

프로젝트 기반 데이터 과학자 양성과정(Data Science) Machine Learning 및 분석실습

1주차 Machine Learning 개요

강사 : 최영진

목차

1. 머신러닝의 개요
2. 데이터 구조
3. 데이터셋 구성을 통한 검증 방법
4. 통계 리뷰

Course Information

❖ Textbook (Required)



Scikit-learn으로 머신러닝 마스터 2/e

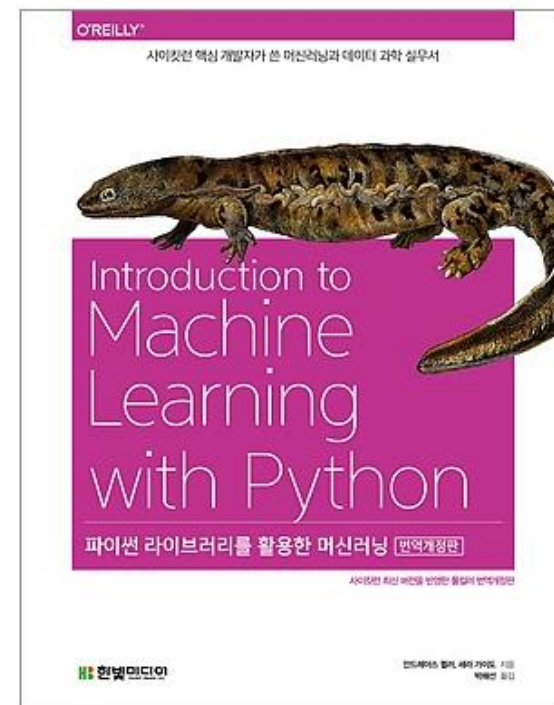
<https://github.com/PacktPublishing/Mastering-Machine-Learning-with-scikit-learn-Second-Edition>

파이썬을 이용한 머신러닝, 딥러닝 실전개발 입문

<https://github.com/suites/actual-deeplearning>

파이썬 라이브러리를 활용한 머신러닝

https://github.com/rickiepark/introduction_to_ml_with_python



Course Information

❖ Topics To Be Covered

- 데이터 마이닝 및 머신러닝 개요
- 데이터 전처리 및 구조
- 회귀분석
- 지도학습
 - K-NN(k-최근접 이웃 알고리즘)
 - 나이브베이즈
 - 서포트벡터머신
 - 의사결정트리
 - 랜덤포레스트
 - Xgboost
- 비지도학습
 - 군집화
 - 주성분 분석

머신러닝 개요

❖ 데이터마이닝(Datamining)이란?

- 대용량 데이터에 존재하는 데이터 간의 관계, 패턴, 규칙 등을 찾아내고 모형화해서 기업의 경쟁력 확보를 위한 의사결정을 돕는 일련의 과정

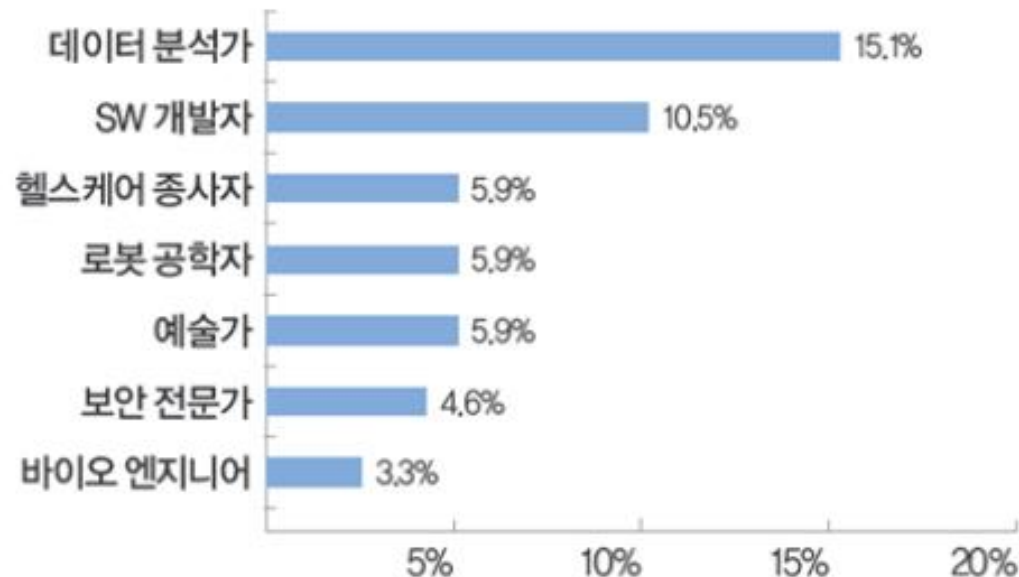


머신러닝 개요

❖ 데이터 사이언티스트

- 국내 각 분야에서 종사하고 있는 전문가들이 뽑은 10년 뒤 유망한 직업1위
- 보유 데이터 규모가 급증하면서 어떤 데이터를 어떻게 활용할 것인지 해결책을 제시할 수 있는 데이터 전문가에 대한 수요 증가

◆ 10년 후 유망한 직업(복수응답)

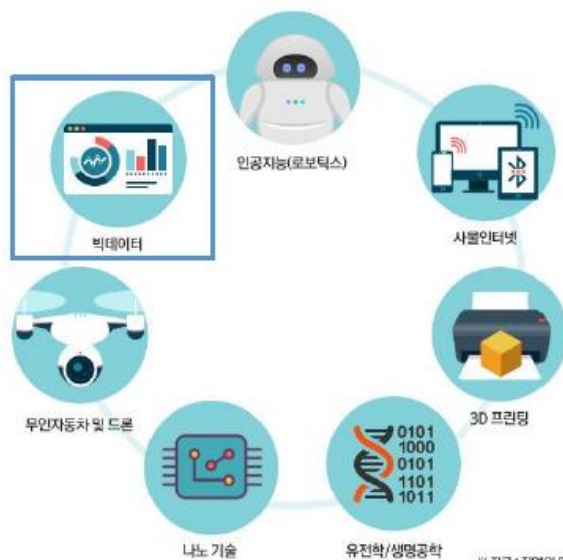


머신러닝 개요

❖ 데이터 사이언티스트

- 세계 빅데이터 시장의 규모는 연평균 성장률 17%를 보이며, 2020년 약 60조원 예상
- 4차 산업혁명을 이끄는 핵심 기술의 하나로, 특히 IoT, 클라우드, 센서기술 등의 결합

4차 산업혁명과 함께하는 미래의 대표기술



※ 자료 : 직업의 미래(The Future of Jobs) 보고서

〈세계 빅데이터 시장규모 및 전망〉

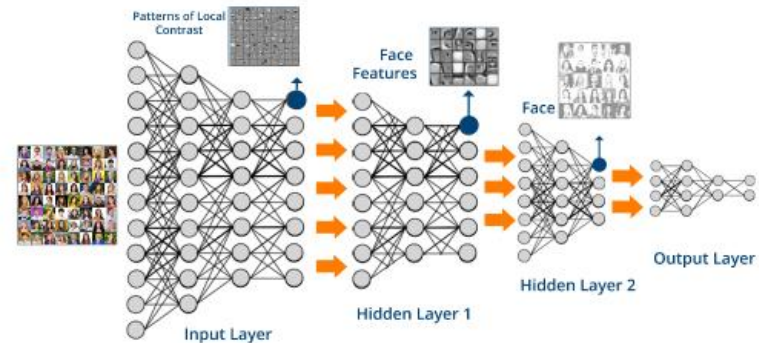
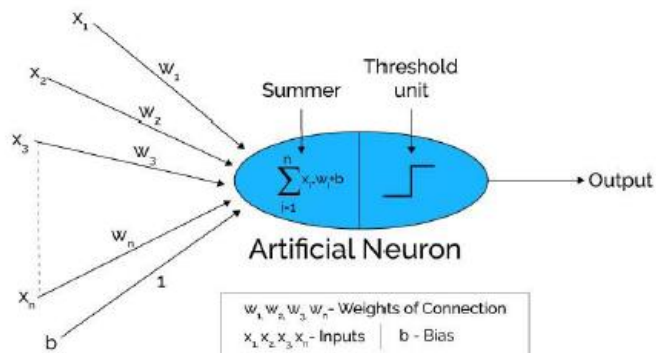


(단위 : 십억 달러)

구분	2013	2014	2015	2016	2017	2018	2019	2020
시장규모	19.6	18.3	22.6	27.3	33.5	40.8	49	57.3
성장률	2013~2020년 CAGR 17%							

※ 출처 : Forecast of Big Data market size, based on revenue, from 2011 to 2026 (in billion U.S. dollars).
The Statistics Portal, 2016

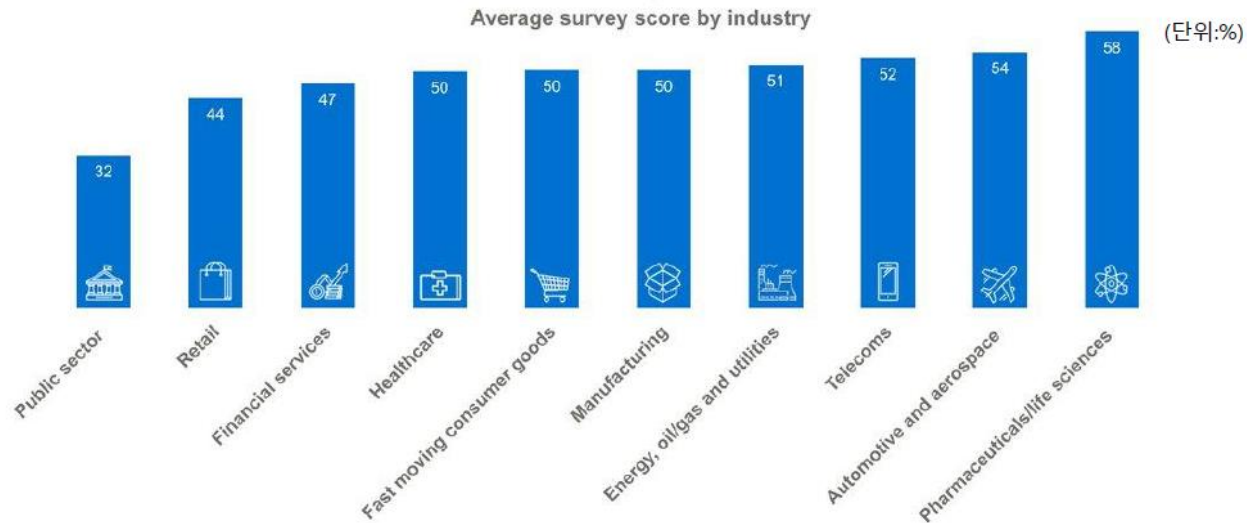
❖ 데이터 사이언티스트가 되기 위해서는?



8

머신러닝 개요

❖ 산업별 인공지능 기술 성숙도



시간 흐름에 따른 관심도 변화 (?)



<https://trends.google.co.kr/trends/?geo=KR>

머신러닝 개요

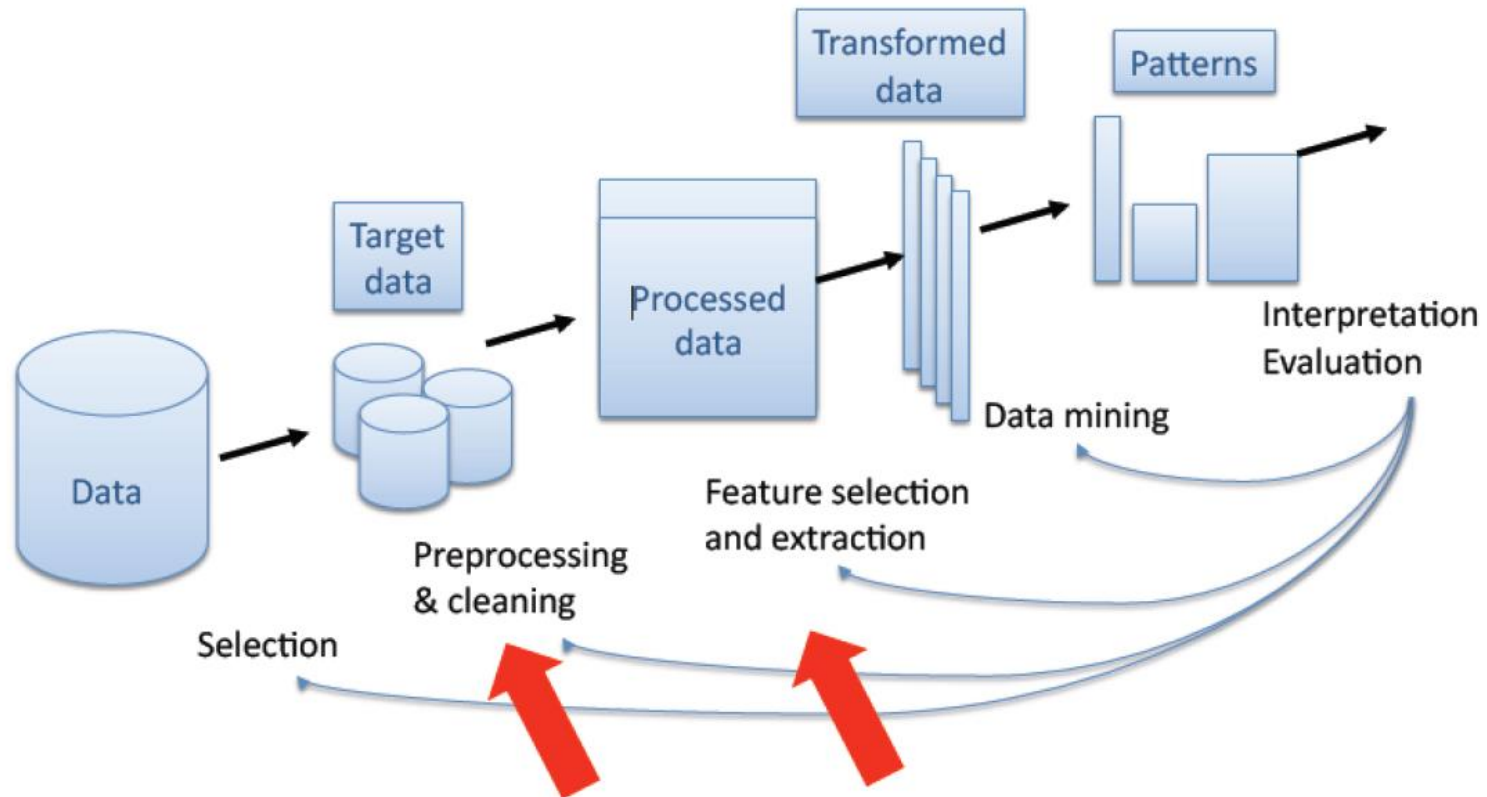
❖ 데이터 마이닝의 관련분야

- 기계학습 (Machin Learning) – 자동적인 학습 기법 설계, 구현
- 패턴인식 (Pattern Recognition) – 문자인식, 이미지분류
- 통계학 (Statistics) – 다변량 (판별분석, 주성분분석, 군집분석), 회귀분석 등

머신러닝 개요

❖ 데이터 마이닝의 절차

- 데이터 선정 - 정제 - 변형 - 데이터마이닝 - 해석



머신러닝 개요

❖ 데이터 마이닝의 분석 프레임

▪ 데이터 수집

- 결제 바로 전, 유저의 행동 패턴을 기술할 수 있는 로그 항목을 수집 (문서 오픈, 편집 등)

▪ 데이터 추출

- Extraction, Preprocessing (SQL 필터, 조인 등)

▪ 데이터 전처리

- Feature Engineering (분포 변환, PCA, 결측치 및 이상치 처리 등)

▪ 모델 구축 및 파라미터 설정

- Classification Models (Logistic Regression, Random Forest, etc)
- Cross Validation, Grid Search, Pipeline

▪ 모델 평가

- Precision, Recall, F1-score

머신러닝 개요

❖ 통계분석 VS 데이터마이닝

■ 전통적 통계분석

- 대상집단이 있으며, 모집단의 분포 혹은 모형 등 여러 가지 가정을 전제로 하게 되며 이 전제 조건하에서 분석을 실시
표본(Sample)의 관찰을 통해 모수(Population) 전체를 추론(Inference)하는 과정

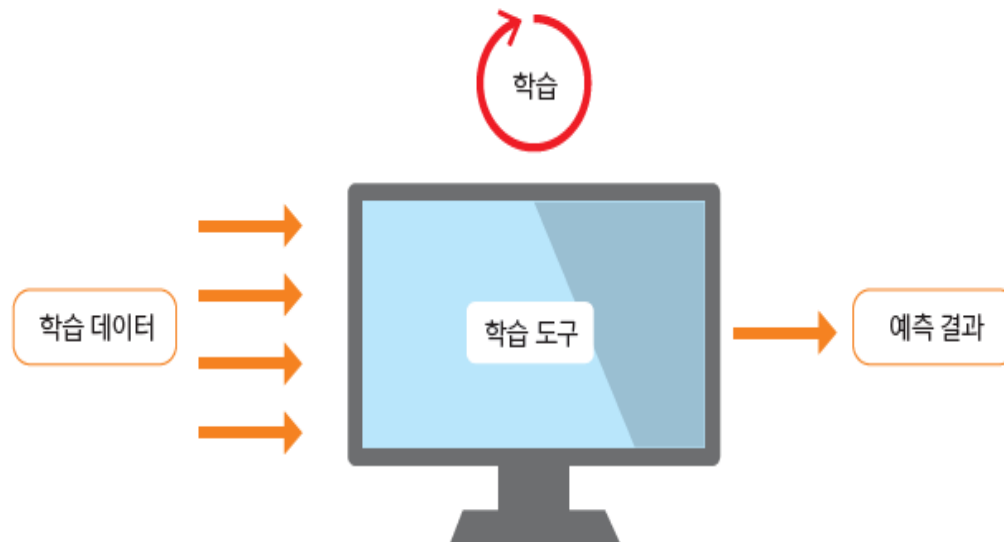
■ 데이터마이닝

- 표본조사/실험에서 필연적으로 수반되는 분포라든지 모형에 대한 전제조건이 필요하지 않음
모집단의 전체자료를 이용하여 필요한 정보/지식을 추출하는 과정
- 대용량 자료여야 한다는 전제조건 있음

머신러닝 개요

❖ 머신러닝(기계학습)이란?

- 머신러닝(기계 학습)은 여러분에게 주어진 문제, 과제, 환경에 따라 컴퓨터 스스로 문제를 해결하려고 학습한 예측 결과를 활용하는 것
- 과거의 경험을 미래의 결정에 활용하는 소프트웨어를 디자인하고 연구하는 분야로 데이터로부터 학습하는 프로그램을 연구하는 것



머신러닝 개요

❖ 머신러닝이란?

