```
# 판다스 공식 문서 : https://pandas.pydata.org/docs/
# 라이브러리 최초 설치 시 느낌표를 앞에 붙임 (ex. !pip install pandas)
```

```
!pip install pandas
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (2.0.3)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.1)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.25.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
```

데이터 다루기 기본

0) 판다스 자료형

- Series()
- DataFrame()

+ 코드 + 텍스트

```
import pandas as pd
pd.Series([1,2,3], index = ['a','b','c'])
```

```
a    1
b    2
c    3
dtype: int64
```

```
import pandas as pd
data = pd.DataFrame({'이름' : ['홍길동','홍길산','홍길영'], '나이' : [10,20,30]})
```

```
data
```

	이름	나이
0	홍길동	10
1	홍길산	20
2	홍길영	30

1) 데이터 불러오기

- from ~ import 문
- import ~ as 문
- pd.read_csv('file', encoding = 'cp949')

```
from sklearn.preprocessing import MinMaxScaler
```

```
import pandas as pd
data = pd.read_csv('./sample_data/california_housing_test.csv',encoding = 'cp949')
```

2) 데이터 살펴보기

- df.shape
- df.info()
- df.describe()
- df.head()
- df.tail()
- df.unique()
- df.value_counts()

```
import seaborn as sns
df = sns.load_dataset('titanic')
```

```
print('행의 수 : ', df.shape[0])
print('열의 수 : ', df.shape[1])
```

```
행의 수 :  891
열의 수 :  15
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype   
---  ------       --------------  -----   
 0   survived     891 non-null    int64   
 1   pclass       891 non-null    int64   
 2   sex          891 non-null    object  
 3   age          714 non-null    float64 
 4   sibsp        891 non-null    int64   
 5   parch        891 non-null    int64   
 6   fare         891 non-null    float64 
 7   embarked     889 non-null    object  
 8   class        891 non-null    category
 9   who          891 non-null    object  
 10  adult_male   891 non-null    bool    
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object  
 13  alive        891 non-null    object  
 14  alone        891 non-null    bool    
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

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

```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

```
df.describe()
```

	survived	pclass	age	sibsp	parch	fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

```
df.tail(12)
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
879	1	1	female	56.0	0	1	83.1583	C	First	woman	False	C	Cherbourg	yes	False
880	1	2	female	25.0	0	1	26.0000	S	Second	woman	False	NaN	Southampton	yes	False
881	0	3	male	33.0	0	0	7.8958	S	Third	man	True	NaN	Southampton	no	True
882	0	3	female	22.0	0	0	10.5167	S	Third	woman	False	NaN	Southampton	no	True
883	0	2	male	28.0	0	0	10.5000	S	Second	man	True	NaN	Southampton	no	True
884	0	3	male	25.0	0	0	7.0500	S	Third	man	True	NaN	Southampton	no	True
885	0	3	female	39.0	0	5	29.1250	Q	Third	woman	False	NaN	Queenstown	no	False
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True	NaN	Southampton	no	True
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	B	Southampton	yes	True
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False	NaN	Southampton	no	False
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	C	Cherbourg	yes	True
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True	NaN	Queenstown	no	True

```
df['survived'].unique()
```

```
array([0, 1])
```

```
df['pclass'].value_counts()
```

```
pclass
3    491
1    216
2    184
Name: count, dtype: int64
```

3) 결측치 확인 및 처리

- isnull()
- fillna()
- dropna(inplace = True)
- drop_duplicates() 중복행삭제

```
df1 = df.copy()
df2 = df.copy()
```

```
df1['age'].fillna(df1['age'].mean(), inplace = True)
```

```
df2.dropna(inplace = True)
```

```
df2
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
6	0	1	male	54.0	0	0	51.8625	S	First	man	True	E	Southampton	no	True
10	1	3	female	4.0	1	1	16.7000	S	Third	child	False	G	Southampton	yes	False
11	1	1	female	58.0	0	0	26.5500	S	First	woman	False	C	Southampton	yes	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
871	1	1	female	47.0	1	1	52.5542	S	First	woman	False	D	Southampton	yes	False
872	0	1	male	33.0	0	0	5.0000	S	First	man	True	B	Southampton	no	True
879	1	1	female	56.0	0	1	83.1583	C	First	woman	False	C	Cherbourg	yes	False
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	B	Southampton	yes	True
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	C	Cherbourg	yes	True

