

In [1]:

02.01 데이터 전처리 기초

[전처리 사용 패키지]

`missingno` : 결측 데이터 검색
`sklearn.impute` : 결측 데이터 대체
`patsy` : 데이터 선택, 변환, 추가, 스케일링
`sklearn.preprocessing` : 스케일링, 변환

1. 결측 데이터 (`msno, sklearn.impute`)1) `missingno` 패키지

결측 검색 대상 : pandas 데이터프레임 상 `NaN`(**not** a number)

주의사항)

`NaN` 값 : 부동소수점 실수 자료형에만 존재
 (원래 정수형, 시간자료형에는 존재하지 않음)

1. 데이터프레임 속 정수값 : `Int64Dtype` 자료형을 명시

2. 데이터프레임 속 시간값 : `parse_dates`로 날짜시간형 파싱!

*그래야 `datetime64[ns]` 자료형이 되어 결측데이터가 `NaT`(**not** a time) 값이 됨

1. 결측데이터 위치 탐색

```
df.isnull()  
df.isnull().sum()  
msno.matrix(df)  
msno.bar(df)
```

2. 결측데이터 대체 (삭제 OR 대체)

- 결측 데이터가 너무 많은 경우 : 해당 데이터 열 전체로 삭제 가능 (가짜 데이터 많이 채우기보단..)
- 결측 데이터가 일부인 경우 : 가장 그럴듯한 값으로 대체(imputation)

[삭제]

`df.dropna(axis=1)` *`axis=1`, 열단위 결측데이터 존재 열 삭제(행단위 `axis=0`)
`df.dropna(axis=1, thresh=7)`, 열단위 결측데이터 존재 열 삭제(7개 이상 비결측 값 있으면 남기, `thresh`)

thresh: 7이하가 몇개 이상 있으면 예외처리하지 마라.

[대체]

sklearn 패키지 - `SimpleImputer` 패키지 ==> `fit_transform` 메서드

ex)

```
from sklearn.impute import SimpleImputer  
  
imputer = SimpleImputer(strategy = 'most_frequent')  
df = pd.DataFrame(imputer.fit_transform(df), columns = df.columns)
```

*`strategy` 인수 : `mean, median, most_frequent`

`mean` : 데이터가 실수 연속값인 경우, 값의 분포가 대칭적일 때

`median` : 데이터가 실수 연속값인 경우, 값의 분포가 비정규적인 경우

`most_frequent` : 데이터가 범주값, 정수값인 경우 (가장 많이 나온값으로 채우면 그래도 가장 많이 맞을 수 있다.)

2. 데이터 가공 (`patsy`)

1) patsy 패키지

*design_matrix = dmatrix
 *formula = 데이터 열 이름 기반으로 구성된 문자열 (1열 + 2열 + 0 = 상수항 생성하지 말고 1열과 2열만 가져와라)

[데이터 선택]

```
dmatrix(formula문자열, data)
```

[데이터 변형]

```
dmatrix(formula문자열, data)
```

*formula 문자열 = 수식을 포함해, 원하는 컬럼대로 기입 (def로 만든 함수등 파이썬이 해석 가능한 모든 명령 문자열로 기입)

```
ex) dmatrix("x1 + np.log(np.abs(x2))", df)
```

```
ex) dmatrix("x1 + x2 + x1:x2", df)
```

* $x1:x2 = x1$ 과 $x2$ 를 곱한 값 = 상호작용항. 회귀분석 시 반드시 체크해야 함

*각각 들어갔을 때는 없었던 효과가, 상호작용에서는 보이는 경우가 있기 때문에 반드시 체크!

```
ex) dmatrix("x1 + x2 + I(x1+x2) + I(x1/x2) + 0", df)
```

*상호작용항 제외, 연산값 컬럼을 만들고 싶다면, I() 연산자 활용

*맨 끝에 +0은 상수항 만들지 말라는 의미 (상수항은 회귀분석 용 데이터 시 필수이긴 함)

