데이터분석을위한 PANDAS

데이터 분석 이해

데이터 분석 이해

◆ 데이터 분석이란?

유용한 **정보**를 **발굴**하고 **결론** 내용을 알리며

의사결정을 지원하는 것을 **목표**로

데이터를 정리, 변환, 모델링하는 과정이다.

과거의 데이터를 토대로 미래를 분석한다.

데이터 분석 이해

◆ 데이터 분석 접근 방법

❖ 확증적 분석(CDA: Confirmatory Data Analysis)

→ **전통적 분석 기법**으로 **추론 통계**에 주로 사용

→ **가설 설정 후 수집 데이터로 가설 평가/추정**

→ **기존 연구 기반**으로 수행되되 **엄격한 절차와 방법**

→ 장점 ➔ **검증된 이론/모형** 갖추고 있으며 **구체적인 질문/답 도출 가능**

→ 단점 ➔ 선입견 개입되어 **예상치 못한 결과의 사전 탐지 어려움**

데이터 분석 이해

◆ 데이터 분석 접근 방법

❖ 확증적 분석(CDA: Confirmatory Data Analysis)

가설 설정	데이터 수집	통계 분석	가설 검증
CCTV 설치하면 범죄 예방 효과 있다.	• 지역별 CCTV 설치 현황 데이터 • 지역별 범죄발생 기록 데이터	• CCTV 설치 지역과 범죄 발생 빈도 상관관계 • CCTV 설치 전후 범죄발생 변화율	분석결과 통해 가설 채택 또는 가설 기각

데이터 분석 이해

◆ 데이터 분석 접근 방법

❖ 탐색적 분석(EDA: Exploratory Data Analysis)

→ 귀납적 분석 기법으로 기술 통계에 주로 사용

→ 시각화 기법을 통해 데이터 특징, 구조로부터 통찰 얻는 기법

→ 장점 ➔ 선입견 없이 **유연하게 데이터 분석하며 가설 설정 가능**

→ 단점 ➔ 명확한 분석 **목표 없으면 방황할 가능성 높음**

데이터 분석 이해

◆ 데이터 분석 접근 방법

❖ 탐색적 분석(EDA: Exploratory Data Analysis)

데이터 수집	시각화/탐색	패턴 도출	인사이트 발견
• 서울 지역별, 시기별 배달 음식 주문 건수 기록 데이터 확보	• 시각과 지역 변화에 따른 주문별 변화를 다양한 관점으로 시각화/탐색	• 시각화 자료로부터 음식별/시기별/지역별 일정한 패턴 있음 발견	• 시기와 지역별 주문이 많은 배달 음식 예상 • **창업에 활용**

데이터 분석 이해

◆ EDA 5 단계

1 문제정의

- 분석 대상 이해
- 객관적/구체적 대상 정의

2 데이터 수집

- 필요 데이터 요건 정의
- 데이터 소재 파악 및 확보

3 데이터 전처리

- 오류 사항 점검/조치
- 데이터 구조 및 특성 변경

4 데이터 모델링

- 단순 모델 시도·특징 중요도 분석·가설 검증 등

5 시각화 및 탐색

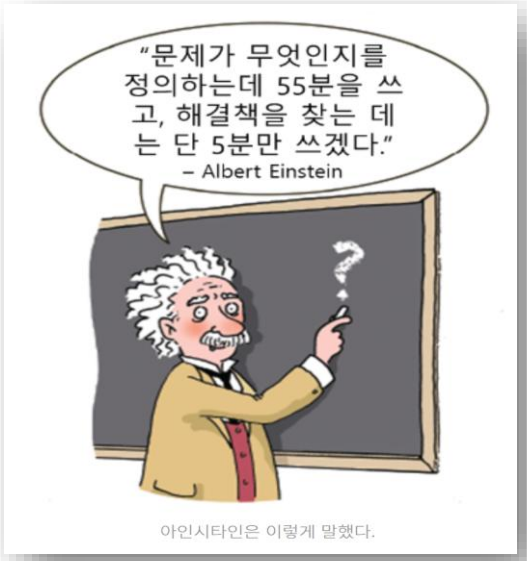
- 다양한 유형 시각화
- 분포, 상관관계, 패턴 확인
- 인사이트 도출

◆ EDA 5 단계

1 문제 정의

▪ **가장 중요 & 가장 어려운 단계**

- 많은 사람들 공감할 만한 가치 있는 문제
- 문제 해결위한 구체적인 행동 수반
- 데이터 제약사항
- 데이터 분석 전문가 및 기간 확보



데이터 분석 이해

◆ EDA 5 단계

2 데이터 수집

- **주변 >> 온라인 >> 오프라인**
  - 나의 PC / 나의 그룹 & 회사
  - 온라인 & 오프라인
  - 데이터 제공 기관 및 집단
    - → 공공 : 공공데이터 포털, 통계빅데이터서비스, 한국복지패널
    - → 지도 : 국가 공간정보 포털
    - → 기상 : 기상자료개방포털
    - → 관광 : 관광데이터랩
    - → 건축 : 건축데이터 민간 개방 시스템, 국가공간정보포털, 등기정보광장

◆ EDA 5 단계

3 데이터 전처리

▪ **가장 많은 수고 및 시간 소요**

- 수집 데이터를 그대로 사용하는 경우 거의 없음
  - → 결측치 처리
  - → 중복 처리
  - → 이상치 처리
  - → 데이터 연계 / 통합
  - → 데이터 구조 변경

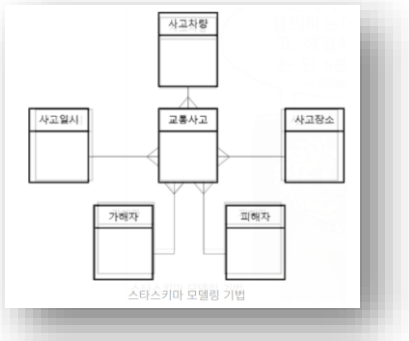
데이터 분석 이해

◆ EDA 5 단계

4 데이터 모델링

▪ **관점별 데이터 분류 및 관계 설정**

- 한 개의 핵심 사실(Fact)와 여러 개의 추가적인 사실로 구성
- **분석 대상 나누어 그 결과를 사실과 추가 사실로 구성**하는 것
- **데이터 설계과정**으로 **도식화 표현**
- 모델링 예시)



◆ EDA 5 단계

5 시각화 및 탐색

▪ **패턴 찾고 인사이트 도출**

- 문제에 대한 **답을 찾는 단계**
- 대량 데이터 요약, 사람이 판단하기 쉬운 형태의 **이미지로 표현**
- 데이터에 숨겨진 **유의미한 인사이트(Insite) 발견**할 수 있도록 도움
- 데이터 요약 설명 방법 ➔ **기술 통계(Descriptive Statistics)**
  → 수집 데이터 **요약, 묘사, 설명**
  → **대표값(중심경향 : 평균, 중앙값, 최빈값 등)과 분포** 이용 설명