Machine Learning  Machine Learned

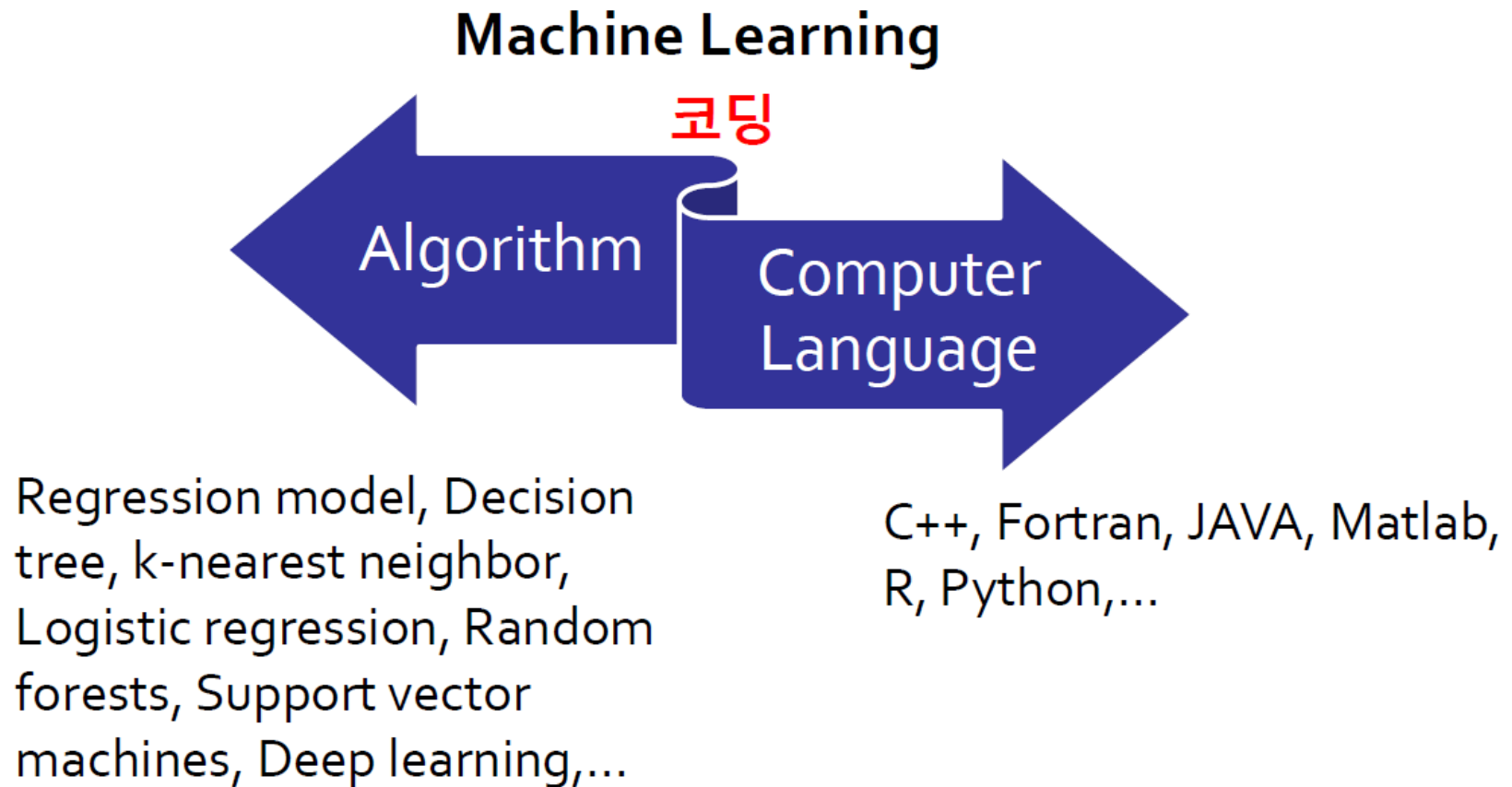
Machine Learned



= Artificial
Intelligence

머신러닝 개요

❖ 머신러닝이란?



머신러닝 개요

❖ 머신러닝 사용 예

- 스팸메일분류
- 주가예측
- 기계번역(Google 번역)
- 사기탐지
- 감성분석
- 음성분석
- 얼굴인식
- 텍스트요약등

머신러닝 개요

❖ 머신러닝 용어

▪ 특징(feature)

- 각각의 아이টে을 설명하는데 사용하는 구분가능한 특성 또는 특징의 개수

▪ 데이터(data)

- 문서, 사진, 음성, 동영상, 데이터베이스 등

▪ 특징벡터(feature vector)

- 어떤 대상을 표현하는 특징으로 이루어진 n 차원의 벡터

▪ 특징추출(feature extraction)

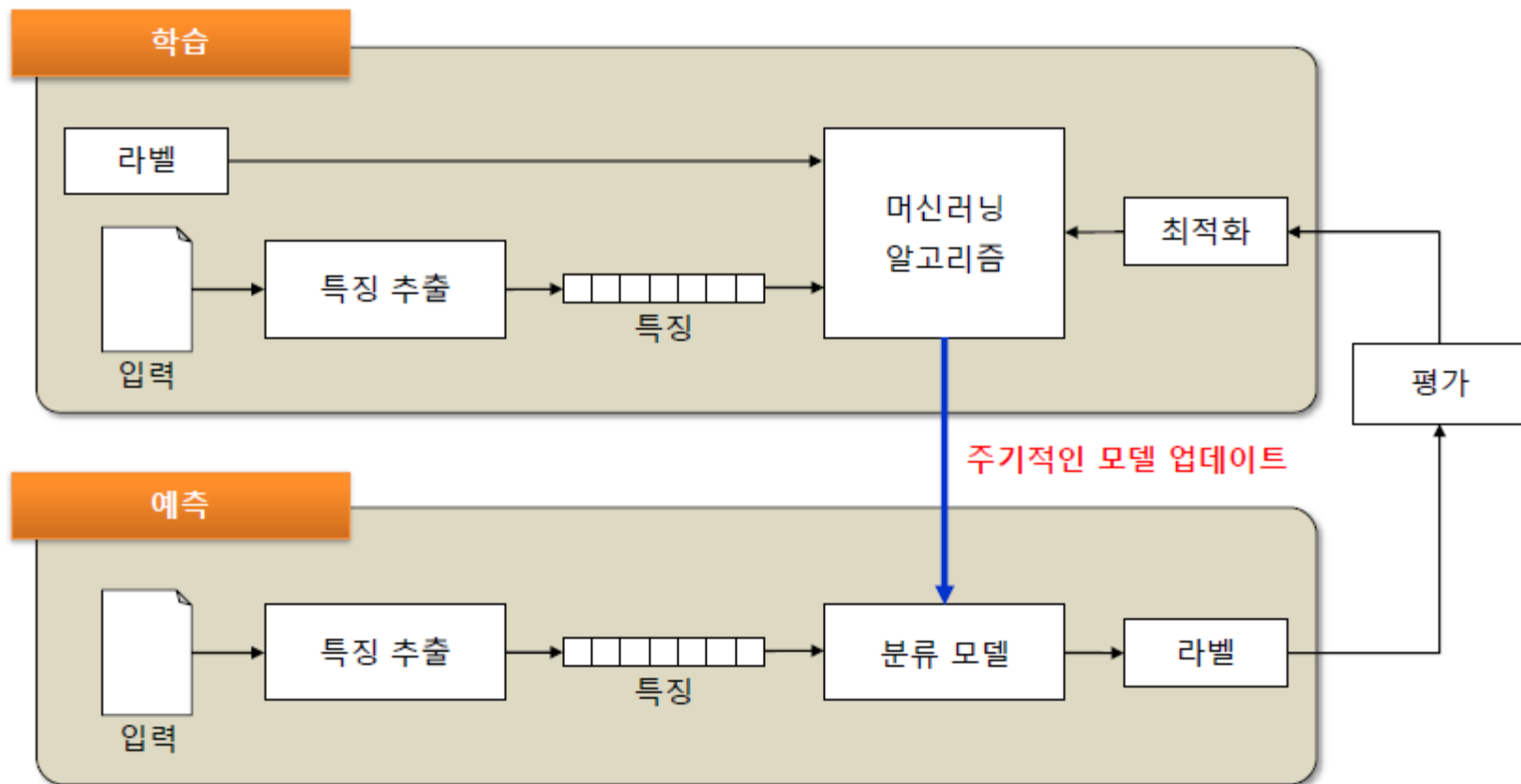
- 특징 벡터의 준비
- 차원 감소기법을 사용하기도 함

▪ 학습셋(training set)

- 학습에 사용하는 데이터셋

머신러닝 개요

❖ 머신러닝 흐름도



데이터 구조

❖ 머신러닝의 유형

▪ 지도학습(Supervised learning)

- 레이블이 붙어있는 문제의 정답을 컴퓨터에 입력해 머신러닝 모델을 학습

▪ 비지도학습(Unsupervised learning)

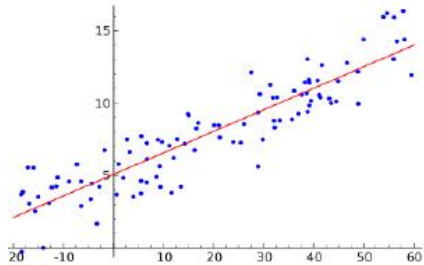
- 데이터의 특징과 종속 변수(정답)를 조합해 학습하는 방법
- 데이터 포인트를 유사한 패턴을 가진 그룹별로 나누려하는 것

▪ 강화 학습(Reinforcement learning)

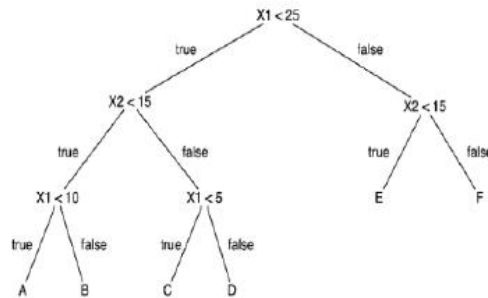
- 계속된 행동으로 얻은 보상으로부터 올바른 행동을 학습
- 지도학습의 한쪽 끝 영역 근처에 위치하고, 레이블이 달린 입력과 출력쌍으로 학습하지 않음

데이터 구조

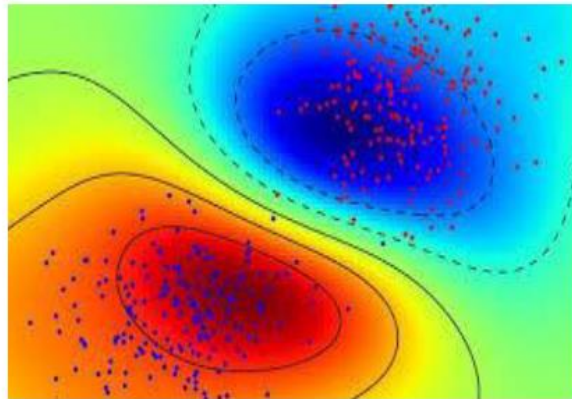
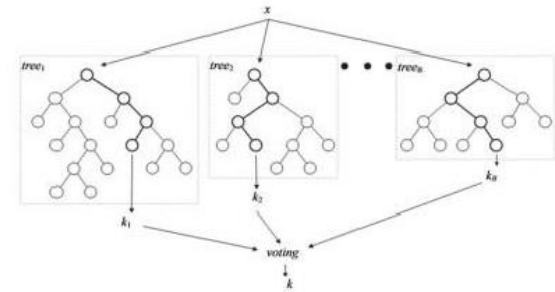
❖ 머신러닝의 유형



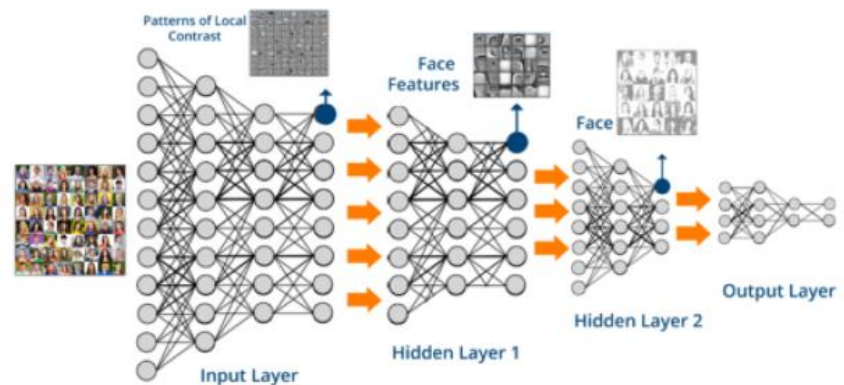
회귀



의사결정트리



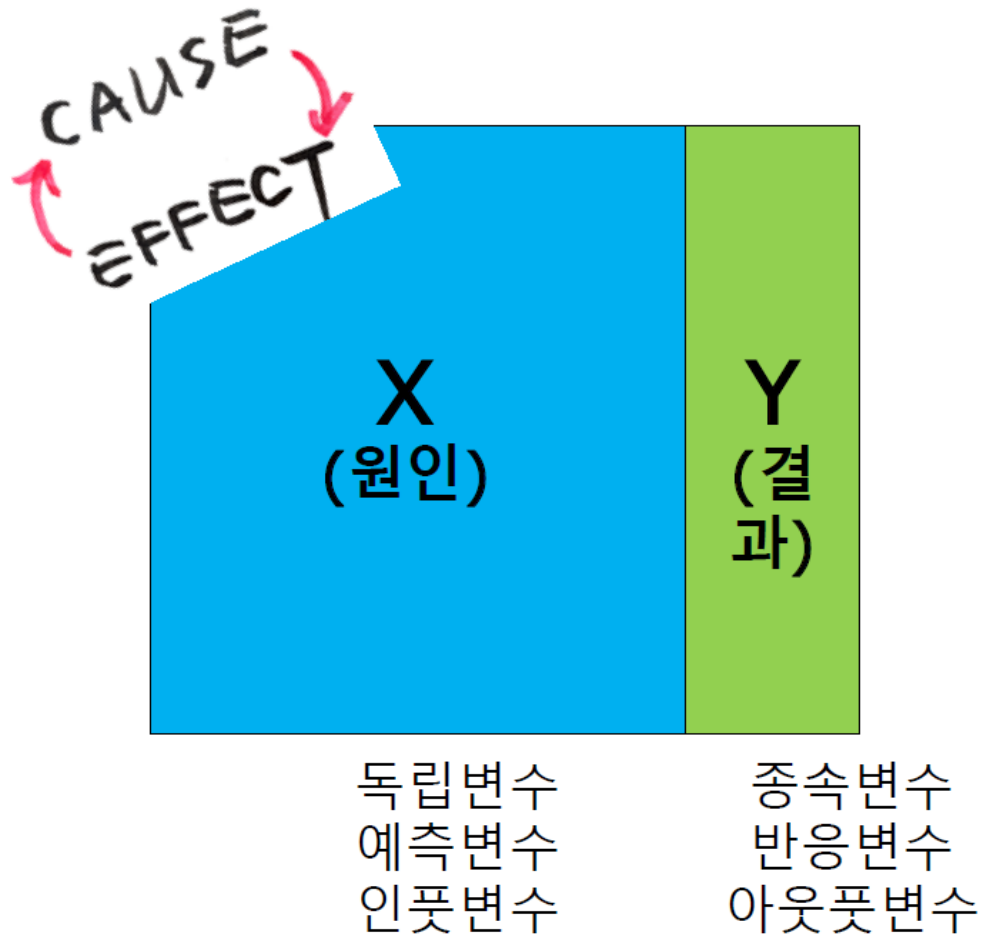
SVM



인공지능

데이터 구조

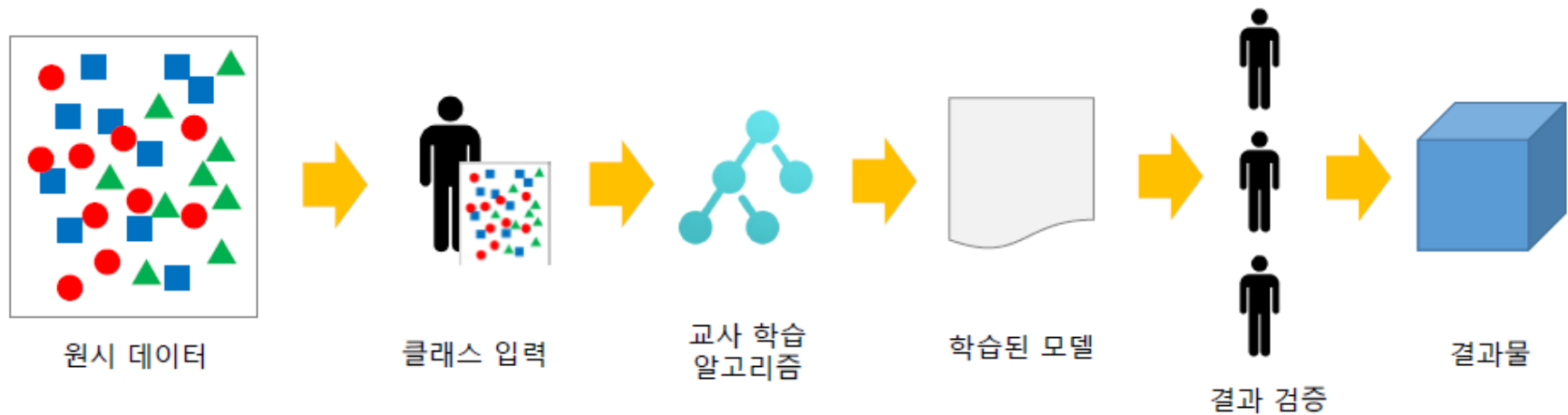
❖ 데이터 구조



데이터 구조

❖ 지도학습(Supervised learning)