*스케일링 작업 = 선형회기 분석 시, 조건수(condition number)의 영향으로 데이터의 다중공선성이 있을 수 있음

==> 이를 잡기 위해 스케일링 작업을 해줘야 함 (일종의 표준정규분포로 정규화하는 작업)

==> 평균 0, 표준편차 1로 변환

center(): 평균을 0으로 스케일링

standardize(): 평균을 0으로하고 표준편차를 1로 스케일링

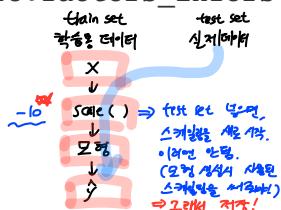
scale(): standardize() 과 같음

```
ex) dmatrix("center(x1) + 0", df)
```

* $x1 - \text{mean of } x1 = \text{center}(x1)$

* $\text{center}(x1)$ 의 값들은 dm.design_info.factors_infor 내부에 저장됨

*왜 스케일링 값을 저장하는가? (18p)



3. 데이터 가공 (sklearn.preprocessing)

1) sklearn.preprocessing

- StandardScaler 클래스

```
scaler = StandardScaler()
scaler.fit_transform(x)
```

- RobustScaler 클래스

```
scaler = RobustScaler()
scaler.fit_transform(x)
```

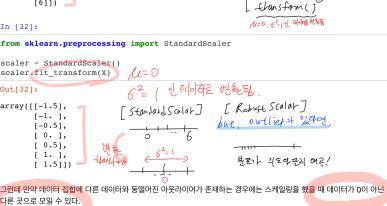
데이터가 0 주위에 분산된
로 모이도록 스케일링!

그런데 만약 데이터 집합에 다른 데이터와 동떨어진 아웃라이어가 존재하는 경우에는 스케일링을 했을 때 데이터가 0이 아닌 다른 곳으로 모일 수 있다.

이 때는 RobustScaler 클래스를 사용한다. 이 클래스는 중앙값이 0, IQR(interquartile range)이 1이 되도록 변환하기 때문에 아웃라이어가 섞여 있어도 대부분의 데이터는 0 주위에 남아있게 된다.

In [35]:

```
from sklearn.preprocessing import RobustScaler
scaler = RobustScaler()
scaler.fit_transform(X2)
```



- PolynomialFeatures 클래스 (21p)

`PolynomialFeatures` 클래스는 입력 데이터 x 를 다음과 같이 여러개의 다항식으로 변환한다.
 $x \rightarrow [1, x, x^2, x^3, \dots]$

다음과 같은 입력 인수를 가진다.

- `degree` : 차수
- `include_bias` : 상수항 생성 여부

In [36]:

```
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree=2)
poly.fit_transform(X)
```

Out[36]:

```
array([[ 1.,  0.,  0.],
       [ 1.,  1.,  1.],
       [ 1.,  2.,  4.],
       [ 1.,  3.,  9.],
       [ 1.,  4., 16.],
       [ 1.,  5., 25.],
       [ 1.,  6., 36.]])
```

- FunctionTransformer 클래스(21, 22p)

FunctionTransformer 클래스는 사용자가 지정한 함수를 사용하여 입력값 x 를 변환한다.

$x \rightarrow [f_1(x), f_2(x), f_3(x), \dots]$

각도 데이터는 그각도로 예측 문제 정답값 x \rightarrow 전처리 필요

데이터 변환은 비선형 회귀분석에서 원하는 목표값을 더 잘 예측하기 위한 새로운 데이터를 만들 때 사용된다. 예를 들어 '360도'와 같은 각도 데이터는 그 자체로 예측문제의 입력값으로 넣을 수 없다. 0도와 360도, 10도와 370도는 사실 같은 각도지만 다른 숫자로 표현되기 때문이다. 이 때는 각도 θ 를 다음과 같이 삼각함수값의 쌍으로 바꾸면 같은 각도를 같은 숫자쌍으로 표현할 수 있다.

$$\theta \rightarrow (\sin \theta, \cos \theta)$$

다음 예제 코드에서 원래 데이터 x 는 각도 표시로 되어 있다.

