2장. 회귀분석

2장. 회귀분석

1절. 회귀분석 개요

회귀분석(回歸分析, regression analysis)

관찰된 연속형 변수들에 대해 두 변수 사이의 모형을 구한 뒤 적합도를 측정해 내는 분석

회귀분석(回歸分析, regression analysis)

- 회귀분석은 선형적인 상관성을 가진 변수들 사이의 인과관계를 증명하는 것
- 원인이 되는 독립변수, 결과가 되는 종속변수
- 회귀분석은 시간에 따라 변화하는 데이터나 어떤 영향, 가설적 실험, 인과 관계의 모델링 등의 통계적 예측에 이용될 수 있음
- 어떤 연관성을 가지고 있는 종속변수의 변동이나 분산을 설명하기 위하여 종속변수와 관계가 있는 독립변수들 중 각각의 독립변수가 설명력을 얼마나 가지고 있는가를 결정할 때 사용

회귀분석 전제 사항

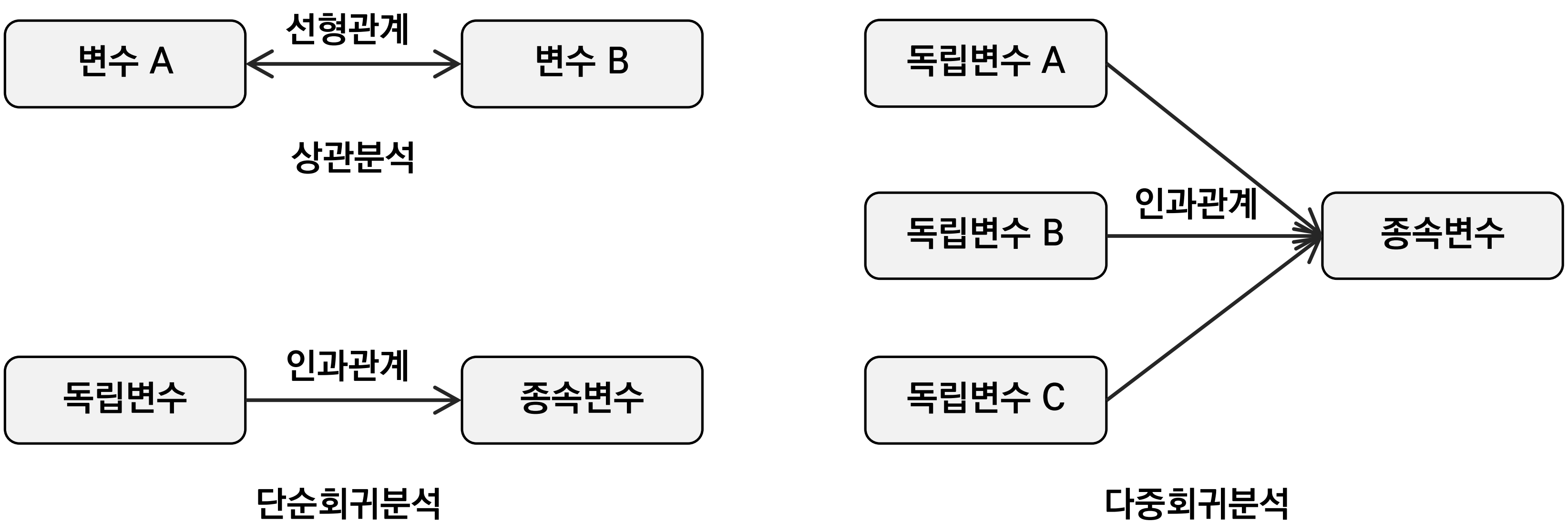
- 선형성 : 독립 변수의 변화에 따라 종속 변수도 일정 크기로 변함
- 독립성 : 오차와 독립 변수의 값이 관련이 없음
- 등분산성 : 독립 변수의 모든 값에 대해 오차들의 분산이 일정
- 비상관성 : 관측치의 오차들 사이에 상관관계가 없음
- 정상성 : 오차가 정규 분포를 따름

상관분석, 단순회귀분석, 다중회귀분석

상관분석 두 변수 사이의 원인과 결과가 아닌 서로 상관(相關)적 영향이 있는지를 분석하는 것

회귀분석 인과관계(因果關係)로서 독립변수가 종속변수에 얼마만큼 영향을 주는지를 분석하는 것

- 단순회귀분석 : 독립변수가 1개이며 종속변수도 1개인 것
- 다중회귀분석 : 만일 독립변수가 2개 이상인 경우의 회귀분석

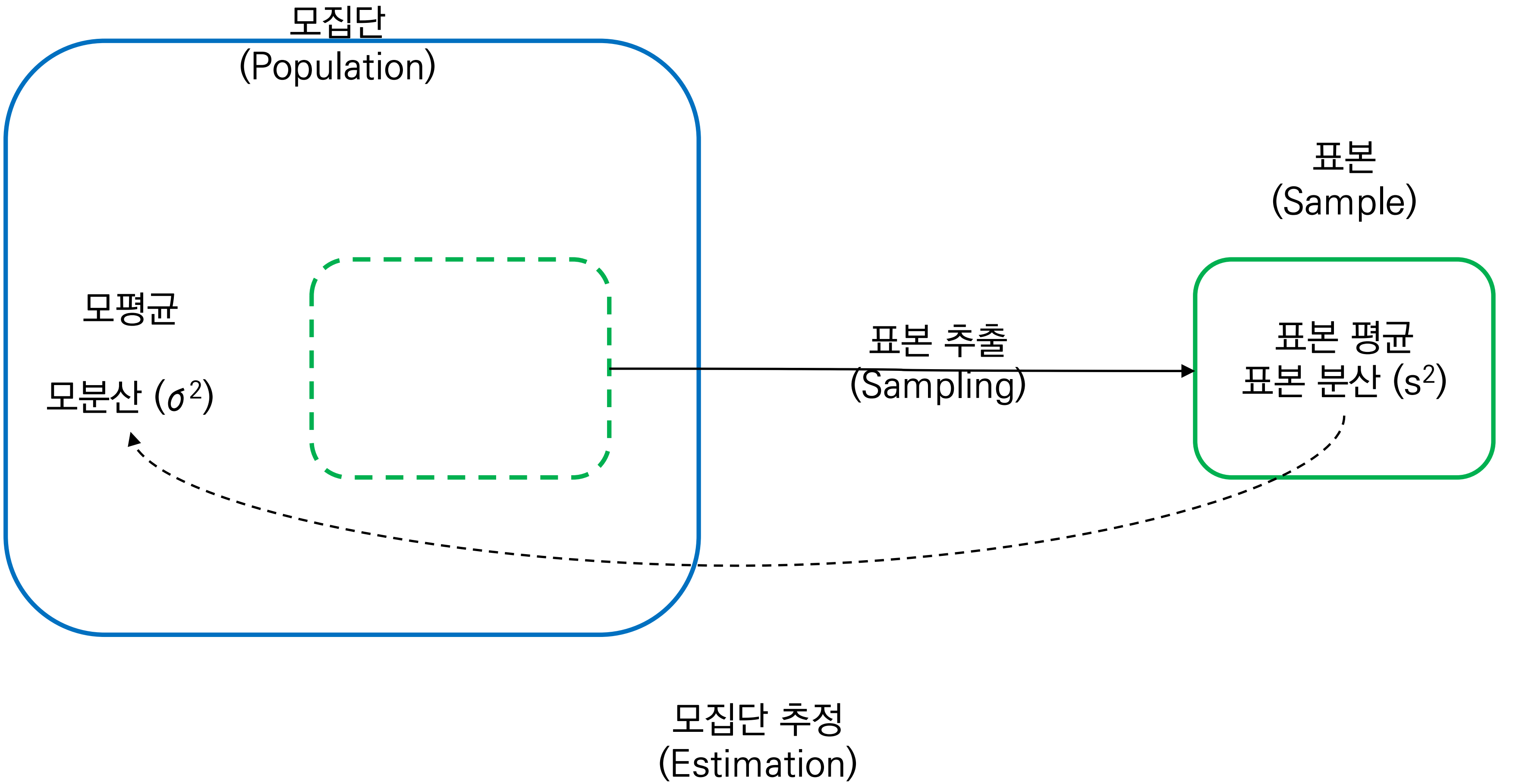


2장. 회귀분석

2절. 분포와 추론

모집단과 표본

- 특정한 집단이나 불확실한 현상을 대상으로 자료를 수집해 대상 집단에 대한 정보를 구하고, 적절한 통계분석 방법을 이용해 의사결정을 하는 과정



실험 (Experiment)

- 특정 목적 하에서 실험 대상에게 처리를 가한 후에 그 결과를 관측해 자료를 수집하는 방법
- 측정 (Measurement) : 추출된 원소들이나 실험 단위로부터 주어진 목적에 적합하도록 관측해 자료를 얻는 것

표본공간 (Sample space, Ω)	▪ 어떤 실험할 때 나타날 수 있는 모든 결과들의 집합

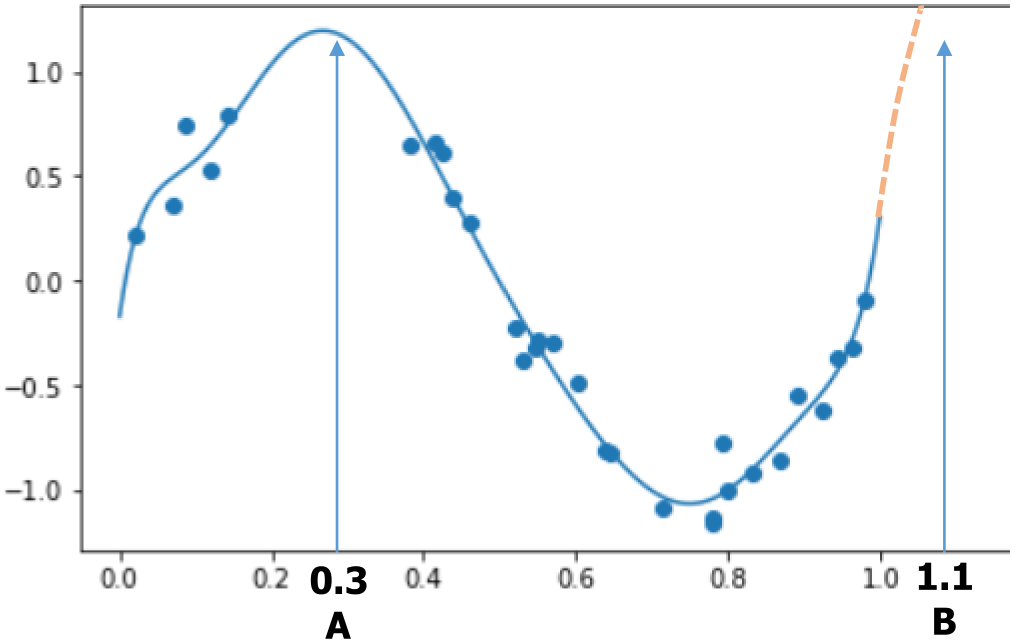
사건 (Event, E)	▪ 표본공간에 있는 몇 개의 원소들로 이루어진 부분 집합
	▪ 확률 (Probability)
	  ▪ 특정 사건이 일어날 가능성의 척도
	  ▪ $P(E) = n(E) / n(\Omega)$
	▪ 확률 변수 (Random variable)
	  ▪ 특정 사건에 대해 실수값을 갖는 변수(x)를 정의하면, 특정 사건이 일어날 확률은 변수가 특정 값을 가질 확률(f(x))로 표현할 수 있다.
	  ▪ x : 특정 값이 나타날 가능성이 확률적으로 주어지는 변수
	  ▪ f(x) : 확률분포함수 (Probability distribution Function), 정의역(Domain)이 표본공간이고 치역(Range)이 실수값인 함수