데이터 분석 이해

◆ 데이터 분류

- 데이터 종류에 따른 분석 방법 설정
- 데이터 분석 시 제일 먼저 진행되어야 하는 분류

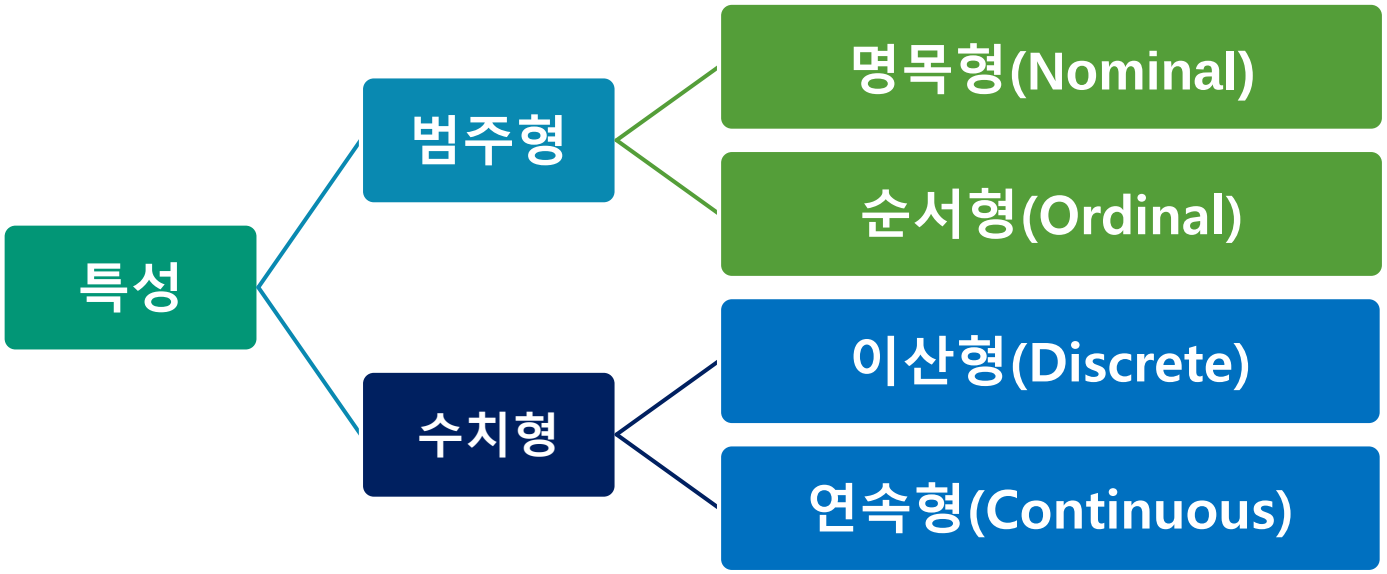
특성에 따른 분류

개수에 따른 분류

데이터 분석 이해

◆ 데이터 분류

- **특성**에 따른 분류



데이터 분석 이해

◆ 데이터 분류

▪ 특성에 따른 분류

범주형
(Categorical Data)

- **질적(Qualitative data : 정성적 자료)** 라고도 함
- **수치 측정 불가능**
- 성별, 혈액형, 종교, 순위 등 범주 또는 그룹으로 구분
- 산술 연산 분석 불가능

→ **명목형(Nominal)** 단순 분류 위해 사용되는 순서 없는 데이터
성별, 혈액형 등

→ **순서형(Ordinal)** 서열이나 순위를 나타낼 수 있는 데이터
학점, 만족도

데이터 분석 이해

◆ 데이터 분류

▪ 특성에 따른 분류

수치형
(Numerical Data)

- **양적(Quantitative data : 정량적 자료)**
- **수치 측정 가능**
- 온도, 가격, 주가지수, 실업률, 매출액, 키 등 숫자로 구성
- 대소 비교, 평균, 최댓값, 최솟값 등 **산술연산 분석 가능**

→ **이산형(Discrete)** 셀 수 있지만 특정 구간이 존재하는 형태
년도, 월

→ **연속형(Continuous)** 값 사이 무수히 많은 연속적인 값가진 형태
키, 몸무게 등

데이터 분석 이해

◆ 데이터 분류

▪ 개수에 따른 분류

변수 (Variable Data)	• **연구 · 조사 · 관찰하고 싶은 대상** • 예)키, 몸무게, 혈액형, 매출액, 습도, 먼지 농도 등등....
단일변수 자료 (Univariate Data)	• 연구 · 조사 · 관찰하고 싶은 대상 **1개로만 구성** • **일변량 자료**라고도 함
다중변수 자료 (Multivariate Data)	• 연구 · 조사 · 관찰하고 싶은 대상 **2개 이상으로 구성** • **다변량 자료**라도도 함

데이터 분석 이해

◆ 데이터 분류

▪ 목적에 따른 분류

피쳐/특성/속성/데이터/독립
Independent Variable
Explanatory Variable
Predictor Variable

- 부가 데이터 변수
- 원인 변수 / 예측 변수
- **다른 변수에 영향을 주고, 영향 받지 않는 변수**

목적/타겟/라벨/정답/종속
Dependent Variable
Response Variable
Outcome Variable

- 데이터 **분석에 목적이 되는 변수**
- 측정되는 결과 변수/ 반응 변수
- 다른 변수에 **영향을 받는 변수**

PANDAS 데이터 확인

PANDAS 데이터 확인

◆ 데이터 정보 확인

- **전체 데이터 구조 확인**
- 열 단위 데이터의 **산술 통계 값 확인**
- 데이터의 형태, 크기(개수) 확인
- 데이터 분석을 위한 **기초 정확 수집**

PANDAS 데이터 확인

◆ 데이터 정보 확인

▪ 정보 함수

앞부분 데이터 보기 : DataFrame 객체. **head (n) 기본 5개**

뒷부분 데이터 보기 : DataFrame 객체. **tail (n)**

데이터 기본 정보 보기 : DataFrame 객체. **info()**

데이터 기술 통계정보 : DataFrame 객체. **describe()**

데이터 기술 통계정보 : DataFrame 객체. **describe(include ='all')**

PANDAS 데이터 확인

◆ 데이터 정보 확인

▪ 데이터 개수

✓ **전체 열별 데이터 개수 반환**

- DataFrame 객체.**count( )**

✓ **데이터별 데이터 개수 반환**

- DataFrame객체[열이름].**value_counts( )**
  - → dropna = False (기본값)
  - → 결측치(Missing value , NaN) 포함 여부

PANDAS 데이터 확인

◆ 데이터 정보 확인

▪ 데이터 종류

✓ **고유값**

- 중복되지 않은 유일한 값들

✓ **메서드**

- DataFrame/Series 객체.**nique( )** 고유한 값들 반환
- DataFrame/Series 객체.**unuique( )** 고유한 값 개수 반환

PANDAS 데이터 확인

◆ 데이터 정보 확인

▪ 데이터 종류

```
## 모듈 로딩
import pandas as pd

## 데이터 준비
numSR = pd.Series([2, 1, 3, 3], name='A')

## 데이터 출력
print("numSR", numSR, sep='₩n', end='₩n₩n')

## 고유값
print("고유값", numSR.unique( ), end='₩n₩n')

## 고유값 수
print("고유값 개수", numSR.nunique(), end='₩n₩n')
```

```
numSR
0    2
1    1
2    3
3    3
Name: A, dtype: int64

고유값 [2 1 3]

고유값 개수 3
```

PANDAS 데이터 확인

◆ 데이터 통계 개념

▪ 값 사이 분포

- **편차 (Deviation)**

  관측값에서 평균(mean) 뺀 값, 평균에서 얼마나 떨어진 값인지 확인

- **분산(Variance)**

  편차 제곱한 값. 평균에서 얼마나 떨어진 값인지 확인

- **표준편차(Standard deviation)**

  분산 값에 제곱근(√루트) 취해 제곱 보정

PANDAS 데이터 확인

◆ 데이터 통계 개념

- 값 사이 분포

A **1, 3, 5, 7, 9**

편차	-4 , -2, 0 , 2 , 4
편차의 제곱	16 , 4, 0 , 4 , 16
분산 (편차의 제곱의 평균)	$(16+4+0+4+16) \div 5 = 8$
표준 편차 (분산의 제곱근)	$\sqrt{8}$

B **3, 4, 5, 6, 7**

편차	-2 , -1, 0 , 1 , 2
편차의 제곱	4 , 1, 0 , 1 , 4
분산 (편차의 제곱의 평균)	$(4+1+0+1+4) \div 5 = 2$
표준 편차 (분산의 제곱근)	$\sqrt{2}$

PANDAS 데이터 확인

◆ 데이터 통계 개념

▪ 대표값

- **평균 (mean)**

  모든 관측값의 합을 자료의 개수로 나눈 값

- **중앙(간)값(median)**

  전체 관측값을 크기 순서로 정렬했을 때 가운데 위치한 값

  → 홀수 경우: (n+1) /2 번째

  → 짝수 경우: (n/2)번째 관측값과 (n+1) /2번째 관측값의 평균

- **최빈값 (mode)**

  데이터 중 가장 많은 빈도로 나타나는 값

PANDAS 데이터 확인

◆ 데이터 통계 개념

▪ 분포(Distribution)

