

방학 2주차

# DATA PREPROCESSING

수나로움

# Data Preprocessing

## (1) Data Preprocessing (전처리)

데이터 분석을 진행하기 전에 미리 진행하는 과정

전처리를 하지 않으면

- (1) 오류가 나거나
- (2) 원하지 않은 결과가 나오거나
- (3) 시간이 엄청나게 늘어남

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

## (0) Data Exploration

사용할 데이터를 찾는 과정

**실습에서 사용할 데이터 다운받기**

```
curl -L -O https://raw.githubusercontent.com/sunaroum/studynote/main/dataset/busan_estates.csv
```

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

## (1) Data Cleaning

문제 있는 데이터를 제대로 바꾸는 과정

- (A) 쓸모 없는 데이터
- (B) 데이터 누락
- (C) 데이터 중복
- (D) 데이터 오류
- (E) 튀는 데이터

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

## (1) Data Cleaning

### (A) 쓸모 없는 데이터

```
1 import pandas as pd
2
3 df = pd.read_csv("/content/busan_estates.csv")
```

Unnamed: 0	기준 연도	기준 월	법정동	행정동	행정동 코드	집계구 코드	평균 보증금 (만원)	평균 월세 (만원)	평균 면적 (㎡)	전월세	건물 종류
0	0	2021	7 부산광역시 중구 중앙동1가	부산광역시 중구 중앙동	2611051000	2101051010002	833.33	52.33	27.23	월세	오피스텔
1	1	2021	7 부산광역시 중구 대창동1가	부산광역시 중구 중앙동	2611051000	2101051010201	1500.00	62.50	54.75	월세	아파트
2	2	2021	7 부산광역시 중구 중앙동4가	부산광역시 중구 중앙동	2611051000	2101051020001	3000.00	91.67	54.89	월세	아파트
3	3	2021	7 부산광역시 중구 중앙동4가	부산광역시 중구 중앙동	2611051000	2101051020001	28650.00	0.00	55.01	전세	아파트
4	4	2021	7 부산광역시 중구 중앙동4가	부산광역시 중구 중앙동	2611051000	2101051020001	2008.33	43.42	25.37	월세	오피스텔
...	...	...	...	...	...	...	...	...	...	...	...
72451	72451	2023	6 부산광역시 해운대구 중동	부산광역시 해운대구 중2동	2635054000	2109054010201	13500.00	0.00	59.96	전세	아파트
72452	72452	2023	6 부산광역시 해운대구 중동	부산광역시 해운대구 중2동	2635054000	2109054011801	3000.00	50.00	59.90	월세	아파트
72453	72453	2023	6 부산광역시 해운대구 중동	부산광역시 해운대구 중2동	2635054000	2109054012202	2300.00	150.60	89.38	월세	아파트
72454	72454	2023	6 부산광역시 해운대구 중동	부산광역시 해운대구 중2동	2635054000	2109054012202	34000.00	0.00	98.71	전세	아파트
72455	72455	2023	6 부산광역시 해운대구 중동	부산광역시 해운대구 중2동	2635054000	2109054012301	25000.00	0.00	78.78	전세	아파트

72456 rows × 12 columns

# Data Preprocessing

## ★ 실습과 함께 하는 데이터 전처리 ★

### (1) Data Cleaning

#### (A) 쓸모 없는 데이터

**df.drop()**

`DataFrame.drop(labels=None, *, axis=0, index=None, columns=None, level=None, inplace=False, errors='raise')`

**labels** : 삭제할 위치    **axis** : 삭제할 차원 (0은 행, 1은 열, ....)

ex)

	col1	col2	col3
row1	1	2	3
row2	4	5	6
row3	7	8	9

→ `df.drop(labels="row1", axis=0)` →

	col1	col2	col3
row2	4	5	6
row3	7	8	9

# Data Preprocessing

## ★ 실습과 함께 하는 데이터 전처리 ★

### (1) Data Cleaning

#### (A) 쓸모 없는 데이터

**df.drop()**

`DataFrame.drop(labels=None, *, axis=0, index=None, columns=None, level=None, inplace=False, errors='raise')`

index : 삭제할 행의 이름 ('labels' + 'axis=0')

ex)

	col1	col2	col3
row1	1	2	3
row2	4	5	6
row3	7	8	9

→ `df.drop(index="row2")` →

	col1	col2	col3
row1	1	2	3
row3	7	8	9

# Data Preprocessing

## ★ 실습과 함께 하는 데이터 전처리 ★

### (1) Data Cleaning

#### (A) 쓸모 없는 데이터

**df.drop()**

`DataFrame.drop(labels=None, *, axis=0, index=None, columns=None, level=None, inplace=False, errors='raise')`

**columns** : 삭제할 열의 이름 ('labels' + 'axis=1')

ex)

	col1	col2	col3
row1	1	2	3
row2	4	5	6
row3	7	8	9

→ `df.drop(index="row2")` →

	col1	col3
row1	1	3
row2	4	6
row3	7	9



# Data Preprocessing

## ★ 실습과 함께 하는 데이터 전처리 ★

### (1) Data Cleaning

#### (A) 쓸모 없는 데이터

**df.drop()**

`DataFrame.drop(labels=None, *, axis=0, index=None, columns=None, level=None, inplace=False, errors='raise')`

level : index나 column이 여러줄일 때 사용

ex)

value		
ind1	ind2	
a	1	1
	2	2
b	1	3
	2	4
c	1	5
	2	6

# Data Preprocessing

## ★ 실습과 함께 하는 데이터 전처리 ★

### (1) Data Cleaning

#### (A) 쓸모 없는 데이터

**df.drop()**

`DataFrame.drop(labels=None, *, axis=0, index=None, columns=None, level=None, inplace=False, errors='raise')`

**inplace** : 원본도 변경할지 말지에 대한 여부

- True : 원본을 변경하고 return은 None으로 함(= return값이 없음)
- False : 원본은 그대로 두고 return값으로 drop된 dataframe을 제공함

# Data Preprocessing

## ★ 실습과 함께 하는 데이터 전처리 ★

### (1) Data Cleaning

#### (A) 쓸모 없는 데이터

**df.drop()**

`DataFrame.drop(labels=None, *, axis=0, index=None, columns=None, level=None, inplace=False, errors='raise')`

**error : 삭제할 데이터가 없을 때 오류를 피울지에 대한 여부**

- **raise : 오류를 띄움**
- **ignore : 오류를 띄우지 않음**

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(1) Data Cleaning

(A) 쓸모 없는 데이터

```
1 df.drop('Unnamed: 0', axis=1, inplace=True)
2
3 df
```

	기준 연도	기준 월	법정동	행정동	행정동 코드	집계구 코드	평균 보증금 (만원)	평균 월세 (만원)	평균 면적 (㎡)	전월세	건물 종류
0	2021	7	부산광역시 중구 중앙동1가	부산광역시 중구 중앙동	2611051000	2101051010002	833.33	52.33	27.23	월세	오피스텔
1	2021	7	부산광역시 중구 대창동1가	부산광역시 중구 중앙동	2611051000	2101051010201	1500.00	62.50	54.75	월세	아파트
2	2021	7	부산광역시 중구 중앙동4가	부산광역시 중구 중앙동	2611051000	2101051020001	3000.00	91.67	54.89	월세	아파트
3	2021	7	부산광역시 중구 중앙동4가	부산광역시 중구 중앙동	2611051000	2101051020001	28650.00	0.00	55.01	전세	아파트
4	2021	7	부산광역시 중구 중앙동4가	부산광역시 중구 중앙동	2611051000	2101051020001	2008.33	43.42	25.37	월세	오피스텔
...	...	...	...	...	...	...	...	...	...	...	...
72451	2023	6	부산광역시 해운대구 중동	부산광역시 해운대구 중2동	2635054000	2109054010201	13500.00	0.00	59.96	전세	아파트
72452	2023	6	부산광역시 해운대구 중동	부산광역시 해운대구 중2동	2635054000	2109054011801	3000.00	50.00	59.90	월세	아파트
72453	2023	6	부산광역시 해운대구 중동	부산광역시 해운대구 중2동	2635054000	2109054012202	2300.00	150.60	89.38	월세	아파트
72454	2023	6	부산광역시 해운대구 중동	부산광역시 해운대구 중2동	2635054000	2109054012202	34000.00	0.00	98.71	전세	아파트
72455	2023	6	부산광역시 해운대구 중동	부산광역시 해운대구 중2동	2635054000	2109054012301	25000.00	0.00	78.78	전세	아파트

72456 rows x 11 columns

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(1) Data Cleaning

(B) 데이터 누락

누락의 종류

## MCAR

Missing Completely At Random

완전 무작위 결측

완전히 독립적인 결측값인 경우

## 예시

완전히 실수로 기입을 깜빡한 경우

실수로 자료에 커피를 흘린 경우

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(1) Data Cleaning

(B) 데이터 누락

누락의 종류

## MNAR

Missing Not At Random

비 무작위 결측

결측된 열을 포함하여 의존적인 경우

## 예시

체중 감소제 임상실험에 참가했는데  
몸무게 재는 게 부끄러워서 재지 않음

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(1) Data Cleaning

(B) 데이터 누락

누락의 종류

## MAR

Missing At Random

무작위 결측

관측된 열에만 의존적인 경우

## 예시

거식증 약 임상 실험에 참가했는데  
앞선 결과들이 너무 위험해서 투약 중단이 된 경우

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(1) Data Cleaning

(B) 데이터 누락

해결하는 방법

(1) 행/열 제거

(2) 특정 값으로 채우기

+

(3) 머신러닝 기법들을 사용함



# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(1) Data Cleaning

(B) 데이터 누락

`df.isna().sum()`

`df.isna()` : 각 값들이 NaN값인지 확인함.

ex)

```
1 display(df.isna())
```

	col1	col2	col3
row1	False	False	False
row2	False	True	False
row3	False	False	False

