

# Introduction to AutoML

한국원자력연구원 류승형

# 과정 개요 (Day 1)

시간표			
일차	시간	시간	내용
1일차 7/5 (수)	09:20~09:30	10분	등록 및 오리엔테이션
	09:30~11:30	120분	- AutoML 개요 AutoML 개념, 목적 데이터 형식 이해 데이터 형식에 따른 적정 모델 선택 이해
	11:30~12:30	60분	점심식사
	12:30~13:30	60분	표 형식 데이터 분석 - 표 형식 데이터 전처리, 학습, 평가데이터 분리방법
	13:30~17:30	240분	- AutoML 기본 이해 AutoML Solution 종류, 모델 탐색 - AutoML 활용 위한 학습, 평가데이터셋 준비, 최적 모델 저장 및 추론 실습
2일차 7/6 (목)	09:00~12:30	210분	- AutoML 활용 표 형식 데이터 모델 탐색 인공신경망 모델 이해 AutoKeras 활용한 표 형식 데이터 분류 - AutoKeras 활용한 표 형식 데이터 회귀 모델 탐색 실습
	12:30~13:30	60분	점심식사
	13:30~17:30	210분	- AutoML 활용 이미지 데이터 모델 탐색 AutoKeras 활용한 이미지 데이터 분류 및 회귀 모델 탐색 실습

1. Machine learning process 전반
2. Tabular data 실습
3. ML 기초 pipeline 실습
4. AutoML 실습

# 세부 시간표

- 09:30 ~ 10:30 : 강의 슬라이드 (Slide 1, Chapter. 1)
- 10:30 ~ 11:30 : 강의 슬라이드 (Slide 1, Chapter. 2&3)
- 11:30 ~ 12:30 : 점심식사
- 12:30 ~ 13:30 : 코드 실습 / Tabular data 다루기
- 13:30 ~ 14:30 : 강의슬라이드 (Slide 2)
- 14:30 ~ 15:30 : 코드 실습 / ML 모델 파이프라인
- 15:30 ~ 16:30 : 코드 실습 / ML 모델 프로세스별 자동화 라이브러리 (전반)
- 16:30 ~ 17:30 : 코드 실습 / ML 모델 프로세스별 자동화 라이브러리 (후반)

# 목차

- Chapter 1. Brief introduction on ML models
- Chapter 2. AutoML Methodologies
- Chapter 3. Hands on AutoML

# **Chapter 1.**

## **Brief introduction on ML models**

# 다양한 머신러닝 모델들

- 선형 회귀 (Linear Regression)
- 로지스틱 회귀 (Logistic Regression)

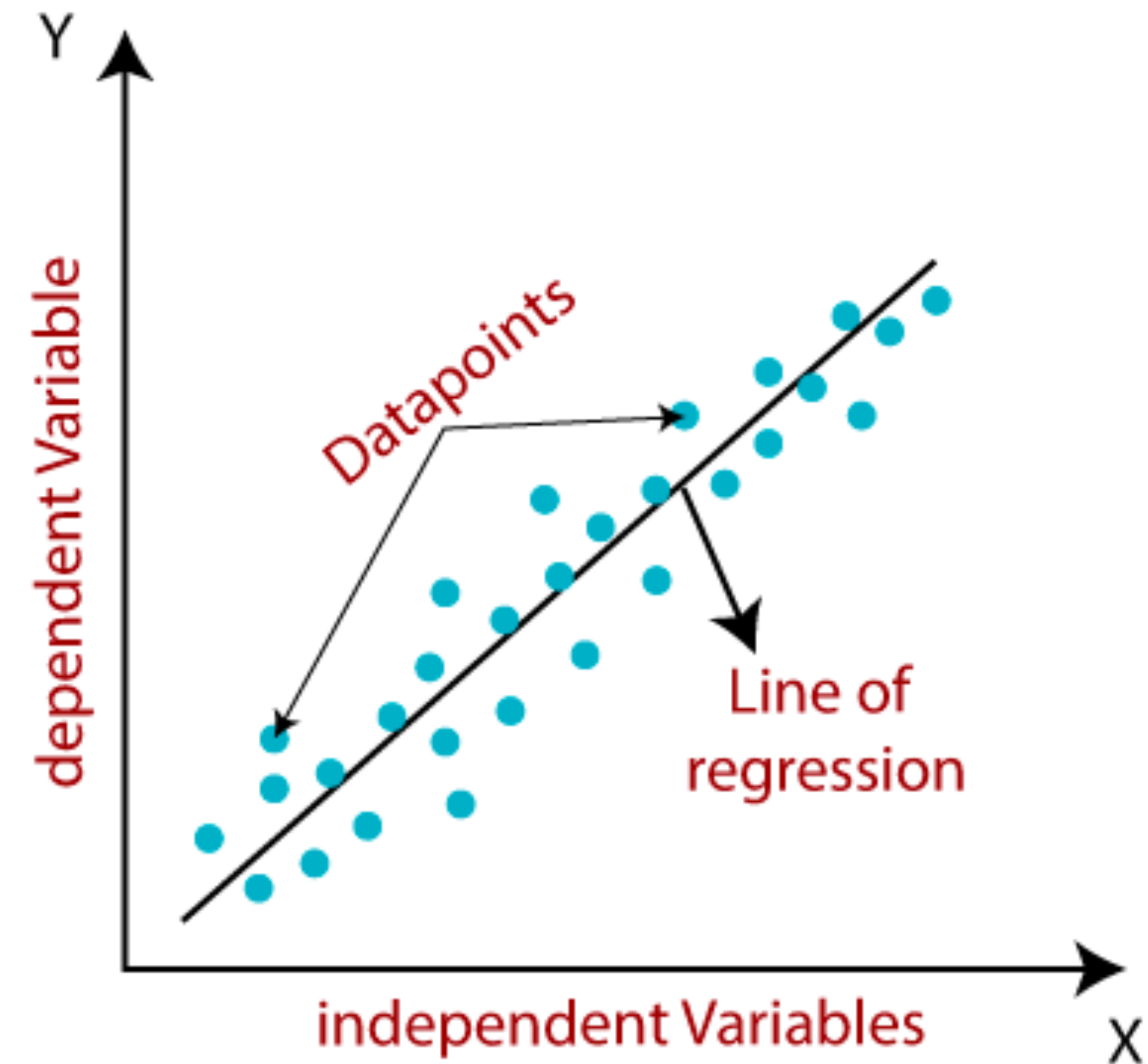
- 의사결정 트리 (Decision Tree)
- 랜덤 포레스트 (Random Forest)
- **그래디언트 부스팅 트리 (Gradient Boosting Tree)**
  - XGBoost
  - LightGBM
  - CatBoost

- 서포트 벡터 머신 (Support Vector Machine, SVM)
- 나이브 베이즈 (Naive Bayes)
- K-최근접 이웃 (K-Nearest Neighbors, KNN)

- **신경망 (Neural Networks)**
  - 다층 퍼셉트론 (Multi-Layer Perceptron, MLP)
  - 컨볼루션 신경망 (Convolutional Neural Network, CNN)
  - 순환 신경망 (Recurrent Neural Network, RNN)
  - 트랜스포머 (Transformer)

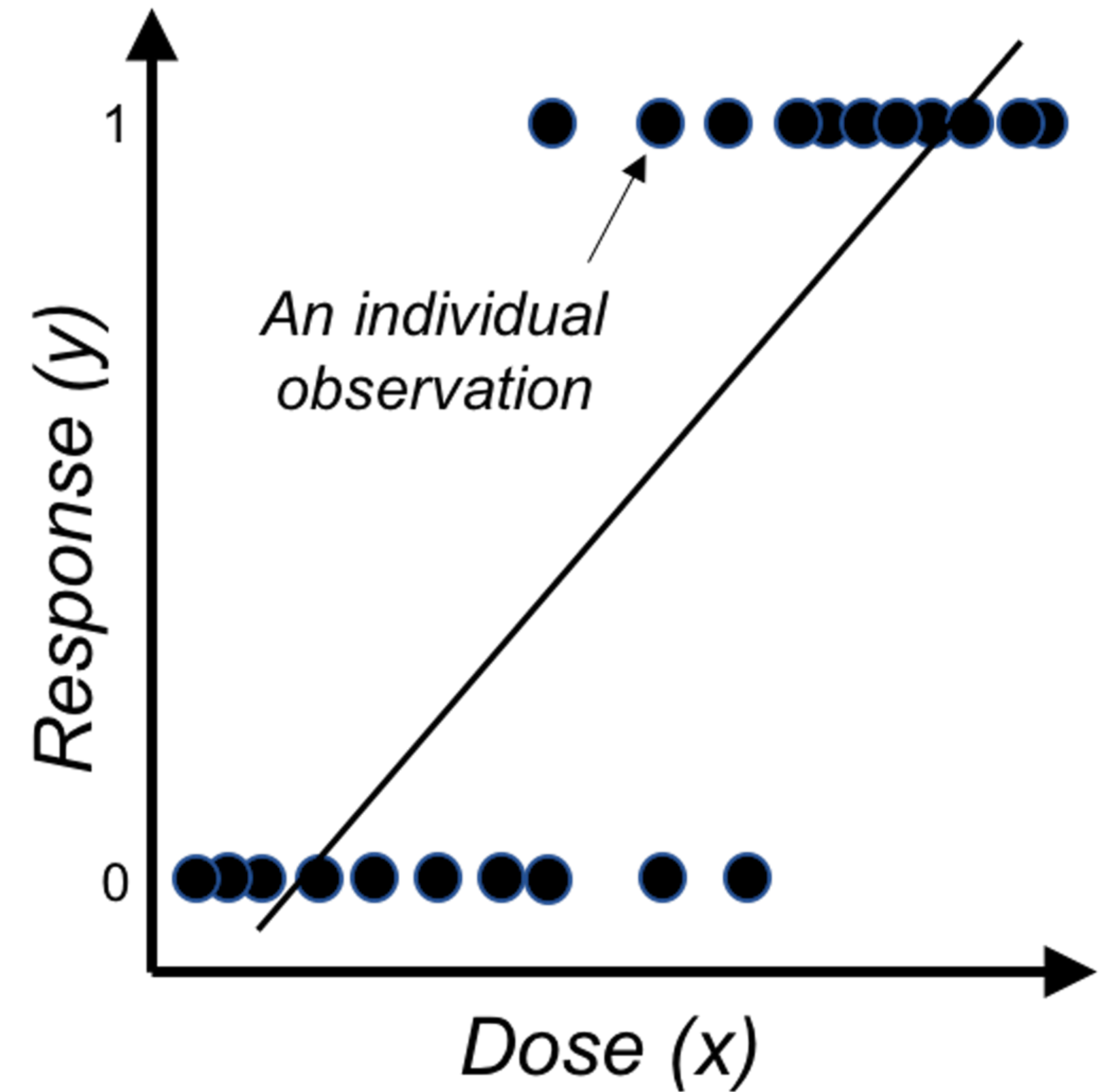
# 선형 회귀

- 종속 변수  $Y$ 와 독립 변수  $X$  사이의 관계
- $Y = a + bx_1 + cx_2 + \dots + \epsilon$
- 회귀 계수( $a, b, c$ )와 선형 관계