182 rows × 15 columns

```
df.drop_duplicates(subset = ['survived','pclass'])
```

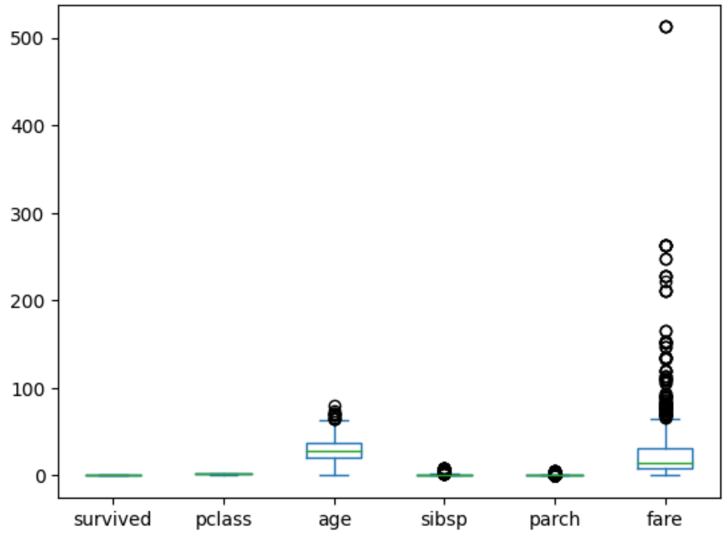
	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
6	0	1	male	54.0	0	0	51.8625	S	First	man	True	E	Southampton	no	True
9	1	2	female	14.0	1	0	30.0708	C	Second	child	False	NaN	Cherbourg	yes	False
20	0	2	male	35.0	0	0	26.0000	S	Second	man	True	NaN	Southampton	no	True

4) 이상치 확인 및 조정

- box-plot
- IQR = Q3 - Q1
- Q1 − 1.5 * IQR 미만이나 Q3 + 1.5 * IQR 초과데이터를 이상치로 탐지

```
import seaborn as sns
df = sns.load_dataset('titanic')
df.info()
df.plot(kind = 'box')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
<Axes: >
```

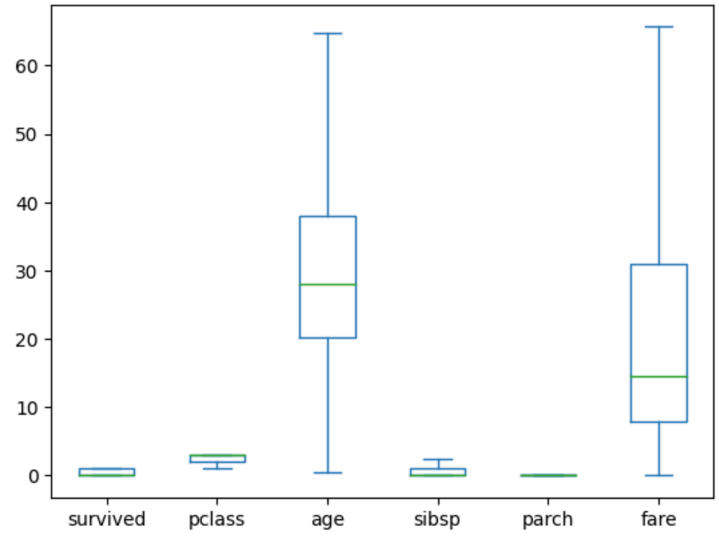


```
numeric_df = df.select_dtypes(include = [int, float])
numeric_df.columns # 6개
numeric = list(numeric_df.columns)

for col in numeric:
  IQR = df[col].quantile(0.75) - df[col].quantile(0.25)
  upper_bound = df[col].quantile(0.75) + IQR*1.5
  lower_bound = df[col].quantile(0.25) - IQR*1.5
  df[col] = df[col].clip(lower_bound, upper_bound)

# 이상치가 제거된 DataFrame의 박스 플롯
df.plot(kind = 'box')
```

```
<Axes: >
```