# Data Preprocessing

## ★ 실습과 함께 하는 데이터 전처리 ★

### (1) Data Cleaning

#### (B) 데이터 누락

**df.isna().sum()**

**df.isna().sum** : 각 열 별로 NaN값이 얼마나 되는지 확인함.

**ex)**

```
1 display(df.isna().sum())  
col1    0  
col2    1  
col3    0  
dtype: int64
```

# Data Preprocessing

## ★ 실습과 함께 하는 데이터 전처리 ★

### (1) Data Cleaning

#### (B) 데이터 누락 - (1) 행/열 제거

**df.dropna()**

`df.dropna(*, axis=0, how='', thresh='', subset=None, inplace=False, ignore_index=False)`

**axis** : 결측치가 속한 행과 열 중 어느 걸 제거할 건지 여부

**ex)**

```
1 display(df.dropna(axis=0))
```

	col1	col2	col3
row1	1	2.0	3
row3	7	8.0	9

# Data Preprocessing

## ★ 실습과 함께 하는 데이터 전처리 ★

### (1) Data Cleaning

#### (B) 데이터 누락 - (1) 행/열 제거

**df.dropna()**

`df.dropna(*, axis=0, how='', thresh='', subset=None, inplace=False, ignore_index=False)`

**how** : 제거 기준을 어떻게 할 것인지에 대한 여부

- any : 하나라도 결측치가 있으면 제거 (default)
- all : 해당 행/열의 모든 값이 결측치면 제거

# Data Preprocessing

## ★ 실습과 함께 하는 데이터 전처리 ★

### (1) Data Cleaning

#### (B) 데이터 누락 - (1) 행/열 제거

**df.dropna()**

`df.dropna(*, axis=0, how='', thresh='', subset=None, inplace=False, ignore_index=False)`

**thresh** : '결측되지 않은 값'을 기준으로 dropna의 실행 기준을 정함.

**ex)**

```
1 display(df.dropna(axis=0, thresh=2))
```

	col1	col2	col3
row1	1	2.0	3
row2	4	NaN	6
row3	7	8.0	9

# Data Preprocessing

## ★ 실습과 함께 하는 데이터 전처리 ★

### (1) Data Cleaning

#### (B) 데이터 누락 - (1) 행/열 제거

**df.dropna()**

`df.dropna(*, axis=0, how='', thresh='', subset=None, inplace=False, ignore_index=False)`

**subset : 특정 행/열에 대해서만 진행하고 싶은 경우에 사용함**

**ex)**

```
1 display(df.dropna(axis=1, subset=['row1', 'row2']))
```

	col1	col3
row1	1.0	3.0
row2	4.0	6.0
row3	7.0	9.0
row4	NaN	NaN

- axis가 행이면 subset은 열, axis가 열이면 subset은 행으로 작성

# Data Preprocessing

## ★ 실습과 함께 하는 데이터 전처리 ★

### (1) Data Cleaning

#### (B) 데이터 누락 - (1) 행/열 제거

**df.dropna()**

`df.dropna(*, axis=0, how='', thresh='', subset=None, inplace=False, ignore_index=False)`

**inplace** : 원본도 변경할지 말지에 대한 여부

- True : 원본을 변경하고 return은 None으로 함(= return값이 없음)
- False : 원본은 그대로 두고 return값으로 drop된 dataframe을 제공함

# Data Preprocessing

## ★ 실습과 함께 하는 데이터 전처리 ★

### (1) Data Cleaning

#### (B) 데이터 누락 - (1) 행/열 제거

**df.dropna()**

`df.dropna(*, axis=0, how='', thresh='', subset=None, inplace=False, ignore_index=False)`

**ignore\_index : drop된 결과를 기준으로 새로 index를 지정할지에 대한 여부**

- True : 새로 index를 지정함
- False : index를 그대로 넘둠



# Data Preprocessing

## ★ 실습과 함께 하는 데이터 전처리 ★

### (1) Data Cleaning

#### (B) 데이터 누락 - (2) 특정 값으로 채우기

**df.mean()**

`df.mean(axis=0, skipna=True, numeric_only=False, **kwargs)`

**axis : 행과 열 중 어느 것을 합해서 평균을 구할지에 대한 여부**

**ex)**

```
1 display(df.mean(axis=0))  
col1    4.0  
col2    5.0  
col3    6.0  
dtype: float64
```

- **axis가 0이면 행들을 더해서 열의 평균을 낸**
- **axis가 1이면 열들을 더해서 행의 평균을 낸**

# Data Preprocessing

## ★ 실습과 함께 하는 데이터 전처리 ★

### (1) Data Cleaning

#### (B) 데이터 누락 - (2) 특정 값으로 채우기

**df.mean()**

`df.mean(axis=0, skipna=True, numeric_only=False, **kwargs)`

**skipna** : 평균을 구할 때 NaN 값을 무시하고 계산할 건지에 대한 여부

**ex)**

```
1 display(df.mean(axis=0, skipna=False))  
col1    4.0  
col2    NaN  
col3    6.0  
dtype: float64
```

• False를 할 경우 NaN값이 있는 열은 NaN이 됨

# Data Preprocessing

## ★ 실습과 함께 하는 데이터 전처리 ★

### (1) Data Cleaning

#### (B) 데이터 누락 - (2) 특정 값으로 채우기

**df.mean()**

`df.mean(axis=0, skipna=True, numeric_only=False, **kwargs)`

**numeric\_only** : 평균을 구할 때 '숫자값만 존재하는 열'들에 대해서만 계산할지의 여부

**ex)**

```
1 display(df.mean(axis=0, numeric_only=True))
```

```
col1    4.0  
col3    6.0  
dtype: float64
```

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(1) Data Cleaning

(B) 데이터 누락 - (2) 특정 값으로 채우기

**df.median()**

`df.median(axis=0, skipna=True, numeric_only=False, **kwargs)`

**axis** : 행과 열 중 어느 것으로 중앙값을 구할지에 대한 여부

**ex)**

```
1 display(df.median(axis=0))  
col1    4.0  
col2    5.0  
col3    6.0  
dtype: float64
```

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(1) Data Cleaning

(B) 데이터 누락 - (2) 특정 값으로 채우기

`df.median()`

`df.median(axis=0, skipna=True, numeric_only=False, **kwargs)`

`skipna` : 중앙값을 구할 때 NaN 값을 무시하고 계산할 건지에 대한 여부

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(1) Data Cleaning

(B) 데이터 누락 - (2) 특정 값으로 채우기

`df.median()`

`df.median(axis=0, skipna=True, numeric_only=False, **kwargs)`

`numeric_only` : 중앙값을 구할 때 '숫자값만 존재하는 열'들에 대해서만 계산할지의 여부

# Data Preprocessing

## ★ 실습과 함께 하는 데이터 전처리 ★

### (1) Data Cleaning

#### (B) 데이터 누락 - (2) 특정 값으로 채우기

**df.mode()**

`df.mode(axis=0, numeric_only=False, dropna=True)`

**axis : 행과 열 중 어느 것으로 최빈값을 구할지에 대한 여부**

**ex)**

	col1	col2	col3
row1	1	2	3
row2	1	2	6
row3	7	2	3

=>

```
1 display(df.mode(axis=0))
```

	col1	col2	col3
0	1	2	3

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(1) Data Cleaning

(B) 데이터 누락 - (2) 특정 값으로 채우기

`df.mode()`

`df.mode(axis=0, numeric_only=False, dropna=True)`

`numeric_only` : 최빈값을 구할 때 '숫자값만 존재하는 열'들에 대해서만 계산할지의 여부



# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(1) Data Cleaning

(B) 데이터 누락 - (2) 특정 값으로 채우기

**df.mode()**

```
df.mode(axis=0, numeric_only=False, dropna=True)
```

**dropna** : 결측치를 계산에서 제외할지 말지에 대한 여부

# Data Preprocessing

## ★ 실습과 함께 하는 데이터 전처리 ★

### (1) Data Cleaning

#### (B) 데이터 누락 - (2) 특정 값으로 채우기

**df.fillna()**

`df.fillna(value=None, *, method=None, axis=None, inplace=False, limit=None, downcast=None)`

value : NaN값에 들어갈 값

ex)

	col1	col2	col3
row1	1	2.0	3
row2	1	NaN	6
row3	7	2.0	3

=>

```
1 display(df.fillna(value=0))
```

	col1	col2	col3
row1	1	2.0	3
row2	1	0.0	6
row3	7	2.0	3

# Data Preprocessing

## ★ 실습과 함께 하는 데이터 전처리 ★

### (1) Data Cleaning

#### (B) 데이터 누락 - (2) 특정 값으로 채우기

**df.fillna()**

`df.fillna(value=None, *, method=None, axis=None, inplace=False, limit=None, downcast=None)`

**method : 값을 채워넣는 방식**

- backfill : 바로 아래 값이랑 똑같이 채워넣음
- bfill : =backfill
- ffill : 바로 위 값이랑 똑같이 채워넣음
- None : 내가 원하는 값으로 넣음
- value와 method는 상호배타적

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(1) Data Cleaning

(B) 데이터 누락 - (2) 특정 값으로 채우기

**df.fillna()**

`df.fillna(value=None, *, method=None, axis=None, inplace=False, limit=None, downcast=None)`

**axis** : fillna에서 method의 기준이 되는 것 (0 or 1)

# Data Preprocessing

## ★ 실습과 함께 하는 데이터 전처리 ★

### (1) Data Cleaning

#### (B) 데이터 누락 - (2) 특정 값으로 채우기

**df.fillna()**

`df.fillna(value=None, *, method=None, axis=None, inplace=False, limit=None, downcast=None)`

**inplace** : 원본도 변경할지 말지에 대한 여부

- True : 원본을 변경하고 return은 None으로 함(= return값이 없음)
- False : 원본은 그대로 두고 return값으로 fill된 dataframe을 제공함

