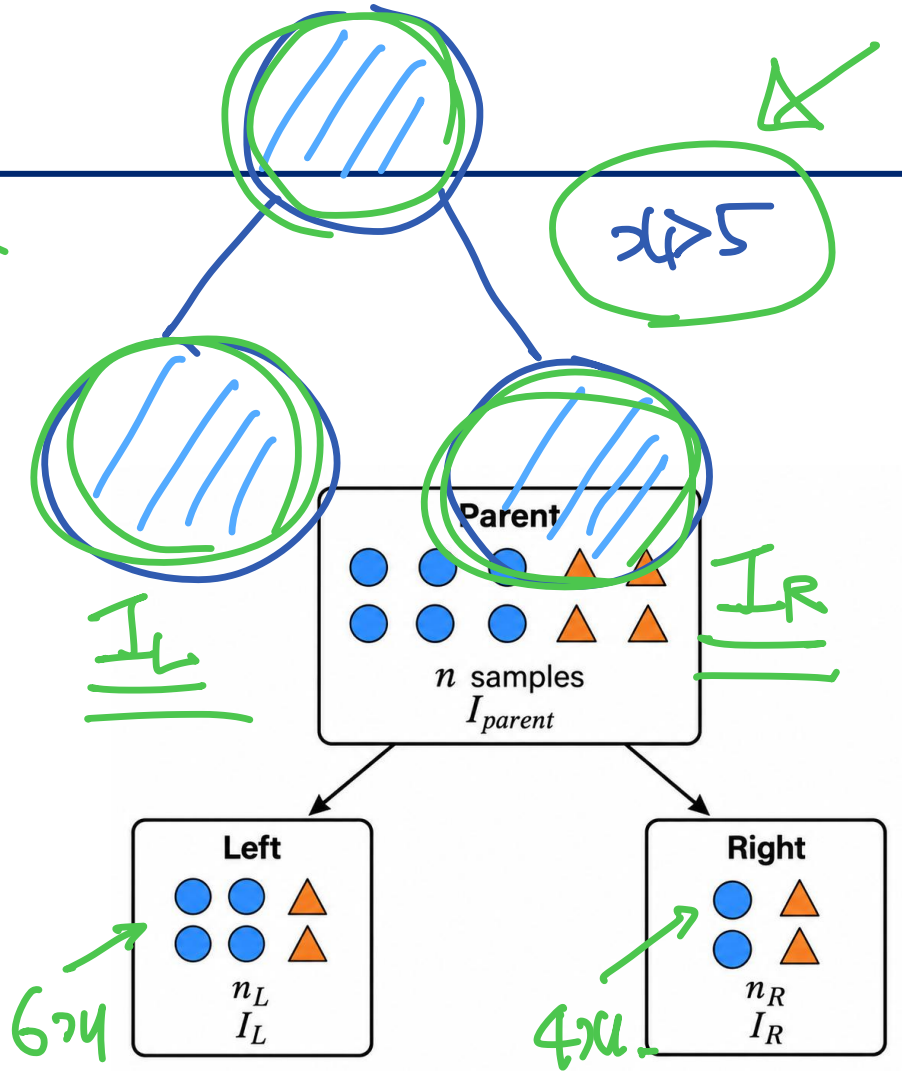
- Weighted impurity

$$I_{\text{split}} = \frac{n_L}{n} I_L + \frac{n_R}{n} I_R$$

  - Child impurities are weighted by their sizes

- Impurity reduction

$$\Delta I = I_{parent} - I_{split}$$



- Better split → larger ΔI
- Choose the split that maximizes ΔI

Split Score

- Example
  - Parent: 6 blue / 4 orange
  - After split
    - Left: 5 blue / 0 orange
    - Right: 1 blue / 4 orange

$$G_P = 1 - 0.6^2 - 0.4^2 = 0.48$$

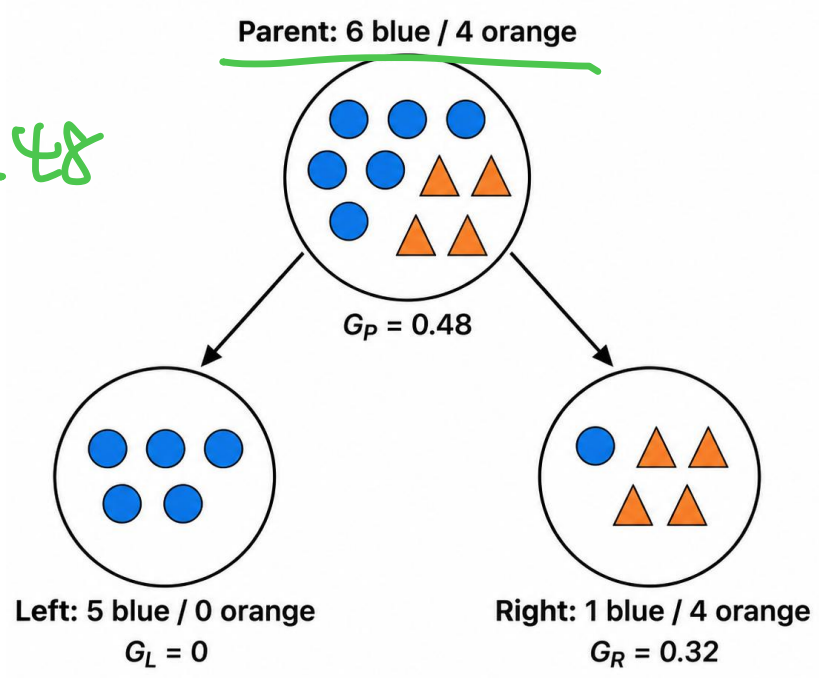
$$1 - \left(\frac{6}{10}\right)^2 - \left(\frac{4}{10}\right)^2$$

$$1 - 0.2^2 - 0.8^2$$

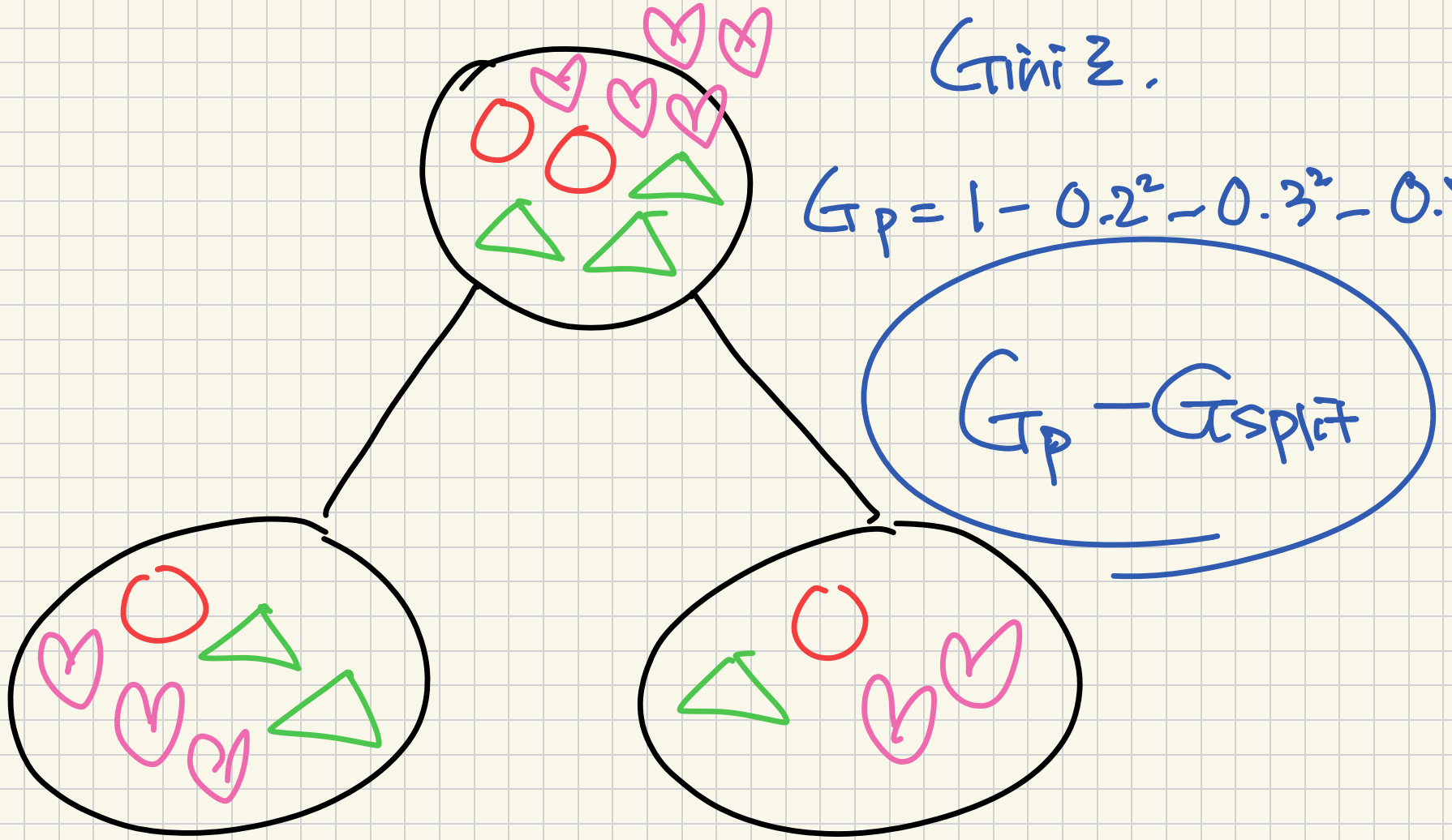
$$G_L = 0, \qquad G_R = 0.32$$

$$G_{split} = \frac{5}{10}(0) + \frac{5}{10}(0.32) = 0.16$$

$$\Delta G = 0.48 - 0.16 = 0.32$$



5 5



Gini 2.