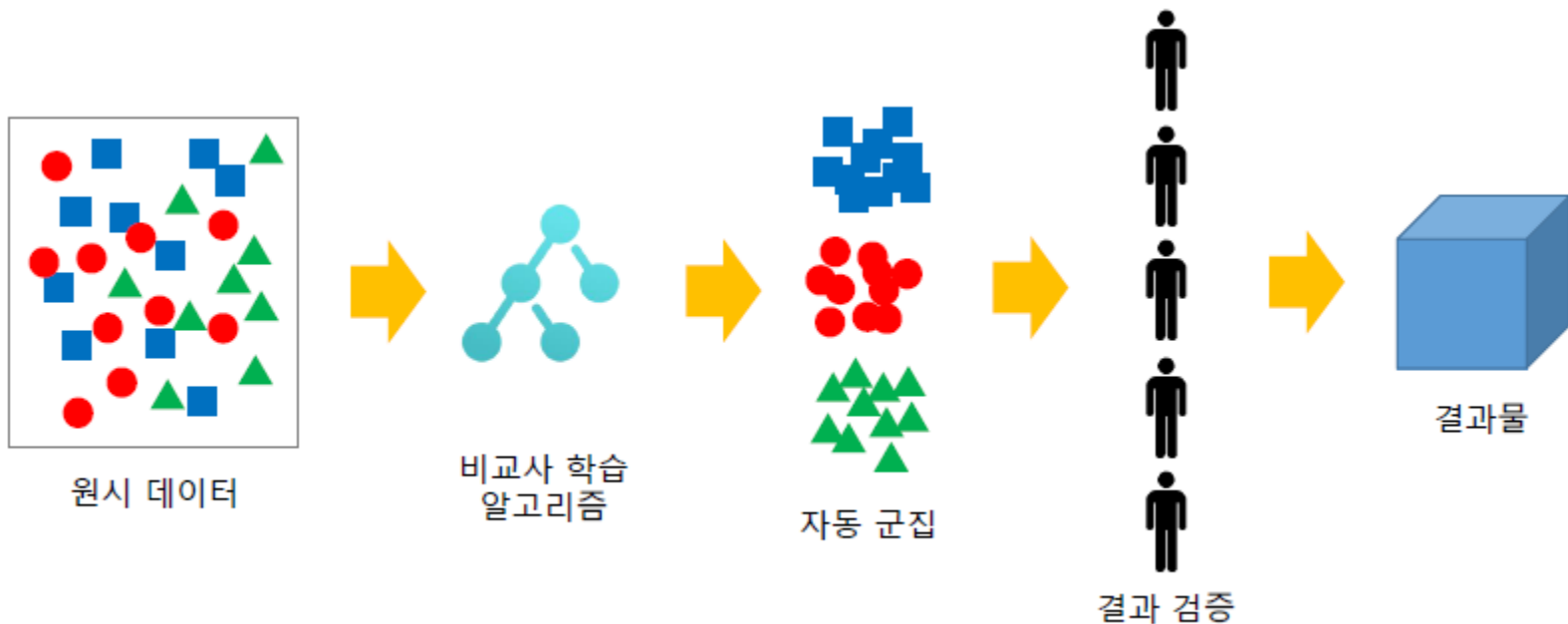
- 학습데이터의 정확한 클래스가 알려져 있음
- 지도학습에는 크게 분류(classification)과 회귀(regression)



데이터 구조

❖ 비지도 학습(Unsupervised Learning)

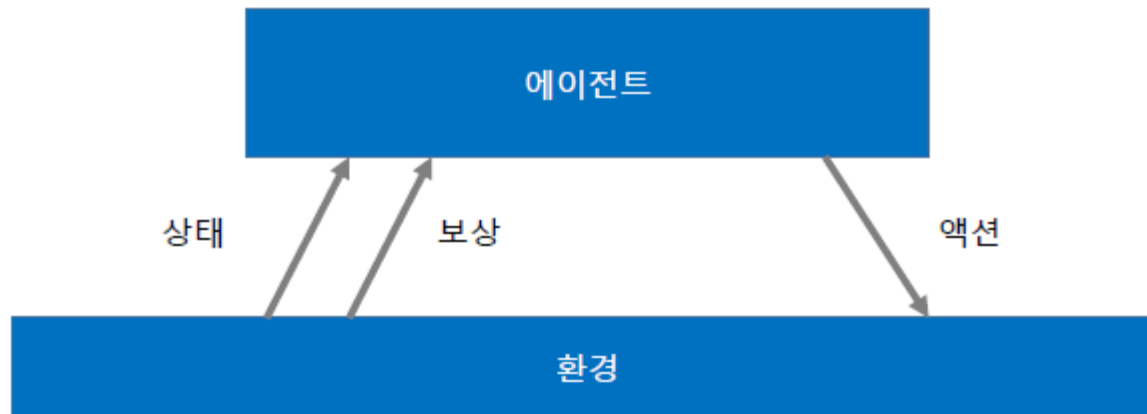
- 정답을 따로 알려주지 않고(label이 없음)
- 비슷한 데이터들을 군집화 하는 것
- 실무에서는 지도학습에서의 적절한 feature를 찾아내기 위한 전처리 방법으로 비지도 학습 사용



데이터 구조

❖ 강화학습(Reinforcement Learning)

- 기계 또는 소프트웨어 에이전트는 환경으로부터의 피드백을 기반으로 동작을 학습
- 한번에 모두 학습하거나 또는 시간이 지남에 따라 계속해서 적응 할 수 있음
- 알고리즘은 사람의 입력에 의해 지속적으로 훈련됨
- 자동적으로 최대한 정확도를 높이고 벽돌깨기가 대표적인 예



데이터 구조

❖ 머신러닝 적용 분야

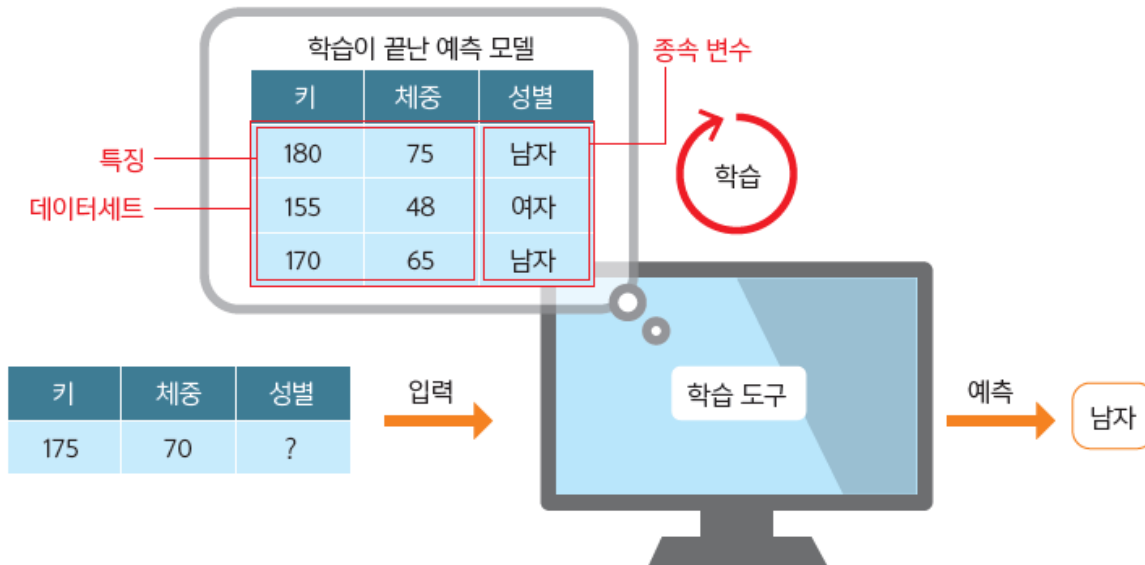
- 분류(classification): 데이터로부터 클래스를 예측
- 군집(clustering): 데이터로부터 의미 있는 그룹을 나눔
- 회귀(regression): 데이터분석을 통해 값을 예측

데이터 구조

❖ 지도학습(Supervised learning)

▪ 분류(Classification)

- 사전에 미리 정의된 집단 중 어디에 해당할 것인지를 예측, 분류기준이 미리 정하는 것
- 이진분류 / 다중클래스 분류
- 특징 feature 혹은 독립 변수 independent variable
- 정답 을 나타내는 데이터를 레이블 label 혹은 종속 변수 dependent variable

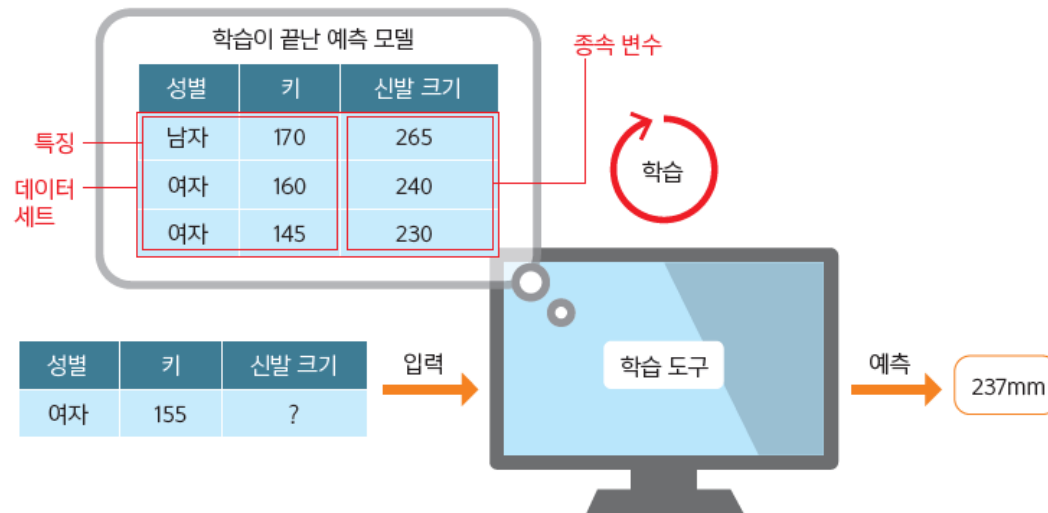


데이터 구조

❖ 지도학습(Supervised learning)

▪ 회귀(regression)

- 회귀 문제는 숫자 값의 크고 작음에 의미를 부여해 예측하는 방법
- 회귀는 하나의 변수 평균값(예: 출력)과 다른 변수의 해당 값(예: 시간 및 비용) 간의 관계를 측정
- 회귀분석은 변수 간의 관계를 추정하기 위한 통계적인 방법
- 회귀는 학습데이터를 이용해 결과값을 예측하는 것
- 가장 대중적인 회귀로는 로지스틱 회귀(binary)가 있음



데이터 구조

❖ 지도학습(Supervised learning)

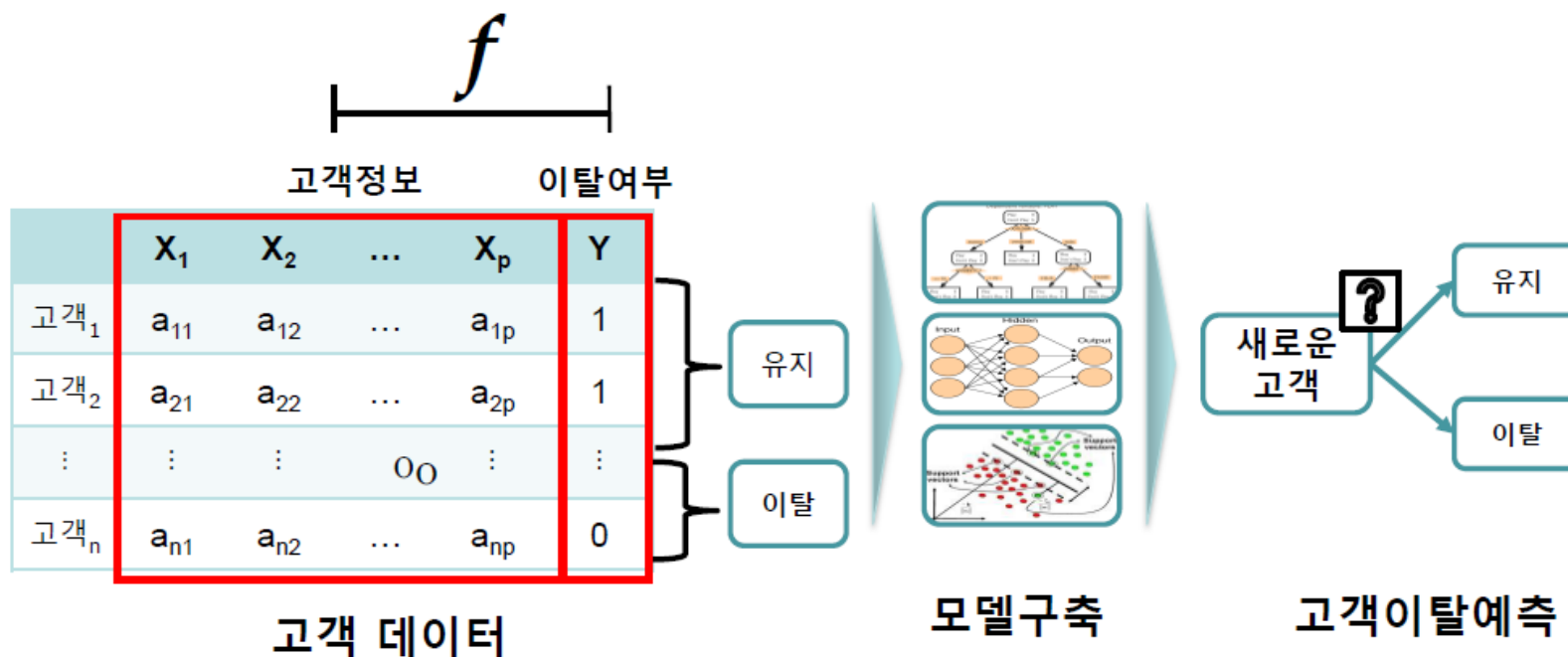
▪ 분류 vs 회귀

분류	회귀
분류는 결과를 클래스로 그룹화하는 것을 의미	회귀 분석은 학습 데이터를 사용해 출력 값을 예측하는 것을 의미
훈련 데이터를 사용하여 사기거래와 정상 거래를 예측하는 분류	훈련 데이터로부터 주식 가격을 예측하는 회귀
이산/범주형 변수인 경우 분류 문제임	실수/연속형 변수이면 회귀 문제임

데이터 구조

❖ 지도학습(Supervised learning)

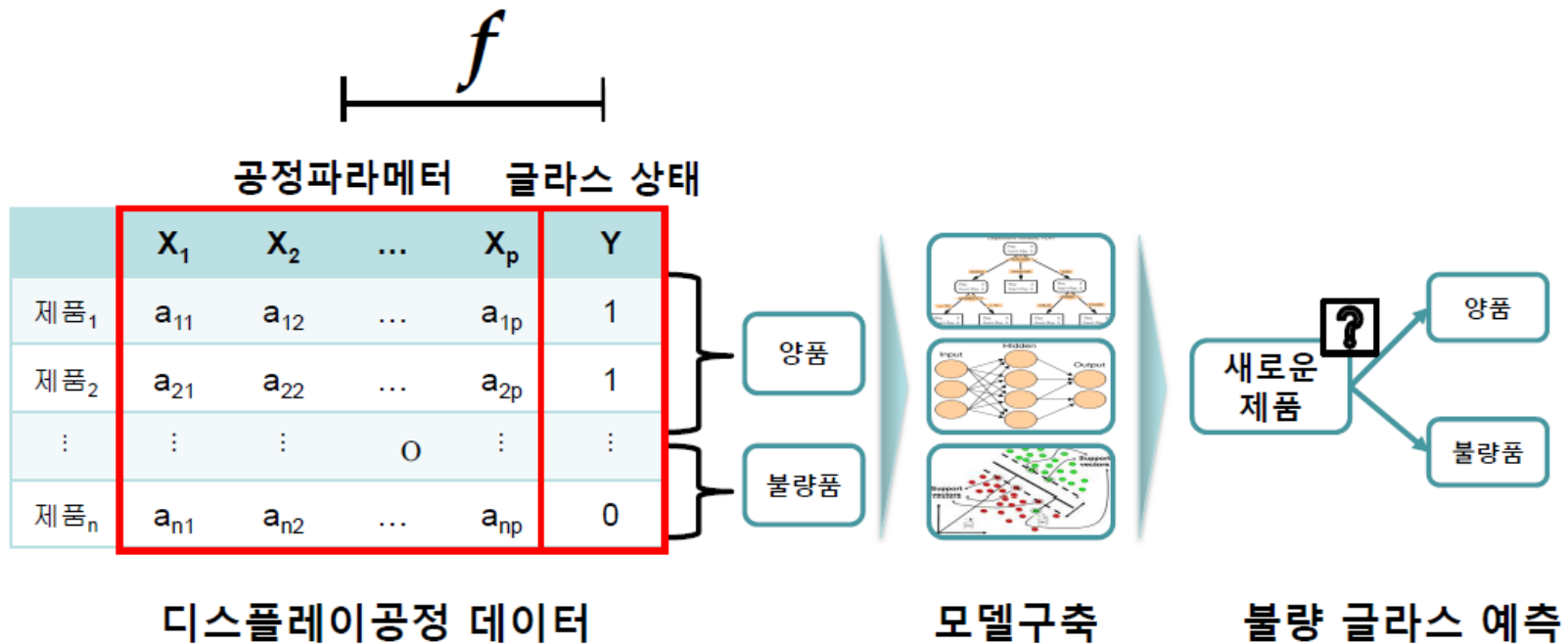
- 범주예측예제-고객이탈예측
- 고객의 정보(성별, 연령, 직업, 연봉 등)를 이용하여, 고객 이탈 여부를 예측



데이터 구조

❖ 지도학습(Supervised learning)

- 범주예측예제-불량예측
- 디스플레이 공정에서 공정 파라미터의 측정값들을 이용하여, 해당 글라스가 양품인지 불량품인지를 여부를 예측



데이터 구조

❖ 지도 학습(Supervised learning)

▪ 지도 학습 알고리즘과 분류 및 회귀 문제

번호	알고리즘 이름	분류 문제	회귀 문제
01	선형회귀(linear regression)	×	○
02	정규화(regularization)	×	○
03	로지스틱 회귀(logistic regression)	○	×
04	서포트 벡터 머신(support vector machine)	○	○
05	커널(kernel) 기법을 적용한 서포트 벡터 머신	○	○
06	나이브 베이즈 분류(Naïve Bayes classification)	○	×
07	랜덤 포레스트(random forest)	○	○
08	신경망(neural network)	○	○
09	k-최근접 이웃 알고리즘(k-nearest neighbors algorithm, kNN)	○	○

데이터 구조

❖ 비지도 학습(Unsupervised learning)

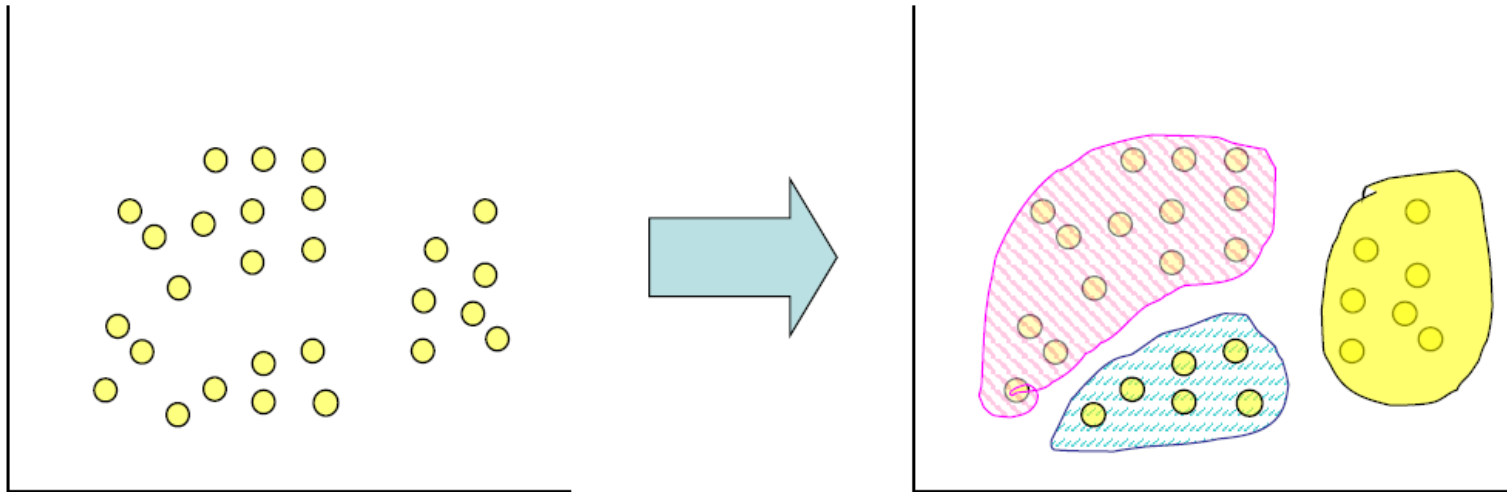
- 군집화(cluster)
- 군집은 서로 더 비슷한 개체 집합을 동일한 그룹(군집이라고 함)으로 그룹화하는 작업
- 개체는 사전에 미리 정의 되어있지 않음
 - 1반 남자
 - 1반 여자
 - 2반 여자
 - 2반 남자
 - '1반 '과 '2반' 또는 ' 남자' 와 '여자' 두 가지 카테고리로 군집화 할 수 있음
- 가장 대중적인 군집 알고리즘으로는 k-means 군집과 계층적 군집이 있음

