KIRD
국가과학기술인력개발원

Introduction to AutoML

세종대학교 AI로봇학과 류승형

목차

Chapter 1.
Brief introduction on ML models

태스크의 구분

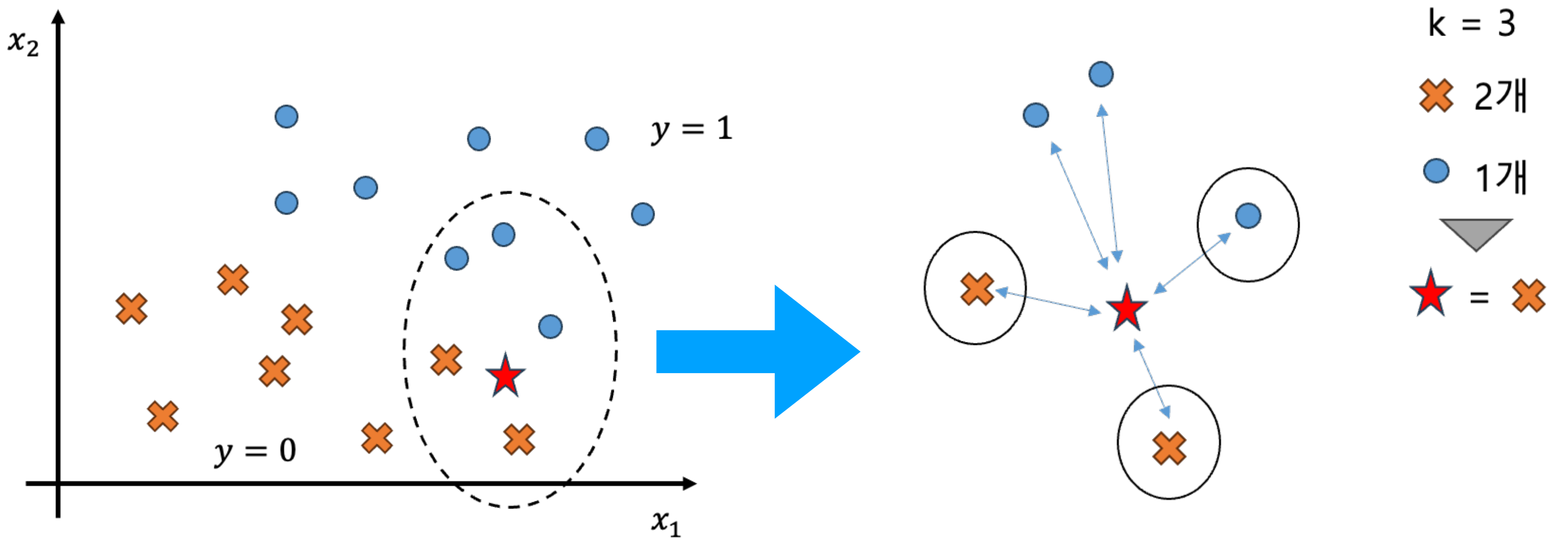
- 회귀 : 실수값, 연속적인 값 예측
- 분류 : 어떤 클래스에 해당하는지 예측
- 클러스터링 : 데이터를 여러개의 그룹으로 나누기
- 차원축소 : 데이터를 잘 설명하는 저차원 데이터로의 변환

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)

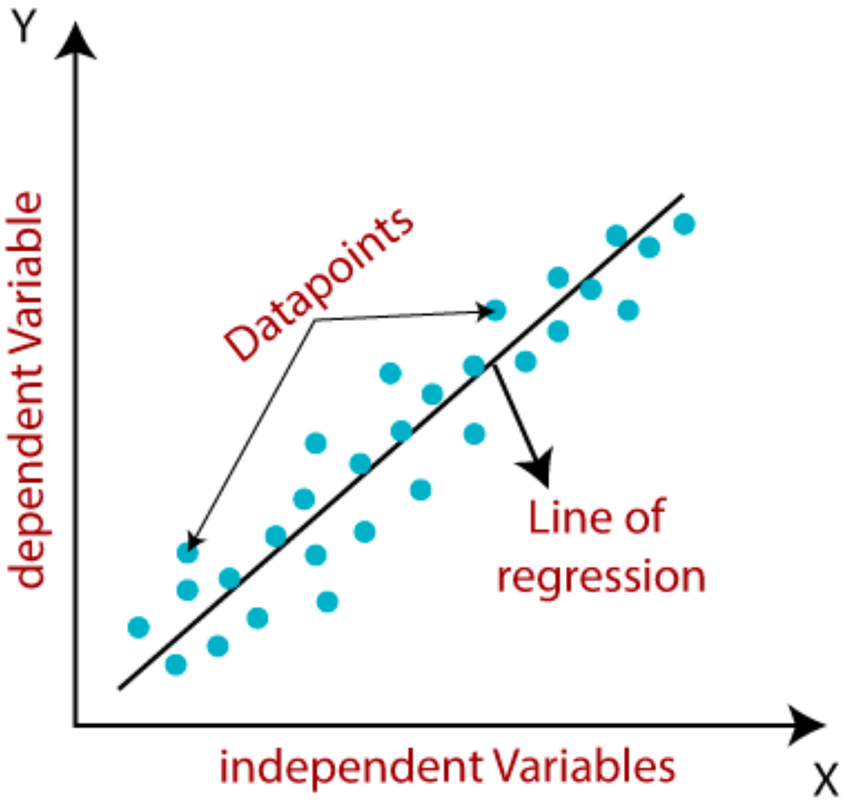
K-최근접 이웃 (KNN)

- 유유상종 : 비슷한 데이터는 가까이에 모여있을 것
- 최근접 이웃의 수, 거리계산 방법에 따라 결과가 달라짐
- 데이터에 대한 별도의 사전 학습 단계가 없음 → 추론시에 거리 계산 및 분류 수행, Lazy learning



선형 회귀

- 종속 변수 Y와 독립 변수 X 사이의 관계
- $Y = a + bx_1 + cx_2 + \ldots + \epsilon$
- 회귀 계수(a, b, c)와 선형 관계
- 선형 회귀에서의 학습은?
- 오차를 최소화 하는 회귀 계수를 찾는 과정
- 정규방정식과 경사하강법

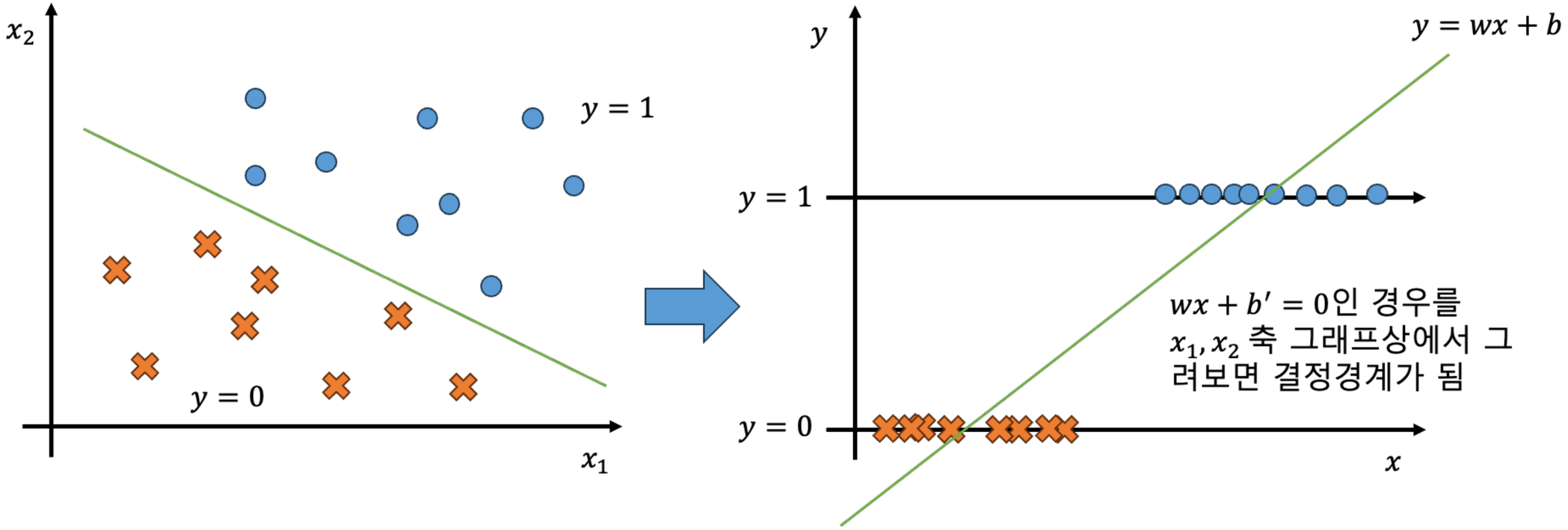


선형 회귀

특성	**Lasso regression** ***L*1 페널티**	**Ridge regression** ***L*2 페널티**
목적	회귀 계수를 0으로 만들어 모델을 단순화가능 특성 선택 가능	계수의 크기를 제한하여 모든 계수가 작게 유지되도록 함
수학적 표현	$\lambda \sum_{i=0}^{d} \lvert w_i \rvert$	$\lambda \sum_{i=0}^{d} (w_i)^2$
효과	변수의 수를 줄임으로써 sparse한 모델을 생성	계수를 축소하지만 모든 변수를 모델에 포함
최적화 난이도	경사하강법을 통한 최적화 수행	정규방정식을 통한 해석적 해가 존재
모델 해석성	높음 (중요 변수만 선택)	낮음 (모든 변수 포함, 계수 축소에 중점)
상황	변수의 수가 많고 중요 변수를 선택하고자 할 때 유용	변수 간의 상관관계가 높고, 변수 제거 없이 모델 복잡도를 제어하고자 할 때 유용