하지만 x_2 는 삼각함수로 변환되었다. 각도 0도와 각도 360도가 모두 같은 값인 (0,1)로 표현된 것을 확인할 수 있다.

In [38]:

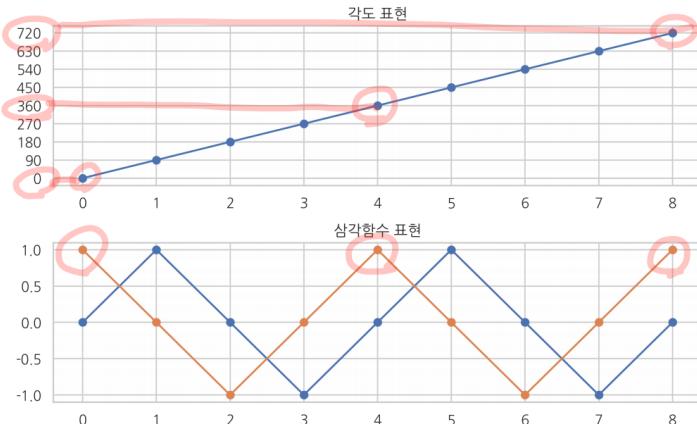
```
from sklearn.preprocessing import FunctionTransformer

def degree2sincos(X):
    x0 = np.sin(X * np.pi / 180)
    x1 = np.cos(X * np.pi / 180)
    X_new = np.hstack([x0, x1])
    return X_new

X2 = FunctionTransformer(degree2sincos).fit_transform(X)
```

1

변경까지!



In []:

02.02 범주형 데이터 처리

*실수형 데이터 = 연속적, 크기 및 가치비교가 가능

*대부분의 데이터 모형 = 숫자만 입력으로 받을 수 있음. 따라서, 범주형 데이터는 숫자로 변환해야함

*범주형 데이터의 변형

1. 더미변수화
2. 카테고리 임베딩

1. 더미변수화 (풀랭크 / 축소랭크) 클래스=k개

"부터 K까지 값을 가질 수 있는 더미변수 \Rightarrow K개 더미"

1) 더미변수

더미변수 : 0, 1 (이진 변수), 특징 유무만 표현 ex) 남자/여자
더미변수의 갯수 = 클래스의 갯수 (2p, 혈액형)

혈액형은 4개의 더미변수 (d_1, d_2, d_3, d_4)로 표현할 수 있다. d_1 은 혈액형이 A형인지 아닌지를 나타내는 더미변수이고 d_2 은 혈액형이 B형인지 아닌지를 나타내는 더미변수, d_3, d_4 는 각각 혈액형이 O형인지 아닌지를 나타내는 더미변수, AB형 인지 아닌지를 나타내는 더미변수.

$$\begin{aligned} x = A\text{형} &\rightarrow d_1 = 1, d_2 = 0, d_3 = 0, d_4 = 0 \\ x = B\text{형} &\rightarrow d_1 = 0, d_2 = 1, d_3 = 0, d_4 = 0 \\ x = O\text{형} &\rightarrow d_1 = 0, d_2 = 0, d_3 = 1, d_4 = 0 \\ x = AB\text{형} &\rightarrow d_1 = 0, d_2 = 0, d_3 = 0, d_4 = 1 \end{aligned}$$

위 예제들에서 볼 수 있듯이 1부터 K까지의 값을 가질 수 있는 범주형값은 K개의 더미변수 벡터로 표시할 수 있다. 각 더미 변수는 특정한 하나의 카테고리값인가 아닌가를 나타내는 지시자(indicator)가 된다.