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(1) Data Cleaning

(B) 데이터 누락 - (2) 특정 값으로 채우기

**df.fillna()**

`df.fillna(value=None, *, method=None, axis=None, inplace=False, limit=None, downcast=None)`

**limit : limit 값이 있는 경우 위에서부터 그 개수만큼만 채움**

# Data Preprocessing

## ★ 실습과 함께 하는 데이터 전처리 ★

### (1) Data Cleaning

#### (B) 데이터 누락 - (2) 특정 값으로 채우기

`df.fillna()`

`df.fillna(value=None, *, method=None, axis=None, inplace=False, limit=None, downcast=None)`

**downcast** : downcast를 적용할지 말지 여부를 정함.

- 정수값만 있음에도 '32.0' 형태로 표시할 때 `downcast = 'infer'`을 통해 '32' 형태로 바꿈

# Data Preprocessing

## ★ 실습과 함께 하는 데이터 전처리 ★

### (1) Data Cleaning

#### (B) 데이터 누락

#### (결측치와 마찬가지로 값을 NaN으로 만들기)

```
1 df.loc[(df['전월세'] == '월세') & (df['평균 월세 (만원)'] == 0), '평균 월세 (만원)'] = float('nan')
2
3 df
```

	기준 연도	기준 월	법정동	행정동	행정동 코드	집계구 코드	평균 보증금 (만원)	평균 월세 (만원)	평균 면적 (㎡)	전월세	건물 종류
0	2021	7	부산광역시 중구 중앙동1가	부산광역시 중구 중앙동	2611051000	2101051010002	833.33	52.33	27.23	월세	오피스텔
1	2021	7	부산광역시 중구 대창동1가	부산광역시 중구 중앙동	2611051000	2101051010201	1500.00	62.50	54.75	월세	아파트
2	2021	7	부산광역시 중구 중앙동4가	부산광역시 중구 중앙동	2611051000	2101051020001	3000.00	91.67	54.89	월세	아파트
3	2021	7	부산광역시 중구 중앙동4가	부산광역시 중구 중앙동	2611051000	2101051020001	28650.00	0.00	55.01	전세	아파트
4	2021	7	부산광역시 중구 중앙동4가	부산광역시 중구 중앙동	2611051000	2101051020001	2008.33	43.42	25.37	월세	오피스텔
...	...	...	...	...	...	...	...	...	...	...	...
72451	2023	6	부산광역시 해운대구 중동	부산광역시 해운대구 중2동	2635054000	2109054010201	13500.00	0.00	59.96	전세	아파트
72452	2023	6	부산광역시 해운대구 중동	부산광역시 해운대구 중2동	2635054000	2109054011801	3000.00	50.00	59.90	월세	아파트
72453	2023	6	부산광역시 해운대구 중동	부산광역시 해운대구 중2동	2635054000	2109054012202	2300.00	150.60	89.38	월세	아파트
72454	2023	6	부산광역시 해운대구 중동	부산광역시 해운대구 중2동	2635054000	2109054012202	34000.00	0.00	98.71	전세	아파트
72455	2023	6	부산광역시 해운대구 중동	부산광역시 해운대구 중2동	2635054000	2109054012301	25000.00	0.00	78.78	전세	아파트

72456 rows × 11 columns



# Data Preprocessing

## ★ 실습과 함께 하는 데이터 전처리 ★

### (1) Data Cleaning

#### (B) 데이터 누락

#### (결측치 확인)

```
기존 연도      0
기존 월        0
법정동        0
행정동        0
행정동 코드   0
집계구 코드   0
평균 보증금 (만원)  0
평균 월세 (만원)   1
평균 면적 (m2)    0
전월세        0
건물 종류      0
dtype: int64
```

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(1) Data Cleaning

(B) 데이터 누락

(결측치 제거)

1 df.dropna(axis=0, ignore\_index=True)

	기준 연도	기준 월	법정동	행정동	행정동 코드	집계구 코드	평균 보증금 (만원)	평균 월세 (만원)	평균 면적 (㎡)	전월세	건물 종류
0	2021	7	부산광역시 중구 중앙동1가	부산광역시 중구 중앙동	2611051000	2101051010002	833.33	52.33	27.23	월세	오피스텔
1	2021	7	부산광역시 중구 대창동1가	부산광역시 중구 중앙동	2611051000	2101051010201	1500.00	62.50	54.75	월세	아파트
2	2021	7	부산광역시 중구 중앙동4가	부산광역시 중구 중앙동	2611051000	2101051020001	3000.00	91.67	54.89	월세	아파트
3	2021	7	부산광역시 중구 중앙동4가	부산광역시 중구 중앙동	2611051000	2101051020001	28650.00	0.00	55.01	전세	아파트
4	2021	7	부산광역시 중구 중앙동4가	부산광역시 중구 중앙동	2611051000	2101051020001	2008.33	43.42	25.37	월세	오피스텔
...	...	...	...	...	...	...	...	...	...	...	...
72450	2023	6	부산광역시 해운대구 중동	부산광역시 해운대구 중2동	2635054000	2109054010201	13500.00	0.00	59.96	전세	아파트
72451	2023	6	부산광역시 해운대구 중동	부산광역시 해운대구 중2동	2635054000	2109054011801	3000.00	50.00	59.90	월세	아파트
72452	2023	6	부산광역시 해운대구 중동	부산광역시 해운대구 중2동	2635054000	2109054012202	2300.00	150.60	89.38	월세	아파트
72453	2023	6	부산광역시 해운대구 중동	부산광역시 해운대구 중2동	2635054000	2109054012202	34000.00	0.00	98.71	전세	아파트
72454	2023	6	부산광역시 해운대구 중동	부산광역시 해운대구 중2동	2635054000	2109054012301	25000.00	0.00	78.78	전세	아파트

72455 rows × 11 columns

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(1) Data Cleaning

(C) 데이터 중복

**중복값은 제거합니다**

# Data Preprocessing

## ★ 실습과 함께 하는 데이터 전처리 ★

### (1) Data Cleaning

#### (C) 데이터 중복

**df.duplicated()**

**df.duplicated(subset=None, keep='first')**

**subset : 중복인지 확인할 열들**

**ex)**

	Name	Age	City	Salary	Department	Gender
0	John	25	New York	50000	HR	Male
1	Jane	30	London	60000	Finance	Female
2	Tom	22	Paris	45000	IT	Male
3	Emily	28	Tokyo	55000	IT	Female
4	John	25	New York	50000	HR	Male
5	Alex	35	Sydney	70000	Marketing	Male

=>

```
1 display(df.duplicated(subset=['Name']))  
0    False  
1    False  
2    False  
3    False  
4     True  
5    False  
dtype: bool
```

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(1) Data Cleaning

(C) 데이터 중복

`df.duplicated()`

`df.duplicated(subset=None, keep='first')`

**keep : 중복이 있을 때 어느 쪽을 True로 할 것인지 여부**

- **first** : 첫번째 것만 빼고 나머지를 True로 함
- **last** : 마지막 것만 빼고 나머지를 True로 함
- **false** : 중복이면 전부다 True로 함

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

## (1) Data Cleaning

### (C) 데이터 중복

`df.drop_duplicates()`

`df.drop_duplicates(subset=None, *, keep='first', inplace=False, ignore_index=False)`

subset: 중복인지 확인할 열들

	Name	Age	City	Salary	Department	Gender
0	John	25	New York	50000	HR	Male
1	Jane	30	London	60000	Finance	Female
2	Tom	22	Paris	45000	IT	Male
3	Emily	28	Tokyo	55000	IT	Female
4	John	25	New York	50000	HR	Male
5	Alex	35	Sydney	70000	Marketing	Male

=>

```
1 display(df.drop_duplicates(subset=['Name']))
```

	Name	Age	City	Salary	Department	Gender
0	John	25	New York	50000	HR	Male
1	Jane	30	London	60000	Finance	Female
2	Tom	22	Paris	45000	IT	Male
3	Emily	28	Tokyo	55000	IT	Female
5	Alex	35	Sydney	70000	Marketing	Male

# Data Preprocessing

## ★ 실습과 함께 하는 데이터 전처리 ★

### (1) Data Cleaning

#### (C) 데이터 중복

`df.drop_duplicates()`

`df.drop_duplicates(subset=None, *, keep='first', inplace=False, ignore_index=False)`

**keep : 중복이 있을 때 어느 쪽을 지울 것인지 여부**

- **first** : 첫번째 것만 빼고 나머지를 지움
- **last** : 마지막 것만 빼고 나머지를 지움
- **false** : 중복이면 전부다 지움

# Data Preprocessing

## ★ 실습과 함께 하는 데이터 전처리 ★

### (1) Data Cleaning

#### (C) 데이터 중복

`df.drop_duplicates()`

`df.drop_duplicates(subset=None, *, keep='first', inplace=False, ignore_index=False)`

`ignore_index` : drop된 결과를 기준으로 새로 index를 지정할지에 대한 여부

- True : 새로 index를 지정함
- False : index를 그대로 뱉둠



# Data Preprocessing

## ★ 실습과 함께 하는 데이터 전처리 ★

### (1) Data Cleaning

#### (B) 데이터 누락 - (1) 행/열 제거

**df.dropna()**

`df.dropna(*, axis=0, how='', thresh='', subset=None, inplace=False, ignore_index=False)`

**ignore\_index : drop된 결과를 기준으로 새로 index를 지정할지에 대한 여부**

- True : 새로 index를 지정함
- False : index를 그대로 넘둠

# Data Preprocessing

## ★ 실습과 함께 하는 데이터 전처리 ★

### (1) Data Cleaning

#### (C) 데이터 중복

```
df.drop_duplicates()
```

```
df.drop_duplicates(subset=None, *, keep='first', inplace=False, ignore_index=False)
```

