빅데이터 프로그래밍
Ch10. Statistics

School of Information Convergence
Prof. Dong-Hyuk Im

광운대학교
KwangWoon University

Preview

- 기술통계
- 회귀분석
- 결과 시각화
- 상관 관계
- 로지스틱 회귀 분석

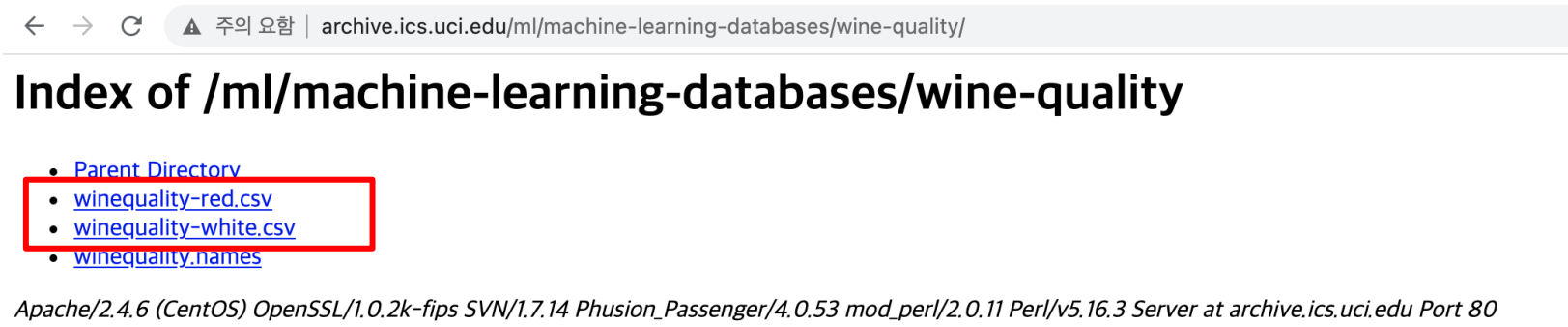
*“데이터 과학 기반의 파이썬 빅데이터 분석“ , 이재영, 한빛 아카데미

통계 분석의 핵심 개념

- 기술 통계
  - 데이터의 특성을 나타내는 수치를 이용해 분석하는 기본적인 통계 방법
  - 평균, 중앙값, 최빈값등을 구함
- 회귀 분석
  - 독립변수 x와 종속 변수 y간의 상호 연관성 정도를 파악하기 위한 분석 기법
  - 하나의 변수가 변함에 따라 대응되는 변수가 어떻게 변하는지를 측정하는 것
  - 변수 간의 인과 관계를 분석할 때 많이 사용
  - 독립 변수가 한 개이면 단순 회귀분석, 두 개 이상이면 다중 회귀 분석
  - 독립 변수와 종속 변수의 관계에 따라 선형 회귀 분석과 비선형 회귀 분석으로 나눠짐

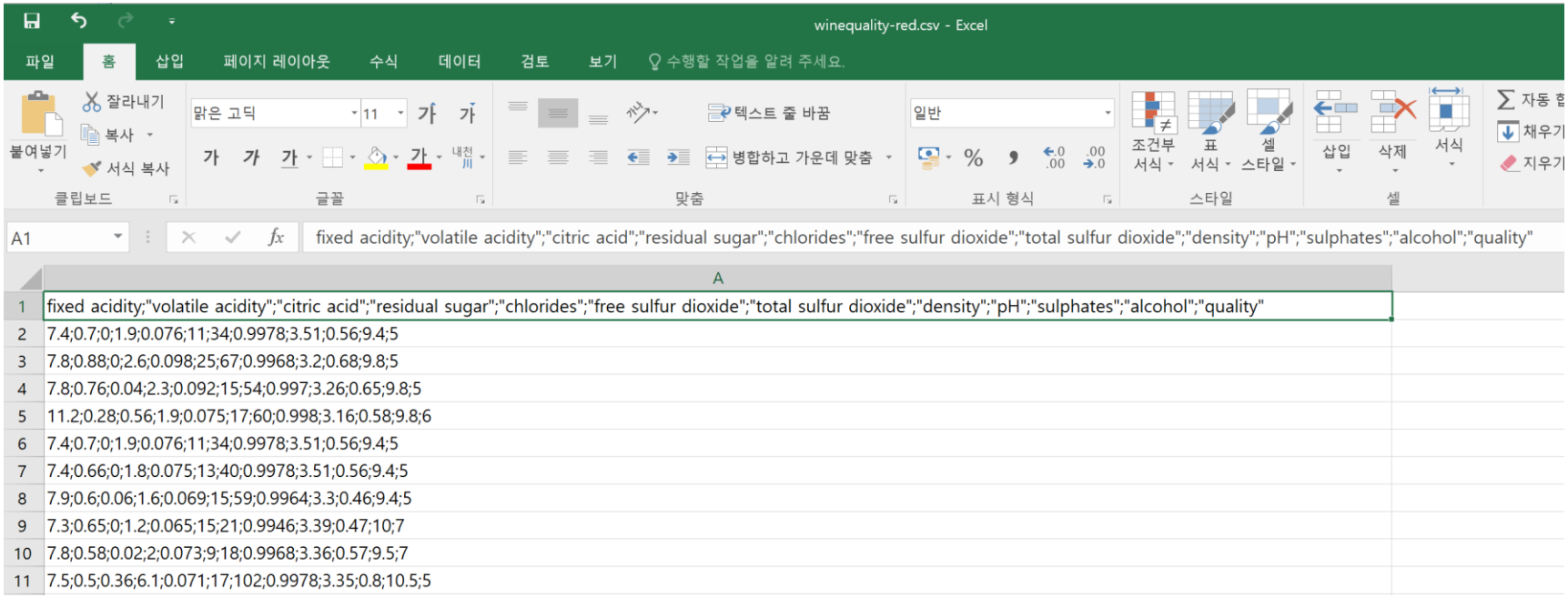
데이터 수집: 와인 데이터 셋

- 캘리포니아 어바인 대학의 머신러닝 저장소에서 제공하는 오픈 데이터를 사용
- http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/



데이터 준비

- 제공받은 데이터: ‘;’으로 열 구분(일반 엑셀은 ‘,’구분)
- Pandas로 구현
  - fixed_acidity(고정산), volatile acidity(휘발산), critric acid(구연산), residual sugar(잔당), chlorides(염화물), free sulfur dioxide(유리이산화황), total sulfur dioxide(총 이산화황), density(밀도), pH, sulphates(황산염), alcohol