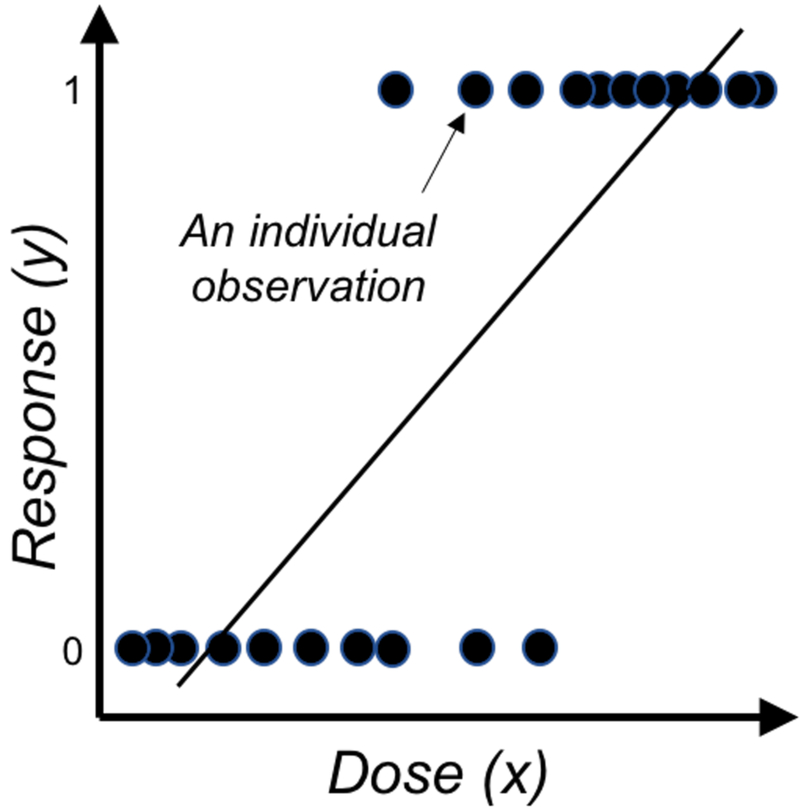
로지스틱 회귀

- 분류 문제를 위한 모델
- 선형 회귀 + 범주형 종속 변수 Y
  → 이진 분류 수행 p 또는 1-p의 확률로 범주 구분
- 선형 모델 사용시의 문제점은?

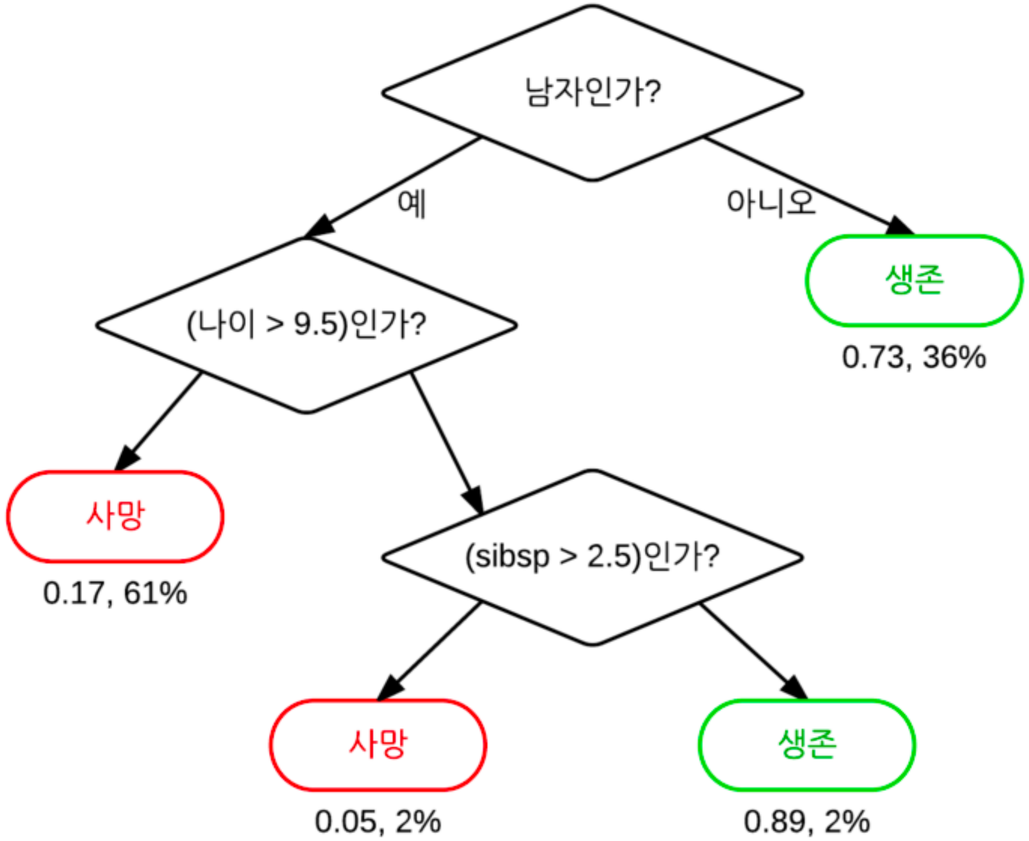


로지스틱 회귀

- Log-odd을 독립 변수의 선형 조합으로 예측
- Recall : 선형회귀로 실수값을 예측
- Odds 란? 어떤 사건이 발생할 확률을 p라 할 때 발생확률과 발생하지 않을 확률의 비율
  - p/(1-p) → 클래스가 1일 확률과 1이 아닐 확률
  - p=0일 때 0, p가 1에 가까워지면 ∞
  - Log-odd란? odds를 로그변환 → - ∞ ~ + ∞ (실수범위)
- Logistic function을 통해 확률로 변환: $p = 1/(1 + \exp(-z))$



의사 결정 트리

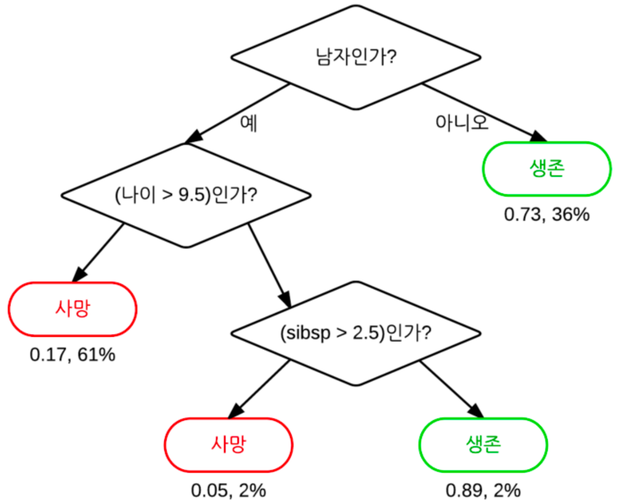


타이타닉 생존자 분류를 위한 의사 결정 트리

https://wooono.tistory.com/104

의사 결정 트리

- 일련의 분류 규칙을 통해 데이터를 구분
- 엔트로피(또는 불순도)가 낮아지는 방향으로 노드를 확장 → 규칙 생성
  - 높은 엔트로피 = 클래스가 섞여있다 = 데이터가 구분이 어려움
- 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.