데이터 구조

❖ 비지도 학습(Unsupervised learning)

▪ 군집화

- 비슷한 특성을 가진 데이터들끼리 그룹화 같은 그룹내 요소들은 아주 유사하고 다른 그룹과는 확연히 다름

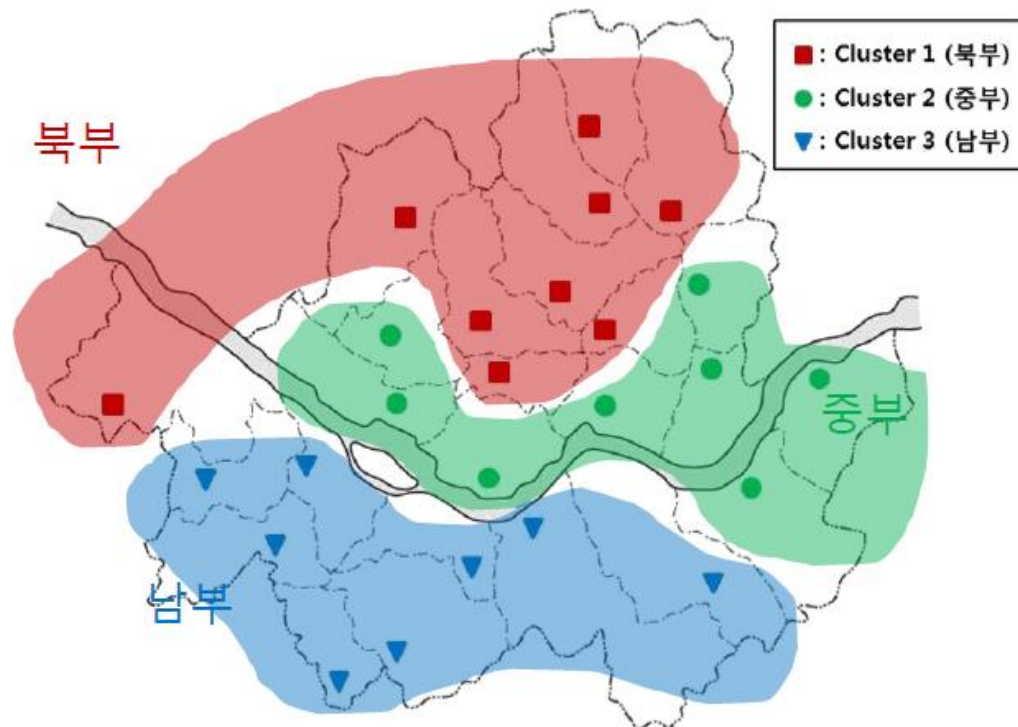


데이터 구조

❖ 비지도 학습(Unsupervised learning)

▪ 군집화

- 비슷한 특성을 가진 데이터들끼리 그룹화 같은 그룹내 요소들은 아주 유사하고 다른 그룹과는 확연히 다름



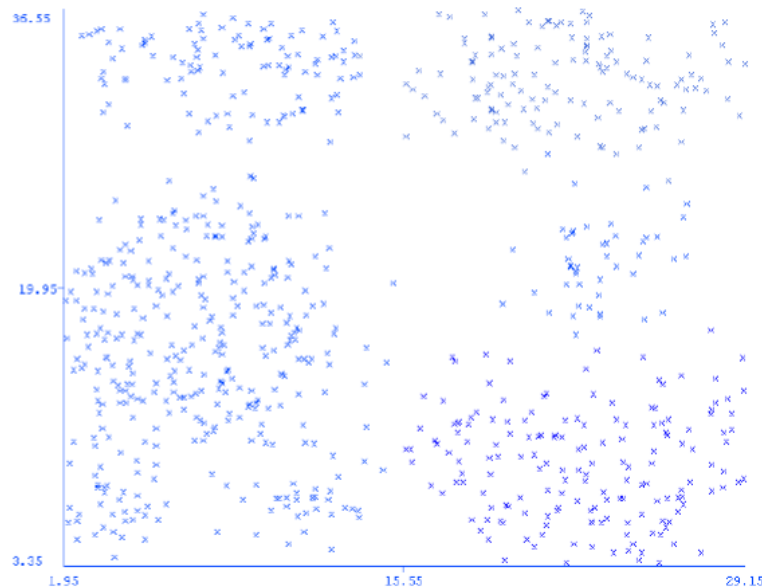
데이터 구조

❖ 비지도 학습(Unsupervised learning)

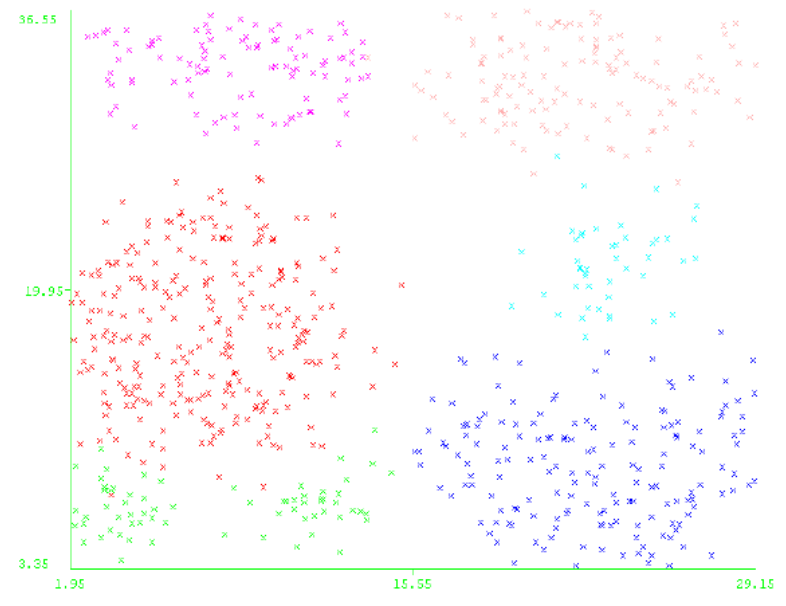
■ 군집화 - k-means 군집

- 각 관측치가 가장 가까운 평균을 갖는 클러스터에 속하고 클러스터의 프로토타입 역할을 하는 k개의 클러스터로 n개의 관측치를 분할함

군집되지 않은 원본 데이터



군집된 데이터

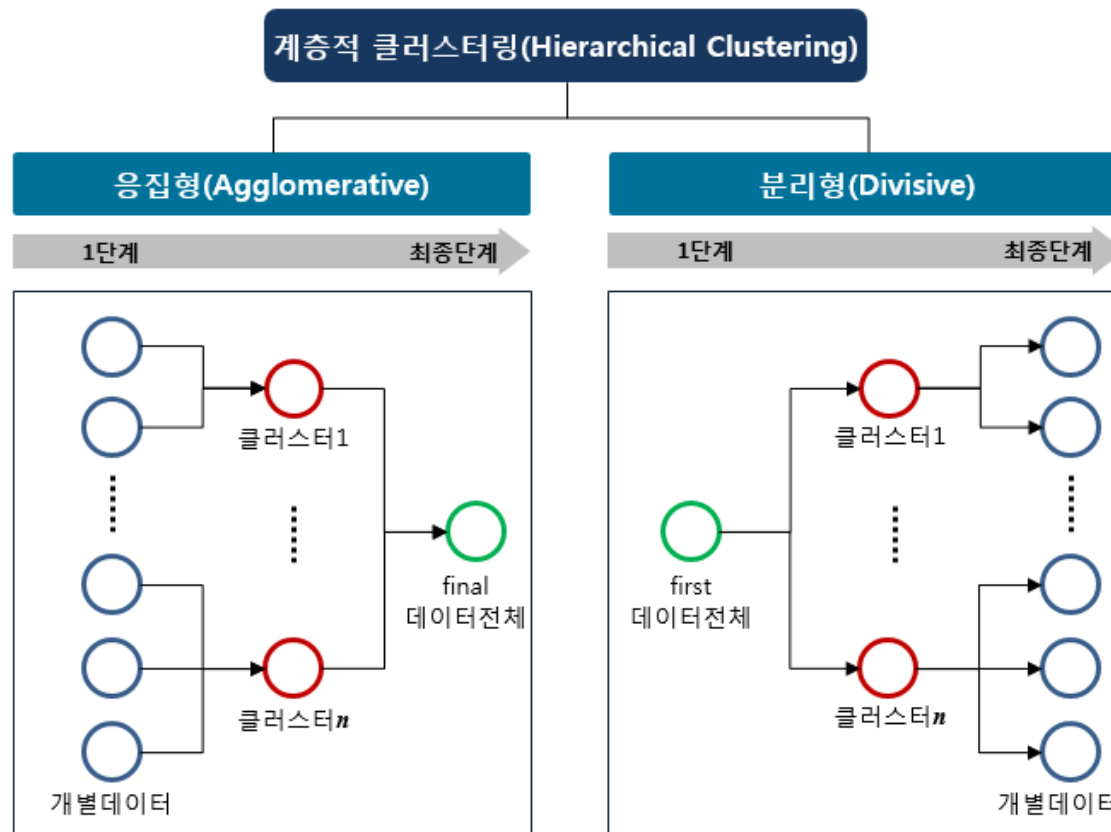


데이터 구조

❖ 비지도학습(Unsupervised learning)

▪ 군집화 – 계층적 군집화

- 특정 알고리즘에 의해 데이터를 연결하여 계층적 클러스터를 구성해 나가는 방법

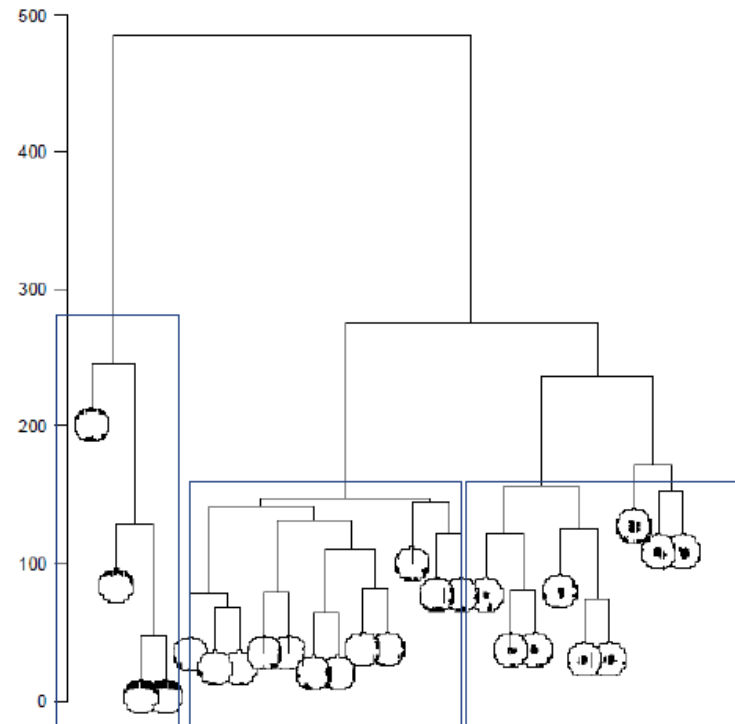
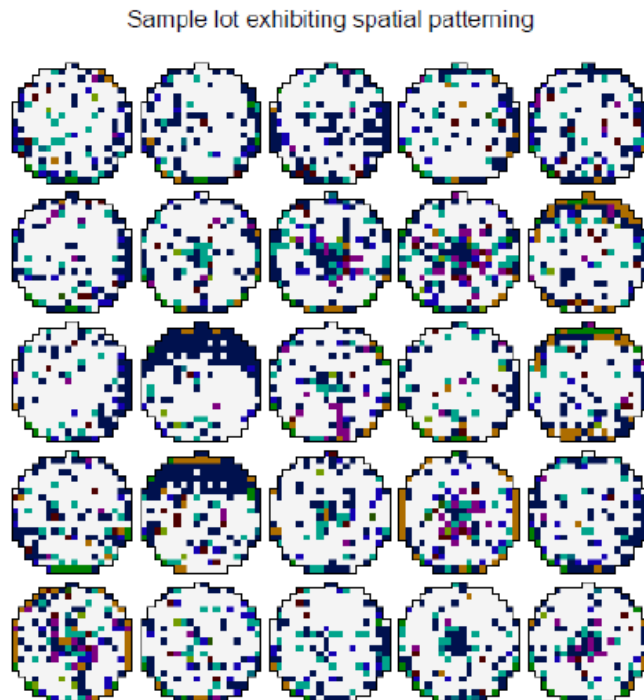


데이터 구조

❖ 비지도 학습(Unsupervised learning)

▪ 군집화 – 계층적 군집화

- 웨이퍼 map 군집화를 통한 웨이퍼 불량 패턴 파악

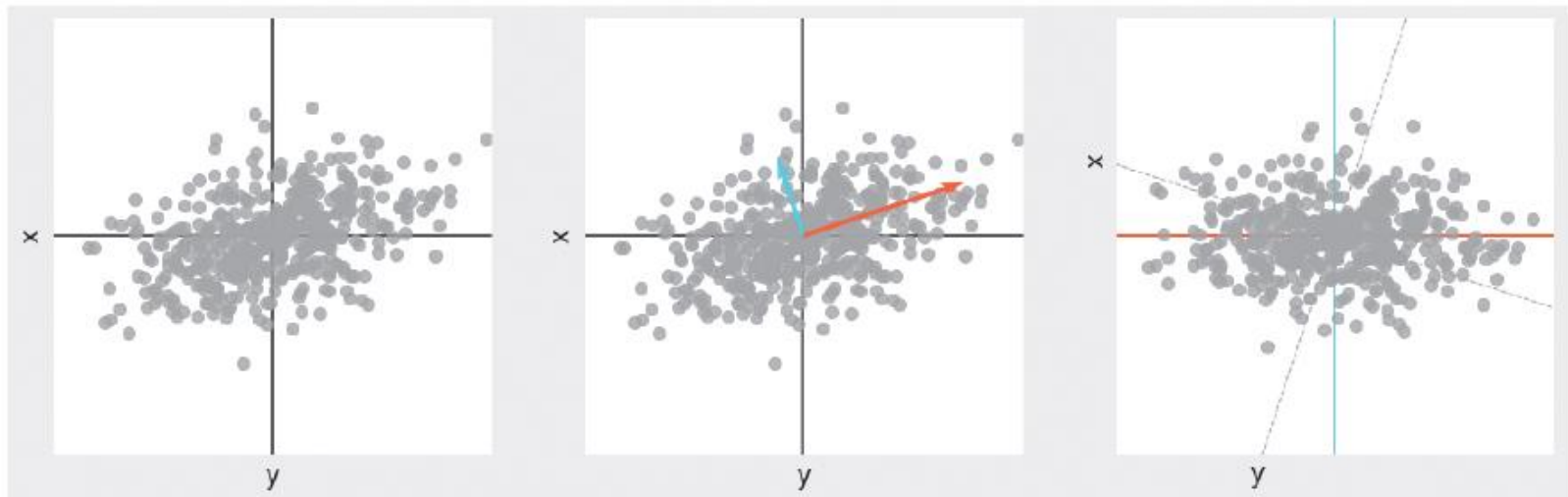


데이터 구조

❖ 비지도학습(Unsupervised learning)

■ 주성분분석

- 수천개 또는 수백만개의 특징을 가지는 경우 차원 축소를 통해 반응변수를 가장 크게 변화시키는 특징을 추려내는 과정
- 데이터의 분산(variance)을 최대한 보존하면서 서로 직교하는 새 기저(축)를 찾아, 고차원 공간의 표본들을 선형 연관성이 없는 저차원 공간으로 변환하는 기법



데이터 구조

❖ 비지도 학습(Unsupervised learning)

▪ 비지도 학습 알고리즘과 차원 축소 및 클러스터링 여부

번호	알고리즘 이름	차원 축소	클러스터링
10	주성분 분석(principal component analysis, PCA)	○	×
11	잠재 의미 분석(latent semantic analysis, LSA)	○	×
12	음수 미포함 행렬 분해(non-negative matrix factorization, NMF)	○	×
13	잠재 디리클레 할당(latent Dirichlet allocation, LDA)	○	×
14	k-평균 알고리즘(k-means algorithm)	×	○
15	가우시안 혼합 모델(Gaussian mixture model)	×	○
16	국소 선형 임베딩(local linear embedding, LLE)	○	×
17	t-분포 확률적 임베딩(t-distributed stochastic neighbor embedding, t-SNE)	○	×

데이터 구조

❖ 머신러닝을 활용한 데이터 분석 예

- Toyota Corolla 중고차가격예측
- 데이터: 차의 특징에 따른 Toyota Corolla 중고차 1,442대의 가격

Price	Age	KM	Fuel_type	HP	Metallic	Automatic	cc	Doors	Quarterly_Tax	Weight
13500	23	46986	Diesel	90	1	0	2000	3	210	1165
13750	23	72937	Diesel	90	1	0	2000	3	210	1165
13950	24	41711	Diesel	90	1	0	2000	3	210	1165
14950	26	48000	Diesel	90	0	0	2000	3	210	1165
13750	30	38500	Diesel	90	0	0	2000	3	210	1170
12950	32	61000	Diesel	90	0	0	2000	3	210	1170
16900	27	94612	Diesel	90	1	0	2000	3	210	1245
18600	30	75889	Diesel	90	1	0	2000	3	210	1245
21500	27	19700	Petrol	192	0	0	1800	3	100	1185
12950	23	71138	Diesel	69	0	0	1900	3	185	1105