5) 데이터 붙이기

- concat(), merge()

```
import pandas as pd
df1 = pd.read_csv('신상정보1.csv', encoding = 'cp949')
df2 = pd.read_csv('신상정보2.csv', encoding='cp949')
print(df1)
print('-----')
print(df2)
```

```
    이름  나이    키
0  김철수  30  177
1  이영희  20  165
2  박민지  24  158
3  정소라  21  163
-----
    이름      과목  학점
0  김철수  통계학개론   A
1  김철수   재료공학  B-
2  김철수   이산수학  A+
3  정소라   재료공학   C
4  유바다   이산수학  B+
5  이영희  통계학개론  A+
```

```
concat_0 = pd.concat([df1, df2], axis = 0, ignore_index = True)
```

```
pd.concat([df1, df2], axis = 1)
```

	이름	나이	키	이름	과목	학점
0	김철수	30.0	177.0	김철수	통계학개론	A
1	이영희	20.0	165.0	김철수	재료공학	B-
2	박민지	24.0	158.0	김철수	이산수학	A+
3	정소라	21.0	163.0	정소라	재료공학	C
4	NaN	NaN	NaN	유바다	이산수학	B+
5	NaN	NaN	NaN	이영희	통계학개론	A+

```
# df1 => 민지 , df2 = 바다
pd.merge(df1, df2, how = 'right')
```

	이름	나이	키	과목	학점
0	김철수	30.0	177.0	통계학개론	A
1	김철수	30.0	177.0	재료공학	B-
2	김철수	30.0	177.0	이산수학	A+
3	정소라	21.0	163.0	재료공학	C
4	유바다	NaN	NaN	이산수학	B+
5	이영희	20.0	165.0	통계학개론	A+

```
df2.columns = ['성함','과목','학점']
```

```
pd.merge(df1, df2, how = 'right', left_on = '이름', right_on = '성함')
```

	이름	나이	키	성함	과목	학점
0	김철수	30.0	177.0	김철수	통계학개론	A
1	김철수	30.0	177.0	김철수	재료공학	B-
2	김철수	30.0	177.0	김철수	이산수학	A+
3	정소라	21.0	163.0	정소라	재료공학	C
4	NaN	NaN	NaN	유바다	이산수학	B+
5	이영희	20.0	165.0	이영희	통계학개론	A+

6) 그룹으로 묶어서 보기

- groupby()

```
import seaborn as sns
df = sns.load_dataset('iris')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

```
df.groupby('species').mean()
```

	sepal_length	sepal_width	petal_length	petal_width
species				
setosa	5.006	3.428	1.462	0.246
versicolor	5.936	2.770	4.260	1.326
virginica	6.588	2.974	5.552	2.026

```
df.groupby('species').agg({'sepal_length' : 'median',
                           'sepal_width' : 'var',
                           'petal_length':'mean',
                           'petal_width' : 'max'})
```

	sepal_length	sepal_width	petal_length	petal_width
species				
setosa	5.0	0.143690	1.462	0.6
versicolor	5.9	0.098469	4.260	1.8
virginica	6.5	0.104004	5.552	2.5

7) 행/열

7-1) 행/열 선택 및 조건 필터링

- .iloc[]
- .loc[]
- &, |

7-2) 열 이름 변경

- df.rename()
- df.columns = ['new','new2']
- df.columns = df.columns.str.replace('기존문자' , '대체문자')

7-3) 열 삭제

- df.drop()

