데이터 분석 처리 과정

충북대학교 소프트웨어학과
류관희

목 차

01

데이터처리 과정

- 비즈니스 데이터 분석 사이클
- 테이터 분석 사이클
- 타이타닉 데이터 이해
- 데이터 전처리

02

데이터 정제

- 행과 열 제거
- 중복데이터 제거
- NULL 데이터 처리

03

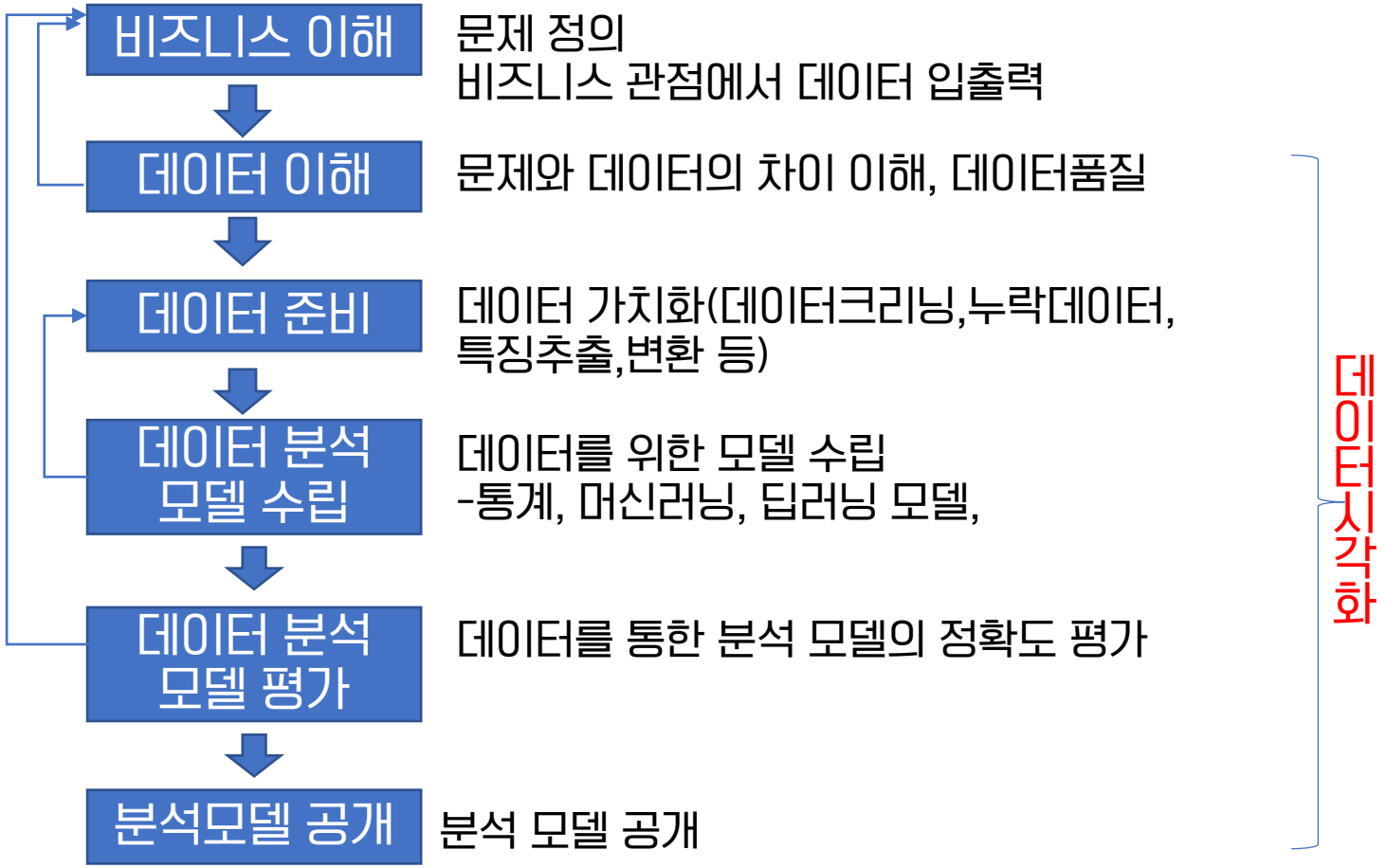
데이터 가공 과정

- 데이터 변화
- 데이터 정규화
- 이상한 데이터 제거

학습목표

- **이번 파트에서는 데이터 분석을 위해 다음과 같은 내용을 다룬다.**
  - 비즈니스 데이터 분석 사이클
  - 데이터 분석 사이클
  - 타이타닉 데이터 이해
  - 데이터 전처리

비즈니스 데이터 분석 사이클



데이터 분석 사이클

데이터수집
분석하고자 하는 데이터를 모으는 과정
-크롤링, ETL(Extract, transform, and load)
-로그수집, ftp, http, DB

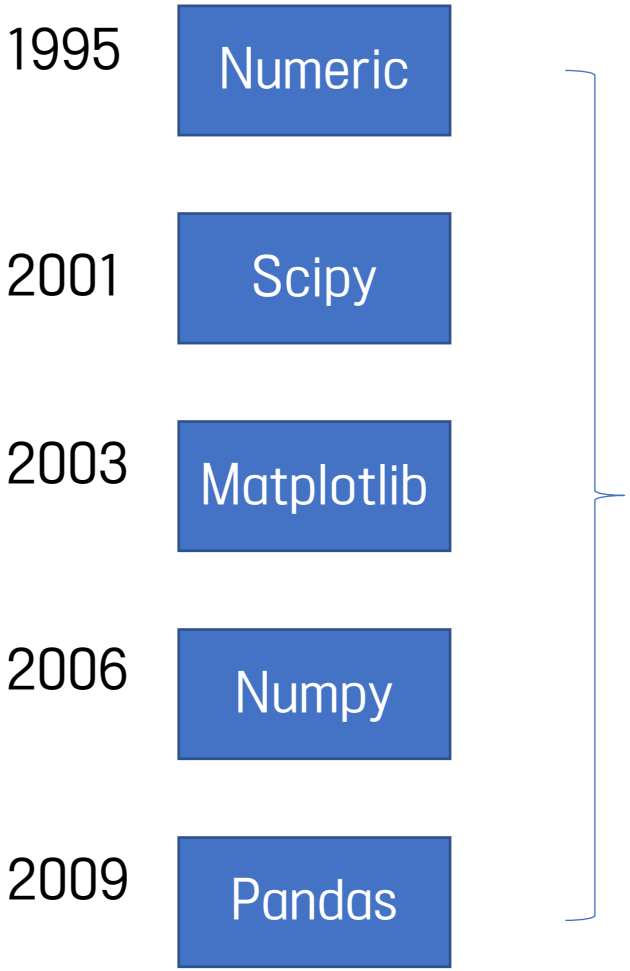
데이터 전처리
분석에 적합하게 데이터를 전처리하는 과정
-데이터 셋 확인 – 결측값 처리
-이상값 처리 – Feature Engineering

모델 기반 데이터 분석
데이터로부터 모델을 사용하여 패턴을 찾는 과정
통계분석:평균,분산,분포,추정 등
머신러닝:회귀분석,분류분석,연관분석,군집분석 등
딥러닝: CNN, RNN, LSTM 등

데이터분석 결과 평가
사용한 데이터 분석 모델의 평가

데이터시각화

파이션 패키지


1995
Numeric
2001
Scipy
2003
Matplotlib
2006
Numpy
2009
Pandas

타이타닉 선상 문제 정의

- 아래와 같은 조건에 따른 타이타닉 탑승 인원에 대한 생존율 분석
  - ✓ 생존여부(survived)
  - ✓ 객실등급(Pclass)
  - ✓ 이름, 성별, 나이,
  - ✓ 형제와배우자(SibSp)
  - ✓ 자식과부모(Parch)
  - ✓ 티켓
  - ✓ 요금
  - ✓ Cabin
  - ✓ 탑승한곳(Embarked)

