03. 데이터 전처리(실습)

소프트웨어융합대학

인공지능학부

이수미

목차

01

데이터 전처리 기본

01. 데이터 전처리

I. 결측치 처리

- 결측치
  - 데이터 수집 과정에서 값이 기록되지 않은 것.
  - 넘파이 배열에서는 결측치를 np.nan으로 표현.
  - 판다스 데이터프레임에서는 결측치를 NaN으로 표현.

[코드 8-1] 결측치가 있는 데이터프레임

```
import pandas as pd
import numpy as np

df = pd.DataFrame({'A': [1, 2, np.nan, 4, 5],
                   'B': [6, 7, 8, np.nan, 10],
                   'C': [11, 12, 13, np.nan, np.nan]})
df
```

	A	B	C
0	1.0	6.0	11.0
1	2.0	7.0	12.0
2	NaN	8.0	13.0
3	4.0	NaN	NaN
4	5.0	10.0	NaN

01. 데이터 전처리

I. 결측치 처리

- 결측치
  - 판다스의 isna( ) 함수로 데이터프레임에서 결측치가 어디에 있는지 빠르게 확인.
  - 데이터가 있으면 False로 표시되고 결측치는 True로 표시됨.

[코드 8-2] 결측치 위치 확인

```
pd.isna(df)
```

	A	B	C
0	False	False	False
1	False	False	False
2	True	False	False
3	False	True	True
4	False	False	True

01. 데이터 전처리

I. 결측치 처리

- 결측치
  - sum( ) 함수를 이용하여 열별 결측치 개수를 확인.

[코드 8-3] 열별 결측치 개수 확인

```
pd.isna(df).sum( )
```

```
A 1
B 1
C 2
dtype: int64
```

  - A열, B열, C열에 결측치가 각각 1개, 1개, 2개.

I. 결측치 처리

- 결측치 제거
  - 결측치를 제거할 때는 제거해도 전체 데이터에 이상이 없는지 점검하여 결측치 중 어떤 것을 제거할지 정해야 함.

[코드 8-4] 행별로 모든 결측치 제거

```
df_drop_nan = df.dropna( )
df_drop_nan
```

	A	B	C
0	1.0	6.0	11.0
1	2.0	7.0	12.0

  - B와 C에 있는 결측치만 제거되어 다섯 행 중 세 행이 남음.

01. 데이터 전처리

I. 결측치 처리

- 결측치 대체
  - 데이터 양이 많지 않을 경우 결측치를 제거하기보다 다른 값으로 대체.

[코드 8-6] 숫자로 결측치 대체

```
df_0 = df['C'].fillna(0)
print(df_0)
```

```
0 11.0
1 12.0
2 13.0
3 0.0
4 0.0
Name: C, dtype: float64
```

  - 결측치가 fillna( ) 함수에 입력한 인자로 대체됨.
  - C열에 있는 결측치를 모두 0으로 대체.

I. 결측치 처리

- 결측치 대체
  - 데이터 양이 많지 않을 경우 결측치를 제거하기보다 다른 값으로 대체.

[코드 8-7] 문자로 결측치 대체

```
df_missing = df['A'].fillna('missing')
df_missing
```

```
0 1.0
1 2.0
2 missing
3 4.0
4 5.0
Name: A, dtype: object
```

  - A열에 있는 결측치를 문자열 'missing'으로 대체.

I. 결측치 처리

- 결측치 대체
  - 평균으로 대체하면 데이터 분포에 영향을 적게 주면서 결측을 해결할 수 있음.

[코드 8-8] 평균으로 결측치 대체

```
# df.fillna(df.mean( ), inplace=True)
df_mean = df.fillna(df.mean( ))
print(df, '\n')
print(df_mean)
```

```
    A   B   C
0 1.0 6.0 11.0
1 2.0 7.0 12.0
2 NaN 8.0 13.0
3 4.0 NaN NaN
4 5.0 10.0 NaN

    A   B   C
0 1.0 6.00 11.0
1 2.0 7.00 12.0
2 3.0 8.00 13.0
3 4.0 7.75 12.0
4 5.0 10.00 12.0
```

I. 결측치 처리

- 결측치 대체
  - 결측치 바로 위나 아래 행의 값으로 대체할 수 있음.

[코드 8-9] 주변 데이터로 결측치 대체

```
print(df, '\n')
#결측치 바로 위의 값으로 대체하기
df_ffill = df.fillna(method='ffill')
print(df_ffill, '\n')
#결측치 바로 아래의 값으로 대체하기
df_bfill = df.fillna(method='bfill')
print(df_bfill)
```

```
     A    B    C
0  1.0  6.0 11.0
1  2.0  7.0 12.0
2  NaN  8.0 13.0
3  4.0  NaN  NaN
4  5.0 10.0  NaN

     A    B    C
0  1.0  6.0 11.0
1  2.0  7.0 12.0
2  2.0  8.0 13.0
3  4.0  8.0 13.0
4  5.0 10.0 13.0

     A    B    C
0  1.0  6.0 11.0
1  2.0  7.0 12.0
2  4.0  8.0 13.0
3  4.0 10.0  NaN
4  5.0 10.0  NaN
```

  - df_ffill은 fillna( ) 함수의 method 인자 값을 'ffill'로 입력하여 결측치를 바로 위의 데이터로 대체.
  - df_bfill은 method 인자 값을 'bfill'로 입력하여 결측치를 바로 아래의 데이터로 대체.

I. 결측치 처리

- 결측치 대체
  - 각 열에 서로 다른 값을 할당할 수도 있음.

[코드 8-10] 각 열을 서로 다른 값으로 대체

```
fill_dict = {'A': df['A'].mean( ), 'B': '12/25', 'C': 'missing'}
df_filled = df.fillna(value=fill_dict)
df_filled
```

	A	B	C
0	1.0	6.0	11.0
1	2.0	7.0	12.0
2	3.0	8.0	13.0
3	4.0	12/25	missing
4	5.0	10.0	missing

  - A열의 결측치는 평균으로 대체하고, B열의 결측치는 '12/25', C열의 결측치는 'missing'으로 대체.

01. 데이터 전처리

II. 이상치

– 이상치(Outlier)는 데이터셋에서 대부분의 데이터가 모인 범위를 크게 벗어난 값.

표 8-1 이상치가 있는 데이터

비교	이상치가 없는 데이터	이상치가 있는 데이터
데이터	1, 2, 2, 3, 3, 5	1, 2, 2, 3, 3, 500
평균	2.86	85.17
중앙값	3	2.5
표준편차	1.35	203.23

– 이상치가 있는 데이터셋은 이상치가 없는 데이터셋과 비교하여 평균과 표준편차가 유의하게 다름.

– 이상치에 면밀히 주의하지 않으면 통계량을 크게 잘못 추정할 수 있음.

II. 이상치

- 이상치의 원인
  - 이상치가 발생하는 원인에 따라 다음 세 가지로 분류.
    ① 데이터 입력 오류: 데이터 수집과 기록 과정에서 발생하는 오류.
    ② 측정 오류: 이상치의 가장 일반적인 원인.
    ③ 자연 이상치: 이상치의 원인이 인위적이지 않다면 자연 이상치.
- 이상치 제거
  - 이상치를 식별할 때 일반적으로 데이터의 분포를 시각화.
  - 이상치를 식별한 후에는 이를 제거하는 방법을 선택.
    » 이상치가 몇 개 없을 때는 이상치를 수동으로 삭제하거나 수정.
    » 이상치가 많다면 데이터 분석 결과에 영향을 주지 않는 정도의 이상치를 자동으로 제거.
  - 이상치를 너무 많이 제거하면 유용한 데이터까지 잃을 수 있음.

II. 이상치

- 이상치를 제거하는 기법으로 IQR이나 Z 점수와 같은 통계 기법이 있음.
- IQR(Interquartile range)은 제1사분위수에서 제3사분위수까지의 거리.
- IQR의 1.5배보다 멀리 떨어진 데이터를 이상치로 간주하여 제거.
  - Q1은 데이터의 첫번째 사분위수(25번째 백분위수).
  - Q2는 데이터의 두번째 사분위수(중앙값, 50번째 백분위수).
  - Q3은 데이터의 세번째 사분위수(75번째 백분위수).
  - Q1 - 1.5 × IQR을 최솟값. Q3 + 1.5 × IQR을 최댓값으로 정함. 최솟값보다 작거나 최댓값보다 큰 값은 모두 이상치이므로 제거.

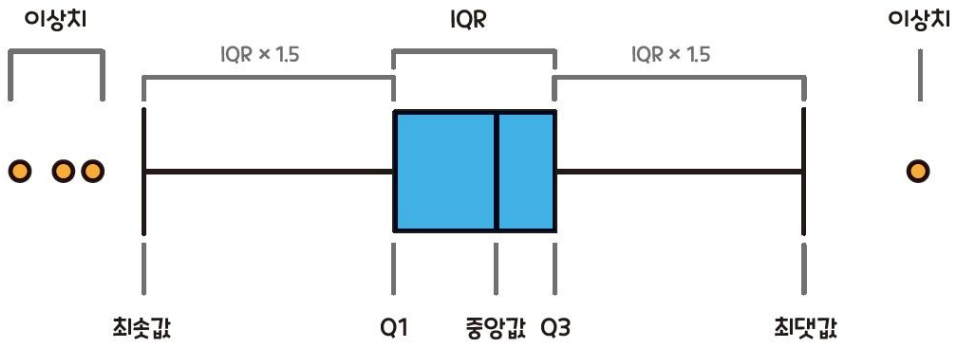


그림 8-1 상자 그래프와 IQR

II. 이상치

- 데이터 시각화 이용
  - 이상치를 확인할 때 데이터 시각화를 이용하면 매우 편리함.