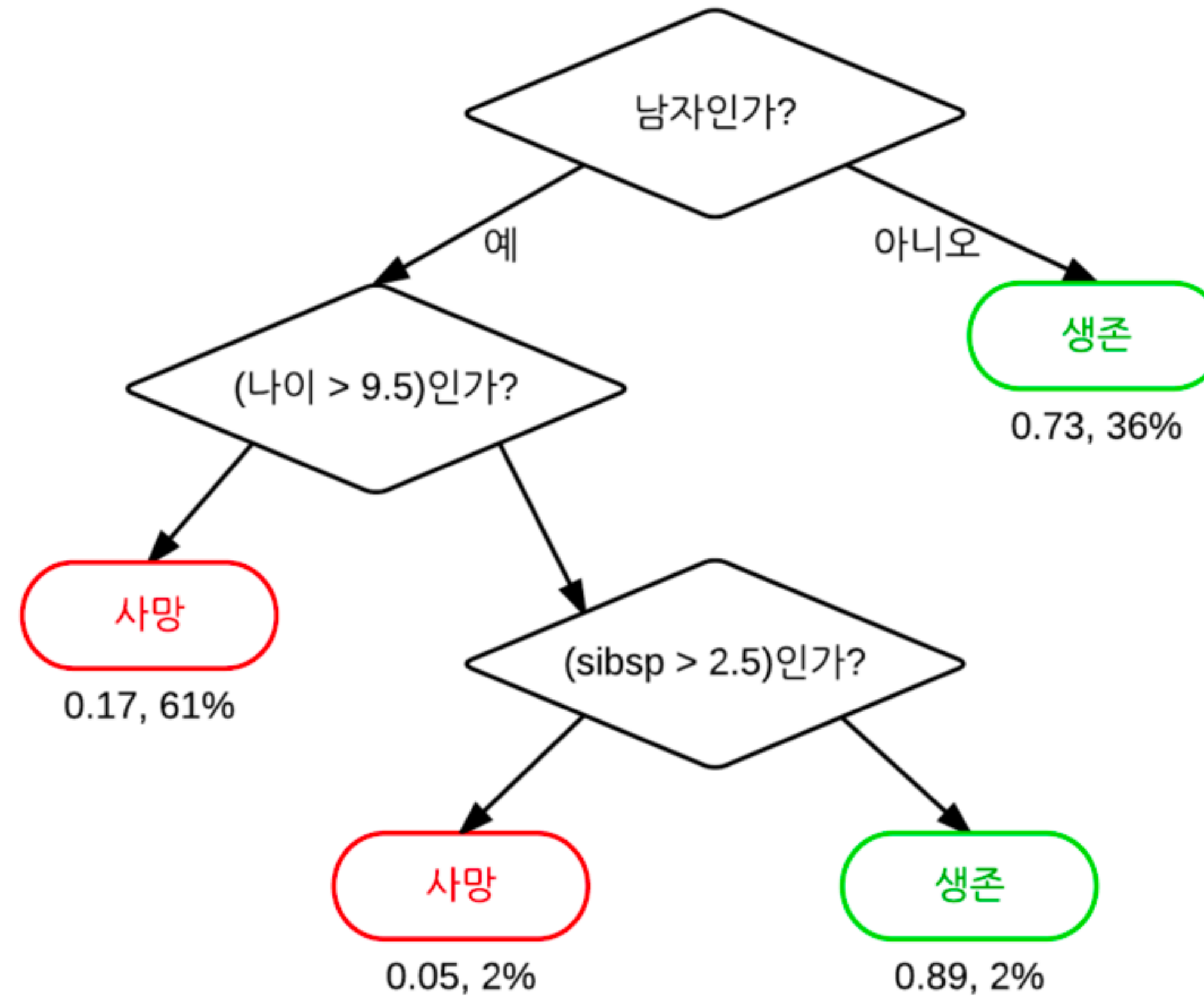
# 로지스틱 회귀

- 선형 회귀 + 범주형 종속 변수  $Y$
- Log-odd을 독립 변수의 선형 조합으로.
  - Log-odd란? 어떤 사건이 발생할 확률( $p$ )과 발생하지 않을 확률( $1-p$ )의 비율의 로그값 (실수)
- Logistic function으로 확률 변환.





# 의사 결정 트리

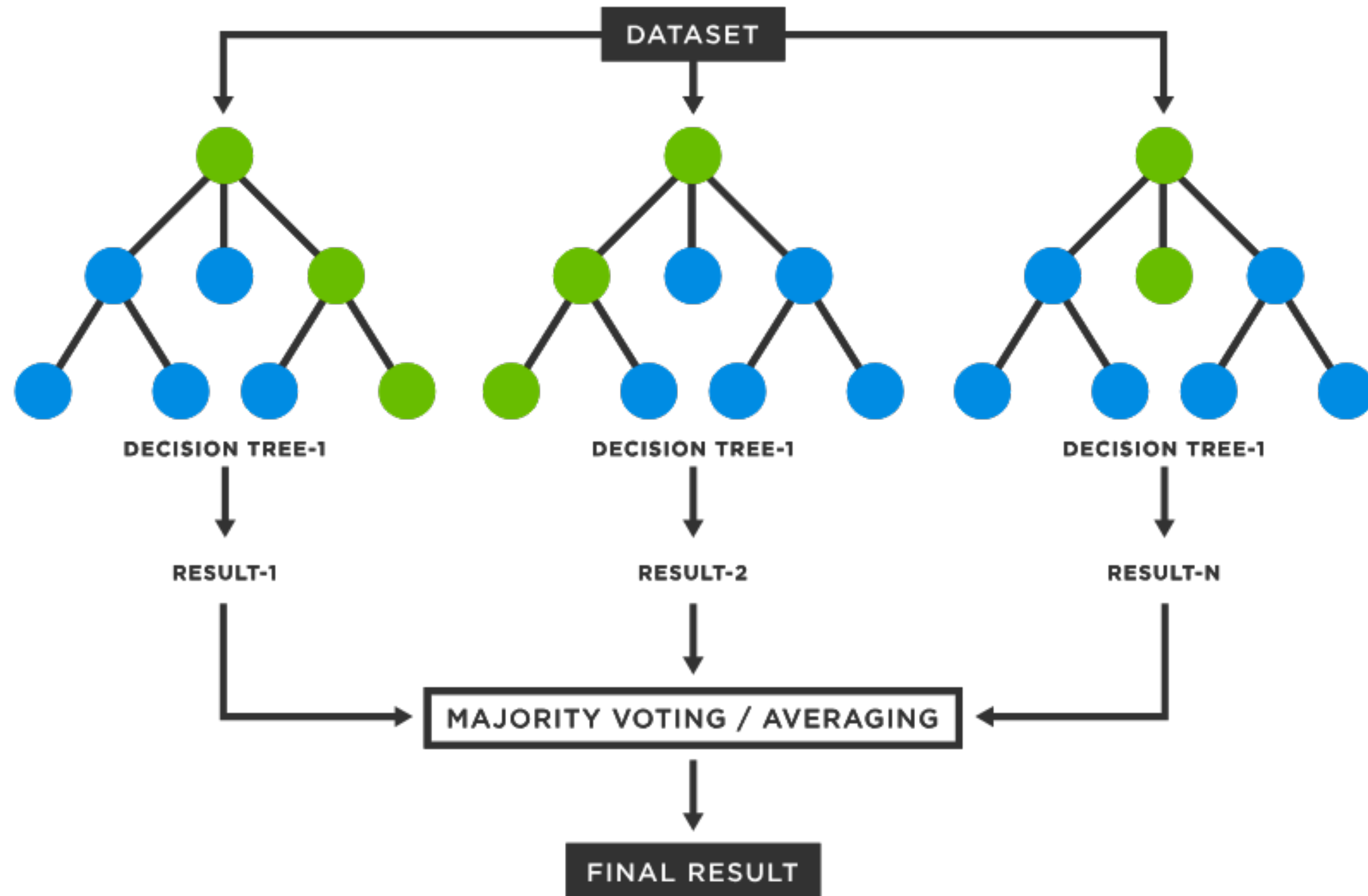


타이타닉 생존자 분류를 위한 의사 결정 트리

# 의사 결정 트리

- 일련의 분류 규칙을 통해 데이터를 구분
- 엔트로피(또는 불순도)가 낮아지는 방향으로 노드를 확장 → 규칙 생성
  - 높은 엔트로피 = 무작위적으로 섞여있다. = 데이터가 구분이 잘 안된다.
- 여러 개의 규칙 중 Information gain을 통해 규칙 결정.
  - Information gain이란? 부모 노드의 엔트로피 - 자녀 노드의 엔트로피
- 이후 overfitting 된 tree에 대한 pruning (가지치기)
- 따라서 의사 결정 트리에서의 학습이란 트리를 확장시켜 나가는 개념으로 볼 수 있음.