- **데이터**는 언뜻 보면 아무 의미 없는것 같지만, 만약 **어떤 분포를 따르는 잘 알려진 현상(상황)이라면 이는 분포를 통해 다른 현상들을 해석하거나 예측할 수 있게 되는 것**

- 대표적인 분포 종류
  - 이산 확률 분포 ➔ 이항분포, 베르누아 분포, 기하분포, 푸아송 분포
  - 연속 확률 분포 ➔ 정규(Z)분포, T분포, F 분포, χ2분포

PANDAS 데이터 확인

◆ 데이터 통계 개념

▪ 정규분포(Normal Distribution) : 수학

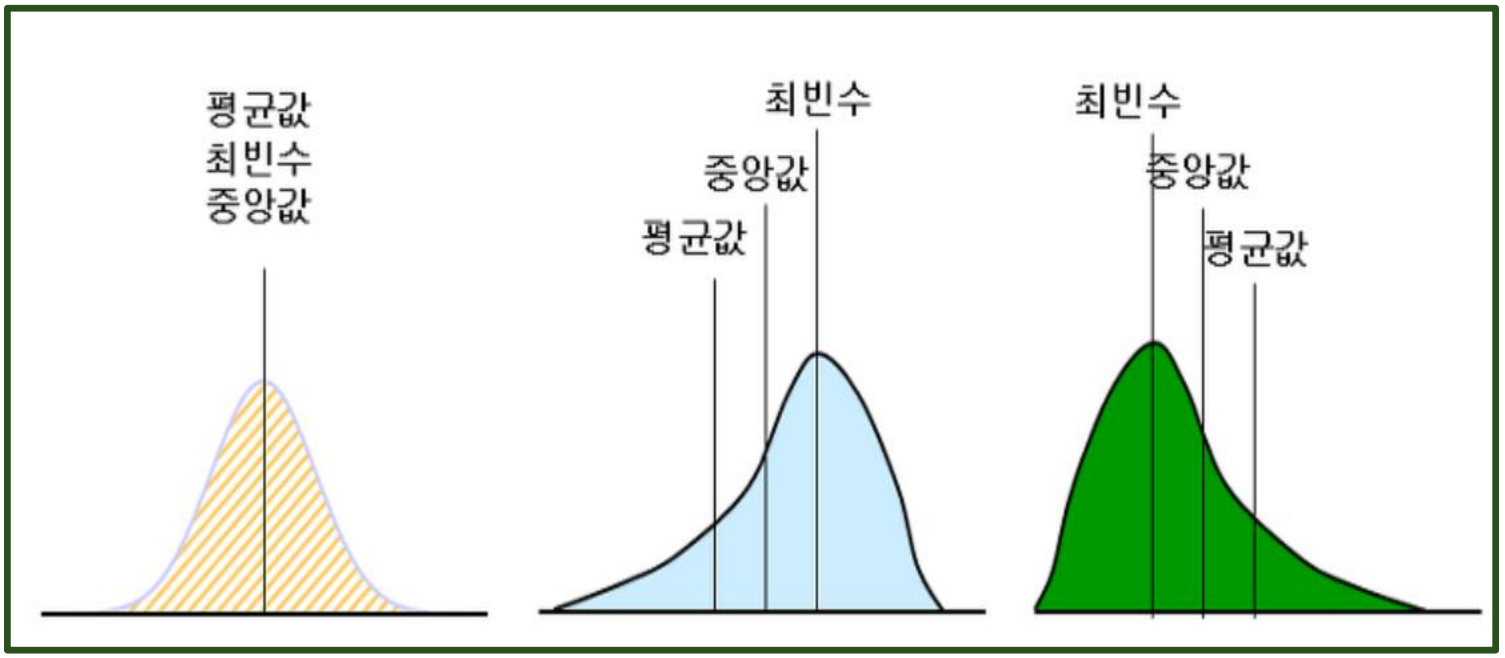
- **공학**에서는 **가우시안(Gaussian) 분포**라고도 함
- **수집된 자료의 분포를 근사하는 데에 자주 사용**
- 중심극한정리에 의하여 독립적인 확률변수들의 평균은 정규분포에 가까워지는 성질이 있기 때문

- 평균, 중앙값 일치 : 평균, 최빈값 일치/평균 중심으로 좌우 대칭
- **평균 == 최빈값 == 중앙값**

PANDAS 데이터 확인

◆ 데이터 통계 개념

- **분포(Distribution)**



PANDAS 데이터 확인

◆ 데이터 통계 개념

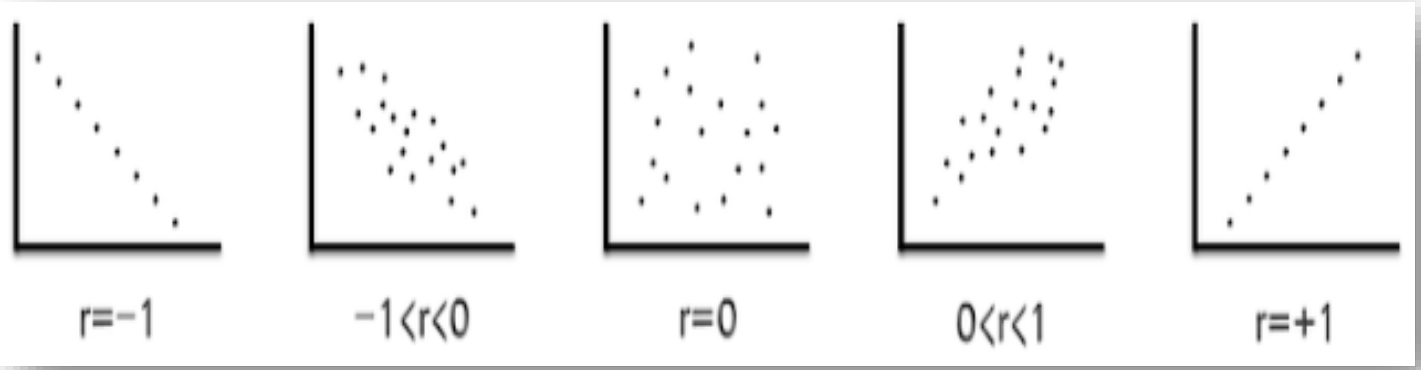
▪ 값들의 관계

▪ 상관계수(correlation coefficient)

두 열(column) 사이 관계 정도를 나타내는 수치

→ 범위 : -1 ~ 1 사이

→ -1(음의 상관관계), 1(양의 상관관계)에 가까울 수록 관계 밀접



PANDAS 데이터 확인

◆ 데이터 통계 개념

▪ 결측값 Missing Value

- 측정되지 않았거나 **누락된 값, 입력되지 않은 값**
- 분석 **정확성 떨어짐 / 분석 결과 왜곡** 될 수 있음

- 처리 방법 : 제거, 치환 & 대체, 특별한 범주 간주

- 표기방식
  - → NaN Not a Number 약자. 수치 결측값
  - → NaT Not a Time 약자. 시간/날짜 결측값
  - → NA Not Available. 누락 데이터
  - → None 파이썬에서 누락 데이터
  - → null NA와 동일 의미

PANDAS 데이터 확인

◆ 데이터 통계 개념

▪ 이상치/특이값 Outlier

- 정상적인 데이터 **분포 범위 밖에 존재하는 값**
- 논리적으로 존재할 수 없는 값
- 값에 큰 영향을 미쳐 **데이터의 분포 왜곡 및 성능 저하**

- 처리 방법
  삭제 / 변환(영향 ▼) / 대체 / 모델링 / 분석 목적 따른 판단

PANDAS 데이터 확인

◆ 데이터 통계 개념

▪ 통계 / 집계 함수

→ 모든 열 평균값 = DataFrame 객체.**mean()**
→ 특정 열 평균값 = DataFrame 객체**[열이름]. mean()**

→ 모든 열 중간값 = DataFrame 객체.**median()**
→ 특정 열 중간값 = DataFrame 객체**[열이름]. median()**