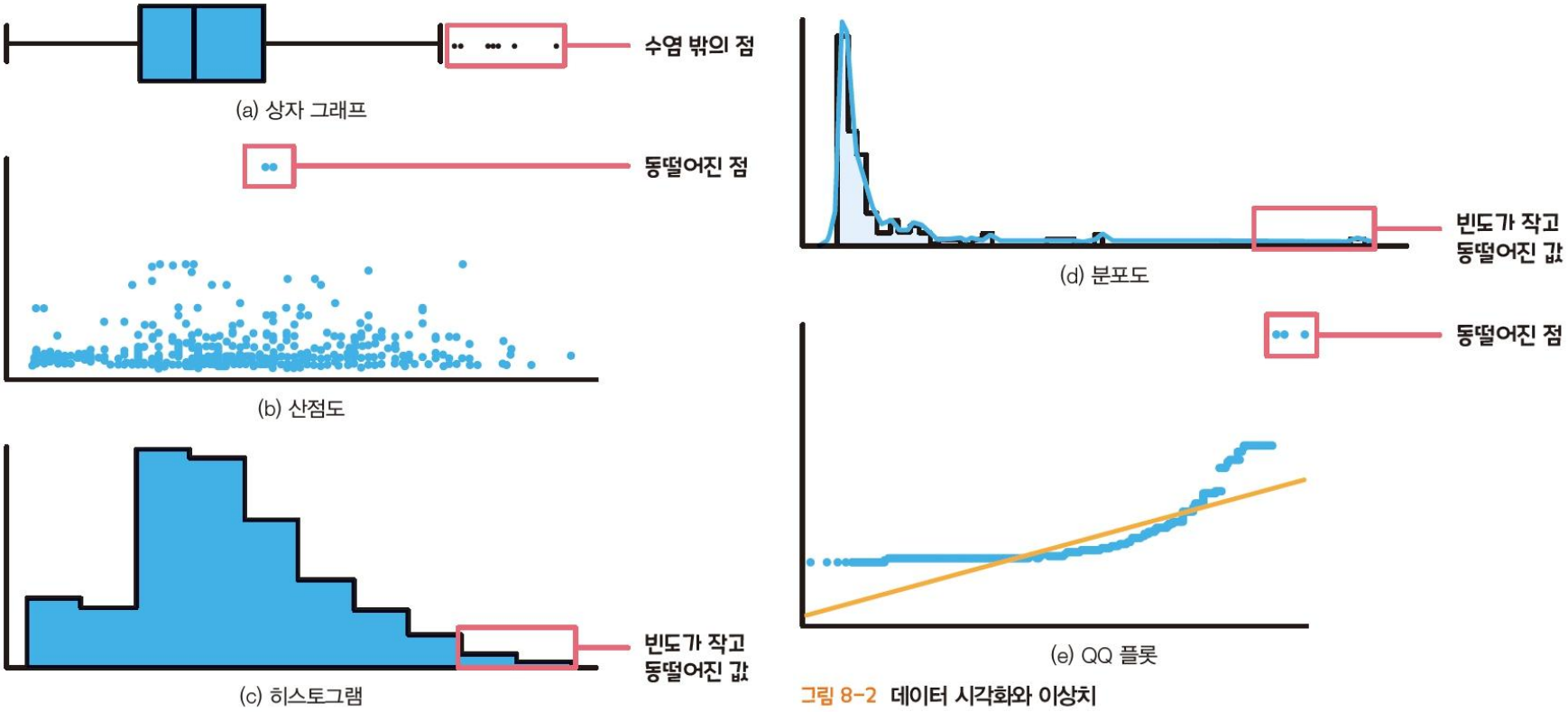


그림 8-2 데이터 시각화와 이상치

III. 표준화와 정규화

- 데이터에서 특성(Feature) 또는 특징이란 분석 대상에 영향을 주는 속성을 말함.
- 데이터를 분석할 때는 특성 중 어느 것이 분석 대상에 더 크게 영향을 미치는지 비교하여 선택하게 됨.
- 단위가 다르면 비교가 어려움. 키가 170cm인 사람과 몸무게가 70kg인 사람 중 누가 더 큰지 말할 수 없는 것과 마찬가지.

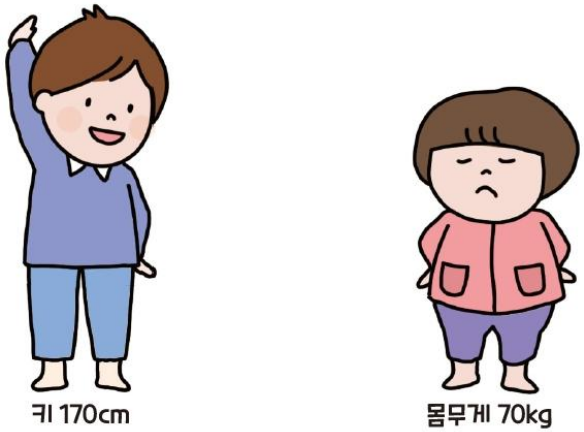


그림 8-3 단위가 다른 값

III. 표준화와 정규화

- 단위가 같아도 값의 범위가 크게 차이 난다면 비교하기 어려움.
- A 도시에서 모든 식당의 자장면 값이 5,000원이고 떡볶이 값은 2,000원부터 2만 원까지 분포하여 평균이 5,000원. 이때 A 도시에서 자장면과 떡볶이 중 어느 쪽이 더 비싼지 말할 수 없음.

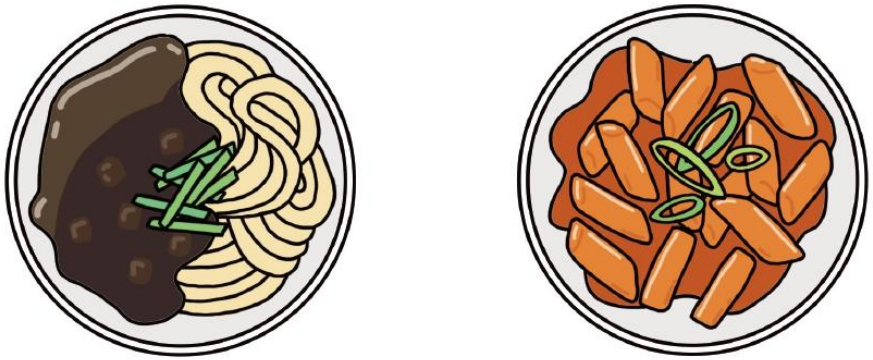


그림 8-4 범위가 다른 값

III. 표준화와 정규화

- 데이터를 분석하려면 특성들의 상대적인 차이를 줄여 특성들이 모두 비슷한 정도로 대상에 영향력을 행사하도록 값을 변환해야 함.
- 표준화와 정규화를 수행하면 특성의 단위에 관계없이 값을 바로 비교할 수 있음.
- 표준화와 정규화 작업을 특성 스케일링(Feature scaling) 또는 데이터 스케일링(Data scaling) 이라고 함.

III. 표준화와 정규화

- 표준화
  - 표준화(Standardization)는 데이터 분포를 평균이 0이고 표준편차가 1이 되게 변환하는 데이터 전처리 기법.
    - » 여러 개의 변수가 있을 때 서로 다른 변수들을 비교하기 편리하게 만듦.
  - Z 점수(Z-score)는 어떤 값이 평균에서 얼마나 떨어져 있는지를 나타내는 수치.
    - » 확률변수 X가 평균 μ(뮤)로부터 표준편차 σ(시그마)의 몇 배만큼 떨어져 있는지를 Z 점수로 정의.
    - » Z 점수를 표준화 점수 또는 표준 점수라고도 함.

$$Z\text{ 점수} = \frac{(X - \text{평균})}{\text{표준편차}}$$

III. 표준화와 정규화

- 표준화
  - Z 점수(Z-score)는 어떤 값이 평균에서 얼마나 떨어져 있는지를 나타내는 수치.
    - » Z 점수는 표준편차의 배수로 계산됨.
    - » 표준편차는 데이터의 분포를 나타낸 값이기 때문에,
      Z 점수를 알면 서로 다른 분포로부터 나온 데이터를 비교할 수 있음.

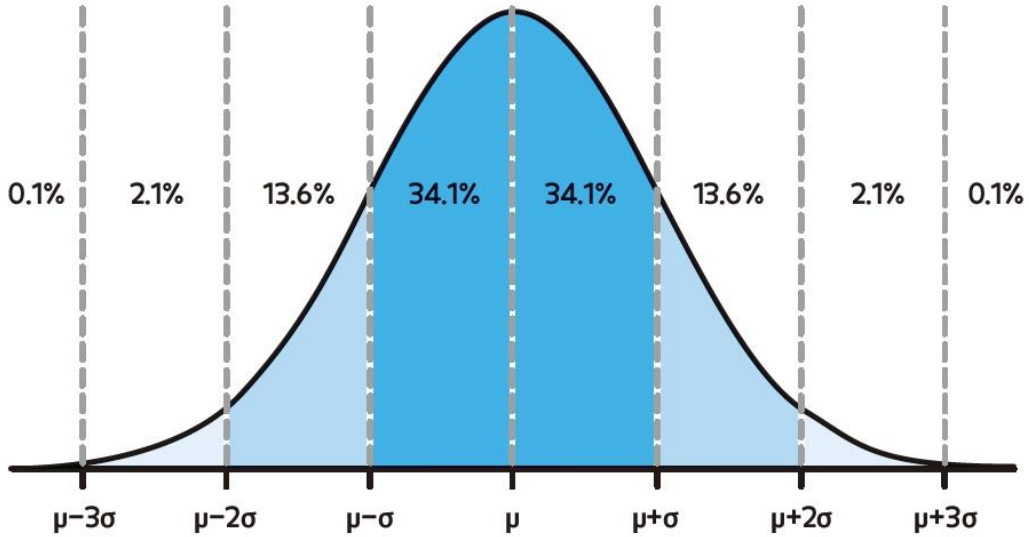


그림 8-5 정규분포곡선

III. 표준화와 정규화

- 표준화
  - 평균을 중심으로 하고 표준편차 단위로 정규 곡선 아래의 영역을 나누었을 때 데이터 점은 다음과 같이 분포.
    - » 데이터 점의 68.3%는 평균(μ) ± 1×표준편차(σ) 안에 있음.
    - » 데이터 점의 95.4%가 평균(μ) ± 2×표준편차(σ) 안에 있음.
    - » 데이터 점의 99.7%가 평균(μ) ± 3×표준편차(σ) 안에 있음.
  - 따라서 정규분포곡선에서 Z 점수 절댓값이 3 이하인 데이터가 전체의 99.7%, 이 데이터를 이상치로 간주하여 제거할 수 있음.

III. 표준화와 정규화

- 표준화
  - Z 점수는 응시자 수가 많은 시험에서 개개인의 성적이 전체에서 어떤 위치에 있는지 보여줄 때 사용하기도 함.
    - » 정규분포를 따르는 학년 영어점수 평균이 60점이고 표준편차가 10점일 때, 세일이의 점수가 70점이라면 Z 점수는 1.
    - » 정규분포를 따르는 학년 국어점수 평균이 60점이고 표준편차가 15점일 때, 세일이의 점수가 75점이라면 Z 점수는 똑같이 1.
    - » 세일이의 영어점수와 국어점수는 Z 점수가 같으므로 학년 분포에서 같은 위치에 있다고 할 수 있음.
    - » 이 학년 학생 68.2%는 국어점수가 45점(Z 점수가 -1)에서 75점(Z 점수가 1) 사이일 것.