# 랜덤 포레스트

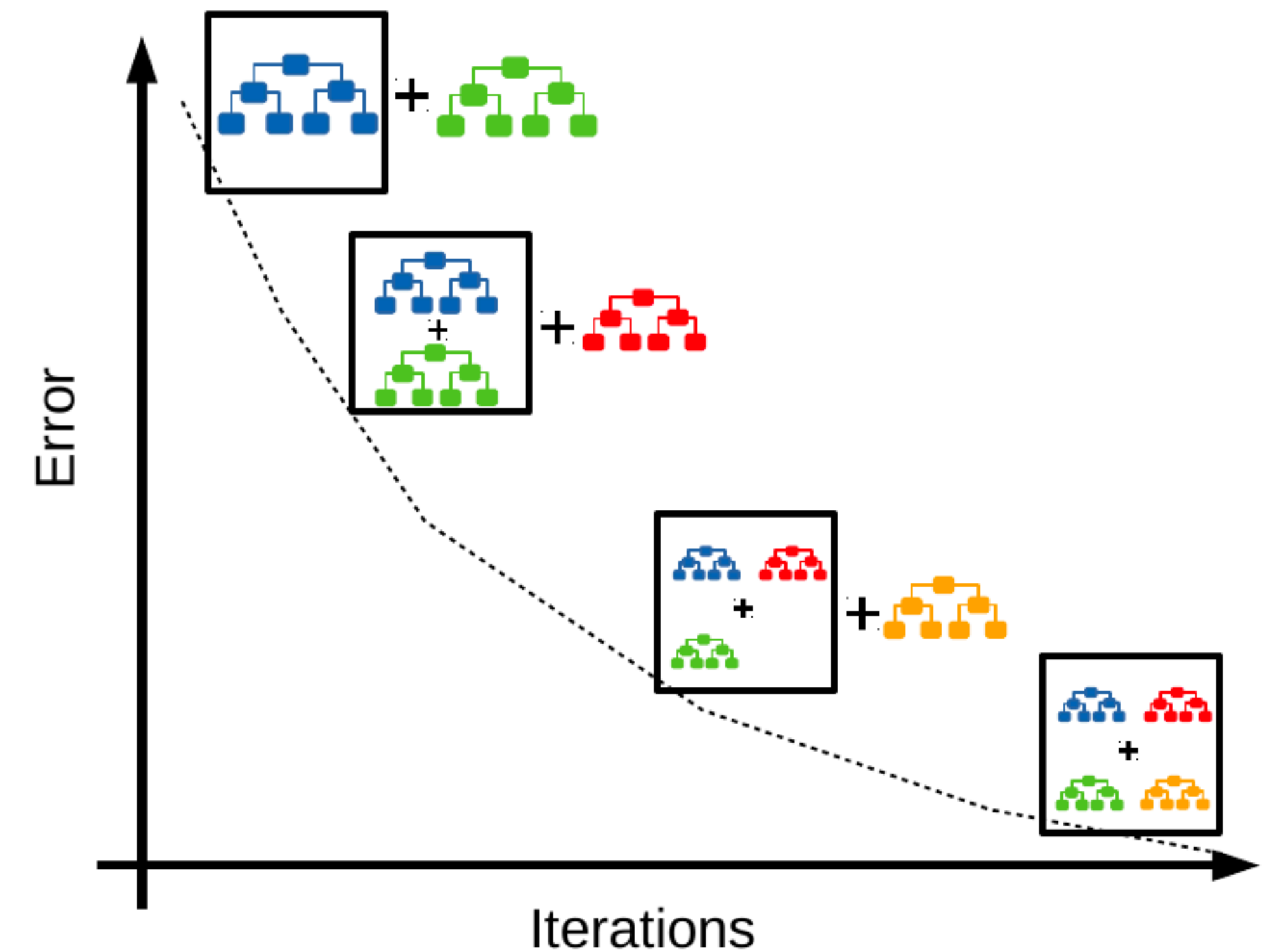


# 랜덤 포레스트

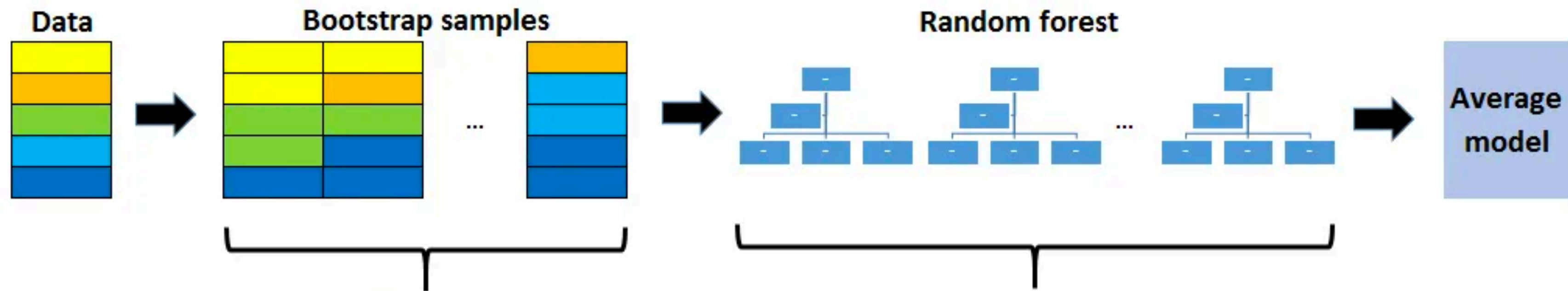
- 의사 결정 트리 모델들의 앙상블
- Bagging (Bootstrap Aggregating) :  
부트스트랩 샘플링한 데이터로 다수의 모델을 학습
  - 부트스트랩 샘플링 : 데이터를 중복 샘플링하여 새로운 데이터셋 생성
- Random feature selection :  
노드 확장시 랜덤하게 선택된 변수들을 바탕으로 규칙 생성

# Gradient Boosted Decision Tree

- Gradient : Residual error를 줄이는 방향
- Boosting : 다수의 weak learner (model)  
→ 하나의 strong learner (model)
- Decision Tree : 결정트리 모델을 활용
- XGBoost, LightGBM, CatBoost

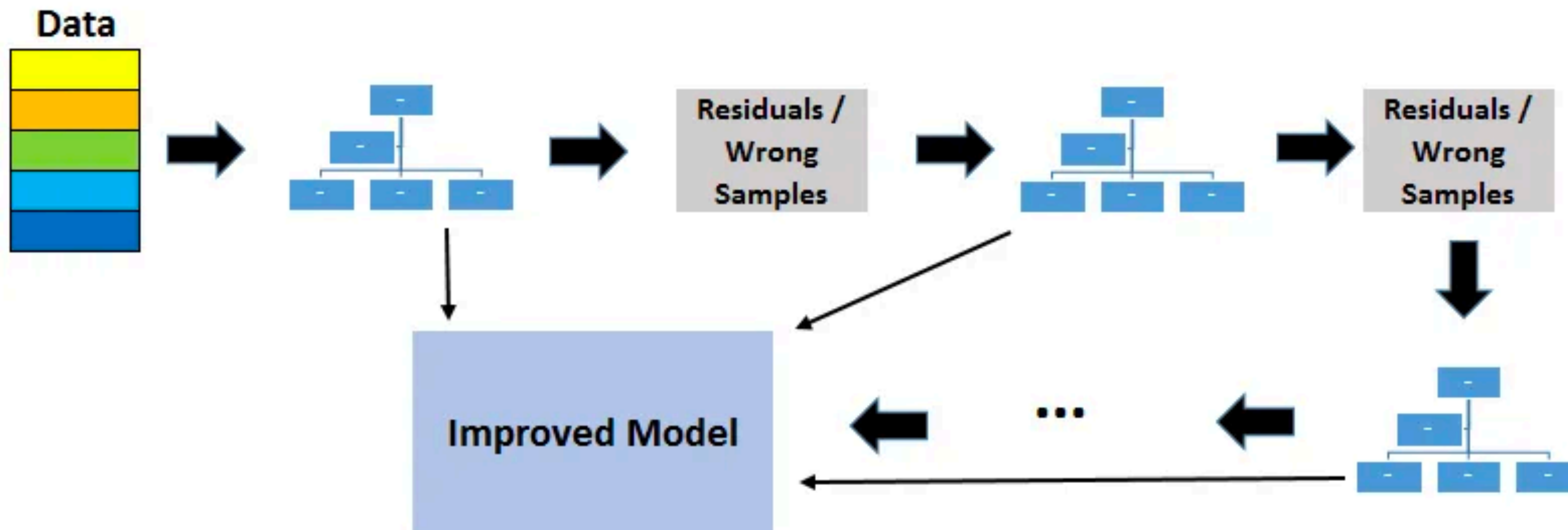


# RF vs GBDT



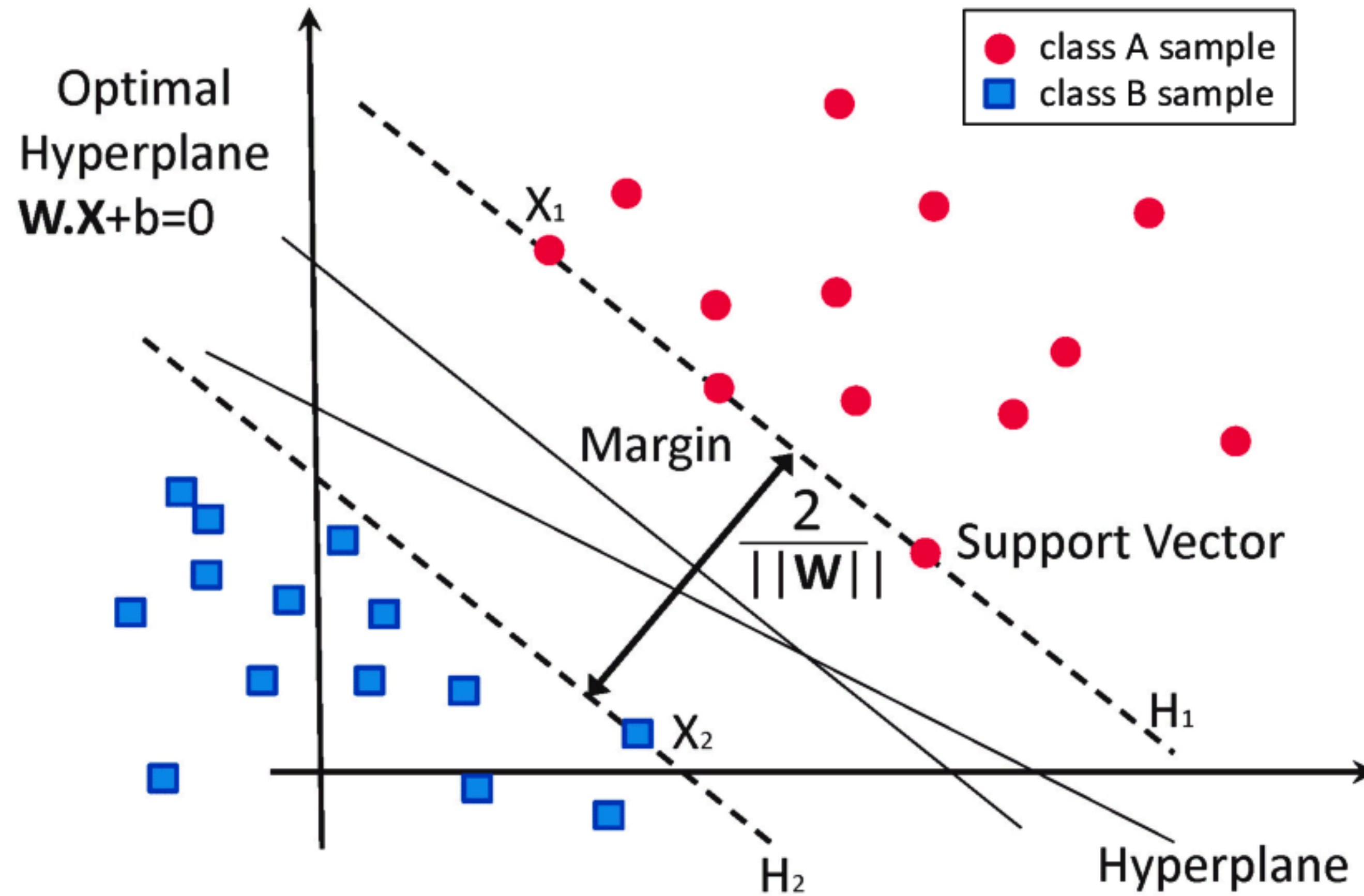
Bootstrap samples are created by randomly selecting samples from original dataset with replacement.

