

# 파이썬 실습



# 실습



## 개념과 생성

NumPy와 패키지의 핵심은 `ndarray` 객체이다.

- `ndarray` 는 fixed-size homogeneous multidimensional array 정도로 이해할 수 있으며, 기본적으로 vectorization과 broadcasting을 지원한다.
- Python에서 제공하는 `list`, `tuple` 등의 시퀀스 자료형은 서로 다른 데이터타입을 저장할 수 있고(heterogeneous sequence), 크기가 자동으로 커질 수 있다. 반면에 `ndarray` 는 성능향상을 위해 같은 데이터타입만을 요소로 가질 수 있고, 크기 역시 고정되어 있다. 만약 크기를 변경하면 새로 메모리에 할당되고 이전 값은 삭제된다.

`ndarray`는 `list` 나 `tuple` 같은 시퀀스 자료형으로 부터 생성한다.

```
>>> import numpy as np
>>> x = np.array((0.1,0.2,0.3))    # np.array([0.1,0.2,0.3])도 가능
>>> x
array([0.1, 0.2, 0.3])
>>> x.shape
(3,)
>>> x.dtype
dtype('float64')
>>> y = np.array(((1,2,3),(4,5,6)))    # [(1,2,3),(4,5,6)], [[1,2,3],[4,5,6]] 등도 가능
>>> y
array([[1, 2, 3],
       [4, 5, 6]])
>>> y.dtype
dtype('int32')
>>> y.shape
(2, 3)
```

- `x` 는 `float64` 를 요소 타입으로 갖는 크기 3의 1차원 배열이다.
- `y` 는 `int32` 을 요소 타입으로 하는 `(2,3)` 크기의 2차원 배열이다.
- 요소 타입을 `dtype` 멤버에서 확인할 수 있으며, 차원은 `shape` 에서 확인할 수 있다.
- 1차원 배열의 `shape` 은 `(m,)` 형태이고, 2차원 배열은 `(m,n)` 형태이다. 3차원은 `(p,q,r)` 등과 같다.
- 생성시 입력된 값을 통해 `dtype` 을 추정하는데, 강제로 지정하는 것은 다음과 같다.



```
z = np.array([1,2,3],dtype='float64')
```

`ndarray`의 중요 속성을 정리하면 다음과 같다.

- `shape` : 배열의 형태
- `dtype` : 요소의 데이터 타입, `int32`, `float32` 등등
- `ndim` : 차원수. `x.ndim = 1`, `y.ndim=2` 등이며 `len(x.shape)` 와 동일
- `size` : 요소의 개수. `shape` 의 모든 값의 곱. `x.size = 3`, `y.size=6` 등
- `itemsize` : 요소 데이터 타입의 크기(byte 단위), `x.itemsize=8` 등
- `data` : 실제 데이터. 직접 사용자가 접근할 필요는 없음

## 초기화관련 편의 함수

```
>>> Y = np.zeros((3,3))
>>> Y
array([[ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.]])
>>> Y = np.ones((3,3),dtype='int32')
>>> Z = np.empty((3,3))
```

## 초기화관련 편의 함수 - 미리 크기를 정하지 않고 순차적 생성

```
import numpy as np

A = np.array([])
for i in range(3):
    A = np.append(A, [1, 2, 3])

>>> A
array([ 1.,  2.,  3.,  1.,  2.,  3.,  1.,  2.,  3.] )
```

단순한 시퀀스는 `range()` 함수의 실수버전인 `arange(from,to,step)` 이나 `linspace(from,to,npoints)` 를 사용하면 편리하다.  
또한 단위행렬을 위한 `eye(n)` 함수를 제공한다.

```
>>> np.arange(1,2,0.1)
array([ 1. ,  1.1,  1.2,  1.3,  1.4,  1.5,  1.6,  1.7,  1.8,  1.9])
>>> np.arange(10)    # start, step 생략가능. 정수로 생성
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> np.arange(10.)   # start, step 생략가능. 실수로 생성
array([ 0.,  1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9.])
>>> np.linspace(0.,20.,11)
array([ 0.,  2.,  4., ..., 16., 18., 20.])
>>> np.eye(3)
array([[ 1.,  0.,  0.],
       [ 0.,  1.,  0.],
       [ 0.,  0.,  1.]])
```



## Shape과 dtype변경

```
>>> X = np.arange(0,9,1.)
>>> X
array([ 0.,  1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.])
>>> Y = np.reshape(X,(3,3))    # 또는 Y=X.reshape((3,3))
>>> Y
array([[ 0.,  1.,  2.],
       [ 3.,  4.,  5.],
       [ 6.,  7.,  8.]])
```

```
>>> X.shape = (3,3)
>>> X
array([[ 0.,  1.,  2.],
       [ 3.,  4.,  5.],
       [ 6.,  7.,  8.]])
```

```
>>> a = np.arange(3);
>>> a.astype(int)    # a.astype('int32') 와 동일
>>> a.astype('int32')
>>> a.astype('int64')
>>> a.astype(float)  # a.astype('float64')
>>> a.astype('float32')
>>> a.astype('float64')
```

## 인덱싱

`ndarray` 에서 인덱싱하는 것은 `A[2]`, `A[2,3]` 등과 같은 `[]` 연산자를 사용한다. `A[1:3,1:2]` 등과 같이 `:`를 이용한 범위 지정을 통해 부분 배열을 얻어낼 수 있다. `A[-1]` 등과 같이 음의 정수를 사용하면 뒤에서부터 인덱싱 된다.

```
>>> a = np.array([1.2, -1.3, 2.2, 5.3, 3.7])
>>> a
array([ 1.2, -1.3,  2.2,  5.3,  3.7])
>>> a[0]
1.2
>>> a[0:3]
array([ 1.2, -1.3,  2.2])
>>> a[-1]
3.7000000000000002
>>> a[-2]
5.2999999999999998
>>> a[0:-1]
array([ 1.2, -1.3,  2.2,  5.3])
```

```
>>> a = np.array([1.2, -1.3, 2.2, 5.3, 3.7])
>>> idx = [0, 3]
>>> a[idx]
array([ 1.2,  5.3])
```

## 행렬 합치기

```
>>> a = np.array([[1, 2], [3, 4]])
>>> b = np.array([[5, 6]])
>>> np.concatenate((a, b), axis=0)
array([[1, 2],
       [3, 4],
       [5, 6]])
>>> np.concatenate((a, b.T), axis=1)
array([[1, 2, 5],
       [3, 4, 6]])
```

```
>>> a = np.array([1,2,3])
>>> b = np.array([4,5,6])
>>> np.vstack((a,b))
array([[1, 2, 3],
       [4, 5, 6]])
>>> np.hstack((a,b))
array([1, 2, 3, 4, 5, 6])
>>> c = np.append(a,b)
array([1, 2, 3, 4, 5, 6])
```

`concatenate((A,B,...),axis=0)` 는 axis 방향으로 합치라는 의미이다(axis=0은 row, axis=2는 column 방향 등) 위에서 b가 (1,2) 2-d 배열이다. 만약 b가 (2,)의 1-d 배열이면 에러가 발생한다. 만약 2차원 행렬을 대상으로 한다면 `hstack((A,B,...))` 과 `vstack((A,B,...))` 이 보다 편리하다. 이 둘 함수는 (n,) 형태의 1D 배열을 (1,n) 배열로 취급한다. `np.append(obj,val)` 을 사용해도 된다.