그림 8-6 Z 점수

III. 표준화와 정규화

- 정규화
  - 정규화(Normalization)는 데이터를 특정 범위 내의 값으로 조정하는 데이터 전처리 기법.
  - 정규화를 하면 다양한 범위와 단위의 데이터를 서로 비교할 수 있음.

$$X' = \frac{(X - Xmin)}{Xmax - Xmin}$$

  - 대표적인 정규화 방법으로 최소-최대 정규화(Min-Max scaling).
    - » 데이터의 최솟값과 최댓값을 사용하여 데이터를 0과 1 사이의 값으로 변환.
  - 정규화를 수행하면 여러 특성을 같은 범위로 맞출 수 있고, 특성 간 비교가 쉬워져 빅데이터 분석에 유리하고 인공지능 모형의 성능을 높일 수 있음.

III. 표준화와 정규화

- 표준화와 정규화 비교

표 8-2 표준화와 정규화 비교

비교	표준화	정규화
사용하는 값	평균과 표준편차를 사용	최댓값, 최솟값을 사용
목적	평균을 0으로, 표준편차를 1로 만들기	데이터의 범위를 서로 맞추기
전처리 후 데이터	범위가 제한되지 않음	[0, 1] 또는 [−1, 1] 사이로 스케일링
전처리 전 데이터	데이터가 정규분포일 때 유용함	데이터 분포를 모를 때 유용함

02

데이터 전처리 활용

I. 지하수 데이터 수집

– 데이터를 분석하기 전에는 먼저 이상치 여부를 확인하여 처리해야 함.

– 2009년부터 2016년까지 대전 지하수 데이터를 준비하여 이상치가 있는지 확인하고 제거해보기

• 지하수 데이터 준비하기

– 국가지하수정보센터(http://www.gims.go.kr)에 접속하고 [고객지원] 메뉴를 클릭.

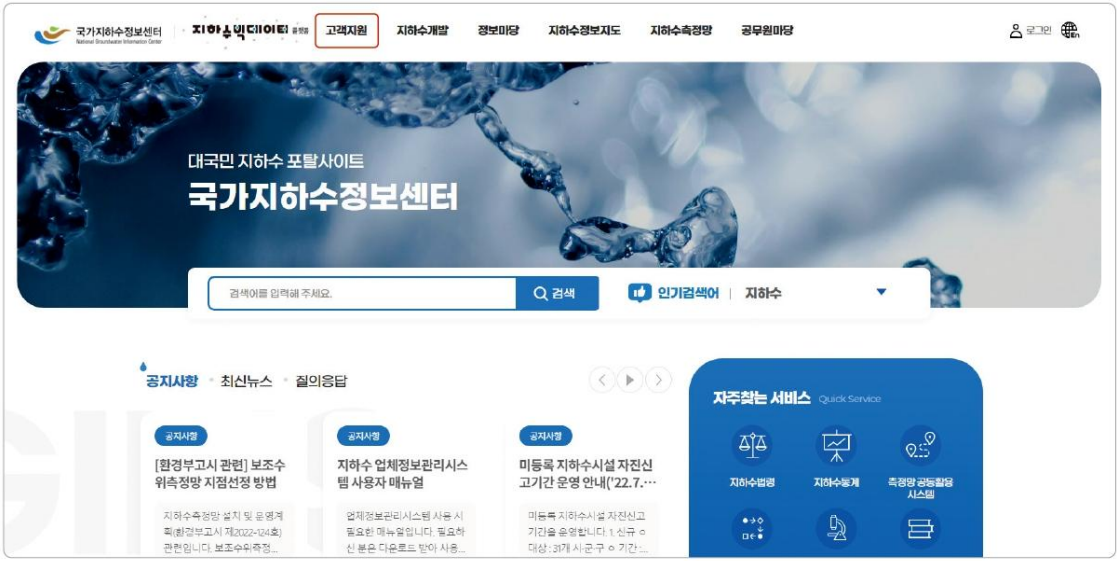


그림 8-7 국가지하수정보센터

I. 지하수 데이터 수집

- 지하수 데이터 준비하기
  - [지하수정보 신청]으로 이동하여 [자료신청 바로가기] 버튼을 클릭.

그림 8-8 지하수정보 신청

I. 지하수 데이터 수집

- 지하수 데이터 준비하기
  - [검색] 탭에서 [측정자료]를 선택, [대전광역시]의 [전체] 지역을 선택하고, 자료 종류는 측정자료(시), 측정망은 [대전태평(암반)], 연도는 [2009]를 선택.
  - [자료검색] 버튼을 클릭하고 가장 아래의 [신청서작성]을 클릭.

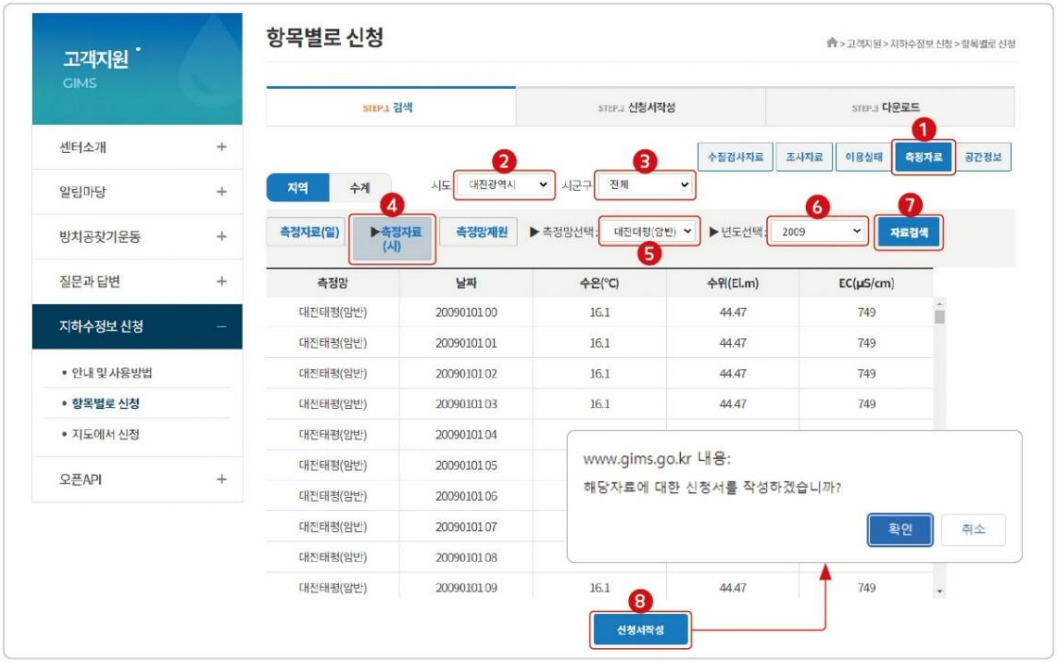


그림 8-9 지하수 데이터 검색

I. 지하수 데이터 수집

- 지하수 데이터 준비하기
  - 다음 단계인 [신청서작성] 탭에서 [신청서 제출]을 클릭.

그림 8-10 신청서 제출

  - 마지막 단계인 [다운로드] 탭에서 아래 파란색 [다운로드] 클릭.

그림 8-11 다운로드

I. 지하수 데이터 수집

- 지하수 데이터 준비하기
  - 파일 준비에 시간이 오래 걸릴 수 있음. 다운로드한 파일의 압축을 해제.

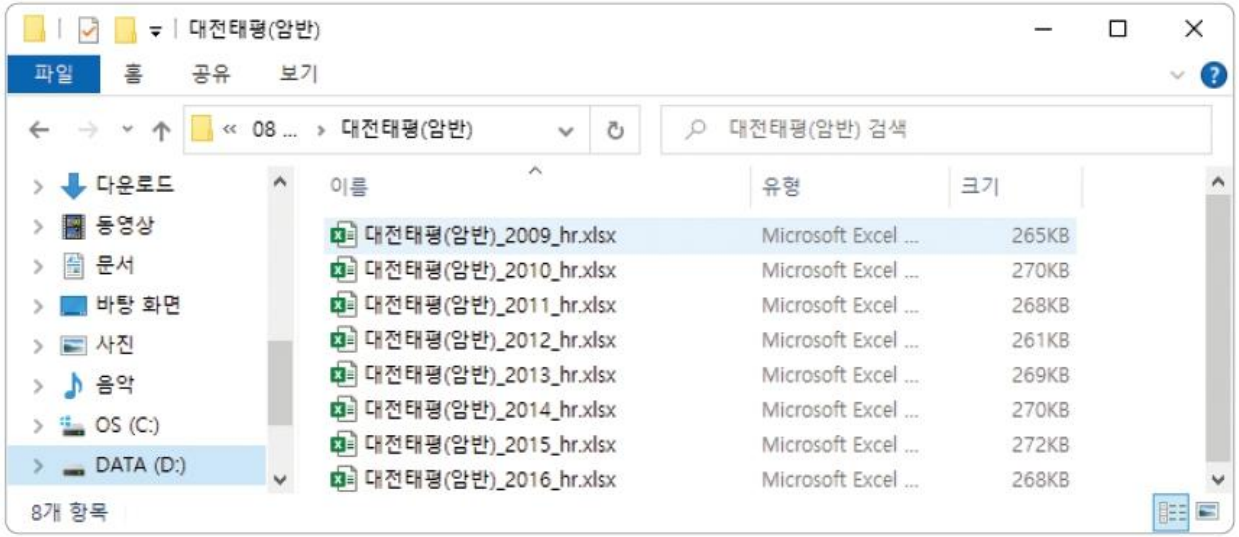


그림 8-12 지하수 데이터 파일

I. 지하수 데이터 수집

- 지하수 데이터 준비하기
  - Colab에서 새 노트북을 열고 디렉토리에서 마우스 오른쪽 버튼을 클릭하여 메뉴 도구에서 [새 폴더]를 선택.
  - 폴더 [Untitled Folder]의 이름을 대전태평(암반)으로 변경.