```
df.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

```
df.iloc[:,0]
```

```
0      5.1
1      4.9
2      4.7
3      4.6
4      5.0
      ...
145    6.7
146    6.3
147    6.5
148    6.2
149    5.9
Name: sepal_length, Length: 150, dtype: float64
```

```
# setosa라는 품종만 선택해서 가져오기
condition = (df['species'] == 'setosa')
df.loc[condition , 'petal_length']
```

```
0     1.4
1     1.4
2     1.3
3     1.5
4     1.4
5     1.7
6     1.4
7     1.5
8     1.4
9     1.5
10    1.5
11    1.6
12    1.4
13    1.1
14    1.2
15    1.5
16    1.3
17    1.4
18    1.7
```

```
19    1.5
20    1.7
21    1.5
22    1.0
23    1.7
24    1.9
25    1.6
26    1.6
27    1.5
28    1.4
29    1.6
30    1.6
31    1.5
32    1.5
33    1.4
34    1.5
35    1.2
36    1.3
37    1.4
38    1.3
39    1.5
40    1.3
41    1.3
42    1.3
43    1.6
44    1.9
45    1.4
46    1.6
47    1.4
48    1.5
49    1.4
Name: petal_length, dtype: float64
```

```
# 품종이 setosa이면서, petal_length가 1.4이상인 것만 가져오기
condition2 = (df['species'] == 'setosa') | (df['petal_length'] >= 1.4)
df.loc[condition2]
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
5	5.4	3.9	1.7	0.4	setosa
6	4.6	3.4	1.4	0.3	setosa
7	5.0	3.4	1.5	0.2	setosa
8	4.4	2.9	1.4	0.2	setosa
9	4.9	3.1	1.5	0.1	setosa
10	5.4	3.7	1.5	0.2	setosa
11	4.8	3.4	1.6	0.2	setosa
12	4.8	3.0	1.4	0.1	setosa
15	5.7	4.4	1.5	0.4	setosa
17	5.1	3.5	1.4	0.3	setosa
18	5.7	3.8	1.7	0.3	setosa
19	5.1	3.8	1.5	0.3	setosa
20	5.4	3.4	1.7	0.2	setosa
21	5.1	3.7	1.5	0.4	setosa
23	5.1	3.3	1.7	0.5	setosa
24	4.8	3.4	1.9	0.2	setosa
25	5.0	3.0	1.6	0.2	setosa
26	5.0	3.4	1.6	0.4	setosa
27	5.2	3.5	1.5	0.2	setosa
28	5.2	3.4	1.4	0.2	setosa
29	4.7	3.2	1.6	0.2	setosa
30	4.8	3.1	1.6	0.2	setosa
31	5.4	3.4	1.5	0.4	setosa
32	5.2	4.1	1.5	0.1	setosa
33	5.5	4.2	1.4	0.2	setosa
34	4.9	3.1	1.5	0.2	setosa
37	4.9	3.6	1.4	0.1	setosa

```
'''
df.rename()
df.columns = df.columns.str.replace('기존문자' , '대체문자')
'''
df.rename(columns = {'species' : '품종'}, inplace = True)
```

```
df.columns = df.columns.str.replace('품종', 'species')
```

47	4.6	3.2	1.4	0.2	setosa

```
df
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

8) 정렬

- sort_index()
- sort_index(axis = 1)
- sort_values(by='컬럼명')
- sort_values(by=['컬럼명', '컬럼명2'])

```
df.sort_index(ascending = False)
```

	sepal_length	sepal_width	petal_length	petal_width	species
149	5.9	3.0	5.1	1.8	virginica
148	6.2	3.4	5.4	2.3	virginica
147	6.5	3.0	5.2	2.0	virginica
146	6.3	2.5	5.0	1.9	virginica
145	6.7	3.0	5.2	2.3	virginica
...	...	...	...	...	...
4	5.0	3.6	1.4	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
0	5.1	3.5	1.4	0.2	setosa