→ 모든 열 표준편차 = DataFrame 객체.**std()**
→ 특정 열 표준편차 = DataFrame 객체**[열이름]. std()**

PANDAS 데이터 확인

◆ 데이터 통계 개념

▪ 통계 / 집계 함수

→ 모든 열 최대값 = DataFrame 객체.**max()**
→ 특정 열 최대값 = DataFrame 객체**[열이름]. max()**

→ 모든 열 최소값 = DataFrame 객체.**min()**
→ 특정 열 최소값 = DataFrame 객체**[열이름]. min()**

→ 모든 열 상관계수 = DataFrame 객체.**corr()**
→ 특정 열 상관계수 = DataFrame 객체**[열이름].corr()**

PANDAS 데이터 전처리

PANDAS 데이터 전처리

◆ 전처리(Preprocessing)

데이터 품질을 높이기 위한 과정

분석 목적에 맞게 변형하는 과정

많은 시간 소요, 필수 과정

- **데이터 처리 =>** 누락 데이터, 중복 데이터, 이상 데이터
- **데이터 변환 =>** 단위 표준화, 자료형 표준화
- **데이터 크기 =>** 정규화

◆ 데이터 처리 – 결측치

▪ 결측치 데이터 관련 메서드

- **결측 데이터 체크**
  - → bool = 객체.**isnull( )** : null이면 True
  - → bool = 객체.**isna( )** : na이면 True

- **정상 데이터 체크**
  - → bool = 객체.**notnull( )** : null이면 False
  - → bool = 객체.**notna( )** : na이면 False

PANDAS 데이터 전처리

◆ 데이터 처리 – 결측치

▪ 체크 메서드

객체.**isnull( )**

	survived	pclass	sex	age	...	deck	embark_town	alive	alone
0	False	False	False	False	...	True	False	False	False
1	False	False	False	False	...	False	False	False	False
2	False	False	False	False	...	True	False	False	False
3	False	False	False	False	...	False	False	False	False
4	False	False	False	False	...	True	False	False	False

PANDAS 데이터 전처리

◆ 데이터 처리 – 결측치

▪ 체크 메서드

객체.**isnull( ).sum( )** False 0, True 1로 계산

	survived	pclass	sex	age	...	deck	embark_town	alive	alone
0	False	False	False	False	...	True	False	False	False
1	False	False	False	False	...	False	False	False	False

	survived	pclass	sex	age	...	deck	embark_town	alive	alone
0	0	0	0	177	...	688	2	0	0

PANDAS 데이터 전처리

◆ 데이터 처리 – 결측치

▪ 체크 메서드

```
## DF 생성
df = pd.DataFrame( { 'age' : [ 5, 6, np.nan ],
                     'born': [ pd.NaT,
                               pd.Timestamp('1939-05-27'),
                               pd.Timestamp('1940-04-25')],
                     'name' : ['Alfred', 'Batman', ''],
                     'toy'     : [None, 'Batmobile', 'Joker']})

## DF 출력
print( df )
```

	age	born	name	toy
0	5.0	NaT	Alfred	None
1	6.0	1939-05-27	Batman	Batmobile
2	NaN	1940-04-25		Joker

PANDAS 데이터 전처리

◆ 데이터 처리 – 결측치

▪ 체크 메서드

```
## 결측치 체크
df.isna( )
```

	age	born	name	toy
0	False	True	False	True
1	False	False	False	False
2	True	False	False	False

	age	born	name	toy
0	5.0	NaT	Alfred	None
1	6.0	1939-05-27	Batman	Batmobile
2	NaN	1940-04-25		Joker

◆ 데이터 처리 – 결측치

▪ 처리 방법 ① 제거

객체.**dropna( )** **결측값 존재 행/열 삭제**

- 매개변수(Parameter)
  - → axis = 0 행(row) 삭제
  - → axis = 1 열(column) 삭제
  - → how = 'all' 모든 값 결측치인 경우 삭제
  - → how = 'any' 하나라도 결측치인 경우 삭제 (기본)
  - → thresh = 숫자 Non-NA개 이상, 임계치 설정
  - → subset = 조건 하위 조건

PANDAS 데이터 전처리

◆ 데이터 처리 – 결측치

▪ 처리 방법 ① 제거

```
## DF 생성
df = pd.DataFrame({"name": ['Alfred', 'Batman', 'Catwoman'],
                   "toy"   : [np.nan, 'Batmobile', 'Bullwhip'],
                   "born" : [pd.NaT, pd.Timestamp("1940-04-25"), pd.NaT]})

## DF 출력
print( df )
```

	name	toy	born
0	Alfred	NaN	NaT
1	Batman	Batmobile	1940-04-25
2	Catwoman	Bullwhip	NaT

PANDAS 데이터 전처리

◆ 데이터 처리 – 결측치

▪ 처리 방법 ① 제거

```
## axis : 결측치 검사를 기준 축 설정
##        0, 'index'   – 행
##        1, 'columns' – 열
df.dropna(axis='index')   <= 기본값
```

	name	toy	born
0	Alfred	NaN	NaT
1	Batman	Batmobile	1940-04-25
2	Catwoman	Bullwhip	NaT

	name	toy	born
1	Batman	Batmobile	1940-04-25

```
df.dropna( axis='columns' )
```

	name	toy	born
0	Alfred	NaN	NaT
1	Batman	Batmobile	1940-04-25
2	Catwoman	Bullwhip	NaT

	name
0	Alfred
1	Batman
2	Catwoman

◆ 데이터 처리 – 결측치

▪ 처리 방법 ① 제거

```
## how   : 제거 방식/기준 설정
##          'all'   - 모든 속성이 결측치인 행
##          'any'  - 1개 이상 속성이 결측치인 행
df.dropna( how='all' )
```

	name	toy	born
0	Alfred	NaN	NaT
1	Batman	Batmobile	1940-04-25
2	Catwoman	Bullwhip	NaT

	name	toy	born
0	Alfred	NaN	NaT
1	Batman	Batmobile	1940-04-25
2	Catwoman	Bullwhip	NaT

PANDAS 데이터 전처리

◆ 데이터 처리 – 결측치

▪ 처리 방법 ① 제거

```
## thresh : 결측치가 아닌 데이터  최소 개수 설정
##          how와 함께 사용 불가
df.dropna( thresh=2 )
```

	name	toy	born
1	Batman	Batmobile	1940-04-25
2	Catwoman	Bullwhip	NaT

```
df.dropna( thresh=1 )
```

	name	toy	born
0	Alfred	NaN	NaT
1	Batman	Batmobile	1940-04-25
2	Catwoman	Bullwhip	NaT

◆ 데이터 처리 – 결측치

▪ 처리 방법 ① 제거

subset : 특정 컬럼 또는 행만 결측치 검사하도록 설정

```
df.dropna( subset=['name'] )
```

	name	toy	born
0	Alfred	NaN	NaT
1	Batman	Batmobile	1940-04-25
2	Catwoman	Bullwhip	NaT

```
df.dropna( subset=['name', 'toy'] )
```

	name	toy	born
1	Batman	Batmobile	1940-04-25
2	Catwoman	Bullwhip	NaT

◆ 데이터 처리 – 결측치

▪ 처리 방법 ② 치환/대체

객체.**fillna(** 치환값 **)** **특정값, 평균값, 최빈값으로 치환**

- 매개변수(Parameter) : **value**
  - → fillna(0) : 모두 동일 값
  - → fillna({ 컬럼: 0, 컬럼: 1, … , 컬럼: N }) : 컬럼별 다른 값
  - → fillna(평균값)
  - → fillna(최빈값)
  - → fillna(중앙값)

◆ 데이터 처리 – 결측치

▪ 처리 방법 ② 치환/대체

객체.**fillna(** 치환값 **)** **특정값, 평균값, 최빈값으로 치환**

- 매개변수(Parameter) : **method**
  - → 'ffill' 직전 행(row) 값으로 치환
  - → 'bfill' / 'backfill' 바로 다음 행(row) 값으로 치환