$G_P = 1 - 0.2^2 - 0.3^2 - 0.5^2 = ?$

$G_P - G_{split}$

$G_L = 1 - \left(\frac{1}{6}\right)^2 - \left(\frac{2}{6}\right)^2 - \left(\frac{3}{6}\right)^2$

$G_R = 1 - \left(\frac{1}{4}\right)^2 - \left(\frac{1}{4}\right)^2 - \left(\frac{2}{4}\right)^2$

$G_{split} = \frac{6}{10} G_L + \frac{4}{10} G_R = ??$

Candidate Splits

- For a numerical feature
  - Sort the feature values
  - Consider thresholds between neighboring values
  - Example: $x = [20, 24, 30, 35]$
  - Possible thresholds: $22,\ 27,\ 32.5$



- Search over candidate splits

$$(j^*, t^*) = \arg\max_{j,t} \Delta I(j, t)$$

  - j: feature, t: threshold
  - Evaluate the impurity reduction for each feature and threshold
  - Select the best feature and threshold

키, 나이, 몸무게

민혜 = (160, 30, 55) → 여

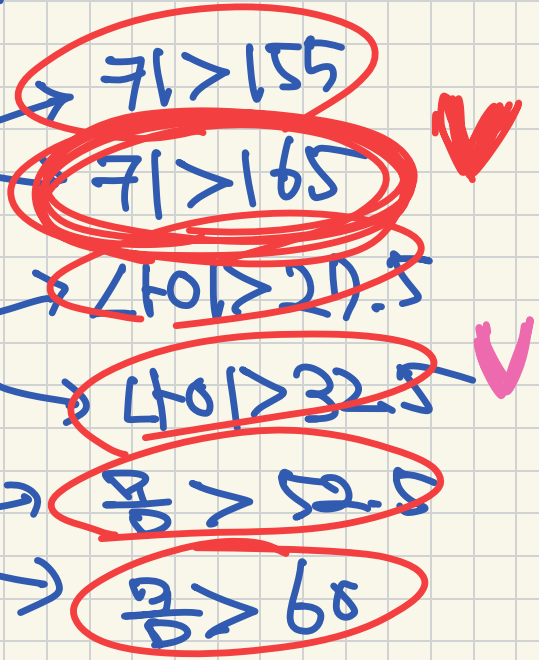
수희 = (150, 35, 50) → 여

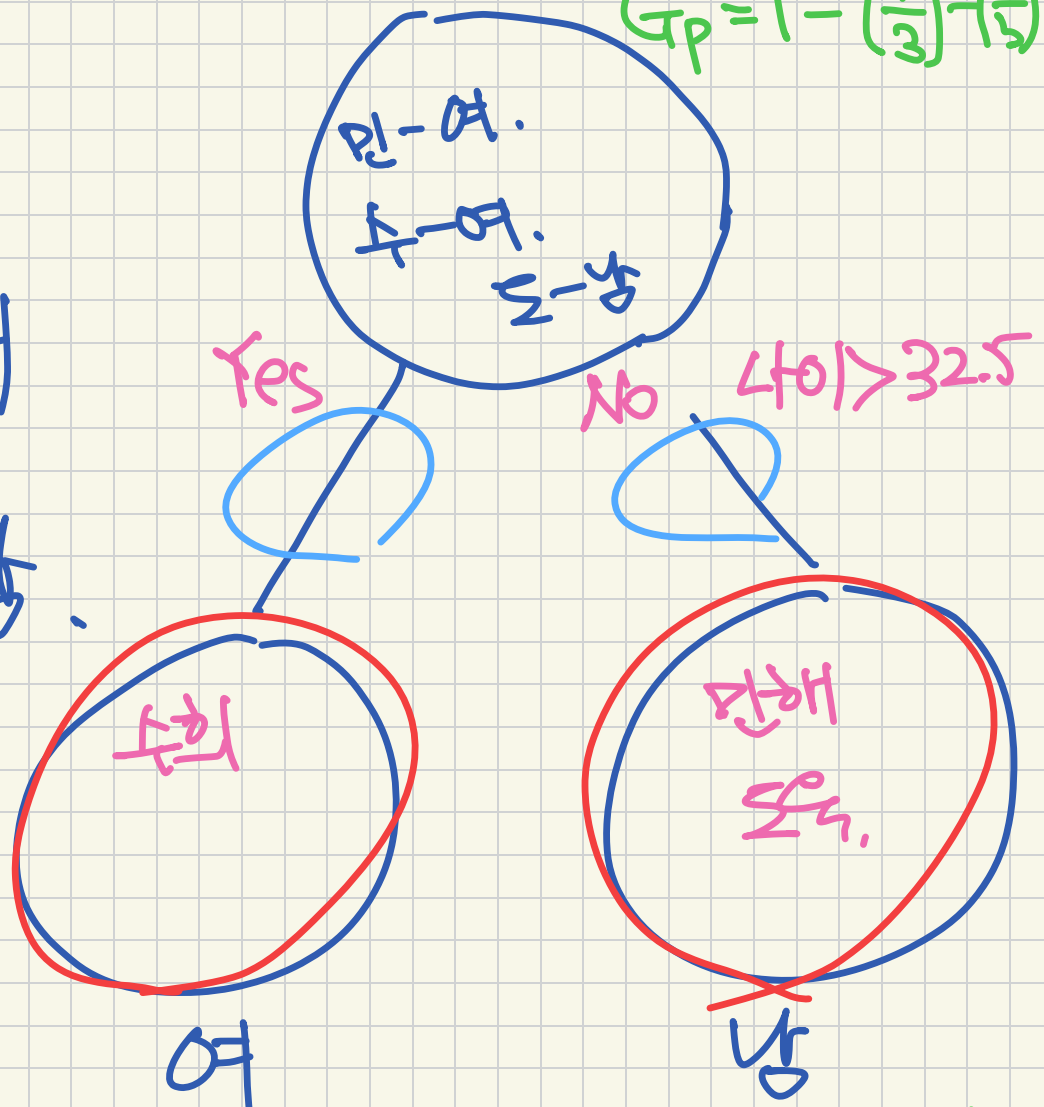
소윤 = (170, 25, 65) → 남.