150 rows × 5 columns

```
df.sort_values(by = 'sepal_length', ascending = False)
```

	sepal_length	sepal_width	petal_length	petal_width	species
131	7.9	3.8	6.4	2.0	virginica
117	7.7	3.8	6.7	2.2	virginica
135	7.7	3.0	6.1	2.3	virginica
122	7.7	2.8	6.7	2.0	virginica
118	7.7	2.6	6.9	2.3	virginica
...	...	...	...	...	...
41	4.5	2.3	1.3	0.3	setosa
42	4.4	3.2	1.3	0.2	setosa
38	4.4	3.0	1.3	0.2	setosa
8	4.4	2.9	1.4	0.2	setosa
13	4.3	3.0	1.1	0.1	setosa

150 rows × 5 columns

9) 자료형 변경

- df.dtypes
- df['컬럼명'] = df['컬럼명'].astype('타입')
- df.convert_dtypes() 가장 적절한 dtype으로 변경

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

```
df['sepal_length'].dtypes
df['sepal_length'] = df['sepal_length'].astype('int')
df
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5	3.5	1.4	0.2	setosa
1	4	3.0	1.4	0.2	setosa
2	4	3.2	1.3	0.2	setosa
3	4	3.1	1.5	0.2	setosa
4	5	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6	3.0	5.2	2.3	virginica
146	6	2.5	5.0	1.9	virginica
147	6	3.0	5.2	2.0	virginica
148	6	3.4	5.4	2.3	virginica
149	5	3.0	5.1	1.8	virginica

150 rows × 5 columns

```
conv_df = df.convert_dtypes()
conv_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    Int64  
 1   sepal_width   150 non-null    Float64
 2   petal_length  150 non-null    Float64
 3   petal_width   150 non-null    Float64
 4   species       150 non-null    string 
dtypes: Float64(3), Int64(1), string(1)
memory usage: 6.6 KB
```

10) 문자열 데이터 다루기

- df['컬럼명'].str.split("기준문자", expand = True)

```
import pandas as pd
df = pd.read_csv('./한국지역난방공사_날씨정보.csv', encoding = 'cp949')
```

```
df.head()
df['연도'] = df['일자'].str.split('-').str.get(0)
df['월'] =df['일자'].str.split('-').str.get(1)
df['일'] =df['일자'].str.split('-').str.get(2)
df
```

	일자	시간	날씨	연도	월	일
0	2017-02-14	1	맑음	2017	02	14
1	2017-02-14	2	맑음	2017	02	14
2	2017-02-14	5	맑음	2017	02	14
3	2017-02-14	9	맑음	2017	02	14
4	2017-02-14	10	맑음	2017	02	14
...	...	...	...	...	...	...
46090	2023-06-26	1	비	2023	06	26
46091	2023-06-26	2	비	2023	06	26
46092	2023-06-26	3	비	2023	06	26
46093	2023-06-26	4	비	2023	06	26
46094	2023-06-26	5	비	2023	06	26

46095 rows × 6 columns

```
import seaborn as sns
iris = sns.load_dataset('iris')
iris
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

```
iris['species']= iris['species'] + '$'
iris
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa$
1	4.9	3.0	1.4	0.2	setosa$
2	4.7	3.2	1.3	0.2	setosa$
3	4.6	3.1	1.5	0.2	setosa$
4	5.0	3.6	1.4	0.2	setosa$
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica$
146	6.3	2.5	5.0	1.9	virginica$
147	6.5	3.0	5.2	2.0	virginica$
148	6.2	3.4	5.4	2.3	virginica$
149	5.9	3.0	5.1	1.8	virginica$

150 rows × 5 columns

```
iris['species'] = iris['species'].str.split('$').str.get(0)
iris
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

11) 날짜 데이터 핸들링

- datetime 모듈
- dt 모듈

```
import pandas as pd
df = pd.read_csv('한국지역난방공사_날씨정보.csv', encoding = 'cp949', parse_dates = ['일자'])
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 46095 entries, 0 to 46094
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   일자      46095 non-null  datetime64[ns]
 1   시간      46095 non-null  int64
 2   날씨      46095 non-null  object
dtypes: datetime64[ns](1), int64(1), object(1)
memory usage: 1.1+ MB
```

```
df['year']= df['일자'].dt.year
df['month']= df['일자'].dt.month
df['day']= df['일자'].dt.day
df
```