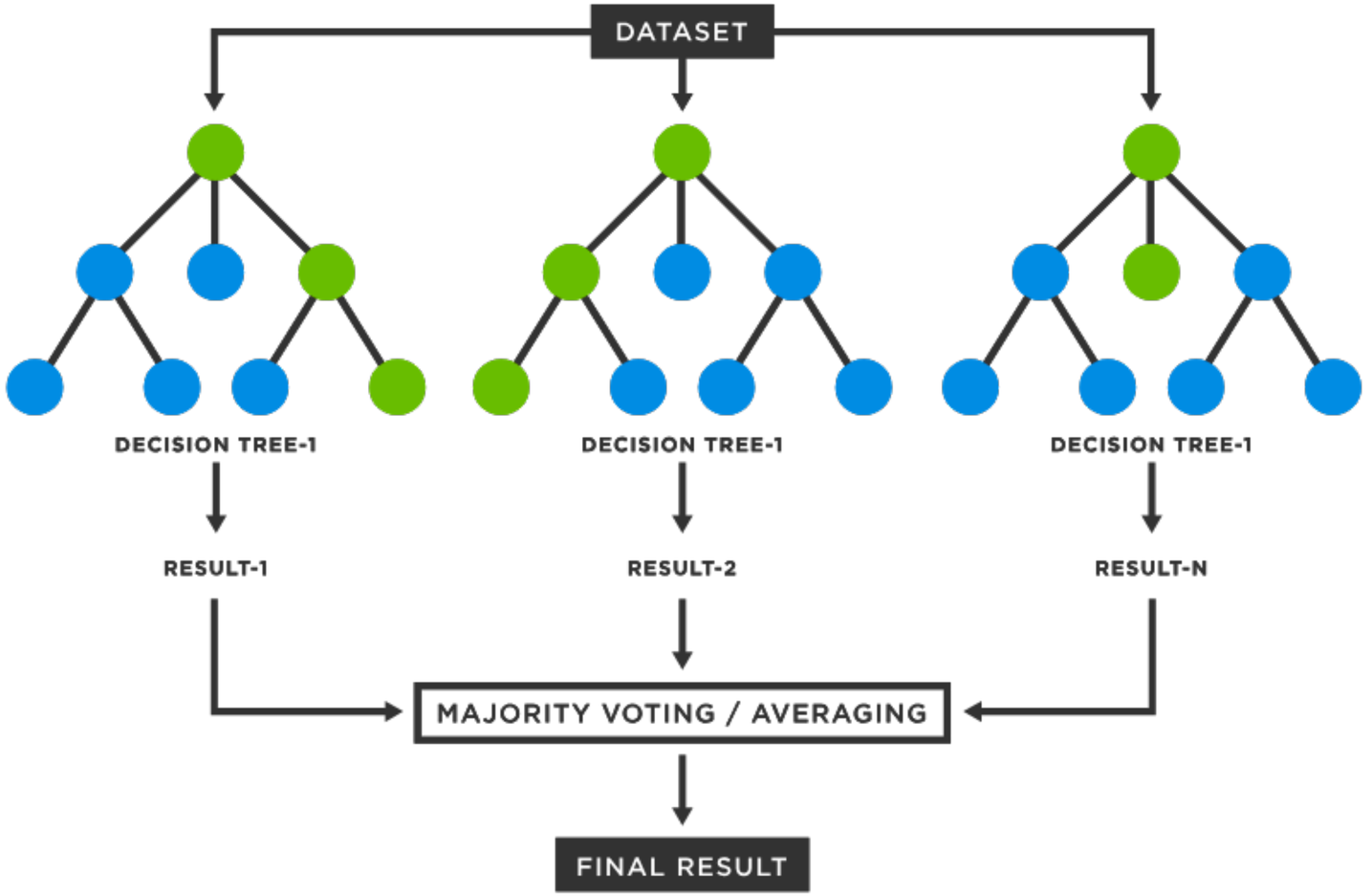


의사결정트리

앙상블 모델

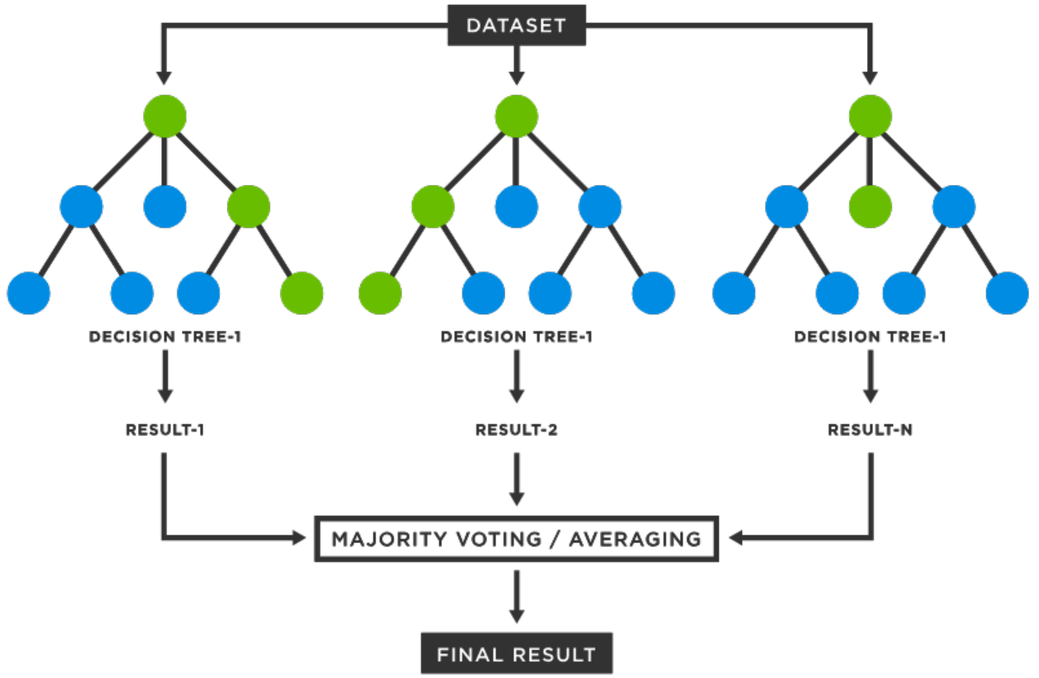
- 여러 모델을 결합하여 성능 향상을 이끌어 내는 방법
- 간단한 다수의 모델 >= 복잡한 단일 모델 → 집단 지성의 힘?
- 가장 단순하게는 여러 모델 학습 후 결과의 평균값을 활용 가능
- e.g., 사람의 키 예측하기
- 결정트리의 앙상블 → 랜덤 포레스트, GBDT

랜덤 포레스트


DATASET
DECISION TREE-1
DECISION TREE-1
DECISION TREE-1
RESULT-1
RESULT-2
RESULT-N
MAJORITY VOTING / AVERAGING
FINAL RESULT