소스코드

```
import pandas as pd
red_df = pd.read_csv('winequality-red.csv',sep=';',header=0,engine='python')
white_df = pd.read_csv('winequality-white.csv',sep=';',header=0,engine='python')
red_df.to_csv('winequality-red2.csv',index=False)
white_df.to_csv('winequality-white2.csv',index=False)
```

fixed acidit	volatile aci	citric acid	residual su	chlorides	free sulfur	total sulfur	density	pH	sulphates	alcohol	quality
7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5
7.8	0.88	0	2.6	0.098	25	67	0.9968	3.2	0.68	9.8	5
7.8	0.76	0.04	2.3	0.092	15	54	0.997	3.26	0.65	9.8	5
11.2	0.28	0.56	1.9	0.075	17	60	0.998	3.16	0.58	9.8	6
7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5
7.4	0.66	0	1.8	0.075	13	40	0.9978	3.51	0.56	9.4	5
7.9	0.6	0.06	1.6	0.069	15	59	0.9964	3.3	0.46	9.4	5
7.3	0.65	0	1.2	0.065	15	21	0.9946	3.39	0.47	10	7
7.8	0.58	0.02	2	0.073	9	18	0.9968	3.36	0.57	9.5	7
7.5	0.5	0.36	6.1	0.071	17	102	0.9978	3.35	0.8	10.5	5
6.7	0.58	0.08	1.8	0.097	15	65	0.9959	3.28	0.54	9.2	5
7.5	0.5	0.36	6.1	0.071	17	102	0.9978	3.35	0.8	10.5	5
5.6	0.615	0	1.6	0.089	16	59	0.9943	3.58	0.52	9.9	5
7.8	0.61	0.29	1.6	0.114	9	29	0.9974	3.26	1.56	9.1	5
8.9	0.62	0.18	3.8	0.176	52	145	0.9986	3.16	0.88	9.2	5
8.9	0.62	0.19	3.9	0.17	51	148	0.9986	3.17	0.93	9.2	5
8.5	0.28	0.56	1.8	0.092	35	103	0.9969	3.3	0.75	10.5	7
8.1	0.56	0.28	1.7	0.368	16	56	0.9968	3.11	1.28	9.3	5
7.4	0.59	0.08	4.4	0.086	6	29	0.9974	3.38	0.5	9	4
7.9	0.32	0.51	1.8	0.341	17	56	0.9969	3.04	1.08	9.2	6

데이터 병합

- 레드 와인과 화이트 와인 파일 합치기
  - Type 열 추가 : red

```
red_df.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

```
red_df.insert(0,column='type',value='red')
red_df.head()
```

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	red	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	red	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	red	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	red	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	red	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

데이터 결합

- Type 추가: white
- pd.concat() 함수를 이용하여 두 데이터 결합

```
white_df.insert(0,column='type',value='white')
white_df.head()
```

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	white	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
1	white	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
2	white	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
3	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
4	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6

```
red_df.shape
```

```
(1599, 13)
```

```
white_df.shape
```

```
(4898, 13)
```

```
wine = pd.concat([red_df,white_df])
wine.shape
```

```
(6497, 13)
```

```
wine.to_csv('wine.csv',index=False)
```

데이터 탐색

- Info()함수를 이용하여 기본 정보를 확인
  - 샘플 수: 6,497개
  - 속성: 13개 열
  - 실수 타입(float64): 11개
  - 정수 타입(int64):1개
  - 객체 타입(object):1개
  - 독립 변수(x) : 12개
  - 종속 변수(y) : 1개(quality)

```
print(wine.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6497 entries, 0 to 4897
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   type                  6497 non-null   object
 1   fixed acidity         6497 non-null   float64
 2   volatile acidity      6497 non-null   float64
 3   citric acid           6497 non-null   float64
 4   residual sugar        6497 non-null   float64
 5   chlorides             6497 non-null   float64
 6   free sulfur dioxide   6497 non-null   float64
 7   total sulfur dioxide  6497 non-null   float64
 8   density               6497 non-null   float64
 9   pH                    6497 non-null   float64
 10  sulphates             6497 non-null   float64
 11  alcohol               6497 non-null   float64
 12  quality               6497 non-null   int64
dtypes: float64(11), int64(1), object(1)
memory usage: 710.6+ KB
None
```

기술 통계 구하기

- Pandas를 이용한 describe(),unique(),value_counts()
  - describe(): 속성별 개수, 평균, 표준편차, 최소값, 백분위수(25%, 50%, 75%), 최대값

```
wine.columns = wine.columns.str.replace(' ','_')
wine.describe()
```

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxide	density	pH	sulphates
count	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000
mean	7.215307	0.339666	0.318633	5.443235	0.056034	30.525319	115.744574	0.994697	3.218501	0.531268
std	1.296434	0.164636	0.145318	4.757804	0.035034	17.749400	56.521855	0.002999	0.160787	0.148806
min	3.800000	0.080000	0.000000	0.600000	0.009000	1.000000	6.000000	0.987110	2.720000	0.220000
25%	6.400000	0.230000	0.250000	1.800000	0.038000	17.000000	77.000000	0.992340	3.110000	0.430000
50%	7.000000	0.290000	0.310000	3.000000	0.047000	29.000000	118.000000	0.994890	3.210000	0.510000
75%	7.700000	0.400000	0.390000	8.100000	0.065000	41.000000	156.000000	0.996990	3.320000	0.600000
max	15.900000	1.580000	1.660000	65.800000	0.611000	289.000000	440.000000	1.038980	4.010000	2.000000

기술 통계 구하기

- unique()함수: quality 속성값 중에서 유일한 값을 출력
- value_counts()함수: quality 속성값에 대한 빈도수

```
sorted(wine.quality.unique())
```

```
[3, 4, 5, 6, 7, 8, 9]
```

```
wine.quality.value_counts()
```

```
6    2836
5    2138
7    1079
4     216
8     193
3      30
9       5
Name: quality, dtype: int64
```

데이터 모델링

- Describe() 함수로 그룹 비교하기
  - Type에 따라 그룹을 나눈 뒤, 종속 변수인 qualit에 describe()함수를 사용
  - 그룹별로 count, mean, std, min, 25%,50%,75%, max 비교