## 연산

```
>>> a = np.array([1,2,3,4])
>>> b = np.array([4,5,6,7])
>>> a+b
array([ 5,  7,  9, 11])
>>> a*b
array([ 4, 10, 18, 28])
>>> a**2
array([ 1,  4,  9, 16])
>>> a+2 # 상수 2는 같은 크기의 배열로 인식
array([3, 4, 5, 6])
>>> 10*np.sin(a) # NumPy의 universal 함수 sin() 적용
array([ 8.41470985,  9.09297427,  1.41120008, -7.56802495])
>>> a<3
array([ True,  True, False, False], dtype=bool)
>>> a *= b # a = a*b와 동일
>>> a
array([ 4, 10, 18, 28])
```

## 브로드캐스팅

Numpy에서 차원이 맞지 않은 객체끼리 연산

```
>>> import numpy as np
>>> A = np.arange(9.).reshape(3,3)    # 2d array : (3,3)
>>> x = np.array([1.,0,0])           # 1d array : (3,)
>>> y = x.reshape(1,3)               # 2d array : (1,3)
>>> z = x.reshape(3,1)               # 2d array : (3,1)
.....
>>> A+1    # (3,3) + scalar ==> (3,3) + scalar*I
array([[ 1.,  2.,  3.],
       [ 4.,  5.,  6.],
       [ 7.,  8.,  9.]])
.....
>>> x
array([ 1.,  0.,  0.])
>>> y
array([[ 1.,  0.,  0.]])
>>> z
array([[ 1.],
       [ 0.],
       [ 0.]])
```



## 브로드캐스팅

Numpy에서 차원이 맞지 않은 객체끼리 연산

```
>>> A+1    # (3,3) + scalar ==> (3,3) + scalar*I  
array([[ 1.,  2.,  3.],  
       [ 4.,  5.,  6.],  
       [ 7.,  8.,  9.]])
```

## 브로드캐스팅

### 1차원 배열 x 를 확장

```
# (3,3) + (3,) ==> (3,3)
# x = [1,0,0] --> expand with (3,3) 1 0 0
#
#                                1 0 0
#                                1 0 0

>>> A+x
array([[ 1.,  1.,  2.],
       [ 4.,  4.,  5.],
       [ 7.,  7.,  8.]])
```

```
# (3,3) + (1,3) ==> (3,3)
# y = [[1,0,0]] --> expand with (3,3) 1 0 0
#
#                                1 0 0
#                                1 0 0

>>> A+y
array([[ 1.,  1.,  2.],
       [ 4.,  4.,  5.],
       [ 7.,  7.,  8.]])
```

## 브로드캐스팅

```
# (3,3) + (3,1)
# z = 1 --> expand with (3,3) 1 1 1
#      0                0 0 0
#      0                0 0 0

>>> A+z
array([[ 1.,  2.,  3.],
       [ 3.,  4.,  5.],
       [ 6.,  7.,  8.]])
```

```
# (1,3) + (3,1) --> each dimensions are expanded
#
# y = [1,0,0] --> (3,3) 1 0 0      z = 1      1 1 1
#                  1 0 0          0      0 0 0
#                  1 0 0          0      0 0 0

>>> y+z
array([[ 2.,  1.,  1.],
       [ 1.,  0.,  0.],
       [ 1.,  0.,  0.]])
```

```
import pandas as pd

df = pd.read_csv("train.csv")
df
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

## df.loc[]

Access a group of rows and columns by label(s) or a boolean array.

```
df.loc[행 인덱싱 값, 열 인덱싱 값]
```

`loc` 은 location의 약자이다.

데이터 프레임 행/열의 라벨을 통해 가져오는 방법이다.

쉽게 생각해 칼럼 '이름' 같은 것으로 생각하면 될 것 같다.

```
df.loc[0]
```

PassengerId	1
Survived	0
Pclass	3
Name	Braund, Mr. Owen Harris
Sex	male
Age	22.0
SibSp	1
Parch	0
Ticket	A/5 21171
Fare	7.25
Cabin	NaN
Embarked	S

Name: 0, dtype: object



- (참고로 `df.loc[[0]]` 으로 하면 이렇게 볼 수도 있다)

```
df.loc[[0]]
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S

두 번째 row의 Name 이 알고 싶다면?

- 

```
df.loc[1, 'Name']
```

```
'Cumings, Mrs. John Bradley (Florence Briggs Thayer)'
```

## 슬라이싱을 통해 여러 값 가져오기

```
df.loc[:, 'Name']
```

```
0          Braund, Mr. Owen Harris
1  Cumings, Mrs. John Bradley (Florence Briggs Th...
2          Heikkinen, Miss. Laina
3  Futrelle, Mrs. Jacques Heath (Lily May Peel)
4          Allen, Mr. William Henry
...
886          Montvila, Rev. Juozas
887          Graham, Miss. Margaret Edith
888  Johnston, Miss. Catherine Helen "Carrie"
889          Behr, Mr. Karl Howell
890          Dooley, Mr. Patrick
Name: Name, Length: 891, dtype: object
```

```
df.loc[:, : 'Name']
```



	PassengerId	Survived	Pclass	Name
0	1	0	3	Braund, Mr. Owen Harris
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...
2	3	1	3	Heikkinen, Miss. Laina
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)
4	5	0	3	Allen, Mr. William Henry
...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas
887	888	1	1	Graham, Miss. Margaret Edith
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"
889	890	1	1	Behr, Mr. Karl Howell
890	891	0	3	Dooley, Mr. Patrick

891 rows × 4 columns

```
df.loc[:4, : 'Name']
```



	PassengerId	Survived	Pclass	Name
0	1	0	3	Braund, Mr. Owen Harris
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...
2	3	1	3	Heikkinen, Miss. Laina
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)
4	5	0	3	Allen, Mr. William Henry



```
condition = (df['Pclass'] == 3) & (df['Survived'] == 1)
df.loc[condition]
```



PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
10	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.7000	G6	S
19	20	1	3	Masselmani, Mrs. Fatima	female	NaN	0	0	2649	7.2250	NaN	C
22	23	1	3	McGowan, Miss. Anna "Annie"	female	15.0	0	0	330923	8.0292	NaN	Q
...	...	...	...	...	...	...	...	...	...	...	...	...
838	839	1	3	Chip, Mr. Chang	male	32.0	0	0	1601	56.4958	NaN	S
855	856	1	3	Aks, Mrs. Sam (Leah Rosen)	female	18.0	0	1	392091	9.3500	NaN	S
858	859	1	3	Baclini, Mrs. Solomon (Latifa Qurban)	female	24.0	0	3	2666	19.2583	NaN	C
869	870	1	3	Johnson, Master. Harold Theodor	male	4.0	1	1	347742	11.1333	NaN	S
875	876	1	3	Najib, Miss. Adele Klamie "Jane"	female	15.0	0	0	2667	7.2250	NaN	C

119 rows x 12 columns

## df.iloc[]

Purely integer-location based indexing for selection by position.

`iloc` 은 Integer location의 약자이다.

데이터 프레임 행/열의 순서를 나타내는 정수를 통해 가져오는 방법이다. `df.loc[]` 이 라벨을 사용한다면 `df.iloc[]` 은 각 행렬의 순번을 사용하는 차이가 있다.

## 첫번째 ROW 추출하기

```
df.iloc[0]
```

PassengerId	1
Survived	0
Pclass	3
Name	Braund, Mr. Owen Harris
Sex	male
Age	22.0
SibSp	1
Parch	0
Ticket	A/5 21171
Fare	7.25
Cabin	NaN
Embarked	S

Name: 0, dtype: object

## 두번째 Row의 Name

```
df.iloc[1, 3]
```

```
'Cumings, Mrs. John Bradley (Florence Briggs Thayer)'
```

```
df.iloc[:, 3]
```



```
0          Braund, Mr. Owen Harris
1  Cumings, Mrs. John Bradley (Florence Briggs Th...
2          Heikkinen, Miss. Laina
3  Futrelle, Mrs. Jacques Heath (Lily May Peel)
4          Allen, Mr. William Henry
...
886          Montvila, Rev. Juozas
887          Graham, Miss. Margaret Edith
888  Johnston, Miss. Catherine Helen "Carrie"
889          Behr, Mr. Karl Howell
890          Dooley, Mr. Patrick
Name: Name, Length: 891, dtype: object
```



```
df.iloc[:, :3]
```

	PassengerId	Survived	Pclass
0	1	0	3
1	2	1	1
2	3	1	3
3	4	1	1
4	5	0	3
...	...	...	...

```
df.iloc[:3, :3]
```

	PassengerId	Survived	Pclass
0	1	0	3
1	2	1	1
2	3	1	3

바로 인덱싱하기 df['column명']

```
df[ 'Name' ]
```

```
0          Braund, Mr. Owen Harris
1  Cumings, Mrs. John Bradley (Florence Briggs Th...
2          Heikkinen, Miss. Laina
3  Futrelle, Mrs. Jacques Heath (Lily May Peel)
4          Allen, Mr. William Henry
...
886          Montvila, Rev. Juozas
887          Graham, Miss. Margaret Edith
888  Johnston, Miss. Catherine Helen "Carrie"
889          Behr, Mr. Karl Howell
890          Dooley, Mr. Patrick
Name: Name, Length: 891, dtype: object
```