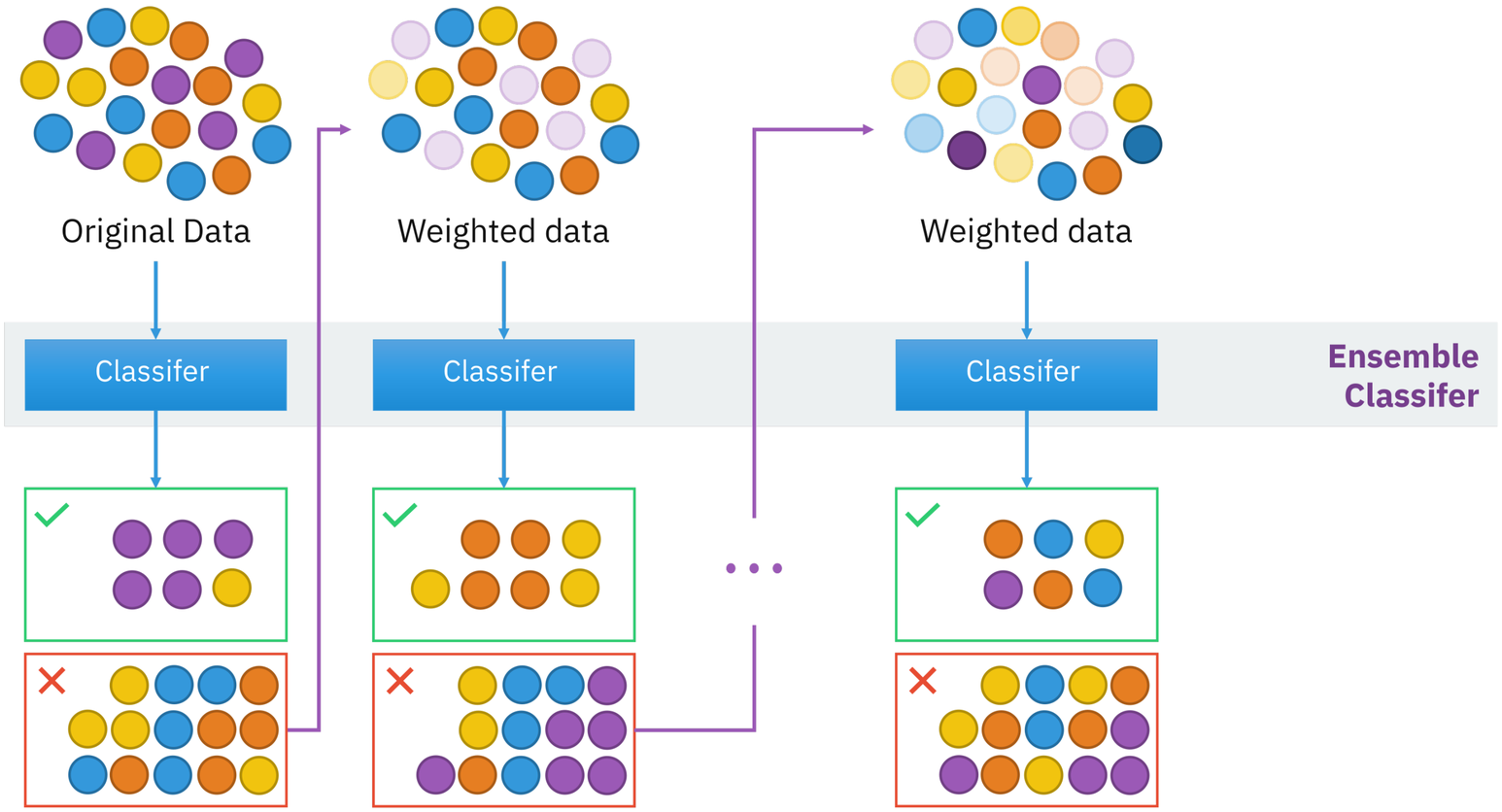
랜덤 포레스트

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



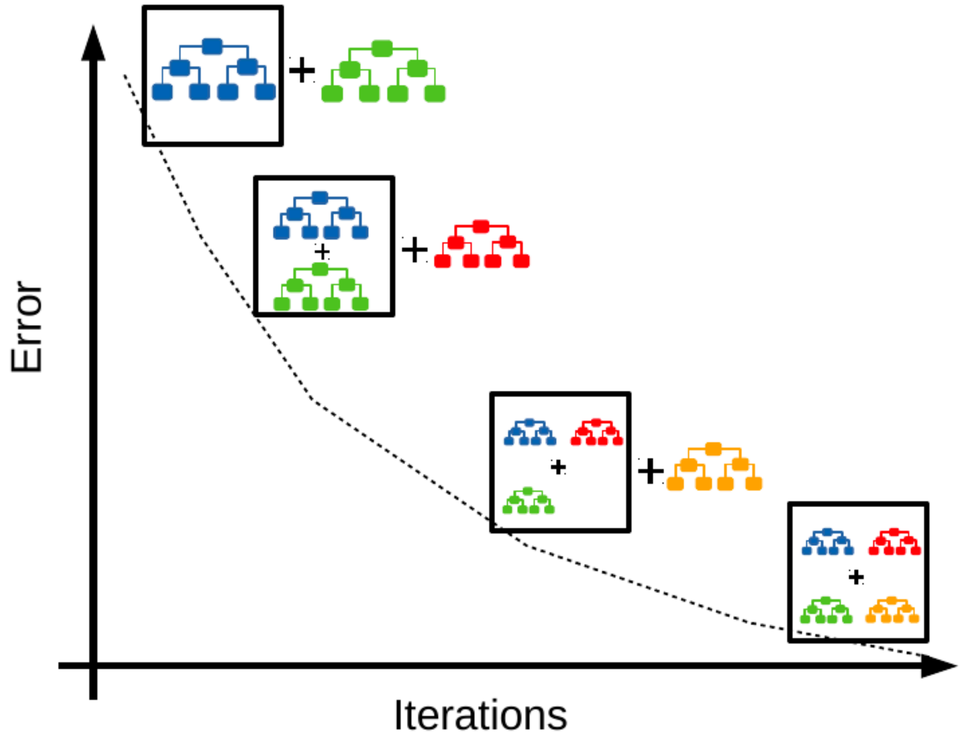
Boosting

- Boosting 기법
- 분류기들을 순차적으로 학습
- 이전에 잘못했던 부분을 더 잘 할 수 있도록

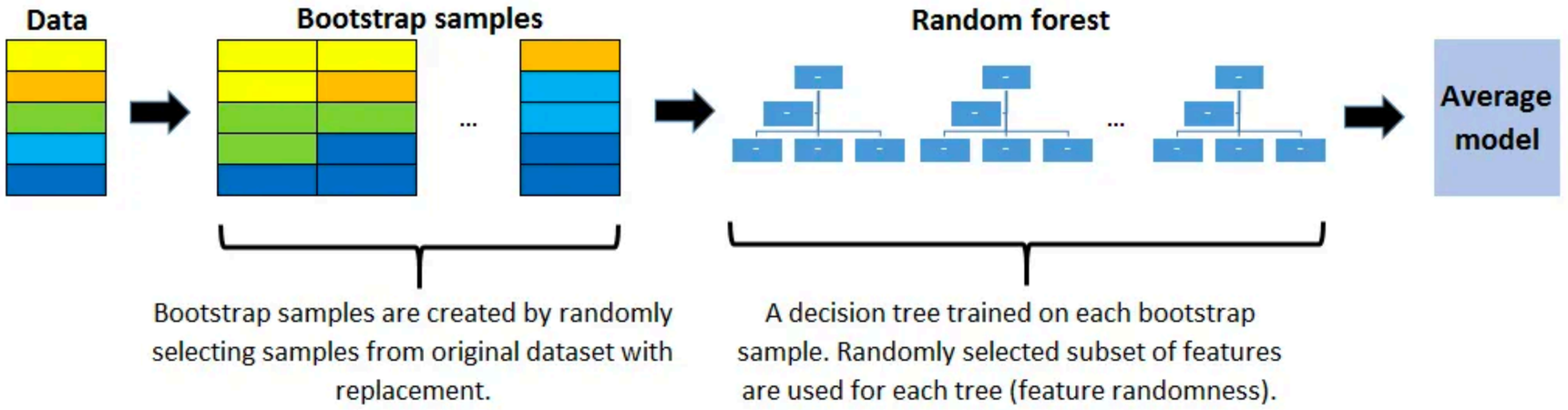


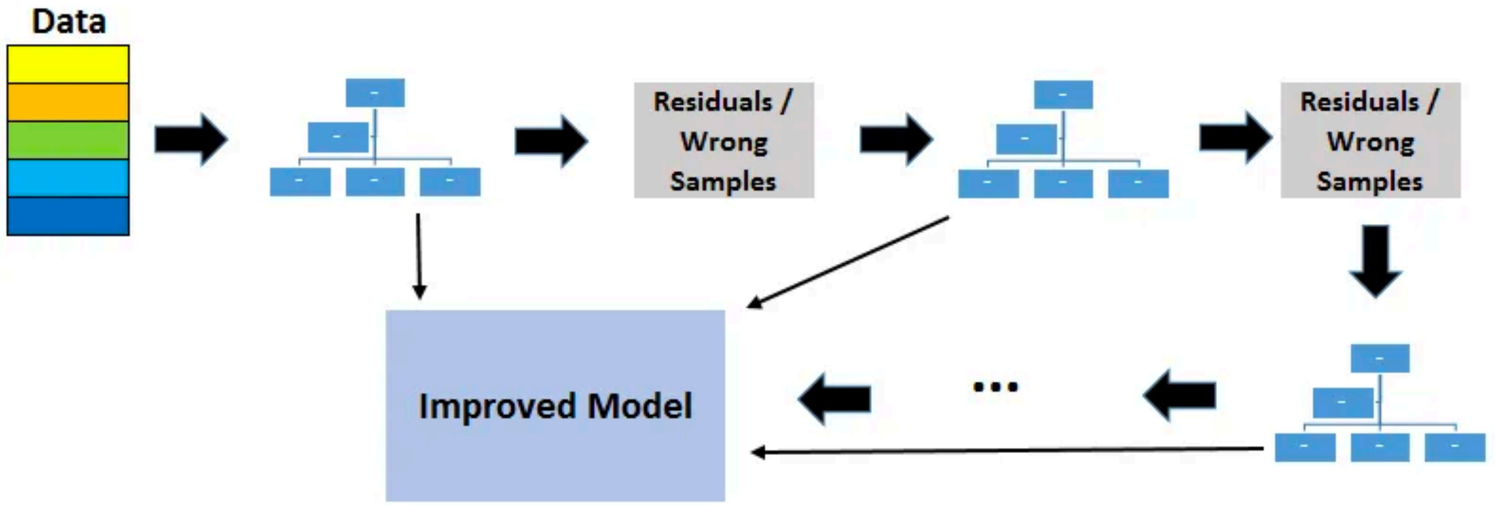
Gradient Boosted Decision Tree

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

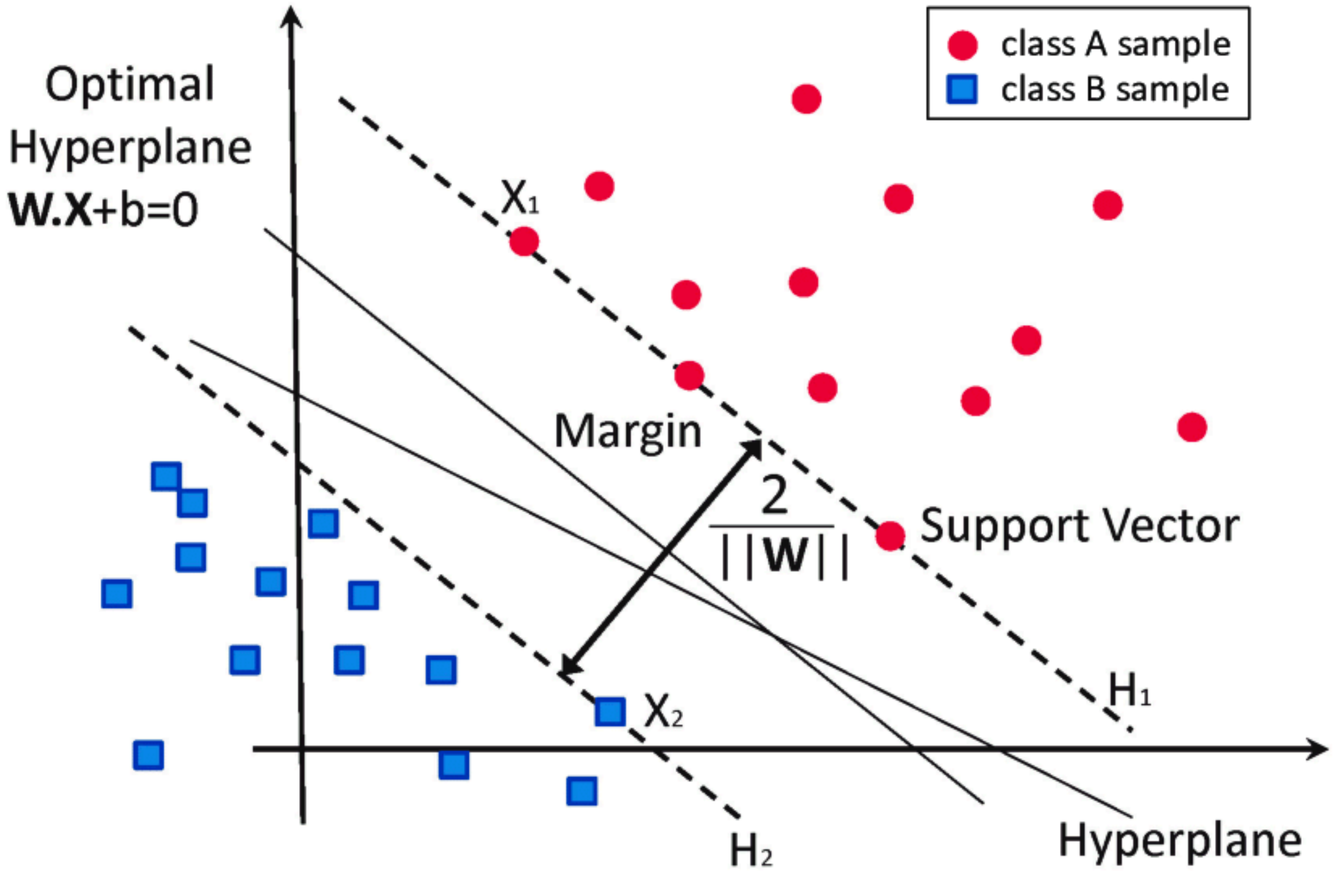


RF vs GBDT


Data
Bootstrap samples
Random forest
Average model
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).