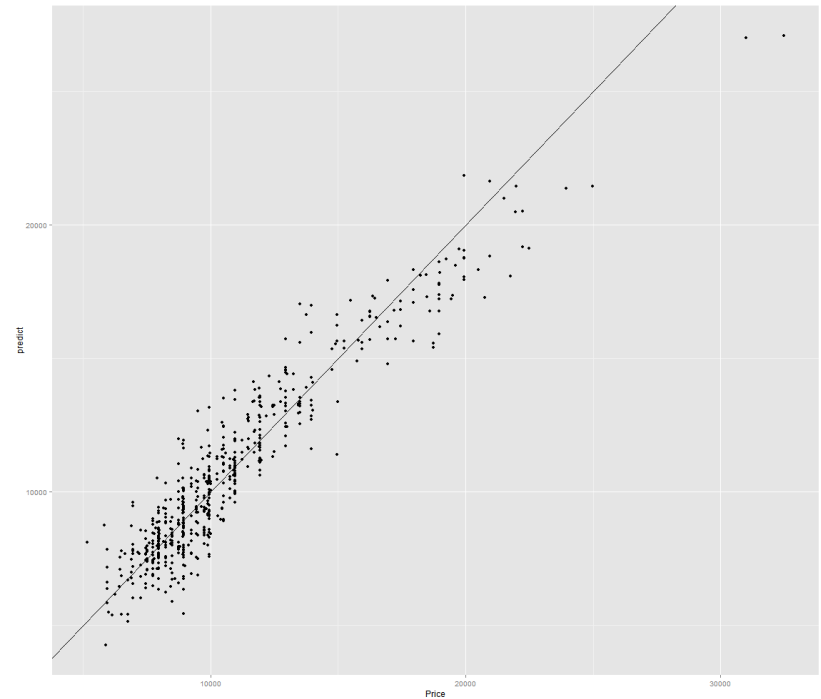
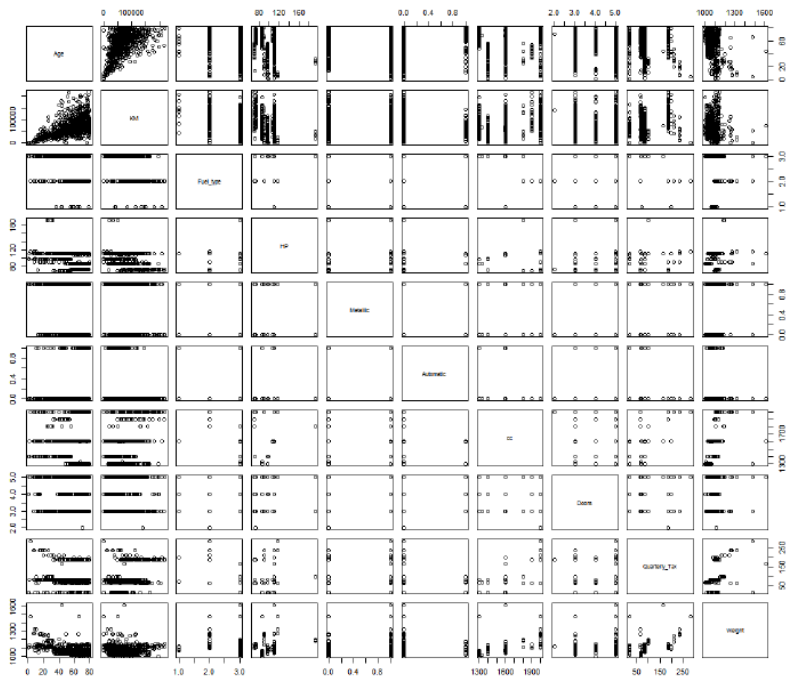
- Corolla 중고차가격 = $305.6 - 128.4 \cdot \text{Age} - 0.015 \cdot \text{KM} + 5885.7 \cdot \text{Fuel_type_Diesel} + 2610.3 \cdot \text{Fuel_type_Petrol} + 63.4 \cdot \text{HP} + 493.6 \cdot \text{Automatic} - 3.9 \cdot \text{CC} + 14.7 \cdot \text{Quarterly_Tax} + 13.2 \cdot \text{weight}$

데이터 구조

❖ 머신러닝을 활용한 데이터 분석 예

▪ 상관관계 분석을 통해 특징 선정

▪ 성능 평가



데이터 구조

❖ 머신러닝을 활용한 데이터 분석 예

▪ 어플리케이션에 적용

도요타 선택 초기화					
일반등록		20개씩 보기 ▼			
차량정보		연식 ▼	주행거리 ▼	가격 ▼	지역 / 등록일 ▼
	도요타 코롤라 1.8 CL 10세대 오토 가솔린 성능기록	11/12식	90,000km	1,150만원	인천 05/15
	도요타 코롤라 1.8 LE 10세대 오토 가솔린	11/03식	87,029km	1,090만원	대구 05/15
	도요타 코롤라 CE 4Dr Sedan 오토 가솔린 성능기록	06/01식	152,760km	370만원	대전 05/15
	도요타 코롤라 1.8 CO 10세대 오토 가솔린 성능기록	10/12식	99,392km	850만원	대구 05/15

<http://www.encar.com>

▲ TOP

데이터 구조

❖ 회귀분석 실습

- Boston house prices dataset

[01]	CRIM	자치시(town) 별 1인당 범죄율
[02]	ZN	25,000 평방피트를 초과하는 거주지역의 비율
[03]	INDUS	비소매상업지역이 점유하고 있는 토지의 비율
[04]	CHAS	찰스강에 대한 더미변수(강의 경계에 위치한 경우는 1, 아니면 0)
[05]	NOX	10ppm 당 농축 일산화질소
[06]	RM	주택 1가구당 평균 방의 개수
[07]	AGE	1940년 이전에 건축된 소유주택의 비율
[08]	DIS	5개의 보스턴 직업센터까지의 접근성 지수
[09]	RAD	방사형 도로까지의 접근성 지수
[10]	TAX	10,000 달러 당 재산세율
[11]	PTRATIO	자치시(town)별 학생/교사 비율
[12]	B	$1000(Bk - 0.63)^2$, 여기서 Bk는 자치시별 흑인의 비율을 말함.
[13]	LSTAT	모집단의 하위계층의 비율(%)
[14]	MEDV	본인 소유의 주택가격(중앙값) (단위: \$1,000)

데이터 구조

❖ 회귀분석 실습

■ 데이터 불러오기

```
1 from sklearn import datasets
2 boston_house_prices = datasets.load_boston()
3
4 # 로드한 boston 전체 데이터에 key 값을 출력
5 print(boston_house_prices.keys())
6 # boston 전체 데이터 중 data에 대한 전체 행, 열 길이를 출력
7 print(boston_house_prices.data.shape)
8 # boston 데이터에 컬럼 이름을 출력
9 print(boston_house_prices.feature_names)
```

```
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename'])
(506, 13)
['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO'
 'B' 'LSTAT']
```

```
1 print(boston_house_prices.DESCR)
```

- 변수 설명
 - `Boston_house_prices` : sklearn패키지에서 제공하는 데이터셋 `boston`을 저장한 변수
- 모듈 설명
 - `datasets` : sklearn 패키지에서 제공하는 open dataset을 로드할 때 사용하는 모듈
- 함수 설명
 - `datasets.load_boston()` : sklearn의 내장 데이터셋 `boston`을 로드하는 함수
 - `dict_keys.keys()` : 딕셔너리 형식에 데이터에서 key값만 출력하는 함수
- 코드 설명
 - `print(boston_house_prices.keys())` : 로드한 boston 전체 데이터에 key 값을 출력
 - `print(boston_house_prices.data.shape)` : boston 전체 데이터 중 data에 대한 전체 행, 열 길이를 출력
 - `print(boston_house_prices.feature_names)` : boston 데이터에 컬럼 이름을 출력

데이터 구조

❖ 회귀분석 실습

■ 판다스로 데이터 변환 및 columns 지정

```
1 import pandas as pd
2 data = pd.DataFrame(boston_house_prices.data)
3 data.tail()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99	9.67
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90	9.08
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90	5.64
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45	6.48
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90	7.88

- 변수 설명
 - data_frame : boston_house_price 변수에 전체 데이터 중 data에 해당하는 값을 pandas.DataFrame형으로 변경 후 저장한 변수
- 함수 설명
 - tailer.tail() : 전체 데이터 중 기본 값으로 마지막 5개 데이터 출력함 (괄호 안에 숫자를 입력해 데이터 개수 교체가 가능)

```
1 data.columns = boston_house_prices.feature_names
2 data.tail()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99	9.67
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90	9.08
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90	5.64
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45	6.48
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90	7.88

데이터 구조

❖ 회귀분석 실습

▪ 판다스로 데이터 변환 및 columns 지정

```
1 data['Price'] = boston_house_prices.target
2 data.tail()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	Price
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99	9.67	22.4
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90	9.08	20.6
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90	5.64	23.9
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90	7.88	11.9

- 코드 설명
- `data_frame['Price'] = boston_house_prices.target :`
data_frame에 Price란 컬럼을 추가하고 데이터는 target에 저장된 데이터를 사용

데이터 구조

❖ 회귀분석 실습

■ 상관관계 분석

```
1 data.corr()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	Price
CRIM	1.000000	-0.200469	0.406583	-0.055892	0.420972	-0.219247	0.352734	-0.379670	0.625505	0.582764	0.289946	-0.385064	0.455621	-0.388305
ZN	-0.200469	1.000000	-0.533828	-0.042697	-0.516604	0.311991	-0.569537	0.664408	-0.311948	-0.314563	-0.391679	0.175520	-0.412995	0.360445
INDUS	0.406583	-0.533828	1.000000	0.062938	0.763651	-0.391676	0.644779	-0.708027	0.595129	0.720760	0.383248	-0.356977	0.603800	-0.483725
CHAS	-0.055892	-0.042697	0.062938	1.000000	0.091203	0.091251	0.086518	-0.099176	-0.007368	-0.035587	-0.121515	0.048788	-0.053929	0.175260
NOX	0.420972	-0.516604	0.763651	0.091203	1.000000	-0.302188	0.731470	-0.769230	0.611441	0.668023	0.188933	-0.380051	0.590879	-0.427321
RM	-0.219247	0.311991	-0.391676	0.091251	-0.302188	1.000000	-0.240265	0.205246	-0.209847	-0.292048	-0.355501	0.128069	-0.613808	0.695360
AGE	0.352734	-0.569537	0.644779	0.086518	0.731470	-0.240265	1.000000	-0.747881	0.456022	0.506456	0.261515	-0.273534	0.602339	-0.376955
DIS	-0.379670	0.664408	-0.708027	-0.099176	-0.769230	0.205246	-0.747881	1.000000	-0.494588	-0.534432	-0.232471	0.291512	-0.496996	0.249929
RAD	0.625505	-0.311948	0.595129	-0.007368	0.611441	-0.209847	0.456022	-0.494588	1.000000	0.910228	0.464741	-0.444413	0.488676	-0.381626
TAX	0.582764	-0.314563	0.720760	-0.035587	0.668023	-0.292048	0.506456	-0.534432	0.910228	1.000000	0.460853	-0.441808	0.543993	-0.468536
PTRATIO	0.289946	-0.391679	0.383248	-0.121515	0.188933	-0.355501	0.261515	-0.232471	0.464741	0.460853	1.000000	-0.177383	0.374044	-0.507787
B	-0.385064	0.175520	-0.356977	0.048788	-0.380051	0.128069	-0.273534	0.291512	-0.444413	-0.441808	-0.177383	1.000000	-0.366087	0.333461
LSTAT	0.455621	-0.412995	0.603800	-0.053929	0.590879	-0.613808	0.602339	-0.496996	0.488676	0.543993	0.374044	-0.366087	1.000000	-0.737663
Price	-0.388305	0.360445	-0.483725	0.175260	-0.427321	0.695360	-0.376955	0.249929	-0.381626	-0.468536	-0.507787	0.333461	-0.737663	1.000000

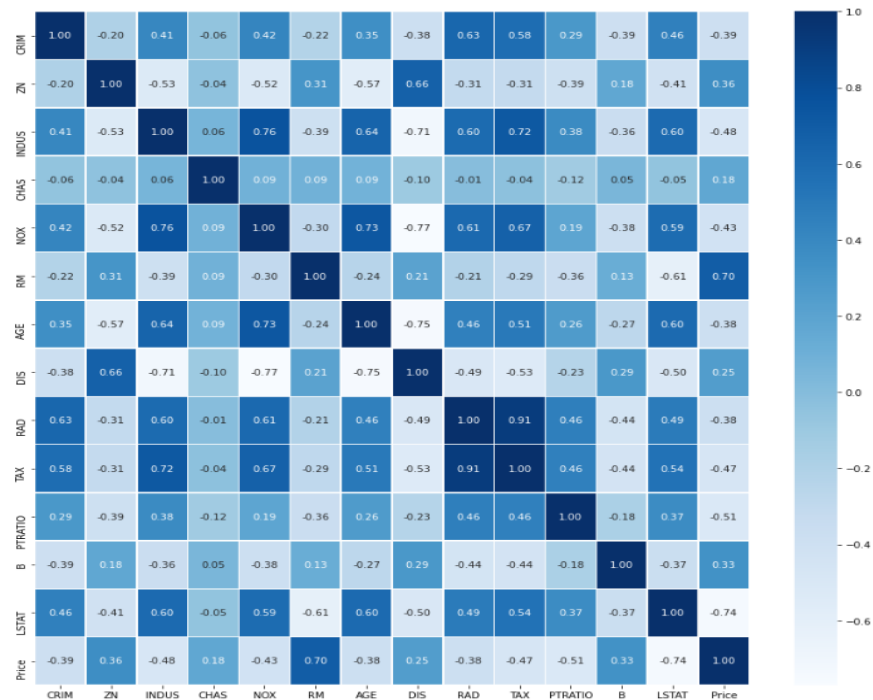
데이터 구조

❖ 회귀분석 실습

■ 상관관계 분석

```
1 import matplotlib.pyplot as plt
2 import seaborn as sns
```

```
1 plt.figure(figsize=(15,15))
2 sns.heatmap(data = data.corr(), annot=True,
3   fmt = '.2f', linewidths=.5, cmap='Blues')
```



데이터 구조

❖ 회귀분석 실습

■ 상관관계 분석



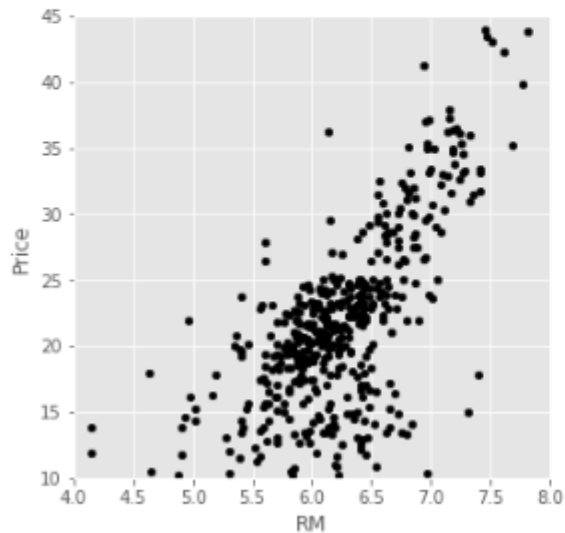
데이터 구조

❖ 회귀분석 실습

■ 산점도

```
1 import matplotlib.pyplot as plt
2 import matplotlib
3 %matplotlib inline
4 matplotlib.style.use('ggplot')
5
6 data.plot(kind='scatter', x="RM", y="Price", figsize=(5, 5), color='black', xlim=(4,8), ylim=(10,45))
```

<matplotlib.axes._subplots.AxesSubplot at 0x1e56b5a5828>



데이터 구조

❖ 회귀분석 실습

■ 회귀분석

```
1 from sklearn import linear_model
```

```
1 linear_regression = linear_model.LinearRegression()  
2 linear_regression.fit(X=pd.DataFrame(data['RM']), y=data['Price'])  
3 prediction = linear_regression.predict(X=pd.DataFrame(data['RM']))  
4 print('a value: ', linear_regression.intercept_)  
5 print('b value: ', linear_regression.coef_)
```

```
a value: -34.670620776438554  
b value: [9.10210898]
```

```
1 residuals = data['Price'] - prediction  
2 residuals.describe()
```

```
count    5.060000e+02  
mean     -7.863714e-16  
std       6.609606e+00  
min      -2.334590e+01  
25%      -2.547477e+00  
50%       8.976267e-02  
75%       2.985532e+00  
max       3.943314e+01  
Name: Price, dtype: float64
```

■ 코드 설명

- `print(linear_regression.intercept_)` : 선형회귀분석에 a 회귀계수 출력
- `print(linear_regression.coef_)` : 선형회귀분석에 b 회귀계수 출력

■ 변수 설명

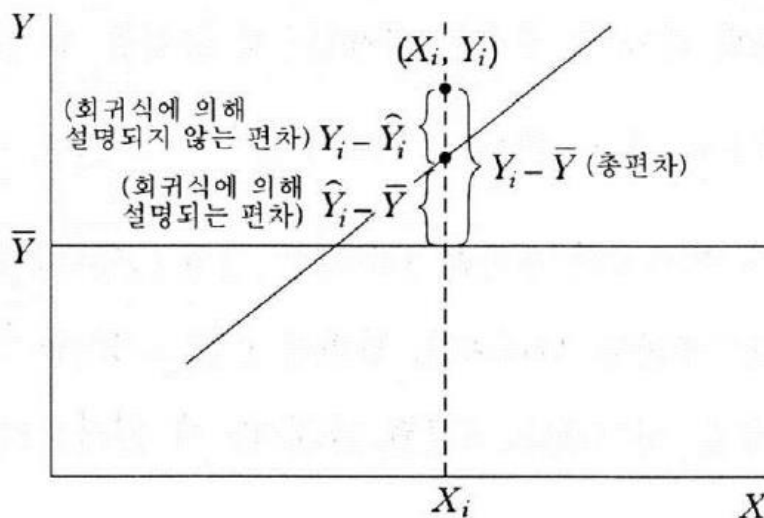
- `prediction` : 'RM' 데이터를 선형회귀 분석모델을 통해 예측한 값을 저장한 변수
- `residuals` : 예측하고자 하는 'Price'값에서 모델을 통해 예측된 값을 빼서 오차 값을 저장한 변수