타이타닉 데이터 이해


```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('fivethirtyeight')
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

```
data=pd.read_excel('titanic.xlsx')
data.head()
```

- survived : 생존 여부
- pclass : 승객의 클래스
- sex : 성별. male, female로 표기
- sibsp : 형제 혹은 자매의 수
- parch : 부모 혹은 자녀의 수
- fare : 탑승 요금
- embarked : 출발지의 고유 이니셜
- class : 선실의 클래스
- who : male, female을 man, woman으로 표기
- adult_male : 성인 남성 인지 아닌지 여부
- deck : 선실 고유 번호의 가장 앞자리 알파벳(A ~ G)
- embark_town : 출발지
- alive : 생존 여부 데이터를 yes 혹은 no로 표기
- alone : 가족이 없는 경우 True

타이타닉 데이터 이해

```
1 data=pd.read_excel('titanic.xlsx')
2 data
```

	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
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
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

891 rows × 15 columns

데이터 전처리

- 데이터의 특징
  - Null 데이터 여부 확인
  - 데이터 품질 점검

- 분석을 위한 의미있는 데이터 만들어야 함
  - 데이터 전처리 수행 시간 비율: 80%~90%
  - 실제로 데이터 분석 알고리즘 자체 수행 시간 비율: 10%~20%

- 데이터전처리: 분석하기 좋게 데이터를 고치는 모든 작업

데이터 전처리

- 데이터의 전체 정보 살펴 보기
- info()

```
1  data.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    object
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    object
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), float64(2), int64(4), object(7)
memory usage: 92.4+ KB
```

데이터 전처리

- null 데이터가 있는지 확인
- data,.isnull.sum()

```
1 data.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
```

문제풀이

- 비즈니스 데이터 분석 사이클을 설명하시오.
- 데이터 전처리를 위해 확인해야 하는 항목을 설명하시오.

요약

- 비즈니스 데이터 분석과 데이터분석 과정에 대해 공부하였음
- 타이타닉 데이터 예제를 통해 문제 정의하고 데이터를 이해하는 방법을 공부하였음.
- 데이터 전처리의 의미를 공부하였음

01

데이터처리 과정

- 비즈니스 데이터 분석 사이클
- 테이터 분석 사이클
- 타이타닉 데이터 이해
- 데이터 전처리

02

데이터 정제

- 행과 열 제거
- 중복데이터 제거
- NULL 데이터 처리

03

데이터 가공 과정

- 데이터 변화
- 데이터 정규화
- 이상한 데이터 제거

학습목표

- **이번 파트에서는 데이터 정제를 위해 다음과 같은 내용을 다룬다.**
  - 행과 열 제거
  - 중복데이터 제거
  - NULL 데이터 처리

데이터 변환

- columns
  - 데이터셋의 열 이름을 출력
    - titanic_data.columns

```
1  titanic_data.columns
```

```
Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',
       'embarked', 'who', 'adult_male', 'deck', 'embark_town', 'alive',
       'alone'],
      dtype='object')
```

데이터 변환

- rename()
  - dict를 통해 특정 또는 모든 열의 이름 바꾸기

```
1  titanic_data.rename(columns={'survived': 'SURVIVED'}, inplace=True)
2  titanic_data.columns
```

```
Index(['SURVIVED', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',
       'embarked', 'who', 'adult_male', 'deck', 'embark_town', 'alive',
       'alone'],
      dtype='object')
```

  - 시도하기
    - 모든 행 이름을 대문자로 바꾸어 보기 바람.

데이터 정제

- 행과 열을 제거하는 과정
- drop()
  - data.drop(columns='class', inplace=True)

```
1  data.drop(columns='class', inplace=True)
```

```
1  data
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	man	True	NaN	Southampton	no	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S	man	True	NaN	Southampton	no	True
887	1	1	female	19.0	0	0	30.0000	S	woman	False	B	Southampton	yes	True
888	0	3	female	NaN	1	2	23.4500	S	woman	False	NaN	Southampton	no	False
889	1	1	male	26.0	0	0	30.0000	C	man	True	C	Cherbourg	yes	True
890	0	3	male	32.0	0	0	7.7500	Q	man	True	NaN	Queenstown	no	True