Data
Residuals / Wrong Samples
Residuals / Wrong Samples
Improved Model

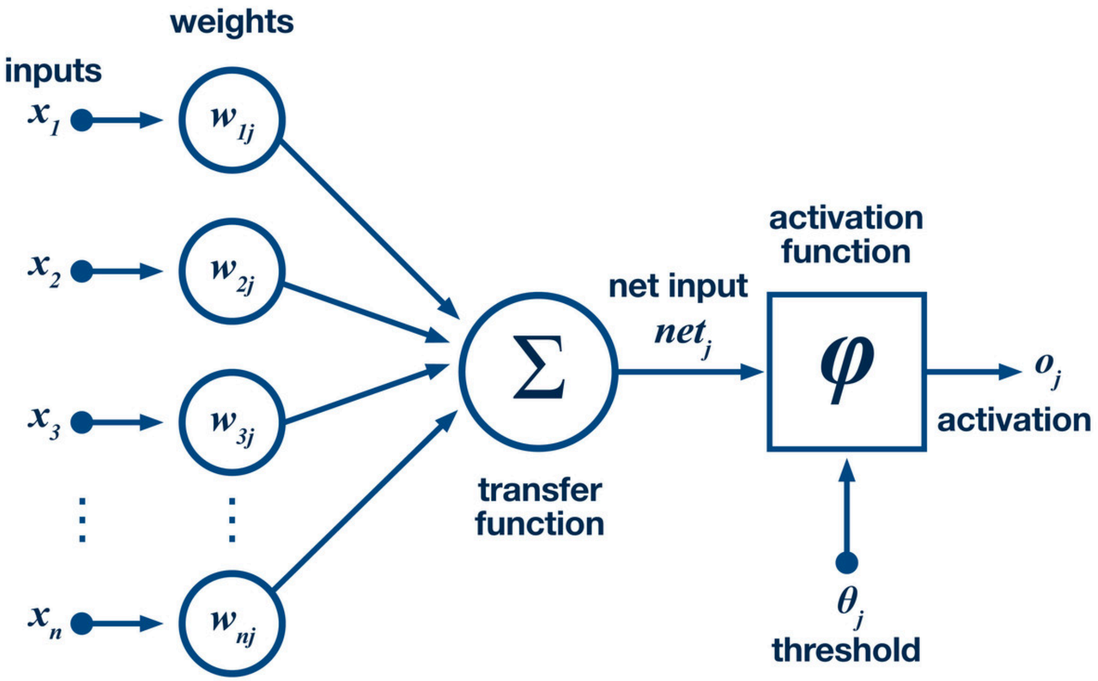
서포트 벡터 머신


class A sample
class B sample
Optimal
Hyperplane
W.X+b=0
X1
X2
Margin
2
||W||
Support Vector
H1
H2
Hyperplane

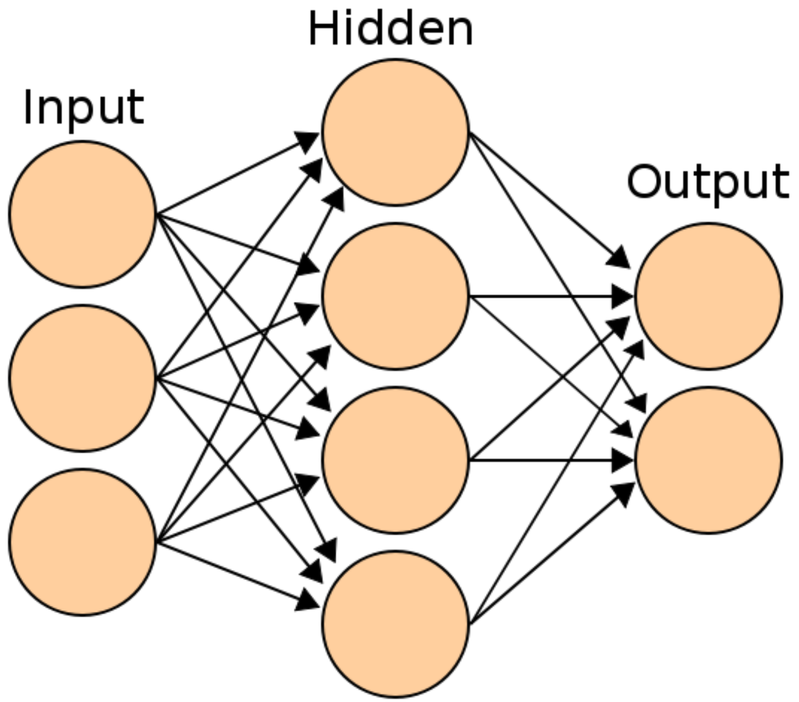
서포트 벡터 머신

- 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

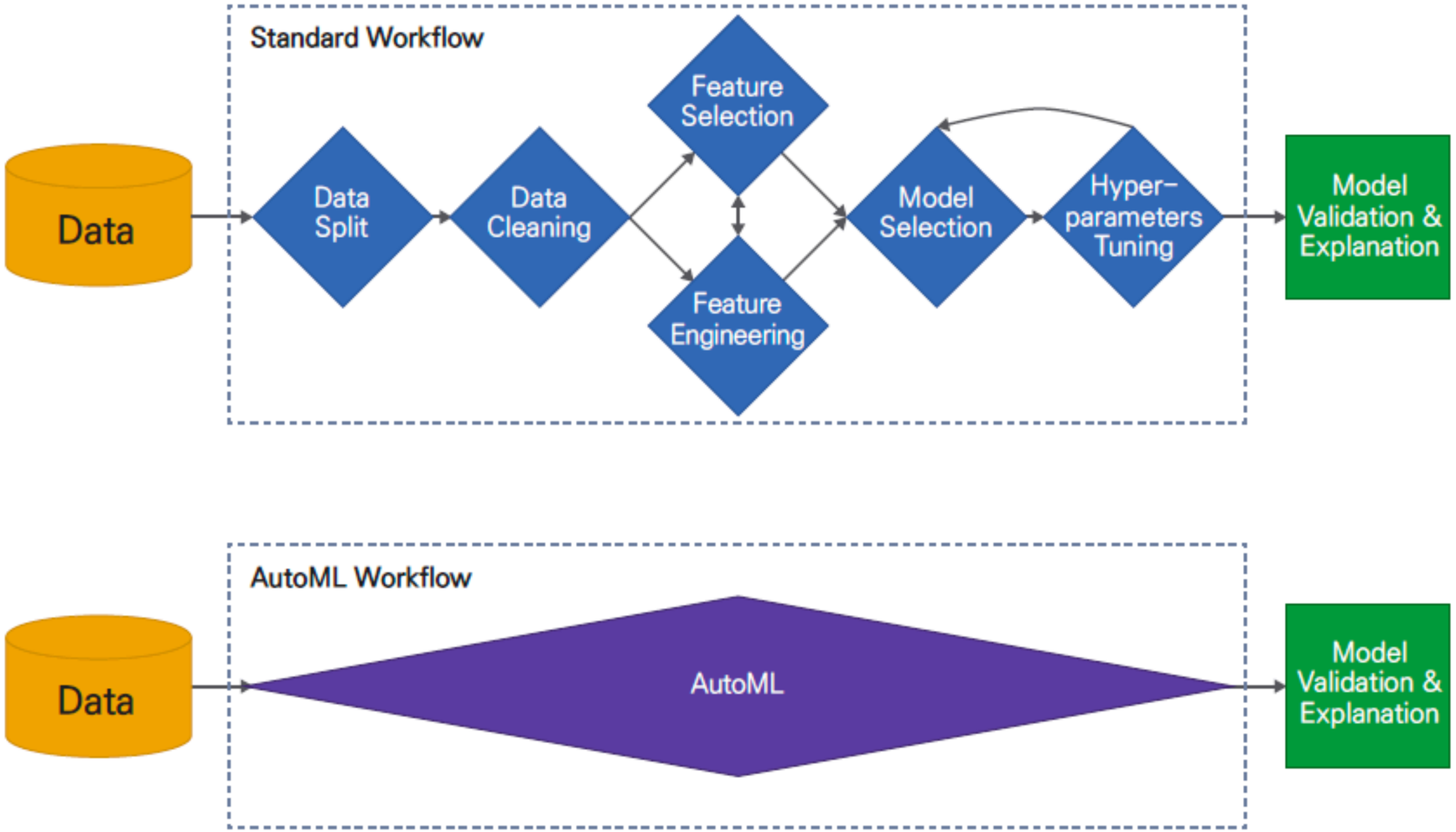
- 인공 신경 : Vector-to-Scalar Nonlinear function
- 레이어 : Vector-to-Vector Nonlinear function (입력을 공유하는 인공신경의 집합)
- 인공신경망 : Vector-to-Vector Nonlinear function (레이어들의 집합)

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.
- 최적화 방법론 : CASH, FMS

CASH

- CASH (Combined Algorithm Selection and Hyperparameter optimization)
- 하이퍼파라미터 최적화의 범위:
  CASH는 주로 알고리즘 선택과 그 알고리즘의 하이퍼파라미터 최적화에 중점
- 예시: 랜덤 포레스트와 SVM 중 어떤 알고리즘이 더 나은지 선택하고, 선택된 알고리즘의 최적의 하이퍼파라미터(예: 트리의 수, 커널 파라미터 등)을 탐색

FMS

- FMS (Full Model Selection)

- 하이퍼파라미터 최적화의 범위:
  FMS는 하이퍼파라미터 최적화뿐만 아니라 데이터 전처리, 피처 선택, 알고리즘 선택 및 하이퍼파라미터 최적화까지 포함한 전체 모델링 프로세스를 다룸

- 중점: 전체 머신러닝 파이프라인의 각 단계에서 최적의 방법을 선택하고, 이들을 조합하여 최적의 파이프라인을 구성하는 것이 목표

- 예시: 데이터 전처리 단계에서 결측치 처리 방법(예: 평균 대체, 중앙값 대체 등), 피처 선택 방법(예: L1 규제, 피처 중요도 등), 모델 선택(예: SVM, 랜덤 포레스트 등), 그리고 각 모델의 하이퍼파라미터 최적화를 모두 포함하여 최적의 조합을 탐색

HPO 문제 정의

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

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

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

Challenges in HPO

- 함수 평가가 매우 비용이 많이 들 수 있음.
  → 하이퍼파라미터가 다른 여러 함수에 대한 평가가 수반되어야 함.