`df[ ['a column', 'b comumn'] ]`

- 참고로 리스트 슬라이싱을 할 때 칼럼은 안됨
- row에 대한 슬라이싱은 `df['a':'d']`
- 대신 `loc`, `iloc`을 주로 사용

```
df[['Pclass', 'Name']]
```

	Pclass	Name
0	3	Braund, Mr. Owen Harris
1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...
2	3	Heikkinen, Miss. Laina
3	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)
4	3	Allen, Mr. William Henry
...	...	...
886	2	Montvila, Rev. Juozas
887	1	Graham, Miss. Margaret Edith
888	3	Johnston, Miss. Catherine Helen "Carrie"
889	1	Behr, Mr. Karl Howell
890	3	Dooley, Mr. Patrick

891 rows × 2 columns

```
# example 1
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0,1,50)

y1 = np.cos(4*np.pi*x)
y2 = np.cos(4*np.pi*x)*np.exp(-2*x)

plt.plot(x,y1)
plt.plot(x,y2)

plt.show()
```

# matplotlib 실습

```
# example 2
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0,1,50)

y1 = np.cos(4*np.pi*x)
y2 = np.cos(4*np.pi*x)*np.exp(-2*x)

plt.plot(x,y1,'r-*',
         label=r'$sin(4 \pi x)$',lw=1)
plt.plot(x,y2,'b--o',
         label=r'$ e^{-2x} sin(4\pi x) $',lw=1)
plt.title(r'$sin(4 \pi x)$ vs. $ e^{-2x} sin(4\pi x)$')
plt.xlabel('x')
plt.ylabel('y')
plt.text(0.5,-1.0,r'This is sample')
plt.axis([0,1,-1.5,1.5])
plt.grid(True)
plt.legend(loc='upper left')
plt.tight_layout()
plt.show()
```

- `plt.plot(x,y,'r-*',label='sin',lw=1)` 에서 `'r-*'` 은 문자열은 색상, 선타입, 마커 등의 속성을 나타낸다. `'r-*'` 는 red, solid line, \* 마커를 의미한다. `label` 은 레전드에 표시될 내용이고, `lw` 는 line width이다.
- `plt.title(title)` , `plt.xlabel(xlabel)` , `plt.ylabel(ylabel)` 은 제목, X와 Y축 제목을 나타낸다.
- `plt.text(x,y,text)` 로 (x,y) 위치에 text를 출력한다
- `plt.axis([xmin,xmax,ymin,ymax])` 는 축의 범위를 지정한다.
- `plt.xlim([xmin, xmax])` , `plt.ylim([ymin, ymax])` 을 사용할 수도 있다.
- `plt.grid(True)` 는 격자를 그리고, `plt.legend()`는 레전드를 표시한다.
- `plt.tight_layout()` 은 여백을 조정하는 역할을 한다.
- `latex` 수식을 `title()` , `xlabel()` , `ylabel()` , `text()` 등에 지정할 수 있다. 다만 `r'text'` 등과 같은 `r`을 앞에 기입하여 raw 문자열이어야 한다.

## Subplot

Subplot은 Matlab처럼 `plt.subplot(nrow,ncol,inum)` 을 호출하는 것으로 그릴 수 있다.

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

x = np.linspace(0,1,50)

y1 = np.cos(4*np.pi*x)
y2 = np.cos(4*np.pi*x)*np.exp(-2*x)

plt.subplot(2,1,1)
plt.plot(x,y1,'r-* ',lw=1)
plt.grid(True)
plt.ylabel(r'$sin(4 \pi x)$')
plt.axis([0,1,-1.5,1.5])
```

```
plt.subplot(2,1,2)
plt.plot(x,y2,'b--o ',lw=1)
plt.grid(True)
plt.xlabel('x')
plt.ylabel(r'$ e^{-2x} sin(4 \pi x) $')
plt.axis([0,1,-1.5,1.5])

plt.tight_layout()
plt.show()
```

## Line and scatter plot 예제

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np

# line plot
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
# ax = fig.gca(projection='3d')

theta = np.linspace(-4*np.pi, 4*np.pi, 100)
z = np.linspace(-2, 2, 100)
```

```
r = z**2+1
x = r*np.sin(theta)
y = r*np.cos(theta)

ax.plot(x,y,z,label='parametric curve')
ax.scatter(x,y,z)
ax.legend()

ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')

fig.tight_layout()
plt.show()
```



```
plt.text(20,30,"test')      # 일반
plt.text(0.2,0.3, horizontalalignment='center',verticalalignment='center',transform=plt.gca().transAxes)  #
transform=ax.transAxes
```

## 주로 사용하는 선과 마커

다음은 선의 색상 및 스타일, 마커 등의 옵션이다.

### Color

character	color
'b'	blue
'g'	green
'r'	red
'c'	cyan
'm'	magenta
'y'	yellow
'k'	black
'w'	white

### LineStyle

character	description
'-'	solid line style
'--'	dashed line style
'-.'	dash-dot line style
':'	dotted line style

### Marker

character	description
'.'	point marker
','	pixel marker
'o'	circle marker
'v'	triangle_down marker
'^'	triangle_up marker
'<'	triangle_left marker
'>'	triangle_right marker
'1'	tri_down marker
'2'	tri_up marker
'3'	tri_left marker
'4'	tri_right marker
's'	square marker
'p'	pentagon marker
'*'	star marker

character	description
'h'	hexagon1 marker
'H'	hexagon2 marker
'+'	plus marker
'x'	x marker
'D'	diamond marker
'd'	thin_diamond marker
' '	vline marker
'_'	hline marker

jupyter Untitled Last Checkpoint: 2023.08.06. (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

```
In [1]: import numpy as np
import matplotlib.pyplot as plt
from numpy import pi

a = np.arange(10, 30, 5) # 10부터 30까지 5간격(시작점, 끝점, 간격)
b = np.arange(0, 2, 0.3) # 0부터 2까지 0.3간격

print(a)
print(b)

c = np.linspace(0, 2, 15) # 0~ 2까지 15개 숫자

print(c)

x = np.linspace(0, 2*pi, 100) # 0부터 2*pi 100개 만들기
f = np.sin(x)
print(f)

points = np.arange(-5, 5, 0.01)
xs, ys = np.meshgrid(points, points)

z = np.sqrt(xs**2 + ys**2)

plt.imshow(z, cmap=plt.cm.gray)
plt.colorbar()
plt.title("Image")
```

[10 15 20 25]  
[0. 0.3 0.6 0.9 1.2 1.5 1.8]  
[0. 0.14285714 0.28571429 0.42857143 0.57142857 0.71428571  
0.85714286 1. 1.14285714 1.28571429 1.42857143 1.57142857  
1.71428571 1.85714286 2. ]  
[ 0.00000000e+00 6.34239197e-02 1.26592454e-01 1.89251244e-01  
2.51147987e-01 3.12033446e-01 3.71662456e-01 4.29794912e-01

## Linspace\_sin.ipynb

# 머신러닝 실습

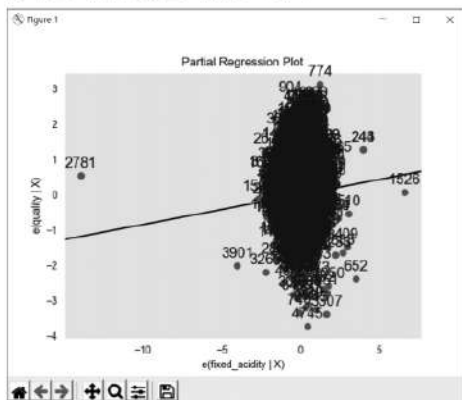
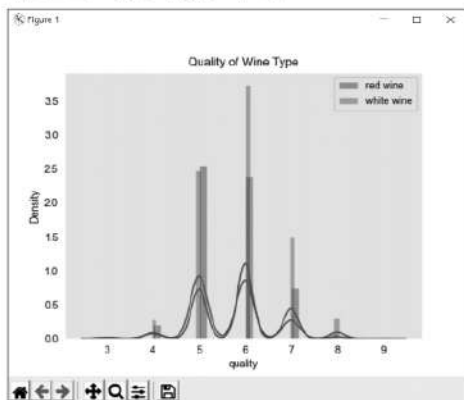
(와인 데이터 분석)