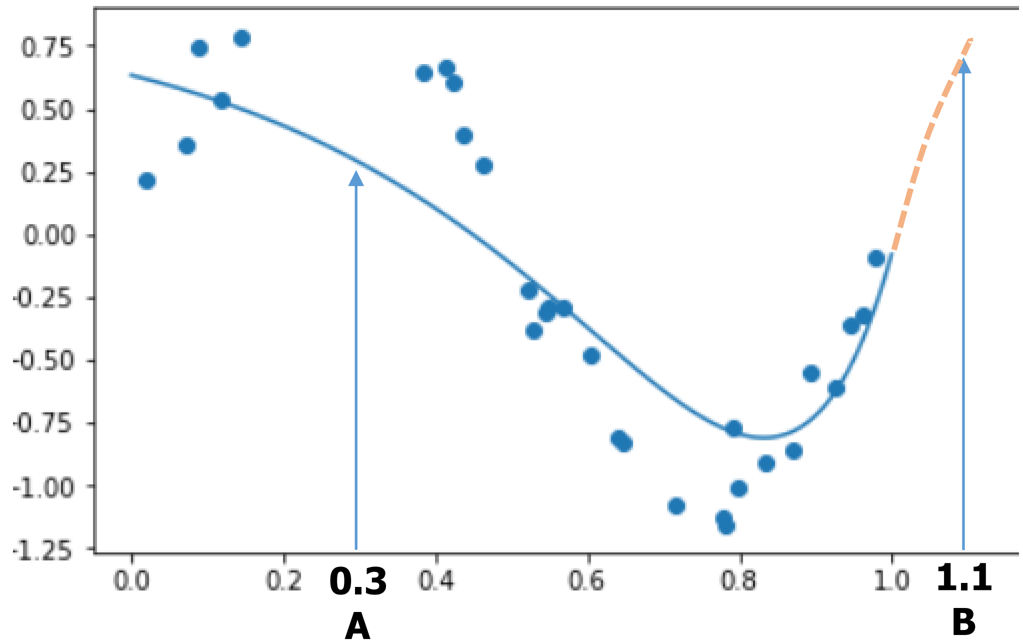
numpy.random 모듈

- 이산 확률 분포
  - 이항 분포(Binomial distribution) : binomial(n, p[, size]) – n trials and p probability
  - 베르누이 분포(Bernoulli distribution) : binomial(n=1, p[, size])
  - 기하 분포(Geometric distribution) : geometric(p[, size])
  - 다항 분포(Multinomial distribution) : multinomial(n, pvals[, size])
  - 포아송 분포(Poisson distribution) : poisson([lam, size])
- 연속 확률 분포
  - 균일분포(Uniform distribution) : uniform([low, high, size])
  - 정규분포(Normal distribution) : normal([loc, scale, size])
  - 지수분포(Exponential distribution) : exponential([scale, size])
  - t-분포(t-distribution) : standard_t(df[, size])
  - 카이제곱분포(chi-squared distribution) : chisquare(df[, size])
  - F-분포(F-distribution) : f(dfnum, dfden[, size])
- 참고 : numpy.random.Generator
  - https://numpy.org/doc/1.18/reference/random/generator.html

- 통계적 추론(Statistical inference)은 수집된 자료를 이용해 대상 집단(모집단)에 대해 의사결정을 하는 것을 의미
- 추정(推定, Estimation)은 입력된 자료가 불완전하거나 불확실하더라도 사용할 수 있는 계산된 결과의 근삿값
  - 점추정(Point estimation) : ‘모수가 특정한 값일 것’이라고 추정하는 것
  - 구간추정(Interval estimation) : ‘확률로 표현된 신뢰도 하에서 모수가 특정한 구간에 있을 것’이라고 선언하는 것

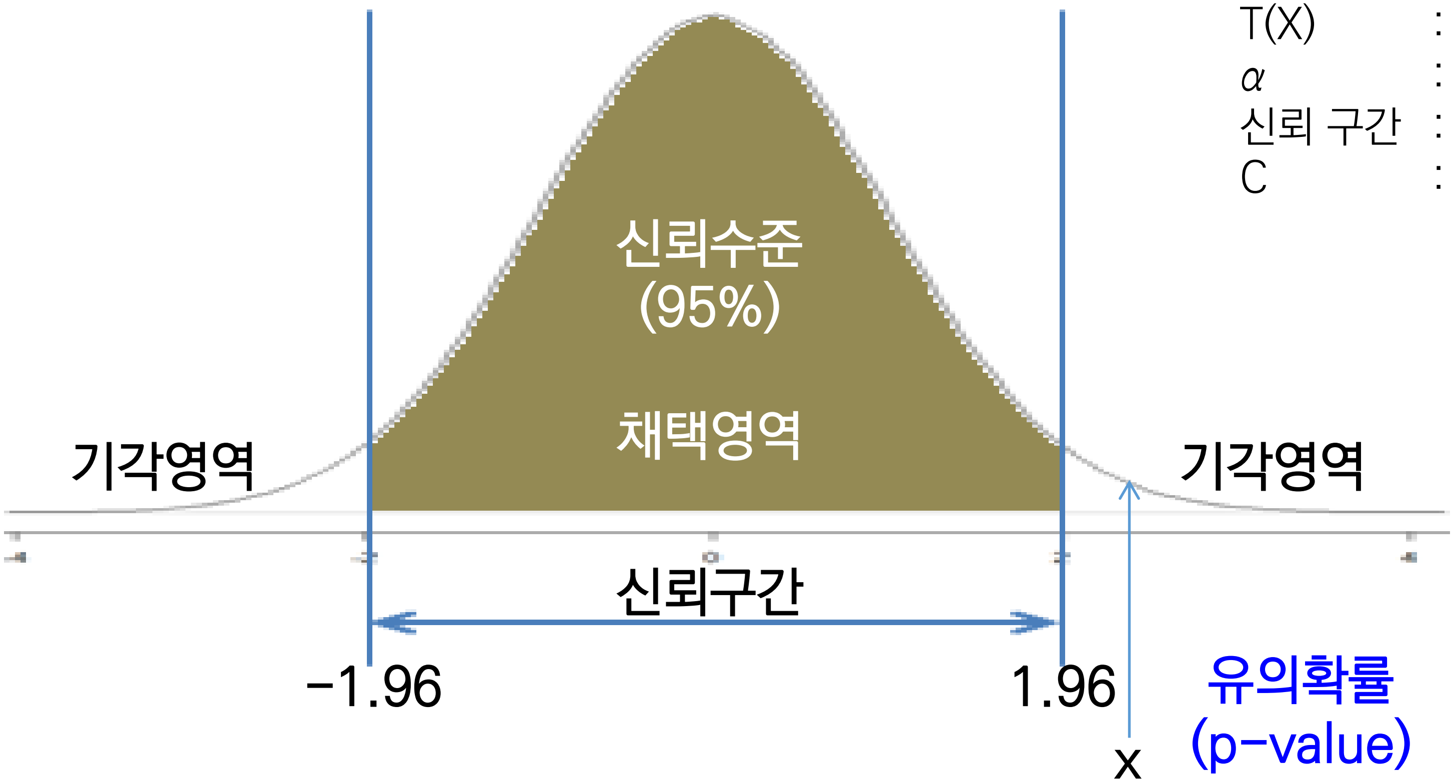
Prediction(A)과 Forecast(B)





• 수집된 자료를 이용해 대상 집단(모집단)에 대해 의사결정

H0	: 모평균은 0
H1	: 모평균은 0이 아님
T(X)	: 표본 평균 (x)
α	: 5%
신뢰 구간	: −1.96 ~ 1.96
C	: −1.96 미만, 1.96 이상



통계적 추론 (Statistical inference)

- 수집된 자료를 이용해 대상 집단(모집단)에 대해 의사결정을 하는 것을 의미
- 통계적 추론을 할 때에 추정(Estimation)과 가설검정(Hypothesis test)이라는 말을 많이 사용 함

추정(Estimation)

- 점추정(Point estimation), 구간추정(Interval estimation)
- 점추정 : '모수가 특정한 값일 것'이라고 추정하는 것
- 구간추정 : '확률로 표현된 신뢰도하에서 모수가 특정한 구간에 있을 것'이라고 선언하는 것
- 구간추정은 항상 분포에 대한 전제가 주어져야 하고, 구해진 구간 안에 모수가 있을 가능성이 주어져야 함

가설검정(Hypothesis test)

모집단의 모수에 대한 어떤 가설을 설정한 뒤에 표본 관찰을 통해 그 가설의 채택 여부를 결정하는 분석 방법

귀무가설(Null hypothesis, H0)과 대립가설(Alternative hypothesis, H1)

- 귀무가설(Null hypothesis, H0) : 검정하고자 하는 모수에 대한 가설이며, 버릴 것을 예상하는 가설
- 대립가설 (Alternative hypothesis, H1) : 연구자가 연구를 통해 입증되기를 기대하는 예상이나 주장을 대립가설로 내세움
- 검정에 사용되는 통계량을 검정통계량(Test statistic, T(X))이라고 하는데, 귀무가설이 옳다는 전제 하에서 검정통계량 값을 구한 후에 이 값이 나타날 가능성의 크기에 의해 귀무가설 채택 여부를 결정함

유의수준과 유의확률

유의수준(Significance level, α)

- 검정 통계량의 값이 나타날 가능성이 “크다” 또는 “작다”의 판단 기준
- 귀무가설을 기각하게 되는 확률의 크기로 “귀무가설이 옳은데도 이를 기각하는 확률의 크기”로 정의

기각역(Critical region, C)

- 귀무가설이 옳다는 전제하에서 구한 검정통계량의 분포에서 확률이 유의 수준 α인 부분을 의미

가설 검정의 오류

- 제1종 오류(Type I error)는 옳은 귀무가설을 기각하는 오류
- 제2종 오류(Type II error)는 옳지 않은 귀무가설을 채택하는 오류

유의확률(p-value, Significance probability)

- 제1종 오류를 발생할 확률을 의미
- 신뢰수준이 95%일 경우 유의 확률(p-value)이 유의수준 (0.05)보다 작으면 귀무가설을 기각 대립가설을 채택하는 것을 의미

모수적 방법과 비모수적 방법

통계적 추론(Statistical inference)에서 모집단의 모수에 대한 검정 방법

모수적 방법(Parametric method)

- 검정하고자 하는 모집단의 분포에 대한 가정이며, 그 가정하에서 검정통계량과 검정통계량의 분포를 유도해 검정을 실시
- 관측된 자료를 이용해 구한 표본평균, 표본분산 등을 이용해 검정을 실시

비모수적 방법(Non-parametric method)