	일자	시간	날씨	year	month	day
0	2017-02-14	1	맑음	2017	2	14
1	2017-02-14	2	맑음	2017	2	14
2	2017-02-14	5	맑음	2017	2	14
3	2017-02-14	9	맑음	2017	2	14
4	2017-02-14	10	맑음	2017	2	14
...	...	...	...	...	...	...
46090	2023-06-26	1	비	2023	6	26
46091	2023-06-26	2	비	2023	6	26
46092	2023-06-26	3	비	2023	6	26
46093	2023-06-26	4	비	2023	6	26
46094	2023-06-26	5	비	2023	6	26

46095 rows × 6 columns

```
date_str = '05/15/2023'
result = pd.to_datetime(date_str, format='%m/%d/%Y')

date_str2 = '2023-06-15 13:40:30'
result = pd.to_datetime(date_str2, format = '%Y-%m-%d %H:%M:%S')
result
```

```
Timestamp('2023-06-15 13:40:30')
```

12) 데이터 스케일링

- 정규화 : MinMaxScaler -> 최대/최소값이 각각 1,0이 되도록 스케일링
- 표준화 : StandardScaler -> 평균을 0,표준편차를 1로 만드는 과정

```
import seaborn as sns
import pandas as pd
titanic = sns.load_dataset('titanic')
titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

```
numeric_df = titanic.select_dtypes(include = ['int','float'])
numeric_df

from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler()
titanic1 = pd.DataFrame(sc.fit_transform(numeric_df), columns=numeric_df.columns)
titanic1.describe()
```

	survived	pclass	age	sibsp	parch	fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	0.654321	0.367921	0.065376	0.063599	0.062858
std	0.486592	0.418036	0.182540	0.137843	0.134343	0.096995
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.500000	0.247612	0.000000	0.000000	0.015440
50%	0.000000	1.000000	0.346569	0.000000	0.000000	0.028213
75%	1.000000	1.000000	0.472229	0.125000	0.000000	0.060508
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

```
from sklearn.preprocessing import StandardScaler
sc2 = StandardScaler()
titanic2 = pd.DataFrame(sc2.fit_transform(numeric_df), columns = numeric_df.columns)
titanic2.describe()
```

	survived	pclass	age	sibsp	parch	fare
count	8.910000e+02	8.910000e+02	7.140000e+02	8.910000e+02	8.910000e+02	8.910000e+02
mean	3.987333e-17	-8.772133e-17	2.388379e-16	4.386066e-17	5.382900e-17	3.987333e-18
std	1.000562e+00	1.000562e+00	1.000701e+00	1.000562e+00	1.000562e+00	1.000562e+00
min	-7.892723e-01	-1.566107e+00	-2.016979e+00	-4.745452e-01	-4.736736e-01	-6.484217e-01
25%	-7.892723e-01	-3.693648e-01	-6.595416e-01	-4.745452e-01	-4.736736e-01	-4.891482e-01
50%	-7.892723e-01	8.273772e-01	-1.170488e-01	-4.745452e-01	-4.736736e-01	-3.573909e-01
75%	1.266990e+00	8.273772e-01	5.718310e-01	4.327934e-01	-4.736736e-01	-2.424635e-02
max	1.266990e+00	8.273772e-01	3.465126e+00	6.784163e+00	6.974147e+00	9.667167e+00

13) 범주형 데이터 인코딩