## ROKEY BOOT CAMP

### 1. 히스토그램을 이용한 시각화

### 2. 부분 회귀 플롯을 이용한 시각화



## • 데이터 수집

- 캘리포니아 어바인 대학의 머신러닝 저장소에서 제공하는 오픈 데이터를 사용
- 다운로드한 파일은 My\_Python/7장\_data 폴더를 만든 후에 저장

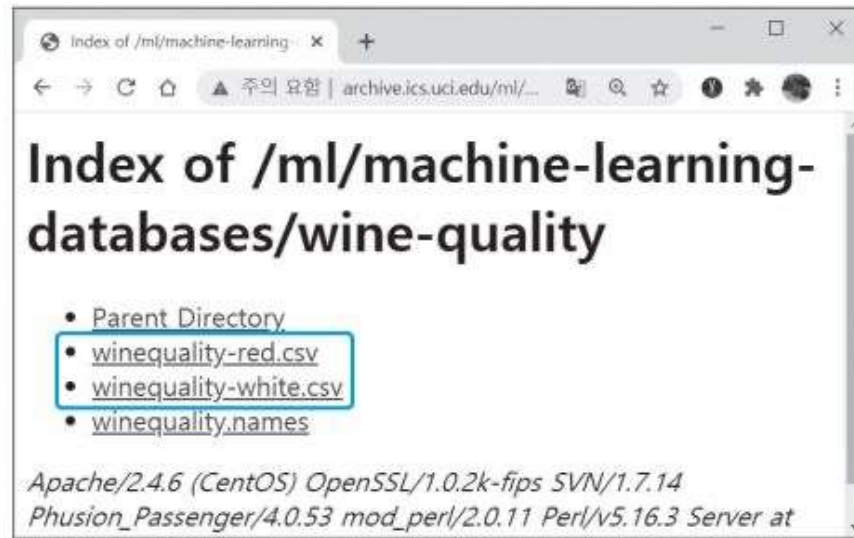


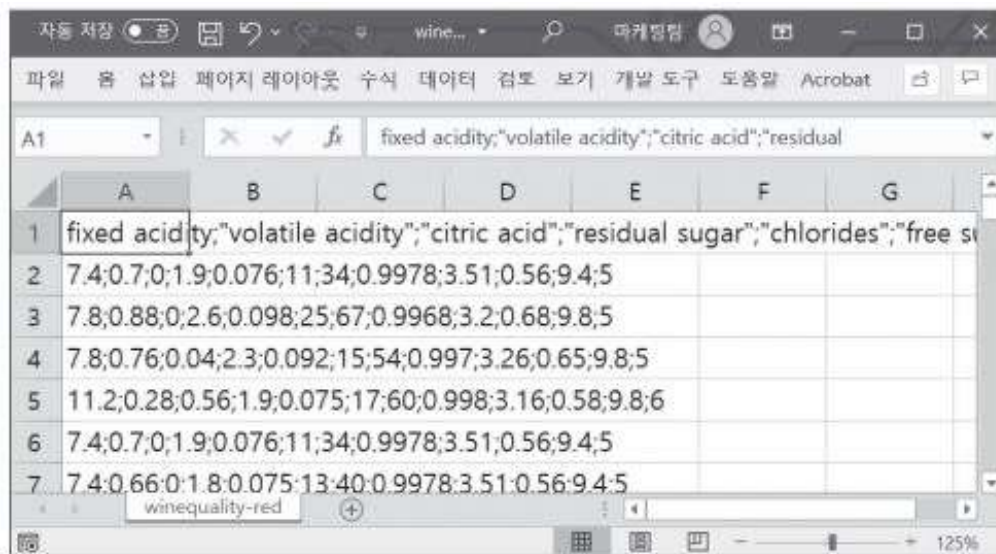
그림 7-1 와인 데이터 다운로드

<https://archive.ics.uci.edu/dataset/186/wine+quality>

## • 데이터 준비

### 1. 다운로드한 CSV 파일 정리하기

- 엑셀은 CSV 파일을 열 때 쉼표를 열 구분자로 사용하므로 열이 깨진 것처럼 보임
- 엑셀에서 세미콜론을 열 구분자로 인식하도록 파일을 다시 저장해야 함



	A	B	C	D	E	F	G
1	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free s	
2	7.4	0.7	0.19	0.076	11.34	0.9978	3.51
3	7.8	0.88	0.26	0.098	25.67	0.9968	3.2
4	7.8	0.76	0.04	2.3	0.092	15.54	0.997
5	11.2	0.28	0.56	1.9	0.075	17.60	0.998
6	7.4	0.7	0.19	0.076	11.34	0.9978	3.51
7	7.4	0.66	0.18	0.075	13.40	0.9978	3.51

그림 7-2 다운로드한 CSV 파일을 엑셀에서 열기

## • 데이터 준비

### 1. 다운로드한 CSV 파일 정리하기

#### 1. 엑셀에서 열 구분자를 세미콜론으로 인식시키기

```
01 import pandas as pd
02 red_df = pd.read_csv('C:/Users/kmj/My_Python/7장_data/winequality-red.csv', sep = ';', header = 0, engine = 'python')
03 white_df = pd.read_csv('C:/Users/kmj/My_Python/7장_data/winequality-white.csv', sep = ';', header = 0, engine = 'python')
04 red_df.to_csv('C:/Users/kmj/My_Python/7장_data/winequality-red2.csv', index = False)
05 white_df.to_csv('C:/Users/kmj/My_Python/7장_data/winequality-white2.csv', index = False)
```

- 01행 테이블 형태의 CSV 파일을 다루기 위해 pandas 라이브러리 패키지를 pd 이름으로 로드
- 02~03행 pandas의 read\_csv() 함수를 사용해 CSV 파일을 읽어온음 이때 CSV 파일 데이터의 열 구분자를 세미콜론으로 지정하기 위해 sep 매개변수 값을 ';'으로 지정
- 04~05행 pandas로 읽은 CSV 데이터는 테이블 형태의 DataFrame 객체(red\_df, white\_df)에 있음

이 상태 그대로 CSV 파일로 저장



# 와인데이터 분석

## • 데이터 준비

### 1. 다운로드한 CSV 파일 정리하기

### 2. 파이썬에서 저장한 CSV 파일을 엑셀에서 열어 상태를 확인

	A	B	C	D	E	F	G	H	I	J	K	L
1	acidity	volatile acid	citric acid	residual sugar	chlorides	free sulfur	total sulfur	density	pH	sulphates	alcohol	quality
2	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5
3	7.8	0.88	0	2.6	0.098	25	67	0.9968	3.2	0.68	9.8	5
4	7.8	0.76	0.04	2.3	0.092	15	54	0.997	3.26	0.65	9.8	5
5	11.2	0.28	0.56	1.9	0.075	17	60	0.998	3.16	0.58	9.8	6
6	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5

(a) winequality-red2.csv

	A	B	C	D	E	F	G	H	I	J	K	L
1	fixed acid	volatile acid	citric acid	residual sugar	chlorides	free sulfur	total sulfur	density	pH	sulphates	alcohol	quality
2	7	0.27	0.36	20.7	0.045	45	170	1.001	3	0.45	8.8	6
3	6.3	0.3	0.34	1.6	0.049	14	132	0.994	3.3	0.49	9.5	6
4	8.1	0.28	0.4	6.9	0.05	30	97	0.9951	3.26	0.44	10.1	6
5	7.2	0.23	0.32	8.5	0.058	47	186	0.9956	3.19	0.4	9.9	6
6	7.2	0.23	0.32	8.5	0.058	47	186	0.9956	3.19	0.4	9.9	6