```
wine.groupby('type')['quality'].describe()
```

type	count	mean	std	min	25%	50%	75%	max
red	1599.0	5.636023	0.807569	3.0	5.0	6.0	6.0	8.0
white	4898.0	5.877909	0.885639	3.0	5.0	6.0	6.0	9.0

```
wine.groupby('type')['quality'].mean()
```

```
type
red      5.636023
white    5.877909
Name: quality, dtype: float64
```

```
wine.groupby('type')['quality'].std()
```

```
type
red      0.807569
white    0.885639
Name: quality, dtype: float64
```

```
wine.groupby('type')['quality'].agg(['mean','std'])
```

type	mean	std
red	5.636023	0.807569
white	5.877909	0.885639

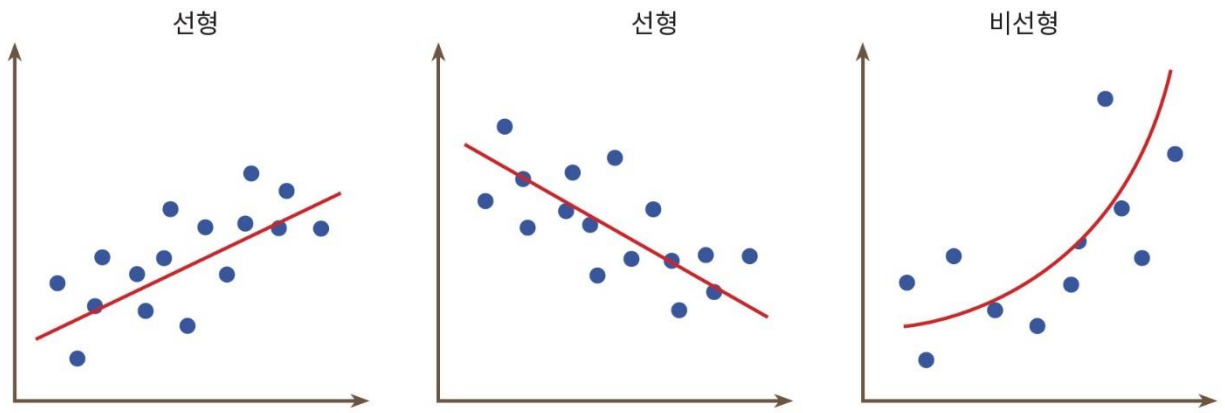
회귀 분석 모델

- 가장 간단하지만 아주 효과적인 모델
- 선형 회귀(linear regression)
- 예측할 결과값을 y라 하고 예측에서 사용하는 값을 $x_1, x_2, \ldots,$라 가정
- 선형 회귀 정의

- 수식 : $y = w_0 + w_1 x_1 + w_2 x_2 + \ldots$
- 출력 값(y)이 입력 값(피처)($x_1, x_2, \ldots$)에 대해 선형적인 관계

선형 회귀

- “회귀”란 일반적으로 데이터들을 2차원 공간에 찍은 후에 이들 데이터들을 가장 잘 설명하는 직선이나 곡선을 찾는 문제라고 할 수 있다.
- y = f(x)에서 출력 y가 실수이고 입력 x도 실수일 때 함수 f(x)를 예측하는 것이 회귀이다.



선형 회귀의 예

- 부모의 키와 자녀의 키의 관계 조사
- 면적에 따른 주택의 가격
- 연령에 따른 실업율 예측
- 공부 시간과 학점 과의 관계
- CPU 속도와 프로그램 실행 시간 예측

선형 회귀

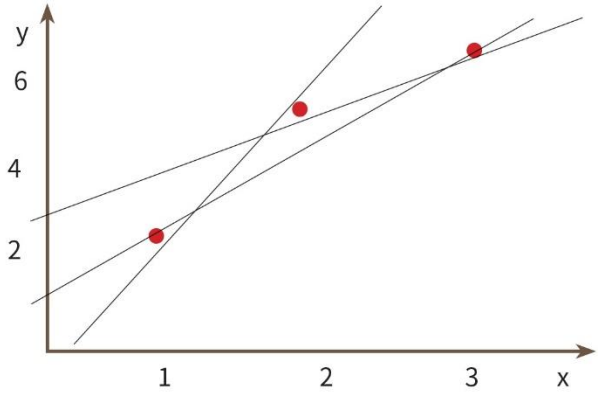
- 직선의 방정식: f(x) = mx+b
- 선형 회귀는 입력 데이터를 가장 잘 설명하는 기울기와 절편값을 찾는 문제이다

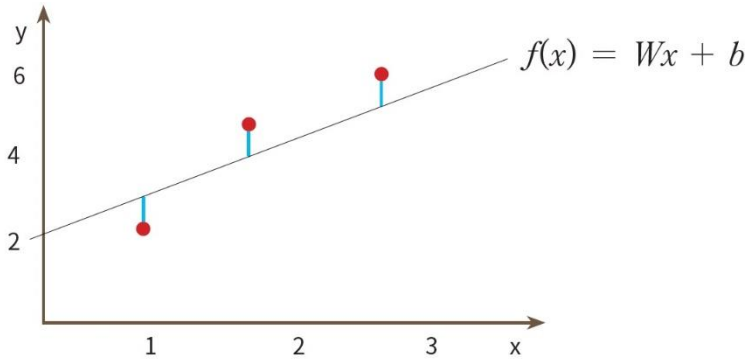
- 선형 회귀의 기본식: f(x) = Wx+b
  - 기울기->가중치
  - 절편->바이어스

선형 회귀의 원리

- 선형 관계

x	y
1	2
2	5
3	6





통계를 위한 라이브러리

- 회귀 분석을 위해서
  - statsmodels 라이브러리 패키지 사용
  - pip3 install statsmodels

  - 선형 회귀 모델 중에서 OLS(Ordinary Least Squares) 모델을 사용
    - statsmodels.formula.api 패키지의 ols() 함수
    - 첫번째 인자로 종속변수와 독립 변수를 구성한 변수를 주고 두 번째 인자는 실제 사용할 변수 값을 가진 데이터 프레임을 저장
    - 완성된 선형 모델은 fit()함수에 의해 실행되어 regression_result에 저장

회귀 분석