- 관측된 데이터가 특정 분포를 가진다고 가정할 수 없는 경우 사용
- 모집단 분포에 대한 가정을 하지 않고 검정을 실시
- 부호검정(Sign test), 윌콕슨의 순위합검정(Rank sum test), 윌콕슨의 부호순위합검정(Wilcoxon signed rank test), 만-위트니의 U검정, 런검정(Run test), 스피어만의 순위상관계수 등
  관측된 자료를 이용해 구한 표본평균, 표본분산 등을 이용해 검정을 실시
- 관측값의 절대적인 크기에 의존하지 않는 관측값들의 순위(rank) 또는 두 관측값 차이의 부호(sign) 등을 이용해 검증

2장. 회귀분석

3절. 상관분석

두 변수 간에 선형적 관계가 있는지 분석하는 것

피어슨 상관계수(Pearson correlation coefficient)

- 수치로 표시된 데이터간의 상관관계를 확인하기 위해 사용
- 두 변수가 모두 연속형 자료일 때
- 두 변수간 선형적인 상관관계의 크기를 모수적(parametric)인 방법으로 나타낸 값

$$\rho_{X,Y} = cov(X,Y) / \sigma_X * \sigma_Y$$

범위	관계
−1.0 ≤ r ≤ −0.7	매우 강한 음(−) 의 상관관계
−0.7 〈 r ≤ −0.3	강한 음(−) 의 상관관계
−0.3 〈 r ≤ −0.1	약한 음(−) 의 상관관계
−0.1 〈 r ≤ 0.1	상관관계 없음
0.1 〈 r ≤ 0.3	약한 양(+) 의 상관관계
0.3 〈 r ≤ 0.7	강한 양(+) 의 상관관계
0.7 〈 r ≤ 1.0	매우 강한 양(+) 의 상관관계

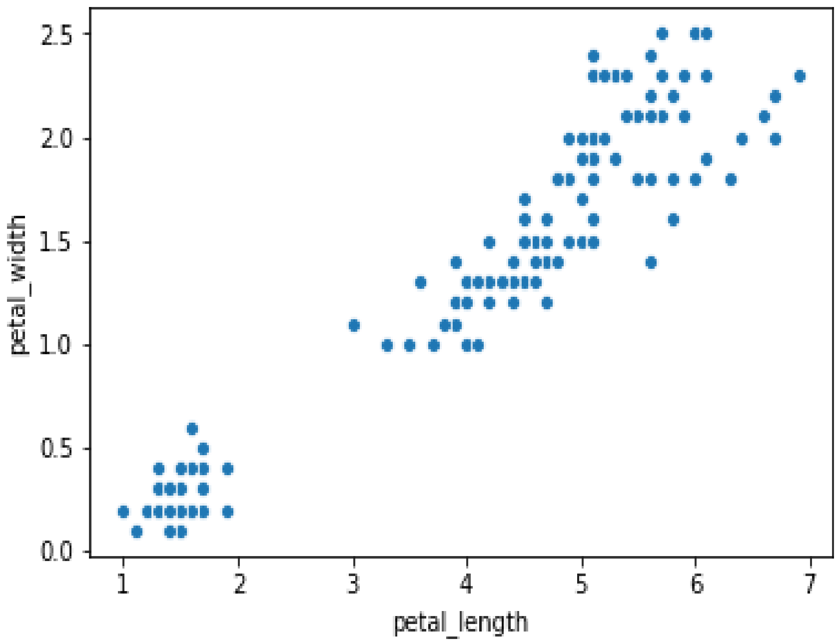
iris 데이터의 상관계수 확인


```
import seaborn as sns
iris = sns.load_dataset("iris")
iris.corr(method='pearson')
```

	sepal_length	sepal_width	petal_length	petal_width
sepal_length	1.000000	-0.117570	0.871754	0.817941
sepal_width	-0.117570	1.000000	-0.428440	-0.366126
petal_length	0.871754	-0.428440	1.000000	0.962865
petal_width	0.817941	-0.366126	0.962865	1.000000

```
from scipy.stats.stats import pearsonr
pearsonr(iris.petal_length, iris.petal_width)

(0.9628654314027961, 4.675003907327543e-86)
```

```
sns.scatterplot(iris.petal_length, iris.petal_width)
```



스피어만 상관계수

스피어만 상관계수(Spearman correlation coefficient)

- 분석하고자 하는 두 연속형 변수의 분포가 심각하게 정규 분포(normal distribution)를 벗어난다거나 또는 두 변수가 순위 척도(ordinal scale) 자료일 때 사용
- 데이터가 서열 척도인 경우, 값 대신 순위를 사용하여 계산한 상관계수

```
import seaborn as sns
iris = sns.load_dataset("iris")
iris.corr(method='spearman')
```

	sepal_length	sepal_width	petal_length	petal_width
sepal_length	1.000000	-0.166778	0.881898	0.834289
sepal_width	-0.166778	1.000000	-0.309635	-0.289032
petal_length	0.881898	-0.309635	1.000000	0.937667
petal_width	0.834289	-0.289032	0.937667	1.000000

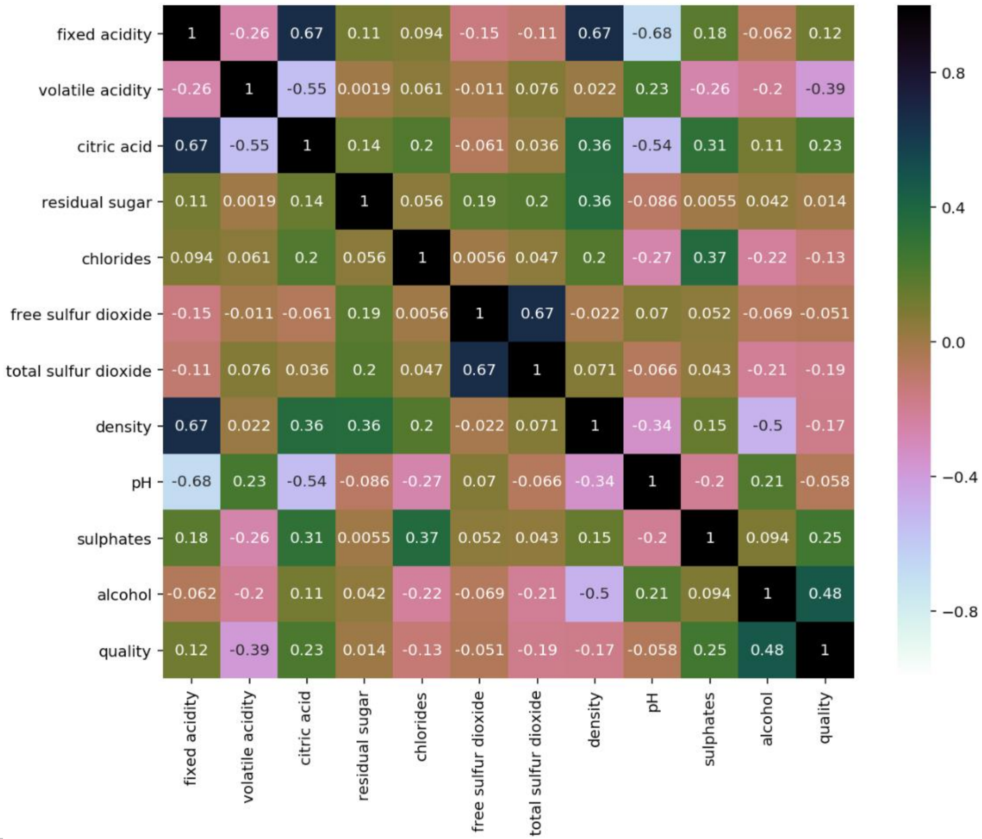
```
from scipy.stats.stats import spearmanr
spearmanr(iris.petal_length, iris.petal_width)

SpearmanrResult(correlation=0.9376668235763412, pvalue=8.156596854126675e-70)
```

상관계수 시각화

redwine 데이터를 이용해서 상관계수 데이터프레임을 만들고 시각화

```
plt.figure(figsize=(10,8))
sns.heatmap(redwine.corr(), vmin=-1, vmax=1, annot=True, cmap="cubehelix_r")
plt.show()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
fixed acidity	1.000000	-0.256131	0.671703	0.114777	0.093705	-0.153794	-0.113181	0.668047	-0.682978	0.183006	-0.061668	0.124052
volatile acidity	-0.256131	1.000000	-0.552496	0.001918	0.061298	-0.010504	0.076470	0.022026	0.234937	-0.260987	-0.202288	-0.390558
citric acid	0.671703	-0.552496	1.000000	0.143577	0.203823	-0.060978	0.035533	0.364947	-0.541904	0.312770	0.109903	0.226373
residual sugar	0.114777	0.001918	0.143577	1.000000	0.055610	0.187049	0.203028	0.355283	-0.085652	0.005527	0.042075	0.013732
chlorides	0.093705	0.061298	0.203823	0.055610	1.000000	0.005562	0.047400	0.200632	-0.265026	0.371260	-0.221141	-0.128907
free sulfur dioxide	-0.153794	-0.010504	-0.060978	0.187049	0.005562	1.000000	0.667666	-0.021946	0.070377	0.051658	-0.069408	-0.050656
total sulfur dioxide	-0.113181	0.076470	0.035533	0.203028	0.047400	0.667666	1.000000	0.071269	-0.066495	0.042947	-0.205654	-0.185100
density	0.668047	0.022026	0.364947	0.355283	0.200632	-0.021946	0.071269	1.000000	-0.341699	0.148506	-0.496180	-0.174919
pH	-0.682978	0.234937	-0.541904	-0.085652	-0.265026	0.070377	-0.066495	-0.341699	1.000000	-0.196648	0.205633	-0.057731
sulphates	0.183006	-0.260987	0.312770	0.005527	0.371260	0.051658	0.042947	0.148506	-0.196648	1.000000	0.093595	0.251397
alcohol	-0.061668	-0.202288	0.109903	0.042075	-0.221141	-0.069408	-0.205654	-0.496180	0.205633	0.093595	1.000000	0.476166
quality	0.124052	-0.390558	0.226373	0.013732	-0.128907	-0.050656	-0.185100	-0.174919	-0.057731	0.251397	0.476166	1.000000



2장. 회귀분석

4절. 단순 회귀분석

독립변수가 하나일 경우의 회귀분석

$$y = \boldsymbol{a} * \boldsymbol{x} + \boldsymbol{b}$$

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \\ \ldots & \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= b_n \end{aligned}$$

x에 대한 선형방정식을 행렬로 표현

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}$$

$$AX = B$$

양쪽 항에 A의 역행렬을 곱함

$$X = A^{-1}B$$

A행렬의 행의 수가 열의 수보다 클 경우

$$X = (A^TA)^{-1}A^TB$$

선형방정식으로 회귀모형 구하기

4절. 단순 회귀분석