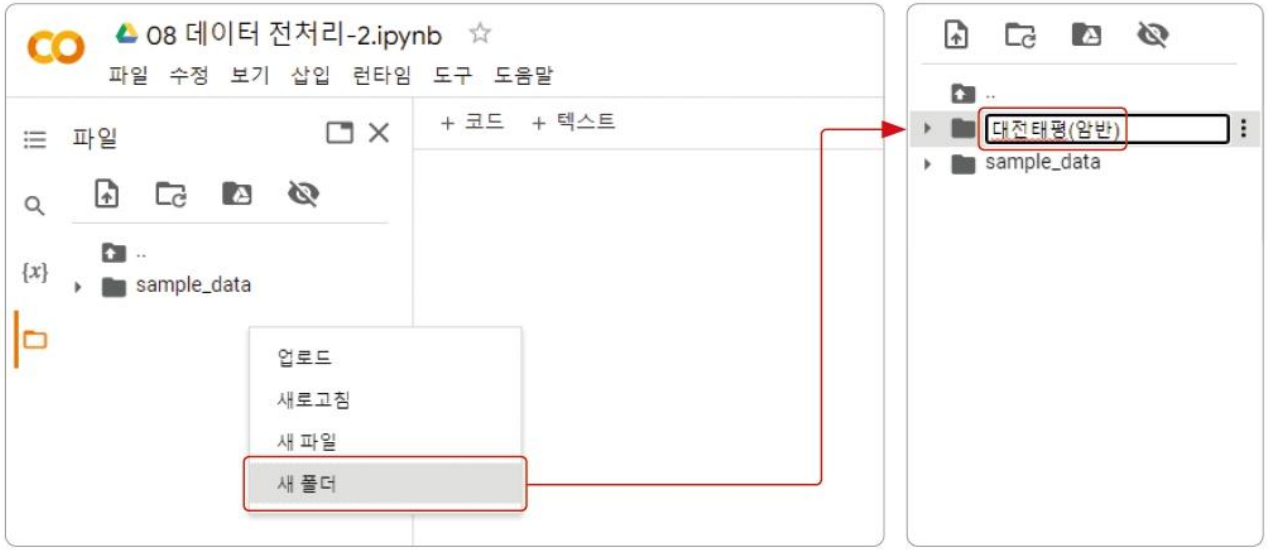


그림 8-13 디렉토리에 새 폴더 생성

I. 지하수 데이터 수집

- 지하수 데이터 준비하기
  - 탐색기에 있는 지하수 데이터 엑셀 파일을 모두 선택한 채로 Colab 디렉토리의 [대전태평(암반)] 폴더까지 드래그하면 한꺼번에 업로드됨.

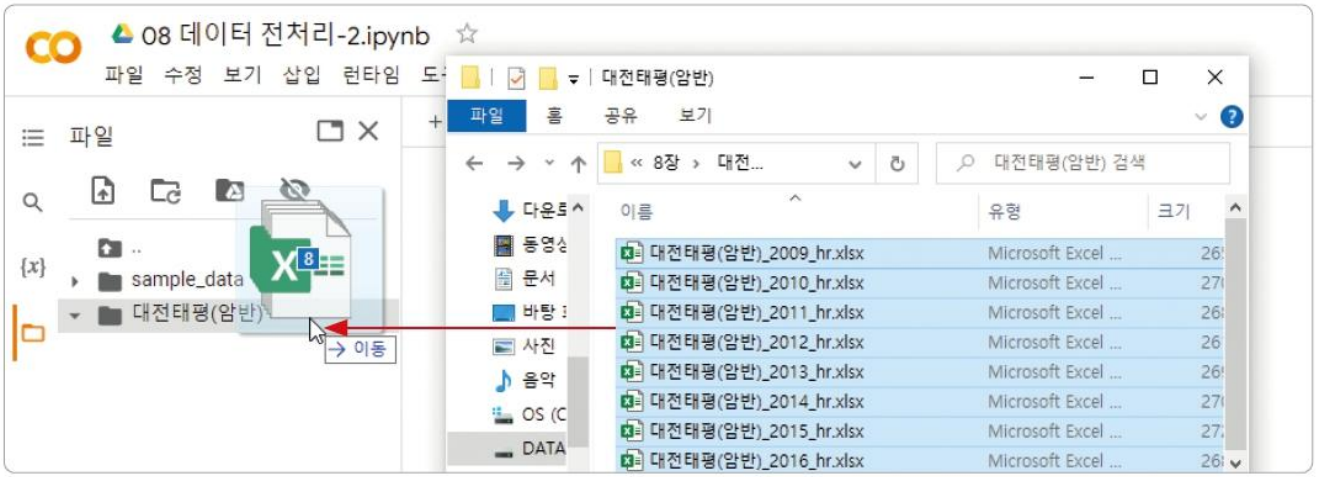


그림 8-14 지하수 데이터 파일 업로드

02. 데이터 전처리 활용

I. 지하수 데이터 수집

- 라이브러리 불러오기
  - 넘파이와 판다스, 맷플롯립 라이브러리가 필요함.

[코드 8-11] 라이브러리

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
from datetime import datetime, date, time

warnings.filterwarnings('ignore')
```

  - os는 path를 지정할 때 사용, warnings는 경고 관련 라이브러리이고 datetime, date, time은 날짜와 시간 데이터를 다루는 데 사용할 모듈.
  - warnings 라이브러리의 filterwarnings( ) 함수로 경고 메시지를 제거.

I. 지하수 데이터 수집

- 데이터 병합
  - 데이터 파일 8개를 하나의 파일로 병합하기.
  - 먼저 [대전태평(암반)] 폴더에 있는 엑셀 파일의 이름을 구하기.

[코드 8-12] 파일명 확인

```
path = './대전태평(암반)/'
file_list = os.listdir(path)
file_list_py = [file for file in file_list if file.endswith('.xlsx')]
file_list_py
```

```
['대전태평(암반)_2010_hr.xlsx',
 '대전태평(암반)_2016_hr.xlsx',
 '대전태평(암반)_2015_hr.xlsx',
 '대전태평(암반)_2009_hr.xlsx',
 '대전태평(암반)_2014_hr.xlsx',
 '대전태평(암반)_2013_hr.xlsx',
 '대전태평(암반)_2011_hr.xlsx',
 '대전태평(암반)_2012_hr.xlsx']
```

- path에 데이터가 업로드된 경로를 할당하고 경로 path에 있는 파일 전체의 파일명 리스트를 file_list에 할당.
- 파일명이 '.xlsx'로 끝나는 파일만 리스트 file_list_py에 저장.

I. 지하수 데이터 수집

- 데이터 병합

[코드 8-13] 데이터프레임 하나로 병합

```
df = pd.DataFrame()
for i in file_list_py:
    data = pd.read_excel(path + i)
    df = pd.concat([df, data])
df
```

- read_excel( ) 함수는
  엑셀 파일의 내용을 읽어옴.
- 데이터프레임 df에
  변수 data의 내용을 추가.

	관측소	구분	날짜	시간	수온(℃)	수위(el.m)	EC(㎲/㎝)
0	대전태평	(암반)	20150101	0	16.3	44.52	736
1	대전태평	(암반)	20150101	1	16.3	44.52	736
2	대전태평	(암반)	20150101	2	16.3	44.52	736
3	대전태평	(암반)	20150101	3	16.3	44.52	736
4	대전태평	(암반)	20150101	4	16.3	44.52	736
...	...	...	...	...	...	...	...
8689	대전태평	(암반)	20141231	4	16.3	44.53	736
8690	대전태평	(암반)	20141231	3	16.3	44.53	736
8691	대전태평	(암반)	20141231	2	16.3	44.52	736
8692	대전태평	(암반)	20141231	1	16.3	44.52	736
8693	대전태평	(암반)	20141231	22	16.3	44.52	736

69548 rows × 7 columns

I. 지하수 데이터 수집

- 데이터 병합
  - 인덱스 열에 0부터 69547까지의 숫자가 중복 없이 나타나도록 행 인덱스를 재설정.

[코드 8-14] 행 인덱스 재설정

```
df = df.reset_index(drop=True)
df
```

	관측소	구분	날짜	시간	수온(℃)	수위(el.m)	EC(㎲/㎝)
0	대전태평	(암반)	20150101	0	16.3	44.52	736
1	대전태평	(암반)	20150101	1	16.3	44.52	736
2	대전태평	(암반)	20150101	2	16.3	44.52	736
3	대전태평	(암반)	20150101	3	16.3	44.52	736
4	대전태평	(암반)	20150101	4	16.3	44.52	736
...	...	...	...	...	...	...	...
69543	대전태평	(암반)	20141231	4	16.3	44.53	736
69544	대전태평	(암반)	20141231	3	16.3	44.53	736
69545	대전태평	(암반)	20141231	2	16.3	44.52	736
69546	대전태평	(암반)	20141231	1	16.3	44.52	736
69547	대전태평	(암반)	20141231	22	16.3	44.52	736

I. 지하수 데이터 수집

- 데이터 병합

[코드 8-15] 데이터프레임 정보 확인

```
df.info( )
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 69548 entries, 0 to 69547
Data columns (total 7 columns):
 # Column Non-Null Count Dtype
--- ------ -------------- -----
0 관측소 69548 non-null object
1 구분 69548 non-null object
2 날짜 69548 non-null int64
3 시간 69548 non-null int64
4 수온(℃) 69548 non-null float64
5 수위(el.m) 69548 non-null float64
6 EC(㎲/㎝) 69548 non-null int64
dtypes: float64(2), int64(3), object(2)
memory usage: 3.7+ MB
```

– df 데이터프레임 모든 열에 결측치가 없음.

I. 지하수 데이터 수집

- 측정시각 열 생성
  - 시간에 따라 변화하는 데이터를 그래프로 확인하고 싶음.
  - 날짜와 시간을 합쳐 열 하나로 만들어야 함.

[코드 8-16] 인덱스 지정

```
df['Date'] = pd.to_datetime(df['날짜'],format='%Y%m%d') +\
          pd.to_timedelta(df['시간'].astype(int),unit='h’)

df.set_index(df['Date'], inplace=True)
df.head(3)
```

Date	관측소	구분	날짜	시간	수온(℃)	수위(el.m)	EC(㎲/㎝)
2010-01-01 00:00:00	대전태평	(암반)	20100101	0	16.2	44.47	741
2010-01-01 01:00:00	대전태평	(암반)	20100101	1	16.2	44.47	741
2010-01-01 02:00:00	대전태평	(암반)	20100101	2	16.2	44.47	741

I. 지하수 데이터 수집

- 분석할 속성만 조회하여 저장
  - 데이터프레임 df의 속성 중에서 수온, 수위, EC만 분석하고 싶음.
  - 데이터프레임 속성을 열 이름으로 조회하고 새로운 데이터프레임 df1에 할당.
  - 열 이름을 영문으로 변경.

[코드 8-17] 분석할 속성만 조회하여 저장