(b) winequality-white2.csv

그림 7-3 엑셀의 열 구분자를 수정한 뒤 저장한 파일 다시 열기

- 데이터 준비

- 2. 데이터 병합하기

- 1. 레드 와인과 화이트 와인 파일 합치기

```
01 >>> red_df.head()
   fixed acidity  volatile acidity  citric acid ...  sulphates  alcohol  quality
0         7.4             0.70         0.00 ...      0.56       9.4         5
1         7.8             0.88         0.00 ...      0.68       9.8         5
2         7.8             0.76         0.04 ...      0.65       9.8         5
3        11.2             0.28         0.56 ...      0.58       9.8         6
4         7.4             0.70         0.00 ...      0.56       9.4         5

[5 rows x 12 columns]
02 >>> red_df.insert(0, column = 'type', value = 'red')
03 >>> red_df.head()
   type  fixed acidity  volatile acidity ...  sulphates  alcohol  quality
0   red         7.4             0.70 ...      0.56       9.4         5
1   red         7.8             0.88 ...      0.68       9.8         5
2   red         7.8             0.76 ...      0.65       9.8         5
3   red        11.2             0.28 ...      0.58       9.8         6
4   red         7.4             0.70 ...      0.56       9.4         5

[5 rows x 13 columns]
04 >>> red_df.shape
(1599, 13)
```

- 데이터 준비

- 2. 데이터 병합하기

- 1. 레드 와인과 화이트 와인 파일 합치기

```
05 >>> white_df.head()
fixed acidity volatile acidity citric acid ... sulphates alcohol quality
0 7.0 0.27 0.36 ... 0.45 8.8 6
1 6.3 0.30 0.34 ... 0.49 9.5 6
2 8.1 0.28 0.40 ... 0.44 10.1 6
3 7.2 0.23 0.32 ... 0.40 9.9 6
4 7.2 0.23 0.32 ... 0.40 9.9 6

[5 rows x 12 columns]
06 >>> white_df.insert(0, column = 'type', value = 'white')
07 >>> white_df.head()
type fixed acidity volatile acidity ... sulphates alcohol quality
0 white 7.0 0.27 ... 0.45 8.8 6
1 white 6.3 0.30 ... 0.49 9.5 6
2 white 8.1 0.28 ... 0.44 10.1 6
3 white 7.2 0.23 ... 0.40 9.9 6
4 white 7.2 0.23 ... 0.40 9.9 6

[5 rows x 13 columns]
08 >>> white_df.shape
(4898, 13)
09 >>> wine = pd.concat([red_df, white_df])
10 >>> wine.shape
(6497, 13)
11 >>> wine.to_csv('C:/Users/kmj/My_Python/7장_data/wine.csv', index = False)
```

## • 데이터 준비

### 2. 데이터 병합하기

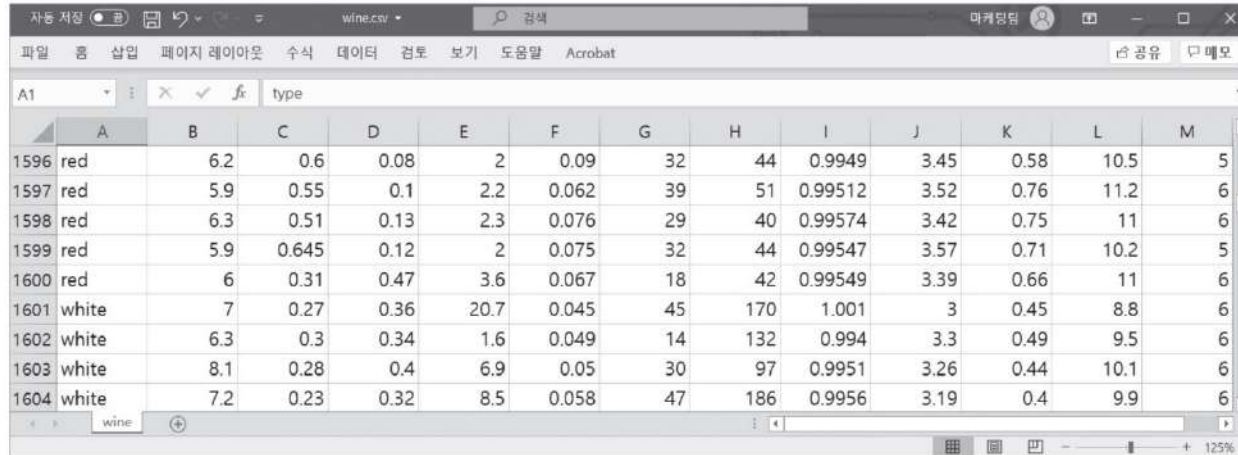
#### 1. 레드 와인과 화이트 와인 파일 합치기

- [01~04행] 레드 와인 파일을 읽고 데이터프레임에 'type' 열 삽입하기
  - 01행 red\_df에 저장된 내용을 위에서부터 5개(0번~4번) 행만 출력하여 확인
  - 02행 이름이 'type'이고 값이 'red'인 열을 만들어 index = 0(첫 번째 열) 자리에 삽입
  - 03행 red\_df에 저장된 내용을 위에서부터 5개(0번~4번) 행만 다시 출력해 삽입된 'type'열을 확인
  - 04행 red\_df.shape를 이용하여 현재 red\_df의 크기를 '(행의 개수, 열의 개수)' 형태로 확인

- [05~08행] 화이트 와인 파일을 읽고 데이터프레임에 'type' 열 삽입하기
  - 05행 white\_df에 저장된 내용을 위에서부터 5개(0번~4번) 행만 출력하여 확인
  - 06행 이름이 'type'이고 값이 'white'인 열을 만들어 index = 0(첫 번째 열) 자리에 삽입
  - 07행 white\_df에 저장된 내용을 위에서부터 5개(0번~4번) 행만 다시 출력해 삽입된 'type' 열을 확인
  - 08행 white\_df.shape를 이용하여 현재 white\_df의 크기를 '(행의 개수, 열의 개수)' 형태로 확인
- [09~10행] red\_df와 white\_df를 하나의 데이터프레임 형식으로 결합하기
  - 09행 pd.concat() 함수를 이용하여 red\_df와 white\_df를 결합
  - 10행 결합된 wine의 현재 크기를 '(행의 개수, 열의 개수)' 형태로 확인
  - 11행 wine을 CSV 파일로 저장

# 와인데이터 분석

- 데이터 준비
  - 2. 데이터 병합하기
    - 2. 결합된 파일 확인하기



	A	B	C	D	E	F	G	H	I	J	K	L	M
1596	red	6.2	0.6	0.08	2	0.09	32	44	0.9949	3.45	0.58	10.5	5
1597	red	5.9	0.55	0.1	2.2	0.062	39	51	0.99512	3.52	0.76	11.2	6
1598	red	6.3	0.51	0.13	2.3	0.076	29	40	0.99574	3.42	0.75	11	6
1599	red	5.9	0.645	0.12	2	0.075	32	44	0.99547	3.57	0.71	10.2	5
1600	red	6	0.31	0.47	3.6	0.067	18	42	0.99549	3.39	0.66	11	6
1601	white	7	0.27	0.36	20.7	0.045	45	170	1.001	3	0.45	8.8	6
1602	white	6.3	0.3	0.34	1.6	0.049	14	132	0.994	3.3	0.49	9.5	6
1603	white	8.1	0.28	0.4	6.9	0.05	30	97	0.9951	3.26	0.44	10.1	6
1604	white	7.2	0.23	0.32	8.5	0.058	47	186	0.9956	3.19	0.4	9.9	6

그림 7-4 레드 와인 데이터셋과 화이트 와인 데이터셋이 결합된 wine.csv 파일

## • 데이터 탐색

### 1. 기본 정보 확인하기

```
01 >>> 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
```

- 전체 샘플은 6,497개이고 속성을 나타내는 열은 13개
- 각 속성의 이름은 type부터 quality까지
- 속성 중에서 실수 타입(float64)은 11개
- 정수 타입(int64)은 1개(quality), 객체 타입(object)이 1개(type)
- 독립 변수(x)는 type부터 alcohol 까지 12개, 종속 변수(y)는 1개(quality)