A decision tree trained on each bootstrap sample. Randomly selected subset of features are used for each tree (feature randomness).





# 서포트 벡터 머신

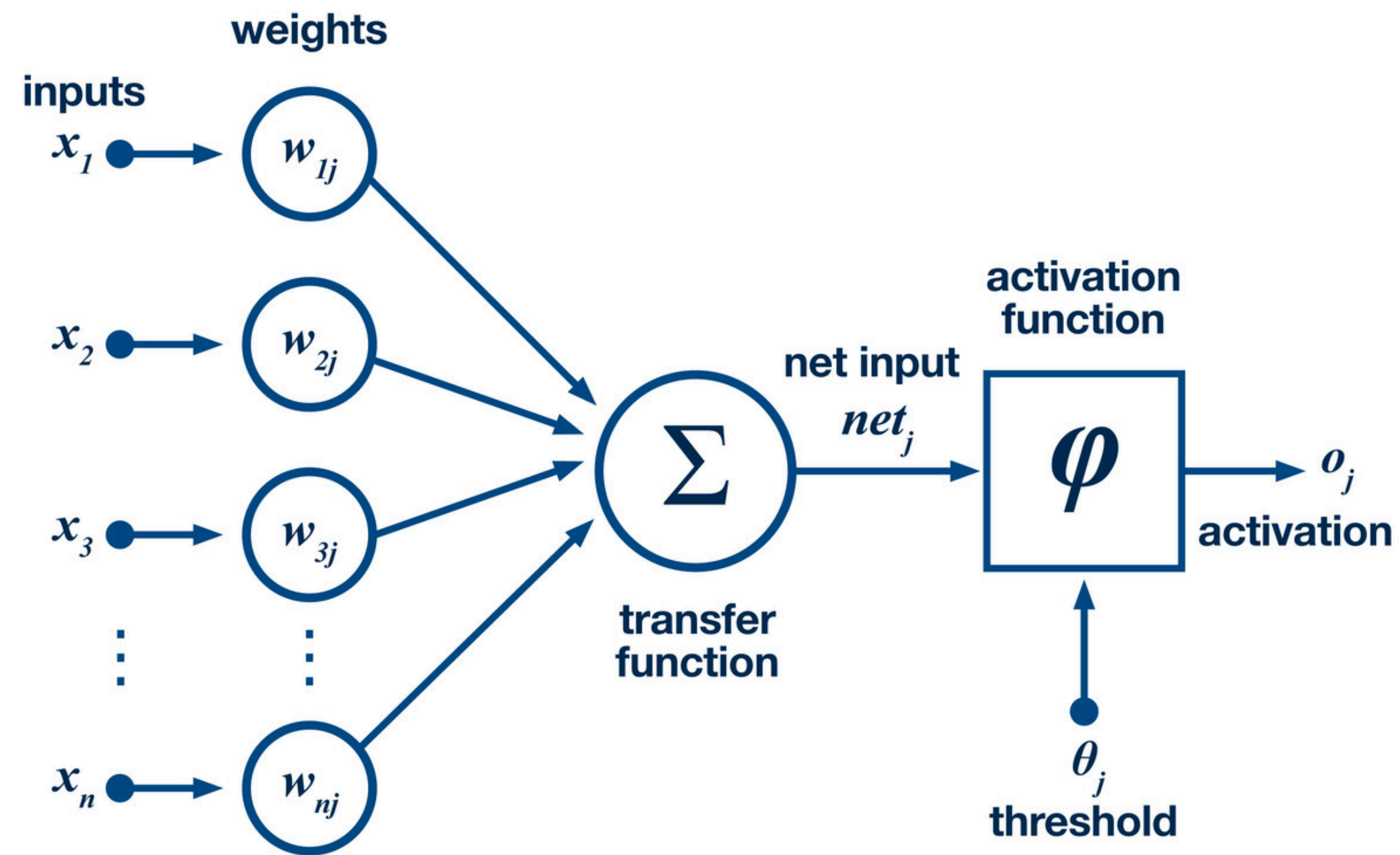


# 서포트 벡터 머신

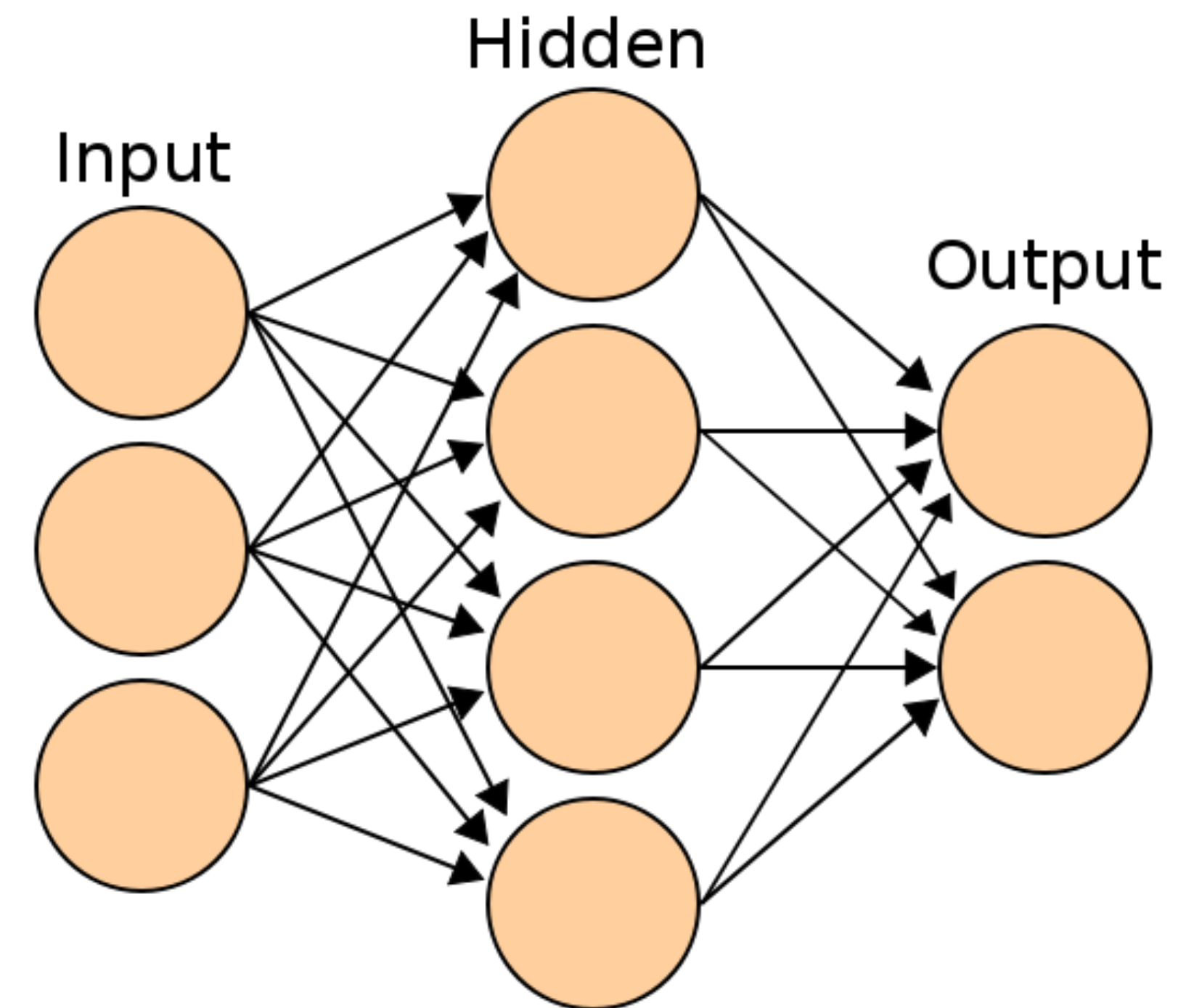
- 두 범주를 최대 마진으로 분리하는 초평면 탐색
- 결정 경계와 가장 가까운 “서포트 벡터”와의 거리를 최대화
- 커널 트릭을 활용한 고차원 공간 상 선형 분리 → 저차원 공간 상 비선형 분리
  - 커널 트릭 : 고차원 공간에서의 내적을 저차원 공간 상에서 계산하는 기법



# 인공신경망



인공 신경



인공 신경망

# 인공신경망

- 인공 신경 : Vector-to-Scalar Nonlinear function
- 레이어 : Vector-to-Vector Nonlinear function (입력을 공유하는 인공신경의 집합)
- 인공신경망 : Vector-to-Vector Nonlinear function (레이어들의 집합)

