Regression and Time Series Analysis

1. Introduction to regression and time series analysis

Presenter: Hayun Lee

Contents

1.1 Introduction to Data Analysis

Data Analysis

- **When given data, data analysis means that**
  - Understand the relationship between data
  - The process of creating new (output) data we want using the identified relationships

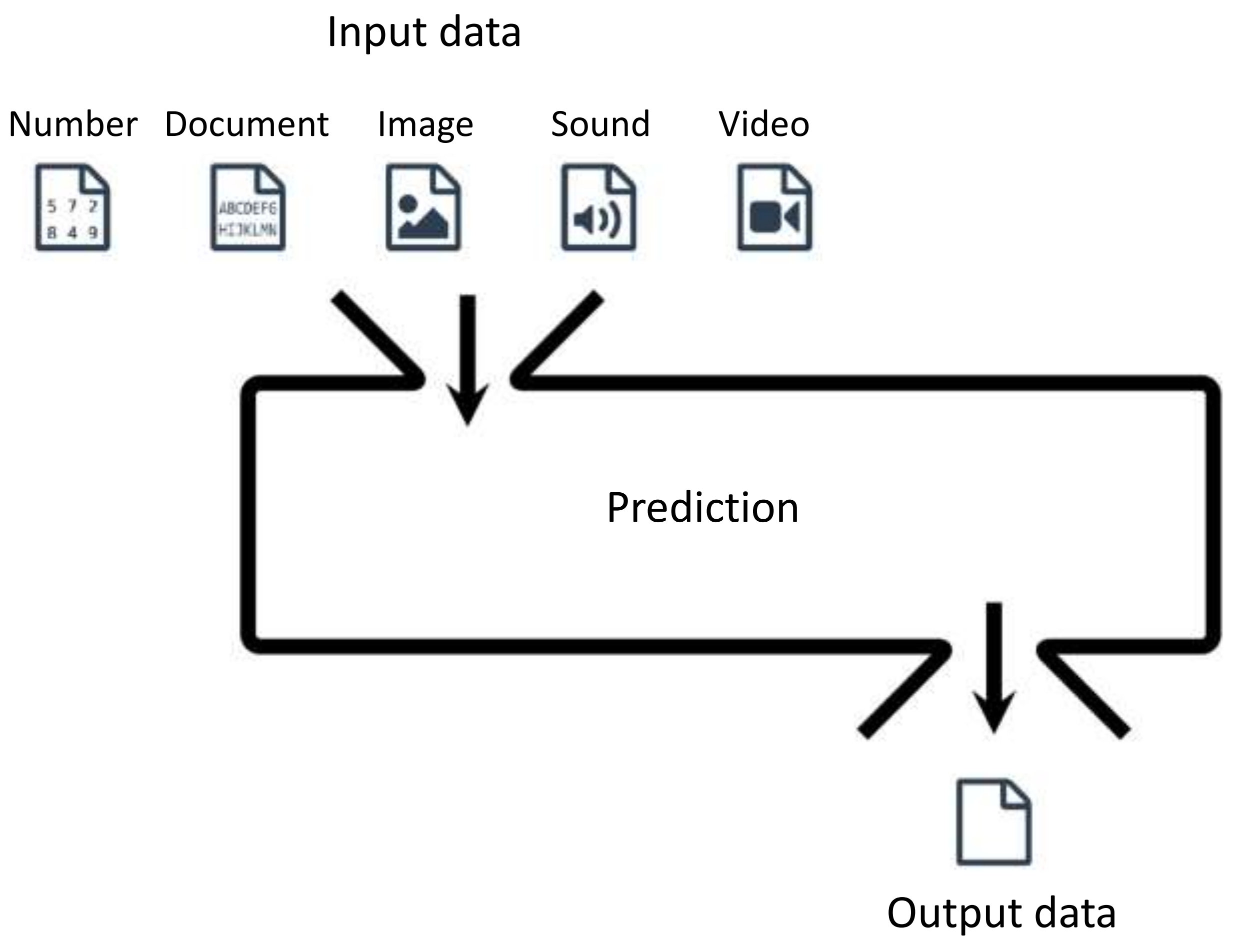
- *Data analysis is a process of inspecting, cleansing, transforming and modeling data with the goal of discovering useful information, informing conclusions and supporting decision-making. – Wikipedia*

- **Problems of data analysis**
  - Prediction
  - Clustering
  - Approximation
  - …

Example) Prediction

- Outputs different data as a result of data analysis when various input data such as numbers, documents, images, audio, and video are given.

- The term prediction in data analysis **does not include the meaning of the future in time**.
  - Time series analysis → forecasting



Input and Output Data

- In the prediction problem, it should be possible to classify data types into two types of data
  - *Input data* and *output data*

- **Input Data (denoted X)**
  - Data on which analysis is based
  - Synonyms: independent variable, feature, explanatory variable

- **Output Data (denoted Y)**
  - Purpose data to estimate or predict
  - Synonyms: dependent variable