## • 데이터 탐색

### 2. 함수를 사용해 기술 통계 구하기

```
01 >>> wine.columns = wine.columns.str.replace(' ', '_')
02 >>> wine.head()
   type  fixed acidity  volatile acidity  ...  sulphates  alcohol  quality
0  red           7.4             0.70    ...      0.56      9.4         5
1  red           7.8             0.88    ...      0.68      9.8         5
2  red           7.8             0.76    ...      0.65      9.8         5
3  red          11.2             0.28    ...      0.58      9.8         6
4  red           7.4             0.70    ...      0.56      9.4         5
[5 rows x 13 columns]
03 >>> wine.describe()
      fixed_acidity  volatile_acidity  ...  alcohol  quality
count  6497.000000    6497.000000    ...  6497.000000  6497.000000
mean     7.215307      0.339666    ...   10.491801    5.818378
std     1.296434      0.164636    ...    1.192712    0.873255
Min     3.800000      0.080000    ...    8.000000    3.000000
25%     6.400000      0.230000    ...    9.500000    5.000000
50%     7.000000      0.290000    ...   10.300000    6.000000
75%     7.700000      0.400000    ...   11.300000    6.000000
max    15.900000      1.580000    ...   14.900000    9.000000
[8 rows x 12 columns]
```

- [01~04행] 열이름 정렬하기
- 01행 열 이름에 공백이 있으면 밑줄로 바꾼 뒤 한 단어로 연결
- 02행 변경된 열 이름을 확인
- 03행 describe() 함수를 사용하여 속성별 개수, 평균, 표준편차, 최소값,
- 전체 데이터 백분율에 대한 25번째 백분위수(25%),
- 중앙값인 50번째 백분위수(50%), 75번째 백분위수(75%)
- 그리고 100번째 백분위수인 최대값max을 출력

- 데이터 탐색

- 2. 함수를 사용해 기술 통계 구하기

```
04 >>> sorted(wine.quality.unique())
      [3, 4, 5, 6, 7, 8, 9]
05 >>> wine.quality.value_counts()
      6      2836
      5      2138
      7      1079
      4       216
      8       193
      3        30
      9         5
      Name: quality, dtype: int64
      9         5
      Name: quality, dtype: int64
```

- 04행 `wine.quality.unique()` 함수를 사용하여 `quality` 속성값 중에서 유일한 값을 출력  
이를 통해 와인 품질 등급 `quality`은 3, 4, 5, 6, 7, 8, 9의 7개 등급이 있다는 것을 알 수 있음
- 05행 `quality.value_counts()` 함수는 `quality` 속성값에 대한 빈도수를 보여줌  
6등급인 샘플이 2,826개로 가장 많고, 9등급인 샘플이 5개로 가장 적은 것을 알 수 있음

## • 데이터 모델링

### 1. describe() 함수로 그룹 비교하기

```
01 >>> wine.groupby('type')['quality'].describe()
      count      mean      std  min  25%  50%  75%  max
type
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

02 >>> wine.groupby('type')['quality'].mean()
type
red      5.636023
white    5.877909
Name: quality, dtype: float64

03 >>> wine.groupby('type')['quality'].std()
type
red      0.807569
white    0.885639
Name: quality, dtype: float64

04 >>> wine.groupby('type')['quality'].agg(['mean', 'std'])
      mean      std
type
red    5.636023  0.807569
white  5.877909  0.885639
```

- 01행 레드 와인과 화이트 와인을 구분하는 속성인 **type**을 기준으로 그룹을 나눈 뒤 그룹 안에서 **quality** 속성을 기준으로 기술 통계를 구함
- 02~04행 기술 통계 전부를 구할 때는 **describe()** 함수를 사용하지만 **mean()** 함수로 평균만 구하거나 **std()** 함수로 표준편차만 따로 구할 수도 있음
- **mean()** 함수와 **std()** 함수를 묶어서 한 번에 사용하려면 04행과 같이 **agg()** 함수를 사용

- 데이터 모델링

- 3. 회귀 분석 모델로 새로운 샘플의 품질 등급 예측하기

```
01 >>> sample1 = wine[wine.columns.difference(['quality', 'type'])]
02 >>> sample1 = sample1[0:5][:]
03 >>> sample1_predict = regression_result.predict(sample1)
04 >>> sample1_predict
0    4.997607
1    4.924993
2    5.034663
3    5.680333
4    4.997607
dtype: float64
05 >>> wine[0:5]['quality']
0    5
1    5
2    5
3    6
4    5
Name: quality, dtype: int64
```

- 데이터 모델링

## 3. 회귀 분석 모델로 새로운 샘플의 품질 등급 예측하기

```
06 >>> 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]}
07 >>> sample2 = pd.DataFrame(data, columns= sample1.columns)
08 >>> sample2
   alcohol chlorides ... total_sulfur_dioxide volatile_acidity
0    9.00    0.055 ...           98.0             0.8
1    0.88    0.040 ...           99.0             0.5
[2 rows x 11 columns]
09 >>> sample2_predict = regression_result.predict(sample2)
10 >>> sample2_predict
0    4.809094
1    7.582129
dtype: float64
```

## • 데이터 모델링

### 3. 회귀 분석 모델로 새로운 샘플의 품질 등급 예측하기

- [01~02행] 예측에 사용할 첫 번째 샘플 데이터 만들기
  - 01행 wine에서 quality와 type 열은 제외하고, 회귀 분석 모델에 사용할 독립 변수만 추출하여 sample1에 저장
  - 02행 sample1에 있는 샘플 중에서 0번~4번 5개 샘플만 추출하고, sample1에 다시 저장하여 예측에 사용할 샘플을 제작
- [03~05행] 첫 번째 샘플의 quality 예측하기
  - 03행 샘플 데이터를 회귀 분석 모델 regression\_result의 예측 함수 predict()에 적용하여 수행한 뒤 결과 예측값을 sample1\_predict에 저장
  - 04행 sample1\_predict를 출력하여 예측한 quality를 확인
  - 05행 wine에서 0번부터 4번까지 샘플의 quality 값을 출력하여 sample1\_predict이 맞게 예측되었는지 확인

- [06~08행] 예측에 사용할 두 번째 샘플 데이터 만들기
  - 06행 회귀식에 사용한 독립 변수에 대입할 임의의 값을 딕셔너리 형태로 제작
  - 07행 딕셔너리 형태의 값과 sample1의 열 이름만 뽑아 데이터프레임으로 묶은 sample2를 제작
  - 08행 sample2를 출력하여 제대로 구성되었는지 확인
- [09~10행] 두 번째 샘플의 quality 예측하기
  - 09행 샘플 데이터를 회귀 분석 모델 regression\_result의 예측 함수 predict()에 적용하여 수행한 뒤 결과 예측값을 sample2\_predict에 저장
  - 10행 sample2\_predict를 출력하여 예측한 quality를 확인

## • 결과 시각화

### 1. 와인 유형에 따른 품질 등급 히스토그램 그리기

1. 명령 프롬프트 창에서 다음 명령을 입력하여 seaborn 라이브러리 패키지를 설치  
그 후 파이썬 셸 창으로 돌아와 임포트



### 2. 파이썬 셸 창에 명령을 입력

```
01 >>> import matplotlib.pyplot as plt
02 >>> import seaborn as sns
03 >>> sns.set_style('dark')
04 >>> sns.distplot(red_wine_quality, kde = True, color = "red", label = 'red
    wine')
    <matplotlib.axes._subplots.AxesSubplot object at 0x000001C843F36348>
05 >>> sns.distplot(white_wine_quality, kde = True, label = 'white wine')
    <matplotlib.axes._subplots.AxesSubplot object at 0x000001C843F36348>
06 >>> plt.title("Quality of Wine Type")
    Text(0.5, 1.0, 'Quality of Wine Type')
07 >>> plt.legend()
    <matplotlib.legend.Legend object at 0x0000014F20DFA548>
08 >>> plt.show()
```