데이터 구조

❖ 회귀분석 실습

▪ 회귀분석

$$R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST}$$



$$SST = \sum_{i=1}^n (Y_i - \bar{Y})^2 \quad SSR = \sum_{i=1}^n (\hat{Y}_i - \bar{Y})^2 \quad SSE = \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

$$\mathbf{SST = SSR + SSE}$$

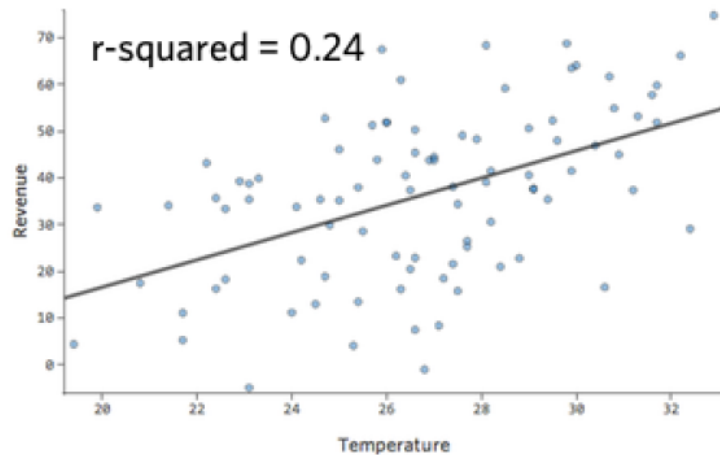
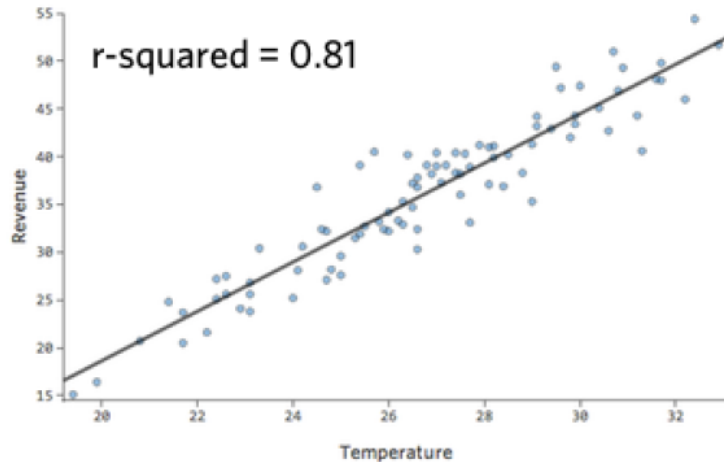
데이터 구조

❖ 회귀분석 실습

▪ 회귀분석

- R^2 는 0과 1 사이에 존재
- $R^2 = 1$: 회귀직선으로 Y의 총변동이 완전히 설명
즉, 모든 측정값들이 회귀직선 위에 있음
- $R^2 = 0$: 추정된 회귀직선은 X와 Y의 관계를 전혀 설명하지 못함
- 사용하고 있는 예측변수(X)가 반응변수(Y)의 분산을 얼마나 줄였는지 정도

$$R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST}$$



데이터 구조

❖ 회귀분석 실습

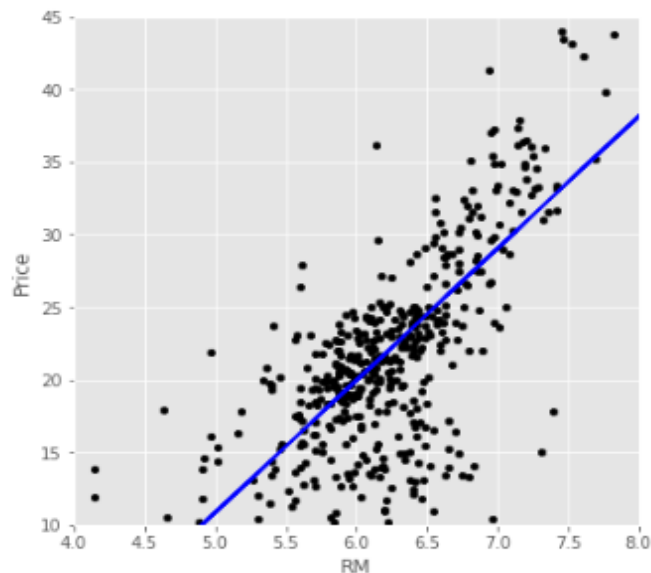
■ 회귀분석

```
1 SSE = (residuals**2).sum()
2 SST = ((data['Price']-data['Price'].mean())**2).sum()
3 R_squared = 1 - (SSE/SST)
4 print('R_squared: ', R_squared)
```

R_squared: 0.48352545599133423

```
1 data.plot(kind='scatter', x='RM', y='Price', figsize=(6,6), color='black',
2           xlim=(4,8), ylim=(10, 45))
3
4 plt.plot(data['RM'], prediction, color='b')
```

[<matplotlib.lines.Line2D at 0x1e56b554e80>]



■ 변수 설명

- SSE : 결정계수 값을 구하기 위해 필요한 SSE 값을 계산 후 저장한 변수
- SST : 결정계수 값을 구하기 위해 필요한 SST 값을 계산 후 저장한 변수
- R_squared : 적합도 검증을 위해 필요한 결정계수 값을 계산 후 저장한 변수

데이터셋 구성을 통한 검증 방법

❖ 데이터셋 구성

▪ Training set(훈련데이터)

- 모델 훈련에 사용하는 데이터 셋으로 관측치의 모음

▪ Validation set(검증 데이터)

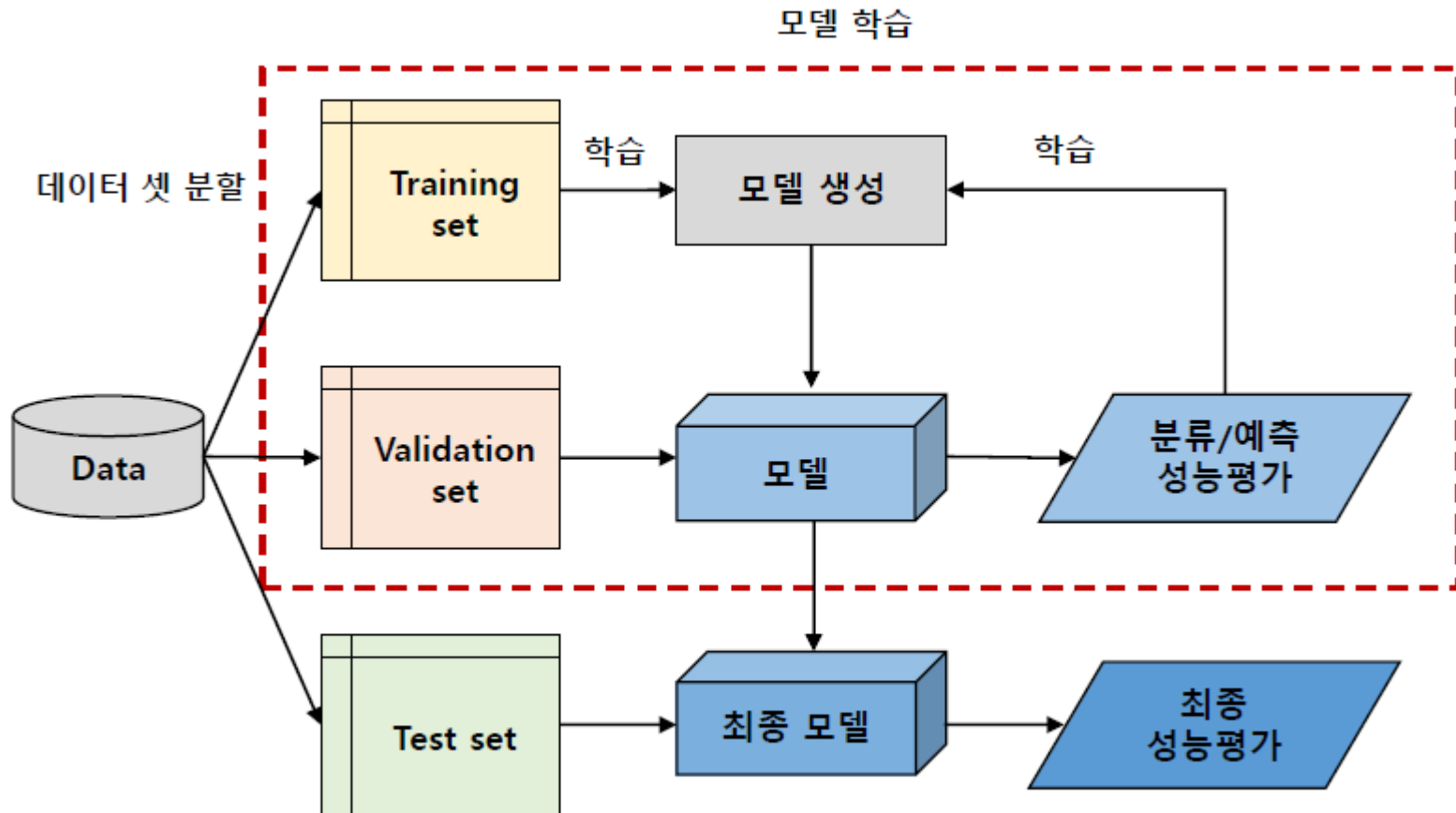
- 모델 훈련에 적절한 지점을 찾기 위해 사용하는 데이터셋으로 과적합(over fitting) 또는 과소적합(under-fitting)을 방지하기 위한 stopping point를 찾음

▪ Test set(테스트 데이터)

- 모델의 성능을 평가하기 위해 사용하는 데이터 셋

데이터셋 구성을 통한 검증 방법

❖ 데이터셋 구성



데이터셋 구성을 통한 검증 방법

❖ 데이터셋 구성

▪ Holdout

- 데이터셋의 일부를 훈련용 데이터, 나머지 일부를 테스트용 데이터로 구성

▪ Random subsampling

- 모델 훈련에 적절한 지점을 찾기 위해 사용하는 데이터셋으로 과적합(over fitting) 또는 과소적합(under-fitting)을 방지하기 위한 stopping point를 찾음

▪ Cross validation

- 모델의 성능을 평가하기 위해 사용하는 데이터 셋

▪ Bootstrap

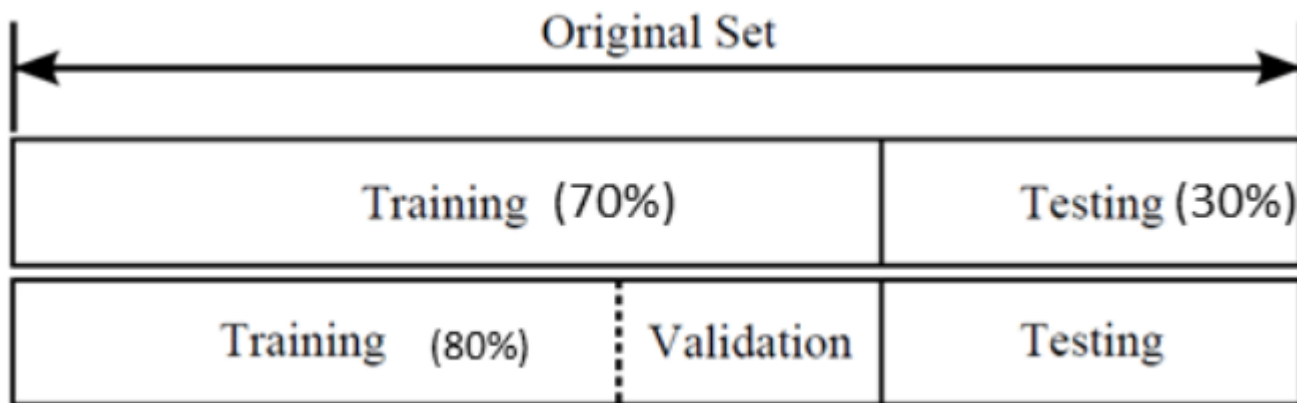
- 중복을 허용하는 샘플 추출

데이터셋 구성을 통한 검증 방법

❖ 데이터셋 구성

▪ Holdout

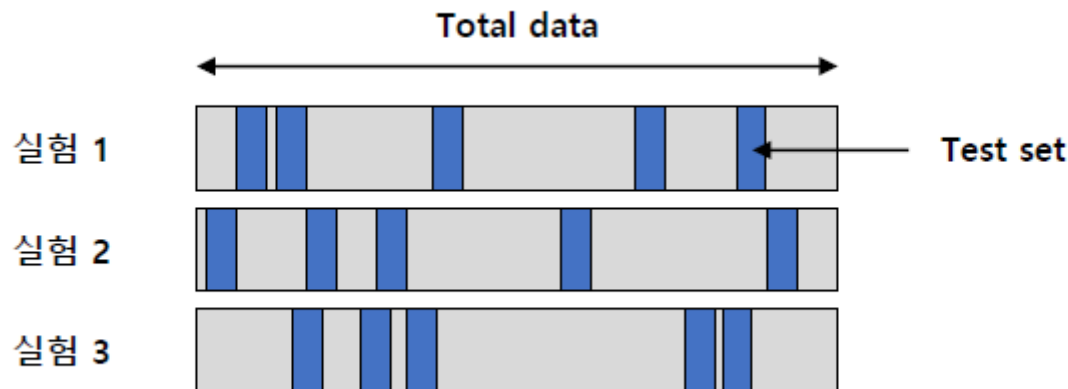
- 데이터셋을 train set, test set 두 셋으로 나눔
- 아래 그림과 같이 train set과 test set의 일정비율을 설정
- Train set이 작으면 모델 정확도의 분산이 커짐
- Train set이 커지면 test set으로 부터 측정한 정확도의 신뢰도 하락
- 위와 같은 단점을 극복하기 위해 hold out을 반복적으로 실행하는 것이 random subsampling



데이터셋 구성을 통한 검증 방법

❖ 데이터셋 구성

- Random subsampling
- K개의 부분 데이터셋을 사용함
 - 각 부분 데이터셋은 랜덤으로 정해짐
 - 각 실험마다 training set으로 모델을 훈련, test set으로 모델을 테스트함
- 최종 성능은 각 실험성능의 평균으로 도출



데이터셋 구성을 통한 검증 방법

❖ 데이터셋 구성

▪ Holdout

```
import numpy as np
X = np.arange(20).reshape(10, 2)

y = np.arange(10)

print(X)
print()
```

```
[[ 0  1]
 [ 2  3]
 [ 4  5]
 [ 6  7]
 [ 8  9]
 [10 11]
 [12 13]
 [14 15]
 [16 17]
 [18 19]]
[0 1 2 3 4 5 6 7 8 9]
```

데이터셋 구성을 통한 검증 방법

❖ 데이터셋 구성

▪ Holdout

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.4,
                                                    shuffle=False, # 순차적으로 병렬
                                                    random_state=1004)

print('X_train shape:', X_train.shape)
print('X_test shape:', X_test.shape)
print('y_train shape:', y_train.shape)
print('y_test shape:', y_test.shape)

print('X_train shape:', X_train)
print('X_test shape:', X_test)

print('y_train shape:', y_train)
print('y_test shape:', y_test)
```

X_train shape: (6, 2)
X_test shape: (4, 2)
y_train shape: (6,)
y_test shape: (4,)
X_train shape: [[0 1]
[2 3]
[4 5]
[6 7]
[8 9]
[10 11]]
X_test shape: [[12 13]
[14 15]
[16 17]
[18 19]]
y_train shape: [0 1 2 3 4 5]
y_test shape: [6 7 8 9]

데이터셋 구성을 통한 검증 방법

❖ 데이터셋 구성

▪ Holdout

```
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.4,
                                                    shuffle=True, # 랜덤 병렬
                                                    random_state=1004)

print('X_train shape:', X_train.shape)
print('X_test shape:', X_test.shape)
print('y_train shape:', y_train.shape)
print('y_test shape:', y_test.shape)
```

```
print('X_train shape:', X_train)
```

```
print('X_test shape:', X_test)
```

```
print('y_train shape:', y_train)
```

```
print('y_test shape:', y_test)
```

```
X_train shape: (6, 2)
```

```
X_test shape: (4, 2)
```

```
y_train shape: (6,)
```

```
y_test shape: (4,)
```

```
X_train shape: [[ 2  3]
```

```
 [ 8  9]
```

```
 [ 6  7]
```

```
 [14 15]
```

```
 [10 11]
```

```
 [ 4  5]]
```

```
X_test shape: [[ 0  1]
```

```
 [12 13]
```

```
 [16 17]
```

```
 [18 19]]
```

```
y_train shape: [1 4 3 7 5 2]
```

```
y_test shape: [0 6 8 9]
```

데이터셋 구성을 통한 검증 방법

❖ 데이터셋 구성

▪ Holdout

```
from sklearn.datasets import load_boston

boston = load_boston()
dfX = pd.DataFrame(boston.data, columns=boston.feature_names)
dfy = pd.DataFrame(boston.target, columns=["MEDV"])
boston_df = pd.concat([dfX, dfy], axis=1)

print('train shape:', boston_df.shape)
```

train shape: (506, 14)

```
from sklearn.model_selection import train_test_split

df_train, df_test = train_test_split(boston_df, test_size=0.3, random_state=0)
df_train.shape, df_test.shape
```

((354, 14), (152, 14))

```
X_train, X_test, y_train, y_test = train_test_split(dfX, dfy, test_size=0.3, random_state=0)
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

((354, 13), (354, 1), (152, 13), (152, 1))

데이터셋 구성을 통한 검증 방법

❖ 데이터셋 구성

▪ Holdout

```
from sklearn.model_selection import train_test_split
```

```
df_train, df_test = train_test_split(boston_df, test_size=0.3, random_state=0)  
df_train.shape, df_test.shape
```

```
((354, 14), (152, 14))
```

```
X_train, X_test, y_train, y_test = train_test_split(dfX, dfy, test_size=0.3, random_state=0)  
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

```
((354, 13), (354, 1), (152, 13), (152, 1))
```

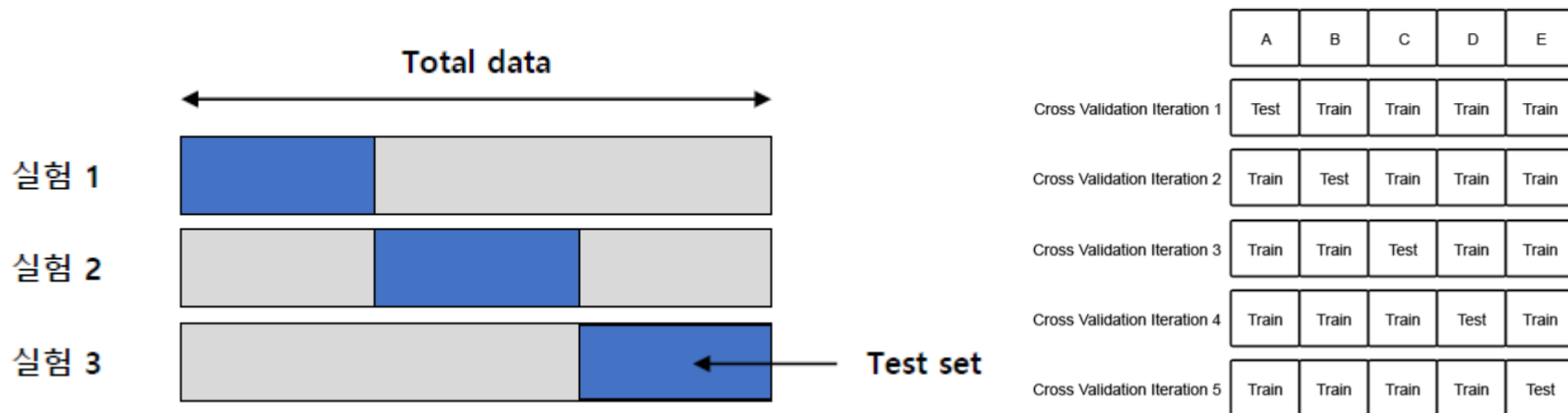
```
X1_train, X1_valid, y1_train, y1_valid = train_test_split(X_train, y_train, test_size=0.2, random_state=0)  
X1_train.shape, y1_valid.shape, X1_valid.shape, y1_valid.shape, X_test.shape, y_test.shape
```