```
from statsmodels.formula.api import ols,glm
Rformula ='quality ~ fixed_acidity + volatile_acidity + citric_acid + residual_sugar + chlorides + free_sulfur_dioxide +
          total_sulfur_dioxide + density + pH + sulphates + alcohol'
regression_result = ols(Rformula, data=wine).fit()
regression_result.summary()
```

OLS Regression Results

Dep. Variable:	quality	**R-squared:**	0.292
Model:	OLS	**Adj. R-squared:**	0.291
Method:	Least Squares	**F-statistic:**	243.3
Date:	Mon, 31 May 2021	**Prob (F-statistic):**	0.00
Time:	18:59:51	**Log-Likelihood:**	-7215.5
No. Observations:	6497	**AIC:**	1.445e+04
Df Residuals:	6485	**BIC:**	1.454e+04
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	55.7627	11.894	4.688	0.000	32.447	79.079
fixed_acidity	0.0677	0.016	4.346	0.000	0.037	0.098
volatile_acidity	-1.3279	0.077	-17.162	0.000	-1.480	-1.176
citric_acid	-0.1097	0.080	-1.377	0.168	-0.266	0.046
residual_sugar	0.0436	0.005	8.449	0.000	0.033	0.054
chlorides	-0.4837	0.333	-1.454	0.146	-1.136	0.168
free_sulfur_dioxide	0.0060	0.001	7.948	0.000	0.004	0.007
total_sulfur_dioxide	-0.0025	0.000	-8.969	0.000	-0.003	-0.002
density	-54.9669	12.137	-4.529	0.000	-78.760	-31.173
pH	0.4393	0.090	4.861	0.000	0.262	0.616
sulphates	0.7683	0.076	10.092	0.000	0.619	0.917
alcohol	0.2670	0.017	15.963	0.000	0.234	0.300

결정계수: 1에 가까울수록 회귀 모델이 데이터에 잘 들어맞는 것이고, 0에 가까울 수록 회귀 모델이 데이터에 들어맞지 않는것

p-value가 0이면 통계적으로 유의하고, 그렇지 않으면 유의하지 않다

새로운 샘플의 품질 등급 예측

- 샘플 데이터: 독립 변수인 11개 속성에 대한 샘플 데이터 필요
  - 기존 wine 에서 quality와 type을 제외한 sample1
  - 임의의 값을 딕셔너리 형태로 만들어 sample 2

```
sample1 = wine[wine.columns.difference(['quality','type'])] #wine에서 qualtiy와 type 열은 제외
sample1 = sample1[0:5][:]
sample1_predict = regression_result.predict(sample1)
sample1_predict
```

```
0    4.997607
1    4.924993
2    5.034663
3    5.680333
4    4.997607
dtype: float64
```

```
wine[0:5]['quality']
```

```
0    5
1    5
2    5
3    6
4    5
Name: quality, dtype: int64
```

새로운 샘플의 품질 등급 예측

```
data = {"fixed_acidity":[8.5,8.1],"volatile_acidity":[0.8,0.5],"citric_acid":[0.3,0.4],"residual_sugar":[6.1,5.8],"chlorides":[0.055,0.04],
        "free_sulfur_dioxide":[30.0,31.0],"total_sulfur_dioxide":[98.0,99],"density":[0.996,0.91],"pH":[3.25,3.01],"sulphates":[0.4,0.35],
        "alcohol":[9.0,0.88]}
sample2 = pd.DataFrame(data,columns=sample1.columns)
sample2
```

	alcohol	chlorides	citric_acid	density	fixed_acidity	free_sulfur_dioxide	pH	residual_sugar	sulphates	total_sulfur_dioxide	volatile_acidity
0	9.00	0.055	0.3	0.996	8.5	30.0	3.25	6.1	0.40	98.0	0.8
1	0.88	0.040	0.4	0.910	8.1	31.0	3.01	5.8	0.35	99.0	0.5

```
sample2_predict = regression_result.predict(sample2)
sample2_predict
```

```
0    4.809094
1    7.582129
dtype: float64
```

결과 시각화

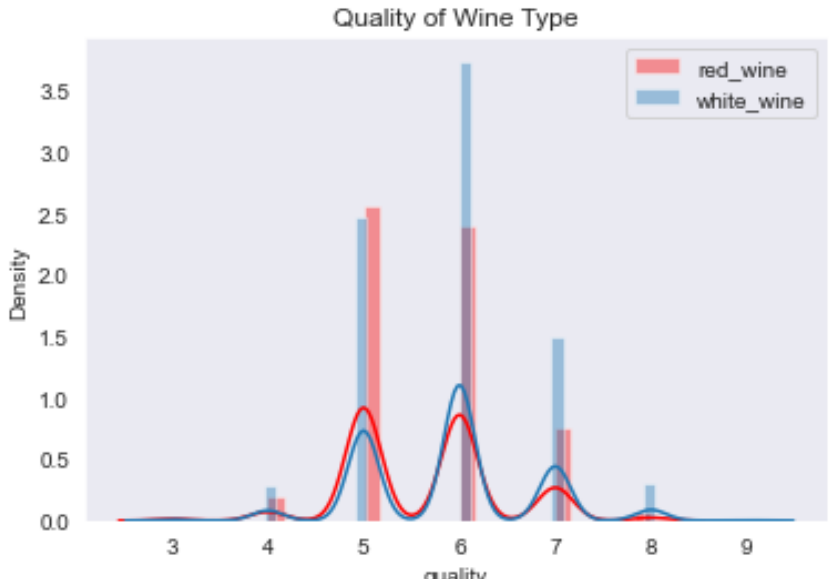
- 와인 유형에 따른 품질 등급을 히스토그램과 부분 회귀 플롯으로 나타냄
  - matplotlib.pyplot 라이브러리 패키지
  - seaborn 라이브러리 패키지
    - pip3 install seaborn

히스토그램 그리기

- kde: 커널 밀도 추정으로 곡선 형태로 나타남