2) (풀랭크) patsy 패키지를 사용한 더미변수화 (dmatrix)

*주의사항 : + 0을 추가하면, 풀랭크 방식으로 더미변수 생성

*dmatrix("x + 0", df) 장수상 X.

C연산자 \Rightarrow 1. 데이터가 범주형 값이지만, 정수로 표시된 경우, 범주형 값임을 명시적으로 지정

C연산자 \Rightarrow 2. 더미변수의 순서도 바꿀 수 있음

3) (축소랭크) 더미변수화

*주의사항 : +0이 없음

*하나의 범주값을 기준값으로 지정(항상 1), 추가적인 특성을 나타내는 더미변수 1 출력

*기준이 되는 더미변수는 이름이 'Intercept'가 됨. (5p)

지금까지 설명한 더미변수 방식을 풀랭크(full-rank) 방식이라고 한다. 이와 달리 축소랭크(reduced-rank) 방식에서는 특정한 하나의 범주값을 기준값(reference, baseline)으로 하고 기준값에 대응하는 더미변수의 가치는 항상 1으로 놓는다.

다른 범주형 값을 가지는 경우는 기준값 더미변수 1이고 추가적인 특성을 나타내는 더미변수 1인 것으로 간주한다. 예를 들어 다음 축소랭크 방식은 $x = A$ 를 기준값으로 하는 경우이다.

$$\begin{aligned} x = A &\rightarrow d_1 = 1, d_2 = 0, d_3 = 0, d_4 = 0 \\ x = B &\rightarrow d_1 = 1, d_2 = 1, d_3 = 0, d_4 = 0 \\ x = AB &\rightarrow d_1 = 1, d_2 = 0, d_3 = 1, d_4 = 0 \\ x = O &\rightarrow d_1 = 1, d_2 = 0, d_3 = 0, d_4 = 1 \end{aligned}$$

반대로 $x = B$ 를 기준값으로 하면 다음과 같아진다.

$$\begin{aligned} x = A &\rightarrow d_1 = 1, d_2 = 1, d_3 = 0, d_4 = 0 \\ x = B &\rightarrow d_1 = 0, d_2 = 1, d_3 = 0, d_4 = 0 \\ x = AB &\rightarrow d_1 = 0, d_2 = 1, d_3 = 1, d_4 = 0 \\ x = O &\rightarrow d_1 = 0, d_2 = 1, d_3 = 0, d_4 = 1 \end{aligned}$$

만약 기준 범주값을 다른 값으로 바꾸려면 Treatment() 함수를 formula 문자열에서 사용한다.

```
In [10]:  
dmatrix("C(x, Treatment('Male'))", df1)  
  
Out[10]:  
DesignMatrix with shape (2, 2)  
Intercept C(x, Treatment('Male'))[T.Female]  
 1 0  
 1 1  
Terms:  
 'Intercept' (column 0)  
 "C(x, Treatment('Male'))" (column 1)
```

4) 두개의 범주형 변수가 있는 경우

- 통합 축소형 방식
- 상호작용 방식

1. 통합축소 방식 (7p)

통합 축소형 방식에서는 다음과 같이 3개의 더미변수 (d_1, d_2, d_3)를 만든다. d_1 은 x_1 이 A, x_2 가 X라는 기준값을 나타낸다. d_2 은 x_1 이 A 아닌 B라는 것을 표시한다. d_3 은 x_2 가 X 아닌 Y라는 것을 표시한다.

$$\begin{aligned} x_1 = A, x_2 = X &\rightarrow d_1 = 1, d_2 = 0, d_3 = 0 \\ x_1 = B, x_2 = X &\rightarrow d_1 = 1, d_2 = 1, d_3 = 0 \\ x_1 = A, x_2 = Y &\rightarrow d_1 = 1, d_2 = 0, d_3 = 1 \\ x_1 = B, x_2 = Y &\rightarrow d_1 = 1, d_2 = 1, d_3 = 1 \end{aligned}$$