- 매개변수(Parameter) : **limit**
  - → int 지정된 수 만큼 치환
  - → None 모두 치환

◆ 데이터 처리 – 결측치

▪ 처리 방법 ② 치환/대체

```
## DF 생성
df = pd.DataFrame( [ [np.nan, 2, np.nan, 0],
                     [3, 4, np.nan, 1],
                     [np.nan, np.nan, np.nan, np.nan],
                     [np.nan, 3, np.nan, 4]],
                     columns=list("ABCD"))

## DF 출력
print( df )
```

	A	B	C	D
0	NaN	2.0	NaN	0.0
1	3.0	4.0	NaN	1.0
2	NaN	NaN	NaN	NaN
3	NaN	3.0	NaN	4.0

◆ 데이터 처리 – 결측치

▪ 처리 방법 ② 치환/대체

```
## value : NaN를 대체할 값 설정
## 모든 NaN 동일 값으로 채우기
df.fillna(value=0 )
```

	A	B	C	D
0	NaN	2.0	NaN	0.0
1	3.0	4.0	NaN	1.0
2	NaN	NaN	NaN	NaN
3	NaN	3.0	NaN	4.0

	A	B	C	D
0	0.0	2.0	0.0	0.0
1	3.0	4.0	0.0	1.0
2	0.0	0.0	0.0	0.0
3	0.0	3.0	0.0	4.0

```
## 컬럼별 NaN 대체값 설정
values = {"A": 0, "B": 1, "C": 2, "D": 3}
df.fillna( value=values )
```

	A	B	C	D
0	NaN	2.0	NaN	0.0
1	3.0	4.0	NaN	1.0
2	NaN	NaN	NaN	NaN
3	NaN	3.0	NaN	4.0

	A	B	C	D
0	0.0	2.0	2.0	0.0
1	3.0	4.0	2.0	1.0
2	0.0	1.0	2.0	3.0
3	0.0	3.0	2.0	4.0

PANDAS 데이터 전처리

◆ 데이터 처리 – 결측치

▪ 처리 방법 ② 치환/대체

```
## value  : NaN를 대체할 값 설정
## limit  : 대체값 채울 개수 설정

df.fillna( value=values, limit=1 )
```

	A	B	C	D
0	NaN	2.0	NaN	0.0
1	3.0	4.0	NaN	1.0
2	NaN	NaN	NaN	NaN
3	NaN	3.0	NaN	4.0

	A	B	C	D
0	0.0	2.0	2.0	0.0
1	3.0	4.0	NaN	1.0
2	NaN	1.0	NaN	3.0
3	NaN	3.0	NaN	4.0

PANDAS 데이터 전처리

◆ 데이터 처리 – 결측치

▪ 처리 방법 ② 치환/대체

```
## method : 이전 값들로 채우기 설정
##          'ffill'          - 이전 값으로 채우기
##          'backfill','bfill' - 다음 값으로 채우기
df.fillna( method='ffill' )
```

	A	B	C	D
0	NaN	2.0	NaN	0.0
1	3.0	4.0	NaN	1.0
2	NaN	NaN	NaN	NaN
3	NaN	3.0	NaN	4.0

	A	B	C	D
0	NaN	2.0	NaN	0.0
1	3.0	4.0	NaN	1.0
2	3.0	4.0	NaN	1.0
3	3.0	3.0	NaN	4.0

```
df.fillna( method='backfill' )
```

	A	B	C	D
0	NaN	2.0	NaN	0.0
1	3.0	4.0	NaN	1.0
2	NaN	NaN	NaN	NaN
3	NaN	3.0	NaN	4.0

	A	B	C	D
0	3.0	2.0	NaN	0.0
1	3.0	4.0	NaN	1.0
2	NaN	3.0	NaN	4.0
3	NaN	3.0	NaN	4.0

◆ 데이터 처리 – 결측치

▪ 처리 방법 ② 치환/대체

객체.**interpolate(** 보간법 **)**

✓ **보간법**

두 개 이상의 알고 있는 데이터 포인트들 사이에 예상되는 값들 추정 기법

✓ **적용 예**

→ 데이터가 누락되었을 때

→ 시계열 데이터에서 연속적인 추세 유지하고 싶을 때

→ 그래프, 곡선, 음성 신호 등에서 매끄러운 변화 표현이 필요할 때

→ 머신러닝 학습 시 결측값 처리 위한 전처리

PANDAS 데이터 전처리

◆ 데이터 처리 – 결측치

▪ 처리 방법 ② 치환/대체

객체.**interpolate(** 보간법 **)**

- **보간법**

  두 개 이상

  - 적용

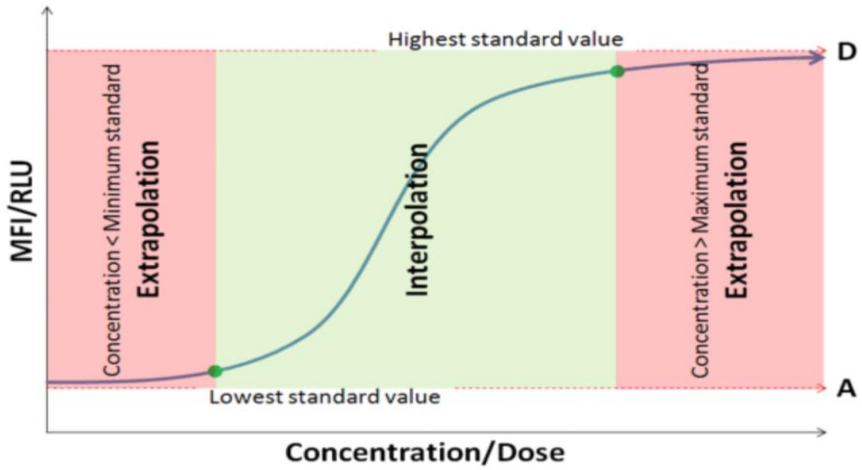
  데이터가

  시계열

  그래프,

  머신러닝

정 기법



◆ 데이터 처리

▪ 중복 데이터

- **하나의 행(row)은 모든 속성값 존재하는 완벽한 데이터셋**
- **모든 속성값 동일한 행/열 ==> 중복 데이터**

- 검사 함수 : 객체.**duplicated( )** 중복이면 True / 아니면 False
- 제거 함수 : 객체.**drop_duplicates( )** 중복 행 제거

◆ 데이터 처리 – 중복

▪ 체크 메서드

객체.**duplicated( )**

- 매개변수(Parameter)
  - → subset = None　　중복 체크 위한 부분 설정 지정
  - → keep = 'first'　　중복 데이터에서 첫 번째 행 유지

PANDAS 데이터 전처리

◆ 데이터 처리 – 중복

▪ 체크 메서드

```
## 모듈로딩
import pandas as pd

## 데이터 준비
df = pd.DataFrame( { 'brand' : [ 'Yum', 'Yum', 'Indo', 'Indo', 'Indo' ],
                     'style'  : [ 'cup', 'cup', 'cup', 'pack', 'pack' ],
                     'rating' : [4, 4, 3.5, 15, 5] })

## 중복체크
result = df.duplicated().sum()

print(f'중복 데이터/행 : {result}개')
```

	brand	style	rating
0	Yum	cup	4.0
1	Yum	cup	4.0
2	Indo	cup	3.5
3	Indo	pack	15.0
4	Indo	pack	5.0

중복

PANDAS 데이터 전처리

◆ 데이터 처리 – 중복

▪ 체크 메서드

```
## 중복체크 - 첫 번째 유지, 나머지 중복
result = df.duplicated( )
```

```
0    False
1     True
2    False
3    False
4    False
dtype: bool
```

```
## 중복체크 - 마지막 유지, 그 외 중복
result = df.duplicated( keep='last' )
```

```
0     True
1    False
2    False
3    False
4    False
dtype: bool
```

```
## 중복체크 - 모두 중복
result = df.duplicated( keep=False )
```

```
0     True
1     True
2    False
3    False
4    False
dtype: bool
```