키 → 150, 160, 170

나이 → 25, 30, 35

몸무게 → 50, 55, 65





$G_P = 1 - \left(\frac{1}{3}\right)^2 - \left(\frac{2}{3}\right)^2$

$G_L = 1 - 1^2 = 0$

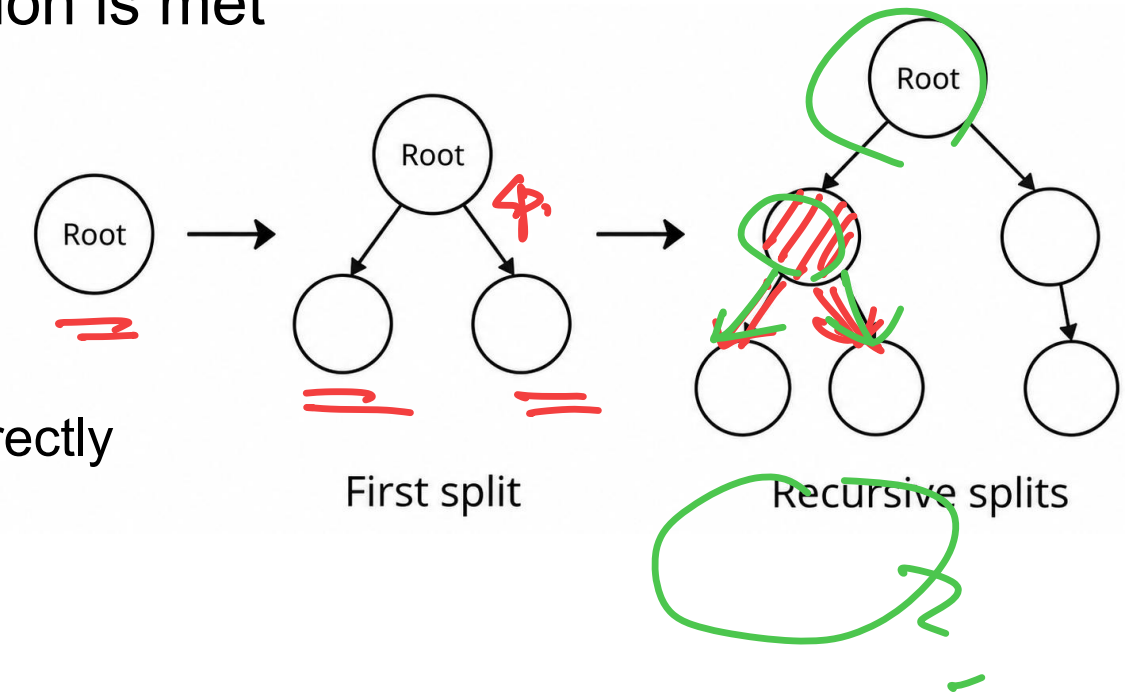
$G_R = 1 - \left(\frac{1}{2}\right)^2 - \left(\frac{1}{2}\right)^2$

$G_{split} = \frac{2}{3} G_R$

Growing Trees

1. Start with all samples at the root
2. Find the best split
3. Split the samples into child nodes
4. Repeat recursively until a stopping criterion is met

- Decision trees are built greedily
  - The best split is chosen at each node
  - The globally optimal tree is not searched directly



Regression Trees

- Regression trees predict continuous values
- Leaf prediction
  - Each leaf predicts the mean target value in that region

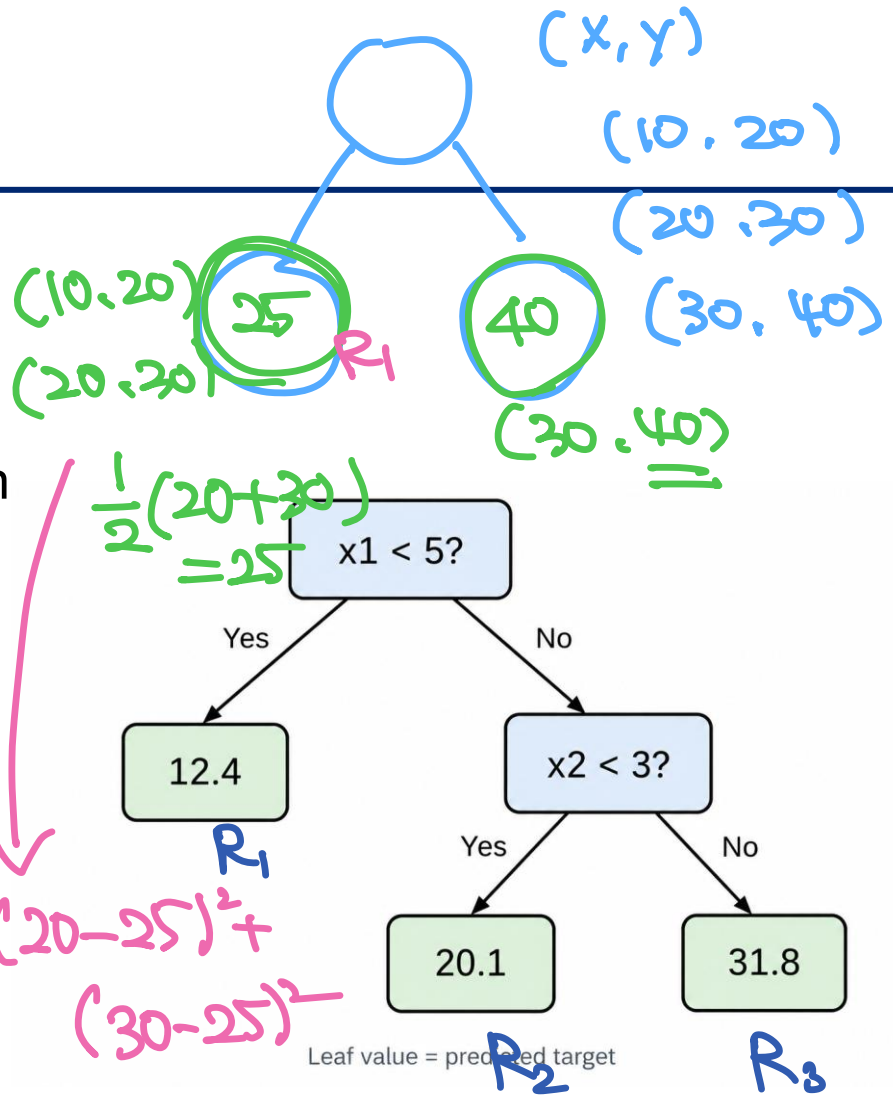
$$c_m = \frac{1}{|R_m|} \sum_{i:\, \mathbf{x}_i \in R_m} y_i$$

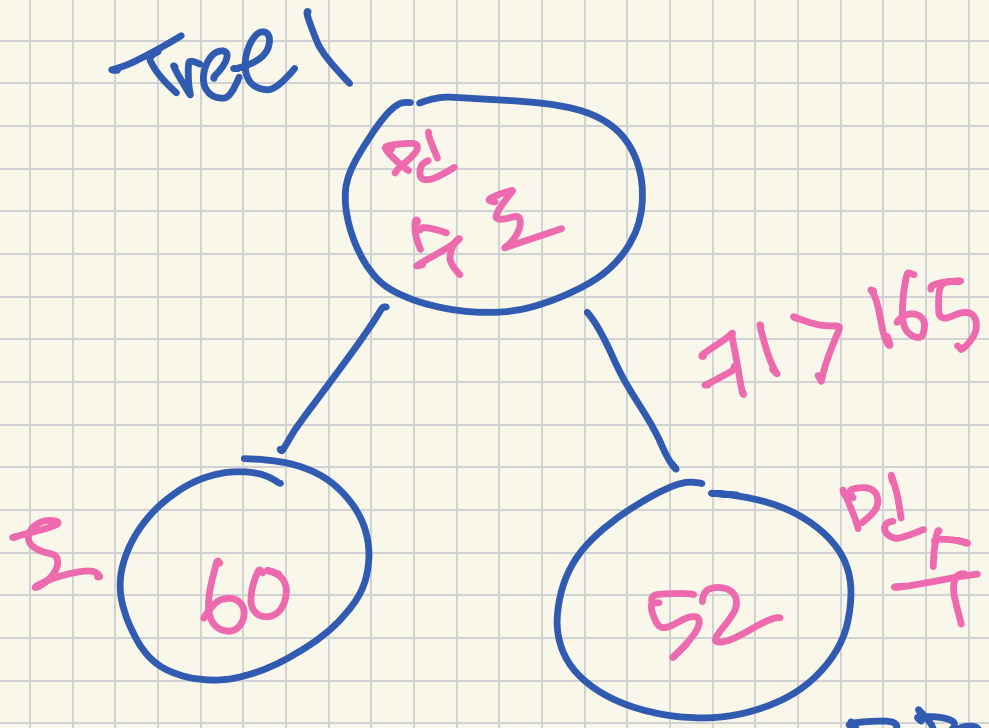
- Split criterion
  - Squared error within a region:

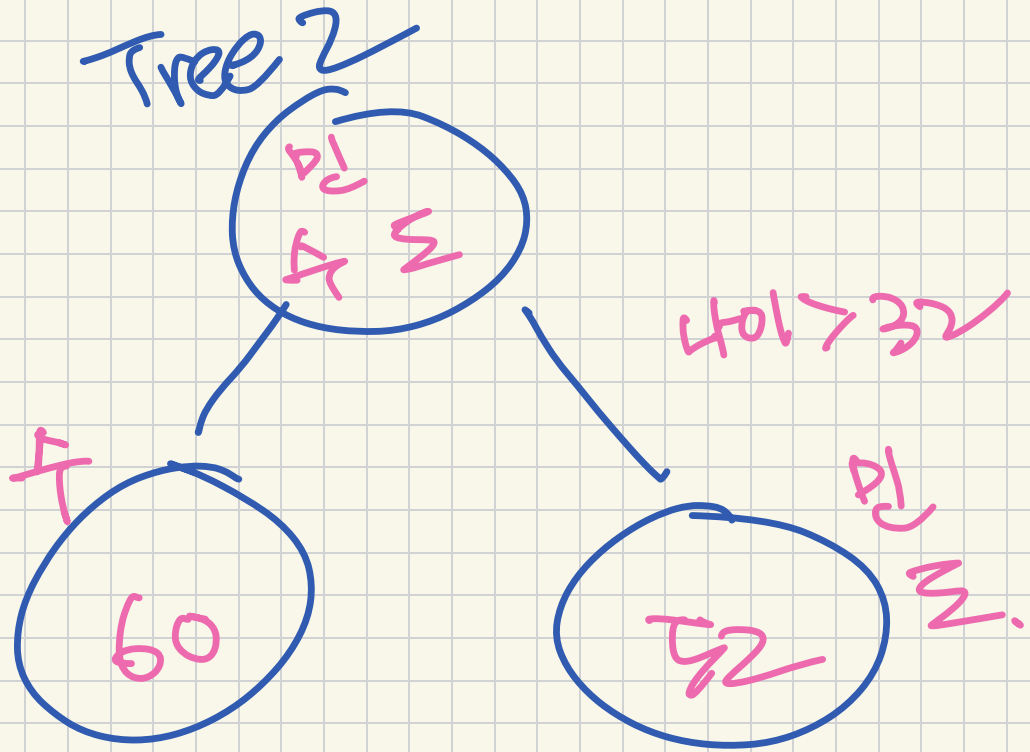
$$I(R_m) = \sum_{i:\, \mathbf{x}_i \in R_m} (y_i - c_m)^2$$

- Choose the split that minimizes the total child error:

$$I(R_L) + I(R_R)$$







⇒ L, 나이, ~~몸무게~~

민혜 = (160, 30) → 55

~~수희~~ = (155, 35) → 50

S = (170, 25) → 70

Tree 1: $I(R_L) = (70-60)^2$

$I(R_R) = (55-52)^2 + (50-52)^2$

Tree 2: $I(R_L) = (50-60)^2$

$I(R_R) = (55-52)^2 + (70-52)^2$

Tree Criteria

	Classification	Regression
Goal	Predict a class	Predict a value
Split criterion	Gini / Entropy	Squared Error
Leaf output	Class / probability	Mean target

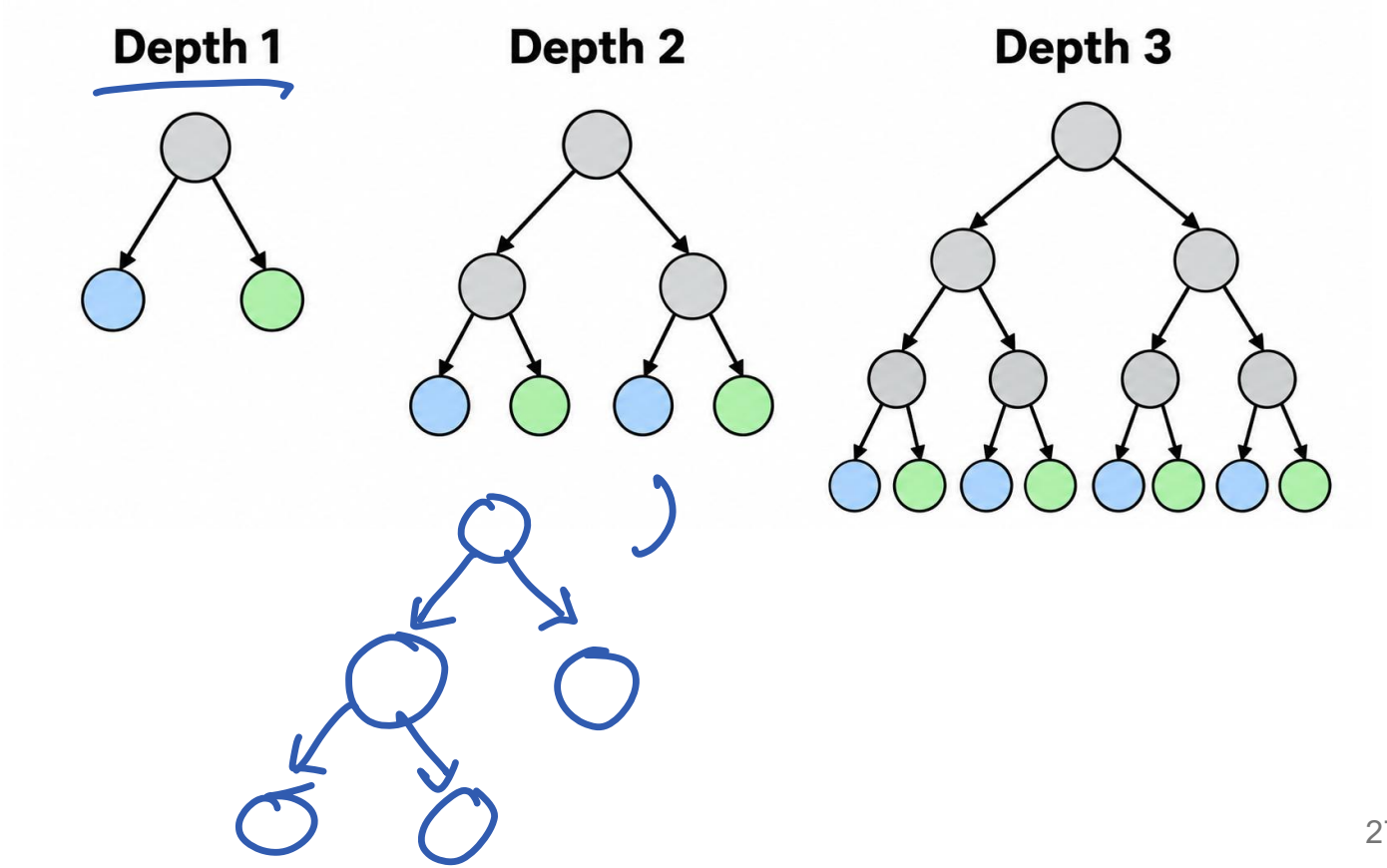
Korea Aerospace University

Part 3. Tree Complexity

Tree Depth

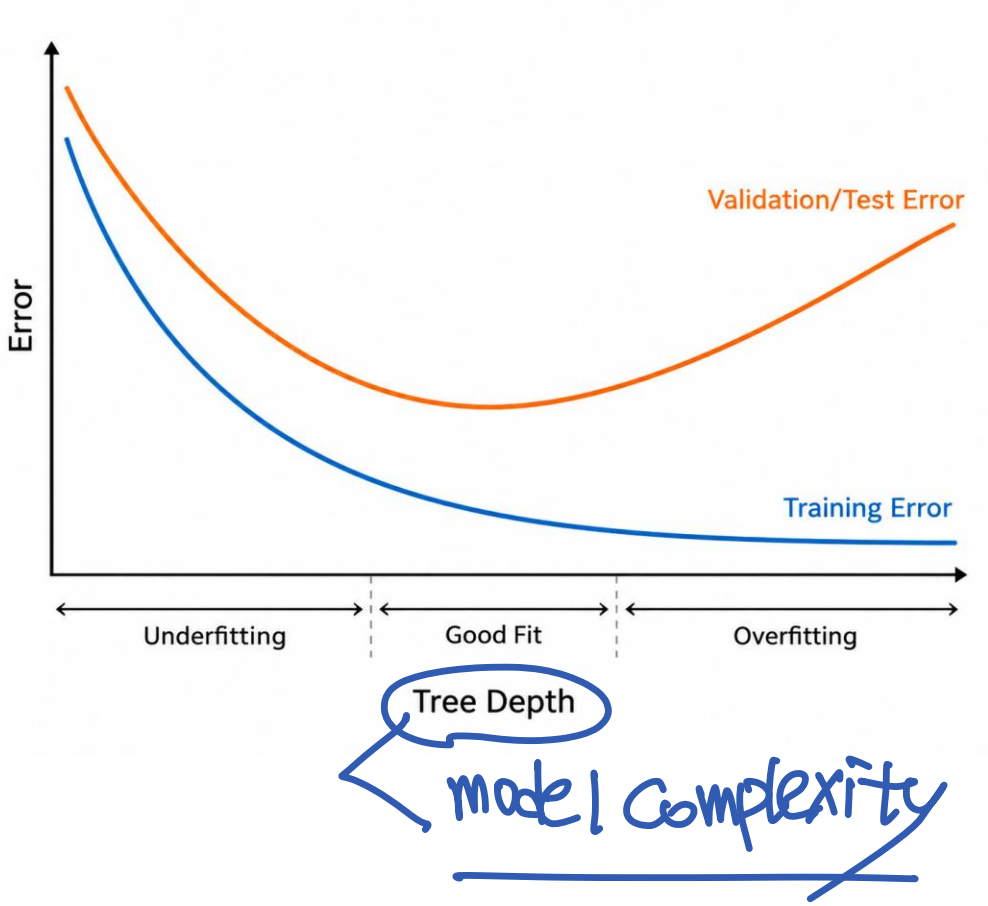
- Tree depth
  - Number of splits along the longest root-to-leaf path