891 rows × 14 columns

data.drop(columns=['Cabin', 'Embarked'], inplace=True)

- Task
  - 'Cabin', 'Embarked'열을 동시에 제거

데이터 정제

- 중복데이터 점검

```
1  ddd=pd.Series(['Dog', 'Cat', 'Dog', 'Cat', 'Bat'])
```

```
1  ddd.duplicated()
```

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

```
1  data.duplicated()
```

```
0      False
1      False
2      False
3      False
4      False
       ...
886     True
887    False
888    False
889    False
890    False
Length: 891, dtype: bool
```

데이터 정제

중복되는 데이터를 알아내기

titanic_data[titanic_data.duplicated()]

```
1 titanic_data[titanic_data.duplicated()]
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	who	adult_male	deck	embark_town	alive	alone
47	1	3	female	NaN	0	0	7.7500	Q	woman	False	NaN	Queenstown	yes	True
76	0	3	male	NaN	0	0	7.8958	S	man	True	NaN	Southampton	no	True
77	0	3	male	NaN	0	0	8.0500	S	man	True	NaN	Southampton	no	True
87	0	3	male	NaN	0	0	8.0500	S	man	True	NaN	Southampton	no	True
95	0	3	male	NaN	0	0	8.0500	S	man	True	NaN	Southampton	no	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
870	0	3	male	26.0	0	0	7.8958	S	man	True	NaN	Southampton	no	True
877	0	3	male	19.0	0	0	7.8958	S	man	True	NaN	Southampton	no	True
878	0	3	male	NaN	0	0	7.8958	S	man	True	NaN	Southampton	no	True
884	0	3	male	25.0	0	0	7.0500	S	man	True	NaN	Southampton	no	True
886	0	2	male	27.0	0	0	13.0000	S	man	True	NaN	Southampton	no	True

107 rows × 14 columns

데이터 정제

- 중복되는 행을 제거하는 명령
  - titanic_data.drop_duplicates(inplace = True)

```
1 titanic_data
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	man	True	NaN	Southampton	no	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
885	0	3	female	39.0	0	5	29.1250	Q	woman	False	NaN	Queenstown	no	False
887	1	1	female	19.0	0	0	30.0000	S	woman	False	B	Southampton	yes	True
888	0	3	female	NaN	1	2	23.4500	S	woman	False	NaN	Southampton	no	False
889	1	1	male	26.0	0	0	30.0000	C	man	True	C	Cherbourg	yes	True
890	0	3	male	32.0	0	0	7.7500	Q	man	True	NaN	Queenstown	no	True

