12장 회귀분석

- 회귀분석 : 인과관계가 의심되는 복수의 변수를 사용하여 어느 변수로부터 다른 변수의 값을 예측하는 기법.
- 이때 원인이 되는 변수를 독립변수라고 하고, 결과가 되는 변수를 종속변수라고 함.
- 즉, 독립변수와 종속변수 사이의 선형식을 구하여 독립변수의 값이 주어졌을 때 종속변수의 값을 예측하고, 종속변수에 대한 독립변수의 영향력을 분석하는 방법.

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy import stats

# 스탯스모델즈 라이브러리 추가 - 회귀분석 또는 분산분석을 위한 함수들이 포함되어있음
import statsmodels.formula.api as smf  # 회귀분석용

%precision 3
%matplotlib inline
```

In [ ]:
```
df = pd.read_csv('../data/ch12_scores_reg.csv')
n = len(df)
print(n)
df.head()
```

1) 단순회귀모형

- 단순회귀분석 : 독립변수(원인)와 종속변수(결과)가 1개씩인 모형. ---> 독립변수가 종속변수에 미치는 영향을 밝히는 통계적 방법으로 상관계수에 기초 함.
- 연구문제 : 쪽지시험 평균점수(독립변수)는 기말고사 점수(종속변수)에 유의한 영향을 미치는가?

In [ ]:
```
# 기말고사점수를 y로, 쪽지시험평균점수를 x, 독립변수 갯수 p는 1로 설정.

x = np.array(df['quiz'])
y = np.array(df['final_test'])
p = 1
```

In [ ]:
```
# 산점도와 회귀직선 그리기 (3장 내용 참조)

poly_fit = np.polyfit(x, y, 1)
poly_1d = np.poly1d(poly_fit)
xs = np.linspace(x.min(), x.max())
ys = poly_1d(xs)

fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111)
ax.set_xlabel('quiz')
ax.set_ylabel('final test')
ax.plot(xs, ys, color='gray',
        label=f'{poly_fit[1]:.2f}+{poly_fit[0]:.2f}x')
ax.scatter(x, y)
```

```
ax.legend()

plt.show()
```

회귀분석에서의 가설

statsmodels에 의한 회귀분석

: statsmodels는 통계 모델링을 위한 강력한 도구로서 분산분석과 회귀분석을 포함한 다양한 분석 기능을 제공

- statsmodels.formula.api.ols()함수는 회귀분석을 수행하는 함수.
- Ordinary Least Squares (OLS, 최소제곱법)방법을 사용.

- statsmodels의 ols함수 인수로는 종속변수와 독립변수의 관계를 나타내는 문자열, 데이터프레임을 전달해야함.
- 추가로 .fit() 메서드를 실행해서 결과를 추출함.
- .summary() 메서드를 이용하여 분석 결과를 표형태로 요약하여 화면에 출력함.

In [ ]:
```
# 종속변수 'final_test'와 독립변수 'quiz' 사이의 관계를 타나내는 문자열 만들기.
formula = 'final_test ~ quiz'

# 종속변수, 독립변수 관계를 나타내는 문자열과, 데이터프레임을 전달하여 단순회귀분석하
result = smf.ols(formula, df).fit()

# 단순회귀분석 결과를 표로 요약하여 화면에 출력하기
result.summary()
```

(결과)

1. 모형 설명력을 확인 해야 함 : R-squared(R제곱)과 Adj. R-squared(조정결정계수)으로 판단.
   - R제곱값은 독립변수가 종속변수를 얼마나 설명하는지 판단하는 수치이다.
   - 위 결과 수정된 R제곱(조정결정계수)값이 0.658이라면 65.8%를 설명한다고 할 수 있다.
   - 조정결정계수가 30% 미만값이면 회귀모형은 의미없다고 분석함.

1. 모형 적합도 확인 해야 함 : F-statistic(F값)과 Prob(P값)으로 판단.
   - p값이 0.05 미만이면 회귀모형이 적합하다고 분석한다.

1. Durbin-Watson통계량으로 잔차의 독립성 여부를 판단.
   - 잔차는 회귀분석에서의 오차 개념임.
   - 규칙성이 없이 랜덤해야하는데, 이 통계량이 2에 가까울수록 잔차에 독립성이 있다고 분석한다.

1. 회귀 계수 : 회귀직선의 식을 추정한 결과 : 기말고사(y) = 23.70 + 6.55 * 퀴즈평균(x)
   - coef은 비표준화 계수 회귀계수의 추정값을 나타냄.
   - Intercept는 회귀직선의 y절편, 독립변수인 quiz 인과관계 계수 추정값으로 회귀직선의 기울기
   - 독립변수인 quiz의 유의확률(p값)이 0.000이므로 유의수준(0.05)보다 미만의 값으로 귀무가설 기각, 대립가설 채택됨.

--> 즉, 독립변수 quiz는 종속변수 기말고사와 유의미하다, 관련성이 있다, 인과관계가 있다고 분석 됨.

아울러, 회귀식으로 퀴즈평균점수로 기말고사 점수를 예측가능하다.

회귀계수

In [ ]:
```
X = np.array([np.ones_like(x), x]).T
X
```

In [ ]:
```
beta0_hat, beta1_hat = np.linalg.lstsq(X, y)[0]
beta0_hat, beta1_hat
```

In [ ]:
```
y_hat = beta0_hat + beta1_hat * x
eps_hat = y - y_hat
```

In [ ]:
```
s_var = np.var(eps_hat, ddof=p+1)
s_var
```

In [ ]:
```
C0, C1 = np.diag(np.linalg.pinv(np.dot(X.T, X)))
```

In [ ]:
```
np.sqrt(s_var * C0), np.sqrt(s_var * C1)
```

In [ ]:
```
rv = stats.t(n-2)