- Increasing depth
  - Creates more decision regions
  - Allows more detailed predictions
  - Increases model complexity



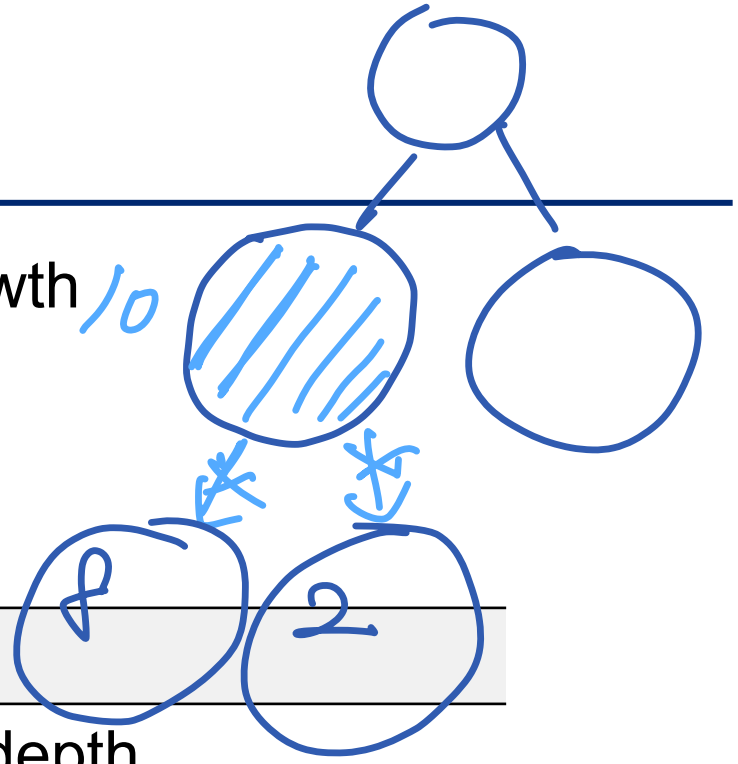
Overfitting

- Deeper trees
  - Usually reduce training error
  - May fit noise and individual samples
- Result
  - Validation/test performance may degrade
  - Low bias but high variance



Regularization

- Tree complexity can be regularized by restricting tree growth



- Common hyperparameters

Hyperparameter	Effect
max_depth	Limits tree depth
$min_samples_split$ =8	Requires enough samples to split
$min_samples_leaf$ =4	Requires enough samples in each leaf
max_leaf_nodes	Limits the number of leaves

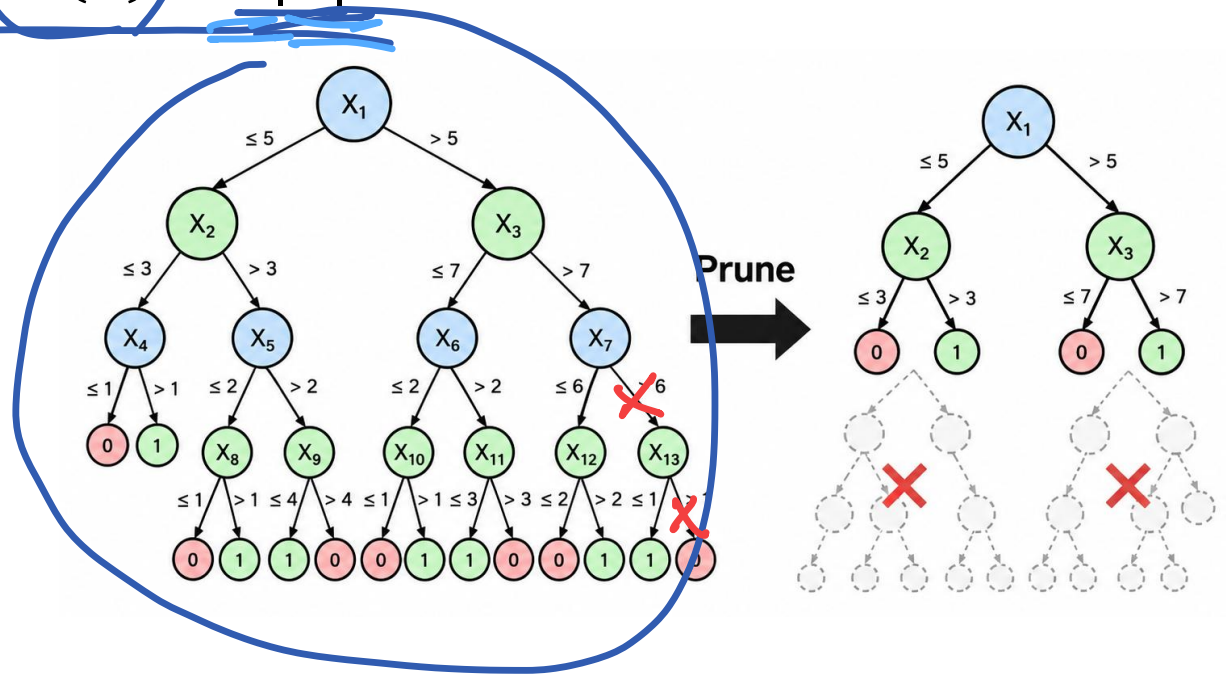
Pruning 가지치기.

- Grow a large tree first
- Remove branches that provide little benefit
- This produces a simpler tree with better generalization

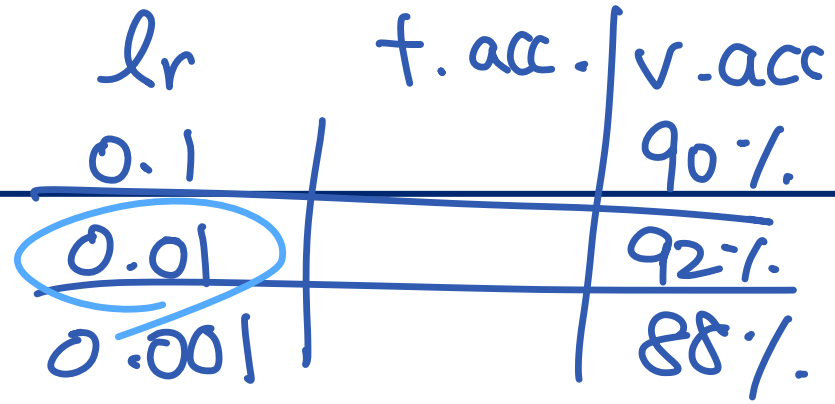
$L_{total} = L_{train} + \lambda R(\theta)$
= reg. loss

$$R_\alpha(T) = R(T) + \alpha|T|$$

- $R(T)$: training loss / empirical risk
- $|T|$: number of leaves
- α: complexity penalty