784 rows × 14 columns

데이터 정제

- Null 데이터
  - 누락되거나 null 값
    - 데이터셋의 품질을 가리킴

  - isnull() 함수를 사용해 누락된 값이 있는 위치를 확인
    - Titanic_data.isnull().head(10)

```
1  titanic_data.isnull()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	who	adult_male	deck	embark_town	alive	alone
0	False	False	False	False	False	False	False	False	False	False	True	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	True	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	True	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
885	False	False	False	False	False	False	False	False	False	False	True	False	False	False
887	False	False	False	False	False	False	False	False	False	False	False	False	False	False
888	False	False	False	True	False	False	False	False	False	False	True	False	False	False
889	False	False	False	False	False	False	False	False	False	False	False	False	False	False
890	False	False	False	False	False	False	False	False	False	False	True	False	False	False

784 rows × 14 columns

데이터 정제

- 누락된 값 다루기
  - 누락된 값의 요약을 확인
    - titanic_data.isnull().sum()
- Null 값 제거
  - dropna() 함수를 사용하여 null 값이 있는 행 제거
    - titanic_data.dropna(inplace=True)
    - titanic_data.isnull().sum()
- Null 값 제거
  - dropna()를 사용해 열을 제거 할 수도 있음
    - titanic_data.dropna(axis=1)
      - 0, or 'index' : 누락된 값이 포함된 행 제거
      - 1, or 'columns' : 누락된 값이 포함된 열 제거
  - 힌트
    - 누락된 데이터가 적은 경우에만 null 데이터를 제거하는 것이 좋음

데이터 정제

- 데이터 대체
  - Null을 채우기 위한 fillna() 함수
    - titanic_data['age'].fillna(titanic_data['age'].mean(), inplace=True)
    - titanic_data.isnull().sum()

문제풀이

- Pandas 데이터프레임에서 행을 제거하는 방법을 설명하시오.

- Pandas 데이터프레임에서 NULL 데이터를 찾는 방법을 설명하시오.

요약

- Pandas 데이터프레임에서 다음과 같은 명령을 수행하는 방법을 공부했음
  - 행과 열을 삭제하는 방법
  - 값을 대체하는 방법
  - NULL 데이터를 처리하는 방법

01

데이터처리 과정

- 비즈니스 데이터 분석 사이클
- 테이터 분석 사이클
- 타이타닉 데이터 이해
- 데이터 전처리

02

데이터 정제

- 행과 열 제거
- 중복데이터 제거
- NULL 데이터 처리

03

데이터 가공 과정

- 데이터 변화
- 데이터 정규화
- 이상한 데이터 제거

학습목표

- **이번 파트에서는 데이터 가공 과정을 위해 다음과 같은 내용을 다룬다.**
  - 데이터 변화
  - 데이터 정규화
  - 이상한 데이터 제거

데이터 변환

- replace()
  - 행 값을 대치하는 방법

```
1  ddd=pd.Series([4,5,6,7,8,9])
```

```
1  ddd.replace(4,10)
```

```
0    10
1     5
2     6
3     7
4     8
5     9
dtype: int64
```

데이터 변환

```
dpd = pd.DataFrame({'X':[0,1,2,3,4,5],
                    'Y': [6,7,8,9,10,11],
                    'Z': ['a','b','c','d','e','f']})
```

```
dpd
```

	X	Y	Z
0	0	6	a
1	1	7	b
2	2	8	c
3	3	9	d
4	4	10	e
5	5	11	f

```
dpd.replace(0, 11)
```

	X	Y	Z
0	11	6	a
1	1	7	b
2	2	8	c
3	3	9	d
4	4	10	e
5	5	11	f

```
dd1=dpd.replace(0, 11)
```

```
dd1.replace(11, 40)
```

	X	Y	Z
0	40	6	a
1	1	7	b
2	2	8	c
3	3	9	d
4	4	10	e
5	5	40	f

데이터 대치

- replace()
  - titanic_data.replace({'sex': {'male': 0}}, inplace = True)

```
1 titanic_data.replace({'sex': {'male': 0}}, inplace = True)
```

```
1 titanic_data
```

	SURVIVED	pclass	sex	age	sibsp	parch	fare	embarked	who	adult_male	deck	embark_town	alive	alone
0	0	3	0	22.0	1	0	7.2500	S	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	woman	False	C	Southampton	yes	False
4	0	3	0	35.0	0	0	8.0500	S	man	True	NaN	Southampton	no	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
885	0	3	female	39.0	0	5	29.1250	Q	woman	False	NaN	Queenstown	no	False
887	1	1	female	19.0	0	0	30.0000	S	woman	False	B	Southampton	yes	True
888	0	3	female	NaN	1	2	23.4500	S	woman	False	NaN	Southampton	no	False
889	1	1	0	26.0	0	0	30.0000	C	man	True	C	Cherbourg	yes	True
890	0	3	0	32.0	0	0	7.7500	Q	man	True	NaN	Queenstown	no	True

784 rows × 14 columns

titanic_data.replace({'sex': {'female': 1}}, inplace = True)

데이터변환