```
df1 = df[['수온(℃)', '수위(el.m)','EC(㎲/㎝)']]
df1.columns = ['temp', 'level', 'EC']
df1.head(3)
```

Date	temp	level	EC
2015-01-01 00:00:00	16.3	44.52	736
2015-01-01 01:00:00	16.3	44.52	736
2015-01-01 02:00:00	16.3	44.52	736

I. 지하수 데이터 수집

- 분석할 속성만 조회하여 저장
  - 결측치 유무도 확인.

[코드 8-18] 결측치 수 확인

```
df1.isnull( ).sum( )
```

```
temp        0
level       0
EC          0
dtype: int64
```

  - isnull( ) 함수는 데이터프레임에서 null의 개수를 반환함
  - 모든 열에 결측치가 없는 것이 확인됨.

I. 지하수 데이터 수집

- 분석할 속성만 조회하여 저장
  - 데이터프레임 df1을 다음에도 사용할 수 있도록 '대전지하수.csv' 파일로 저장.

[코드 8-19] CSV 파일 저장

```
df1.to_csv('./대전지하수.csv', encoding='cp949')
```

  - Colab에서 런타임 연결을 해제하면 디렉토리에 있는 파일이 사라지므로 파일이 생성되면 바로 PC에 다운로드하거나 클라우드에 저장해 두어야 함.

II. 이상치 확인 및 제거

– 지하수 데이터 측정 센서의 이상이나 이물질 등의 원인으로 이상치가 발생할 수 있음.

- 통계량과 그래프를 이용하는 이상치 확인

[코드 8-20] 통계량으로 이상치 확인

```
df = pd.read_csv('./대전지하수.csv', index_col='Date', parse_dates=True,
encoding='cp949')
df.describe()
```

– '대전지하수.csv' 파일을 읽어옴.
Date 열을 인덱스로 지정.

– 데이터프레임 df의 기초 통계량을 확인.

– level 열과 EC 열에서 최솟값과 1사분위수의 차이가 크니 최솟값이 이상치일 가능성이 높음.

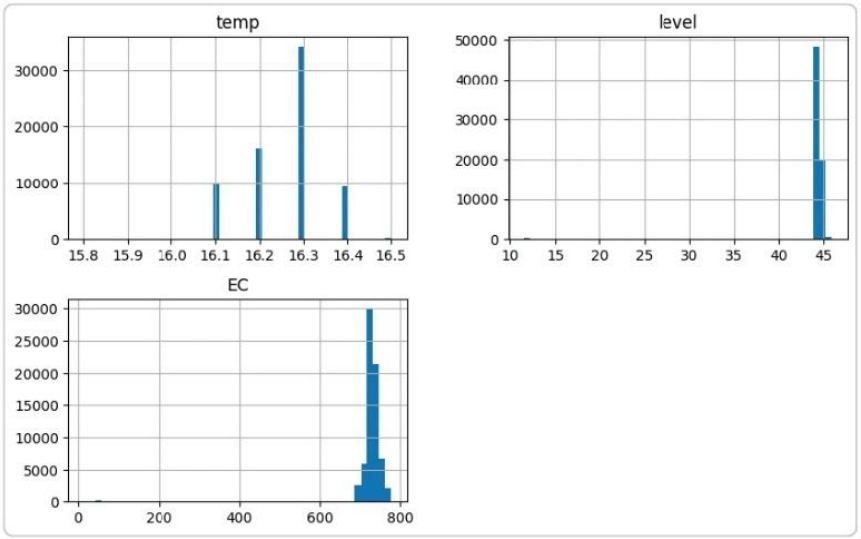
	temp	level	EC
count	69548.000000	69548.000000	69548.000000
mean	16.262491	44.295353	727.024027
std	0.089778	2.539464	51.544590
min	15.800000	11.530000	13.000000
25%	16.200000	44.370000	722.000000
50%	16.300000	44.460000	730.000000
75%	16.300000	44.550000	742.000000
max	16.500000	45.900000	778.000000

II. 이상치 확인 및 제거

- 통계량과 그래프를 이용하는 이상치 확인
  - 히스토그램으로 데이터의 분포를 확인하여 이상치가 있는지 살펴보기.

[코드 8-21] 히스토그램

```
df.hist(bins=50, figsize=(10,6))
plt.show()
```



  - 전체적으로 분포가 오른쪽으로 치우쳐 있을 때 분포의 왼쪽 끝이 이상치일 가능성이 높음.

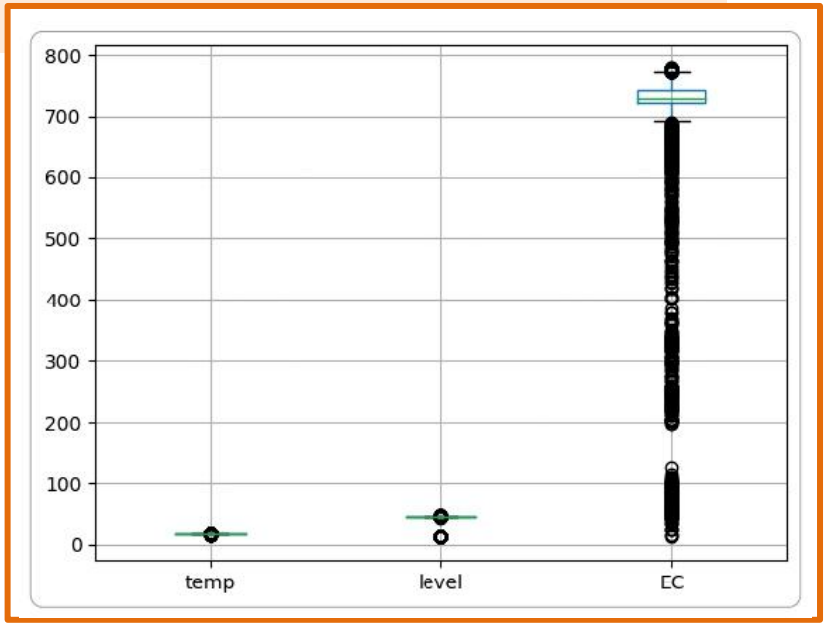
II. 이상치 확인 및 제거

- 통계량과 그래프를 이용하는 이상치 확인
  - 상자 그래프로도 이상치를 확인.

[코드 8-22] 상자 그래프로 이상치 확인

```
df.boxplot( )
```

  - 한꺼번에 세 열의 상자 그래프를 그렸더니 EC 데이터가 다른 열에 비해 값이 커서 temp와 level의 이상치 유무를 확인하기 어려움.

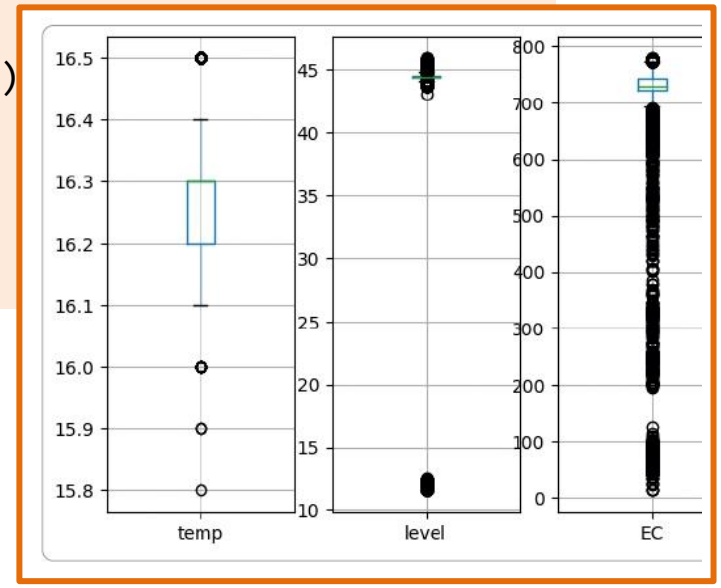


II. 이상치 확인 및 제거

- 통계량과 그래프를 이용하는 이상치 확인
  - 맷플롯립의 다중 그래프를 활용하여 각 열의 데이터로 상자 그래프를 따로 그리기.

[코드 8-23] 다중 상자 그래프로 이상치 확인

```
plt.subplot(1, 3, 1)
df.boxplot(column='temp', return_type='both')
plt.subplot(1, 3, 2)
df.boxplot(column='level', return_type='both')
plt.subplot(1, 3, 3)
df.boxplot(column='EC', return_type='both')
plt.show( )
```



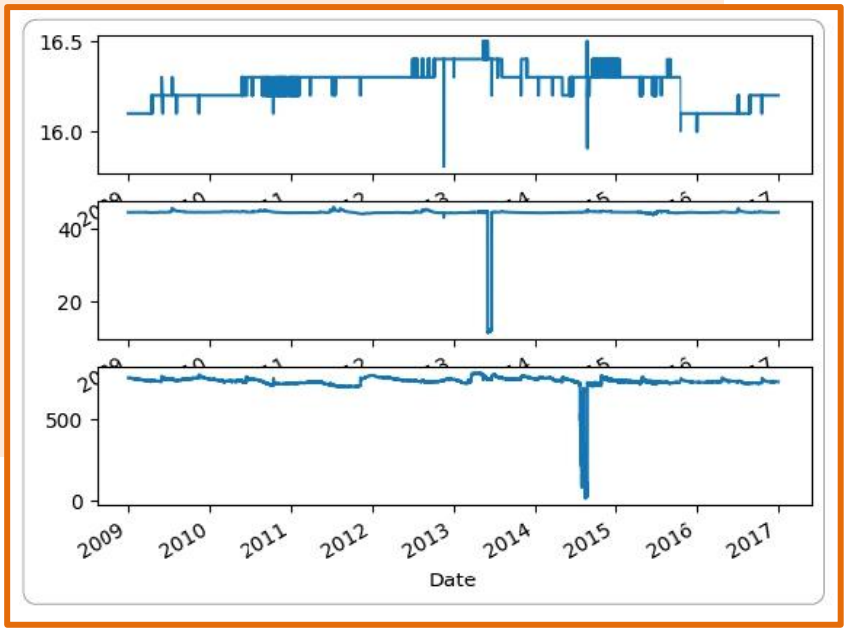
  - 지하수위 그래프에서는 지하수위가 낮은 영역에서 이상치가 나타남.
  - 지하수 전도도는 전도도가 낮은 영역에서 이상치가 나타남.

II. 이상치 확인 및 제거

- 통계량과 그래프를 이용하는 이상치 확인
  - 시간에 따른 데이터의 변화를 나타내는 그래프를 그려서 이상치를 확인할 수도 있음.

[코드 8-24] 다중 시계열 그래프

```
plt.subplot(3, 1, 1)
df['temp'].plot( )
plt.subplot(3, 1, 2)
df['level'].plot()
plt.subplot(3, 1, 3)
df['EC'].plot( )
plt.show( )
```



  - 지하수온이 2012년 후반과 2014년 후반에 평소보다 확연히 낮은데, 연속적으로 변하는 값이므로 이상치일 가능성이 높음.

II. 이상치 확인 및 제거

- IQR을 이용하는 이상치 확인
  - 사분범위(Interquatile Range, IQR)를 계산하고 이를 이용하여 이상치를 탐지.