Tuning Trees



- Training performance alone is not enough

- Select complexity using validation data or cross-validation

max_depth	Train Acc.	Val. Acc.
2	82%	80%
5	94%	89%
10	100%	83%

linear

training data → W, b 학습

validation data → lr, ... 고르기

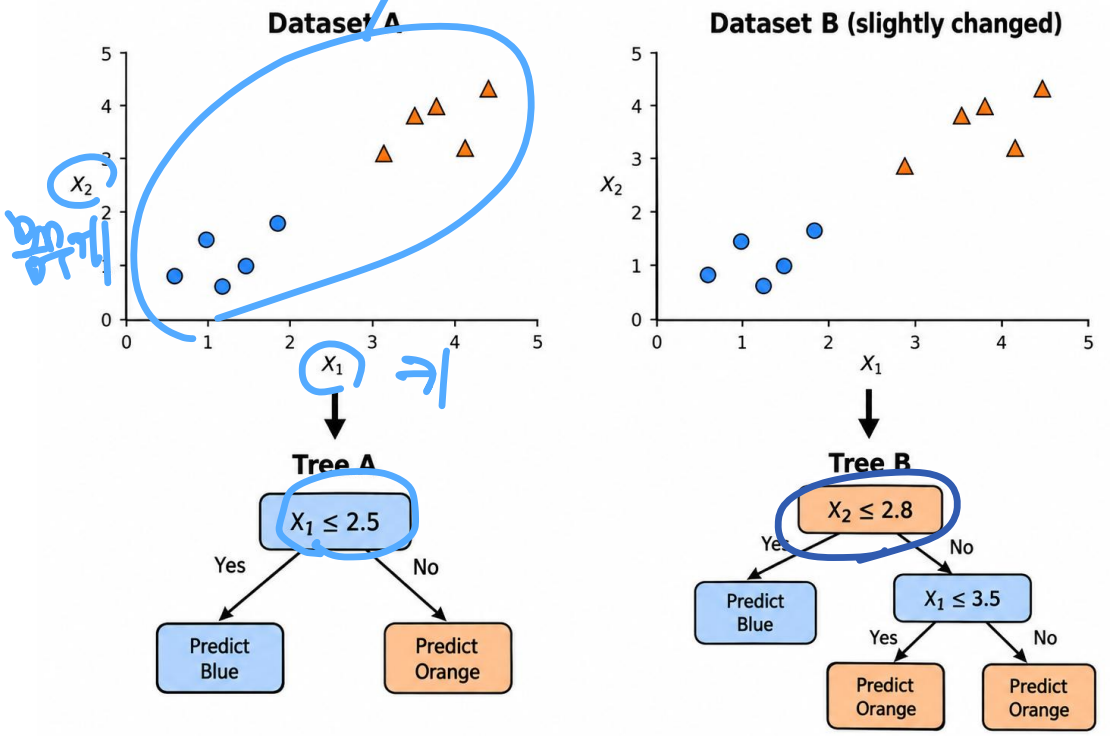
tree

train → split

val → max_depth, max_leaf, min_split ..

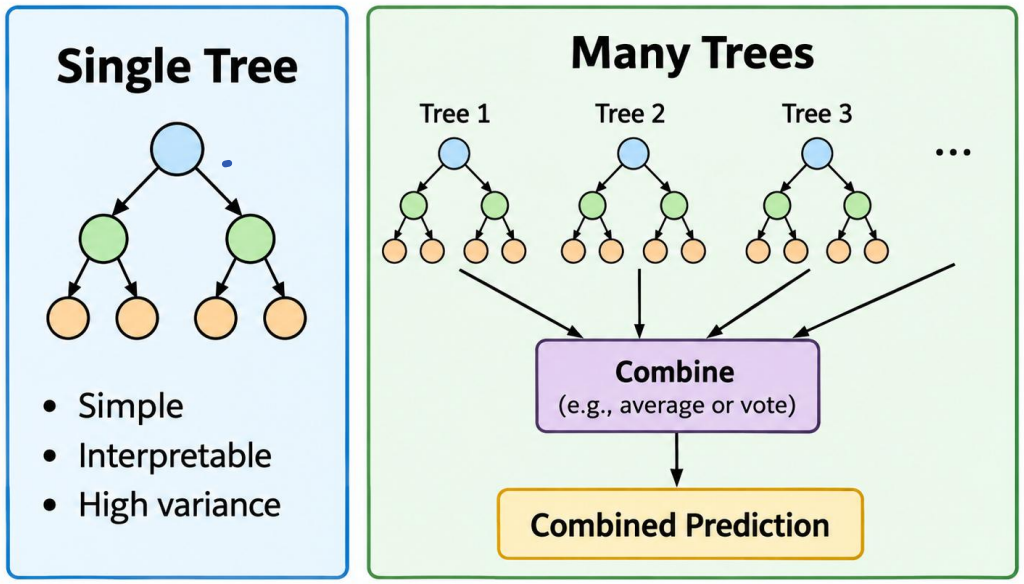
Instability

- Decision trees can have high variance
  - Small changes in training data may change the selected splits
  - A change near the root can affect much of the tree
- Single trees can therefore be unstable



Toward Ensembles

- Single Tree
  - Interpretable
  - High variance
- Can combining many trees reduce variance?



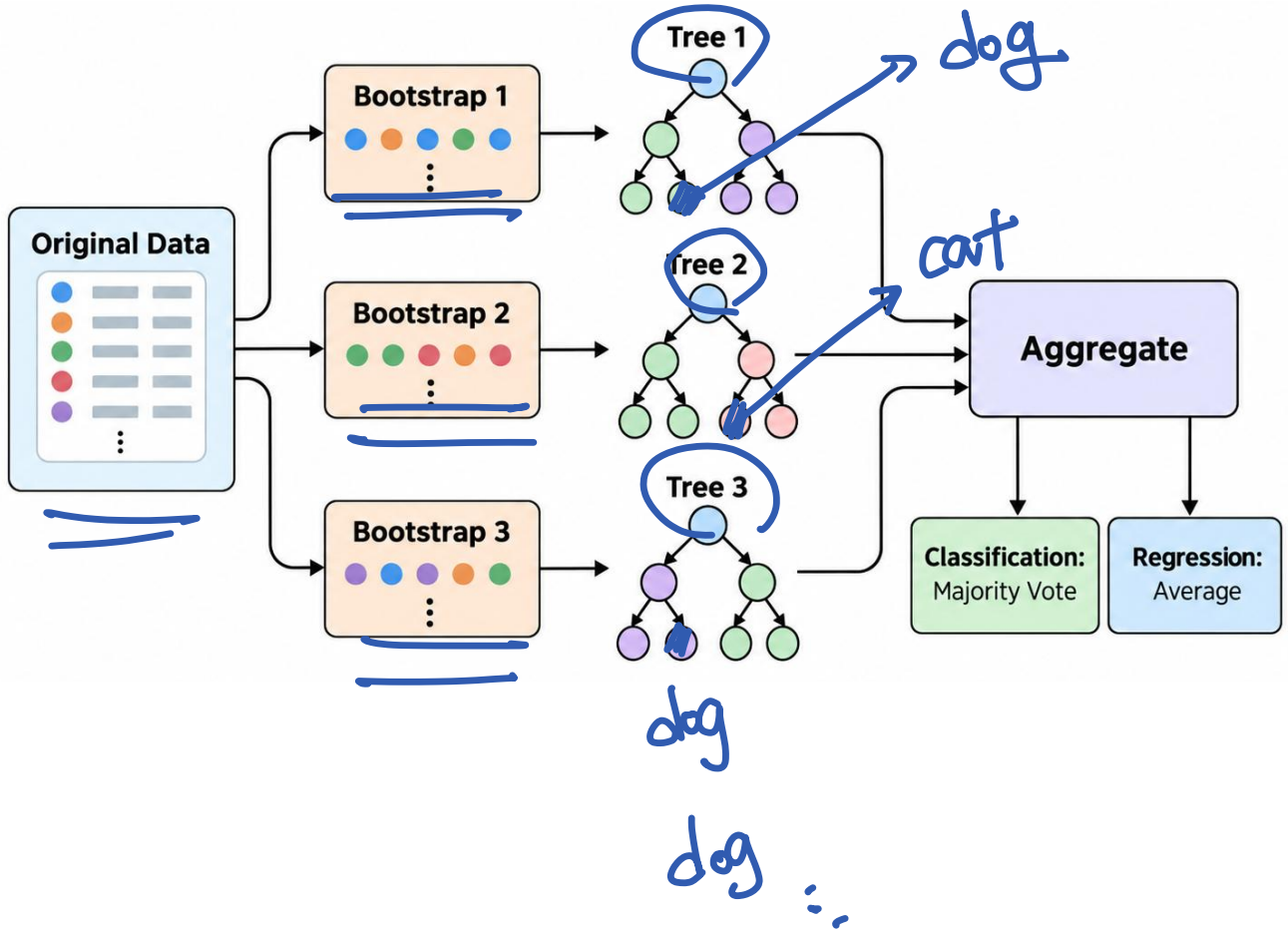
Korea Aerospace University

Part 4. Tree Ensembles

Bagging

- Bagging = Bootstrap Aggregating

- Create multiple bootstrap datasets
- Train one tree on each dataset
- Combine predictions
  - Classification: majority vote
  - Regression: average