기준, A에서 제외되는 기준값, X에서 제외되는 기준값
X, Y, A, B
(A>B) (X>Y)

```
In [13]:  
dmatrix("x1 + x2", df4)
```

2. 상호작용 방식 (7p) \rightarrow 카테고리 변수 생성

상호작용 방식은 두 범주형 변수를 결합해 각각의 변수의 조합을 나타내는 새로운 범주형 변수를 만드는 방식이다. 즉 앞의 예제에서는 범주형 독립변수가 하나가 되고 대신 범주형 값이 두 독립변수의 범주형 값들의 조합인 AX, BX, AY, BY의 네가지가 된다.

```
In [14]:  
dmatrix("x1*x2", df4)  
Out[14]:  
DesignMatrix with shape (4, 4)  
x1(A)x2(X) x1(B)x2(X) x1(A)x2(Y) x1(B)x2(Y)
```

A X Y \Rightarrow A는 X는 카테고리
B X Y \Rightarrow B는 Y는 카테고리
(A, B)
(X, Y)
[A, B, X, Y] \Rightarrow 카테고리변수!

2. 카테고리 임베딩

- 범주값 대신 범주값의 특성을 나타내는 연속값/연속값 벡터를 사용
- 운동선수의 이름을 나타내는 범주값 \rightarrow 나이, 연봉, 신체능력치 등으로 대체

- 지역명을 나타내는 범주값 -> 지역의 면적, 인구수 등으로 대체

In []:

04.01 회귀분석 예제

make_regression 예제(4.1.2)

- 가상의 회귀분석 문제를 만들어줌

- 2개의 독립변수가 있다면, 1개만 종속변수와 상관관계 갖도록 설계 가능
- 두 변수가 독립이 아닌 다중공선성 설계 가능

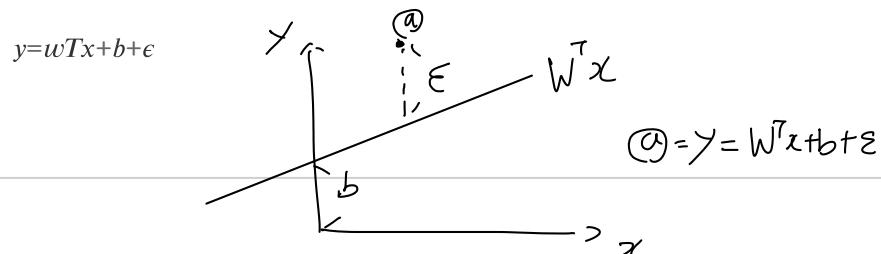
make_regression() 명령은 내부적으로 다음 과정을 거쳐 가상의 데이터를 만듬

1. 독립변수 데이터 행렬 x 를 무작위로 만든다.
2. 종속변수와 독립변수를 연결하는 가중치 벡터 w 를 무작위로 만든다.
3. x 와 w 를 내적하고 y 절편 b 값을 더하여 독립변수와 완전선형인 종속변수 벡터 y_0 를 만든다.
4. 기댓값이 0이고 표준편차가 noise인 정규분포를 이용하여 잡음 ϵ 를 만든다.
5. 독립변수와 완전선형인 종속변수 벡터 y_0 에 잡음 ϵ 를 더해서 종속변수 데이터 y 를 만든다.

* 0개 feature, 1개 feature X.

$$\begin{matrix} n \times 1 \\ y_0 \\ \vdots \end{matrix} = \begin{matrix} n \times m \\ X \\ \vdots \end{matrix} \begin{matrix} m \times 1 \\ w \\ \vdots \end{matrix} + \begin{matrix} n \times 1 \\ b \\ \vdots \end{matrix}$$

$$\begin{bmatrix} y_0(1) \\ \vdots \\ y_0(n) \end{bmatrix} = \begin{bmatrix} x_{11} & \dots & x_{1m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \dots & x_{nm} \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix} + \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$



In []:

04.02 선형회귀분석의 기초

회귀분석 : 종속변수의 값을 실제값과 가장 비슷하게 출력하는 $f(x)$ 함수를 찾는 과정
* $f(x)$ 가 선형이면, 선형회귀분석이다.