## • 결과 시각화

1. 와인 유형에 따른 품질 등급 히스토그램 그리기
2. 파이썬 셀 창에 명령을 입력
  - 01~02행 시각화에 필요한 패키지를 로드
  - [03~08행] 커널 밀도 추정(kde)을 적용한 히스토그램 그리기
    - 03행 히스토그램 차트의 배경색 스타일을 설정
    - 04행 레드 와인에 대한 distplot 객체를 생성
    - 05행 화이트 와인에 대한 distplot 객체를 생성
    - 06행 차트 제목을 설정
    - 07행 차트 범례를 설정
    - 08행 설정한 내용대로 차트를 표시
  - x축: qualit
  - y축: 확률 밀도 함수값

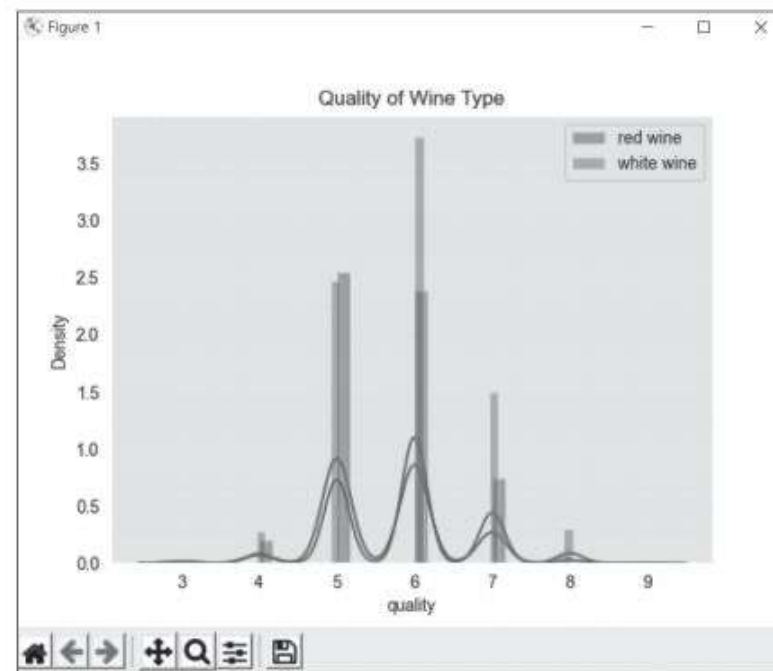


그림 7-5 와인 유형에 따른 품질 등급 히스토그램

## • 결과 시각화

### 2. 부분 회귀 플롯으로 시각화하기

- 독립 변수가 2개 이상인 경우에는 부분 회귀 플롯을 사용하여 하나의 독립 변수가 종속 변수에 미치는 영향력을 시각화 함으로써 결과를 분석할 수 있음

```
01 >>> import statsmodels.api as sm
02 >>> others = list(set(wine.columns).difference(set(["quality", "fixed_acidity"])))
03 >>> p, resid = sm.graphics.plot_partregress("quality", "fixed_acidity", others, data = wine, ret_coors = True)
04 >>> plt.show()
05 >>> fig = plt.figure(figsize = (8, 13))
06 >>> sm.graphics.plot_partregress_grid(regression_result, fig = fig)
    <Figure size 800x1300 with 12 Axes>
07 >>> plt.show()
```

## • 결과 시각화

### 2. 부분 회귀 플롯으로 시각화하기

- 01행 부분 회귀 계산을 위해 statsmodels.api를 로드
- [02~04행] fixed\_acidity가 종속 변수 quality에 미치는 영향력을 시각화하기
  - 02행 부분 회귀에 사용한 독립 변수와 종속 변수를 제외한 나머지 변수 이름을 리스트 others로 추출
  - 03행 나머지 변수는 고정하고 fixed\_acidity가 종속 변수 quality에 미치는 영향에 부분회귀를 수행
  - 04행 부분 회귀의 결과를 플롯으로 시각화하여 나타냄

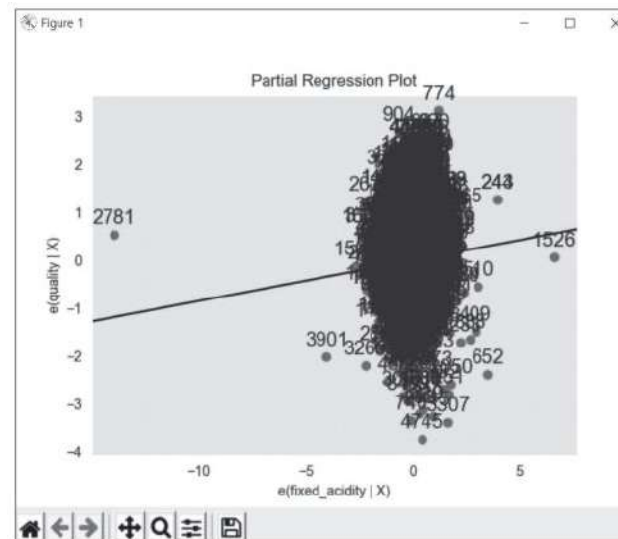


그림 7-6 독립 변수 fixed\_acidity와 종속 변수 quality에 대한 부분 회귀 시각화

## • 결과 시각화

### 2. 부분 회귀 플롯으로 시각화하기

- [05~07행] 각 독립 변수가 종속 변수 quality에 미치는 영향력을 시각화하기
  - 05행 차트의 크기를 지정
  - 06행 다중 선형 회귀 분석 결과를 가지고 있는 `regression_result`를 이용해 각 독립 변수의 부분 회귀 플롯을 구함
  - 07행 부분 회귀 결과를 플롯으로 시각화하여 나타냄

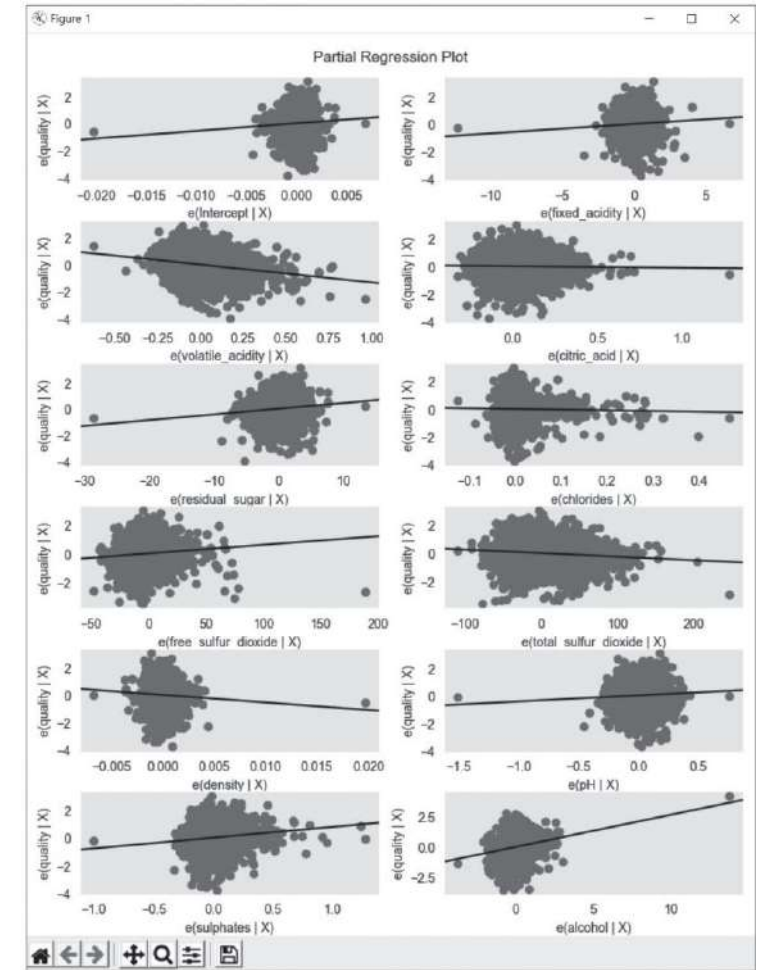


그림 7-7 각각의 독립 변수와 종속 변수 quality에 대한 부분 회귀 시각화

# 머신러닝 실습

(타이타닉호 분석)