**inplace** : 원본도 변경할지 말지에 대한 여부

- True : 원본을 변경하고 return은 None으로 함(= return값이 없음)
- False : 원본은 그대로 두고 return값으로 drop된 dataframe을 제공함

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(1) Data Cleaning

(C) 데이터 중복

(중복 여부 확인)

```
1 print(df.duplicated().sum())
```

0

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(1) Data Cleaning

(D) 데이터 오류

(1) 단순 오타인 경우 : 확인 후 최대한 수정

(2) 오타가 아닌 경우 : 배경지식을 기반으로 수정

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(1) Data Cleaning

(E) 튜는 데이터

	col1	col2	col3
row1	1	2	3
row2	4	500000	6
row3	7	8	9

해결하는 방법

(1) 행/열 제거

(2) 값 변경하기

+

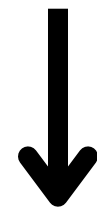
(3) 가중치를 도입하기

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(2) EDA

**EDA = Exploratory Data Analysis**  
**(탐색적 데이터 분석)**



**데이터를 분석하고 이해하는 과정!**

# Data Preprocessing

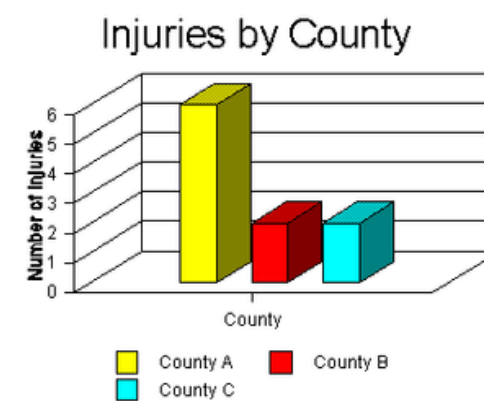
★ 실습과 함께 하는 데이터 전처리 ★

## (2) EDA

### (1) Univariate

- 일변량, 단변량
- 종속변수( $\approx Y$ )가 하나임
- 데이터를 분석하고 데이터의 패턴을 확인함

EDA의 대상



# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(2) EDA

EDA의 대상

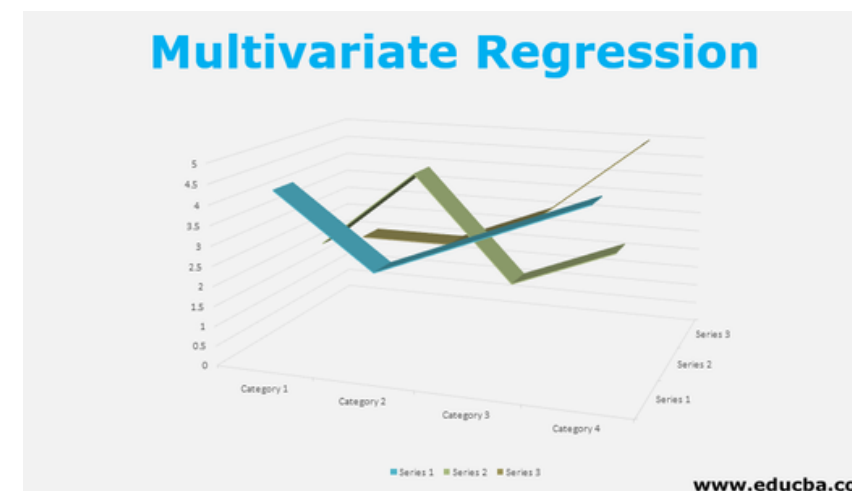
(2) Multivariate

- 다변량
- 종속변수( $\approx Y$ )가 2개 이상임
- 변수간의 관계를 파악함



(3) multiple

- 다중
- 독립변수( $\approx X$ )가 2개 이상임





# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

## (2) EDA

### (1) Graphic

- 통계적 그래프를 통해 데이터의 특성을 보여줌
- 데이터를 한눈에 파악하여 대략적인 형태 파악 가능

## EDA의 종류

### (2) Non-Graphic

- 숫자나 문자 등을 통해 데이터의 특성을 보여줌
- 정확한 값을 파악할 수 있음

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(2) EDA : non-graphic

**df.describe()**

	mpg	cylinders	displacement	weight	accerleration \
count	398.000000	398.000000	398.000000	398.000000	398.000000
mean	23.514573	5.454774	193.425879	2970.424623	15.568090
std	7.815984	1.701004	104.269838	846.841774	2.757689
min	9.000000	3.000000	68.000000	1613.000000	8.000000
25%	17.500000	4.000000	104.250000	2223.750000	13.825000
50%	23.000000	4.000000	148.500000	2803.500000	15.500000
75%	29.000000	8.000000	262.000000	3608.000000	17.175000
max	46.600000	8.000000	455.000000	5140.000000	24.800000

	model year	origin
count	398.000000	398.000000
mean	76.010050	1.572864
std	3.697627	0.802055
min	70.000000	1.000000
25%	73.000000	1.000000
50%	76.000000	1.000000
75%	79.000000	2.000000
max	82.000000	3.000000

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(2) EDA : non-graphic

**df.info()**

```
In [6]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 8 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Customer              100 non-null   int64
1   Type of Customer      100 non-null   object
2   Items                 100 non-null   int64
3   Net Sales             100 non-null   float64
4   Method of Payment     100 non-null   object
5   Gender                100 non-null   object
6   Marital Status        100 non-null   object
7   Age                   100 non-null   int64
dtypes: float64(1), int64(3), object(4)
memory usage: 6.4+ KB
```

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(2) EDA : non-graphic

**df.corr()**

```

count      mpg  cylinders  displacement      weight  accerleration \
mean      23.514573    5.454774    193.425879  2970.424623    15.568090
std        7.815984    1.701004    104.269838    846.841774     2.757689
min         9.000000    3.000000     68.000000  1613.000000     8.000000
25%        17.500000    4.000000    104.250000  2223.750000    13.825000
50%        23.000000    4.000000    148.500000  2803.500000    15.500000
75%        29.000000    8.000000    262.000000  3608.000000    17.175000
max        46.600000    8.000000    455.000000  5140.000000    24.800000

count  model year      origin
mean    76.010050    1.572864
std     3.697627     0.802055
min     70.000000    1.000000
25%     73.000000    1.000000
50%     76.000000    1.000000
75%     79.000000    2.000000
max     82.000000    3.000000

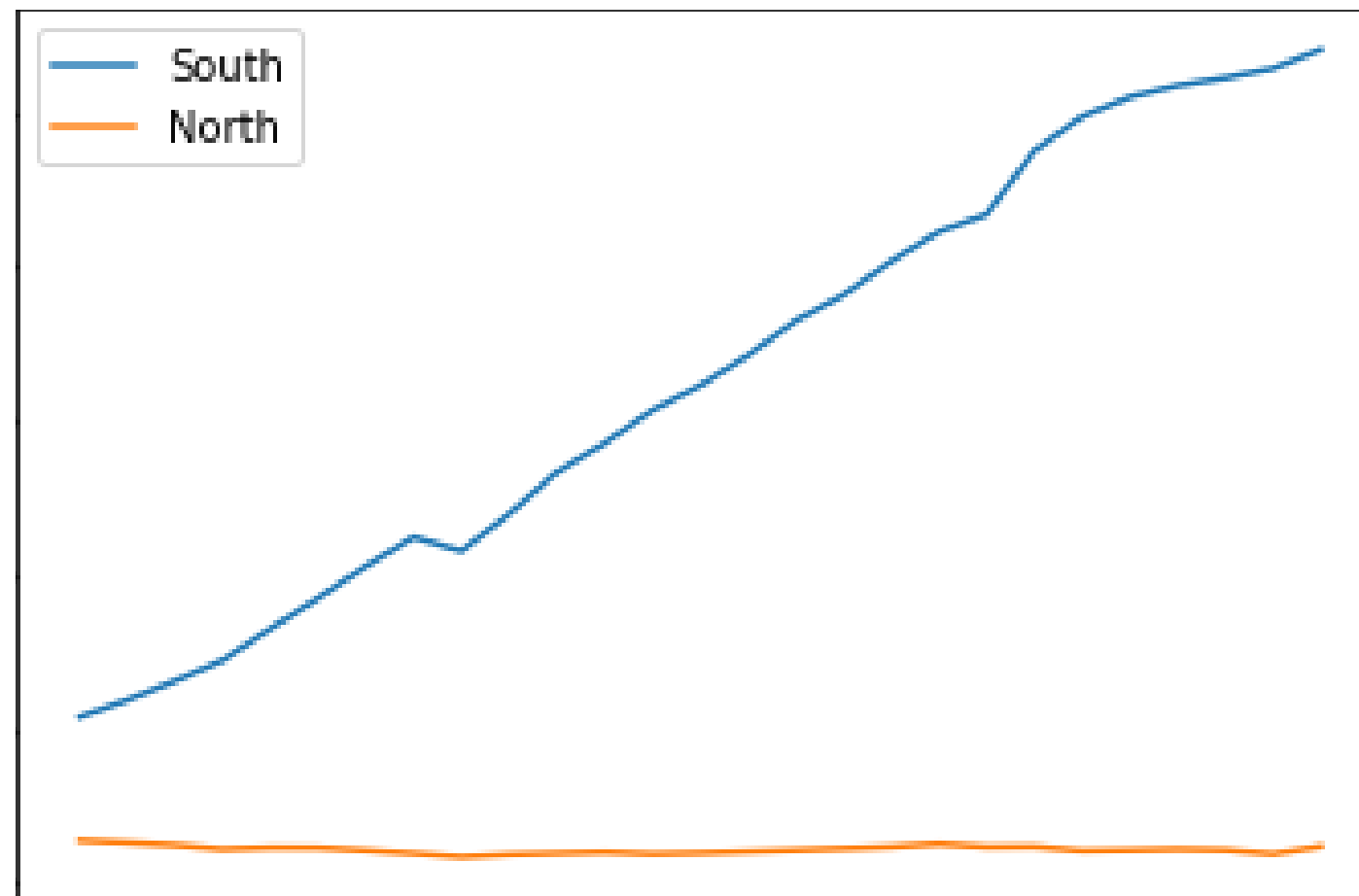
```