```
1 from sklearn.preprocessing import LabelEncoder
```

```
1 le = LabelEncoder()
2 titanic_data['who_labeled'] = le.fit_transform(titanic_data.who)
```

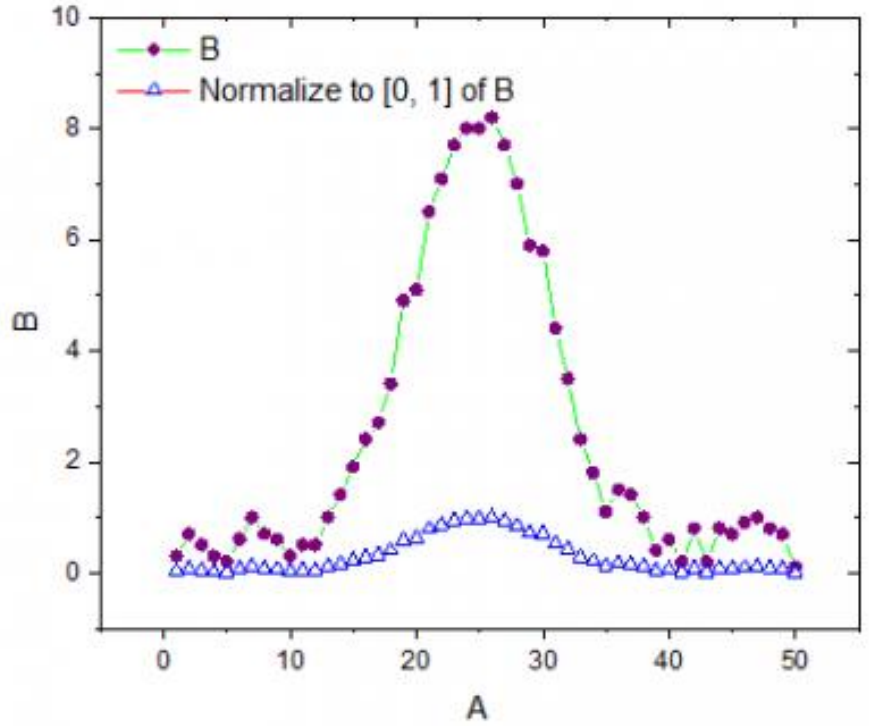
```
1 titanic_data
```

	SURVIVED	pclass	sex	age	sibsp	parch	fare	embarked	who	adult_male	deck	embark_town	alive	alone	who_labeled
0	0	3	0	22.0	1	0	0.014151	S	man	True	NaN	Southampton	no	False	1
1	1	1	1	38.0	1	0	0.139136	C	woman	False	C	Cherbourg	yes	False	2
2	1	3	1	26.0	0	0	0.015469	S	woman	False	NaN	Southampton	yes	True	2
3	1	1	1	35.0	1	0	0.103644	S	woman	False	C	Southampton	yes	False	2
4	0	3	0	35.0	0	0	0.015713	S	man	True	NaN	Southampton	no	True	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
885	0	3	1	39.0	0	5	0.056848	Q	woman	False	NaN	Queenstown	no	False	2
887	1	1	1	19.0	0	0	0.058556	S	woman	False	B	Southampton	yes	True	2
888	0	3	1	NaN	1	2	0.045771	S	woman	False	NaN	Southampton	no	False	2
889	1	1	0	26.0	0	0	0.058556	C	man	True	C	Cherbourg	yes	True	1
890	0	3	0	32.0	0	0	0.015127	Q	man	True	NaN	Queenstown	no	True	1

784 rows × 15 columns

데이터 정규화

- 정규화는 왜 필요한가요?
  - A와 B 데이터간에 차이가 너무 많이 남



데이터 정규화

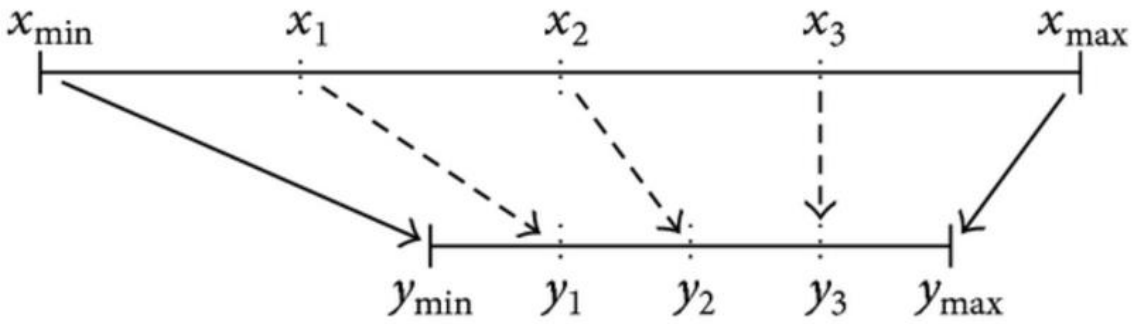
- 정규화는 왜 필요한가요?
  - titanic_data.describe()