```
((283, 13), (71, 1), (71, 13), (71, 1), (152, 13), (152, 1))
```

데이터셋 구성을 통한 검증 방법

❖ 데이터셋 구성

- K fold cross validation
- Data set을 k개의 fold로 나누어 사용
 - K번의 실험을 진행하며, k-1개 fold는 training set, 1개 fold는 test set으로 사용
- Random subsampling과 유사하지만, k-fold cross validation을 사용할 경우 모든 데이터를 train과 test에 적용할 수 있음
- 최종 성능은 k번의 실험 성능의 평균으로 도출



데이터셋 구성을 통한 검증 방법

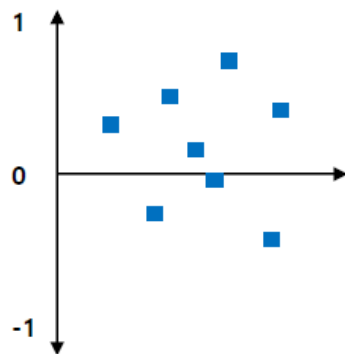
❖ 머신러닝을 활용한 데이터 분석 예

▪ Residuals

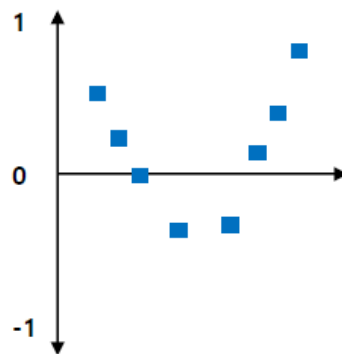
- 회귀 분석 모델의 예측값과 실제값의 차이 즉, 오차(error)를 잔차(residual)라고 함

▪ Residual plot

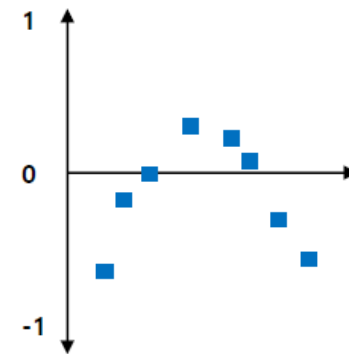
- Residual plot은 세로축에 residual, 가로축에 독립변수를 나타내는 산포도
- 산포도를 통해 어떤 선형모델이 데이터에 적합한지 알 수 있음
- 적합한 선형모델: residual이 x축을 기준으로 랜덤으로 분포하는 상태
- 적합한 비선형 모델: residual이 패턴을 보이며 분포하는 상태



Linear Appropriate



Non-Linear Appropriate



Non-Linear Appropriate

데이터셋 구성을 통한 검증 방법

❖ 모델 성능 평가 척도

▪ Mean squared error(MSE)

- MSE, 평균 제곱 오차
- MSE는 회귀선과 모델 예측값 사이의 오차 (잔차, residual)를 사용
- 오차를 제곱한 값들의 평균

$$MSE = \frac{\sum_{i=1}^n (\hat{y} - y_i)^2}{n}$$

예제

오차: 0.5, 1.3, 1.6, 0.2, 2.9

$$MSE = \frac{0.5^2 + 1.3^2 + 1.6^2 + 0.2^2 + 2.9^2}{5} = 2.59$$

데이터셋 구성을 통한 검증 방법

❖ 모델 성능 평가 척도

▪ Root mean squared error

- MSE에서 구한 값에 루트를 적용한 값 오차를 제공한 값 들의 평균

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (\hat{y} - y_i)^2}{n}}$$

예제

오차: 0.5, 1.3, 1.6, 0.2, 2.9

$$MSE = \frac{0.5^2 + 1.3^2 + 1.6^2 + 0.2^2 + 2.9^2}{5} = 2.59$$

$$RMSE = \sqrt{2.59} = 1.6$$

데이터셋 구성을 통한 검증 방법

❖ 모델 성능 평가 척도

- **Confusion matrix**

- 모델의 성능을 통계적인 수치로 시각화 하는 방법

- **Accuracy**

- 모델이 정확하게 분류 또는 예측하는 데이터의 비율

- **Precision**

- 모델이 검출한 데이터 중 올바르게 검출된 데이터의 비율

- **Recall**

- 실제 해당 데이터중 모델이 올바르게 검출한 데이터의 비율

- **F-Measure**

- Precision과 recall을 통합해 나타내는 정확도

데이터셋 구성을 통한 검증 방법

❖ 모델 성능 평가 척도

- 모델의 예측결과와 성능을 살펴볼 수 있는 척도

		예측값	
		Class = yes	Class = no
실제값	Class = yes	a (TP : true positive)	b (FN : false negative)
	Class = no	c (FP : false positive)	d (TN : true negative)

데이터셋 구성을 통한 검증 방법

❖ 모델 성능 평가 척도

▪ 모델의 예측결과와 성능을 살펴볼 수 있는 척도

- 실제값: 데이터의 실제 카테고리 / 예측값: 모델이 분류, 예측한 데이터의 카테고리
- A, TP (True Positive): 실제**yes** 카테고리의 데이터 중 모델이 **yes** 카테고리로 예측한 데이터의 건 수
- B, FN (False Negative): 실제**yes** 카테고리의 데이터 중 모델이 **no** 카테고리로 예측한 데이터의 건수
- C, FP (False Positive): 실제**no** 카테고리의 데이터 중 모델이 **yes** 카테고리로 예측한 데이터의 건수
- D, TN (True Negative): 실제**no** 카테고리의 데이터 중 모델이 **no** 카테고리로 예측한 데이터의 건수

		예측값	
		Class = yes	Class = no
실제값	Class = yes	a (TP : true positive)	b (FN : false negative)
	Class = no	c (FP : false positive)	d (TN : true negative)

데이터셋 구성을 통한 검증 방법

❖ 모델 성능 평가 척도

▪ Accuracy (정확도)

- 모델이 정확하게 분류 또는 예측하는 데이터의 비율
- TP: 참 긍정개수 TN: 참 부정개수 FP: 거짓 긍정개수 FN : 거짓 부정개수

$$\text{Accuracy} = \frac{a + d}{a + b + c + d} = \frac{TP + TN}{TP + TN + FP + FN}$$

▪ Accuracy의 한계

- 2 class 문제에서 class yes에 해당하는 데이터는 9900건, class no에 해당하는 데이터는 100건이 존재할 경우
- 모델이 모든 데이터를 class yes로 예측할 경우에도
즉, class no를 예측하지 못했음에도 불구하고 accuracy는 $9900/10000=99\%$ 가 됨
- 이러한 경우가 있어서 accuracy 모델은 성능을 측정하는 척도로 적합하지 않음

데이터셋 구성을 통한 검증 방법

❖ 모델 성능 평가 척도

▪ Precision (정밀도)

- 모델이 검출한 데이터 중 올바르게 검출된 데이터의 비율
- 예시) 악성종양을 악성으로 예측한 결과
- TP: 참 긍정개수 TN: 참 부정개수 FP: 거짓 긍정개수 FN : 거짓 부정개수

$$\text{Precision (p)} = \frac{a}{a + c}$$

▪ Recall (재현율)

- 실제 해당 데이터 중 모델이 올바르게 검출한 데이터의 비율
- 예시) 전체 악성종양을 실제 예측한 결과