- $f(x)$ 의 계수 = 가중치 벡터 = 선형회귀모형의 모수 (parameter) (1p 상단)

만약 $f(x)$ 가 다음과 같은 선형함수면 이 함수를 선형회귀모형(linear regression model)이라고 한다. 선형회귀모형을 사용하는 회귀분석은 선형회귀분석이라고 한다.

$$\hat{y} = w_0 + w_1x_1 + w_2x_2 + \dots + w_Dx_D = w_0 + w^T x$$

위 식에서 독립변수 $x = (x_1, x_2, \dots, x_D)$ 는 D 차원 벡터다. 가중치 벡터 $w = (w_0, \dots, w_D)$ 는 함수 $f(x)$ 의 계수 (coefficient)이자 이 선형회귀모형의 모수(parameter)라고 한다.

1) 상수항 결합

- $y = w_0 \cdot Tx_0$, 상수항도 포함해 내적으로 선형회귀분석 식을 간단히 하기 위함
- statsmodels 패키지 - add_constant 함수
- 1p, 2p 상단

회귀분석모형 수식을 간단하게 만들기 위해 다음과 같이 상수항을 독립변수 데이터에 추가하는 것을 상수항 결합(bias augmentation)작업이라고 한다.

$$X_0 = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1D} \\ 1 & x_{21} & x_{22} & \dots & x_{2D} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & x_{N2} & \dots & x_{ND} \end{bmatrix}$$

상수항 결합을 하면 모든 원소가 1인 벡터가 입력 데이터 행렬에 추가된다.

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1D} \\ x_{21} & x_{22} & \dots & x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{ND} \end{bmatrix} \rightarrow X_0 = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1D} \\ 1 & x_{21} & x_{22} & \dots & x_{2D} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & x_{N2} & \dots & x_{ND} \end{bmatrix}$$

이렇게 되면 전체 수식이 다음과 같이 상수항이 추가된 가중치 벡터 w 와 상수항이 추가된 입력 데이터 벡터 x 의 내적으로 간단히 표시된다.

$$f(x) = w_0 + w_1x_1 + w_2x_2 + \dots + w_Dx_D = \begin{bmatrix} 1 & x_1 & x_2 & \dots & x_D \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_D \end{bmatrix} = x^T w_0 + w^T x$$

```
x0 = np.arange(10).reshape(5, 2)
x0
Out[1]:
array([[0, 1],
       [2, 3],
       [4, 5],
       [6, 7],
       [8, 9]])
In [2]:
import statsmodels.api as sm
x = sm.add_constant(x0)
Out[2]:
array([[1, 0, 1, 1],
       [1, 2, 2, 3],
       [1, 4, 4, 5],
       [1, 6, 6, 7],
       [1, 8, 8, 9]])
```

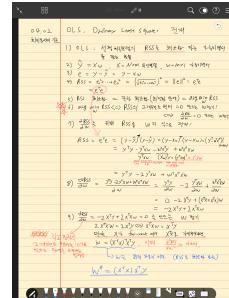
2) 최소자승법 (OLS, Ordinary Least Square)

- 잔차제곱합(RSS)을 최소화하는 가중치 벡터를 구하는 방법

*벡터의 크기 = L2norm으로. 잔차벡터의 norm최소화

= norm은 각 원소들의 제곱 합에 squared = RSS 도출

<OLS 전개>



3) 직교방정식 (normal equation) (3p 하단)

- 그레디언트가 0벡터가 되는 관계를 나타냄

행렬미분법칙 1: 선형 모형

선형 모형을 미분하면 그라디언트 벡터는 가중치 벡터이다.