```
IQR = 3사분위수 (75% 지점) - 1사분위수 (25% 지점)
상한값 = 3사분위수 + IQR * 1.5
하한값 = 1사분위수 - IQR * 1.5
```

  - 이상치는 하한값과 상한값을 벗어나는 영역의 데이터. 반대로 하한값보다 크고 상한값보다 작은 데이터를 정상치.

II. 이상치 확인 및 제거

- IQR을 이용하는 이상치 확인
  - quantile( ) 함수로 직접 사분위수를 계산할 수 있음.

[코드 8-25] 지하수위 IQR

```
q3_level = df['level'].quantile(q=0.75)
q1_level = df['level'].quantile(q=0.25)
iqr_level = q3_level - q1_level
print(iqr_level)
```

```
0.17999999999999972
```

  - 지하수위 데이터의 3사분위수를 q3_level에 저장하고 1사분위수를 q1_level에 저장.
  - iqr_level에 3사분위수와 1사분위수의 차, 즉 IQR을 저장.

II. 이상치 확인 및 제거

- IQR을 이용하는 이상치 확인
  - 계산하여 얻은 3사분위수, 1사분위수, IQR을 이용하여 데이터 상한값과 하한값을 정하고, 이를 벗어나는 데이터를 이상치로 정한다.

[코드 8-26] 지하수위 상하한값과 이상치 개수

```
upper_level = q3_level + 1.5 * iqr_level
lower_level = q1_level - 1.5 * iqr_level
print(upper_level, '/', lower_level)
print((df['level'] > upper_level).sum( ))
print((df['level'] < lower_level).sum( ))
```

```
44.81999999999999 / 44.099999999999994
4492
1273
```

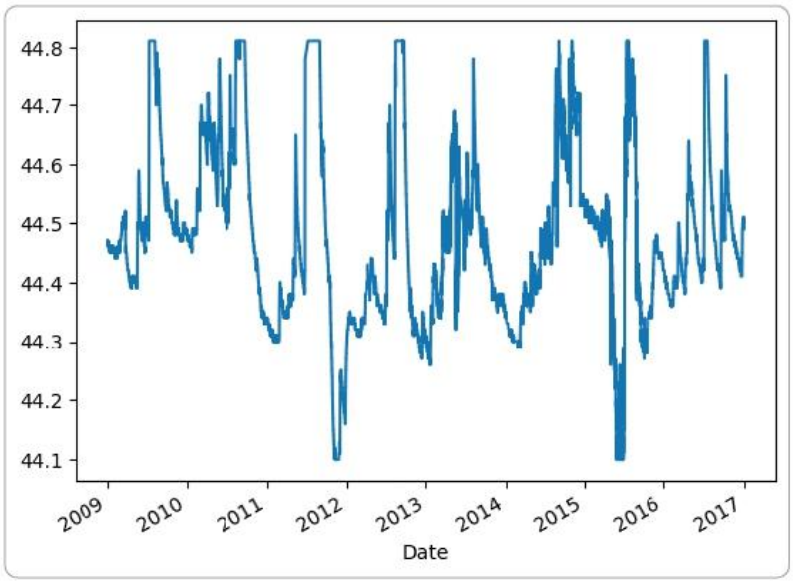
  - 지하수위의 상한값(3분위수로부터 IQR의 1.5배만큼 큰 값)을 변수 upper_level에 저장.
  - 상한값을 벗어나는 이상치는 4,492개, 하한값을 벗어나는 이상치는 1,273개.

II. 이상치 확인 및 제거

- 이상치 제거
  - 이상치를 제거한 데이터만 조회하여 새 데이터프레임에 저장.

[코드 8-27] 이상치를 제거한 지하수위

```
df_iqr_level = df[(df['level'] < upper_level) & (df['level'] > lower_level)]
df_iqr_level['level'].plot( )
```



II. 이상치 확인 및 제거

- 이상치 제거
  - [그림 8-15]는 이상치를 제거하기 전 지하수위의 시계열 그래프. 비교하여 살펴보면 가장 눈에 띄었던 2013년의 이상치가 제거되었고 연중 변화가 명확히 드러나게 되었음.

[코드 8-27] 실행결과

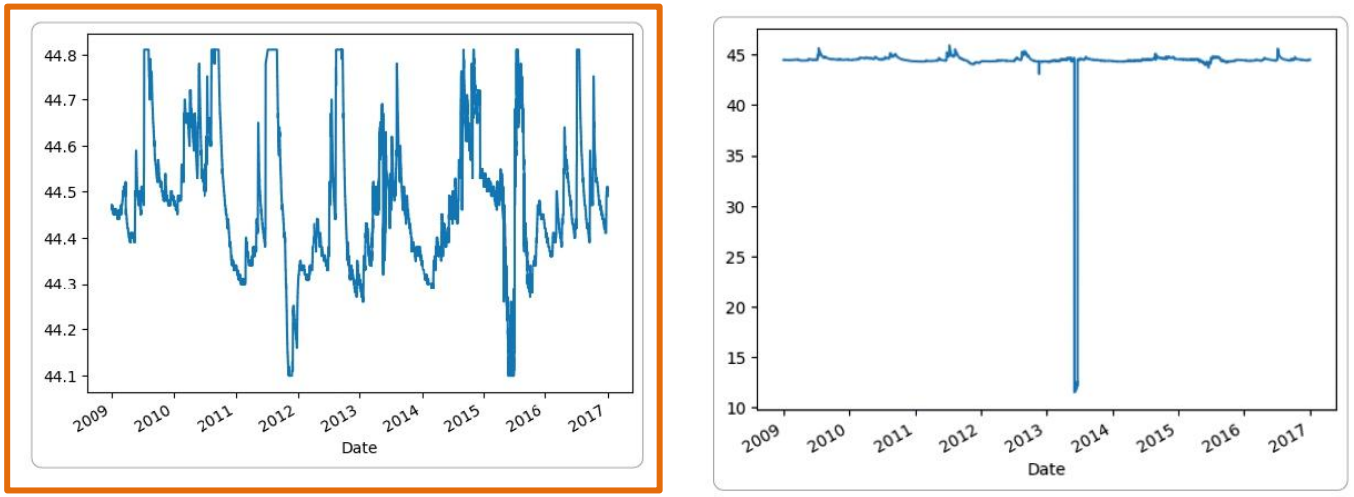


그림 8-15 이상치 제거 전 지하수위 시계열

II. 이상치 확인 및 제거

- 이상치 제거
  - 사분위수를 quantile( ) 함수로 구하면 상한값, 하한값, 이상치 개수까지 계산할 수 있음.

[코드 8-28] 지하수온 IQR과 이상치 개수

```
#사분위수와 IQR 구하기
q3_temp = df['temp'].quantile(q=0.75)
q1_temp = df['temp'].quantile(q=0.25)
iqr_temp = q3_temp-q1_temp
print('IQR:', iqr_temp)
#지하수온 상한값, 하한값, 이상치 개수 구하기
upper_temp = q3_temp + iqr_temp * 1.5
lower_temp = q1_temp - iqr_temp * 1.5
print(upper_temp, '/', lower_temp)
print((df['temp'] > upper_temp).sum( ))
print((df['temp'] < lower_temp).sum( ))
```

```
IQR: 0.10000000000000142
16.450000000000003 / 16.049999999999997
227
15
```

  - 상한값을 벗어나는 이상치는 227개이며 하한값을 벗어나는 이상치는 15개.

II. 이상치 확인 및 제거

- 이상치 제거
  - 전기전도도 IQR 구하고, 마찬가지로 상한값과 하한값을 변수에 저장하여 이상치 개수 확인.

[코드 8-29] 전기전도도 IQR과 이상치 개수

```
#사분위수와 IQR 구하기
q3_ec = df['EC'].quantile(q=0.75)
q1_ec = df['EC'].quantile(q=0.25)
iqr_ec = q3_ec - q1_ec
print('IQR:', iqr_ec)
#지하수온 상한값, 하한값, 이상치 개수 구하기
upper_ec = q3_ec + iqr_ec * 1.5
lower_ec = q1_ec - iqr_ec * 1.5
print(upper_ec, '/', lower_ec)
print((df['EC'] > upper_ec).sum( ))
print((df['EC'] < lower_ec).sum( ))
```

```
IQR: 20.0
772.0 / 692.0
831
788
```

II. 이상치 확인 및 제거

- 이상치 제거
  - 지하수온, 지하수위, 지하수 전기전도도의 이상치를 한꺼번에 제거.
  - 특성 수가 여러 개일 때는 이상치를 한꺼번에 제거해야 데이터 손실을 막을 수 있음.

[코드 8-30] 이상치를 한꺼번에 제거

```
df_iqr = df[(df['temp'] < upper_temp) & (df['temp'] > lower_temp) &\
          (df['level'] < upper_level) & (df['level'] > lower_level) &\
          (df['EC'] < upper_ec) & (df['EC'] > lower_ec)]
```

II. 이상치 확인 및 제거

- 이상치 제거 결과 확인
  - 통계량 및 그래프를 출력하여 이상치 제거 전과 비교.

[코드 8-31] 이상치 제거 후 통계량