Rule-based and Training-based Methods

- **Rule-based Methods**
  - Create rules or algorithms by a person

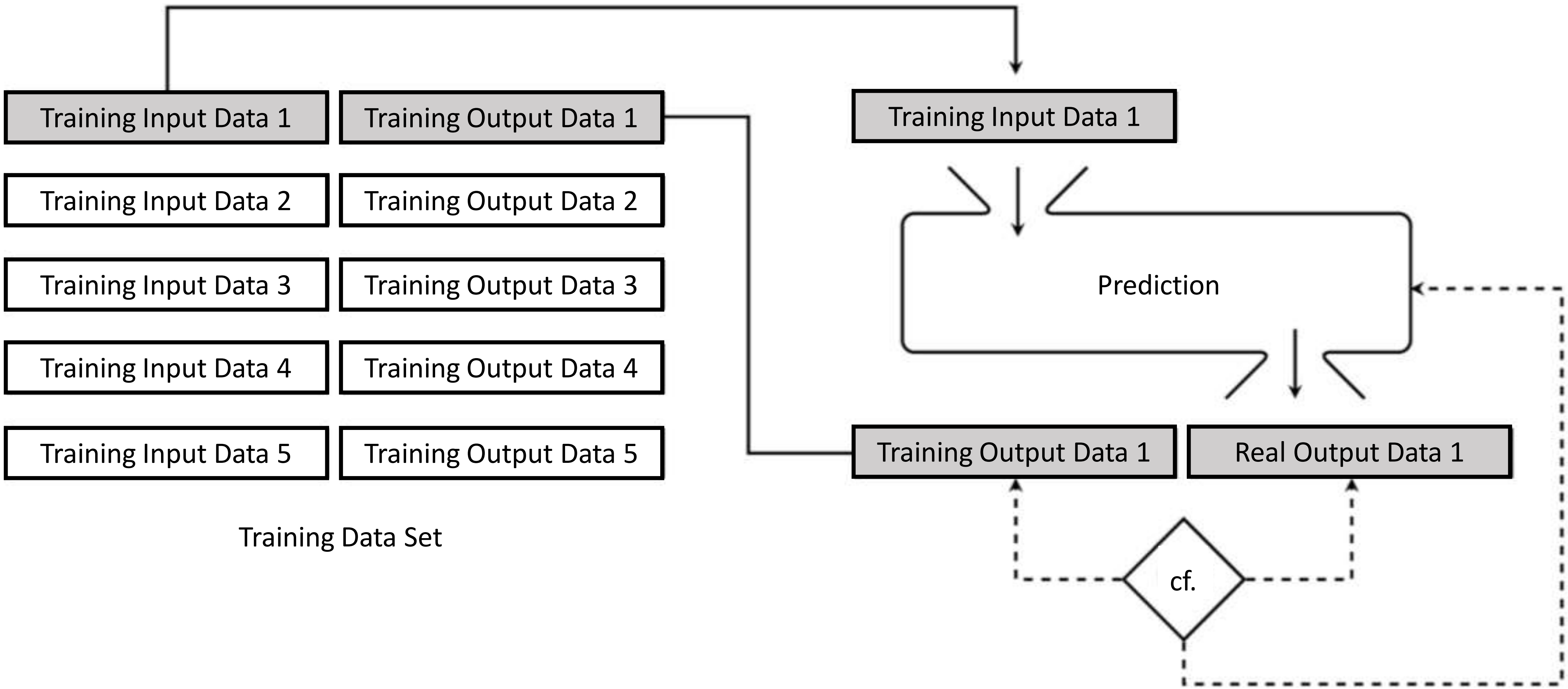
- **Training-based Methods (= Data-based Methods)**
  - Allow computers to create rules on their own by displaying a large amount of data to your computers

Supervised Learning

- *Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. - Wikipedia*

- To use a training-based prediction methodology, **a training data set must be created by a person**.
  - Training data set → Set(<input data, target>)

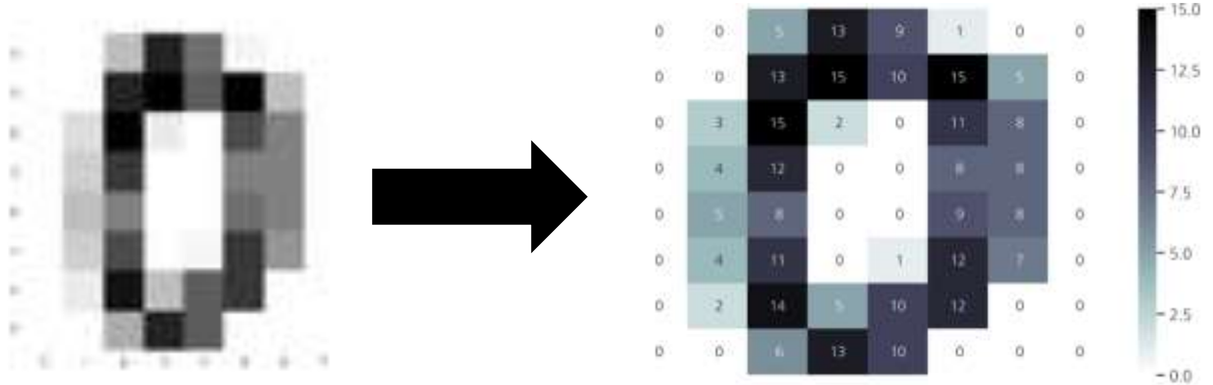
Principles of Supervised Learning

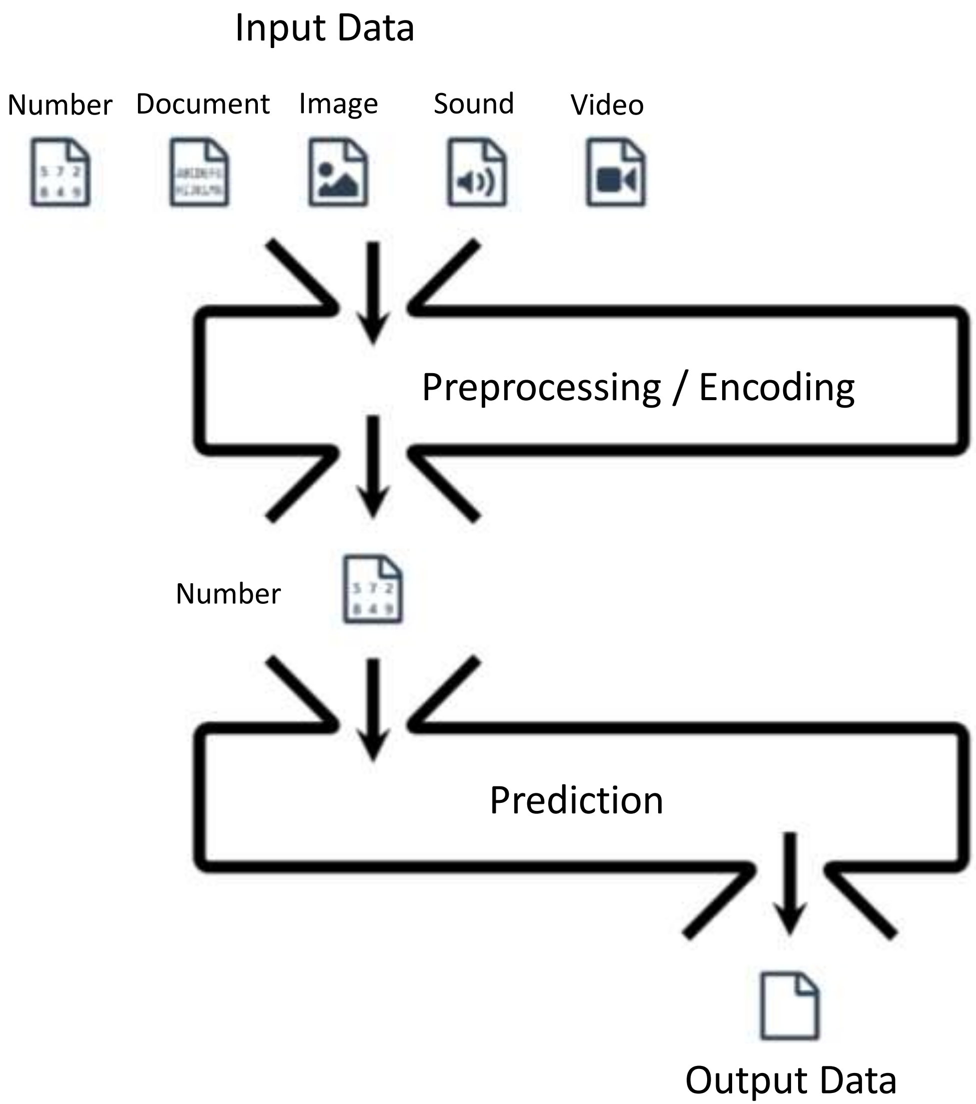


Preprocessing and Encoding

- Data such as documents and images must be converted into ***"number" data*** that can be processed by a computer through a process called ***preprocessing*** or ***encoding***.