$$\text{Recall (r)} = \frac{a}{a + b}$$

		예측값	
		Class = yes	Class = no
실제값	Class = yes	a (TP : true positive)	b (FN : false negative)
	Class = no	c (FP : false positive)	d (TN : true negative)

데이터셋 구성을 통한 검증 방법

❖ 모델 성능 평가 척도

▪ F-Measure

- Precision과 recall은 모델의 성능을 객관적으로 판단하기에 부족
- 두 수치의 trade-off 관계를 통합해 하나의 수치로 정확도 도출

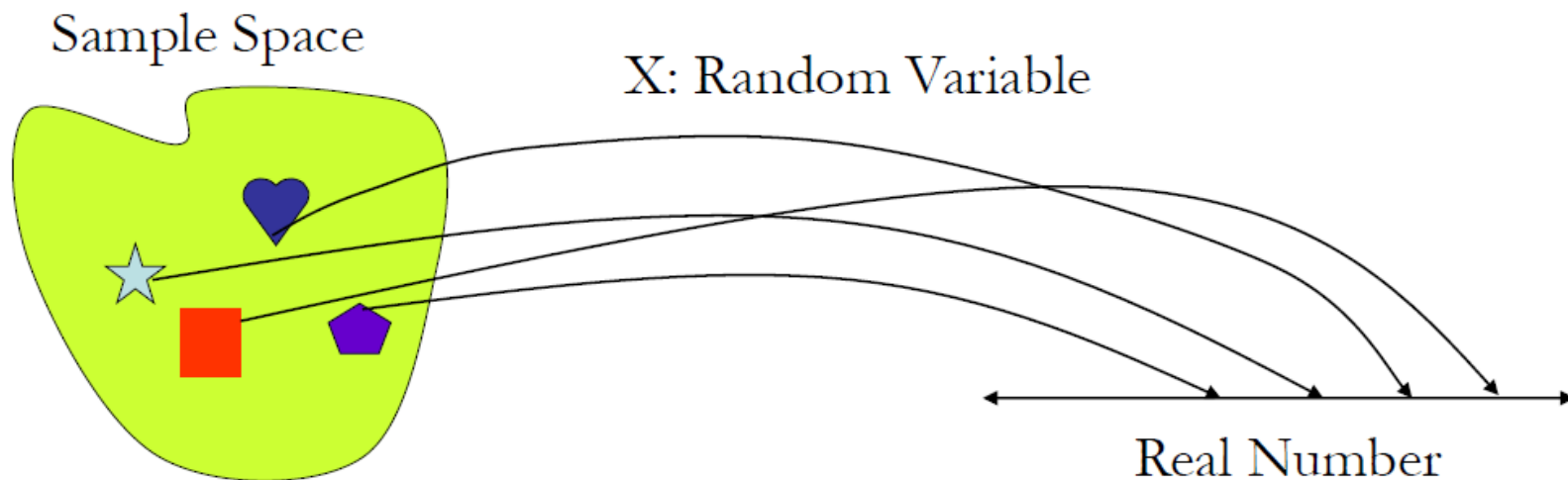
		예측값	
		Class = yes	Class = no
실제값	Class = yes	a (TP : true positive)	b (FN : false negative)
	Class = no	c (FP : false positive)	d (TN : true negative)

$$F - \text{measure} (F) = \frac{2rp}{r + p} = \frac{2a}{2a + b + c}$$

통계 리뷰

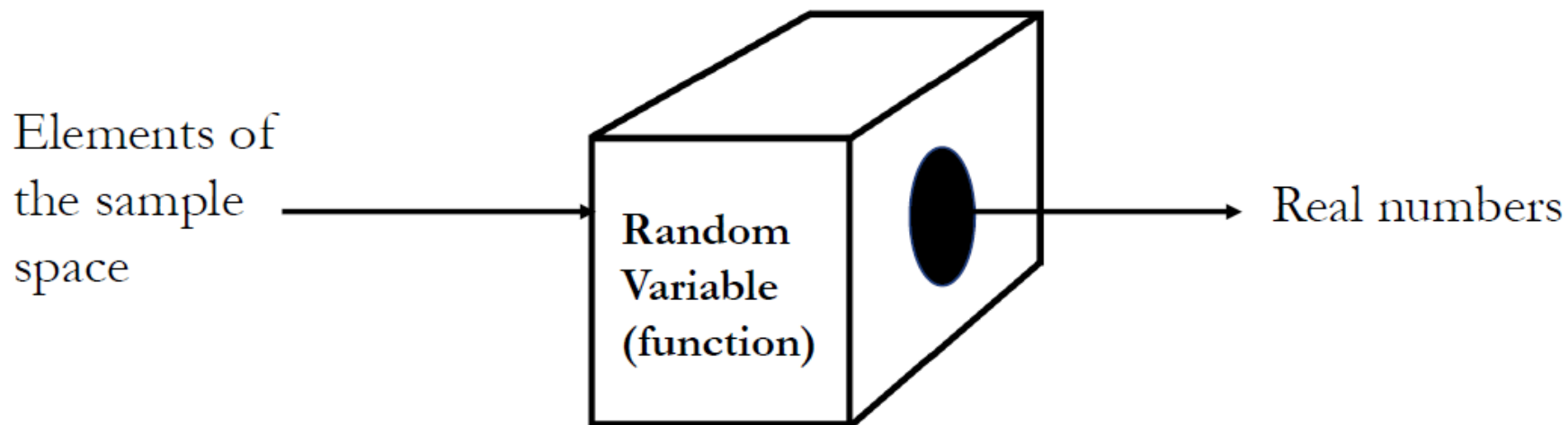
❖ 확률변수(Random Variable)

Random variable is a function that assigns real number to each element of the sample space.



통계 리뷰

❖ 확률변수(Random Variable)



$$\text{Real numbers} = f(\text{Elements of the sample space})$$

통계 리뷰

❖ 확률변수(Random Variable)

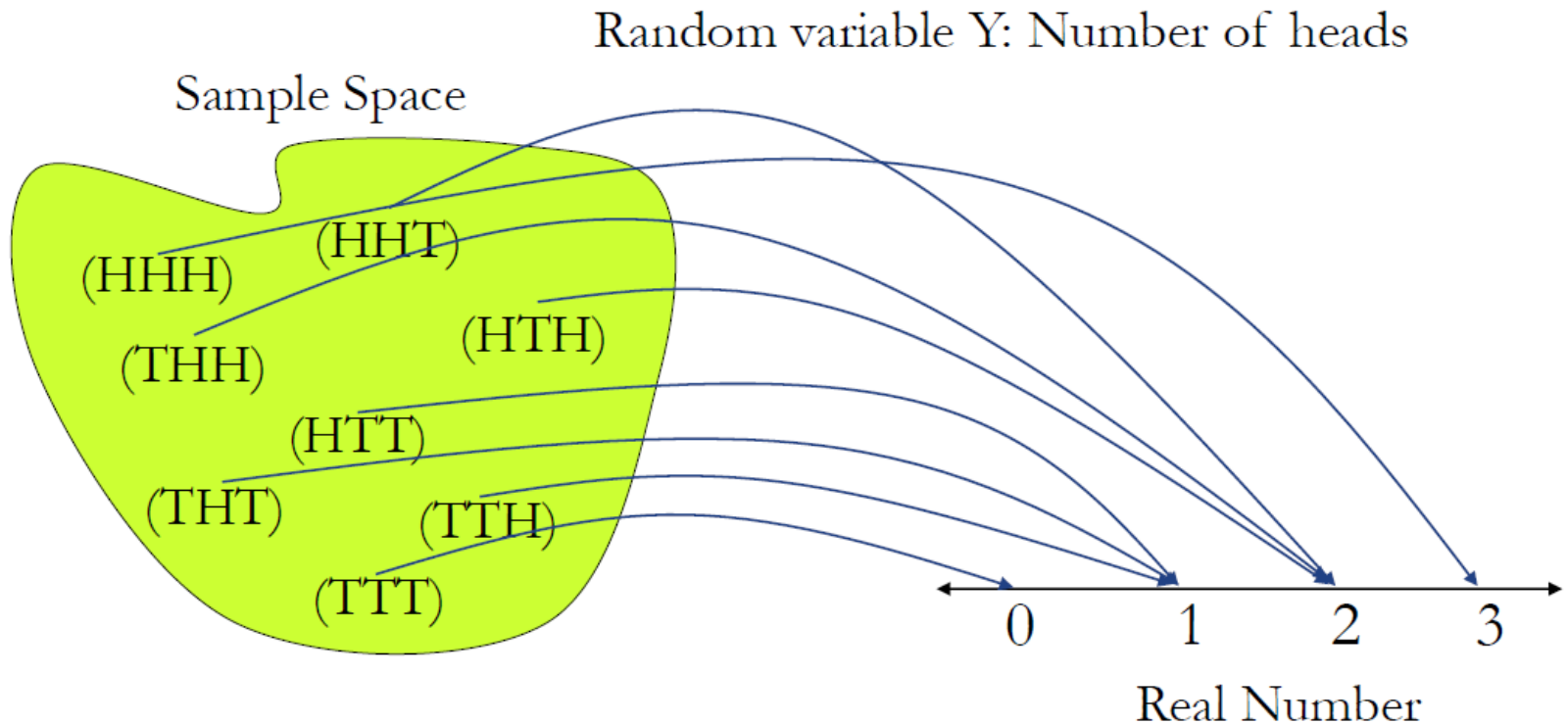
- Flip 3 coins
- Sample space = { (HHH), HHT), HTH), (THH), HTT),(THT), (TTH), (TTT) }
- Define Y =number of heads
- Random variable: Y (why?)
- $Y=\{0, 1, 2, 3\}$

Y	outcomes
0	TTT
1	HTT, THT, TTH
2	HHT, HTH, THH
3	HHH

통계 리뷰

❖ 확률변수(Random Variable)

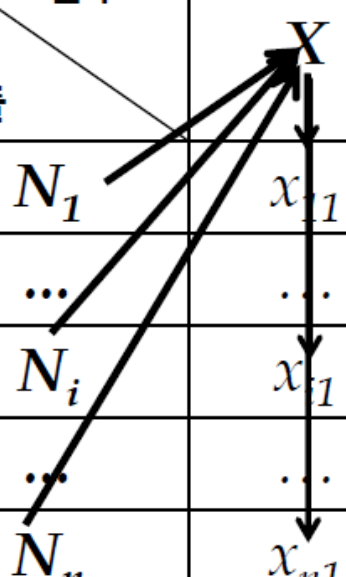
▪ Flip 3 coins



통계 리뷰

❖ 확률변수(Random Variable)

변수 \ 샘플	
N_1	x_{11}
...	...
N_i	x_{i1}
...	...
N_n	x_{n1}



변수 \ 샘플	X_1	...	X_p
N_1	x_{11}	...	x_{1p}
...
N_i	x_{i1}	...	x_{ip}
...
N_n	x_{n1}	...	x_{np}

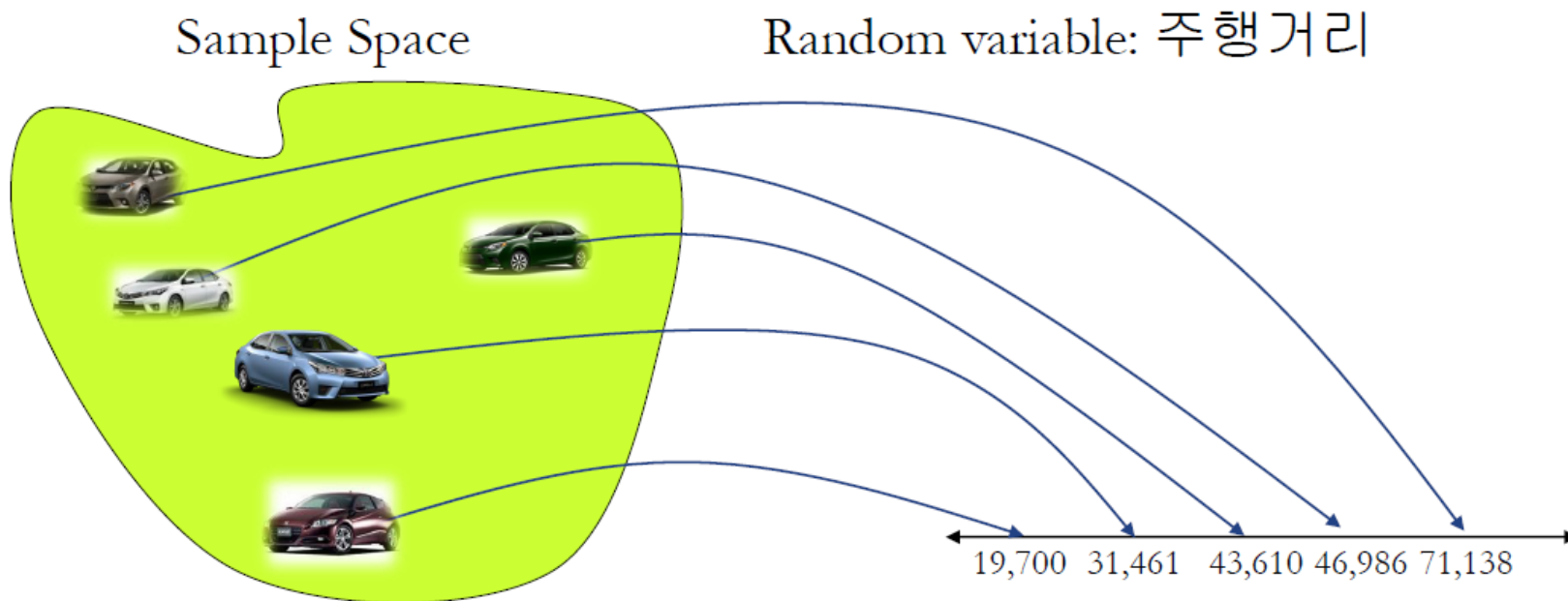
Real numbers = f (Elements of the sample space)

↓
 X

통계 리뷰

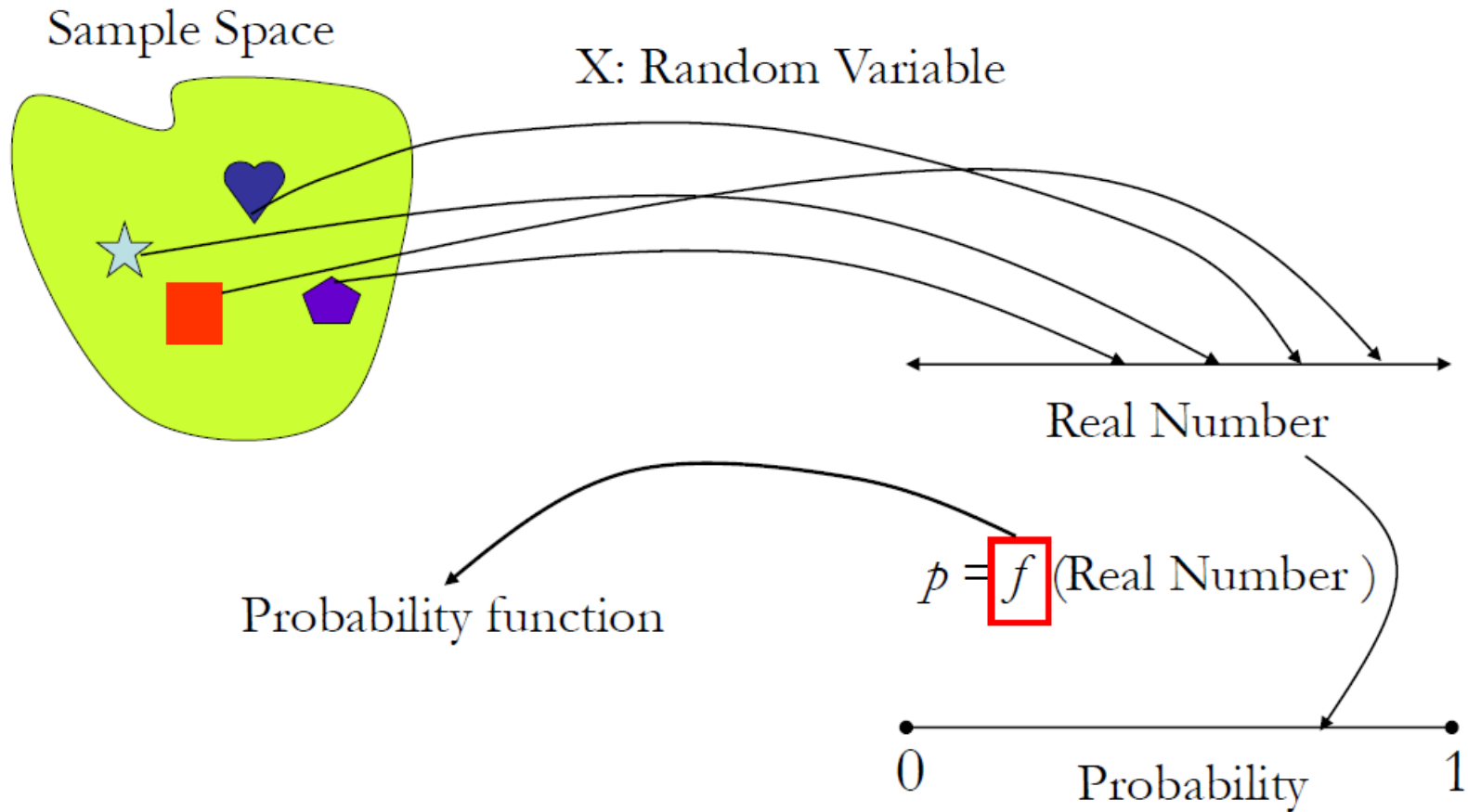
❖ 확률변수(Random Variable)

No	모델	주행거리	마력	용량	가격
1	TOYOTA Corolla 2.0D4D HATCHB TERRA 2/3-Doors	46,986	90	2,000	13,500
2	TOYOTA Corolla 1800T SPORT VVTI 2/3-Doors	19,700	192	1,800	21,500
3	TOYOTA Corolla 1.9D HATCHB TERRA 2/3-Doors	71,138	69	1,900	12,950
4	TOYOTA Corolla 1.8 VVTL-i T-Sport 3-Drs 2/3-Doors	31,461	192	1,800	20,950
5	TOYOTA Corolla 1.8 16V VVTLI 3DR T SPORT BNS 2/3-Doors	43,610	192	1,800	19,950



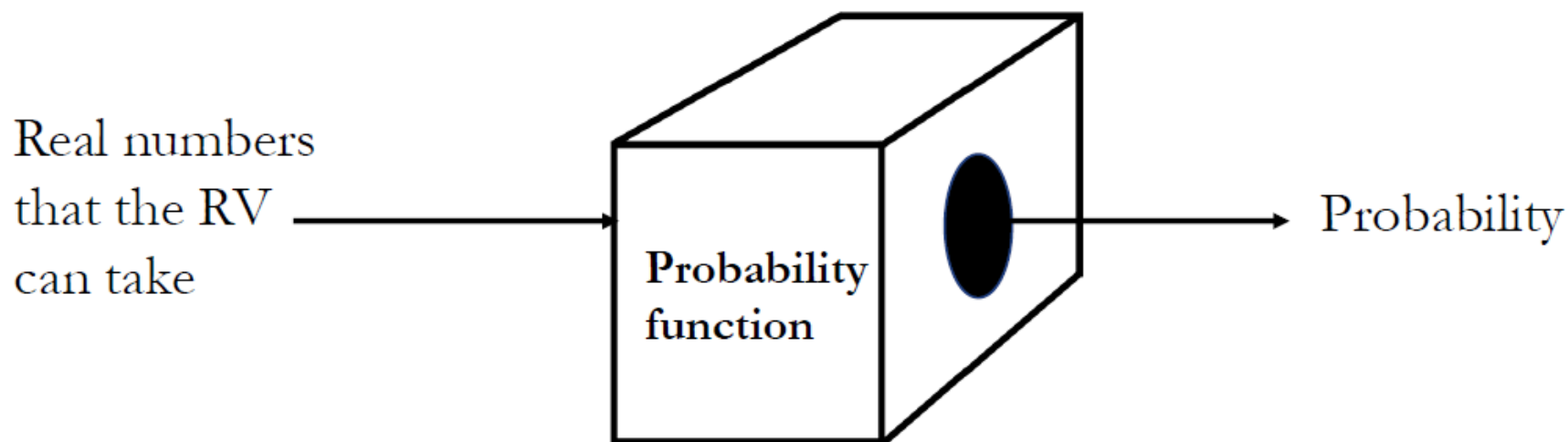
통계 리뷰

❖ 확률함수 (Probability



통계 리뷰

❖ 확률함수 (Probability



$$p = f(\text{Real Number})$$

통계 리뷰

❖ 확률함수 (Probability)

$$p = f \text{ (Real Number)}$$

$$p = f \text{ (Discrete)}$$



Probability mass function
(pmf)

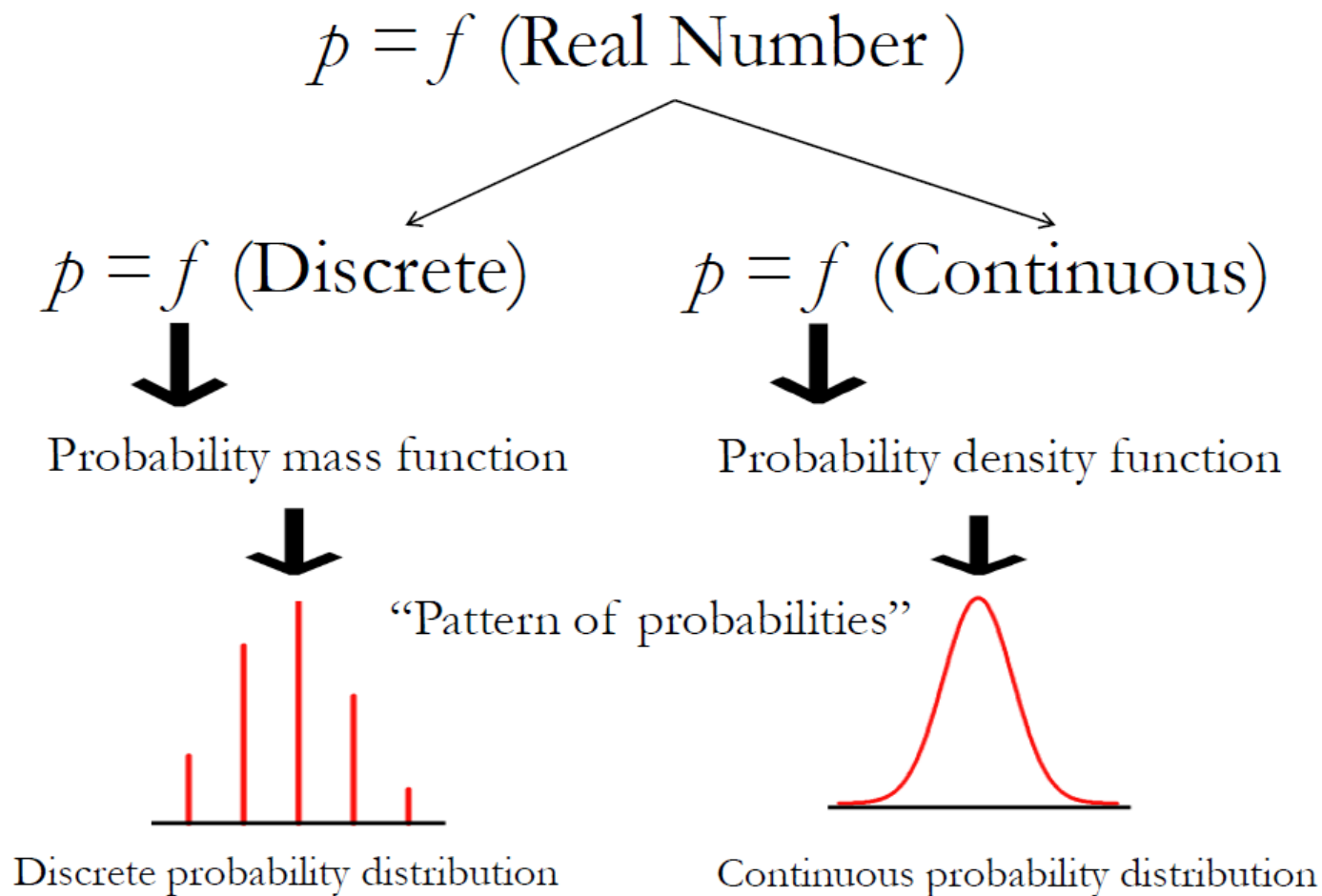
$$p = f \text{ (Continuous)}$$



Probability density function
(pdf)

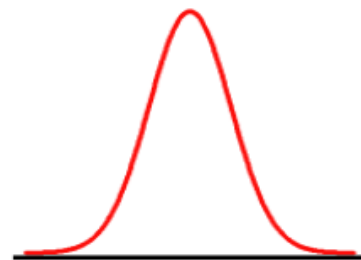
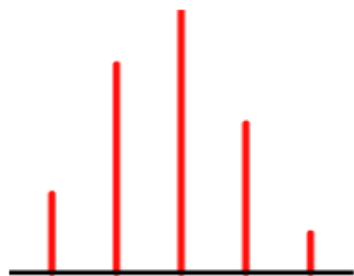
통계 리뷰

❖ 확률함수 (Probability



통계 리뷰

❖ 확률함수 (Probability



The Bernoulli distribution

The Binomial distribution

The Hypergeometric distribution

The Geometric distribution

The Poisson distribution

Uniform distribution

Normal distribution

Exponential distribution

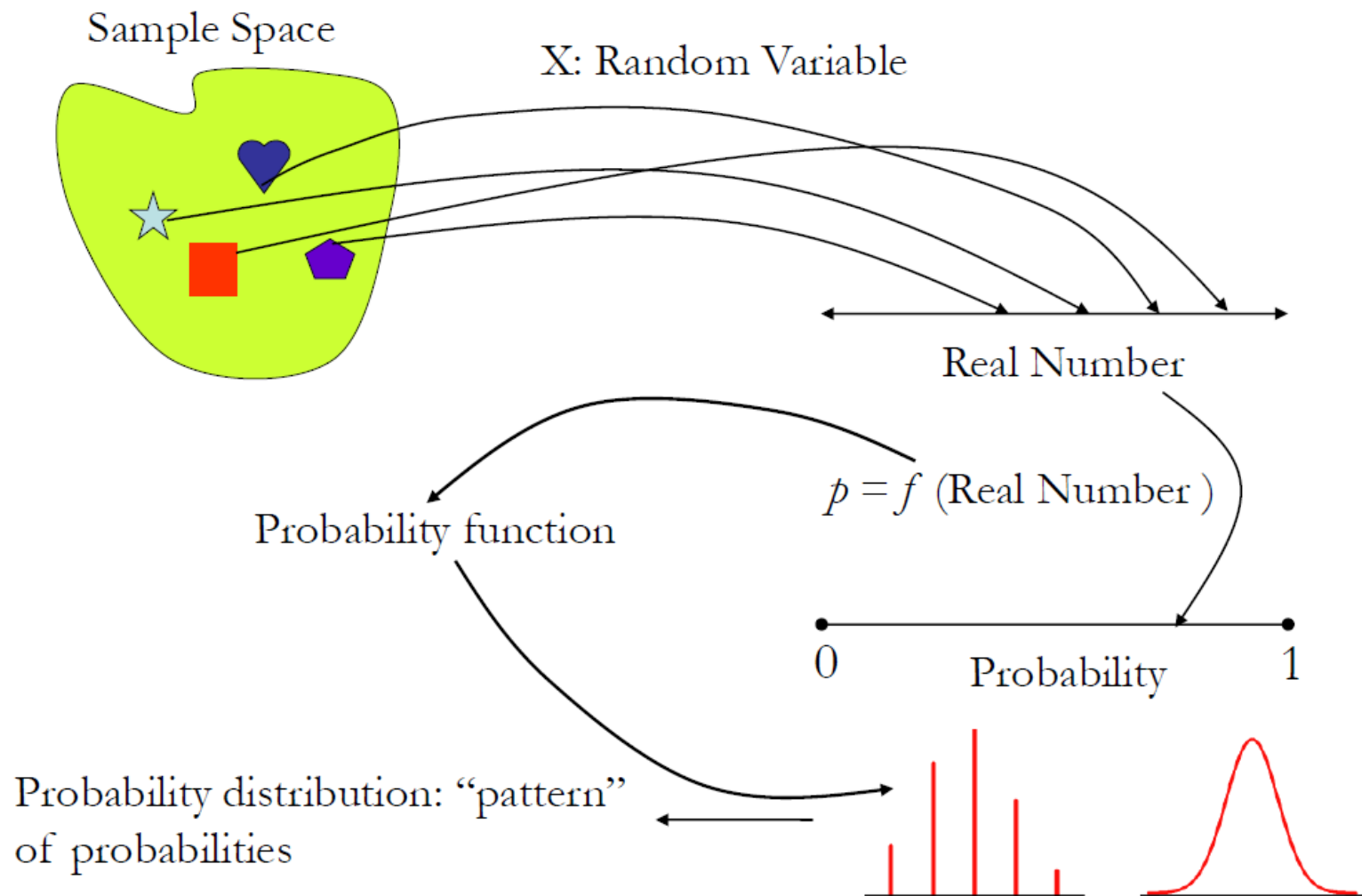
Gamma distribution

Beta distribution

Chi-squared distribution

통계 리뷰

❖ 표본공간 → 확률변수 → 확률함수 → 확률분포



통계 리뷰

❖ 주요통계 용어 리뷰

- 표본공간 (Sample Space): 실험으로 부터 얻은 결과값들의 집합
- 실험 (Experiment): 데이터를 생성하는 모든 과정
- 확률변수 (Random Variable): 표본공간의 요소를 실수로 대응시키는 함수 법칙
- 확률함수 (Probability Function): 확률값을 생성하는 함수
- 확률분포 (Probability Distribution): 확률함수로 부터 나온 확률 값들의 패턴
- 통계량 (Statistic): 샘플로 구성된 함수
- 추정 (Estimation): 알려지지 않은 확률분포의 모수를 추정하는 프로세스
- 검정 (Hypothesis Testing): 알려지지 않은 확률분포의 모수를 검정하는 프로세스
- P value: 귀무가설 (Null 이 참일 가능성을 확률로 표현한 척도

감사합니다