# AutoML 활용 예시

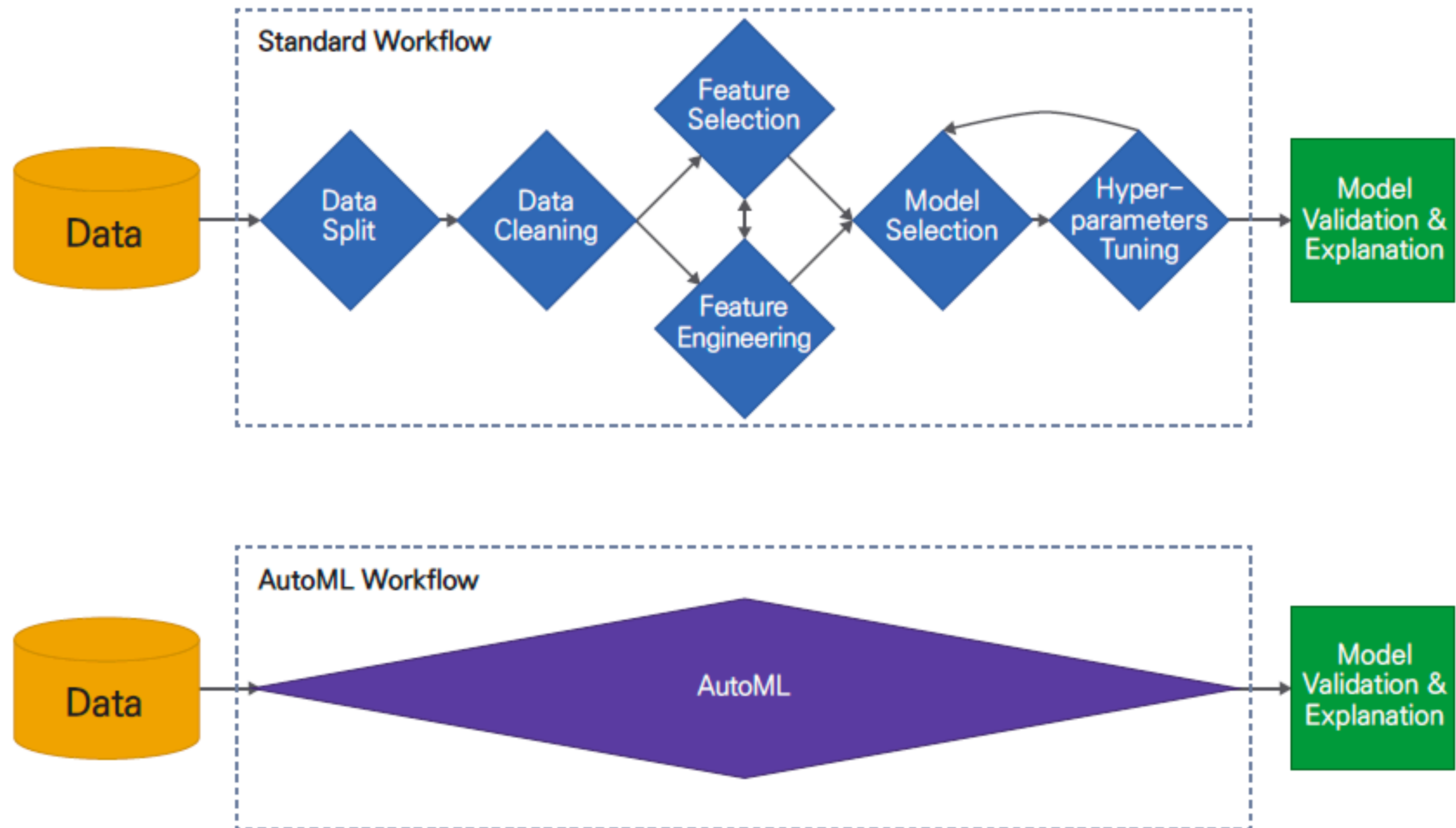
	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lr	Logistic Regression	0.7689	0.8047	0.5602	0.7208	0.6279	0.4641	0.4736	1.5090
ridge	Ridge Classifier	0.7670	0.0000	0.5497	0.7235	0.6221	0.4581	0.4690	0.0360
lda	Linear Discriminant Analysis	0.7670	0.8055	0.5550	0.7202	0.6243	0.4594	0.4695	0.0680
rf	Random Forest Classifier	0.7485	0.7911	0.5284	0.6811	0.5924	0.4150	0.4238	0.1640
nb	Naive Bayes	0.7427	0.7955	0.5702	0.6543	0.6043	0.4156	0.4215	0.0490
catboost	CatBoost Classifier	0.7410	0.7993	0.5278	0.6630	0.5851	0.4005	0.4078	0.2030
gbc	Gradient Boosting Classifier	0.7373	0.7918	0.5550	0.6445	0.5931	0.4013	0.4059	0.0960
ada	Ada Boost Classifier	0.7372	0.7799	0.5275	0.6585	0.5796	0.3926	0.4017	0.1100
et	Extra Trees Classifier	0.7299	0.7788	0.4965	0.6516	0.5596	0.3706	0.3802	0.1560
qda	Quadratic Discriminant Analysis	0.7282	0.7894	0.5281	0.6558	0.5736	0.3785	0.3910	0.0460
lightgbm	Light Gradient Boosting Machine	0.7133	0.7645	0.5398	0.6036	0.5650	0.3534	0.3580	0.2690
knn	K Neighbors Classifier	0.7001	0.7164	0.5020	0.5982	0.5413	0.3209	0.3271	0.0590
dt	Decision Tree Classifier	0.6928	0.6512	0.5137	0.5636	0.5328	0.3070	0.3098	0.0500
xgboost	Extreme Gradient Boosting	0.6853	0.7516	0.4912	0.5620	0.5216	0.2887	0.2922	0.0840
dummy	Dummy Classifier	0.6518	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0410
svm	SVM - Linear Kernel	0.5954	0.0000	0.3395	0.4090	0.2671	0.0720	0.0912	0.0400

# **Chapter 2.**

# **AutoML Methodologies**



# Automation of Machine Learning



# AutoML Methods

Hyperparameter  
Optimization  
(HPO)

Meta  
Learning

Neural  
Architecture  
Search  
(NAS)

# Hyper Parameter Optimization

# HPO

- 머신러닝 모델의 하이퍼 파라미터 최적화
  - 머신러닝 모델 적용에 있어서 반복적인 노력이 필요한 부분에 도움을 줄 수 있음.
  - 하이퍼 파라미터의 탐색을 통해 머신러닝 모델의 성능 향상 효과를 얻을 수 있음.
  - 재현성 향상 및 연구 결과 비교의 공정성을 향상 시킬 수 있음.
- CASH : Combined Algorithm Selection and Hyperparameter optimization
- FMS : Full Model Selection



# HPO 문제 정의

- hyperparameter configuration space as  $\Lambda = \Lambda_1 \times \Lambda_2 \times \cdots \times \Lambda_N$
- Hyperparameter vector  $\lambda \in \Lambda$
- $\mathcal{A}_\lambda$  : Algorithm with hyperparameter  $\lambda$
- $\mathbf{V}$  : validation protocol (e.g., cross validation, hold-out validation)

Given a data set  $\mathcal{D}$ , our goal is to find

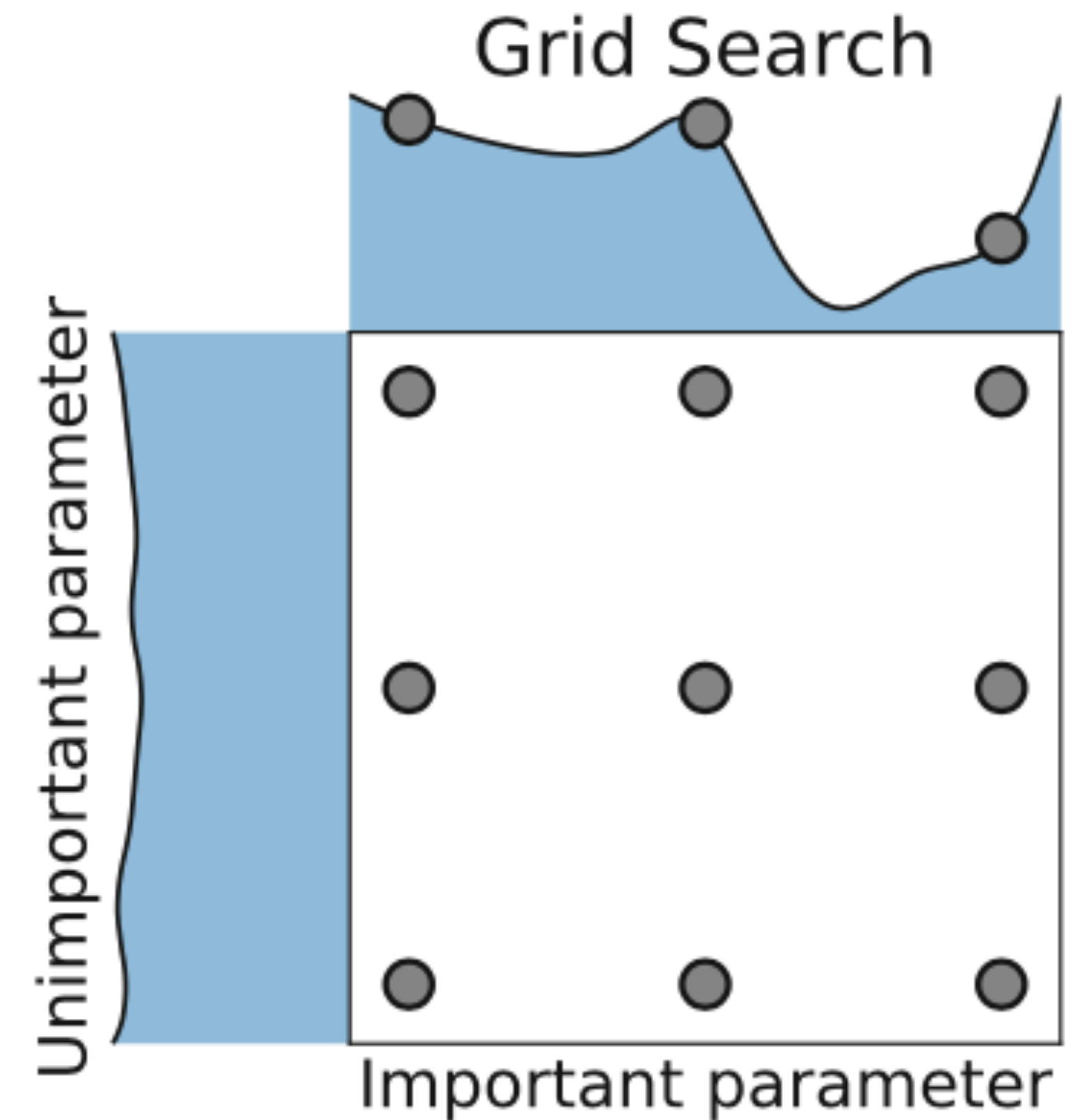
$$\lambda^* = \operatorname{argmin}_{\lambda \in \Lambda} \mathbb{E}_{(D_{train}, D_{valid}) \sim \mathcal{D}} \mathbf{V}(\mathcal{L}, \mathcal{A}_\lambda, D_{train}, D_{valid}),$$

# Challenges in HPO

- **Function evaluations can be extremely expensive.**
- **The configuration space is often complex (comprising a mix of continuous, categorical and conditional hyperparameters) and high-dimensional.**
- **We usually don't have access to a gradient of the loss function with respect to the hyperparameters.**
- **One cannot directly optimize for generalization performance as training datasets are of limited size.**