- Example: MNIST



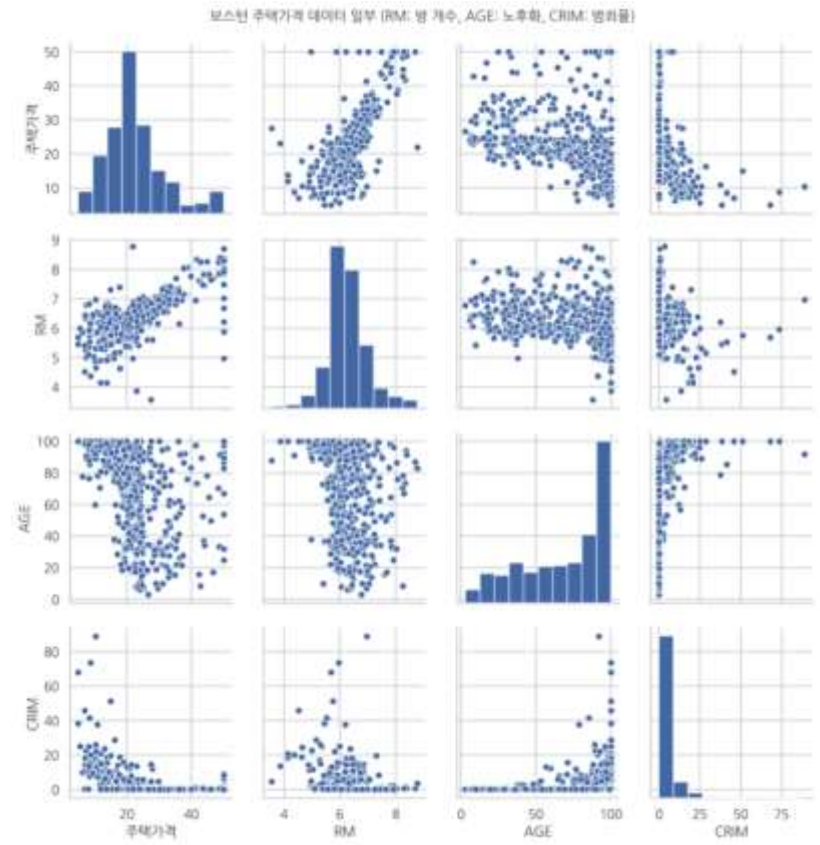


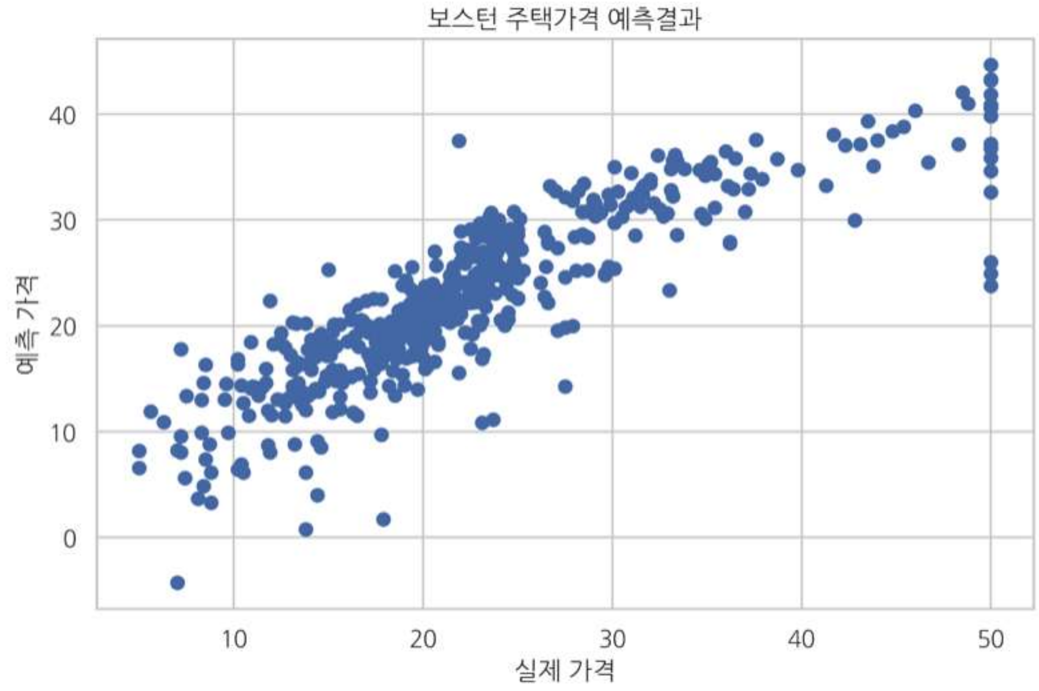
Regression Analysis and Classification

- The prediction problem depends on whether the data to be output is a number value or a category value.
  - Output = number → Regression analysis
  - Output = category → Classification

Example) Regression Analysis

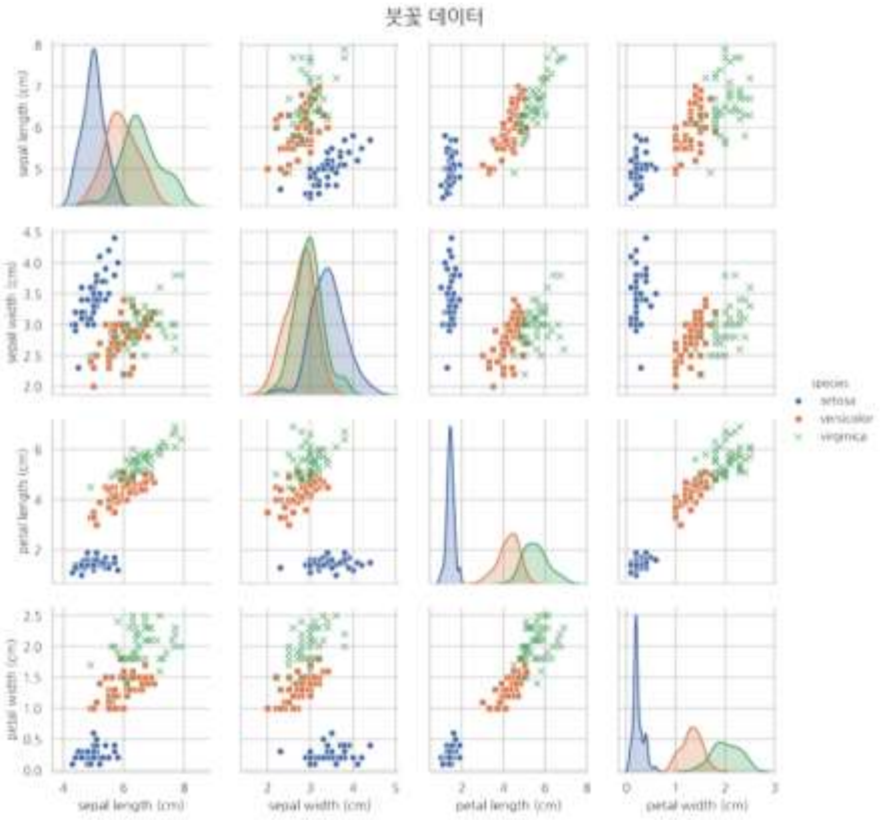
- House price prediction (provided by the scikit-learn package)