```
X = [32, 64, 96, 118, 126, 144, 152, 158]
Y = [17, 24, 62, 49, 52, 105, 130, 125]

import numpy as np
A = np.column_stack((X, np.ones((8))))
B = np.array(Y)
```

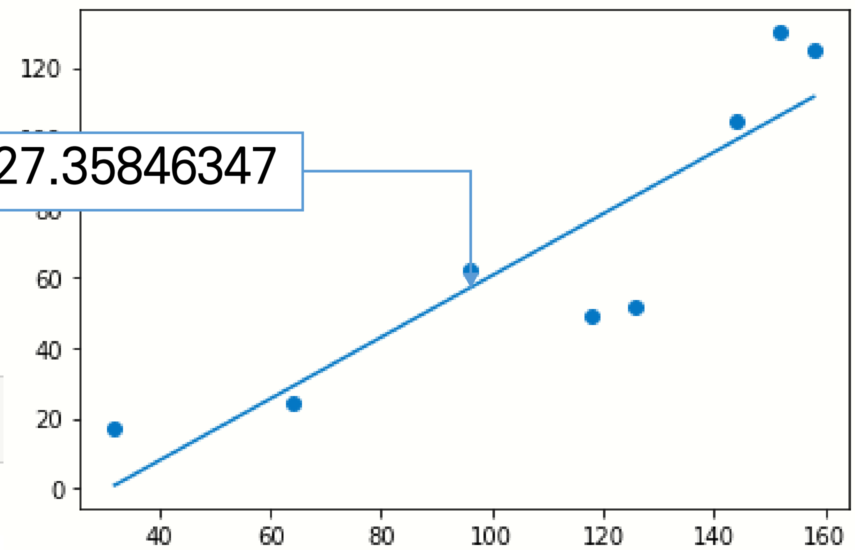
```
A
```

```
array([[ 32.,   1.],
       [ 64.,   1.],
       [ 96.,   1.],
       [118.,   1.],
       [126.,   1.],
       [144.,   1.],
       [152.,   1.],
       [158.,   1.]])
```

```
np.linalg.inv(A.T @ A) @ A.T @ B
```

```
array([  0.87962664, -27.35846347])
```

y = 0.87952664*x – 27.35846347



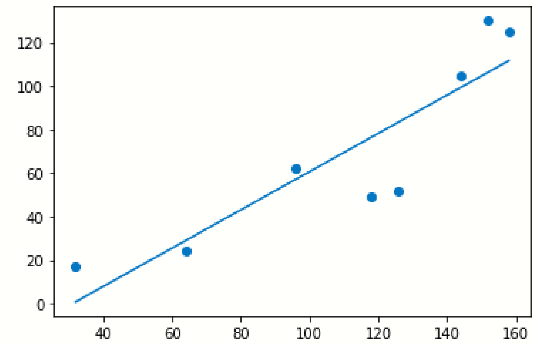
scipy.stats.linregress(*x*, *y=None*)

linregress()함수는 선형 최소 제곱 회귀식을 계산

```
X = [32, 64, 96, 118, 126, 144, 152, 158]
Y = [17, 24, 62, 49, 52, 105, 130, 125]

from scipy import stats
slope, intercept, r_value, p_value, std_err = stats.linregress(X, Y)
print(f"slope: {slope}")
print(f"intercept: {intercept}")
print(f"r_value: {r_value}")
print(f"p_value: {p_value}")
print(f"std_err: {std_err}")
```

```
slope: 0.8796266379465087
intercept: -27.3584634715491
r_value: 0.89008928103186
p_value: 0.003051790677096642
std_err: 0.18388671751663876
```

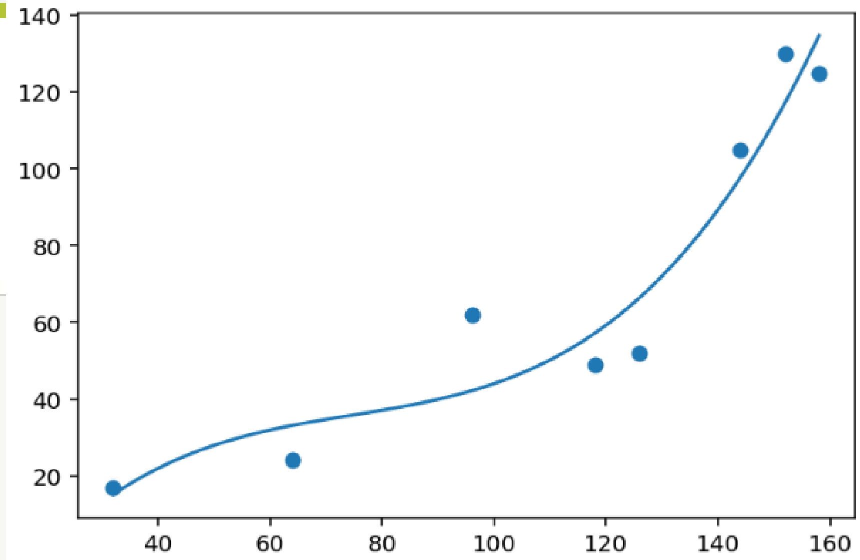
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

plt.scatter(X, Y)
plt.plot(X, slope*np.array(X) + intercept, '-')
plt.show()
```



- y = 0.8796*x - 27.3585
- p-value(유의 확률, Significance probability) 0.003051790677096642가 0.05 보다 작으므로 유의 수준 5%하에서 회귀식이 유의함.
- r_value(결정 계수, R2) 0.89008928103186은 회귀식이 데이터를 약 89% 설명
- std_err : 표준 오차

numpy.polyfit(*x*, *y*, *deg*)

polyfit() 함수는 최소제곱 다항 회귀식을 계산




```
def draw_polyfit(X, Y, deg=1):
    import numpy as np
    fit = np.polyfit(X, Y, deg)
    print(fit)
    fit_fn = np.poly1d(fit)
    sample_X = np.linspace(min(X), max(X), 100)
    plt.scatter(X, Y)
    plt.plot(sample_X, fit_fn(sample_X))
    plt.show()
```

```
draw_polyfit(X, Y, 3)
```

```
[ 1.36900759e-04 -3.06581641e-02  2.52476185e+00 -3.87896220e+01]
```

2장. 회귀분석

5절. 포뮬러를 이용한 회귀식

선형회귀식을 구하기 위해 formula(포뮬러) 구문을 사용하여 통계 모형의 형식을 지정

ols() 함수 또는 OLS클래스의 from_formula()를 사용하면 포뮬러를 이용하여 다항 회귀식을 구할 수 있음

```
statsmodels.formula.api.ols(formula, data, subset=None,
          drop_cols=None, *args, **kwargs)
```

```
statsmodels.regression.linear_model.OLS.from_formula(
          formula, data, subset=None, drop_cols=None,
          *args, **kwargs)
```

구문에서...

– *formula* : 포뮬러, str 형식, 포뮬러의 기본 형식은 다음과 같습니다.

응답 변수 ~ 예측 변수

기호	의미	예
+	이 변수를 포함합니다.	+Z
–	이 변수를 제외합니다.	–Z
:	이 변수들 사이의 상호 작용(interaction)을 포함합니다.	X:Z
*	이 변수들과 그것들을 조합한 모든 상호작용들을 포함합니다.	X*Z
^	예는 모든 상호작용을 최대 세 가지 방법으로 포함합니다.	(X + Z + W)^3
I	수식으로 구성된 새 변수를 추가합니다.	I(expr)
–1	절편(intercept)을 삭제합니다. +0과 같습니다.	X – 1
.	데이터에서 종속변수를 제외한 모든 변수를 독립변수로 사용합니다. .(점)은 R에서는 사용할 수 있지만 파이썬에서는 사용할 수 없습니다.	Y ~ .

Y ~ (X:Z)는 y = a+bxz 형식의 방정식 생성
Y ~ X*Z는 Y ~ X + Z + X:Z와 동일
Y ~ I(X^2) + I(X)는 y = a+bx^2 + cx

Y ~ X + Z + W + X:Z + X:W + Z:W + X:Z:W
Y ~ X * Z * W
Y ~ (X + Z + W)^3

포뮬러를 이용한 1차 방정식 구하기

```
X = [32, 64, 96, 118, 126, 144, 152, 158]
Y = [17, 24, 62, 49, 52, 105, 130, 125]
```

```
import numpy as np
import pandas as pd
df = pd.DataFrame(np.c_[X, Y], columns=["x", "y"])
```

```
from statsmodels.formula.api import ols
model = ols("y ~ x", data=df)
result = model.fit()
result.params
```

```
Intercept    -27.358463
x              0.879627
dtype: float64
```

상수항을 포함하지 않는 3차방정식

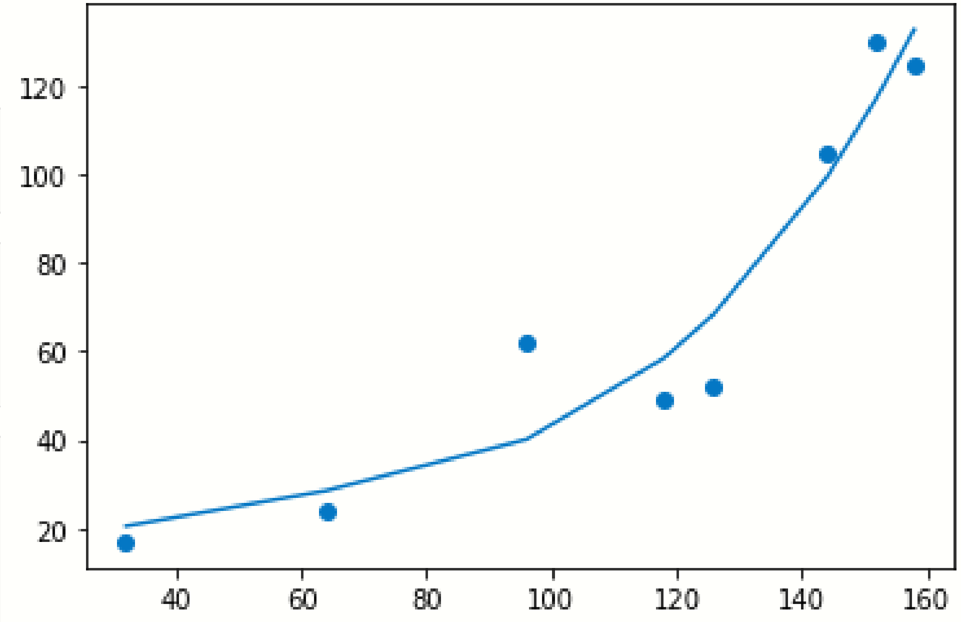

```
model2 = ols("y ~ x + I(x**2) + I(x**3) -1 ", data=df)
result2 = model2.fit()
result2.params
```

```
x            1.005637
I(x ** 2)   -0.013981
I(x ** 3)    0.000082
dtype: float64
```

```
y_ = result2.predict(df.x)
```

```
import matplotlib.pyplot as plt
%matplotlib inline
```

```
plt.scatter(X, Y)
plt.plot(X, y_)
plt.show()
```