```
import matplotlib.pyplot as plt
import seaborn as sns

sns.set_style('dark')
red_wine_quality = wine.loc[wine['type']=='red','quality']
white_wine_quality = wine.loc[wine['type']=='white','quality']
sns.distplot(red_wine_quality, kde=True,color='red', label='red_wine')
sns.distplot(white_wine_quality, kde=True, label='white_wine')
plt.title('Quality of Wine Type')
plt.legend()
plt.show()
```



부분 회귀 플롯

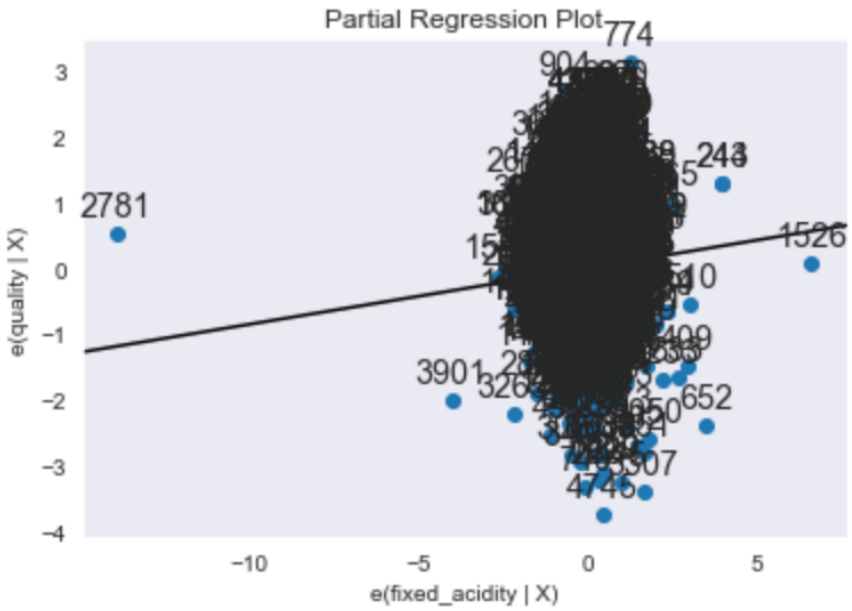
- 부분 회귀 플롯
  - 독립 변수가 2개 이상인 경우에는 부분 회귀 플롯(partial regression plot)을 사용하여 하나의 독립변수가 종속변수에 미치는 영향력을 시각화

부분회귀 플롯

- fixed_acidity가 종속 변수 quality에 미치는 영향력

```
import statsmodels.api as sm

others = list(set(wine.columns).difference(set(["quality","fixed_acidity"])))
p, resids = sm.graphics.plot_partregress("quality","fixed_acidity",others, data=wine,ret_coords=True)
plt.show()
fig=plt.figure(figsize=(8,13))
sm.graphics.plot_partregress_grdi(regression_result,fig=fig)
plt.show()
```



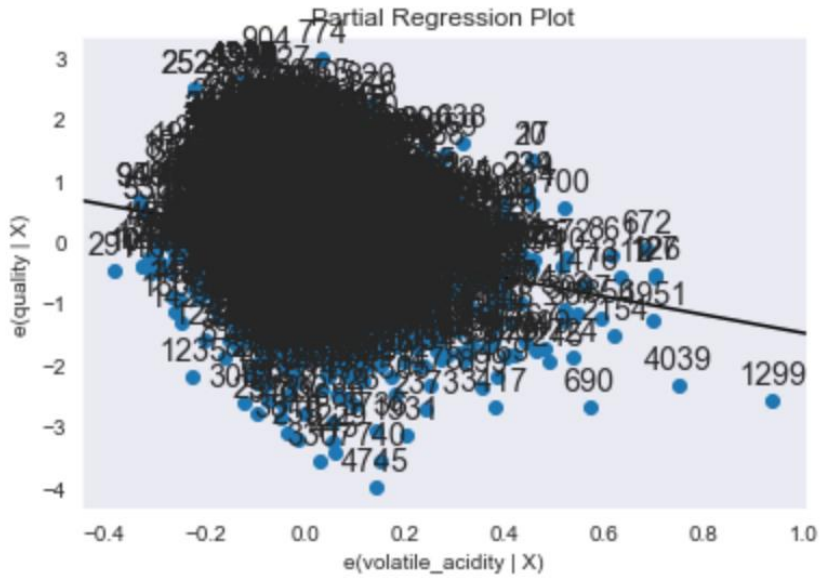
ret_cords: 잔차 데이터의 반환 여부

부분회귀 플롯

- volatile_acidity가 종속 변수 quality에 미치는 영향력

```
import statsmodels.api as sm

others = list(set(wine.columns).difference(set(["quality","volatile_acidity"])))
p, resids = sm.graphics.plot_partregress("quality","volatile_acidity",others, data=wine,ret_coords=True)
plt.show()
fig=plt.figure(figsize=(8,13))
sm.graphics.plot_partregress_grdi(regression_result,fig=fig)
plt.show()
```



상관 분석

- 상관 분석(correlation analysis)
  - 두 변수가 어떤 선형적 관계에 있는지를 분석하는 방법
  - 인과 관계를 분석하는 회귀 분석과 달리 상관계수는 두 변수가 연관된 정도를 나타낼 뿐 인과 관계를 설명하지 않음
  - 단순 상관 분석 : 두 변수의 관계 측정
  - 다중 상관 분석 : 세 개 이상의 변수 간 관계를 측정
  - 상관 계수(p)
    - 관계의 정도(0~1)와 방향(+,-)을 하나의 수치로 요약해주는 지수로 -1에서 +1 사이의 값을 가짐
    - 상관계수가 +이면 야의 상관관계이며 한 변수가 증가하면 다른 변수도 증가
    - 상관계수가 –이면 음의 상관관계이며 한 변수가 증가할 때 다른 변수는 감소한다.

샘플 데이터

```
import pandas as pd
import numpy as np
student = pd.read_csv('student.csv')
student.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   language  20 non-null     int64
 1   english   20 non-null     int64
 2   math      20 non-null     int64
 3   science   20 non-null     int64
dtypes: int64(4)
memory usage: 768.0 bytes
```

language	english	math	science
55	91	33	51
60	85	52	43
45	89	21	18
75	30	95	98
60	88	40	31
60	83	51	47
63	63	62	65
54	49	58	60
53	93	31	42
28	37	89	97
62	88	40	37
73	28	89	98
56	48	58	60
64	61	62	61
62	95	34	47
54	97	28	34
60	88	43	34
44	51	75	78
76	28	100	100
75	87	36	24

피어슨 상관계수

- 상관 분석은 pandas의 데이터프레임에서 corr()함수 사용
- 상관계수는 피어슨 상관 계수 사용