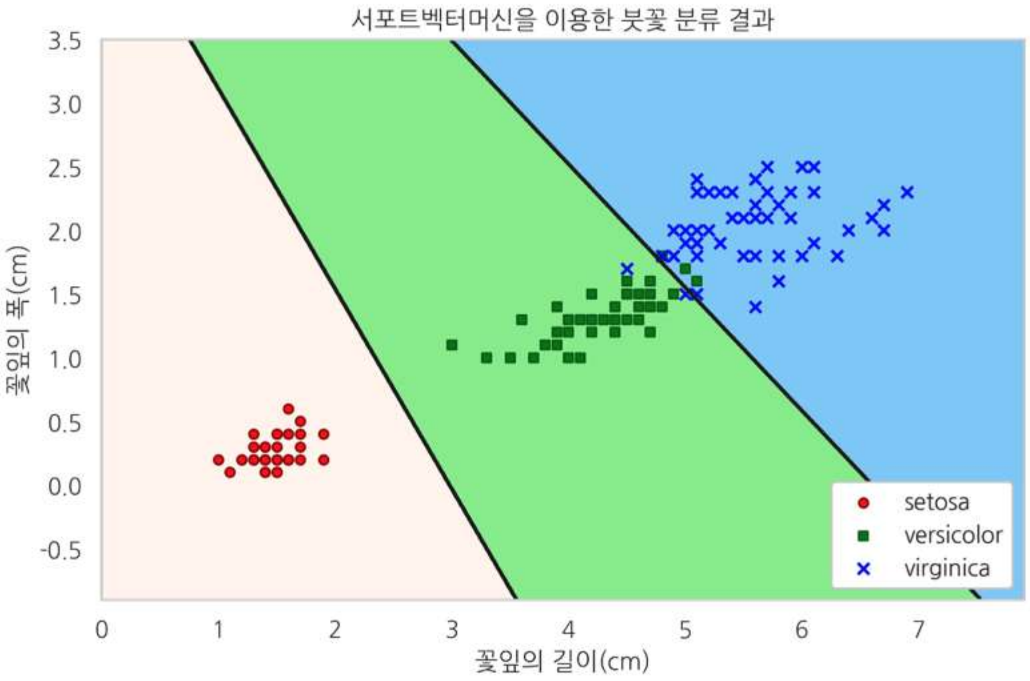




Example) Classification

- Iris classification (provided by the scikit-learn package)



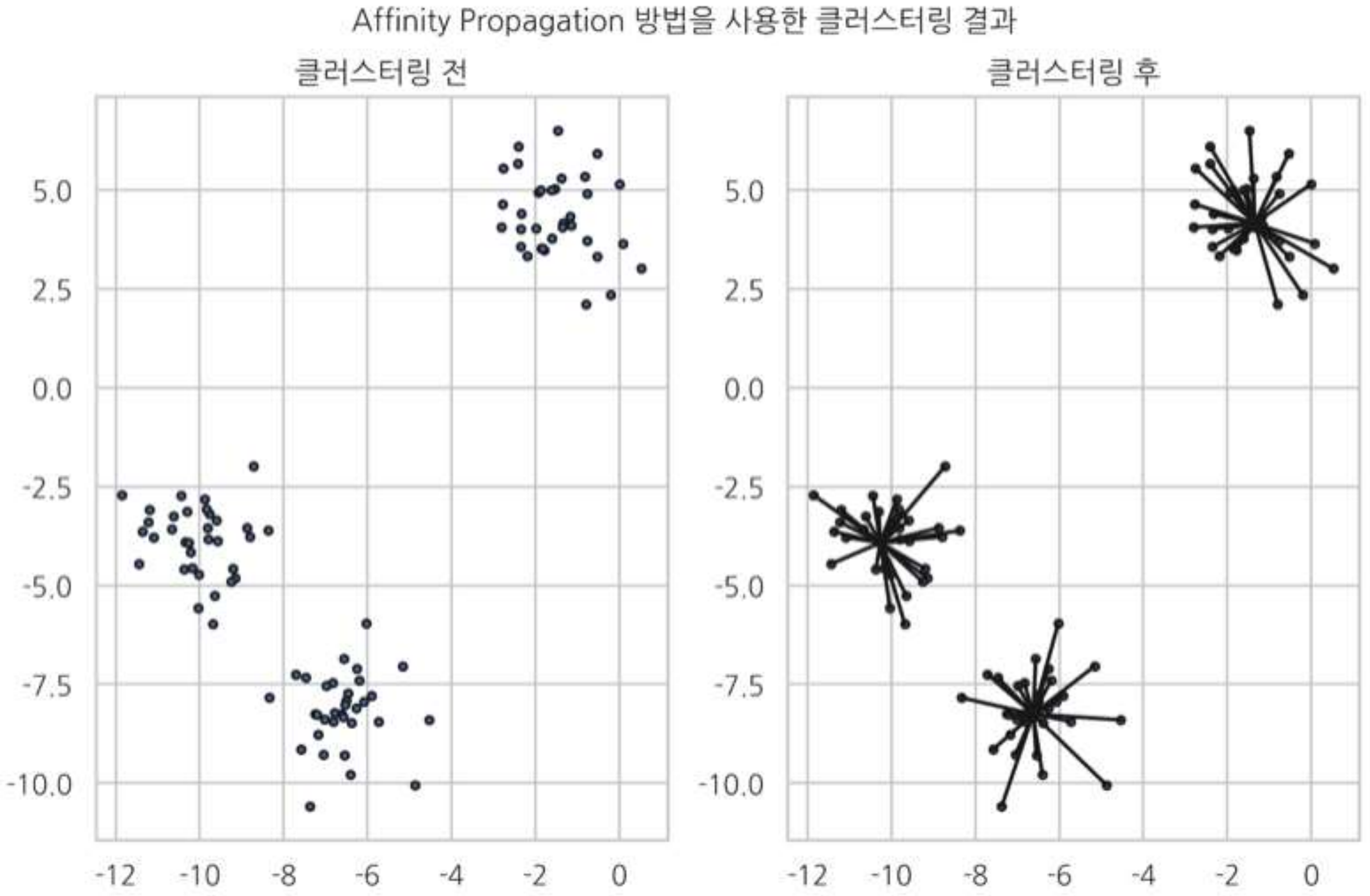


Unsupervised Learning

- *Unsupervised learning is a type of machine learning that looks for previously undetected patterns in a data set with no pre-existing labels and with minimum of human supervision. – Wikipedia*