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(2) EDA : graphic

**df.plot()**



# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(2) EDA : graphic (detailed)

통계적 그래프를  
제공하는 라이브러리

(1) matplotlib (plt)

(2) seaborn (sns)

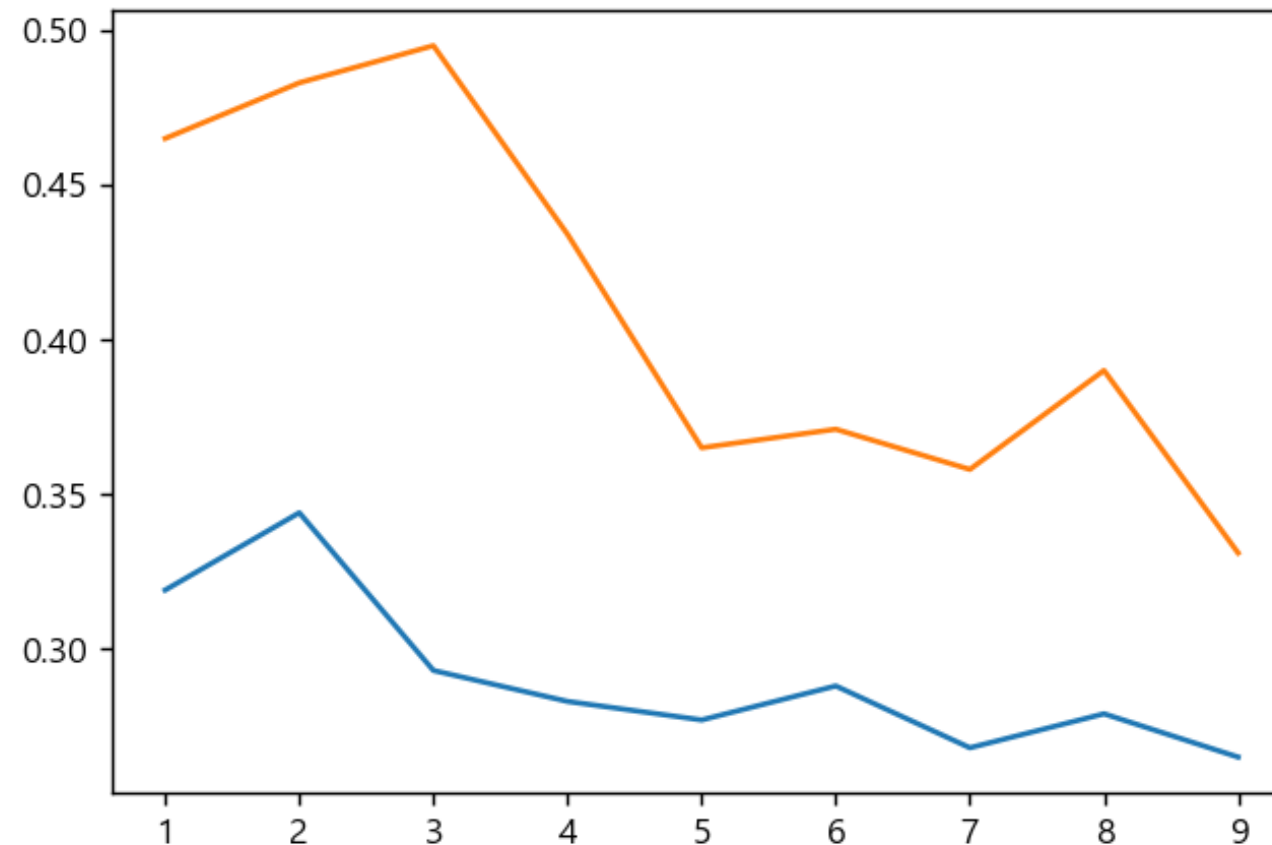
(3) df.plot

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(2) EDA : graphic (detailed)

## 선그래프 (Line Plot)



`df.plot(값)`

`plt.plot(x값, y값)`

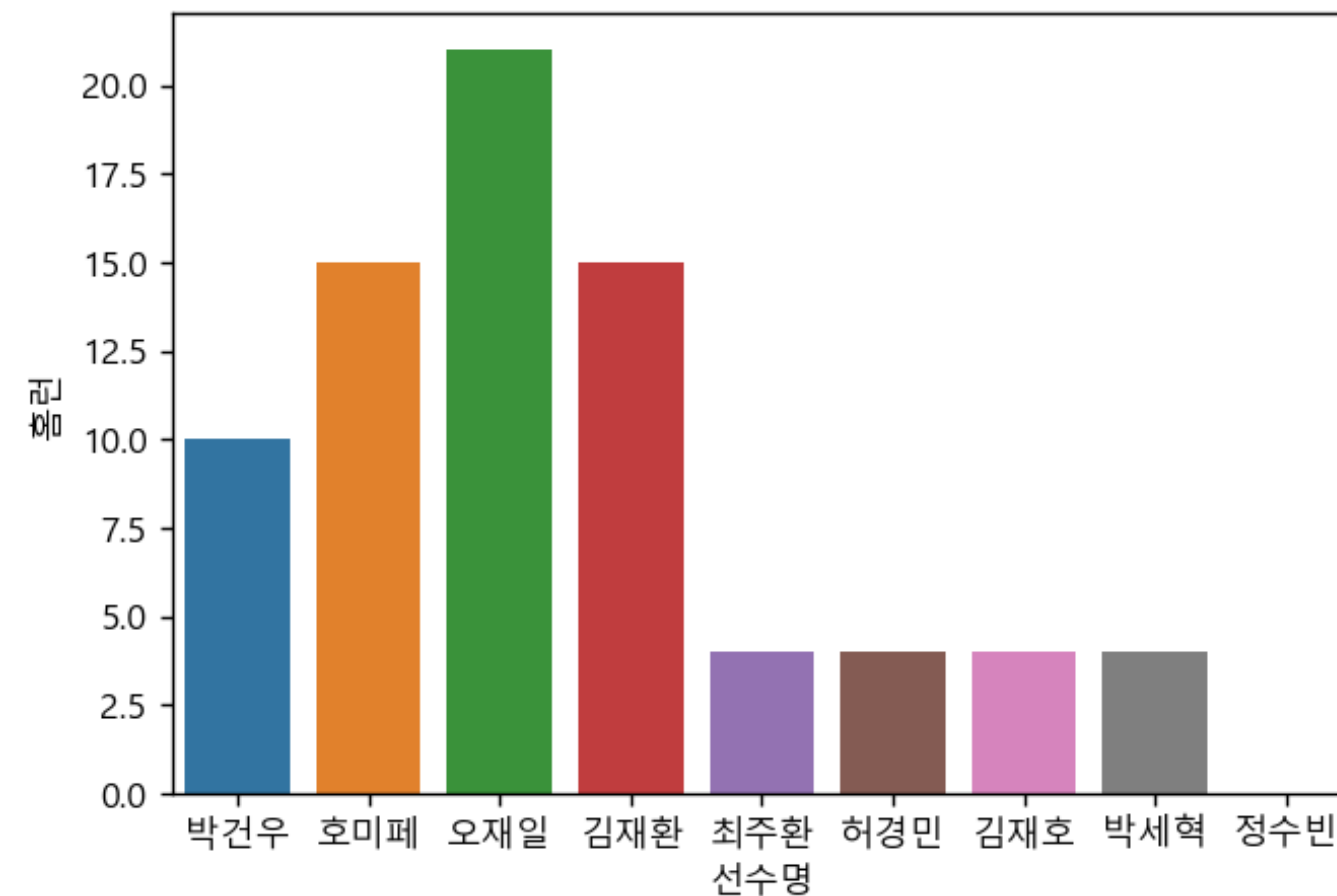
`sns.lineplot(x=x값, y=y값)`

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(2) EDA : graphic (detailed)