```
student.corr(method='pearson')
```

	language	english	math	science
language	1.000000	-0.125833	0.183212	0.090698
english	-0.125833	1.000000	-0.945780	-0.919824
math	0.183212	-0.945780	1.000000	0.951689
science	0.090698	-0.919824	0.951689	1.000000

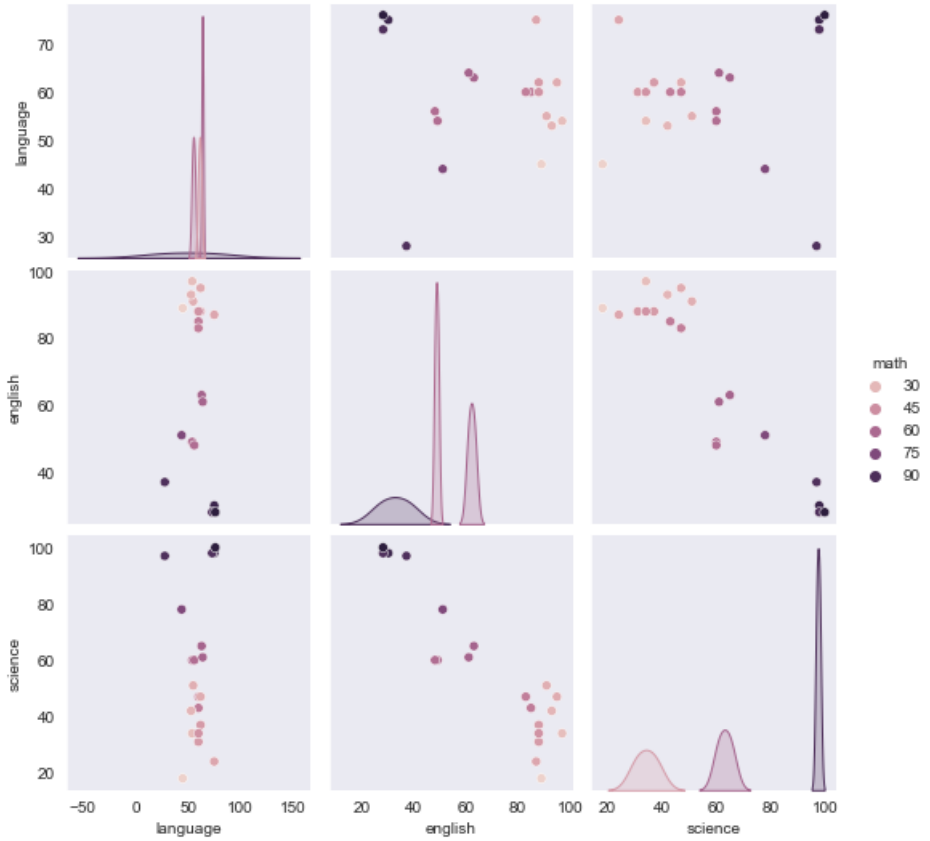
```
student['math'].corr(student['science'])
```

```
0.9516891105367318
```

산점도로 상관 분석 시각화하기

- Seaborn 패키지의 pairplot 함수 사용
  - 그리드 형태로 데이터프레임에 있는 각 데이터 열의 조합을 산점도로 그림