- 탐색해야 할 하이퍼파라미터 조합이 존재하는 공간이 매우 복잡함.
  → 공간은 연속적, 범주형 및 조건부 하이퍼파라미터의 혼합으로 구성되며 고차원

- 대부분의 경우 직접적으로 하이퍼파라미터에 대한 손실 함수의 기울기 사용할 수 없음

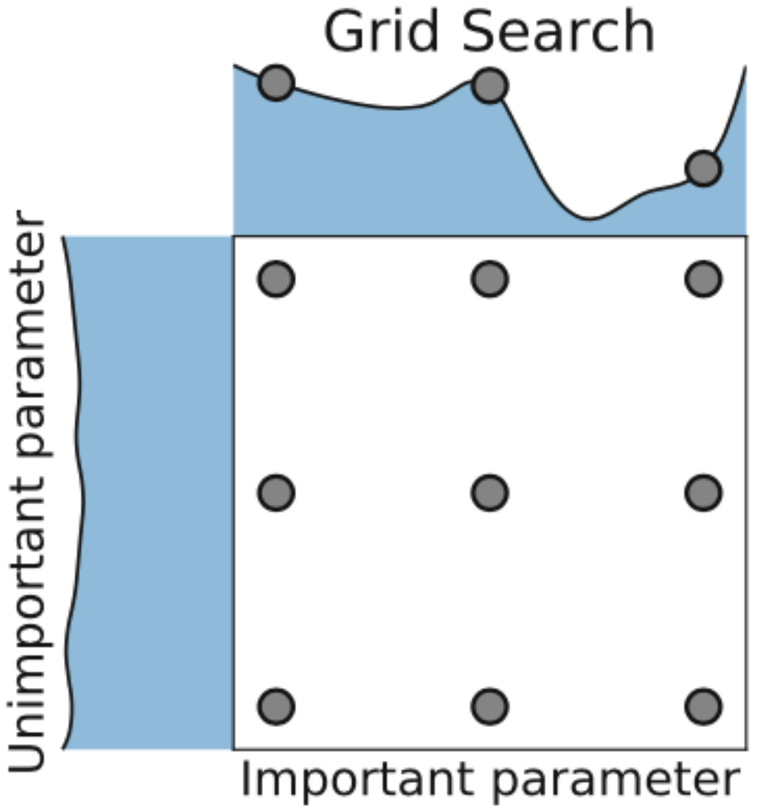
- 제한적인 훈련 데이터셋으로 일반화 성능에 대한 직접적 최적화가 어려움
  →일반화 성능↑ ~ 데이터셋의 크기↑ ~ 함수평가를 위한 비용↑

하이퍼 파라미터 탐색 방법

- Grid Search
- Random Search
- Population-based
- Bayesian Optimization

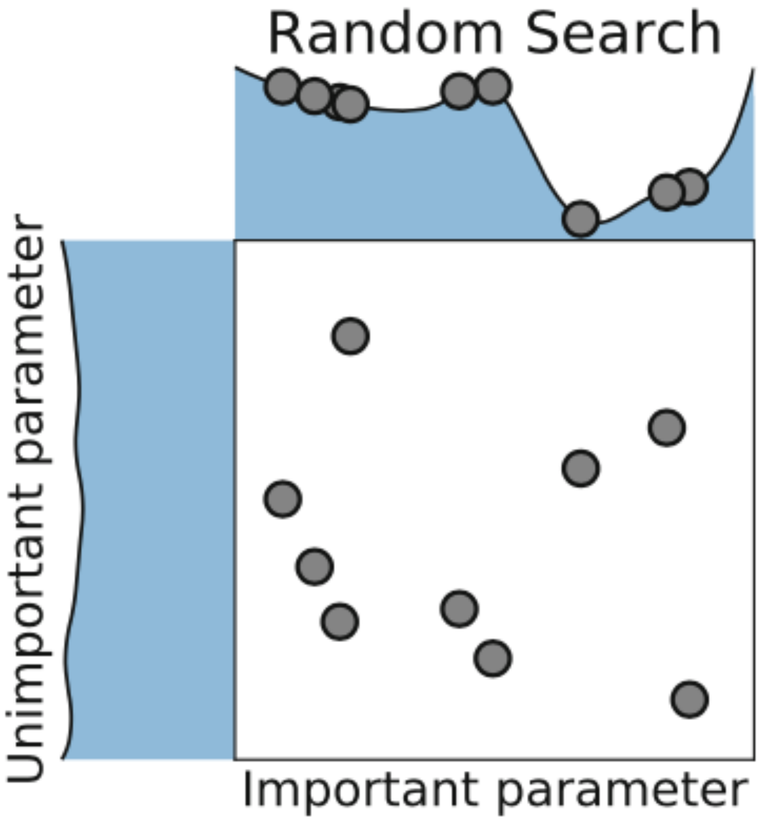
Grid search

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



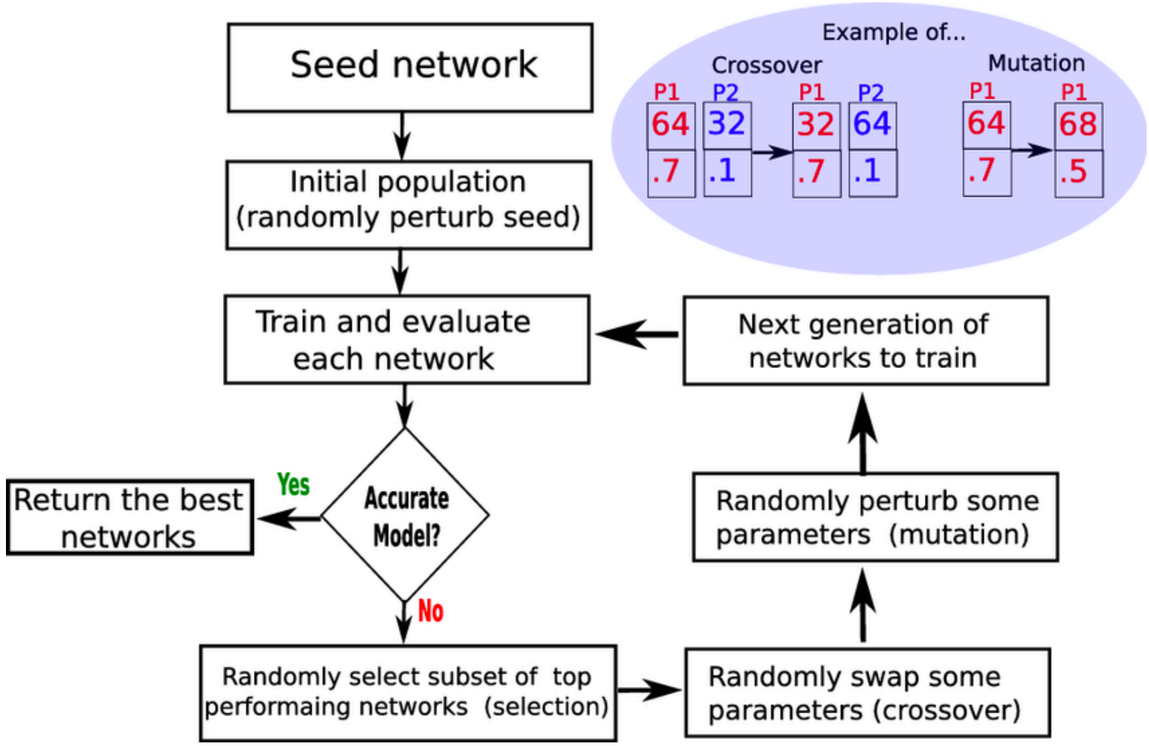
Random search

- 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

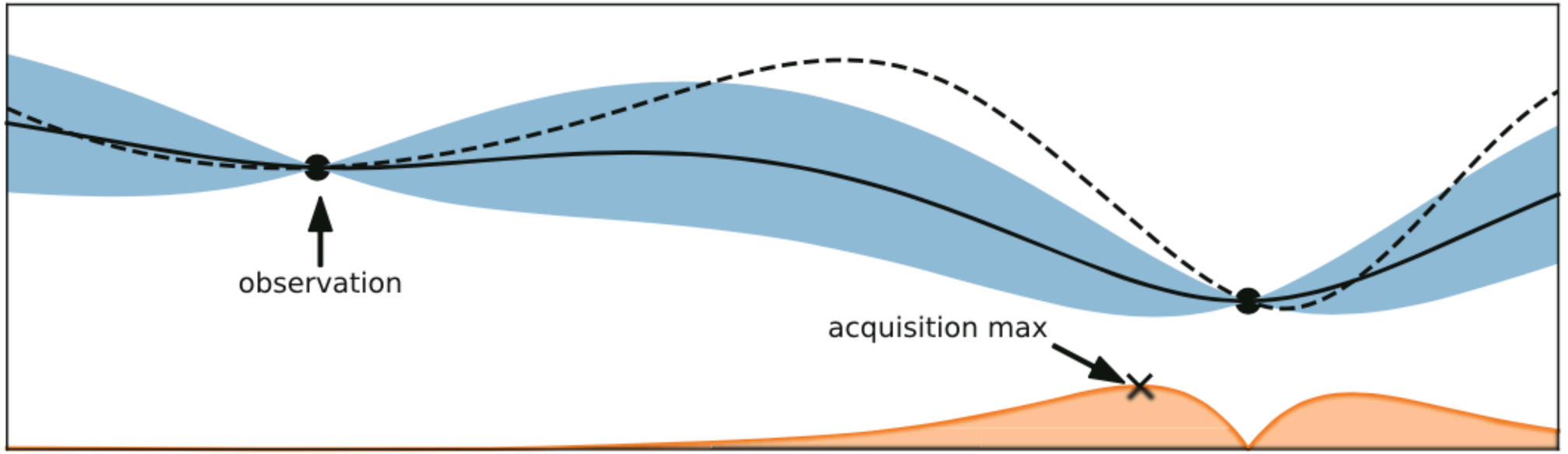
Bayesian Optimization

- 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.