```
df_iqr.describe( )
```

	temp	level	EC
count	61814.000000	61814.000000	61814.000000
mean	16.256793	44.459701	731.698612
std	0.089692	0.128256	13.882905
min	16.100000	44.100000	693.000000
25%	16.200000	44.370000	722.000000
50%	16.300000	44.450000	730.000000
75%	16.300000	44.520000	742.000000
max	16.400000	44.810000	771.000000

	temp	level	EC
count	69548.000000	69548.000000	69548.000000
mean	16.262491	44.295353	727.024027
std	0.089778	2.539464	51.544590
min	15.800000	11.530000	13.000000
25%	16.200000	44.370000	722.000000
50%	16.300000	44.460000	730.000000
75%	16.300000	44.550000	742.000000
max	16.500000	45.900000	778.000000

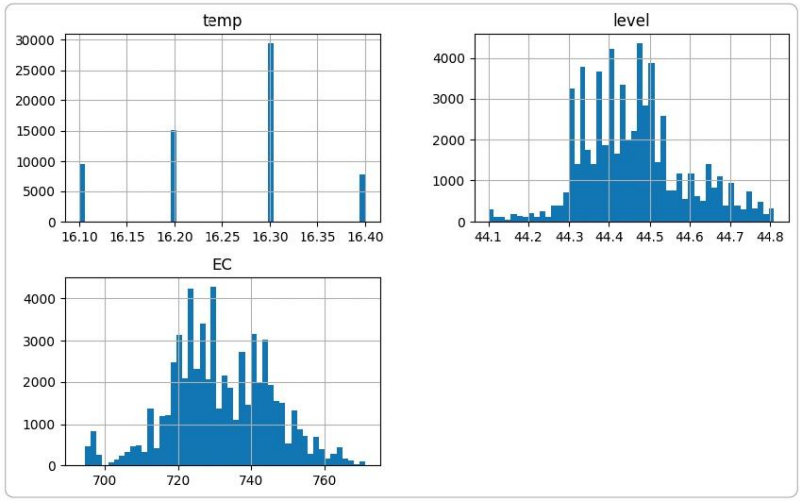
  - 오른쪽 [코드 8-20]의 실행 결과와 비교하면 이상치가 제거된 만큼 데이터 개수가 줄었고 관련 통계가 바뀌었으며 산포가 줄었음.

II. 이상치 확인 및 제거

- 이상치 제거 결과 확인

[코드 8-32] 이상치 제거 후 히스토그램

```
df_iqr.hist(bins=50, figsize=(10,6))
plt.show( )
```



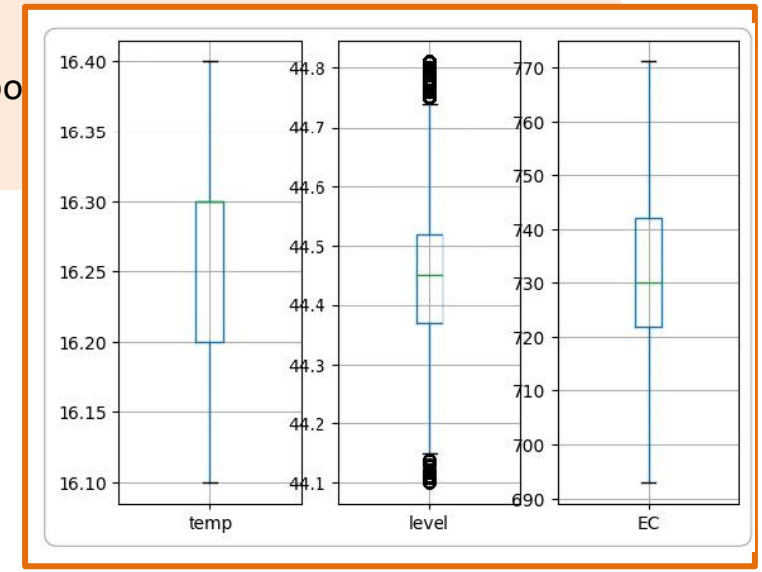
– 이상치를 제거하고 나니 정규분포와 가까우며 한쪽으로 치우치지 않게 됨.

II. 이상치 확인 및 제거

- 이상치 제거 결과 확인

[코드 8-33] 이상치 제거 후 상자 그래프

```
plt.subplot(1, 3, 1)
df_iqr.boxplot(column='temp', return_type='both')
plt.subplot(1, 3, 2)
df_iqr.boxplot(column='level', return_type='both')
plt.subplot(1, 3, 3)
df_iqr.boxplot(column='EC', return_type='bo
plt.show( )
```



– 지하수온과 지하수 전기전도도는 이상치가 모두 제거되어 그래프의 상하 균형이 맞고 수염 바깥쪽이 정리됨.

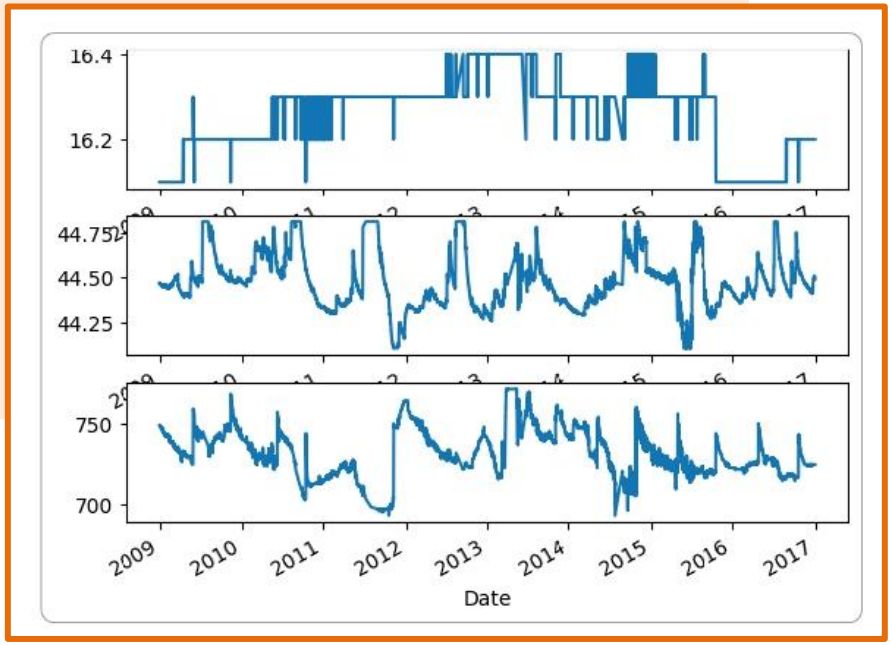
– 다만 지하수위는 상한값 위와 하한값 아래에 이상치가 조금씩 나타남.

II. 이상치 확인 및 제거

- 이상치 제거 결과 확인

[코드 8-34] 이상치 제거 후 추이 그래프

```
plt.subplot(3, 1, 1)
df_iqr['temp'].plot( )
plt.subplot(3, 1, 2)
df_iqr['level'].plot( )
plt.subplot(3, 1, 3)
df_iqr['EC'].plot( )
plt.show( )
```



- 세 변수 모두 시계열에 이상치가 드러나지 않고 안정적인 그래프가 나타남.
- 지하수온은 연중 변화가 거의 없음.
- 지하수위와 전기전도도의 시계열 그래프에서도 이상치가 제거된 상태로, 계절 변화와 연도별 산포만 드러남.

II. 이상치 확인 및 제거

- 정제된 데이터 저장
  - 정제된 데이터를 나중에도 사용할 수 있게 CSV 파일로 저장.

[코드 8-35] CSV 파일로 저장

```
df_iqr.to_csv('대전지하수_정제.csv', encoding='cp949')
```

III. 표준화와 정규화

- 파일 불러오기
  - 정제된 데이터 파일을 불러와 열별로 데이터의 범위를 확인.

[코드 8-36] 파일 불러오기

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df = pd.read_csv('./대전지하수_정제.csv')
df.head( )
```

	Date	temp	level	EC
0	2015-01-01 00:00:00	16.3	44.52	736
1	2015-01-01 01:00:00	16.3	44.52	736
2	2015-01-01 02:00:00	16.3	44.52	736
3	2015-01-01 03:00:00	16.3	44.52	736
4	2015-01-01 04:00:00	16.3	44.52	736

III. 표준화와 정규화

- 파일 불러오기
  - 두 열을 선택하여 스케일링.

[코드 8-37] 스케일링할 열 확인

```
df = df[['level', 'EC']]
df.describe( )
```

  - descibe( ) 함수로 지하수위와 지하수 전기전도도 데이터의 통계량을 나타내기.
  - 지하수 전기전도도는 지하수위보다 100배 이상으로 산포가 크다고 할 수 있음.

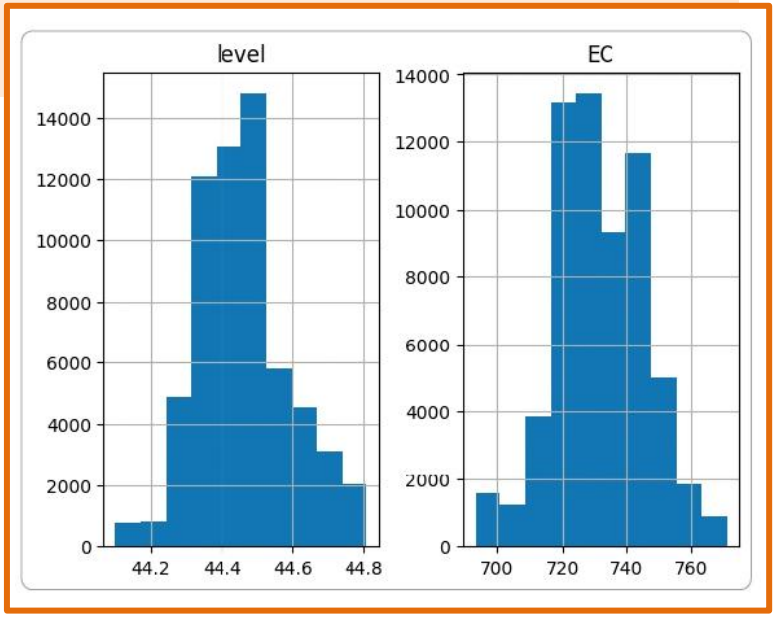
	level	EC
count	61814.000000	61814.000000
mean	44.459701	731.698612
std	0.128256	13.882905
min	44.100000	693.000000
25%	44.370000	722.000000
50%	44.450000	730.000000
75%	44.520000	742.000000
max	44.810000	771.000000

III. 표준화와 정규화

- 파일 불러오기

[코드 8-38] 히스토그램

```
df.hist( )
plt.show( )
```



- 두 변수의 최댓값과 최솟값을 비교하면 EC의 범위가 훨씬 넓음.
- 전기전도도 값이 지하수 위 값보다 15배 이상 크기 때문에 당연.
- 변수의 범위가 많이 다른 것을 고려하지 않으면 전기전도도의 산포도가 지하수위의 산포도보다 크다고 오해할 수 있음.

III. 표준화와 정규화

- 표준화
  - 판다스에서는 데이터프레임 열을 통째로 사칙연산할 수 있어 계산하기가 용이.