```
sns.pairplot(student, hue='math')
plt.show()
```

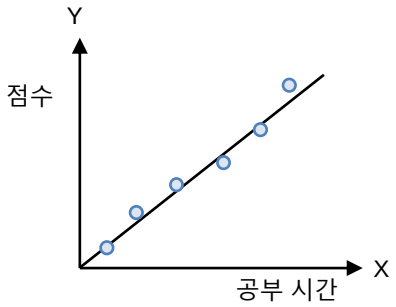


로지스틱 회귀 분석

- 목표
  - 유방암 특징을 측정한 데이터로 로지스틱 회귀 분석을 적용하여 유방암 발생을 예측
- 로지스틱 회귀
  - 분류에 사용하는 기법으로 선형 회귀와 달리 S자 함수를 사용하여 참(True, 1)과 거짓(False, 0)을 분류
- 시그모이드 함수
  - 로지스틱 회귀에서 사용하는 S자 함수
  - x의 값이 커지면 y의 값은 1에 근사하게 되고 x의 값이 작아지면 y의 값은 0에 근사하게 되어 S자 형태의 그래프가 됨
  - 두 개의 값을 분류하는 이진 분류에 많이 사용

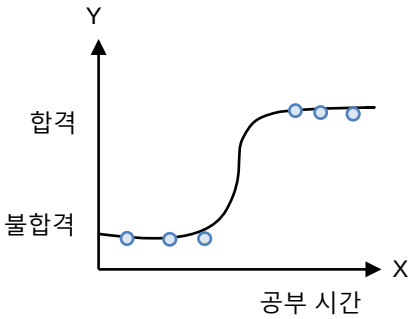
- **방정식** $y = \frac{1}{1+e^{ax+b}}$

공부시간	1	3	5	7	9	11
점수	40	55	65	70	80	95



선형 회귀와 선형 함수

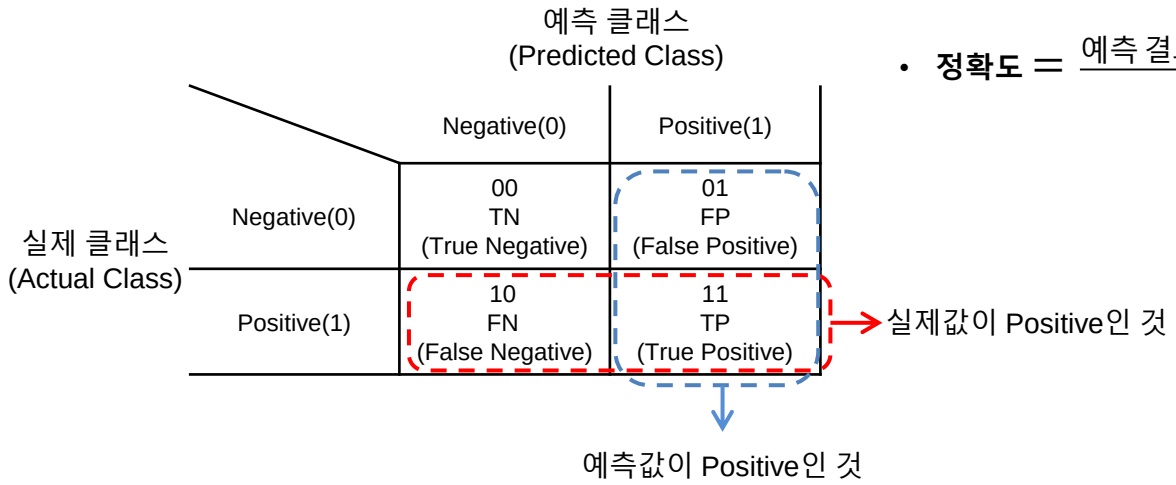
공부시간	1	3	5	7	9	11
점수	불합격	불합격	불합격	합격	합격	합격



로지스틱 회귀와 S자 함수

로지스틱 회귀 분석

- 로지스틱 회귀
  - 선형 회귀 모델은 실제값과 예측값의 오차에 기반한 지표를 사용
  - 로지스틱 회귀 모델은 이진 분류 결과를 평가하기 위해 오차 행렬에 기반한 성능 지표인 정밀도, 재현율, F1 스코어, ROC_AUC를 사용
- 오차 행렬
  - 행렬을 사용해 이진 분류의 예측 오류를 나타내는 지표
  - 행은 실제 클래스의 Negative/Positive 값, 열은 예측 클래스의 Negative/Positive 값
    TN: Negative가 참인 경우 TP: Positive가 참인 경우
    FN: Negative가 거짓인 경우 FP: Positive가 거짓인 경우
    사이킷런에서는 오차 행렬을 구하기 위해 confusion_matrix 함수를 제공



- **정확도** $= \frac{\text{예측 결과와 실제값이 동일한 건수}}{\text{전체 데이터 수}} = \frac{(TN+TP)}{(TN+FP+FN+TP)}$

로지스틱 회귀 분석

- 정밀도
  - 예측이 Positive인 것 (FP+TP) 중에서, 참인 것(TP)의 비율을 의미
  - 정밀도는 Positive 예측 성능을 더 정밀하게 평가하기 위한 지표로 사용
  - 사이킷런에서는 정밀도를 구하기 위해 precision_score 함수를 제공

    - **정밀도** $= \frac{TP}{(FP+TP)}$

- 재현율
  - 실제 값이 Positive인 것(FN+TP) 중에서 참인 것(TP)의 비율을 의미
  - 실제 Positive인 데이터를 정확하게 예측했는지 평가하는 지표 (민감도 또는 쎆)
  - 사이킷런에서는 재현율을 구하기 위해 recall_score함수를 제공

    - **재현율** $= \frac{TP}{(FN+TP)}$

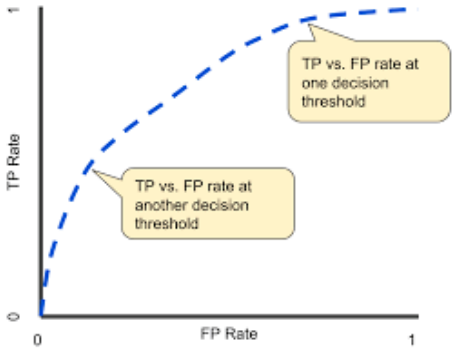
로지스틱 회귀 분석

- F1 스코어
  - 정밀도와 재현율을 결합한 평가 지표
  - 정밀도와 재현율이 서로 트레이드 오프 관계(상충 관계)인 문제점을 고려하여 정확한 평가를 위해 많이 사용
  - 사이킷런에서는 F1 스코어를 구하기 위해 f1_score 함수를 제공

  - **F1 스코어** $= \frac{2}{\frac{1}{\text{재현율}} + \frac{1}{\text{정밀도}}} = 2 \times \frac{\text{정밀도} \times \text{재현율}}{\text{정밀도} + \text{재현율}}$

- ROC 기반 AUC 스코어
  - 오차 행렬의 FPR이 변할 때 TPR이 어떻게 변하는지를 나타내는 곡선
    FPR: 실제 Negative인 데이터를 Positive로 거짓False으로 예측한 비율
    TPR: 실제 Positive인 데이터를 참True으로 예측한 비율(재현율)

    - **FPR** $= \frac{FP}{(FP+TN)}$



  - ROC 기반의 AUC값은 ROC 곡선 밑의 면적을 구한 것으로 1에 가까울수록 좋은 성능을 의미
  - 사이킷런에서는 ROC 기반의 AUC를 구하기 위해 roc_auc_score 함수를 제공

로지스틱 회귀 분석

- 데이터 준비
  - 사이킷런에서 제공하는 데이터 셋

데이터셋	샘플 갯수	독립 변수	종속 변수	데이터 로드 함수
보스톤 주택 가격 데이터	506	13개	주택 가격	load_boston()
붓꽃(아이리스) 데이터	150	4개	붓꽃 종류:setosa, versicolor, virginica	load_iris()
당뇨병 환자 데이터	442	10개	당뇨병 수치	load_diabetes()
숫자 0~9를 손으로 쓴 흑백 데이터	1797	64개	숫자: 0~9	load_digits()
와인의 화학 성분 데이터	178	13개	와인 종류: 0, 1, 2	load_wine()
체력 검사 데이터	20	3개	체력 검사 점수	load_linnerud()
유방암 진단 데이터	569	30개	악성(malignant), 양성(benign): 1, 0	load_breast_cancer()

로지스틱 회귀 분석

- 데이터 준비
  - 사이킷런의 유방암 진단 데이터셋 사용하기

  1. 데이터 준비하기

```
[1]  1 # sklearn 설치
     2 !pip install sklearn
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting sklearn
  Downloading sklearn-0.0.tar.gz (1.1 kB)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (from sklearn) (1.(
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from scik
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.7/dist-packages (from scikit-learn
```

```
[2]  1 #사이킷런에서 제공하는 데이터셋sklearn.datasets중에서
     2 #유방암 진단 데이터셋을 사용하기 위해 load_breast_cancer를 import
     3 import numpy as np
     4 import pandas as pd
     5
     6 from sklearn.datasets import load_breast_cancer
```