- Unsupervised learning methods
    - Clustering
    - …

Example) Clustering



1.2 Python Packages and Data

statsmodels package

- One of the development goals of the statsmodels package
    - Allow users who have previously used R to perform statistical and time series analysis to do the same analysis in Python.

- statsmodels package provides
    - Test and estimation
    - Regression analysis
    - Time-series analysis

- Package import method

```
import statsmodels.api as sm
```

Data Provided by statsmodels

- Rdatasets project supports the use of over 1000 standard datasets used by R.
  - Project homepage: https://github.com/vincentarelbundock/Rdatasets
  - List of provided datasets: https://vincentarelbundock.github.io/Rdatasets/datasets.html
  - Usage:

```
get_rdataset(item, [package="datasets"])
```

Example) get_rdataset

- The *Titanic* data in the *datasets* package is for passengers on the Titanic.

```
data = sm.datasets.get_rdataset("Titanic", package="datasets")

df = data.data
df.tail()
```

	Class	Sex	Age	Survived	Freq
27	Crew	Male	Adult	Yes	192
28	1st	Female	Adult	Yes	140
29	2nd	Female	Adult	Yes	80
30	3rd	Female	Adult	Yes	76
31	Crew	Female	Adult	Yes	20

scikit-learn Package

- The scikit-learn package is a Python package for machine learning training.

- Advantages of the scikit-learn package
  - Provides a variety of machine learning models, or algorithms, all in one package.

- Package import method

```
import sklearn as sk
```

Data Provided by scikit-learn

- The `sklearn.datasets` subpackage provides various example datasets.
  - The commands for loading data can be divided into three types of commands.

  - Load series command
    - Command to get the data included in the scikit-learn installation package.
  - Fetch series command
    - Command to fetch data that can be downloaded from the Internet.
  - Make series command
    - Command to generate virtual data at random.

Regression and Time Series Analysis

2. Basis of linear regression analysis

Presenter: Kyuhun Sim

Contents

2.1 Basis of linear regression analysis

Regression anaylsis

- Estimating the relationships between a dependent variable y and one or more independent variables x

$$\hat{y} = f(x) \approx y$$

- if the relationships between x and y is f(x) -> **linear**

Regression anaylsis

$$\hat{y} = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D = w_0 + w^T x$$

- $w0,\cdots,wD$ is the coefficient of f(x), and also parameter

$$f(x) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D = \begin{bmatrix} 1 & x_1 & x_2 & \cdots & x_D \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_D \end{bmatrix} = x_a^T w_a = w_a^T x_a$$

$$x_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{iD} \end{bmatrix} \rightarrow x_{i,a} = \begin{bmatrix} 1 \\ x_{i1} \\ x_{i2} \\ \vdots \\ x_{iD} \end{bmatrix}$$

Regression analysis –bias augmentation

```
from sklearn.datasets import make_regression

X0, y, coef = make_regression(n_samples=100, n_features=2,
                              bias=100, noise=10, coef=True, random_state=1)
```

- Using numpy

```
X = np.hstack([np.ones((X0.shape[0], 1)), X0])
X[:5]
```

```
array([[ 1.        ,  0.0465673 ,  0.80186103],
       [ 1.        , -2.02220122,  0.31563495],
       [ 1.        , -0.38405435, -0.3224172 ],
       [ 1.        , -1.31228341,  0.35054598],
       [ 1.        , -0.88762896, -0.19183555]])
```

- Using statsmodels

```
import statsmodels.api as sm

X = sm.add_constant(X0)
X[:5]
```

```
array([[ 1.        ,  0.0465673 ,  0.80186103],
       [ 1.        , -2.02220122,  0.31563495],
       [ 1.        , -0.38405435, -0.3224172 ],
       [ 1.        , -1.31228341,  0.35054598],
       [ 1.        , -0.88762896, -0.19183555]])
```

OLS(Ordinary Least Squares)

- method to find the weight vector that minimizes RSS as matrix derivative.

$$\hat{y} = Xw$$

- Residual vector e = $e = y - \hat{y} = y - Xw$

$$\begin{aligned} \text{RSS} &= e^T e \\ &= (y - Xw)^T (y - Xw) \\ &= y^T y - 2y^T Xw + w^T X^T Xw \end{aligned}$$

$$\frac{d\text{RSS}}{dw} = -2X^T y + 2X^T Xw$$

OLS(Ordinary Least Squares)