2장. 회귀분석

6절. 정규화 선형회귀

정규화(Regularized) 선형회귀

회귀 계수(Weight)에 대한 제약 조건을 추가해서 모형의 과적합(Overfitting)을 막는 방법

Lasso 가중치의 절대값의 합을 최소화

$$\text{cost} = \sum e_i^2 + \lambda \sum |w_i|$$

Ridge 가중치들의 제곱합(squared sum of weights)을 최소화

$$\text{cost} = \sum e_i^2 + \lambda \sum w_i^2$$

Elastic Net 가중치의 절대값의 합과 제곱합을 동시에 제약 조건으로 가지는 모형

$$\text{cost} = \sum e_i^2 + \lambda_1 \sum |w_i| + \lambda_2 \sum w_i^2$$

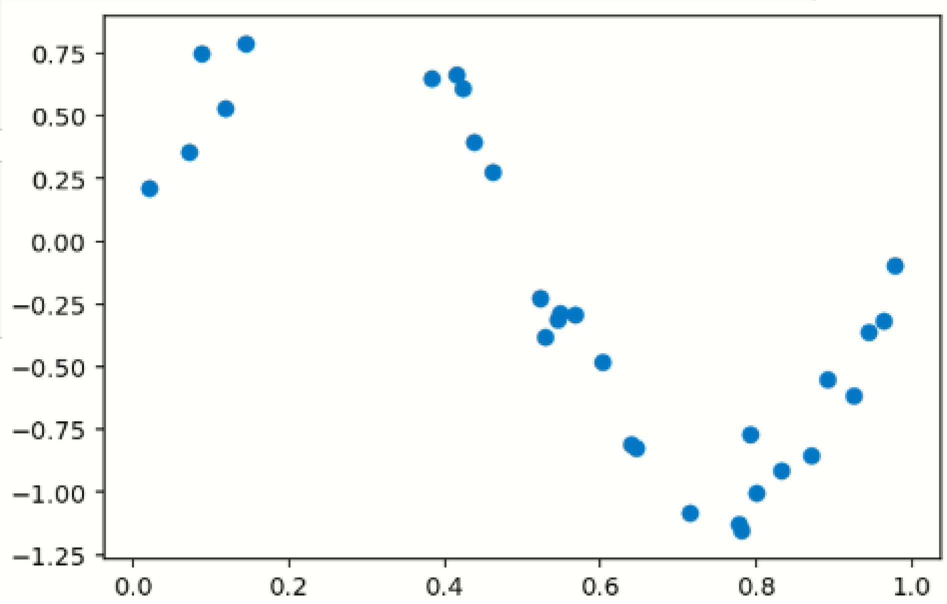
OLS 클래스의 fit_regularized() 메소드를 사용하여 회귀모형 계수를 구할 수 있음

샘플 데이터


```
import numpy as np
n_samples=30
np.random.seed(0)
X = np.sort(np.random.rand(n_samples)) # 0부터 1까지 30개
y = np.sin(2*np.pi * X) + np.random.randn(n_samples)*0.1
```

```
import matplotlib.pyplot as plt
%matplotlib inline
```

```
plt.scatter(X, y)
plt.show()
```



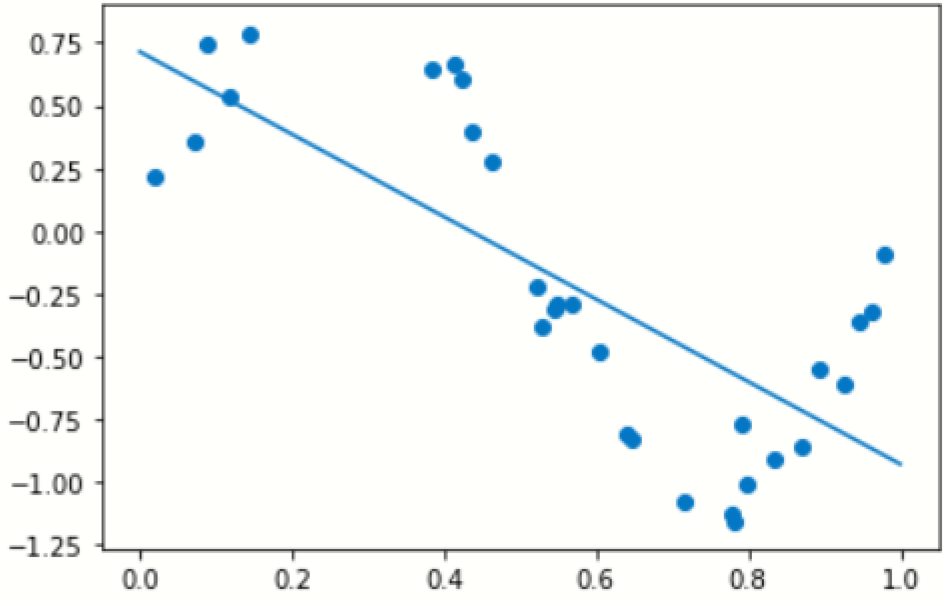
```python
import numpy as np
import pandas as pd
df = pd.DataFrame(np.c_[X, y], columns=["x", "y"])
df.head()
```

	x	y
0	0.020218	0.213138
1	0.071036	0.357444
2	0.087129	0.747487
3	0.118274	0.531167
4	0.143353	0.788347

```python
from statsmodels.formula.api import ols
model = ols("y ~ x", data=df)
result = model.fit()
result.params
```

```
Intercept    0.713959
x           -1.642204
dtype: float64
```

```
plt.scatter(X, y)
xx = np.linspace(0, 1, 1000) # 0부터 1까지 1000개
plt.plot(xx, result.predict({"x":xx}))
plt.show()
```



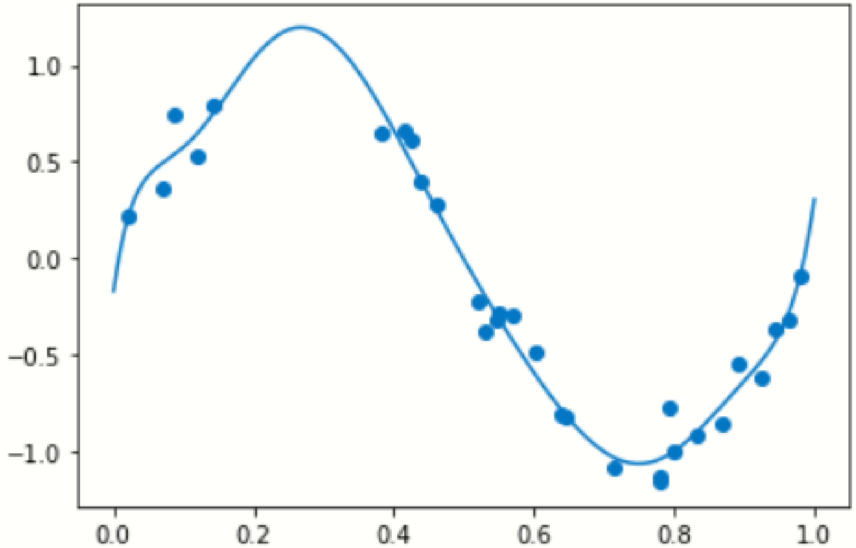
9차 방정식


```
from statsmodels.formula.api import ols
```

```
model9 = ols("y ~ x + I(x**2) + I(x**3) + I(x**4) + I(x**5)\
+ I(x**6) + I(x**7) + I(x**8) + I(x**9)", data=df)

result9 = model9.fit()
result9.params
```

```
Intercept        -0.169863
x                25.735773
I(x ** 2)      -428.141684
I(x ** 3)      3866.723115
I(x ** 4)    -18340.939667
I(x ** 5)     49326.072553
I(x ** 6)    -78884.743085
I(x ** 7)     74538.645164
I(x ** 8)    -38453.132196
I(x ** 9)      8350.254987
dtype: float64
```

```
plt.scatter(X, y)
plt.plot(xx, result9.predict({"x":xx}))
plt.show()
```

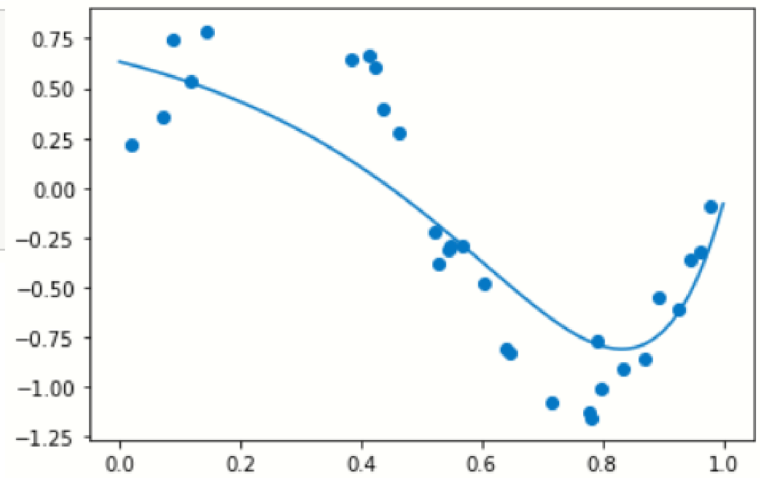


모수 L1_wt가 0이면 순수 Ridge 모형이 됨

```
result9 = model9.fit_regularized(L1_wt=0, alpha=0.01)
print(result9.params)
```

```
[ 0.63308745 -0.75705866 -1.07056551 -0.76835135 -0.355303
67  0.0121939
  0.29917825  0.50969248  0.65793698  0.75851865]
```

```
plt.scatter(X, y)
plt.plot(xx, result9.predict({"x":xx}))
plt.show()
```

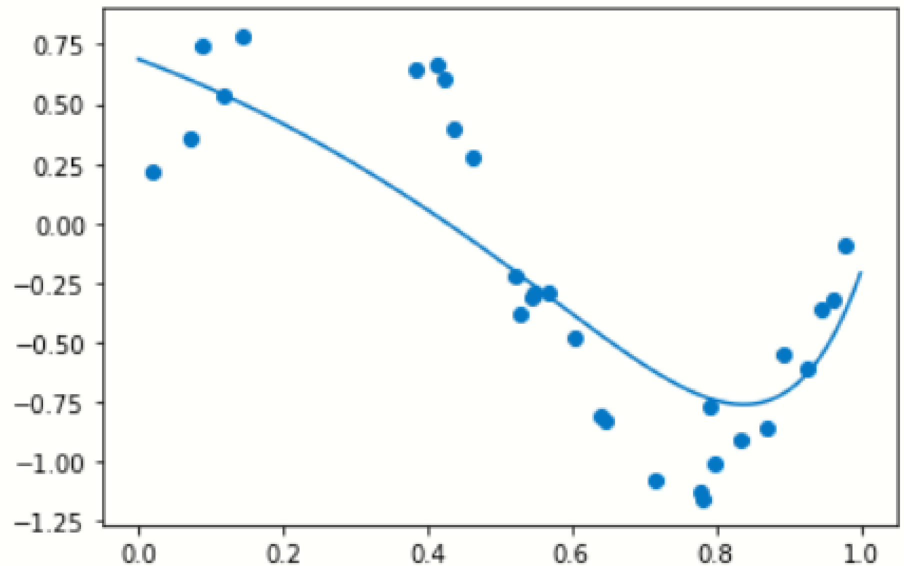


모수 L1_wt가 1이면 순수 Lasso 모형이 됨

```
result9 = model9.fit_regularized(L1_wt=1, alpha=0.01)
print(result9.params)
```

```
Intercept     0.687949
x            -1.129134
I(x ** 2)    -1.124878
I(x ** 3)     0.000000
I(x ** 4)     0.000000
I(x ** 5)     0.000000
I(x ** 6)     0.000000
I(x ** 7)     0.000000
I(x ** 8)     0.281484
I(x ** 9)     1.075281
dtype: float64
```

```
plt.scatter(X, y)
plt.plot(xx, result9.predict({"x":xx}))
plt.show()
```