Bootstrap

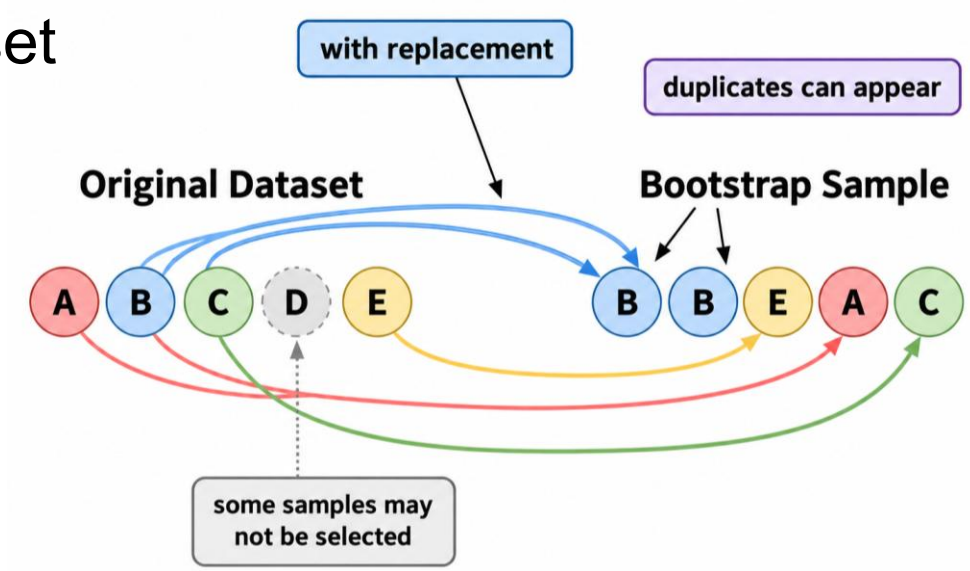
- Example

$$[A, B, C, D, E]$$
$$[B, B, E, A, C]$$

[A, A, A, A, A]

[A, A, B, B, C]

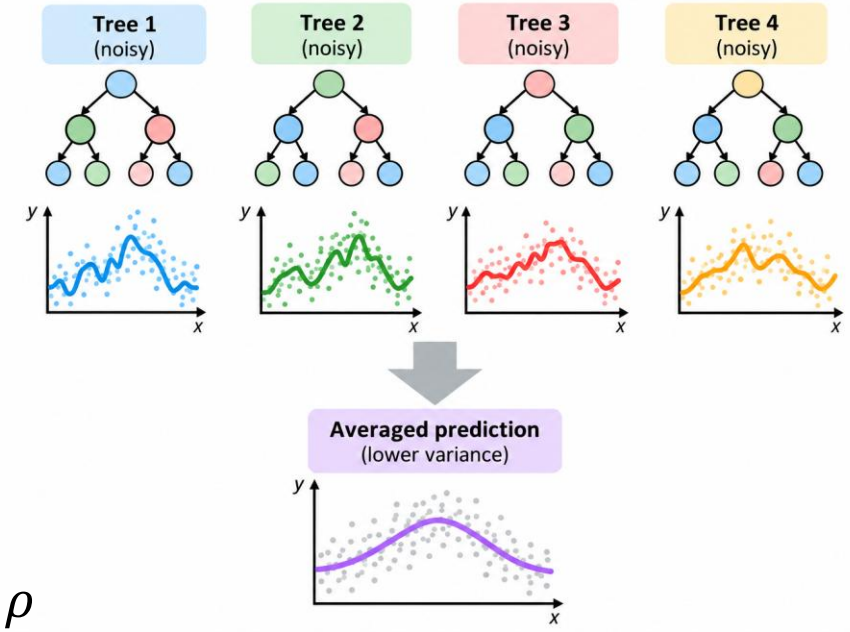
- Sample from the training set with replacement
- Use the same sample size as the original dataset
  - Some observations may appear multiple times
  - Some may not appear at all



Why Bagging?

[A, B, C, D, E] → [A, B, C, D, E] → tree1
→ [B, A, E, C, D] → tree2

- Why it works
  - Decision trees have high variance
  - Bootstrap samples produce different trees
  - Averaging reduces fluctuations 변동.
  - The benefit is larger when trees are less correlated
- Variance of the average
  - Assuming equal variance σ^2 and pairwise correlation ρ



오우지 마셍

$$Var(\bar{f}) = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

variance
correlation

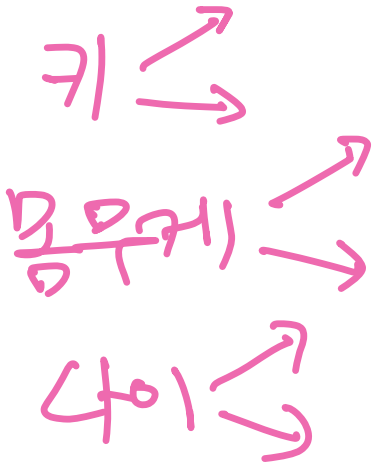
  - B: number of trees
  - ρ: correlation between trees

Random Forests

데이터 다양성 고려 → feature?

$$\text{Random Forest} = \text{Bagging} + \text{Random Feature Subsampling}$$

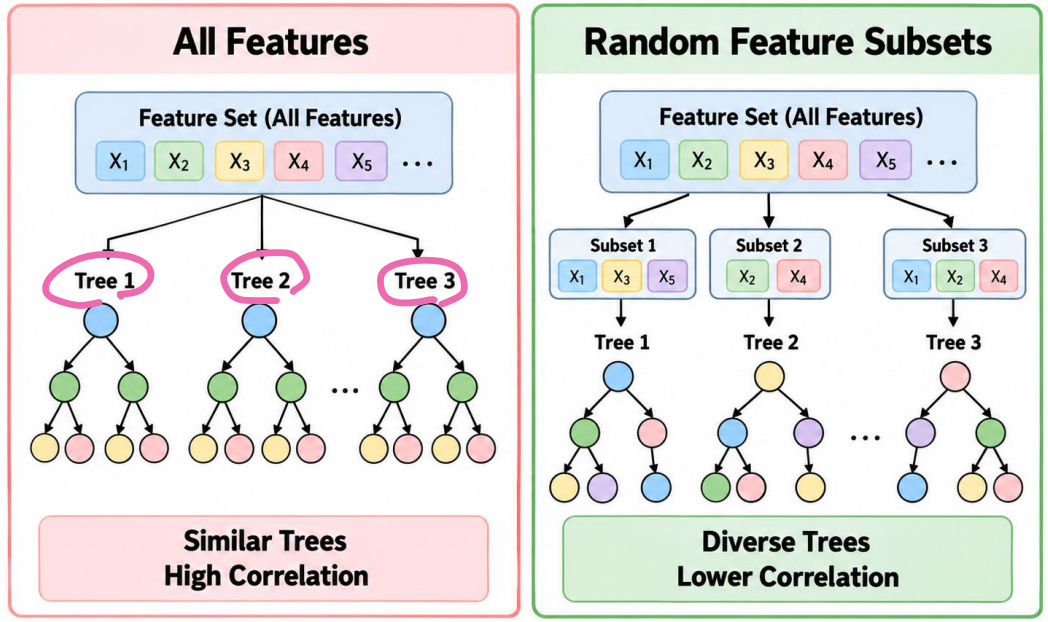
- Each tree is trained on a bootstrap sample
- At each split, only a subset of features is considered
- Predictions are combined across trees



- Why randomly restrict the available features?

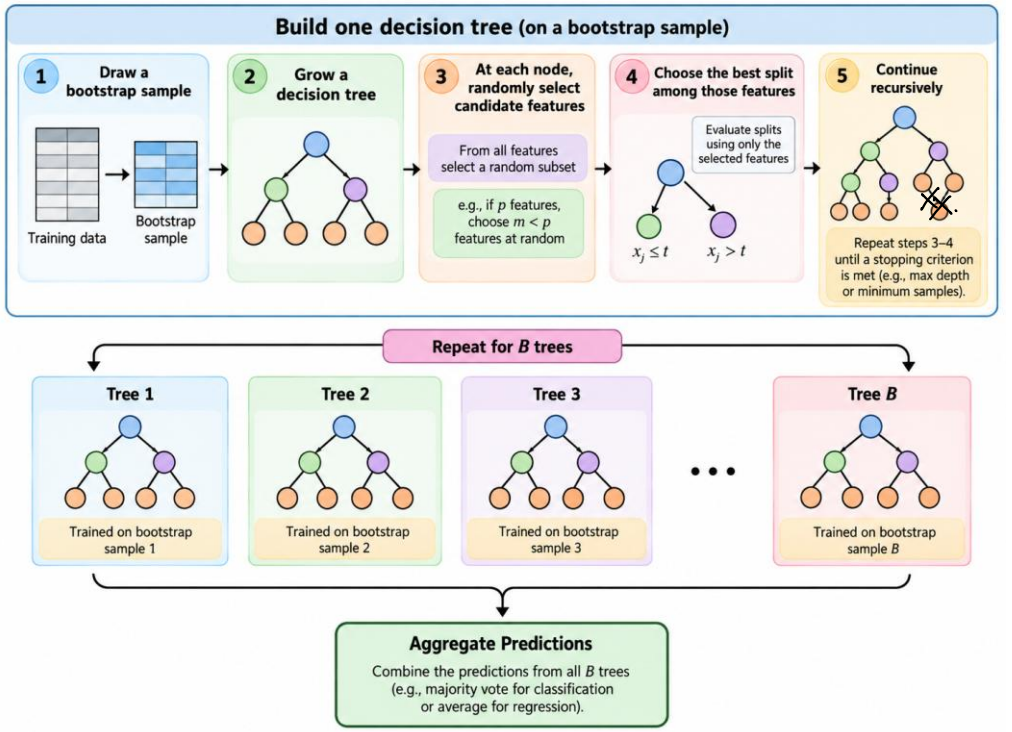
Random Feature Subsets

- Without feature randomness
  - Strong features may dominate
  - Trees can become similar
  - Predictions become correlated
- With random feature subsets
  - Trees become more diverse
  - Lower correlation → stronger variance reduction