$$\frac{d\text{RSS}}{dw} = 0$$

$$X^T X w^* = X^T y$$

- If inverse of xTx matrix exists $w^* = (X^T X)^{-1} X^T y$

OLS(Ordinary Least Squares)-normal equation

$$\frac{d\mathbf{RSS}}{dw} = 0$$

$$X^T X w^* = X^T y$$

Gradient = 0

$$X^T y - X^T X w = 0$$

$$X^T (y - Xw) = 0$$

$$X^T e = 0$$

Linear regression analysis by using numpy

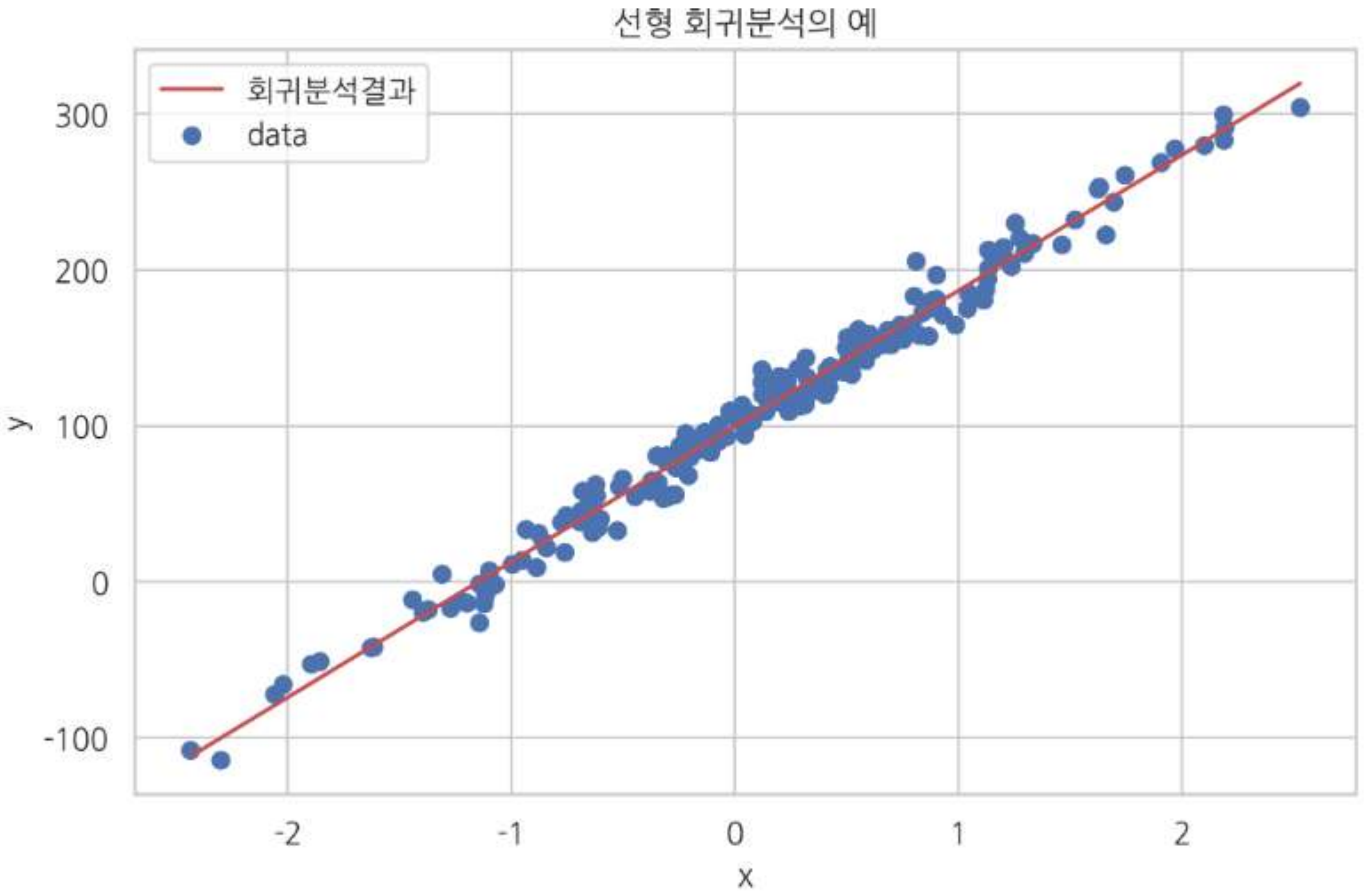
```
from sklearn.datasets import make_regression

bias = 100
X0, y, w = make_regression(
    n_samples=200, n_features=1, bias=bias, noise=10, coef=True, random_state=1
)
X = sm.add_constant(X0)
y = y.reshape(len(y), 1)
```

$\Rightarrow$ $y = 100 + 86.44794301x + \epsilon$

- By using $w^* = (X^T X)^{-1} X^T y$ w = np.linalg.inv(X.T @ X) @ X.T @ y

$\hat{y} = 99.79150869 + 86.96171201x$

Linear regression analysis by using numpy



Linear regression analysis by using scikit-learn

```
model = LinearRegression(fit_intercept=True)
```

- fit_intercept = True-> bias is default, False -> bias does not exist

```
model = model.fit(X, y)
```

- Automatically do bias augmentation -> does not have to use command like add_constant

```
from sklearn.linear_model import LinearRegression

model = LinearRegression().fit(X0, y)
print(model.coef_, model.intercept_)
```

```
[[86.96171201]] [99.79150869]
```

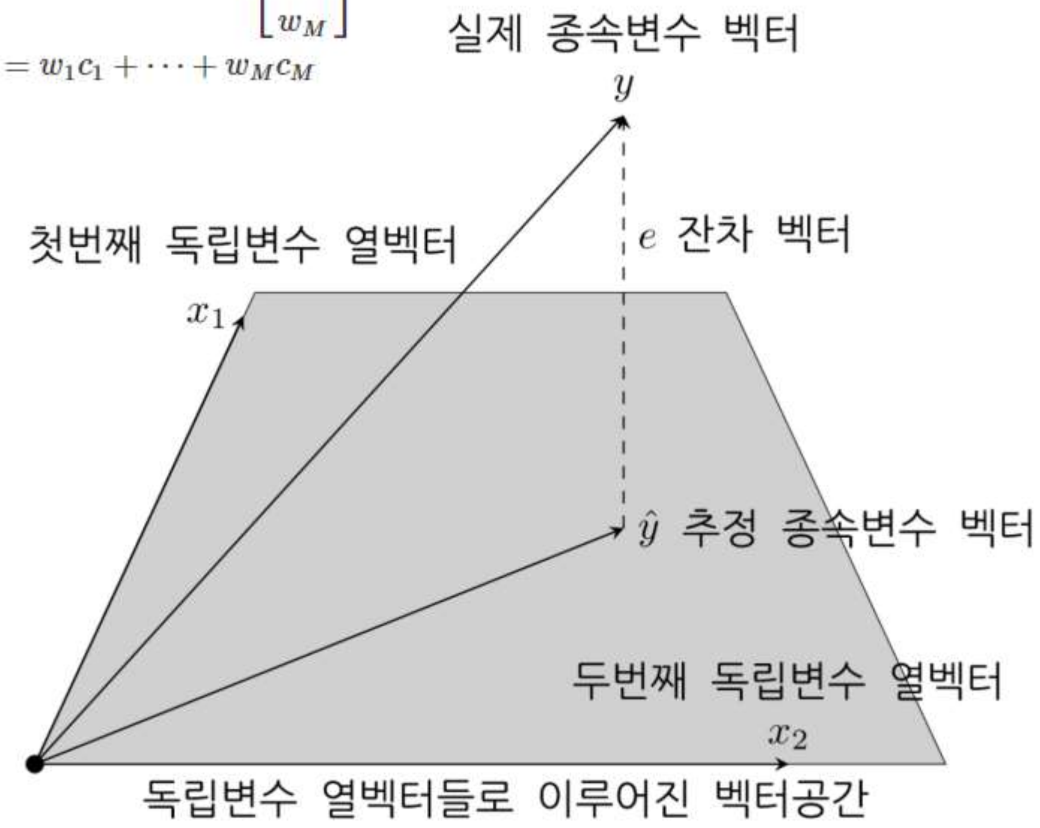
```
model.predict([[3]])
```

```
array([[360.67664473]])
```

2.2 Geometry of regression analysis

Geometry of regression analysis

$$\hat{y} = Xw$$
$$= [c_1 \cdots c_M] \begin{bmatrix} w_1 \\ \vdots \\ w_M \end{bmatrix}$$
$$= w_1 c_1 + \cdots + w_M c_M$$



y^ is a vector projected into a vector space where y is the base vector of each column c1,⋯,cM of X.