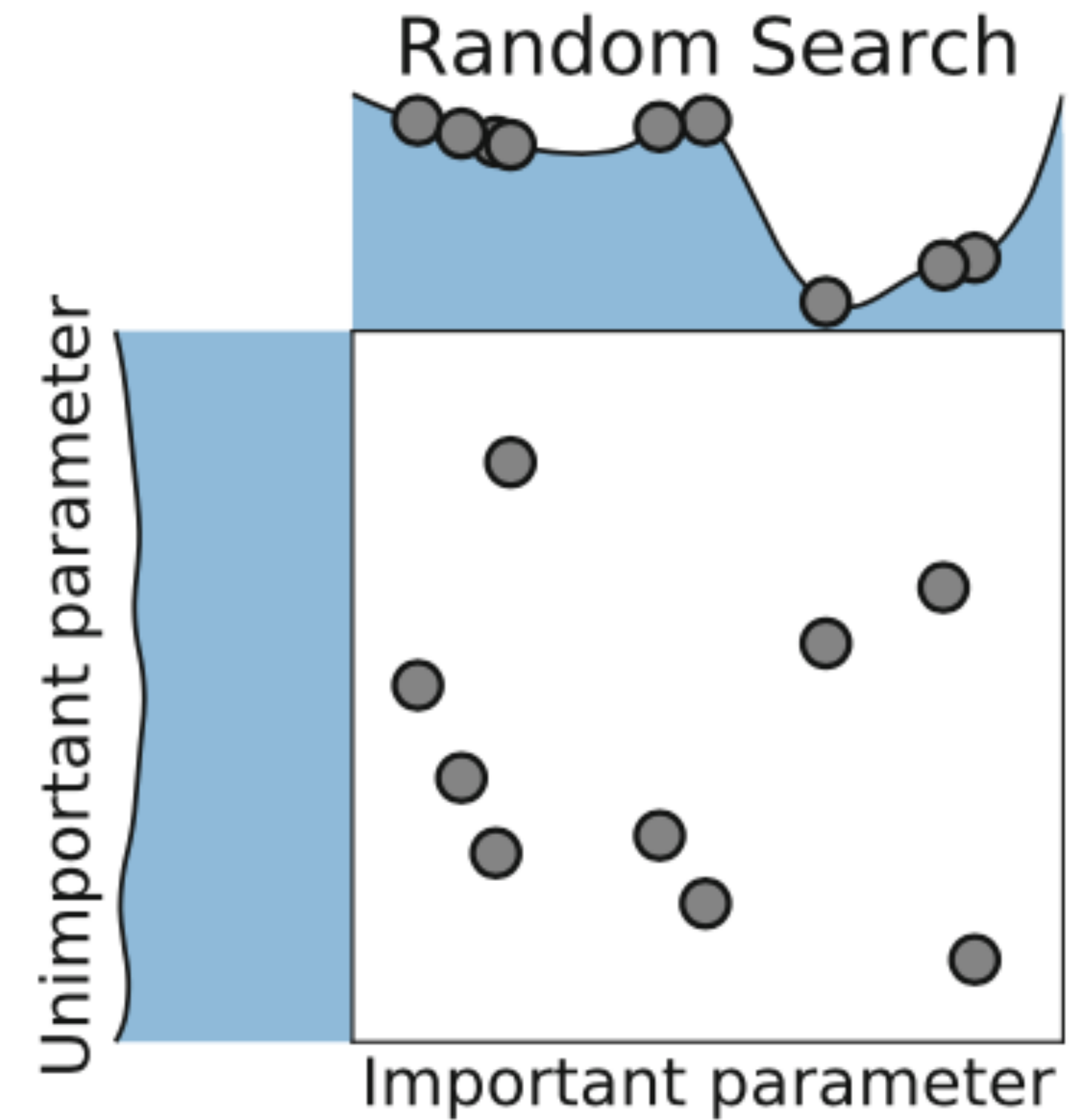
# Grid search

- 하이퍼 파라미터 변수 별로 후보 군을 선정
- 모든 조합에 대하여 모델을 평가함.
- Curse of dimensionality :  
Λ의 차원이 증가함에 따라서 필요한 평가 횟수가  
지수적으로 증가.



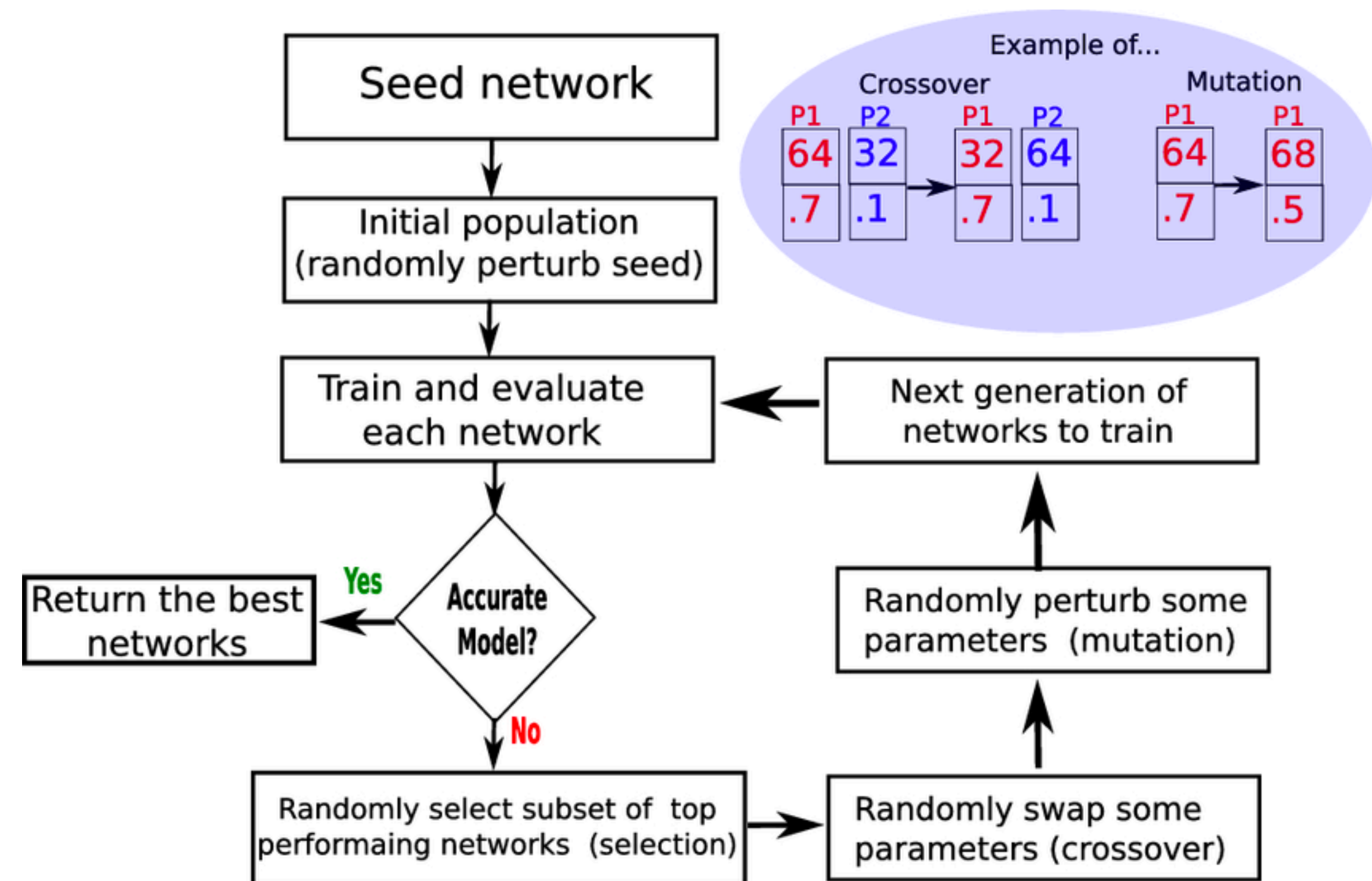
# Random search

- 하이퍼 파라미터 변수의 분포를 설정.
- 설정된 분포 안에서 샘플링하여 평가.
- 특정 변수의 영향이 클때 더욱 잘 동작함.
- 무작위로 샘플링하기때문에 변수값이 매번 달라짐.



# Population-based method

- Genetic algorithm
- Evolutionary algorithm
- Particle swarm optimization.