```
1 titanic_data.describe()
```

	SURVIVED	pclass	sex	age	sibsp	parch	fare
count	784.000000	784.000000	784.000000	678.000000	784.000000	784.000000	784.000000
mean	0.411990	2.243622	0.373724	29.869351	0.522959	0.415816	34.711740
std	0.492507	0.855056	0.484101	14.759076	0.986231	0.836922	52.160151
min	0.000000	1.000000	0.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	1.000000	0.000000	20.000000	0.000000	0.000000	8.050000
50%	0.000000	3.000000	0.000000	28.250000	0.000000	0.000000	15.900000
75%	1.000000	3.000000	1.000000	39.000000	1.000000	1.000000	34.109350
max	1.000000	3.000000	1.000000	80.000000	8.000000	6.000000	512.329200

데이터 정규화

- Min-max 정규화
  - 0과 1 사이의 데이터 정규화



  - 다음 공식에 따라 계산

$$x_{norm} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

데이터 정규화

- Min-max 정규화
  - sklearn의 MinMaxScaler() 함수 사용

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

titanic_data[['fare']] =
scaler.fit_transform(titanic_data[['fare']])

titanic_data.describe()
```

데이터 정규화


```
1 scaler = MinMaxScaler()
2 titanic_data[['fare']] = scaler.fit_transform(titanic_data[['fare']])
3 titanic_data.describe()
```

	SURVIVED	pclass	sex	age	sibsp	parch	fare
count	784.000000	784.000000	784.000000	678.000000	784.000000	784.000000	784.000000
mean	0.411990	2.243622	0.373724	29.869351	0.522959	0.415816	0.067753
std	0.492507	0.855056	0.484101	14.759076	0.986231	0.836922	0.101810
min	0.000000	1.000000	0.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	1.000000	0.000000	20.000000	0.000000	0.000000	0.015713
50%	0.000000	3.000000	0.000000	28.250000	0.000000	0.000000	0.031035
75%	1.000000	3.000000	1.000000	39.000000	1.000000	1.000000	0.066577
max	1.000000	3.000000	1.000000	80.000000	8.000000	6.000000	1.000000

데이터 정규화

- **표준 정규화**
  - 평균 0 / 분산 1
  - **from** sklearn.preprocessing **import** StandardScaler
  - scaler = StandardScaler()
  - scaler.fit_transform(실제데이터)
- **표준 (Z-score) 정규화**
  - (X - 평균) / 표준편차

데이터 정규화

- **로버스트 정규화**
  - 중앙값 = 0 / (1분위-3분위)=1
  - **from** sklearn.preprocessing **import** RobustScaler
  - Scaler = RobustScaler()
  - scaler.fit_transform(실제데이터)