PANDAS 데이터 전처리

◆ 데이터 처리 – 중복

▪ 체크 메서드

```
## 중복체크 - brand 컬럼만 중복인 행
result = df.duplicated( subset='brand' )
```

		brand	style	rating
중복	0	Yum	cup	4.0
	1	Yum	cup	4.0
중복	2	Indo	cup	3.5
	3	Indo	pack	15.0
	4	Indo	pack	5.0

0	False	first
1	True	
2	False	first
3	True	
4	True	

dtype: bool

PANDAS 데이터 전처리

◆ 데이터 처리 – 중복

▪ 체크 메서드

```
## 중복체크 - brand, style 컬럼이 중복인 행
result = df.duplicated( subset=['brand','style'] )
```

중복

중복

	brand	style	rating
0	Yum	cup	4.0
1	Yum	cup	4.0
2	Indo	cup	3.5
3	Indo	pack	15.0
4	Indo	pack	5.0

first

first

```
0    False
1     True
2    False
3    False
4     True
dtype: bool
```

◆ 데이터 처리 – 중복

▪ 처리 방법 ① 제거

객체. **drop_duplicates( )**

- 매개변수(Parameter)
  - → subset = None　　삭제 위한 부분 설정 지정
  - → keep = 'first'　　중복 데이터에서 첫 번째 행 유지
  - → inplace = False　　원본 데이터 유지

PANDAS 데이터 전처리

◆ 데이터 처리 – 중복

▪ 처리 방법 ① 제거

```
## 모듈로딩
import pandas as pd

## 데이터 준비
df = pd.DataFrame( { 'brand' : [ 'Yum', 'Yum', 'Indo', 'Indo', 'Indo' ],
                     'style'  : [ 'cup', 'cup', 'cup', 'pack', 'pack' ],
                     'rating' : [4, 4, 3.5, 15, 5] })

## 중복체크
result = df.duplicated().sum()

print(f'중복 데이터/행 : {result}개')
```

#	brand	style	rating
# 0	Yum	cup	4.0
# 1	Yum	cup	4.0
# 2	Indo	cup	3.5
# 3	Indo	pack	15.0
# 4	Indo	pack	5.0

◆ 데이터 처리 – 중복

▪ 처리 방법 ① 제거

```
## 중복데이터 삭제 - 모든 속성값이 동일한 행 비교
df.drop_duplicates( keep= 'first' )
```

중복

	brand	style	rating
0	Yum	cup	4.0
1	Yum	cup	4.0
2	Indo	cup	3.5
3	Indo	pack	15.0
4	Indo	pack	5.0

첫번째 유지

	brand	style	rating	
0	Yum	cup	4.0	first
2	Indo	cup	3.5	
3	Indo	pack	15.0	
4	Indo	pack	5.0	

◆ 데이터 처리 – 중복

▪ 처리 방법 ① 제거

```
## 중복데이터 삭제 - 모든 속성값이 동일한 행 비교
df.drop_duplicates( keep= 'last' )
```

중복

	brand	style	rating
0	Yum	cup	4.0
1	Yum	cup	4.0
2	Indo	cup	3.5
3	Indo	pack	15.0
4	Indo	pack	5.0

마지막 유지

	brand	style	rating
1	Yum	cup	4.0
2	Indo	cup	3.5
3	Indo	pack	15.0
4	Indo	pack	5.0

last

◆ 데이터 처리 – 중복

▪ 처리 방법 ① 제거

```
## 중복데이터 삭제 - 모든 속성값이 동일한 행 비교
df.drop_duplicates( keep=False )
```

중복

	brand	style	rating
0	Yum	cup	4.0
1	Yum	cup	4.0
2	Indo	cup	3.5
3	Indo	pack	15.0
4	Indo	pack	5.0

모두 삭제

	brand	style	rating
2	Indo	cup	3.5
3	Indo	pack	15.0
4	Indo	pack	5.0

◆ 데이터 처리 – 중복

▪ 처리 방법 ① 제거

```
## 중복데이터 삭제 - brand 컬럼만 동일한 행 비교
df.drop_duplicates( subset='brand' )
```

		brand	style	rating
중복	0	Yum	cup	4.0
	1	Yum	cup	4.0
중복	2	Indo	cup	3.5
	3	Indo	pack	15.0
	4	Indo	pack	5.0

	brand	style	rating	
0	Yum	cup	4.0	first
2	Indo	cup	3.5	

◆ 데이터 처리 – 중복

▪ 처리 방법 ① 제거

```
## 중복데이터 삭제 - brand 컬럼만 동일한 행 비교
df.drop_duplicates( subset=['brand','style'] )
```

		brand	style	rating
중복	0	Yum	cup	4.0
	1	Yum	cup	4.0
	2	Indo	cup	3.5
중복	3	Indo	pack	15.0
	4	Indo	pack	5.0

	brand	style	rating	
0	Yum	cup	4.0	first
2	Indo	cup	3.5	
3	Indo	pack	15.0	first

PANDAS 데이터 전처리

◆ 데이터 처리 – 이상치

▪ 이상치(Outlier) 데이터

- **관측 데이터 범위에서 많이 벗어난 아주 작은 값/ 큰 값**
- 일반적인 **데이터 분포를 따르지 않는 값**

- **발생 원인**
  - → 데이터 수집 과정에서 오류 발생한 경우
  - → 데이터 자체에 이상치 포함한 경우 (오염된 데이터)
  - → 값을 잘못 옮겨적을 경우
  - → 실험 과정의 오류
  - → 의도적인 자료의 조작

◆ 데이터 처리 – 이상치

▪ 인식 및 처리 방법

✓ 이상치 인식 방법

→ ESD(Extreme Studentized Deviate) TEST

→ 기하평균(Geometric Mean)

→ 사분위수(IQR :Interquartile Range)

✓ 처리 방법

→ 삭제 (Delete) / 절단(Trimming)

→ 대체 (Replacement) / 조정(Winsorizing)

→ 축소/과장(Scaling) 적용

→ 최소초대척도 적용

→ 정규화 적용

◆ 데이터 처리 – 이상치

▪ 인식 방법 ① ESD Test

✓ **ESD(Estreme Studentized Deviate : 극단 표준화 편차)**

→ 정규분포 따르는 단변량(univariate) 데이터 집합에서 하나 이상(outliers up to k)의 이상치 탐지하는 통계검정

→ 데이터가 정규분포 따른다면, 평균에서 ±3σ 이상 벗어난 값은 매우 드문 사건

→ 이 원리를 **유의수준(α)과 표본 크기(n)** 을 고려해 **수학적으로 정확한 임계값**으로 확장한 개념

→ 최대 k개의 이상치 가능성을 미리 정해두고 그 범위 내에서 탐색

→ 여러 개의 이상치가 존재할 수 있는 상황에서 유용

◆ 데이터 처리 – 이상치

▪ 인식 방법 ① ESD Test

① **가정 : 데이터가 정규분포를 근사한다**

→ 최대 k개의 이상치가 있을 수 있다는 상한을 사전에 결정 (예: 데이터 10%)

② **검정 통계량 계산 반복 ==> 최대 K수 확정**

→ 데이터 집합의 평균, 표준편차 계산

→ 가장 극단값 하나 제거한 나머지에 대한 평균, 표준편차 계산

③ **임계값(Critical Value) 계산**

→ t-분포를 이용해 통계적으로 허용 가능한 최대 거리 계산

④ **이상치 개수 결정**

→ 실제 거리 > 임계값이면 이상치로 판정

◆ 데이터 처리 – 이상치

▪ 인식 방법 ② 기하평균(Geometric Mean)

✓ **기하평균**

→ 여러 개의 양수 값을 곱한 후, 그 **곱의 n제곱근을 구한 값**

→ 비율, 성장률, 변동이 큰 수치 데이터의 평균 구할 때 사용

→ **산술평균보다 극단값의 영향을 덜 받는 장점**

→ **반드시 양수 값만 사용할 수 있음**

$$G = \sqrt[n]{x_1 \times x_2 \times \ldots \times x_n}$$

◆ 데이터 처리 – 이상치

▪ 인식 방법 ② 기하평균(Geometric Mean)

✓ **기하평균 이상치 탐지**

→ 로그 변환을 하면 큰 값의 영향이 줄어듦

→ 산술평균 대신 기하평균으로 중심 경향을 잡으면 극단적인 큰 값(이상치)을 덜 민감하게 다룰 수 있음

→ 비율 데이터, 주가, 성장률, 소득 분포 등 비대칭 분포(right-skewed) 데이터에서 유용

◆ 데이터 처리 – 이상치

▪ 인식 방법 ② 기하평균(Geometric Mean)

✓ **기하평균 이상치 탐지**

→ 데이터 로그 변환 .