## 막대 그래프 (Bar Plot)



```
df.plot(kind='bar')
```

```
plt.bar(x값, y값)
```

```
sns.barplot(x=x값, y=y값)
```

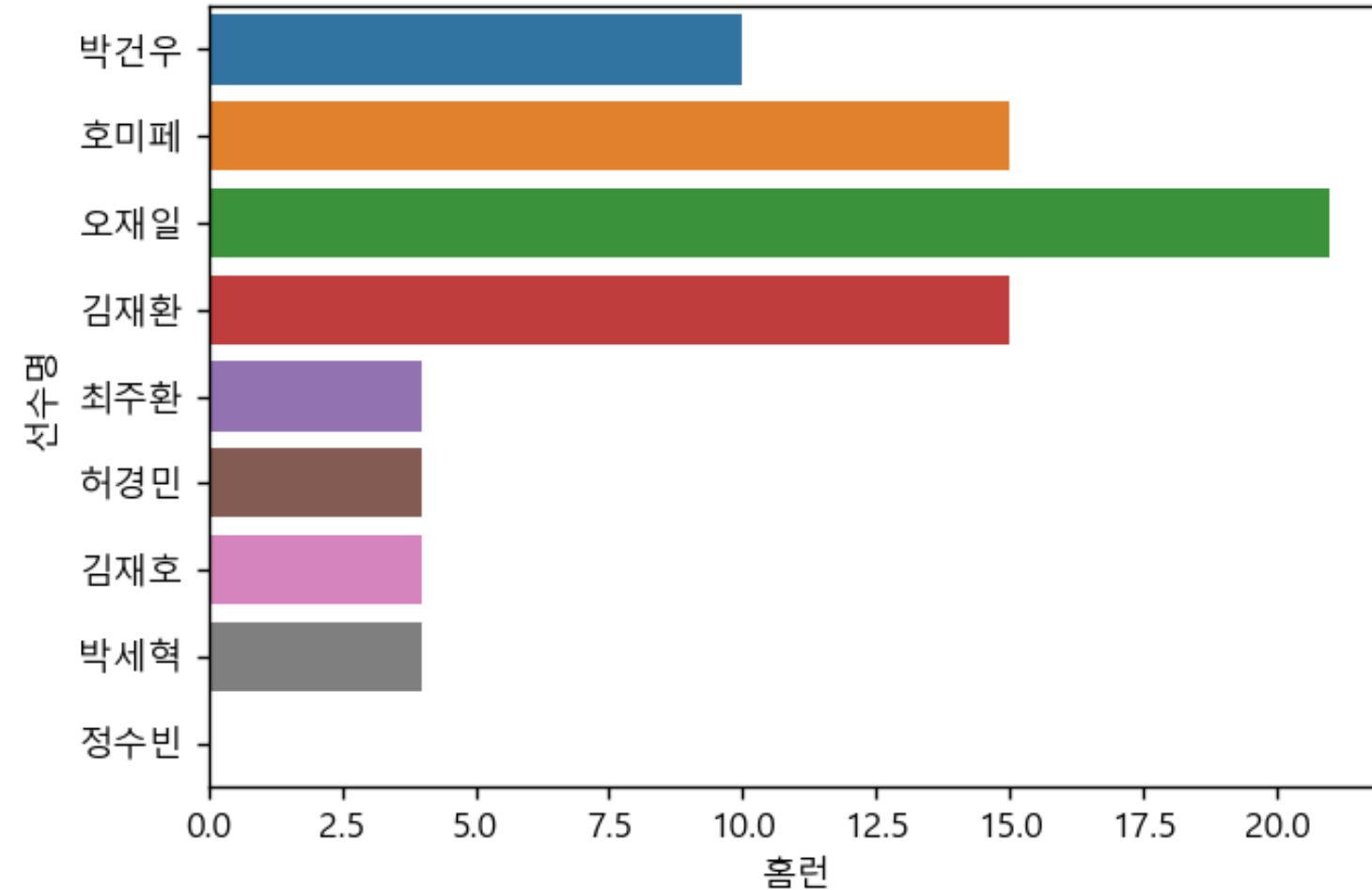


# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(2) EDA : graphic (detailed)

## 가로막대 그래프 (Barh Plot)



```
df.plot(값, kind=barh)
```

```
plt.bar(x값, y값)
```

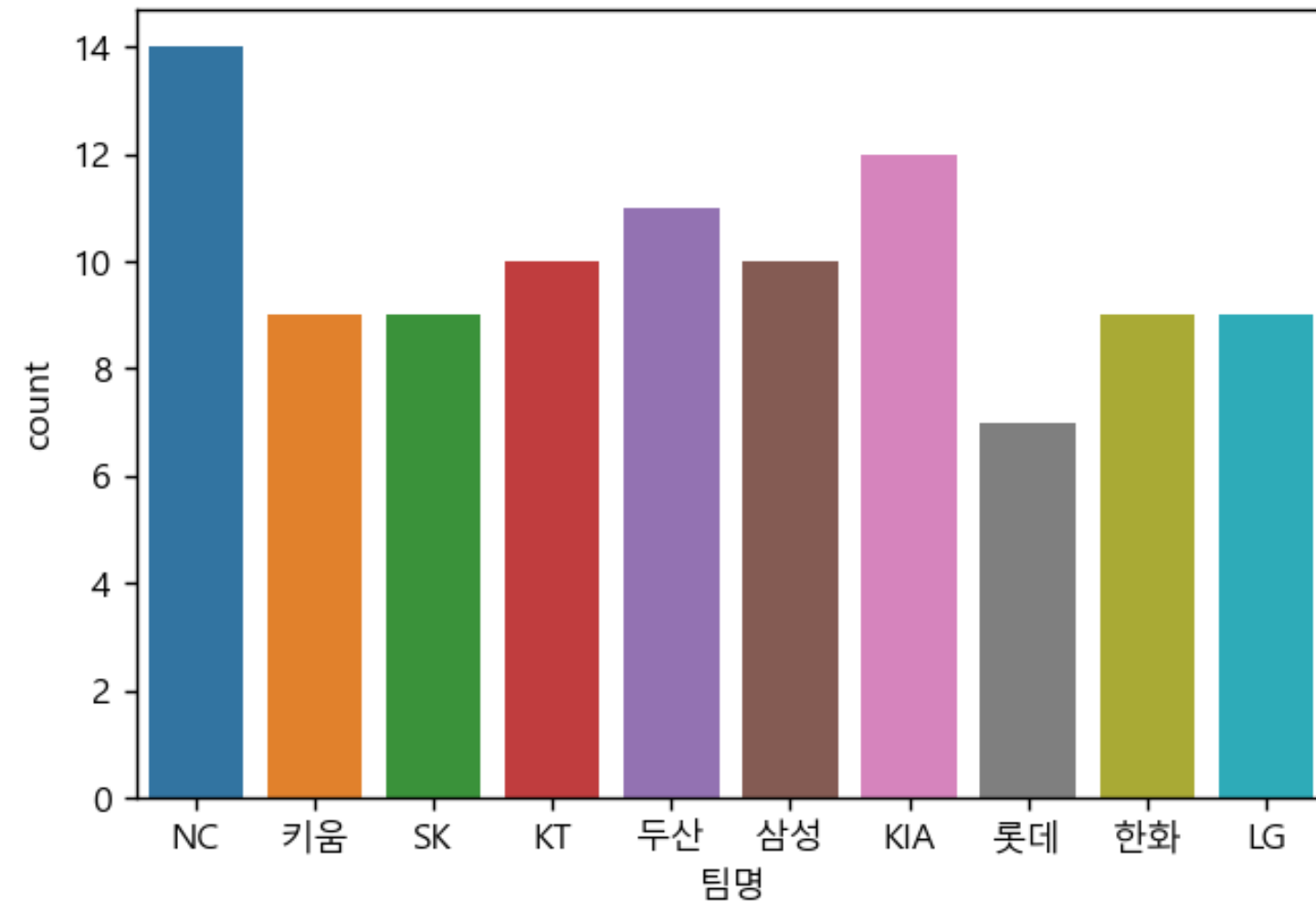
```
sns.barplot(x=x값, y=y값)
```

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(2) EDA : graphic (detailed)

## 개수 그래프 (Count Plot)



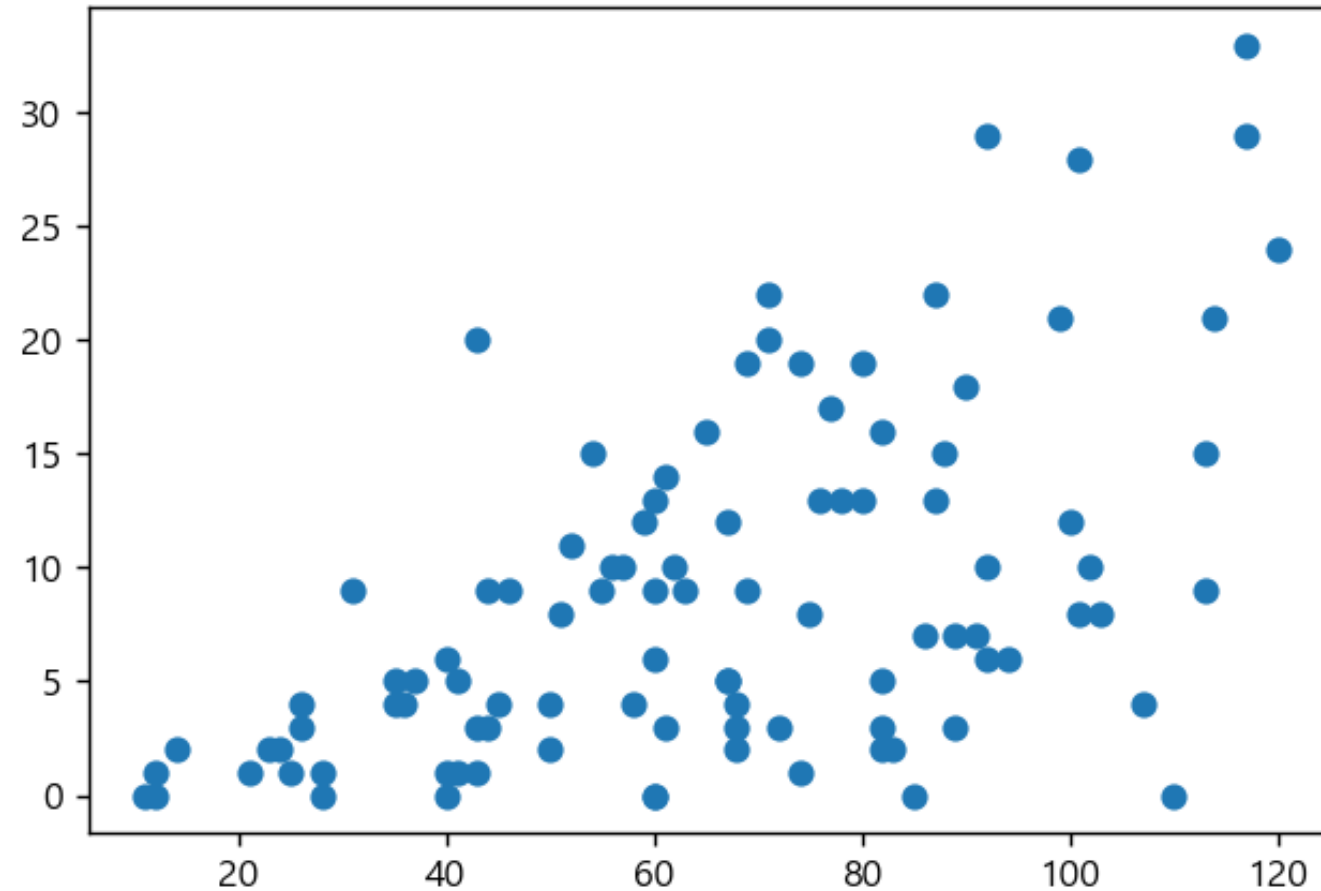
`sns.countplot(x=x값, y=y값)`

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(2) EDA : graphic (detailed)