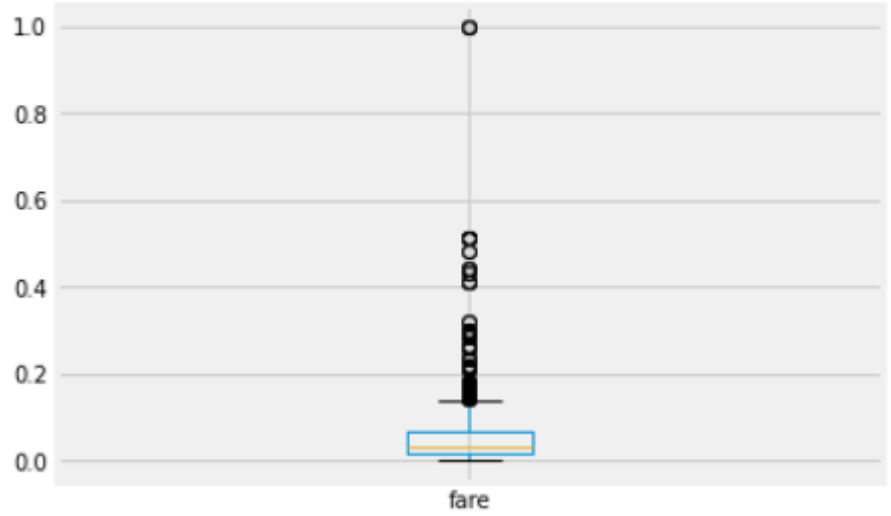
- **MAXAbs 정규화**
  - 0을 기준으로 절대값이 가장 큰수가 1 또는 -1로 변환
  - **from** sklearn.preprocessing **import** MaxAbsScaler
  - Scaler = MaxAbsScaler()
  - scaler.fit_transform(실제데이터)

이상치 데이터 처리

- 이상 값 처리
  - titanic_data['fare'].plot(kind="box")

```
1  titanic_data['fare'].plot(kind="box")
```

```
<AxesSubplot:>
```

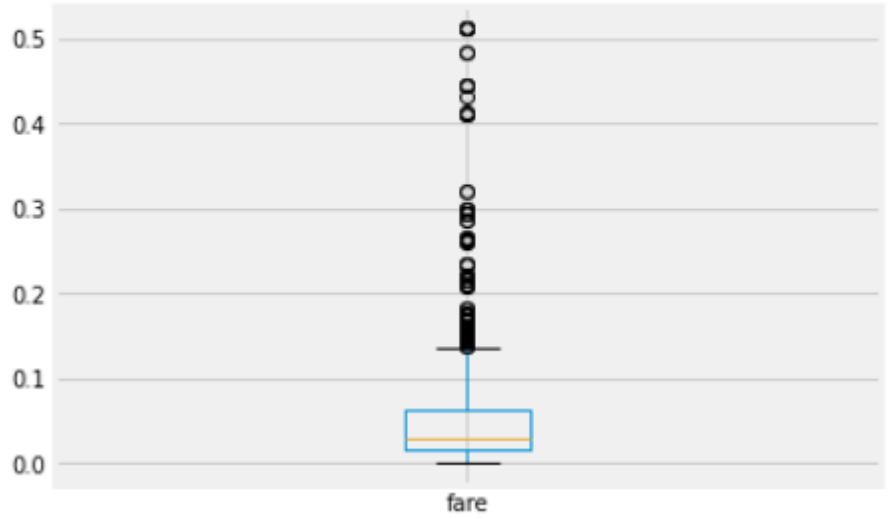


이상치 데이터 처리

- 이상 값 처리
  - titanic_data.drop(titanic_data[titanic_data.fare >= 1].index, inplace = True)
  - titanic_data['fare'].plot(kind="box")

```
1  titanic_data.drop(titanic_data[titanic_data.fare >= 1].index, inplace = True)
2  titanic_data['fare'].plot(kind="box")
```

```
<AxesSubplot:>
```



문제풀이

- 데이터 분석을 위해 왜 데이터 변환이 필요한지 설명하시오.
- 데이터 분석을 위해 왜 데이터 정규화가 필요한지 설명하시오.
- 특히, 데이터 분석시 이상한 데이터를 왜 제거하는게 유리한지 설명하시오.

요약

- 다음과 같은 데이터 전처리 기능에 대해 설명하였음.
  - 데이터 변환
  - 데이터 정규화
  - 이상치 데이터 제거