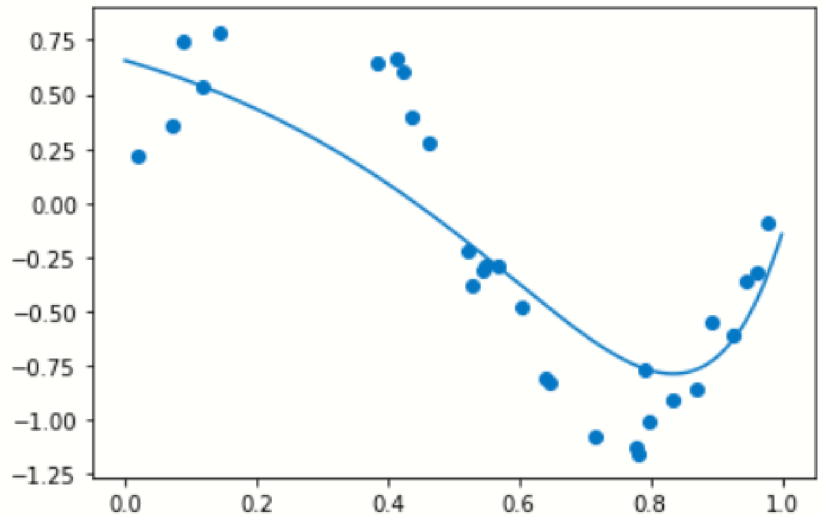


모수 L1_wt가 0.5이면 순수 Elastic Net 모형이 됨

```
result9 = model9.fit_regularized(L1_wt=0.5, alpha=0.01)
print(result9.params)
```

```
Intercept     0.656203
x            -0.849745
I(x ** 2)    -1.262902
I(x ** 3)    -0.425687
I(x ** 4)     0.000000
I(x ** 5)     0.000000
I(x ** 6)     0.000000
I(x ** 7)     0.304049
I(x ** 8)     0.631908
I(x ** 9)     0.801206
dtype: float64
```

```
plt.scatter(X, y)
plt.plot(xx, result9.predict({"x":xx}))
plt.show()
```



Scikit-Learn의 정규화 회귀모형

정규화 회귀모형을 위한 Ridge, Lasso, ElasticNet 이라는 별도의 클래스를 제공

```
sklearn.linear_model.Ridge(alpha=1.0, *, fit_intercept=True,
    normalize=False,  copy_X=True, max_iter=None, tol=0.001,
    solver='auto', random_state=None)
```

```
sklearn.linear_model.Lasso(alpha=1.0, *, fit_intercept=True,
    normalize=False, precompute=False, copy_X=True, max_iter=1000,
    tol=0.0001, warm_start=False, positive=False,
    random_state=None, selection='cyclic')
```

```
sklearn.linear_model.ElasticNet(alpha=1.0, *, l1_ratio=0.5,
    fit_intercept=True, normalize=False, precompute=False,
    max_iter=1000, copy_X=True, tol=0.0001, warm_start=False,
    positive=False, random_state=None, selection='cyclic')
```

샘플 데이터 생성

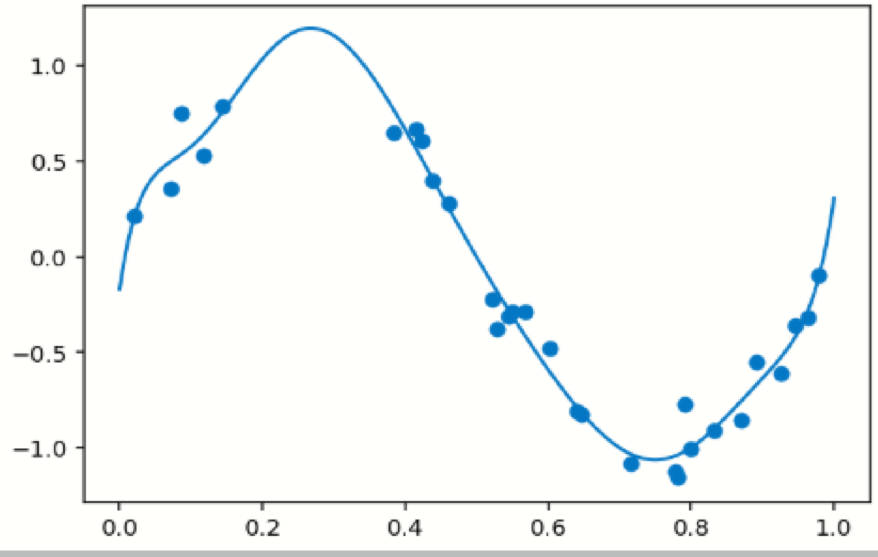

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import statsmodels.api as sm
```

```
np.random.seed(0)
x = np.sort(np.random.rand(30))
y = np.sin(2 * np.pi * x) + np.random.randn(30) * 0.1
X = x[:, np.newaxis]
```

```
def plot_model(model):
    plt.scatter(X, y)
    x = np.linspace(0, 1, 1000)
    plt.plot(x, model.predict(x[:, np.newaxis]))
    plt.show()
```

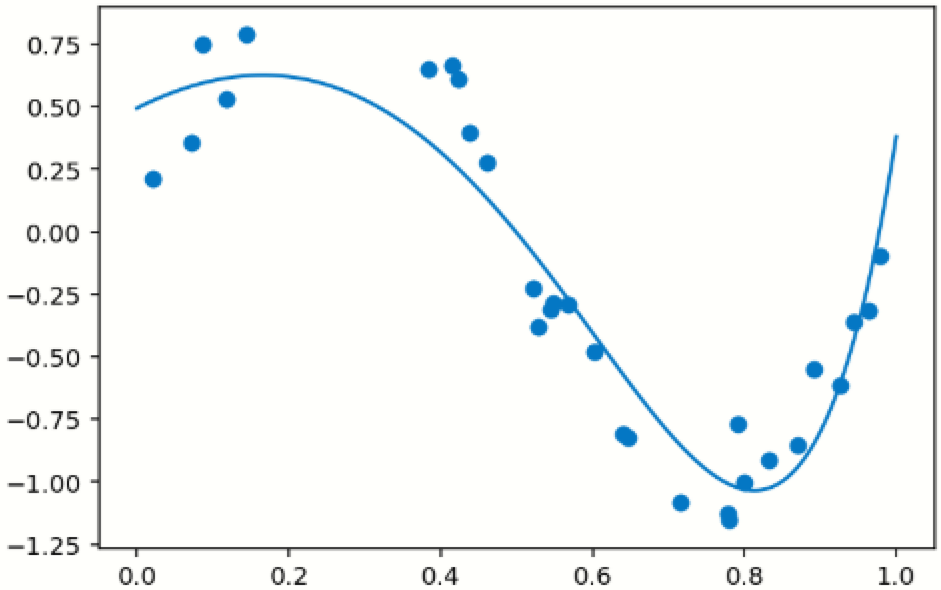
```python
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline
from sklearn.linear_model import  LinearRegression
```

```python
poly = PolynomialFeatures(9)
model = make_pipeline(poly, LinearRegression()).fit(X, y)
#print(model.steps[1][1].coef_)
plot_model(model)
```



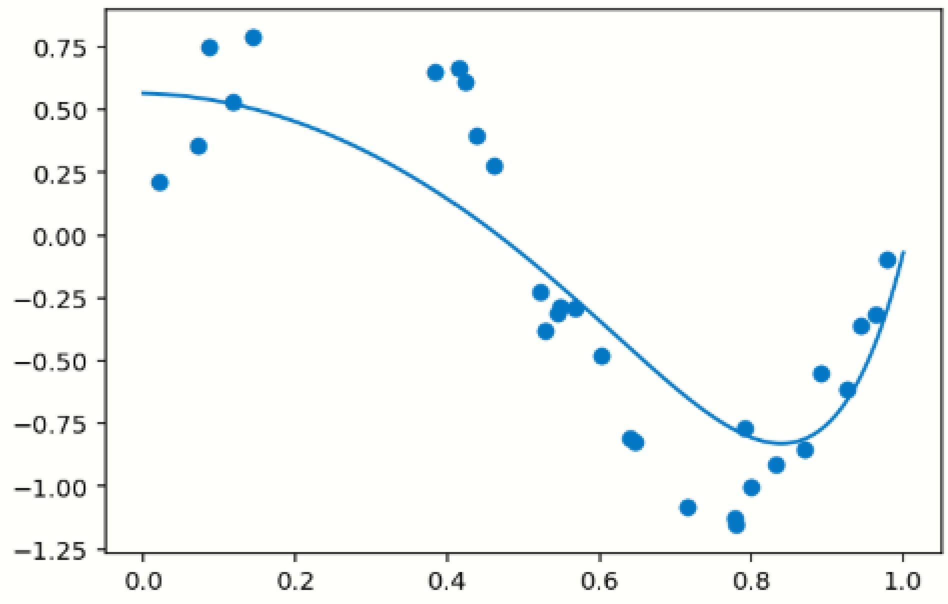
정규화를 하지 않는 선형회귀 모형을 만듬

```python
from sklearn.linear_model import Ridge
model = make_pipeline(poly, Ridge(alpha=0.01)).fit(X, y)
# print(model.steps[1][1].coef_)
plot_model(model)
```



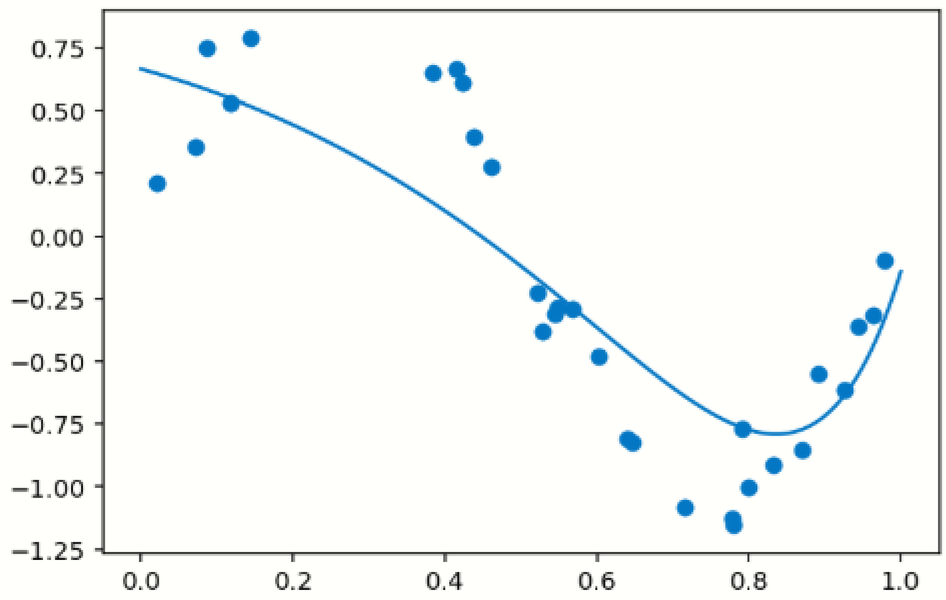
Lasso 회귀모형