## 산점도 (Scatter Plot)



```
df.plot(값, kind=scatter)
```

```
plt.scatter(값)
```

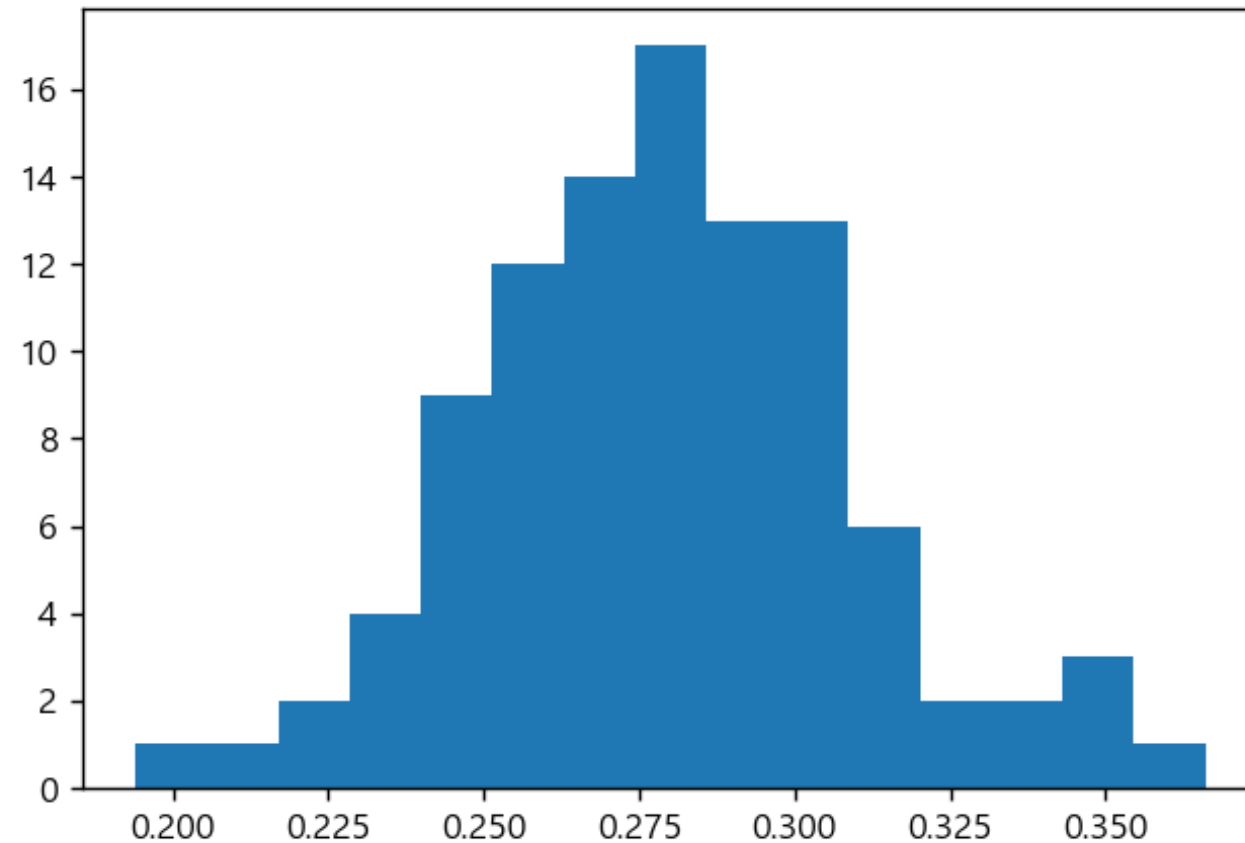
```
sns.scatterplot(x=x값, y=값)
```

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(2) EDA : graphic (detailed)

## 히스토그램 (Histogram)



```
df.plot(값, kind=hist)
```

```
plt.hist(값)
```

```
sns.distplot(x=x값, y=y값)
```

# Data Preprocessing

## ★ 실습과 함께 하는 데이터 전처리 ★

### (2) EDA

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 72456 entries, 0 to 72455
Data columns (total 11 columns):
#   Column                Non-Null Count  Dtype
---  -
0   기준 연도              72456 non-null  int64
1   기준 월                72456 non-null  int64
2   법정동                72456 non-null  object
3   행정동                72456 non-null  object
4   행정동 코드            72456 non-null  int64
5   집계구 코드            72456 non-null  int64
6   평균 보증금 (만원)     72456 non-null  float64
7   평균 월세 (만원)       72455 non-null  float64
8   평균 면적 (m2)         72456 non-null  float64
9   전월세                 72456 non-null  object
10  건물 종류              72456 non-null  object
dtypes: float64(3), int64(4), object(4)
```

	기준 연도	기준 월	행정동 코드	집계구 코드	평균 보증금 (만원)	평균 월세 (만원)	평균 면적 (m2)
count	72456.000000	72456.000000	7.245600e+04	7.245600e+04	72456.000000	72455.000000	72456.000000
mean	2022.034559	6.481754	2.635396e+09	2.109568e+12	9606.568972	23.717508	53.976765
std	0.690776	3.285310	1.320831e+07	5.706023e+09	11220.913150	31.698209	28.569820
min	2021.000000	1.000000	2.611051e+09	2.101051e+12	0.000000	0.000000	9.310000
25%	2022.000000	4.000000	2.626051e+09	2.106051e+12	1500.000000	0.000000	28.200000
50%	2022.000000	6.000000	2.635053e+09	2.109064e+12	5333.330000	15.000000	52.000000
75%	2023.000000	9.000000	2.644056e+09	2.112060e+12	14000.000000	38.200000	74.420000
max	2023.000000	12.000000	2.671033e+09	2.131033e+12	200000.000000	700.000000	263.590000

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(3) Data Transformation

(1) Encoding

(2) Scaling

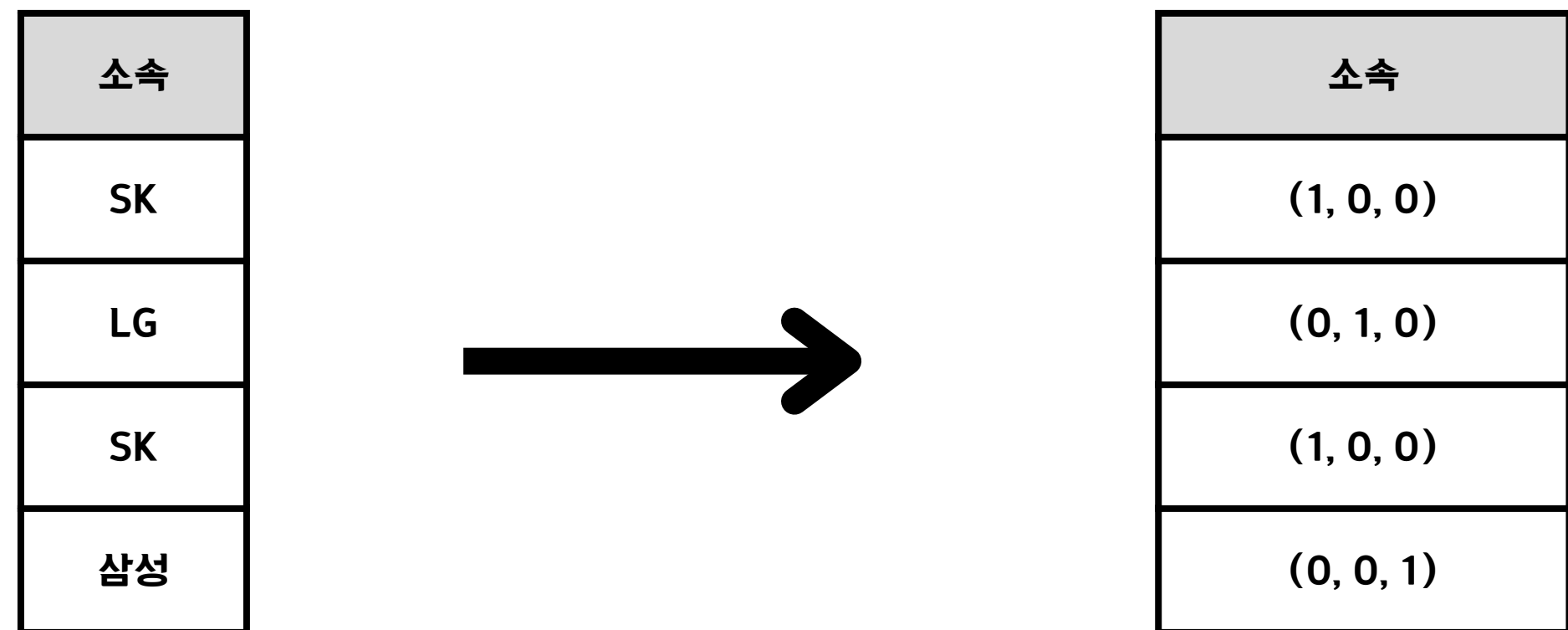
(3) Feature Selection

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

## (3) Data Transformation – Encoding

### (1) One-hot Encoding



# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(3) Data Transformation – Encoding

(2) Label Encoding

소속
SK
LG
SK
삼성



소속
1
2
1
3



# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(3) Data Transformation – Encoding

(3) Ordinal Encoding

생년
1997
2003
1999
2000



생년
1
4
2
3

# Data Preprocessing

## ★ 실습과 함께 하는 데이터 전처리 ★

### (3) Data Transformation – Encoding