```
titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

```
# 원-핫 인코딩
cat_df = titanic.select_dtypes(include = 'object')
pd.get_dummies(titanic,columns = cat_df.columns)
```

	survived	pclass	age	sibsp	parch	fare	class	adult_male	deck	alone	...	embarked_Q	embarked_S	who_child	who_ma
0	0	3	22.0	1	0	7.2500	Third	True	NaN	False	...	False	True	False	Tru
1	1	1	38.0	1	0	71.2833	First	False	C	False	...	False	False	False	Fal
2	1	3	26.0	0	0	7.9250	Third	False	NaN	True	...	False	True	False	Fal
3	1	1	35.0	1	0	53.1000	First	False	C	False	...	False	True	False	Fal
4	0	3	35.0	0	0	8.0500	Third	True	NaN	True	...	False	True	False	Tru
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
886	0	2	27.0	0	0	13.0000	Second	True	NaN	True	...	False	True	False	Tru
887	1	1	19.0	0	0	30.0000	First	False	B	True	...	False	True	False	Fal
888	0	3	NaN	1	2	23.4500	Third	False	NaN	False	...	False	True	False	Fal
889	1	1	26.0	0	0	30.0000	First	True	C	True	...	False	False	False	Tru
890	0	3	32.0	0	0	7.7500	Third	True	NaN	True	...	True	False	False	Tru

891 rows × 23 columns

```
#라벨인코딩
from sklearn.preprocessing import LabelEncoder
cat_df = titanic.select_dtypes(include = ['object','category'])

# 각 컬럼에 대한 LabelEncoder를 저장할 딕셔너리
encoders = {}
for col in cat_df.columns:
  encoder = LabelEncoder()
  titanic[col] = encoder.fit_transform(titanic[col])
  encoders[col] = encoder

titanic
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	1	22.0	1	0	7.2500	2	2	1	True	7	2	0	False
1	1	1	0	38.0	1	0	71.2833	0	0	2	False	2	0	1	False
2	1	3	0	26.0	0	0	7.9250	2	2	2	False	7	2	1	True
3	1	1	0	35.0	1	0	53.1000	2	0	2	False	2	2	1	False
4	0	3	1	35.0	0	0	8.0500	2	2	1	True	7	2	0	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	1	27.0	0	0	13.0000	2	1	1	True	7	2	0	True
887	1	1	0	19.0	0	0	30.0000	2	0	2	False	1	2	1	True
888	0	3	0	NaN	1	2	23.4500	2	2	2	False	7	2	0	False
889	1	1	1	26.0	0	0	30.0000	0	0	1	True	2	0	1	True
890	0	3	1	32.0	0	0	7.7500	1	2	1	True	7	1	0	True

891 rows × 15 columns

14) 데이터 분할

기계 학습 모델을 평가하고 일반화하기 위해 데이터를 두 그룹으로 나누는 데 사용

```
# train(0.7) / test(0.3)
condition = round(titanic.shape[0] * 0.7)
train = titanic.iloc[:condition]
train

test = titanic.iloc[condition :]
test
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
624	0	3	1	21.0	0	0	16.1000	2	2	1	True	7	2	0	True
625	0	1	1	61.0	0	0	32.3208	2	0	1	True	3	2	0	True
626	0	2	1	57.0	0	0	12.3500	1	1	1	True	7	1	0	True
627	1	1	0	21.0	0	0	77.9583	2	0	2	False	3	2	1	True
628	0	3	1	26.0	0	0	7.8958	2	2	1	True	7	2	0	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	1	27.0	0	0	13.0000	2	1	1	True	7	2	0	True
887	1	1	0	19.0	0	0	30.0000	2	0	2	False	1	2	1	True
888	0	3	0	NaN	1	2	23.4500	2	2	2	False	7	2	0	False
889	1	1	1	26.0	0	0	30.0000	0	0	1	True	2	0	1	True
890	0	3	1	32.0	0	0	7.7500	1	2	1	True	7	1	0	True

267 rows × 15 columns

```
X = titanic[['pclass']]
y = titanic['survived']
```

```
from sklearn.model_selection import train_test_split
X_train, X_test,y_train, y_test= train_test_split(X,y, random_state = 2021, stratify =y, test_size = 0.3)
```

```
X_train
X_test
```

```
     pclass
672       2
832       3
435       1
618       2
 49       3
...     ...
329       1
637       2
109       3
141       3
499       3
268 rows × 1 columns
```

15) 그래프 기초

- fig와 axes
- 수치형 그래프
    - 히스토그램, 박스플롯 등