[코드 8-39] 사본 생성

```
df1 = df.copy( )
df1.head(3)
```

	level	EC
0	44.52	736
1	44.52	736
2	44.52	736

  - 데이터프레임 df1은 df와 동일한 내용, df1을 변경하더라도 원본 df에는 영향을 주지 않음.

III. 표준화와 정규화

- 표준화

[코드 8-40] 표준화

```
df1['level_z_score'] = (df1['level'] - df1['level'].mean( )) / df1['level'].std( )
df1['EC_z_score'] = (df1['EC'] - df1['EC'].mean()) / df1['EC'].std( )
df1.head(3)
```

	level	EC	level_z_score	EC_z_score
0	44.52	736	0.470145	0.309833
1	44.52	736	0.470145	0.309833
2	44.52	736	0.470145	0.309833

– 지하수위와 전기전도도 각각 표준화한 열이 추가됨.

III. 표준화와 정규화

- 표준화
  - 통계량과 히스토그램으로 표준화가 잘 수행되었는지 확인.

[코드 8-41] 표준화 후 통계량

```
df1.describe( )
```

	level	EC	level_z_score	EC_z_score
count	61814.000000	61814.000000	6.181400e+04	6.181400e+04
mean	44.459701	731.698612	1.324207e-14	-2.059877e-15
std	0.128256	13.882905	1.000000e+00	1.000000e+00
min	44.100000	693.000000	-2.804553e+00	-2.787501e+00
25%	44.370000	722.000000	-6.993901e-01	-6.986011e-01
50%	44.450000	730.000000	-7.563803e-02	-1.223528e-01
75%	44.520000	742.000000	4.701450e-01	7.420196e-01
max	44.810000	771.000000	2.731246e+00	2.830920e+00

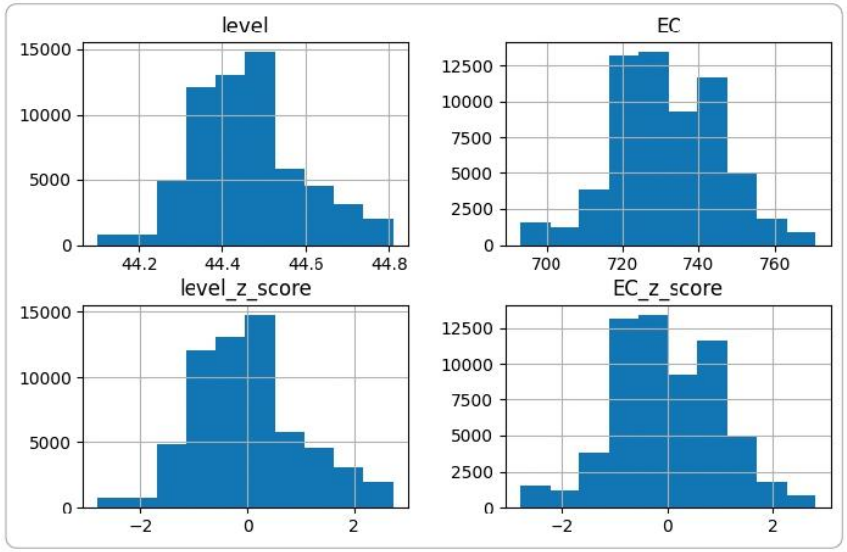
  - 두 변수의 최댓값과 최솟값 범위도 비슷해짐.

III. 표준화와 정규화

- 표준화

[코드 8-42] 표준화 후 히스토그램

```
df1.hist(figsize=(8,5))
plt.show( )
```



– 표준화 후 데이터가 평균 0을 중심으로 비슷한 범위에 분포.

III. 표준화와 정규화

- 표준화
  - 열 이름 대신 데이터프레임 변수 이름을 그대로 입력하면 한꺼번에 여러 열을 계산할 수 있음.

[코드 8-43] 데이터프레임 이름으로 표준화

```
df2 = df.copy( )
df2_standard = (df2-df2.mean( )) / df2.std( )
df2_standard.head(3)
```

	level	EC
0	0.470145	0.309833
1	0.470145	0.309833
2	0.470145	0.309833

III. 표준화와 정규화

- 정규화
  - 정규화는 데이터의 최댓값과 최솟값을 이용하여 0에서 1사이의 값으로 변환하는 전처리 과정.

[코드 8-44] 데이터 사본 정규화

```
df3 = df.copy( )

df3['level_minmax'] = (df3['level']-df3['level'].min()) / (df3['level'].max( )\
                                                -df3['level'].min( ))
df3['EC_minmax'] = (df3['EC'] - df3['EC'].min( )) / (df3['EC'].max( ) -
df3['EC'].min( ))
df3.head(3)
```

	level	EC	level_minmax	EC_minmax
0	44.52	736	0.591549	0.551282
1	44.52	736	0.591549	0.551282
2	44.52	736	0.591549	0.551282

III. 표준화와 정규화

- 정규화
  - 통계량과 히스토그램으로 정규화가 잘 수행되었는지 확인.

[코드 8-45] 정규화 후 통계량

```
df3.describe( )
```

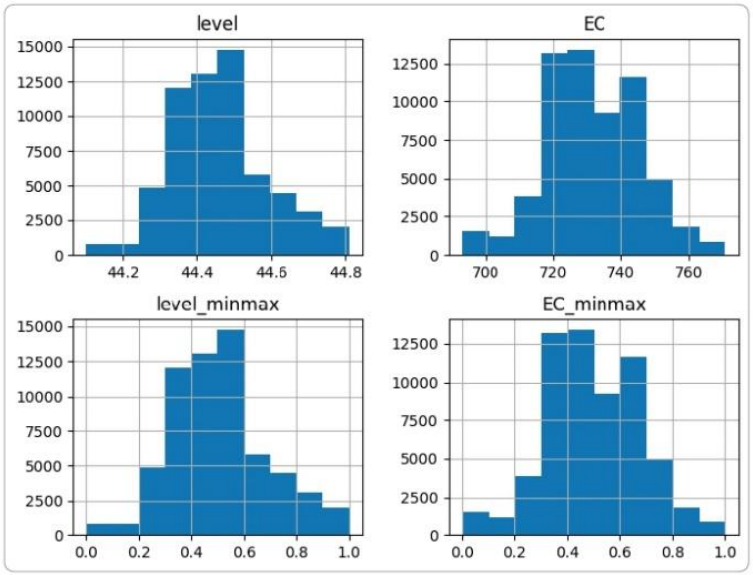
	level	EC	level_minmax	EC_minmax
count	61814.000000	61814.000000	61814.000000	61814.000000
mean	44.459701	731.698612	0.506621	0.496136
std	0.128256	13.882905	0.180642	0.177986
min	44.100000	693.000000	0.000000	0.000000
25%	44.370000	722.000000	0.380282	0.371795
50%	44.450000	730.000000	0.492958	0.474359
75%	44.520000	742.000000	0.591549	0.628205
max	44.810000	771.000000	1.000000	1.000000

III. 표준화와 정규화

- 정규화
  - 통계량과 히스토그램으로 정규화가 잘 수행되었는지 확인.

[코드 8-46] 정규화 후 통계량

```
df3.hist(figsize=(8,6))
plt.show( )
```



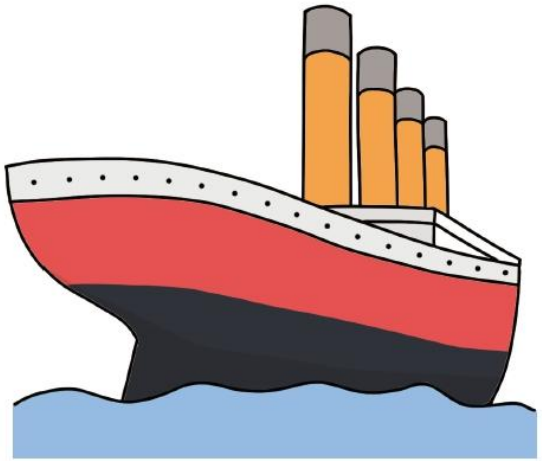
실전분석 타이타닉호 탑승자 데이터 전처리

[문제]

타이타닉호 침몰은 영화로도 제작된 유명한 사건입니다. 1912년 4월 10일 유람선 타이타닉호가 영국 사우샘프턴을 떠나 미국 뉴욕으로 향하는 첫 항해 중 빙산과 충돌하여 침몰했습니다. 당시 배에 2,200여 명이 승선하였으나 그 중 1,500여 명이 사망하였습니다.

타이타닉호 탑승자 중 생존자와 사망자를 집계한 가상의 데이터가 공개되어 있습니다. 앞으로 데이터 분석과 머신러닝을 공부한다면 이 데이터를 여러 번 보게 될 것입니다.

이번 예제에서는 타이타닉 탑승자 데이터에 결측치가 있는지 살펴보고 요금 열의 이상치를 확인해 봅시다.



[해결]

1. 데이터를 읽어와 행과 열 수를 확인. 이번에는 편의를 위해 이 책의 부록 예제소스로 제공된 CSV 파일 이용.

```
import pandas as pd
df = pd.read_csv('./titanic.csv')
df.shape
```

```
(891, 12)
```

승객이 891명이고 열은 12개.

2. snull( ) 함수로 각 열의 결측치 수를 확인.

```
df.isnull( ).sum( )
```

```
PassengerId 0
Survived 0
Pclass 0
Name 0
Sex 0
Age 177
SibSp 0
Parch 0
Ticket 0
Fare 0
Cabin 687
Embarked 2
```

나이 177건, 방 호수 687건, 탑승지 2건이 누락.

[해결]

3. 타이타닉 호는 초호화 유람선이라서 요금이 다른 승객들에 비해 매우 높은 승객이 있었을 것으로 예상됨. 요금 열에서 이상치 개수를 확인.

```
q1 = df['Fare'].quantile(.25)
q3 = df['Fare'].quantile(.75)
IQR = q3 - q1
print('하한값:', q1 - 1.5 * IQR, '상한값:', q3 + 1.5 * IQR)

out1 = df[df['Fare'] < (q1 - 1.5 * IQR)]
out2 = df[df['Fare'] > (q3 + 1.5 * IQR)]
len(out1), len(out2)
```

```
하한값: -26.724 상한값: 65.6344
(0, 116)
```

하한값보다 작은 이상치는 없으나 상한값보다 큰 이상치는 116개나 있음. 예상한 것과 일치함,

4. snull( ) 함수로 각 열의 결측치 수를 확인.

```
sum(out2['Sex'] == 'male')
```

```
46
```

요금을 특별히 많이 낸 승객 116명 중 46명이 남성이고 나머지 70명은 여성.