```
1 from sklearn.preprocessing import LabelEncoder
2
3 # Select the columns you want to label encode
4 columns_to_encode = ['법정동', '행정동', '전월세', '건물 종류']
5
6 # Initialize the LabelEncoder
7 label_encoder = LabelEncoder()
8
9 # Loop through each column and apply label encoding
10 for column in columns_to_encode:
11     df[column] = label_encoder.fit_transform(df[column])
12
13 display(df)
```

```
1 display(df["법정동"].nunique())
```

222

	기준 연도	기준 월	법정동	행정동	행정동 코드	집계구 코드	평균 보증금 (만원)	평균 월세 (만원)	평균 면적 (㎡)	전월세	건물 종류
0	2021	7	204	188	2611051000	2101051010002	833.33	52.33	27.23	0	3
1	2021	7	184	188	2611051000	2101051010201	1500.00	62.50	54.75	0	1
2	2021	7	207	188	2611051000	2101051020001	3000.00	91.67	54.89	0	1
3	2021	7	207	188	2611051000	2101051020001	28650.00	0.00	55.01	1	1
4	2021	7	207	188	2611051000	2101051020001	2008.33	43.42	25.37	0	3
...	...	...	...	...	...	...	...	...	...	...	...
72451	2023	6	217	207	2635054000	2109054010201	13500.00	0.00	59.96	1	1
72452	2023	6	217	207	2635054000	2109054011801	3000.00	50.00	59.90	0	1
72453	2023	6	217	207	2635054000	2109054012202	2300.00	150.60	89.38	0	1
72454	2023	6	217	207	2635054000	2109054012202	34000.00	0.00	98.71	1	1
72455	2023	6	217	207	2635054000	2109054012301	25000.00	0.00	78.78	1	1

72456 rows × 11 columns

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

## (3) Data Transformation – Scaling

### (1) Min-Max Scaling

$$\frac{x - x_{min}}{x_{max} - x_{min}}$$

생년	→	생년
1997		0
2003		1
1999		1/3
2000		1/2

- 값의 범위 : 0 ~ 1
- 이상치(outlier)에 영향을 많이 받음
- 정규분포가 아니거나 표준편차가 적을 때 사용하기 좋음

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(3) Data Transformation – Scaling

(2) Maximum Absolute Scaling

$$x_{scaled} = \frac{x}{\max(|x|)}$$

생년	→	생년
1997		0.9970
2003		1
1999		0.9980
2000		0.9985

- 값의 범위 :  $-1 \sim 1$
- 이상치(outlier)에 영향을 많이 받음
- sparse한 데이터에 효과적임

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(3) Data Transformation – Scaling

(3) Standard Scaling

$$z = \frac{x - \mu}{\sigma}$$

생년	→	생년
1997		-1.2702
2003		1.5012
1999		-0.3464
2000		0.1155

- 평균이 0이고 표준편차가 1임
- 이상치(outlier)가 있으면 사용 불가능
- 최솟값, 최댓값을 몰라도 사용이 가능함

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(3) Data Transformation – Scaling

(4) Robust Scaling

$$X_{\text{scale}} = \frac{x_i - x_{\text{med}}}{x_{75} - x_{25}}$$

생년
1997
2003
1999
2000



생년
-1.1111
1.5556
-0.2222
0.2222

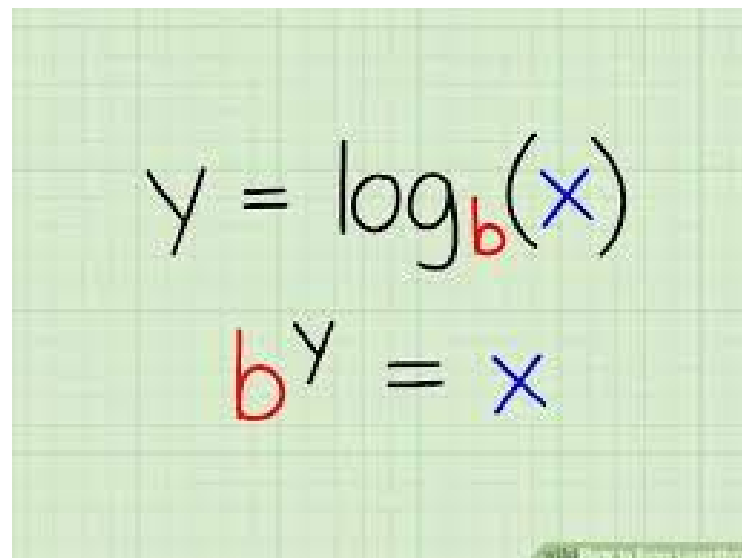
- 중앙값이 0, (75%값과 25%값의 차)가 1
- 이상치(outlier)에 영향을 적게 받음
- 이상치가 필연적일 때 쓰기 좋음

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(3) Data Transformation – Scaling

(5) Log Transformation


$$y = \log_b(x)$$
$$b^y = x$$

생년
1997
2003
1999
2000



생년
3.3004
3.3017
3.3008
3.3010

- 정규분포에 가까운 형태가 됨
- 이상치(outlier)에 영향을 적게 받음
- 표준편차가 클 때 하기 좋음

# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(3) Data Transformation – Feature Selection

(1) Correlation Coefficient

(2) Mutual Information

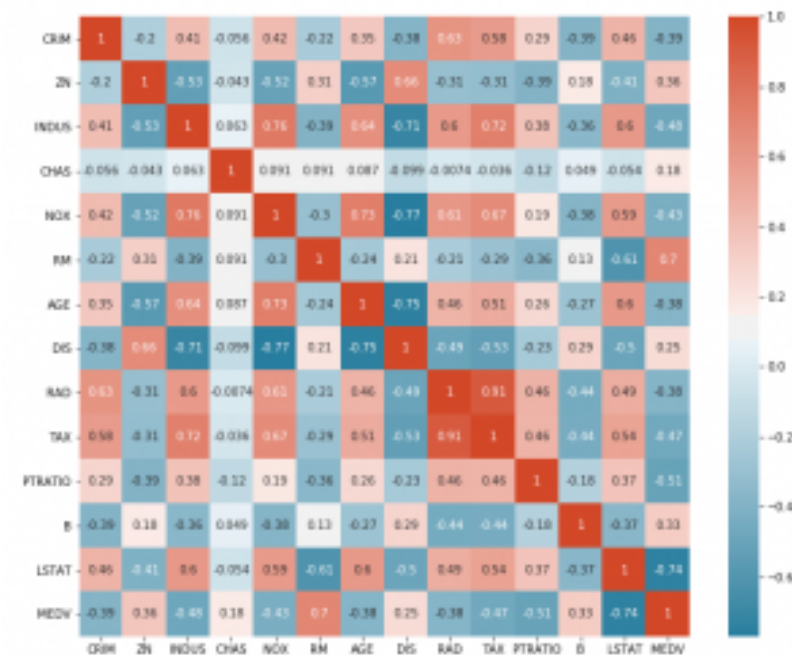


# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

## (3) Data Transformation – Feature Selection

### (1) Correlation Coefficient



상관계수를 기준으로  
feature 간의 연관성을  
파악함

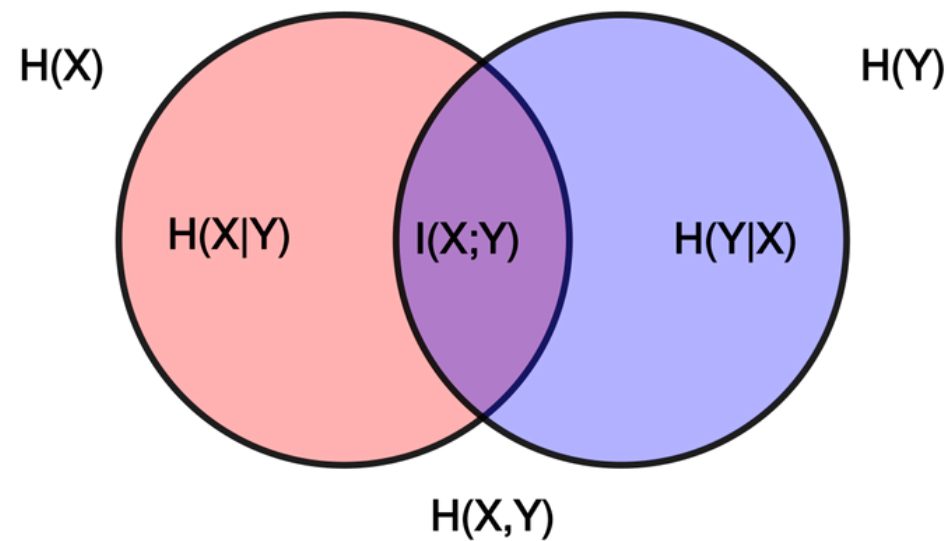
# Data Preprocessing

★ 실습과 함께 하는 데이터 전처리 ★

(3) Data Transformation – Feature Selection

(2) Mutual Information

"Correlation does not imply causation"



독립성을 기준으로  
feature 간의 관계를  
파악함

# Data Preprocessing

## 과제 1

**수업에서 사용한 데이터를 기반으로  
Graphic을 이용한 EDA를 해보기**

line plot, bar plot, barh plot, count plot, scatter plot, histogram

seaborn, df.plot, matplotlib

# Data Preprocessing

## 과제 2

수업에서 사용한 데이터를 기반으로  
Data Scaling을 해보기

min-max, maximum absolute, standard, robust, log

# Data Preprocessing

## 과제 3

수업에서 사용한 데이터를 기반으로  
Feature Selection을 해보기

correlation coefficient, mutual information

수나로움

**THANK YOU**