```python
from sklearn.linear_model import Lasso
model = make_pipeline(poly, Lasso(alpha=0.01)).fit(X, y)
# print(model.steps[1][1].coef_)
plot_model(model)
```



Elastic Net 회귀모형

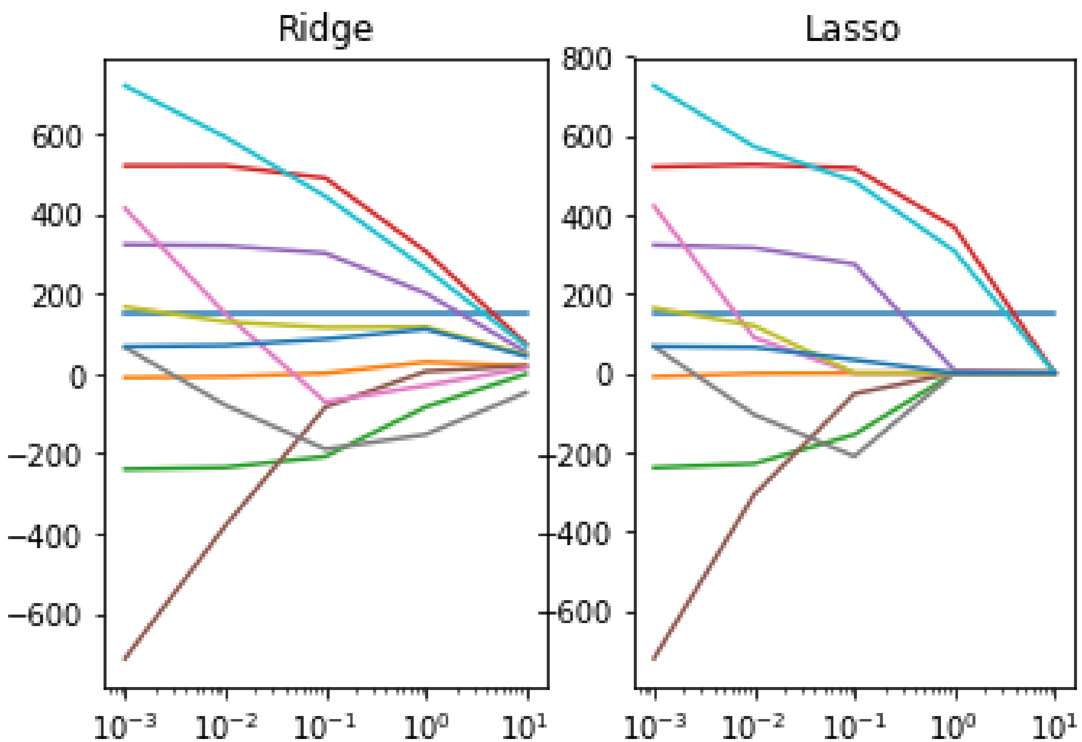

```python
from sklearn.linear_model import ElasticNet
elastic = ElasticNet(alpha=0.01, l1_ratio=0.5)
model = make_pipeline(poly, elastic).fit(X, y)
# print(model.steps[1][1].coef_)
plot_model(model)
```



Ridge 모형과 Lasso 모형의 차이

Ridge 모형은 가중치 계수를 모두 한꺼번에 축소 시킴

Lasso 모형은 일부 가중치 계수가 다른 가중치 계수에 비해 먼저 0으로 수렴하는 특성



2장. 회귀분석

7절. 다중회귀분석

단순회귀분석의 확장판으로 선형 모델을 기초로 독립변수가 2개 이상일 때 사용

다중 회귀분석에서 가장 기본적인 업무는 상수와 베타회귀계수를 구하는 것

$$\widehat{Y} = \widehat{\beta}_0 + \widehat{\beta}_1 X_1 + \widehat{\beta}_2 X_2 + \cdots\cdots + \widehat{\beta}_k X_k$$

^(hat, 추정자)

- 회귀식에서 ^(hat, 추정자)은 잔차(residual)를 의미하며 종속변수와 독립변수와의 관계를 밝히는 통계모형에서 모형에 의하여 추정된 종속변수의 값과 실제 관찰된 종속변수 값과의 차이를 의미

다중회귀분석을 이용한 집값 예측


```
from sklearn.datasets import load_boston
boston = load_boston()
X = boston.data
y = boston.target
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

```
from sklearn.linear_model import LinearRegression
model_boston = LinearRegression()
model_boston.fit(x_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
model_boston.score(x_train, y_train)
```

```
0.7224912856655239
```

```
model_boston.score(x_test, y_test)
```

```
0.7732420592861828
```

```
pred = model_boston.predict(x_test)
```

```
from sklearn.metrics import r2_score
r2_score(y_test, pred)
```

0.7732420592861828

```
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_test, pred) # MSE
mse
```

15.556720572795562

```
import math
math.sqrt(mse) # RMSE
```

3.944200878859438

다중회귀식의 추정방법

- 동시 입력법(전체 입력변수 투입)
  - 모든 독립변수들을 포함하여 분석하는 방법
  - 이를 통해 특정 독립변수의 영향력을 알 수 있음
- 단계적 선택법(Step-wise)
  - 다른 변수들이 회귀식에 존재할 때 종속변수에 영향력이 있는 변수들만을 회귀식에 포함시키는 방법
  - 설명력이 높은 즉, 유의 확률 p가 가장 작은 변수의 순으로 회귀식에 포함시킴
- 후진 소거법
  - 모든 독립변수를 모두 포함시킨 상태에서 기여도가 적은 변수부터 하나씩 제거해서 모델에 남아있는 변수들의 유의확률이 유의수준 이하가 될 때까지 삭제하는 방법
- 전진 선택법
  - 독립변수가 하나도 포함되지 않은 모델에서 시작해서 F 값에 가장 큰 기여를 하는 변수(유의확률 p가 가장 작은)를 순서대로 하나씩 더해가는 방법
- 제거법
  - 독립변수가 없이 절편(상수항, bias)으로 구성된 모형을 만듬

보스턴 집값 데이터를 이용한 다중회귀식 추정

```
import statsmodels.api as sm
Boston = sm.datasets.get_rdataset("Boston", package="MASS")
boston_df = Boston.data
boston_df.head()
```

데이터 불러오기

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	m
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	2
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	2
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	3
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	3
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	3

보스턴 집값 데이터를 이용한 다중회귀식 추정


```
import statsmodels.formula.api as smf
formula = "medv~" + "+".join(boston_df.iloc[:,:-1].columns)
model_boston = smf.ols(formula=formula, data=boston_df).fit()
model_boston.summary()
```

```
import statsmodels.formula.api as smf
formula = "medv ~ rad + zn + rm + chas + age -1"
model_boston2 = smf.ols(formula=formula, data=boston_df).fit()
model_boston2.summary()
```

model_boston.summary() 결과

OLS Regression Results

Dep. Variable:	medv	R-squared:	0.736
Model:	OLS	Adj. R-squared:	0.729
Method:	Least Squares	F-statis	
Date:	Mon, 26 Nov 2018	Prob (F-statis	
Time:	17:52:23	Log-Likeliho	
No. Observations:	506	AIC:	3034.
Df Residuals:	493	BIC:	3088.

model_boston2.summary() 결과

OLS Regression Results

Dep. Variable:	medv	R-squared:	0.932
Model:	OLS	Adj. R-squared:	0.931
			1363.
			5.05e-289
			-1654.6
No. Observations:	506	AIC:	3319.
Df Residuals:	501	BIC:	3340

두 요약의 결과에서 결정계수(R-squared)에 주목

상관계수, 결정계수, 수정된 결정계수

상관계수와 결정계수

- 상관계수(R; correlation coefficient) : 상관분석에서 상관관계의 정도를 나타내는 계수
- 결정계수(R2; coefficient of determination, R–squared) :상관계수를 제곱한 값
- 결정계수는 회귀식이 자료를 얼마나 잘 설명하고 있는지 즉, 독립변수가 종속변수를 얼마나 잘 설명하고 있는지를 나타낸 계수
- 결정계수는 상관계수와 마찬가지로 0~1사이의 값을 가지며, 일반적으로 0.65 (65%) 보다 클 경우 회귀식을 잘 설명한다고 판단

수정된 결정계수

- 독립변수의 수가 늘어날수록 결정계수가 높아지는 단점이 있어 이를 보완하기 위해 도입
- 다중회귀분석에서는 결정계수가 아닌 수정된 결정계수를 언급해야 함
- 여전히 많은 자료에서 결정계수만 언급되고 있지만 자료를 공개하는 대상의 특성에 따라 적절하게 사용할 필요가 있음

OLS Regression Results

Dep. Variable:	medv	**R-squared:**	0.932
Model:	OLS	**Adj. R-squared:**	0.931
Method:	Least Squares	**F-statistic:**	1363.
Date:	Mon, 26 Nov 2018	**Prob (F-statistic):**	5.05e-289
Time:	21:01:30	**Log-Likelihood:**	-1654.6
No. Observations:	506	**AIC:**	3319.
Df Residuals:	501	**BIC:**	3340.
Df Model:	5		
Covariance Type:	nonrobust		

OLS Regression Results

7절. 다중회귀분석

```
import statsmodels.api as sm
Boston = sm.datasets.get_rdataset("Boston",package="MASS")
boston_df = Boston.data
```

```
import statsmodels.formula.api as smf
formula = "medv ~ rad + zn + rm + chas + age -1"
model_boston = smf.ols(formula=formula, data=boston_df).fit()
model_boston.summary()
```

OLS Regression Results

Dep. Variable:	medv	R-squared (uncentered):	0.932
Model:	OLS	Adj. R-squared (uncentered):	0.931
Method:	Least Squares	F-statistic:	1363.
Date:	Sat, 25 Jul 2020	Prob (F-statistic):	5.05e-289
Time:	09:58:02	Log-Likelihood:	-1654.6
No. Observations:	506	AIC:	3319.
Df Residuals:	501	BIC:	3340.
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
rad	-0.2183	0.037	-5.924	0.000	-0.291	-0.146
zn	0.0161	0.015	1.056	0.291	-0.014	0.046
rm	4.7232	0.147	32.166	0.000	4.435	5.012
chas	5.6944	1.128	5.047	0.000	3.478	7.911
age	-0.0792	0.012	-6.358	0.000	-0.104	-0.055

Omnibus:	234.375	Durbin-Watson:	0.708
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1253.471
Skew:	2.008	Prob(JB):	6.49e-273
Kurtosis:	9.583	Cond. No.	299.