→ 변환된 데이터의 평균과 표준편차 계산

→ 아래 식으로 이상치 판단 (k=2 또는 3)

$$\text{if } |\log(x_i) - \text{mean}(\log_x)| > k \times \text{std}(\log_x)$$

◆ 데이터 처리 – 이상치

▪ 인식 방법 ③ IQR(Interquartile Range)

✓ **사분위수(Quartile)**

→ Z-점수(Z-score) 사용하여 이상값을 탐지하는 통계적 방법

→ 평균으로부터 3표준편차 떨어진 값 이상치 인식

→ 데이터의 분포를 **4등분한 통계 지표**

→ 데이터의 퍼짐 정도(IQR)를 이용하기 때문에 아주 널리 쓰임

◆ 데이터 처리 – 이상치

▪ 인식 방법 ③ IQR(Interquartile Range)

✓ **사분위수(Quartile)**

→ 데이터를 오름차순으로 정렬했을 때, 전체를 **4등분**하는 기준점

구분	이름	의미	pandas 함수
Q1	제1사분위수 (25%)	하위 25% 지점	df['x'].quantile(0.25)
Q2	제2사분위수 (50%)	중앙값 (Median)	df['x'].median()
Q3	제3사분위수 (75%)	상위 25% 지점	df['x'].quantile(0.75)

◆ 데이터 처리 – 이상치

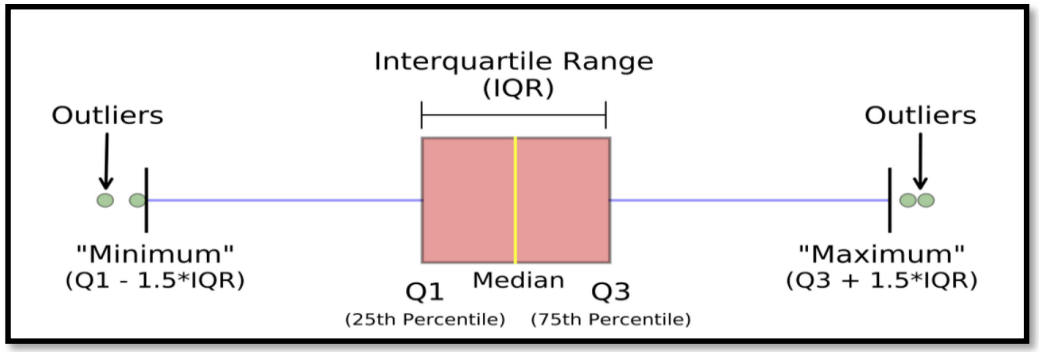
▪ 인식 방법 ③ IQR(Interquartile Range)

✓ **IQR(Inter-Quartile Range, 사분위 범위)**

→ 데이터의 **중간 50% 범위**

- IQR이 작으면 데이터가 중앙에 몰려 있음
- IQR이 크면 데이터가 퍼져 있음

→ IQR=Q3−Q1



◆ 데이터 처리 – 이상치

▪ 인식 방법 ③ IQR(Interquartile Range)

✓ **이상치 탐지에 사용하는 이유**

→ 평균보다 안정적 : 중앙 중심 통계라 영향이 적음

→ 분포 형태에 구애받지 않음 : 정규분포 아니더라도 사용 가능

→ 계산이 간단함 : Q1, Q3만 구하면 바로 적용

→ 시각화(박스플롯)와 연계 쉬움

◆ 데이터 처리 – 이상치

▪ 처리 방법 ① 삭제(Deleting) / 절단(Trimming)

✓ 극단적으로 크거나 작은 값 제거

✓ 주의!!

→ 극단적 값 중 유의미한 경우 많음

→ 삭제로 유의미한 데이터 손실 발생

- **다른 방법들 많이 사용**

◆ 데이터 처리 – 이상치

▪ 처리 방법 ② 조정(Winsorizing)

✓ **이상치 삭제(drop)하지 않고 상/하한값으로 조정(clip) 하는 방법**

✓ 데이터 특성에 따라 다른 기준으로도 수행

→ 백분위수(Percentile) 기반 Winsorizing 가장 일반적

→ IQR은 기초 통계 탐색 단계에서 가장 널리 사용

데이터 유형	추천 Winsorizing 기준
정규분포형(시험 점수, 센서 데이터)	• Z-score (±3σ)
비대칭 분포(가격, 소득, 매출)	• IQR 또는 Percentile (1~99%)
로그 스케일/비율형 데이터	• 로그변환 후 Z-score 또는 Percentile
현업 규칙이 명확할 때	• 도메인 기준값 (예: 나이 ≤ 120)

◆ 데이터 처리 – 이상치

▪ 처리 방법 ③ 축소/과장(Scaling)

✓ **데이터 스케일 자체 변환하여 이상치 영향 줄이거나 보정하는 방법**

✓ 핵심 효과

→ 이상치의 극단적인 영향 감소

→ 수치형 변수 간 균형 유지

→ 거리 기반 알고리즘(k-NN, SVM 등)에 유리

◆ 데이터 처리 – 이상치

▪ 처리 방법 ③ 축소/과장(Scaling)

방식	수식/설명	이상치 영향	사용 예시
① Standard Scaling (표준화)	$\frac{(x - \text{평균})}{\text{표준편차}}$	매우 민감	정규분포 가정 모델 (회귀, PCA 등)
② Min-Max Scaling (정규화)	$\frac{(x-\text{최소값})}{(\text{최대값}-\text{최소값})}$	매우 민감	신경망, 거리 기반 모델
③ Robust Scaling (강건 스케일링)	$\frac{(x - \text{Median})}{\text{IQR}}$	이상치에 강함	이상치 많은 데이터
④ Log Scaling (로그 변환)	log(x+1)	오른쪽 꼬리 완화	매출, 소득, 가격 등 비대칭 분포
⑤ Power / Box-Cox / Yeo-Johnson 변환	비선형 변환으로 분포 정규화	극단치 완화	통계적 모델링 전 정규화용

◆ 데이터 처리 – 이상치

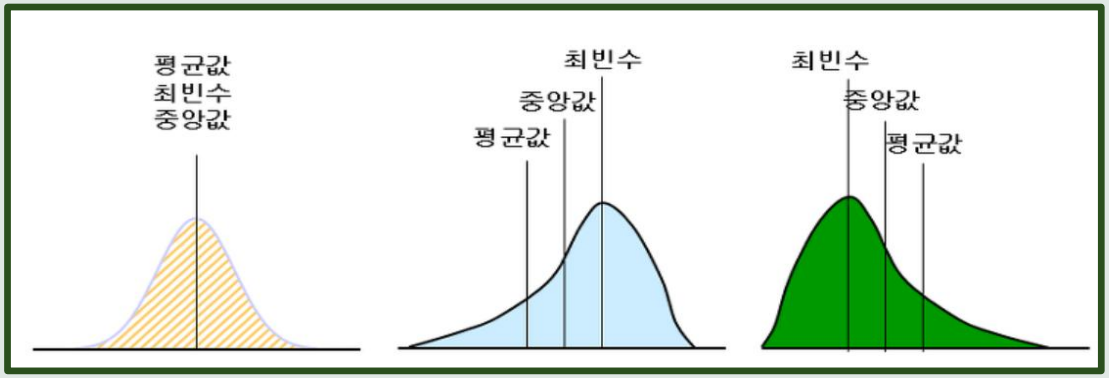
▪ 처리 방법 ③ 축소/과장(Scaling)