lcl = beta0_hat - rv.isf(0.025) * np.sqrt(s_var * C0)
hcl = beta0_hat - rv.isf(0.975) * np.sqrt(s_var * C0)
lcl, hcl
```

In [ ]:
```
rv = stats.t(n-2)

lcl = beta1_hat - rv.isf(0.025) * np.sqrt(s_var * C1)
hcl = beta1_hat - rv.isf(0.975) * np.sqrt(s_var * C1)
lcl, hcl
```

In [ ]:
```
t = beta1_hat / np.sqrt(s_var * C1)
t
```

In [ ]:
```
(1 - rv.cdf(t)) * 2
```

In [ ]:
```
t = beta0_hat / np.sqrt(s_var * C0)
t
```

In [ ]:
```
(1 - rv.cdf(t)) * 2
```

2) 다중회귀모형

In [ ]:
```
formula = 'final_test ~ quiz + sleep_time'
result = smf.ols(formula, df).fit()
result.summary()
```

회귀계수

In [ ]:
```
x1 = df['quiz']
x2 = df['sleep_time']
y = df['final_test']
p = 2
```

In [ ]:
```
X = np.array([np.ones_like(x1), x1, x2]).T
beta0_hat, beta1_hat, beta2_hat = np.linalg.lstsq(X, y)[0]
beta0_hat, beta1_hat, beta2_hat
```

In [ ]:
```
y_hat = beta0_hat + beta1_hat * x1 + beta2_hat * x2
eps_hat = y - y_hat
```

In [ ]:
```
s_var = np.sum(eps_hat ** 2) / (n - p - 1)
C0, C1, C2 = np.diag(np.linalg.pinv(np.dot(X.T, X)))
```

In [ ]:
```
rv = stats.t(n-p-1)

lcl = beta2_hat - rv.isf(0.025) * np.sqrt(s_var * C2)
hcl = beta2_hat - rv.isf(0.975) * np.sqrt(s_var * C2)
lcl, hcl
```

가변수

In [ ]:
```
formula = 'final_test ~ quiz + sleep_time + school_method'
result = smf.ols(formula, df).fit()
result.summary()
```

모형의 선택

In [ ]:
```
x = np.array(df['quiz'])
y = np.array(df['final_test'])
p = 1

formula = 'final_test ~ quiz'
result = smf.ols(formula, df).fit()
result.summary()
```

In [ ]:
```
y_hat = np.array(result.fittedvalues)
y_hat
```

In [ ]:
```
eps_hat = np.array(result.resid)
eps_hat
```

In [ ]:
```
np.sum(eps_hat ** 2)
```

결정계수

In [ ]:
```
total_var = np.sum((y - np.mean(y))**2)
exp_var = np.sum((y_hat - np.mean(y))**2)
unexp_var = np.sum(eps_hat ** 2)
```

In [ ]:
```
total_var, exp_var + unexp_var
```

In [ ]:
```
exp_var / total_var
```

In [ ]:
```
np.corrcoef(x, y)[0, 1] ** 2
```

조정결정계수

In [ ]:
```
1 - (unexp_var / (n - p - 1)) / (total_var / (n - 1))
```

F검정

In [ ]:
```
f = (exp_var / p)  / (unexp_var / (n - p - 1))
f
```

In [ ]:
```
rv = stats.f(p, n-p-1)
1 - rv.cdf(f)
```

최대 로그 우도와 AIC

In [ ]:
```
prob = 0.3
coin_result = [0, 1, 0, 0, 1]

rv = stats.bernoulli(prob)
L = np.prod(rv.pmf(coin_result))
L
```

In [ ]:
```
ps = np.linspace(0, 1, 100)
Ls = [np.prod(stats.bernoulli(prob).pmf(coin_result))
      for prob in ps]

fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111)
ax.plot(ps, Ls, label='likelihood function', color='gray')
ax.legend(fontsize=16)
plt.show()
```

In [ ]:
```
prob = 0.4
rv = stats.bernoulli(prob)
mll = np.sum(np.log(rv.pmf([0, 1, 0, 0, 1])))
mll
```

In [ ]:
```
rv = stats.norm(y_hat, np.sqrt(unexp_var / n))
mll = np.sum(np.log(rv.pdf(y)))
mll
```

In [ ]:
```
aic = -2 * mll + 2 * (p+1)
aic
```

In [ ]:
```
bic = -2 * mll + np.log(n) * (p+1)
bic
```

모형의 타당성

In [ ]:
```
formula = 'final_test ~ quiz + sleep_time'
result = smf.ols(formula, df).fit()
result.summary()
```

In [ ]:
```
eps_hat = np.array(result.resid)
```

정규성의 검정

In [ ]:
```
stats.skew(eps_hat)
```

In [ ]:
```
stats.kurtosis(eps_hat, fisher=False)
```

더빈-왓슨비

In [ ]:
```
np.sum(np.diff(eps_hat, 1) ** 2) / np.sum(eps_hat ** 2)
```

다중공선성

In [ ]:
```
df['mid_test'] = df['quiz'] * 2
df.head()
```

In [ ]:
```
formula = 'final_test ~ quiz + mid_test'
result = smf.ols(formula, df).fit()
result.summary()
```