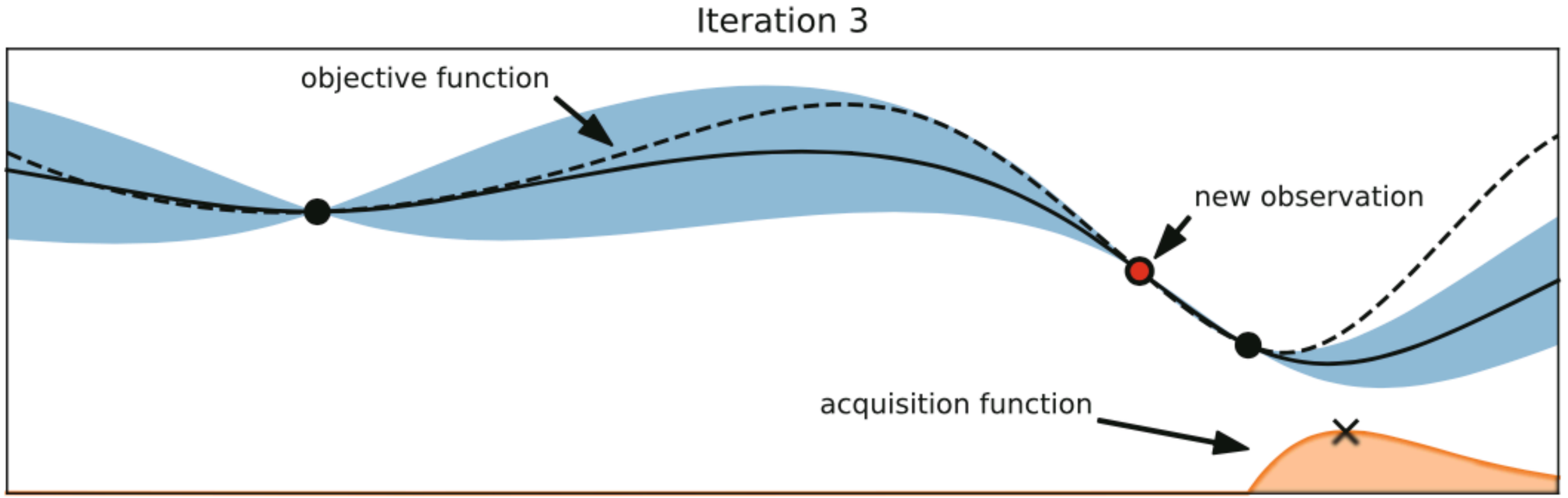
- Acquisition function:
  과거의 결과로부터 탐색 해야 할 파라미터를 결정해주는 함수.

베이지안 최적화 예시

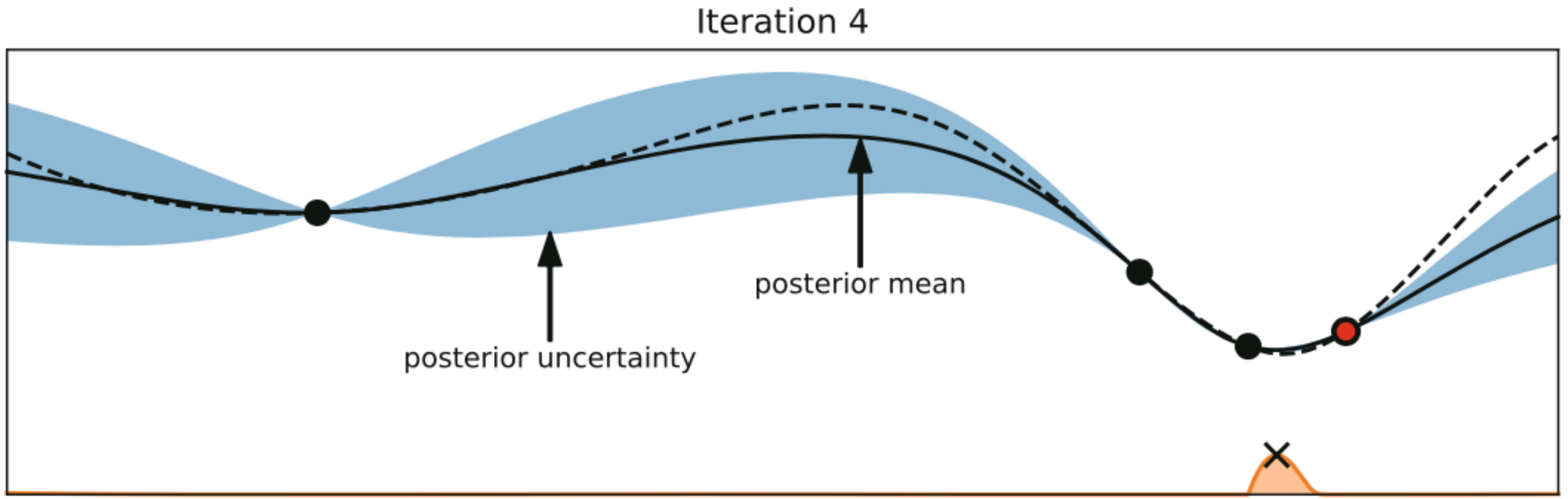


Iteration 3

베이지안 최적화 예시


Iteration 3
objective function
new observation
acquisition function

베이지안 최적화 예시

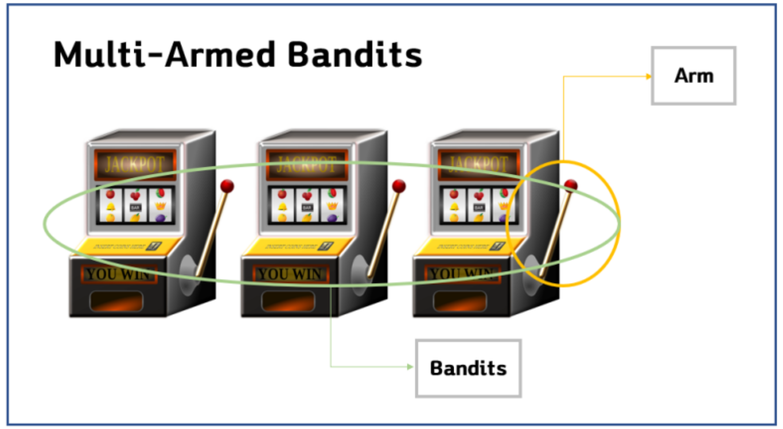


Multi-fidelity optimization

- Fidelity (충실도) : High-fidelity (Hi-Fi) & Low-fidelity (Lo-Fi)
- Recall : 함수 평가가 매우 비용이 많이 들 수 있음.
  → 하이퍼파라미터가 다른 여러 함수에 대한 평가가 수반되어야 함.
- 더 어려운 태스크, 더 복잡한 모델 구조, 더 많은 데이터 → HPO에서의 주요 문제점
- 직관적인 접근은? 간단한 태스크, 간단한 데이터에 대해서 탐색하고 실제 문제로 확장
- 실제 loss function에 대한 Low-fidelity approximation.

Multi-fidelity for HPO

- Learning Curve-Based Prediction for Early Stopping:
  - 탐색 과정에서 학습 곡선을 평가하여 조기종료
    → 리소스를 효율적으로 활용.
- Bandit-Based Algorithm Selection Methods:
  - Multi-armed bandits in 강화학습 - 한정된 자원으로 가장 보상이 높은 것을 탐색 (Exporation & Exploitation)
  - Low fidelity approximation을 바탕으로 후보를 추출
    → 한정된 자원? 학습반복횟수, 데이터량, 파라미터 수 등
  - e.g., Successive Halving, Hyperband.



Successive halving

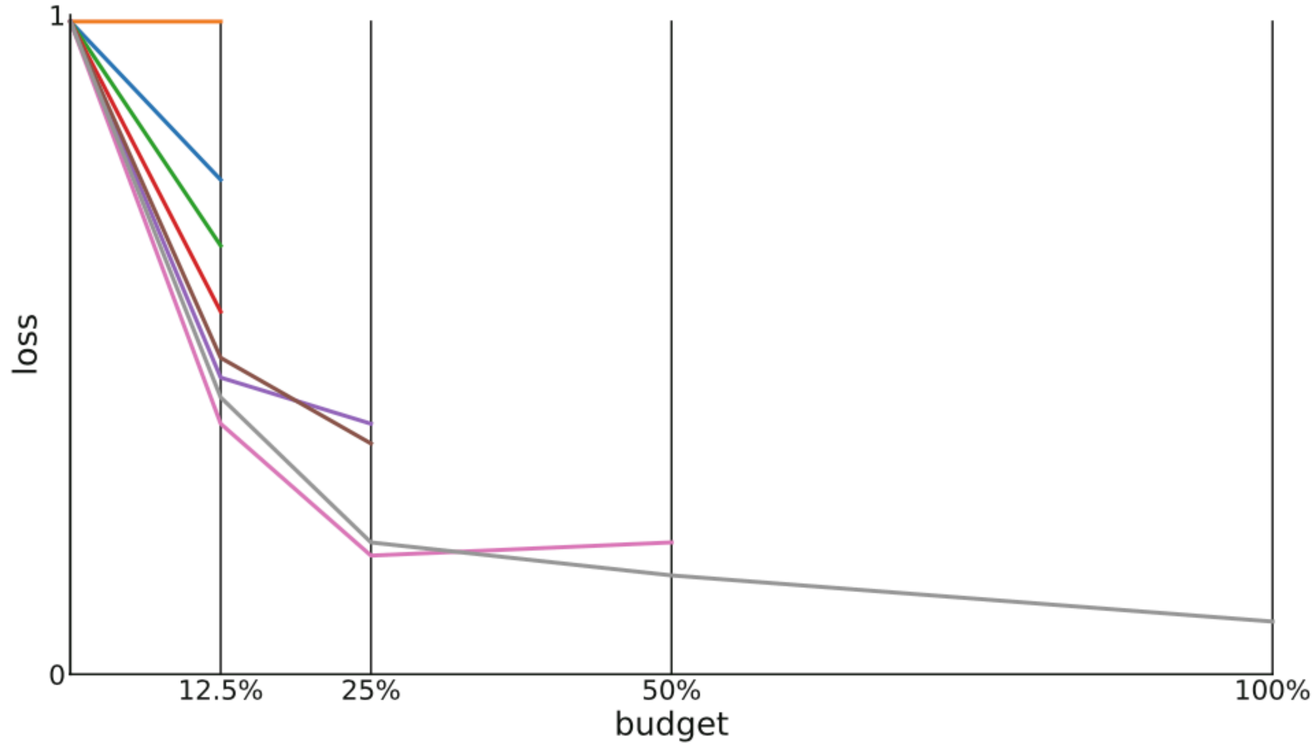


Fig. 1.3 Illustration of successive halving for eight algorithms/configurations. After evaluating all algorithms on $\frac{1}{8}$ of the total budget, half of them are dropped and the budget given to the remaining algorithms is doubled

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
- 이전 학습과정에 대한 정보(메타데이터) 들을 활용함으로써 모델 선택, 하이퍼파라미터, 특징 추출등의 과정을 개선할 수 있음.