✓ **DataFrame['컬럼명'].skew()**

→ 양수 : 평균보다 작은 값 데이터 많음 최빈<중앙<평균

→ 0 : 평균 중심 고르게 분포 최빈==중앙==평균

→ 음수 : 평균보가 큰 값 데이터 많음 평균<중앙<최빈



◆ 데이터 처리 – 이상치

▪ 처리 방법 ③ 축소/과장(Scaling)

✓ **DataFrame['컬럼명'].skew()**

→ Positive/Right Skew >> 값 축소 >> skew() 값 0 근접
로그값 / 제곱근값 처리

→ Negative/Left Skew >> 값 과장 >> skew() 값 0 근접
제곱 / 지수곱 처리

◆ 데이터 처리 – 이상치

▪ 처리 방법 ④ 정규화 (Normalization)

✓ **데이터 값을 일정 범위(보통 0~1)로 맞추는 기법**

→ 데이터 크기를 조정하는 스케일링(Scaling)의 한 종류

→ 최대값 : 1, 최소값 : 0 변환

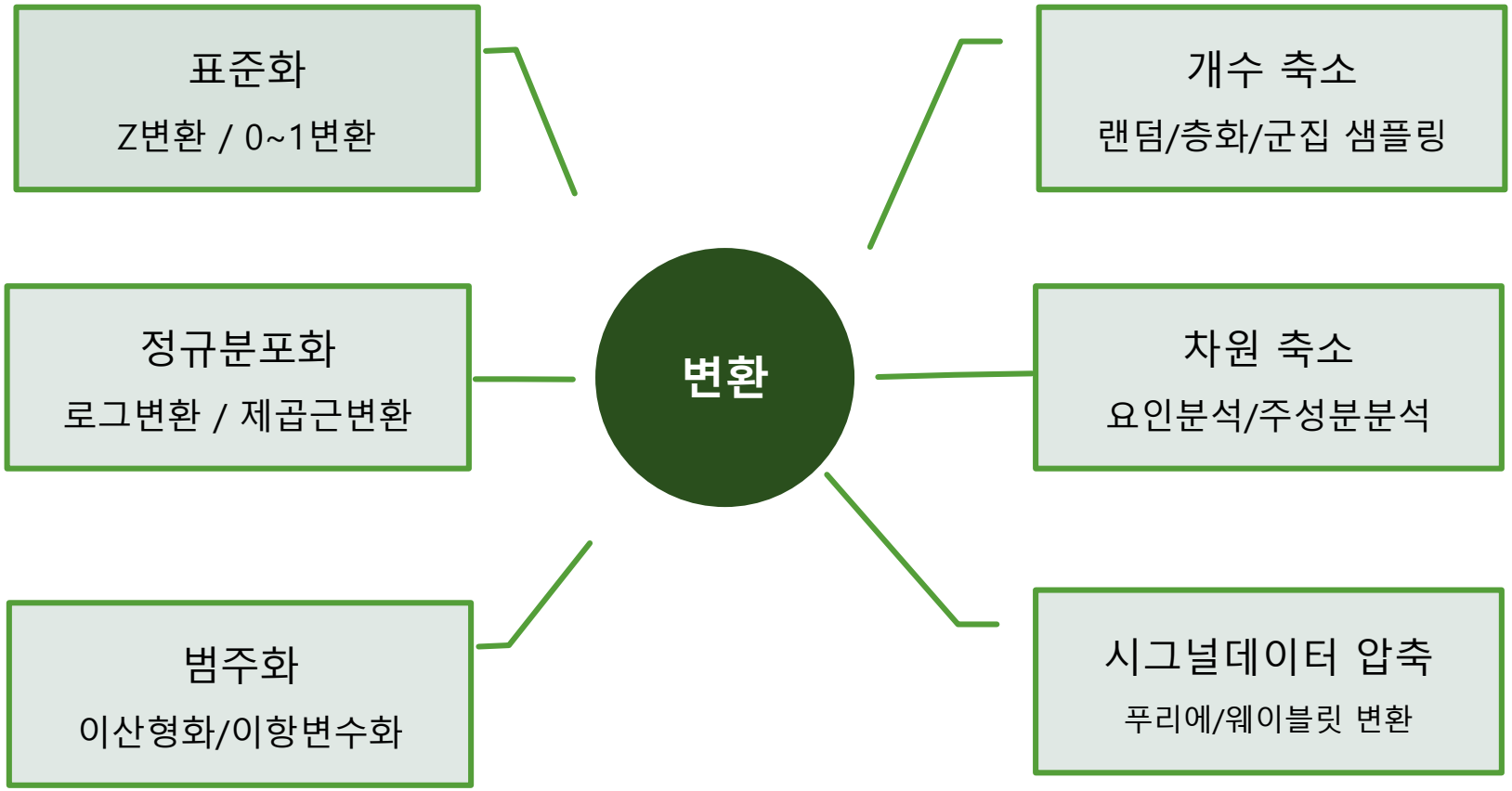
→ 각 구간값을 0 ~ 1 사이로 스케일링

→ 변수 간 크기 차이 맞춰주지만, **이상치가 있으면 [0,1] 구간이 왜곡**

→ 다양한 방식이 존재하며 그중 **Min-Max 정규화**가 가장 대표적

PANDAS 데이터 전처리

◆ 데이터 변환(Transformation)



◆ 데이터 표준화

▪ 표준화(Standardization)란?

- 수집 & 정리된 **데이터의 동일 포맷** 작업
- 여러 가지 제품들 종류/규격을 **표준 따라 제한/통일** 하는 것
- 데이터 **분석 시 정확도 높임**

- 방법

  → 단위 환산 / 자료형 변환 / 범위 변환

◆ 데이터 표준화

▪ 방법 ① 단위 환산

- 수집 데이터의 **나라별 사용 단위 상이**
- **단위별 변환 방식**으로 환산
- 환산된 새로운 **열(column) 추가**

◆ 데이터 표준화

▪ 방법 ② 자료형 변환

- 기 본 자료형 => int, float, str, object
- 데이터 자료형 => 범주형, 연속형

✓ **메서드**

객체.**astype(** 자료형 **)** ➔ 자료형 변환

객체.**dtypes** ➔ 자료형 확인

◆ 데이터 표준화

▪ 방법 ② 자료형 변환

✓ **기본 자료형 사이 변환**

→ int, float, str, object

→ 변환 메서드 : 객체.astype(자료형)

→ 자료형 확인 : 객체.dtypes

◆ 데이터 표준화

▪ 방법 ② 자료형 변환

✓ **데이터 자료형 사이 변환**

→ 연속형 <= = > 범주형

→ 변환 메서드 : 객체.astype(자료형)

→ 자료형 확인 : 객체.dtypes

◆ 데이터 표준화

▪ 방법 ② 자료형 변환

✓ **연속형>>>범주형 : 이산화**

→ 연속형 데이터 일정 구간 >>> 범주형 데이터 변환

```
pandas.cut (    x=df[컬럼명],          # 데이터 배열
                bins=n,                # 경계 값 리스트
                labels=bin_names,      # bin 이름
                include_lowest=True)   # 첫 경계값 포함
)
```

◆ 데이터 표준화

▪ 방법 ② 자료형 변환

✓ **범주형>>>수치형 : 라벨 인코딩**

→ 알파벳 순서에 따라 문자형 데이터를 unique한 숫자형으로 매핑

LabelEncoder

ID	과일
1	사과
2	바나나
3	체리

ID	과일
1	0
2	1
3	2

◆ 데이터 표준화

▪ 방법 ② 자료형 변환

✓ **범주형>>>수치형 : 원 핫 인코딩**

→ 알파벳 순서에 따라 문자형 데이터를 unique한 숫자형으로 매핑

pandas.**get_dummies(** 구간분할 데이터 **)**

ID	과일
1	사과
2	바나나
3	체리

One-Hot Encoding

ID	사과	바나나	체리
1	1	0	0
2	0	1	0
3	0	0	1