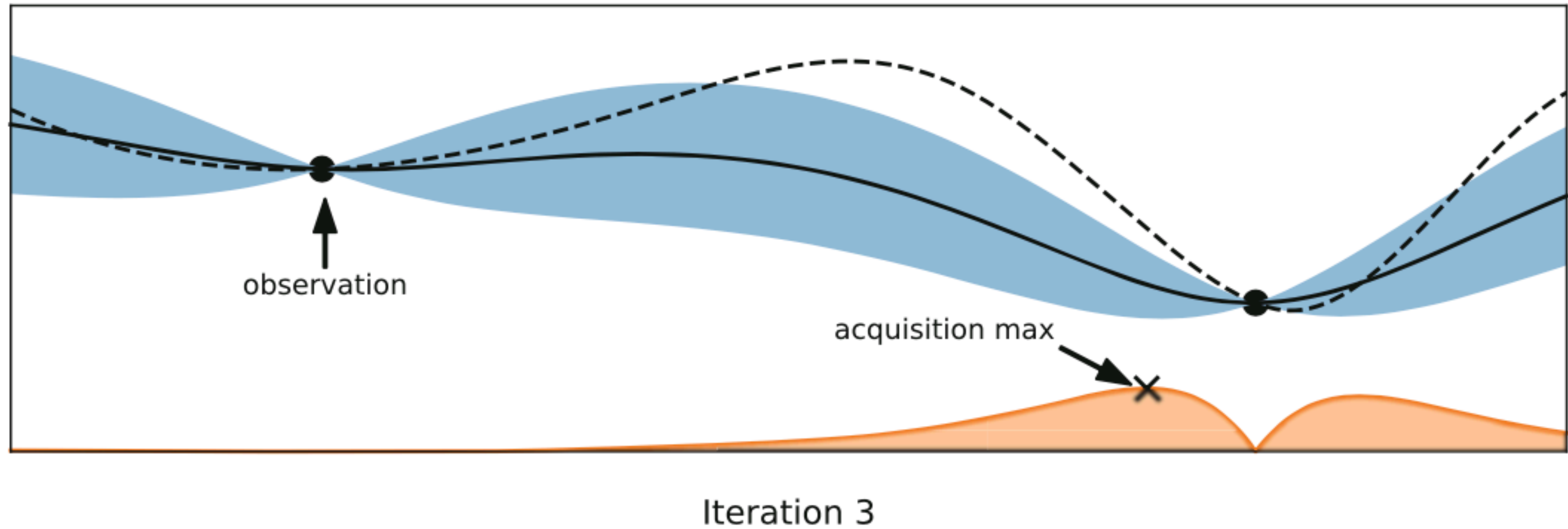


Example of genetic algorithm for HPO

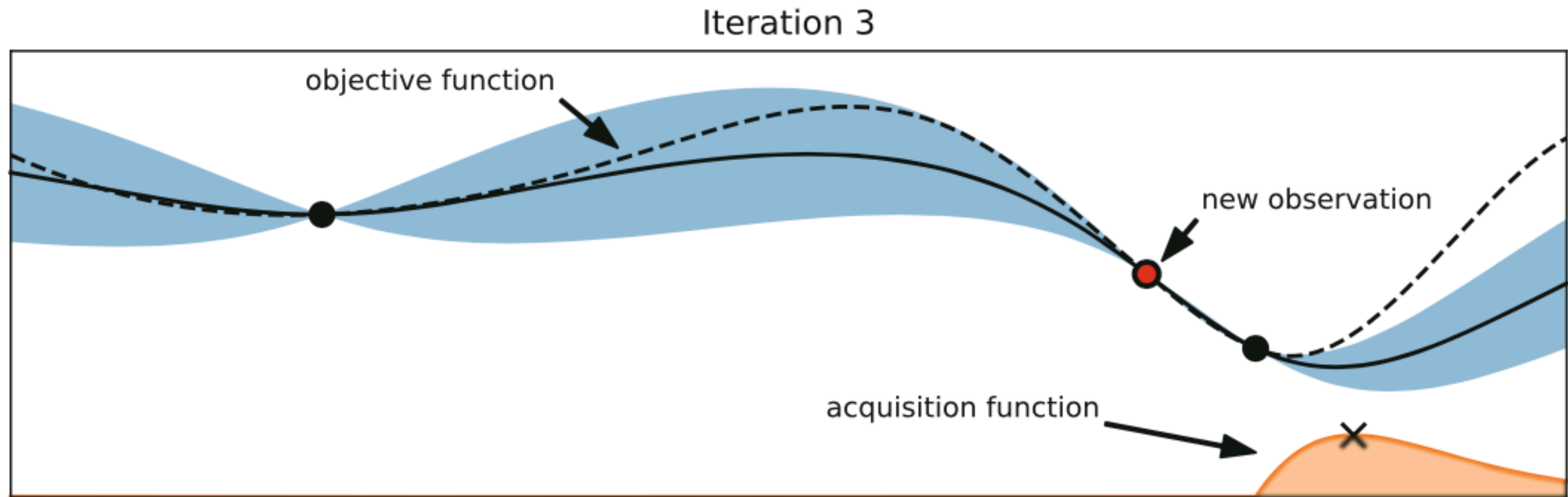
# Bayesian Optimization

- 베이지안 최적화 :  
함수의 형태를 가정하지 않는 블랙박스 함수의 전역 최적화를 위한 순차적 전략
- Surrogate model:  
원래 함수를 대체 하는 모델.  
HPO에서는 하이퍼파라미터에 대한 머신러닝 모델의 성능 함수의 대리 모델로,  
주로 가우시안 프로세스(GP)를 활용. GP 대신 NN, RF 기반 방법도 존재.
- Acquisition function:  
과거의 결과로부터 탐색 해야 할 파라미터를 결정해주는 함수.

# 베이지안 최적화 예시

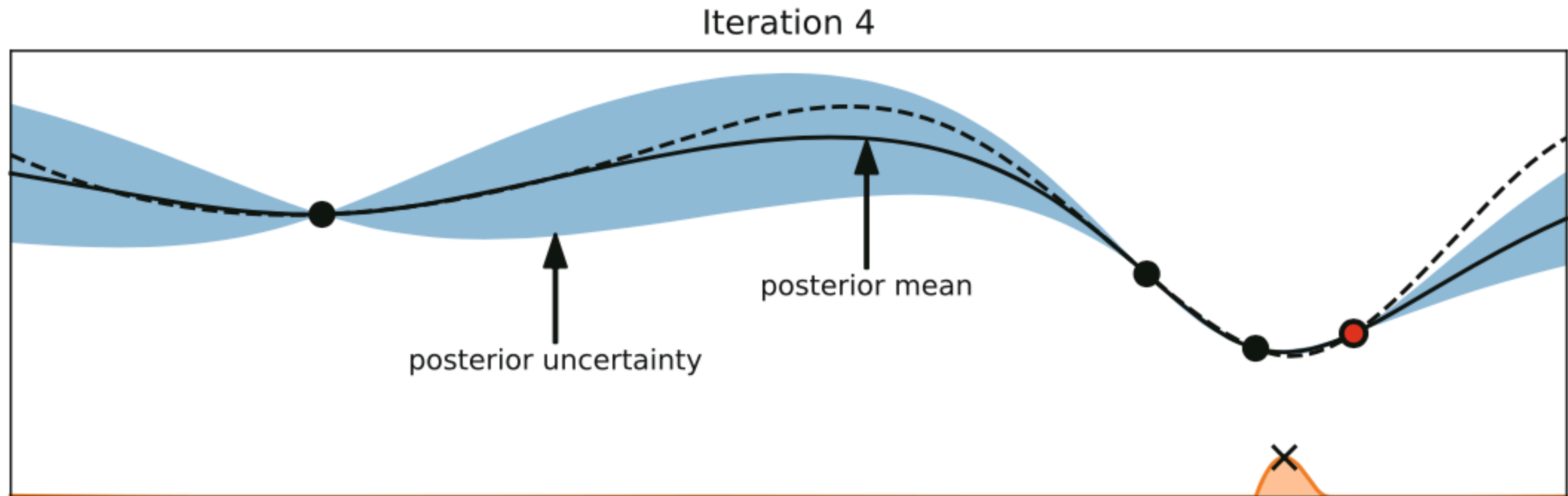


# 베이지안 최적화 예시





# 베이지안 최적화 예시



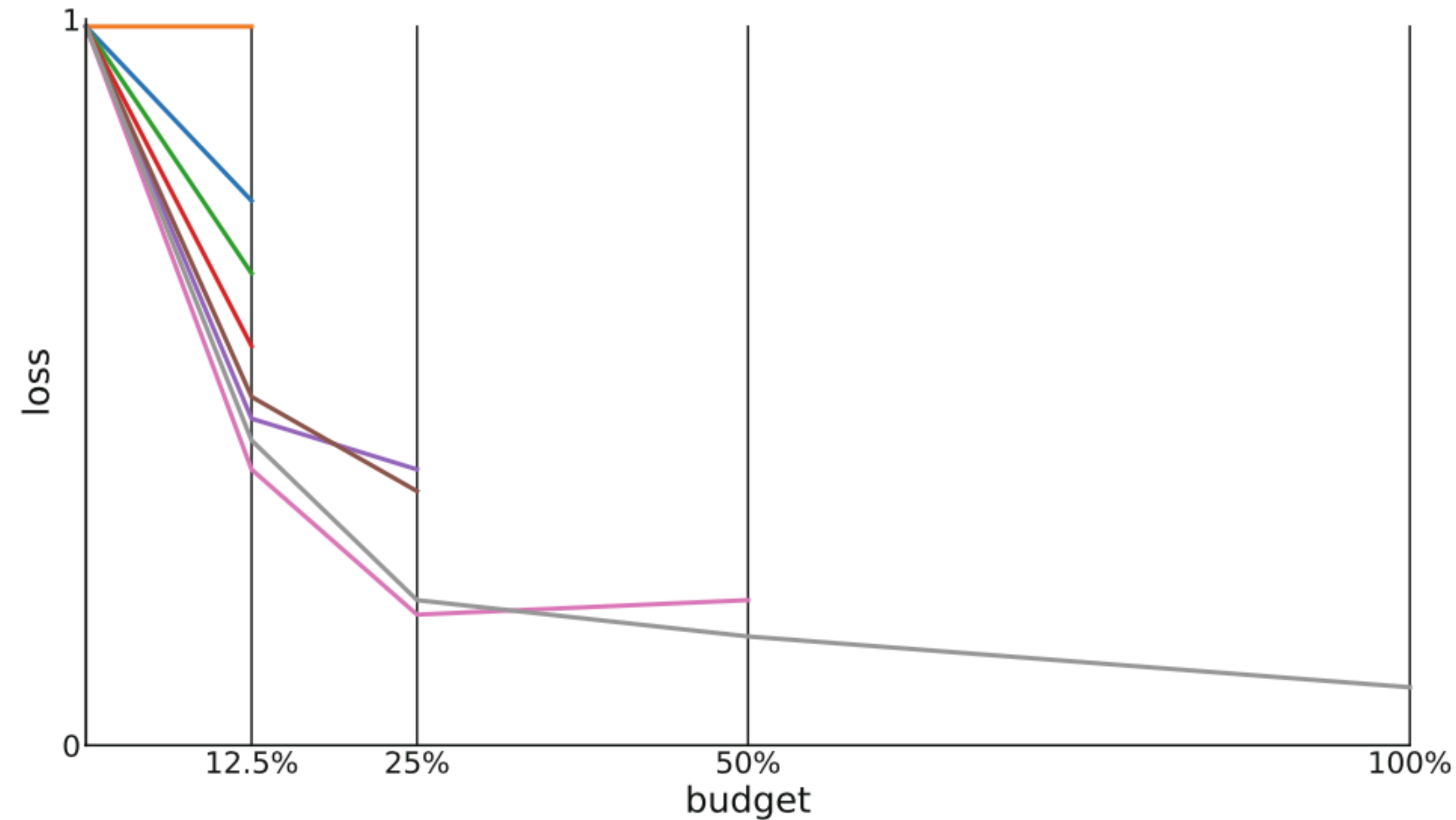
# Multi-fidelity optimization

- Fidelity (충실도) : High-fidelity (Hi-Fi) & Low-fidelity (Lo-Fi)
- Increasing dataset sizes and increasingly complex models are a major hurdle in HPO.
- Low-fidelity approximation of the actual loss function.

# Multi-fidelity for HPO

- Learning Curve-Based Prediction for Early Stopping:  
탐색 과정에서 학습 곡선을 평가하여 조기종료함으로써 리소스를 효율적으로 활용.
- Bandit-Based Algorithm Selection Methods:  
Low fidelity approximation을 바탕으로 후보를 추출.
  - Successive Halving, Hyperband.

# Successive halving



**Fig. 1.3** Illustration of successive halving for eight algorithms/configurations. After evaluating all algorithms on  $\frac{1}{8}$  of the total budget, half of them are dropped and the budget given to the remaining algorithms is doubled

# Meta Learning

# Meta Learning

- Learning to learn.
- 연관된 태스크에 대한 과거의 학습 경험을 바탕으로 현재의 머신러닝 문제에 활용.
- meta-data : data that describe prior learning tasks and previously learned models.
  - e.g., 알고리즘, 하이퍼파라미터, 태스크, 모델 평가 결과 등

# Meta learning problem formulation

- Task :  $t_j \in T$
- Configuration :  $\theta_i \in \Theta$ , e.g., hyperparameter, pipeline component
- Evaluation  $P(\theta_i, t_j)$  : scalar evaluation (e.g., accuracy with CV) of configuration  $i$  on task  $j$ .
- $\mathbf{P}$  : set of prior evaluations
- $\mathbf{P}_{new}$  : set of known evaluation  $P(\theta_i, t_{new})$  on a new task  $t_{new}$
- $L$  : meta learner that learning from  $\mathbf{P}$  &  $\mathbf{P}_{new}$ .