회귀분석의 검증 항목 – 잔차의 독립성

- 회귀분석의 기본 가정 사항 중 잔차의 독립성이 있음
- 잔차가 다른 잔차에 영향을 미치게 되는 경우를 자기상관(Autocorrelation)이라고 하는데 자기상관이 높으면 분석의 신뢰성을 잃게 됨
- 자기상관은 앞의 잔차항이 뒤의 잔차항에 영향을 미치는 경우로 주로 시계열 자료에서 많이 관찰됨
- 회귀모형에서 자기상관이 발생하게 되면, 회귀모형의 기본가정인 ‘잔차항들은 서로 독립이다’라는 가정을 위배하게 됨
- 이러한 것을 무시하고 회귀모형을 적용하면 일반적으로 회귀계수에 대한 검정 통계량인 *t* 값과 회귀모형에 대한 검정 통계량인 F 값, 그리고 R2의 값을 실제보다 증가시키는 경향이 있음
- 더빈 왓슨(Durbin−Watson)의 통계량은 d라고 하며, 자기상관(Autocorrelation)을 검증하는 통계량임
- 잔차의 독립성은 Durbin−Watson(더빈 왓슨)값으로 판단하게 되는데 0에 가까울수록 양의 자기상관, 4에 가까울수록 음의 자기상관이 있다고 판단하며, 2에 가까울수록 자기 상관이 없다고 판단
- 보통 1.5 ~ 2.5 사이의 값을 적용. 더빈 왓슨은 오차항에 자기상관이 있는지 없는지를 판단하기 위해 사용

	coef	std err	t	P>|t|	[0.025	0.975]
rad	-0.2183	0.037	-5.924	0.000	-0.291	-0.146
zn	0.0161	0.015	1.056	0.291	-0.014	0.046
rm	4.7232	0.147	32.166	0.000	4.435	5.012
chas	5.6944	1.128	5.047	0.000	3.478	7.911
age	-0.0792	0.012	-6.358	0.000	-0.104	-0.055

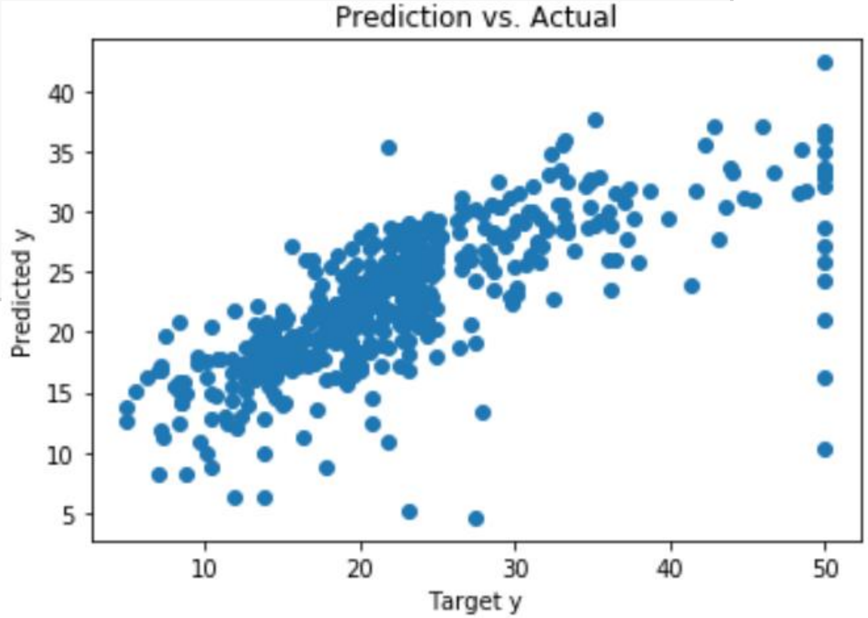
Omnibus:	234.375	Durbin-Watson:	0.708
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1253.471
Skew:	2.008	Prob(JB):	6.49e-273
Kurtosis:	9.583	Cond. No.	299.

회귀분석의 검증 항목 – 잔차의 정규성

잔차의 정규성은 데이터 탐색(summary()) 기능을 이용하는 것이 아닌 그래프를 보고 판단

```
y_pred = model_boston2.predict(boston_df) # boston_df.iloc[:,:-1]
```

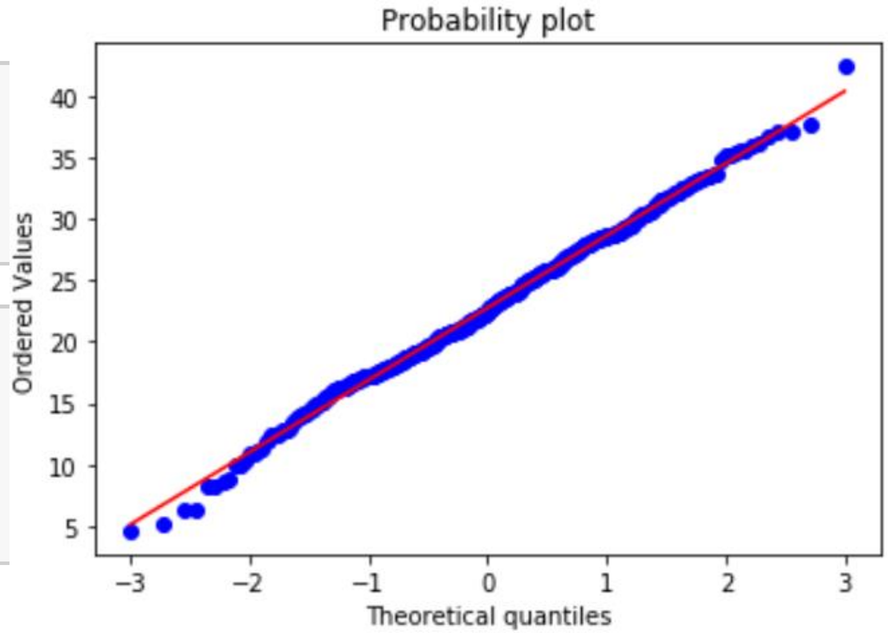
```
import matplotlib.pyplot as plt
%matplotlib inline
fig = plt.figure()
plt.scatter(boston_df.iloc[:,-1], y_pred)
plt.xlabel("Target y")
plt.ylabel("Predicted y")
plt.title("Prediction vs. Actual")
plt.show()
```



- 예측한 집값 데이터(y_pred)를 이용한 P-P 도표
- 대각선을 중심으로 데이터들이 균일하게 분포되어 있어야 함
- 어느 한 데이터가 대각선에서 많이 떨어져 있다면 그 데이터를 삭제하거나 다시 측정해 볼 필요가 있음
- 대각선에 다른 무리들과 떨어져 있는 값이 이상값(이상치)
- 이상값이 많을수록 결정계수는 낮아지며, 그만큼 회귀식의 설명력 또한 낮아지게 됨

```
from scipy import stats
import matplotlib.pyplot as plt
%matplotlib inline
```

```
fig = plt.figure()
res = stats.probplot(y_pred, plot=plt)
plt.title("Probability plot")
plt.show()
```



다중 공선성

- 독립변수가 여러 개일 경우 그 변수들 끼리 상관관계가 높을 경우
- 다중공선성 판단
  - 독립변수들끼리의 상관계수가 90% 이상
  - VIF(Variance Inflation Factor; 분산 확대 인자)가 10 이상으로 나올 경우

$$VIF = 1/(1 - R^2)$$

  - 공차한계(Tolerance)를 통해서도 다중공선성을 판단
    - 공차한계는 어떤 독립변수가 다른 독립변수들에 의해서 설명되지 않는 부분을 의미

$$\text{변수 } i\text{의 공차한계} = 1 - R_i^2$$

    - 결정계수가 클수록 공차한계 값이 작아짐
    - 공차한계 값이 작을수록 그 독립변수가 다른 독립변수들에 의해 설명되는 정도가 크다는 것을 의미
    - 다중공선성이 높다는 것과 같음
- VIF가 10 이상이거나 공차한계가 0.10 이하일 경우에는 공선성이 존재한다고 평가
- 절대적인 기준은 없으므로 적절한 수준에서 판단해야 함

다중공선성 의심 상황

- 다중공선성 의심 상황
  - Data 수에 비해 과다한 독립변수를 사용했을 때
  - 독립변수들의 상관계수가 크게 나타날 때
  - 한 독립변수를 회귀모형에 추가하거나 제거하는 것이 회귀계수의 크기나 부호에 큰 변화를 줄 때
  - 새로운 Data를 추가하거나 기존의 Data를 제거하는 것이 회귀계수의 크기나 부호에 큰 변화를 줄 때
  - 중요하다고 생각되어지는 독립변수에 대한 P 값이 크게 나타나 통계적 차이가 없을 때(회귀계수의 부호가 과거의 경험이나 이론적인 면에서 기대되는 부호와 정반대일 때)

- 다중 공선성이 발생되면, 회귀 모형의 적합성이 떨어지고, 다른 중요한 독립변수가 모형에서 제거될 가능성이 높음
- 결정계수의 값이 과대하게 나타날 수 있거나 설명력은 좋은데 예측력이 떨어질 수 있게 됨
- 공선성을 낮추는 방법은 상관관계가 높은 독립변수를 제거하거나 변수 선택방법을 이용해서 분석해야 함
- 만일 VIF가 10보다 큰 값이 있다면 VIF값이 가장 큰 변수를 제거한 후 회귀분석을 재실행

보스턴 집값 데이터의 VIF 확인


```python
import statsmodels.api as sm
Boston = sm.datasets.get_rdataset("Boston", package="MASS")
boston_df = Boston.data

formula = "medv~" + "+".join(boston_df.columns[:-1])
```

```python
from patsy import dmatrices
y, X = dmatrices(formula, boston_df, return_type="dataframe")
```

```python
import pandas as pd
vif = pd.DataFrame()
```

	VIF Factor	features
0	585.265238	Intercept
10	9.008554	tax
9	7.484496	rad
5	4.393720	nox
3	3.991596	indus
8	3.955945	dis
7	3.100826	age
13	2.941491	lstat
2	2.298758	zn
6	1.933744	rm
11	1.799084	ptratio
1	1.792192	crim
12	1.348521	black
4	1.073995	chas

```python
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif["VIF Factor"] = [variance_inflation_factor(X.values, i)
                     for i in range(X.shape[1])]
vif["features"] = X.columns
vif.sort_values(by="VIF Factor", ascending=False)
```

VIF 계산을 위한 함수 정의


```python
def get_vif(formula, df):
    from patsy import dmatrices
    y, X = dmatrices(formula, df, return_type="dataframe")
    import pandas as pd
    vif = pd.DataFrame()
    from statsmodels.stats.outliers_influence import variance_inflation_factor
    vif["VIF Factor"] = [variance_inflation_factor(X.values, i)
                         for i in range(X.shape[1])]
    vif["features"] = X.columns
    vif.sort_values(by="VIF Factor", ascending=False, inplace=True)
    return vif
```

```python
formula = "medv~" + "+".join(boston_df.columns[:-1]) + "-1"
get_vif(formula, boston_df)
```

	VIF Factor	features
10	85.029547	ptratio
5	77.948283	rm
4	73.894947	nox
9	61.227274	tax
6	21.386850	age
11	20.104943	black
8	15.167725	rad
7	14.699652	dis
2	14.485758	indus
12	11.102025	lstat
1	2.844013	zn
0	2.100373	crim
3	1.152952	chas

2장. 회귀분석

8절. 회귀모형 성능평가

Scikit-learn의 모형 평가 방법

예측 모형의 score 메소드

- 예측 모형들은 score() 메소드를 통해 예측 모형을 평가할 수 있는 기본 기준을 제공합니다.

metrics 함수

- 메트릭(sklearn.metrics) 모듈은 분류, 회귀 그리고 군집 등 예측 모형의 평가를 위한 함수들을 제공합니다.

scoring 매개변수

- sklearn.model_selection 모듈의 cross_val_score() 함수 또는 GridSearchCV 클래스 등 교차 검증(cross-validation)을 사용하는 모형 평가 도구들은 내부적으로 scoring 매개변수를 이용해 모형 평가 규칙을 정의합니다.
- scoring 매개변수의 가능한 값은 sklearn.metrics.SCORES.keys() 함수를 통해 알 수 있습니다.

Regression 모형의 scoring 매개변수의 값과 평가함수

scoring 속성의 값	metrics 모듈의 회귀모형 평가함수	참고
'explained_variance'	metrics.explained_variance_score	설명 분산
'neg_mean_absolute_error'	metrics.mean_absolute_error	
'neg_mean_squared_error'	metrics.mean_squared_error	
'neg_mean_squared_log_error'	metrics.mean_squared_log_error	
'neg_median_absolute_error'	metrics.median_absolute_error	
'r2'	metrics.r2_score	결정계수