$$\frac{\partial f(x)}{\partial x} = w^T x \quad \text{그리디언트} \Rightarrow \frac{\partial f}{\partial x} = \nabla f \quad (4.4.25)$$

$$\nabla f = \frac{\partial f}{\partial x_1} = \frac{\partial f}{\partial x_2} = \dots = \frac{\partial f}{\partial x_N} \quad (4.4.26)$$

$$\frac{\partial(w^T x)}{\partial x} = \begin{bmatrix} \frac{\partial(w^T x)}{\partial x_1} \\ \vdots \\ \frac{\partial(w^T x)}{\partial x_N} \end{bmatrix} = \begin{bmatrix} \frac{\partial(w_1x_1 + w_2x_2 + \dots + w_Nx_N)}{\partial x_1} \\ \vdots \\ \frac{\partial(w_Nx_1 + w_1x_2 + \dots + w_Nx_N)}{\partial x_N} \end{bmatrix} = \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} = w$$

- 2 가지 성질 (4p)

1. 모형에 상수항 존재 시, SUM(잔차벡터의 원소) = 0 (잔차 평균은 0)
2. x 데이터의 평균값에 대한 예측 = y 데이터의 평균값

(1) 모형에 상수항이 있는 경우에 잔차 벡터의 원소의 합은 0이다. 즉, 잔차의 평균은 0이다.

$$\sum_{i=0}^N e_i = 0$$

(2) x 데이터의 평균값 \bar{x} 에 대한 예측값은 y 데이터의 평균값 \bar{y} 이다.

$$\bar{y} = w^T \bar{x}$$

4) Numpy를 이용한 선형 회귀분석

- sklearn의 make_regression으로 선형회귀 모델, 데이터 생성
- np.linalg.inv(넘파이 선형대수 기능)으로 OLS해를 직접 구하기
-> 잡음 때문에 같지는 않지만 비슷한 가중치 벡터 구할 수 있음

5) scikit-learn 패키지를 이용한 선형회귀분석

6) statsmodels 패키지를 이용한 선형회귀분석 (주로 이걸 사용, 8p)

statsmodels 패키지에서는 OLS 클래스를 사용하여 선형 회귀분석을 실시한다. OLS 클래스 사용법은 다음과 같다.

1. 독립변수와 종속변수가 모두 포함된 데이터프레임 생성. 상수항 결합은 하지 않아도 된다.

1. OLS 클래스 객체 생성. 이 때 from_formula 메서드의 인수로 종속변수와 독립변수를 지정하는 formula 문자열을 넣는다. data 인수로는 독립변수와 종속변수가 모두 포함된 데이터프레임을 넣는다.

```
model = OLS.from_formula(formula, data=df)
```

또는 독립변수만 있는 데이터프레임 dfx 와 종속변수만 있는 데이터프레임 dfy 를 인수로 넣어서 만들 수도 있다. 이 때는 독립변수만 있는 데이터프레임 dfx 가 상수항을 가지고 있어야 한다.

model = OLS(dfy, dfx) \Rightarrow 종속변수와 독립변수가 포함된 데이터프레임 (dfx에는 상수항 추가!)

1. fit 메서드로 모형 추정. scikit-learn 패키지와 달리 추정 결과는 별도의 RegressionResults 클래스 객체로 출력된다.

```
result = model.fit()
```

2. RegressionResults 클래스 객체는 결과 리포트용 summary 메서드와 예측을 위한 prediction 메서드를 제공한다.

```
print(result.summary())
```

```
y_new = result.predict(x_new)
```

이 때, 예측을 위한 데이터는 추정시와 동일하게 상수항 결합을 해 주어야 한다.

상수항은 하지 말기.

위 1차원 데이터 예제를 statsmodels의 OLS 명령으로 선형회귀를 하면 다음과 같다. 우선 독립변수와 종속변수가 모두 포함된 데이터프레임 생성. 상수항 결합은 하지 않아도 된다.