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

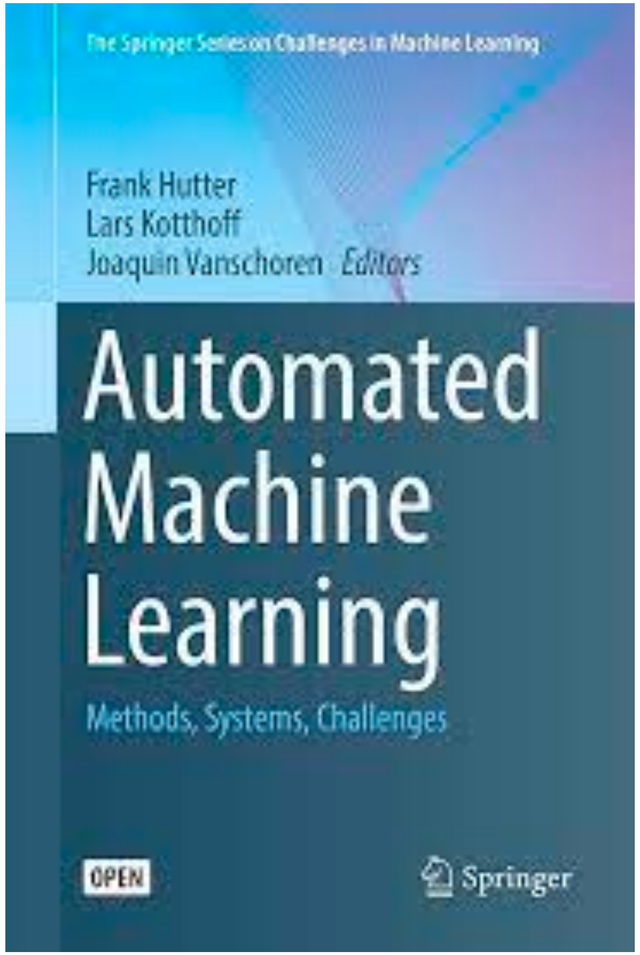
- 인공신경망의 구조는 성능에 매우 큰 영향을 미치며, 일반적으로 연구자의 판단에 근거하여 설계.
- 커널 크기, 레이어의 종류, 레이어의 연결 등 인공신경망의 구조는 굉장히 다양한 조합을 갖을 수 있음.
- 이 모두를 하이퍼 파라미터로 보았을때 인공신경망은 일반적인 머신러닝 모델에 비해 많은 하이퍼 파라미터를 가지고있음.
- NAS는 이러한 네트워크의 구조 설계 과정을 자동화하여 최적의 구조를 찾는 방법임.

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/autoweka/
- Scala
  - TransmogrifyAI : https://transmogrif.ai/

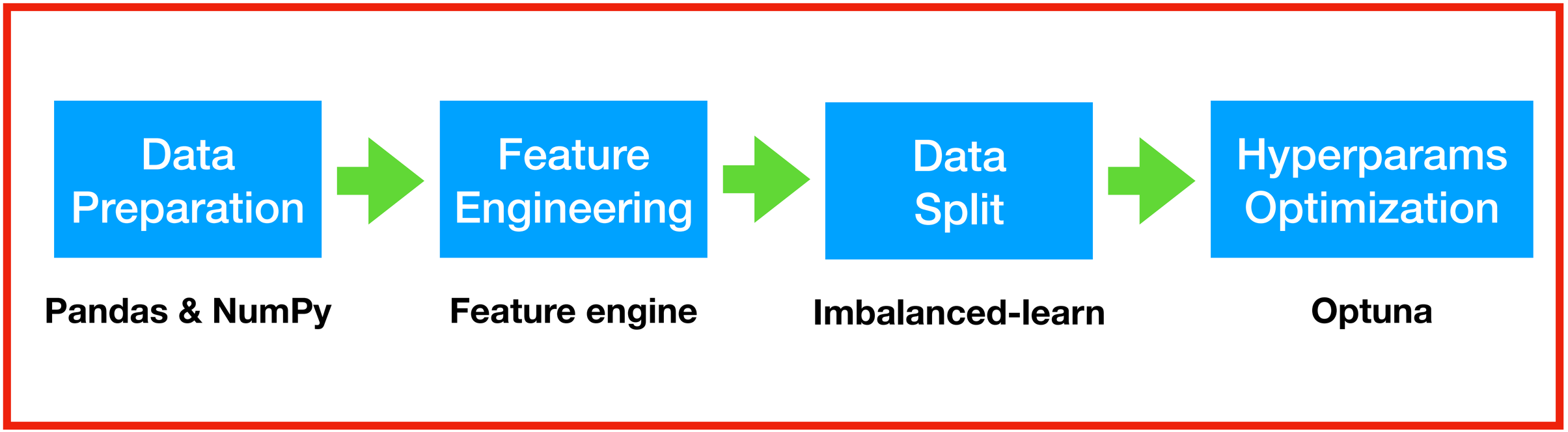
Automated Machine Learning

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



Chapter 3.
Hands on AutoML

실습 내용 소개

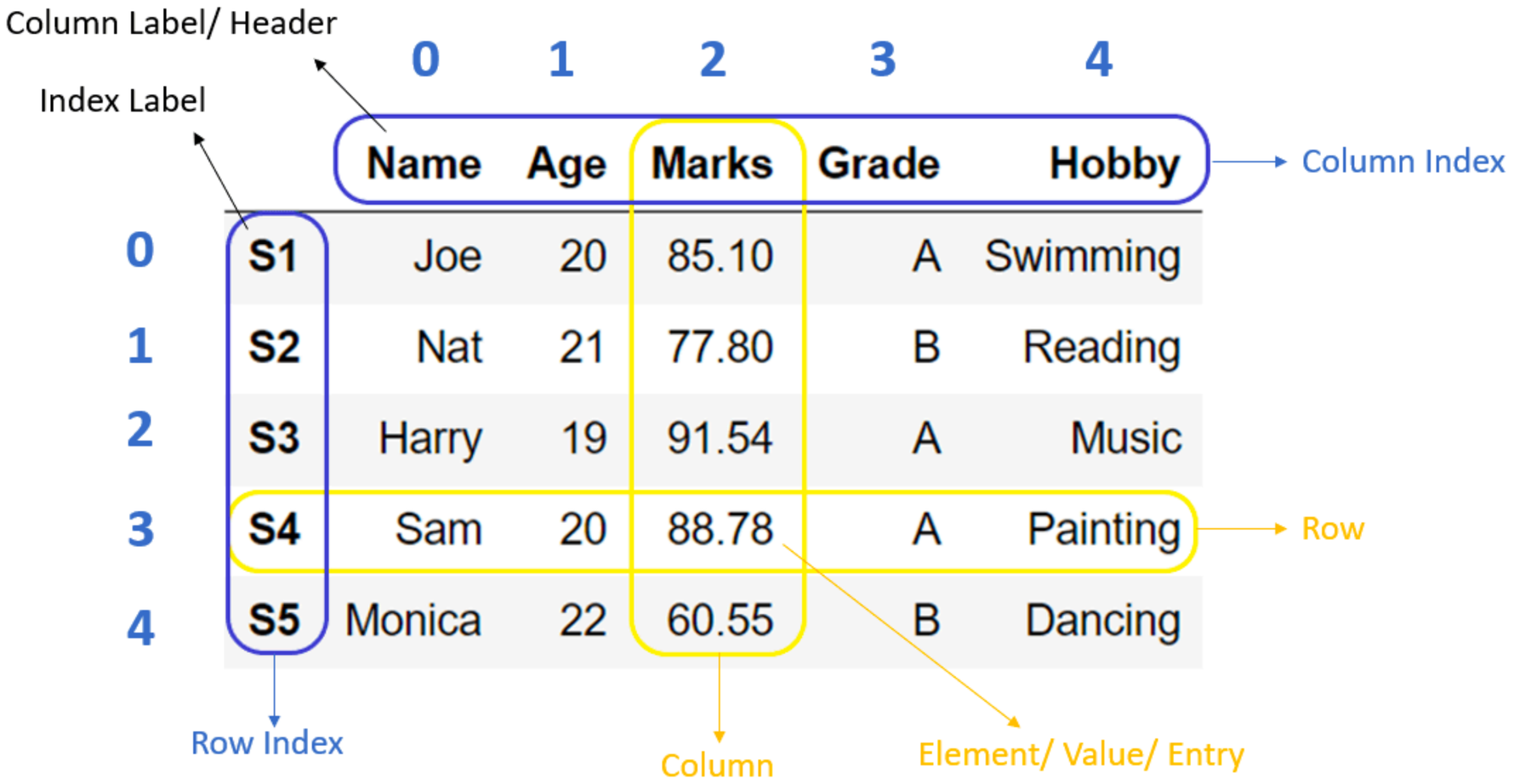


머신러닝 : Scikit Learn, PyCaret
딥러닝 : AutoKeras

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



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.