The residual vector e is the orthogonal vector remaining after projection.

Geometry of regression analysis

- 1) Dependent vector y -> residual vector e: e=My

$$
\begin{aligned}
e &= y - \hat{y} \\
&= y - Xw \\
&= y - X(X^T X)^{-1} X^T y \\
&= (I - X(X^T X)^{-1} X^T) y \\
&= My
\end{aligned}
$$

- 2) Dependent vector y -> predict vector y^ : y^ = Hy

$$
\begin{aligned}
\hat{y} &= y - e \\
&= y - My \\
&= (I - M) y \\
&= X(X^T X)^{-1} X^T y \\
&= Hy
\end{aligned}
$$

Geometry of regression analysis

- Property of M, H
  - Symmetric matrix $M^T = M$

    $H^T = H$
  - Idempotent matrix $M^2 = M$

    $H^2 = H$
  - M and H is orthogonal $MH = HM = 0$
  - M and X is orthogonal $MX = 0$
  - H times X equal to X $HX = X$

2.3 Partial Regression

Partial Regression

Regression analysis

- Just use one independent variable x1

$$y = w_1 x_1 + e$$

After add new independent variable

- w1’ ≠ w1

$$y = w_1' x_1 + w_2' x_2 + e'$$

Weight of the model is always biased unless we include all independent variables that affect the dependent variable in the regression model.

Partial Regression

- Divide independent variables to two groups $X = [\,X_1 \quad X_2\,]$
- Regression analysis using only the independent variable X1 $w_1 = (X_1^T X_1)^{-1} X_1^T y$

- New model add new independent variables $y = \hat{y} + e' = [X_1 \quad X_2] \begin{bmatrix} w'_1 \\ w'_2 \end{bmatrix} + e'$

- By using normal equation $\begin{bmatrix} X_1^T X_1 & X_1^T X_2 \\ X_2^T X_1 & X_2^T X_2 \end{bmatrix} \begin{bmatrix} w'_1 \\ w'_2 \end{bmatrix} = \begin{bmatrix} X_1^T y \\ X_2^T y \end{bmatrix}$

$$w'_1 = w_1 - (X_1^T X_1)^{-1} X_1^T X_2 w'_2$$

Partial Regression – FWL theorem

- y* = Linear regression analysis of dependent variable y with independent variable group X1

- x2* = Linear regression analysis of other independent variable x2 with X1

$$x_2^{*T} x_2^* w_2 = x_2^{*T} y^*$$

Partial Regression – remove mean

- By FWL Theorem

- Regression analysis removed mean from independent variables and removed mean from dependent variables

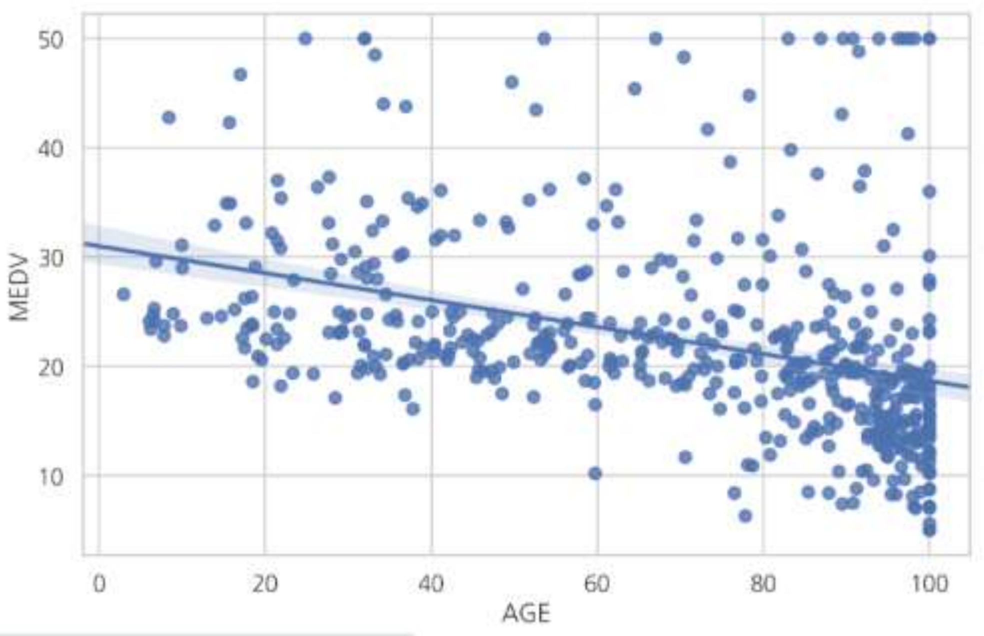
$$=$$

- Regression analysis that included constant

Partial Regression Plot

- When number of independent variables is large
  - To visualize effects of specific independent variables

```
sns.regplot(x="AGE", y="MEDV", data=df)
plt.show()
```