# Meta learning in AutoML

- Recommendation : Configuration set을 추천할 수 있는 모델 학습
- Learning curve estimation : 태스크간 유사성을 바탕으로 학습 곡선 예측
- Performance prediction : 새로운 태스크에 대한 성능 예측



# Neural Architecture Search

# Neural Architecture Search

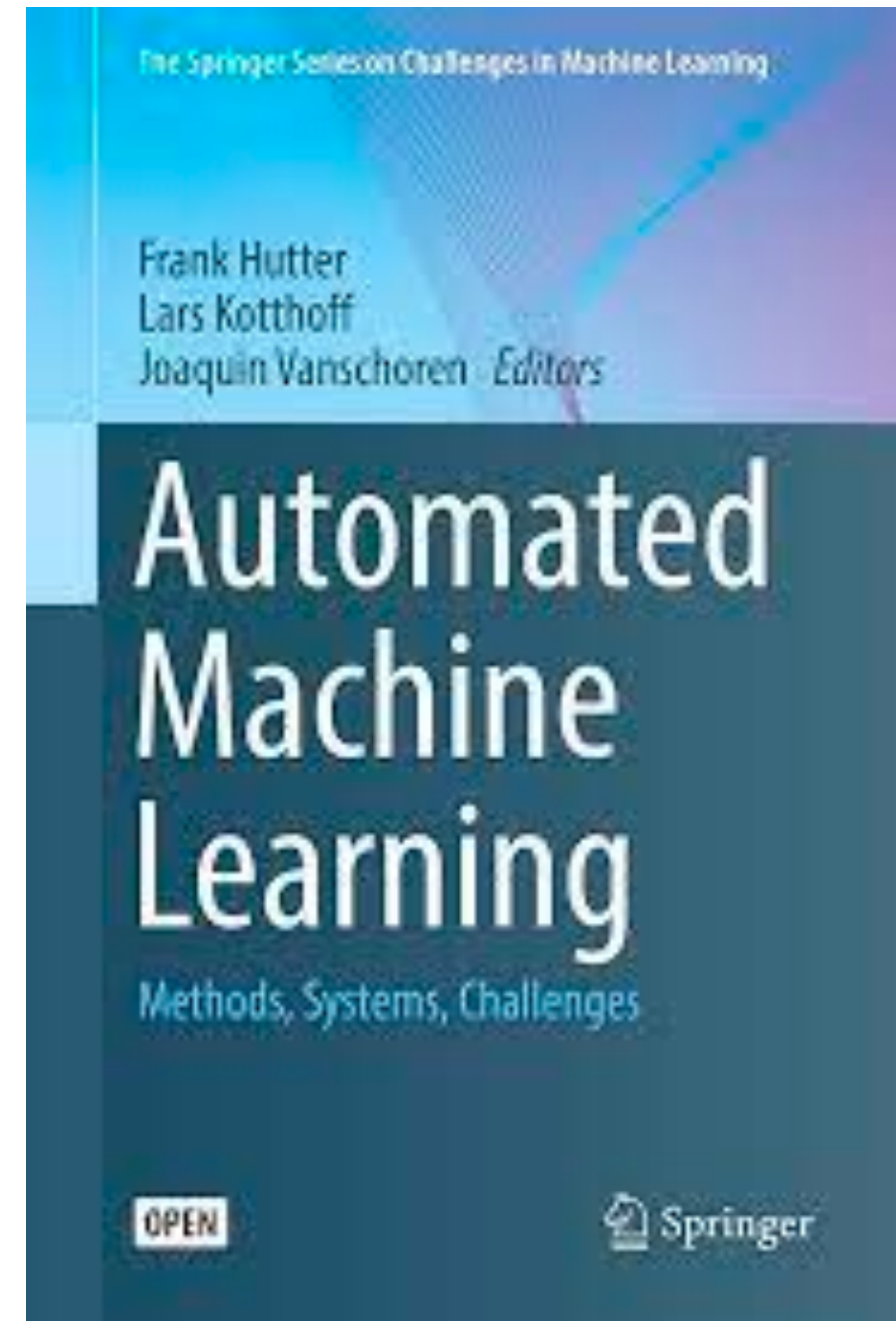
- 인공신경망의 구조는 성능에 매우 큰 영향을 미치며, 일반적으로 연구자의 판단에 근거하여 설계.
- 커널 크기, 레이어의 종류, 레이어의 연결 등 인공신경망의 구조는 굉장히 다양한 조합을 갖을 수 있음.
- 이 모두를 하이퍼 파라미터로 보았을때 인공신경망은 일반적인 머신러닝 모델에 비해 많은 하이퍼 파라미터를 가지고있음.
- NAS는 이러한 네트워크의 구조 설계 과정을 자동화하여 최적의 구조를 찾는 방법임.

# AutoML 라이브러리

- Python
  - Auto-Sklearn : <https://automl.github.io/auto-sklearn/master/#>
  - TPOT : <http://epistasislab.github.io/tpot/>
  - H2O : <https://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html>
  - Pycaret : <https://pycaret.org/>
  - Auto-Keras : <https://autokeras.com/>
  - Auto-Gluon : <https://auto.gluon.ai/stable/index.html>
  - ML-box : <https://mlbox.readthedocs.io/en/latest/>
  - MLJAR : <https://supervised.mljar.com/>
- Java
  - Auto-WEKA : <https://www.cs.ubc.ca/labs/algorithms/Projects/autoweeka/>
- Scala
  - TransmogrifyAI : <https://transmogrif.ai/>

# Automated Machine Learning

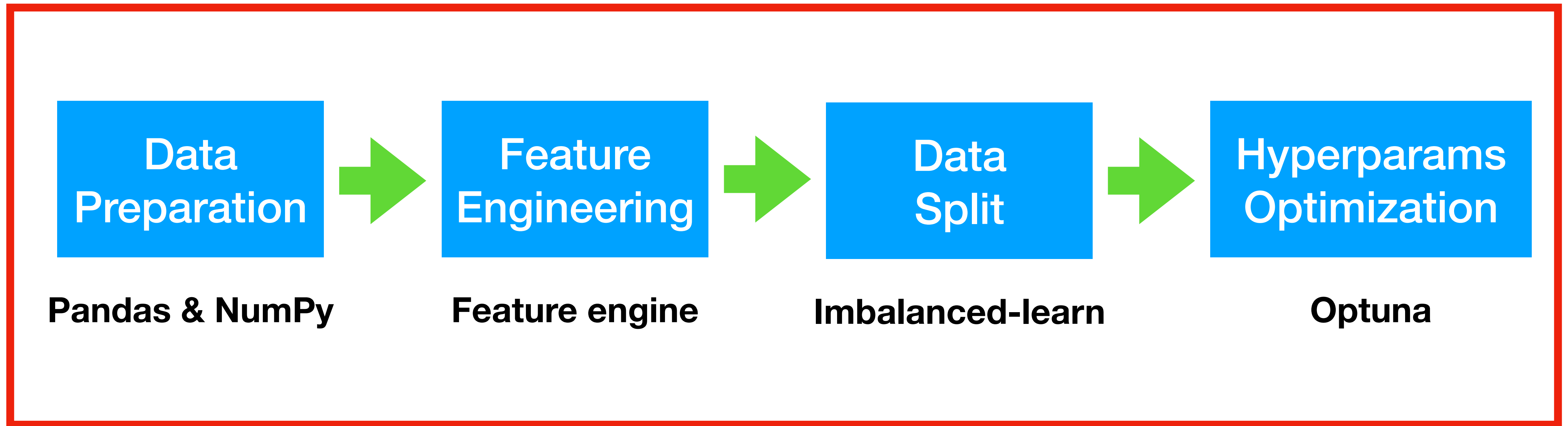
- <https://www.automl.org/book/>



# Chapter 3.

## Hands on AutoML

# 실습 내용 소개



머신러닝 : Scikit Learn, PyCaret

딥러닝 : AutoKeras

# Pandas

- 데이터 분석 및 조작을 위한 오픈 소스 파이썬 라이브러리.
- 데이터 정렬, 데이터 슬라이싱 및 인덱싱, 누락된 데이터 처리, 데이터셋 병합 및 결합 등 데이터 처리에 필요한 다양한 기능을 제공.
- 주요 데이터 구조로는 1차원 배열과 같은 'Series'와 2차원 배열과 같은 'DataFrame'이 있음.
- NumPy, Matplotlib 등 다른 파이썬 라이브러리와 호환성이 높음.

# Pandas dataframe 예시

The diagram illustrates a Pandas DataFrame with the following structure and annotations:

	Column Label/ Header	0	1	2	3	4
Index Label		Name	Age	Marks	Grade	Hobby
0	S1	Joe	20	85.10	A	Swimming
1	S2	Nat	21	77.80	B	Reading
2	S3	Harry	19	91.54	A	Music
3	S4	Sam	20	88.78	A	Painting
4	S5	Monica	22	60.55	B	Dancing

Annotations:

- Column Label/ Header:** Points to the header row (Name, Age, Marks, Grade, Hobby).
- Index Label:** Points to the index column (S1, S2, S3, S4, S5).
- Column Index:** Points to the column headers (0, 1, 2, 3, 4).
- Row Index:** Points to the row indices (0, 1, 2, 3, 4).
- Column:** Points to the 'Marks' column.
- Row:** Points to the row containing 'S4'.
- Element/ Value/ Entry:** Points to the value '88.78' in the 'Marks' column of the row 'S4'.



# Numpy

- 파이썬의 수치 계산을 위한 핵심 라이브러리
- 다차원 배열 객체와 이러한 배열을 처리하기 위한 배열 연산 등 다양한 기능 제공
- Pandas, Matplotlib, Scikit-learn 등 파이썬 데이터 사이언스 라이브러리에 활용됨.

# Feature engine

- Feature engineering을 위한 파이썬 라이브러리.
- Scikit-learn 스타일로 간단하게 데이터 변환 가능 : fit(), transform()
- Feature-engine includes transformers for:
  - Missing data imputation
  - Categorical encoding
  - Discretisation
  - Outlier capping or removal
  - Variable transformation, etc.

# Imbalanced-learn

- 클래스 간 불균형이 존재하는 데이터의 학습을 위한 라이브러리.
- 학습/테스트 데이터 등 데이터셋 분할을 위한 언더샘플링 및 오버샘플링 기능 제공

# Optuna

- HPO를 위한 파이썬 라이브러리
- Sklearn, pytorch, tensor flow 등 다양한 ML 라이브러리와 활용 가능.
- TPE (Tree-structured Parzen Estimator) 및 CMA-ES, 랜덤 서치 등
- 병렬화 및 시각화 기능 제공

# PyCaret

- PyCaret은 Low-code 머신러닝 자동화 파이썬 라이브러리
- Scikit-learn, LightGBM, XGBoost, Optuna 등의 머신러닝 라이브러리 기반
- 데이터 전처리, 모델학습, 최적화, 결과 분석 및 시각화, 모델 배포 등 통합 기능 제공

감사합니다.