Forest Training

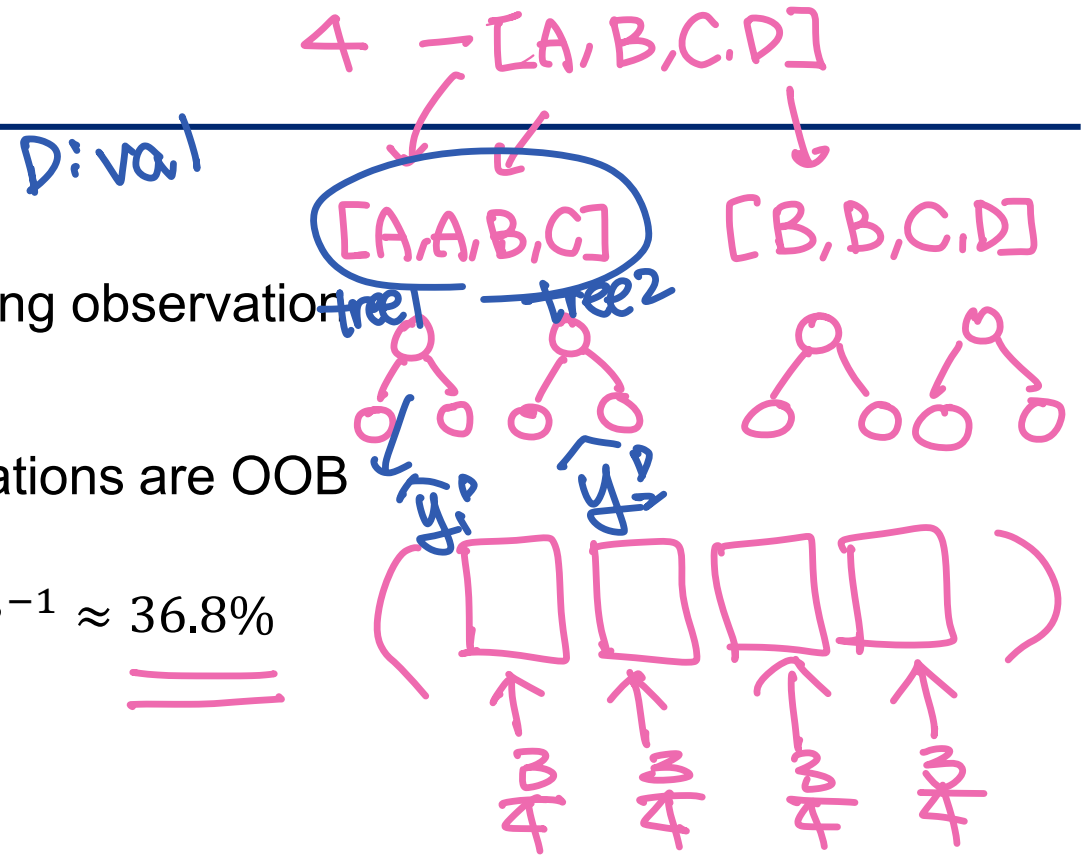
1. Draw a bootstrap sample → 데이터 다양성 ↑
2. Grow a tree
   - Randomly select candidate features at each node
   - Choose the best split among them
   - Repeat recursively
3. Repeat for many trees
4. Aggregate predictions



OOB Samples

- Out-of-bag (OOB) samples
  - A bootstrap sample does not contain every training observation
  - Unselected examples are called OOB samples
  - For a given tree, about 36.8% of training observations are OOB

$$P(\text{not selected}) = \left(1 - \frac{1}{n}\right)^n \approx e^{-1} \approx 36.8\%$$



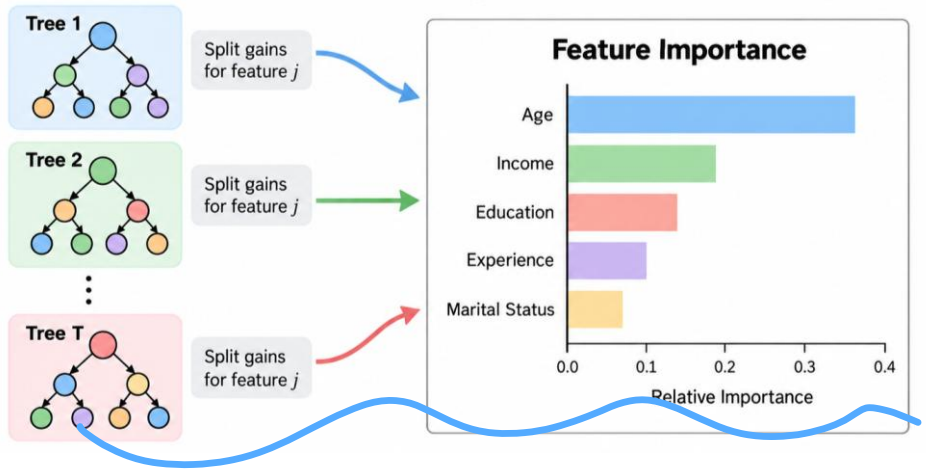
- Why OOB is useful
  - OOB samples can evaluate a tree on data it did not train on
  - Aggregating OOB predictions gives an internal performance estimate

Feature Importance

- Features that produce larger impurity reductions receive higher importance
- A common measure accumulates impurity reduction across splits

$$\text{Importance}(j) = \sum_{\text{splits on } j} \text{weighted impurity reduction}$$

- Feature importance does not imply causality
- Importances are aggregated across trees in a random forest



Hyperparameters

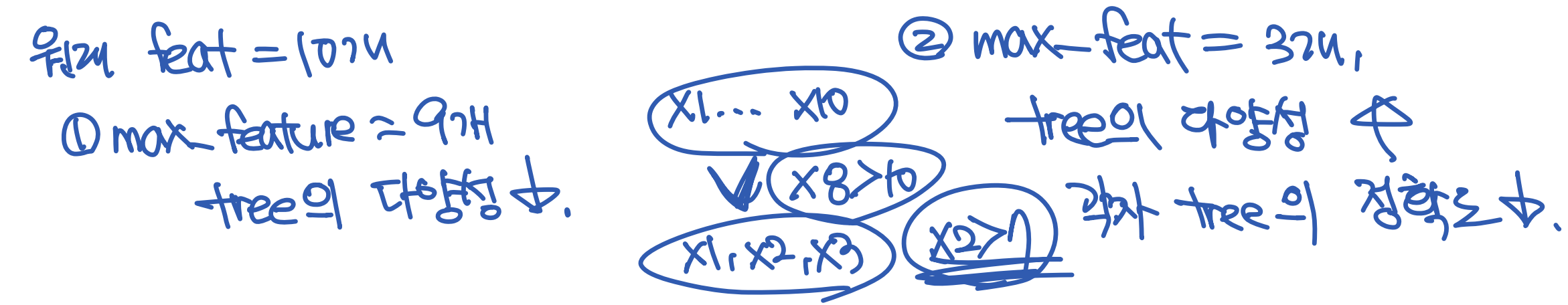
- More trees → more computation, usually greater stability
- Smaller $max_features$ → more diversity, but may weaken individual trees

Hyperparameter	Role
$n_estimators$	Number of trees
$max_features$	Features considered at each split
max_depth	Maximum tree depth
$min_samples_leaf$	Minimum samples in a leaf

for.전

for.전

tree



Tree vs. Forest

- Random forests trade interpretability for improved stability

	Decision Tree	Random Forest
No. of Trees	One	Many
Interpretability	High	Lower
Variance	High	Lower
Stability	Lower	Higher
Computation	Lower	Higher

Next Lecture

- Lecture 6: Support Vector Machines & Boosting
  - Feature Scaling
  - Support Vector Machines (SVM)
  - Kernel Methods
  - Bagging vs. Boosting
  - Gradient Boosting & XGBoost / LightGBM

Korea Aerospace University

Questions?

jiyoonshin@kau.ac.kr