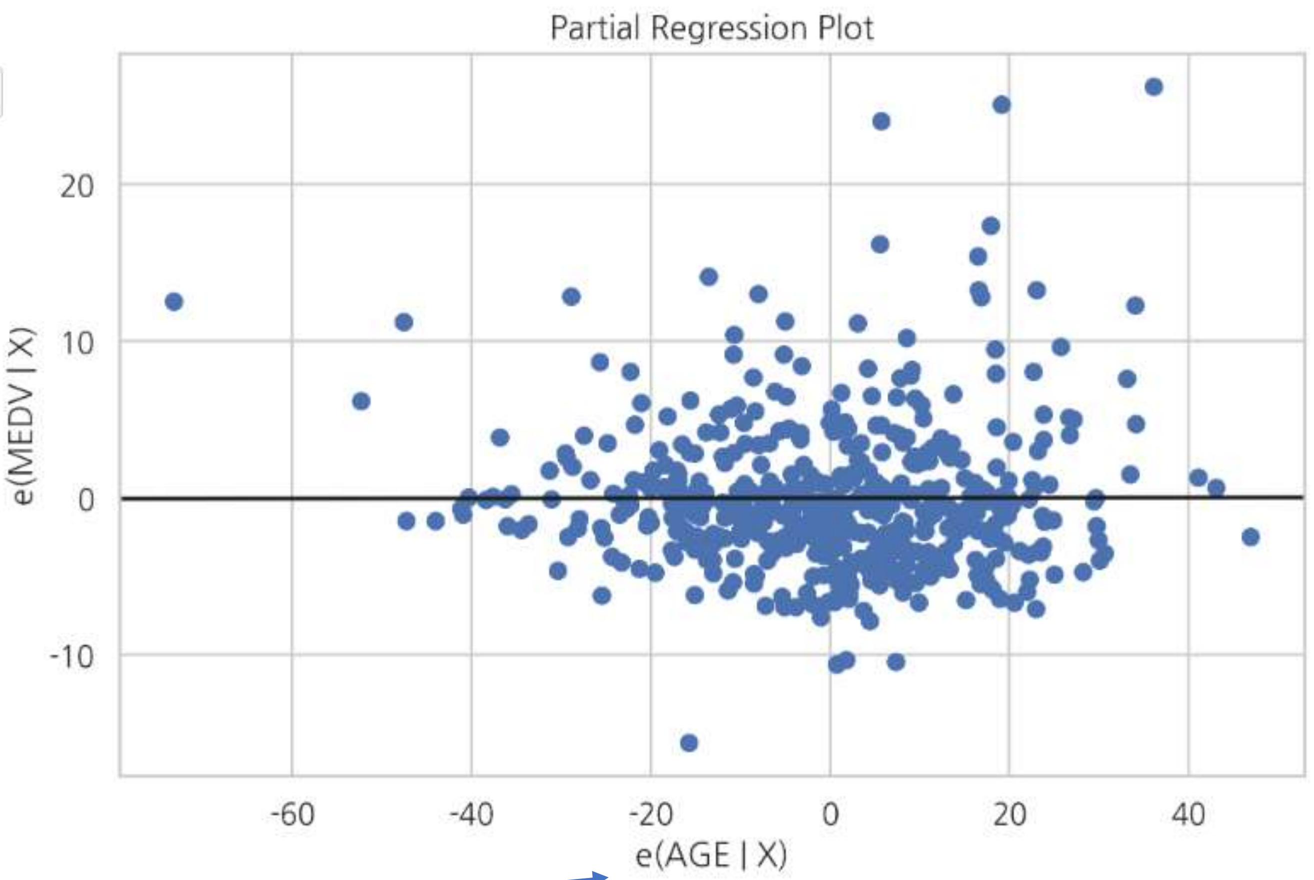
AGE independent variable and the MEDV dependent variable seems to have a negative correlation.

Partial regression Plot

```
plot_partregress(endog, exog_i, exog_others, data=None, obs_labels=True, ret_coords=False)
```

- `endog` : 종속변수 문자열
- `exog_i` : 분석 대상이 되는 독립변수 문자열
- `exog_others` : 나머지 독립변수 문자열의 리스트
- `data` : 모든 데이터가 있는 데이터프레임
- `obs_labels` : 데이터 라벨링 여부
- `ret_coords` : 잔차 데이터 반환 여부

```
others = list(set(df.columns).difference(set(["MEDV", "AGE"])))
p, resids = sm.graphics.plot_partregress(
    "MEDV", "AGE", others, data=df, obs_labels=False, ret_coords=True
)
plt.show()
```



Remove effects of other independent variables

CCPR Plot

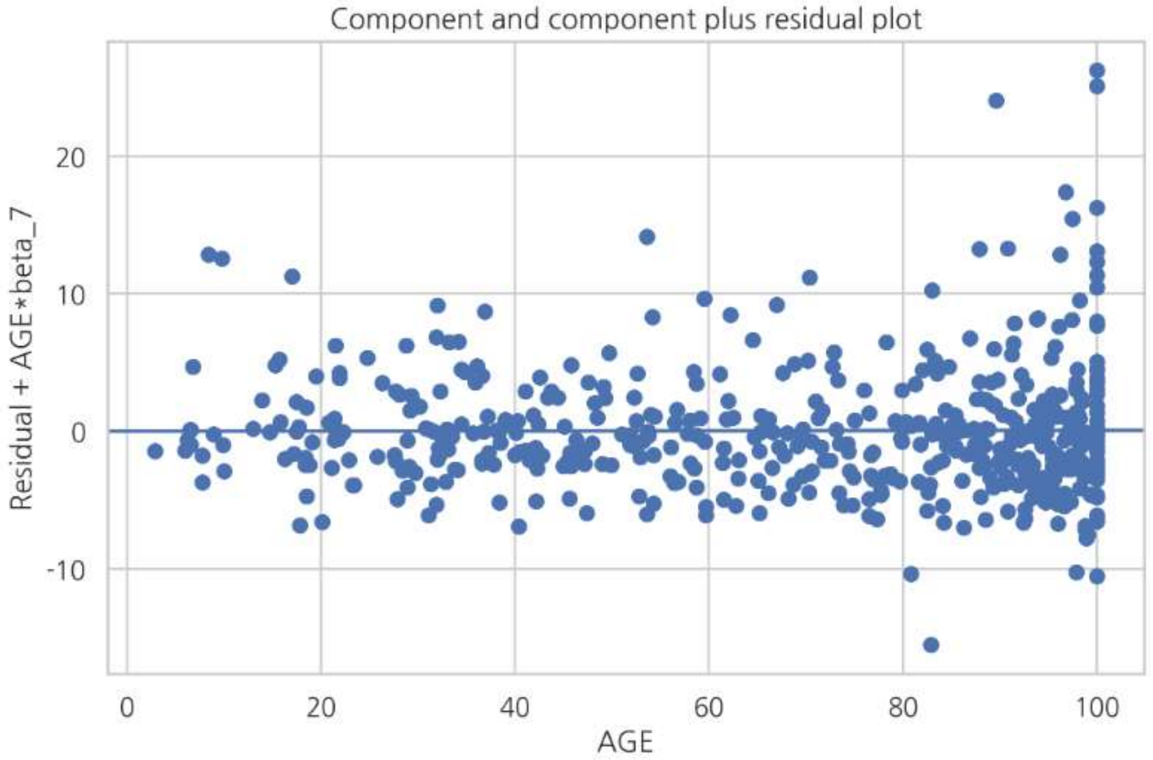
- Unlike the partial regression plot, the independent variable appears as it is.

$$y = \hat{y} + e = w_1 x_1 + w_2 x_2 + \cdots + w_i x_i + \cdots + w_K x_K + e$$

- xi as horizontal axis and wixi+e as vertical axis

CCPR Plot

```
sm.graphics.plot_ccpr(result_boston, "AGE")
plt.show()
```




```
fig = sm.graphics.plot_regress_exog(result_boston, "AGE")
plt.tight_layout(pad=4, h_pad=0.5, w_pad=0.5)
plt.show()
```