```
[3]  1 #데이터셋을 로드하여 객체b_cancer를 생성
     2 b_cancer = load_breast_cancer()
```

로지스틱 회귀 분석

- 데이터 탐색

  2. 데이터 탐색하기

[4]
```
1 #데이터셋에 대한 설명을 확인
2 print(b_cancer.DESCR)
```

[5]
```
1 #데이터셋 객체의 data 배열b_cancer.data, 즉 독립 변수 X가 되는 피처를
2 #DataFrame 자료형으로 변환하여 b_cancer_df를 생성
3 b_cancer_df = pd.DataFrame(b_cancer.data, columns = b_cancer.feature_names)
```

[6]
```
1 #유방암 유무 class로 사용할 diagnosis 컬럼을 b_cancer_df에 추가하고
2 #데이터셋 객체의 target 컬럼b_cancer.target을 저장
3 b_cancer_df['diagnosis']= b_cancer.target
```

[7]
```
1 #b_cancer_df의 데이터 샘플 5개를 출력b_cancer_df.head( )하여 확인
2 b_cancer_df.head()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430

5 rows × 31 columns

로지스틱 회귀 분석

- 데이터 탐색

  3. 데이터셋의 크기와 독립 변수 X가 되는 피처에 대한 정보를 확인

[8]
```
#b_cancer_df.shape를 사용하여 데이터셋의 행의 개수(데이터 샘플 개수)와
#열의 개수(변수 개수)를 확인 행의 개수가 5690이므로 데이터샘플이 569개,
#열의 개수가 310이므로 변수가 31개 있음
print('유방암 진단 데이터셋 크기: ', b_cancer_df.shape)
```

```
유방암 진단 데이터셋 크기:  (569, 31)
```

```
#b_cancer_df에 대한 정보를 확인b_cancer_df.info( ) /  30개의 피처(독립 변수 X)
#이름과 1개의 종속 변수 이름을 확인 가능 diagnosis는 악성이면 1, 양성이면 0의 값이므로
#유방암 여부에 대한 이진 분류의 class로 사용할 종속 변수가 됨
b_cancer_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   mean radius             569 non-null    float64
 1   mean texture            569 non-null    float64
 2   mean perimeter          569 non-null    float64
 3   mean area               569 non-null    float64
 4   mean smoothness         569 non-null    float64
 5   mean compactness        569 non-null    float64
```

로지스틱 회귀 분석

- 데이터 탐색

  4. 로지스틱 회귀 분석에 피처로 사용할 데이터를 평균이 0, 분산이 1이 되는 정규 분포 형태로 맞춤

```
[10]  1 #사이킷런의 전처리 패키지에 있는 정규 분포 스케일러를 임포트하고 사용할 객체scaler를 생성
      2 from sklearn.preprocessing import StandardScaler
      3 scaler = StandardScaler()
```

```
[11]  1 #피처로 사용할 데이터b_cancer.data에 대해 정규 분포 스케일링을 수행
      2 #scaler.fit_ transform( )하여 b_cancer_scaled에 저장
      3 b_cancer_scaled = scaler.fit_transform(b_cancer.data)
```

```
[12]  1 #정규 분포 스케일링 후에 값이 조정된 것을 확인
      2 print(b_cancer.data[0])
```

```
[1.799e+01 1.038e+01 1.228e+02 1.001e+03 1.184e-01 2.776e-01 3.001e-01
 1.471e-01 2.419e-01 7.871e-02 1.095e+00 9.053e-01 8.589e+00 1.534e+02
 6.399e-03 4.904e-02 5.373e-02 1.587e-02 3.003e-02 6.193e-03 2.538e+01
 1.733e+01 1.846e+02 2.019e+03 1.622e-01 6.656e-01 7.119e-01 2.654e-01
 4.601e-01 1.189e-01]
```

```
[13]  1 print(b_cancer_scaled[0])
```

```
[ 1.09706398 -2.07333501  1.26993369  0.9843749   1.56846633  3.28351467
  2.65287398  2.53247522  2.21751501  2.25574689  2.48973393 -0.56526506
  2.83303087  2.48757756 -0.21400165  1.31686157  0.72402616  0.66081994
  1.14875667  0.90708308  1.88668963 -1.35929347  2.30360062  2.00123749
  1.30768627  2.61666502  2.10952635  2.29607613  2.75062224  1.93701461]
```

로지스틱 회귀 분석

- 분석 모델 구축
  1. 로지스틱 회귀를 이용해 분석 모델 구축

[14]
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

[15]
```
#diagnosis를 Y, 정규 분포로 스케일링한 b_cancer_scaled를 X로 설정
Y = b_cancer_df['diagnosis']
X = b_cancer_scaled
```

[16]
```
#전체 데이터 샘플 569개를 학습 데이터:평가 데이터=7:3으로 분할test_size=0.3함
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.3, random_state = 0)
```

[17]
```
#로지스틱 회귀 분석 모델 객체lr_b_cancer를 생성
lr_b_cancer = LogisticRegression()
```

[18]
```
#학습 데이터X_train, Y_train로 모델 학습을 수행fit( )함
lr_b_cancer.fit(X_train, Y_train)
```

```
LogisticRegression()
```

[19]
```
#학습이 끝난 모델에 대해 평가 데이터 X X_test를 가지고 예측을 수행
#predict( )하여 예측값 Y Y_predict를 구함
Y_predict = lr_b_cancer.predict(X_test)
```

로지스틱 회귀 분석

- 결과 분석

  2. 생성한 모델의 성능 확인하기

[20]
```
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score, roc_auc_score
```

[21]
```
#평가를 위해 7:3으로 분할한 171개의 test 데이터에 대해 이진 분류의 성능 평가 기본이
#되는 오차 행렬을 구함 실행 결과를 보면 TN이 60개, FP가 3개, FN이 1개, TP가 107개인 오차 행렬이 구해짐
confusion_matrix(Y_test, Y_predict)
```

```
array([[ 60,   3],
       [  1, 107]])
```

[22]
```
#성능 평가 지표인 정확도, 정밀도, 재현율, F1 스코어, ROC-AUC 스코어를 구함
acccuracy = accuracy_score(Y_test, Y_predict)
precision = precision_score(Y_test, Y_predict)
recall = recall_score(Y_test, Y_predict)
f1 = f1_score(Y_test, Y_predict)
roc_auc = roc_auc_score(Y_test, Y_predict)
```

[23]
```
print('정확도: {0:.3f}, 정밀도: {1:.3f}, 재현율: {2:.3f}, F1: {3:.3f}'.format(acccuracy,precision,recall,f1))
```

```
정확도: 0.977, 정밀도: 0.973, 재현율: 0.991, F1: 0.982
```

[24]
```
print('ROC_AUC: {0:.3f}'.format(roc_auc))
```

```
ROC_AUC: 0.972
```