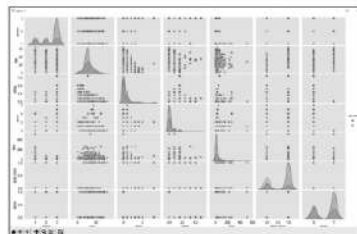


## 02. [상관 분석 + 히트맵] 타이타닉호 생존율 분석하기

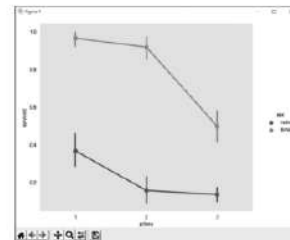
### • 분석 미리보기

타이타닉호 생존율 분석하기	
목표	타이타닉호 승객 변수를 분석하여 생존율과의 상관관계를 찾는다.
핵심 개념	상관 분석, 상관 계수, 피어슨 상관 계수, 히트맵
데이터 수집	타이타닉 데이터: seaborn 내장 데이터셋
데이터 준비	결측치 치환: 중앙값 치환, 최빈값 치환
데이터 탐색	1. 정보 확인: info() 2. 차트를 통한 데이터 탐색: pie(), countplot()
데이터 모델링	1. 모든 변수 간 상관 계수 구하기 2. 지정한 두 변수 간 상관계수 구하기
결과 시각화	

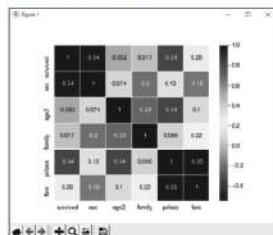
#### 1. 산점도를 이용한 시각화



#### 2. 특정 변수 간 상관관계 시각화



#### 3. 히트맵을 이용한 시각화



## 02. [상관 분석 + 히트맵] 타이타닉호 생존율 분석하기

- 분석 미리보기

- 타이타닉호의 생존자와 관련된 변수의 상관관계를 찾아봄
- 생존과 가장 상관도가 높은 변수는 무엇인지 분석
- 상관 분석을 위해 피어슨 상관 계수를 사용
- 변수 간의 상관관계는 시각화하여 분석

## 02. [상관 분석 + 히트맵] 타이타닉호 생존율 분석하기

- 핵심 개념 이해
  - 상관 분석
    - 두 변수가 어떤 선형적 관계에 있는지를 분석하는 방법
    - 두 변수는 서로 독립적이거나 상관된 관계일 수 있는데, 두 변수의 관계의 강도를 상관관계 라고함
    - 상관 분석에서는 상관관계의 정도를 나타내는 단위로 모상관 계수  $\rho$ 를 사용
    - 상관 계수는 두 변수가 연관된 정도를 나타낼 뿐 인과 관계를 설명하지 않으므로 정확한 예측치를 계산할 수는 없음
  - 단순 상관 분석
    - 두 변수가 어느 정도 강한 관계에 있는지 측정
  - 다중 상관 분석
    - 세 개 이상의 변수 간 관계의 강도를 측정
    - 편상관 분석: 다른 변수와의 관계를 고정하고 두 변수 간 관계의 강도를 나타내는 것
- 상관 계수  $\rho$ 
  - 변수 간 관계의 정도(0~1)와 방향(+, -)을 하나의 수치로 요약해주는 지수로 -1에서 +1 사이의 값을 가짐
  - 상관 계수가 +이면 양의 상관관계이며 한 변수가 증가하면 다른 변수도 증가
  - 상관 계수가 -이면 음의 상관관계이며 한 변수가 증가할 때 다른 변수는 감소
  - 0.0 ~ 0.2: 상관관계가 거의 없음
  - 0.2 ~ 0.4: 약한 상관관계가 있음
  - 0.4 ~ 0.6: 상관관계가 있음
  - 0.6 ~ 0.8: 강한 상관관계가 있음
  - 0.8 ~ 1.0: 매우 강한 상관관계가 있음



## 02. [상관 분석 + 히트맵] 타이타닉호 생존율 분석하기

- 데이터 수집

- 01행 seaborn 패키지를 로드
- 03행 titanic 데이터를 로드
- 04행 데이터를 CSV 파일로 저장

```
01 >>> import seaborn as sns
02 >>> import pandas as pd
03 >>> titanic = sns.load_dataset("titanic")
04 >>> titanic.to_csv('C:/Users/kmj/My_Python/7장_data/titanic.csv', index = False)
```

- 데이터 준비



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
2	0	3	male	22	1	0	7.25	S	Third	man	TRUE		Southamp	no	FALSE
3	1	1	female	38	1	0	71.2833	C	First	woman	FALSE	C	Cherbourg	yes	FALSE
4	1	3	female	26	0	0	7.925	S	Third	woman	FALSE		Southamp	yes	TRUE
5	1	1	female	35	1	0	53.1	S	First	woman	FALSE	C	Southamp	yes	FALSE
6	0	3	male	35	0	0	8.05	S	Third	man	TRUE		Southamp	no	TRUE
7	0	3	male		0	0	8.4583	Q	Third	man	TRUE		Queenst	no	TRUE

그림 7-8 다운로드한 파일(titanic.csv) 열기

- 저장한 titanic.csv 파일을 열어서 데이터 정리 작업이 필요한지 확인

## 02. [상관 분석 + 히트맵] 타이타닉호 생존율 분석하기

- 데이터 준비

```
01 >>> titanic.isnull().sum()
survived      0
pclass        0
sex           0
age          177
sibsp         0
parch         0
fare          0
embarked      2
class         0
who           0
adult_male    0
deck         688
embark_town    2
alive         0
alone         0
dtype: int64
02 >>> titanic['age'] = titanic['age'].fillna(titanic['age'].median())
03 >>> titanic['embarked'].value_counts()
S    644
C    168
Q     77
Name: embarked, dtype: int64
06 >>> titanic['embark_town'] = titanic['embark_town'].fillna('Southampton')
07 >>> titanic['deck'].value_counts()
```

```

C     59
B     47
D     33
E     32
A     15
F     13
G      4
Name: deck, dtype: int64
08 >>> titanic['deck'] = titanic['deck'].fillna('C')
09 >>> titanic.isnull().sum()
survived      0
pclass        0
sex           0
age           0
sibsp         0
parch         0
fare          0
embarked      0
class         0
who           0
adult_male    0
deck          0
embark_town    0
alive         0
alone         0
dtype: int64
```

## 02. [상관 분석 + 히트맵] 타이타닉호 생존율 분석하기

- 데이터 탐색

- 데이터의 기본 정보 탐색하기

```
01 >>> titanic.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
#   Column                Non-Null Count  Dtype
---  -
0   survived              891 non-null    int64
1   pclass                891 non-null    int64
2   sex                   891 non-null    object
3   age                   891 non-null    float64
4   sibsp                 891 non-null    int64
5   parch                891 non-null    int64
6   fare 891              non-null        float64
7   embarked              891 non-null    object
8   class 891             non-null        category
9   who 891               non-null        object
10  adult_male            891 non-null    bool
11  deck 891              non-null        category
12  embark_town           891 non-null    object
13  alive                 891 non-null    object
14  alone 891             non-null        bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.6+ KB

02 >>> titanic.survived.value_counts()
0    549
1    342
Name: survived, dtype: int64
```

- 01행 타이타닉 데이터의 기본 정보를 확인
- 02행 survived 속성값의 빈도를 확인
- 전체 샘플의 수: 891개이고 속성은 15개
- 샘플 891명 중에서 생존자는 342명이고 사망자는 549명
- pclass, class: 객실 등급
- sibsp: 함께 탑승한 형제자매와 배우자 수
- parch: 함께 탑승한 부모/자식 수
- embarked, embark\_town: 탑승 항구
- adult\_male: 성인 남자 여부
- alone: 동행 여부를 True/False로 나타냄

## 02. [상관 분석 + 히트맵] 타이타닉호 생존율 분석하기

- 데이터 탐색

### 2. 차트를 그려 데이터를 시각적으로 탐색하기

- 01행 차트를 그리기 위해 matplotlib.pyplot를 로드
- [02~07행] 남자 승객과 여자 승객의 생존율을 pie 차트로 그리기
  - 02행 한 줄에 두 개의 차트를 그리도록 하고 크기를 설정
  - 03행 첫 번째 pie 차트는 남자 승객의 생존율을 나타내도록 설정
  - 04행 두 번째 pie 차트는 여자 승객의 생존율을 나타내도록 설정
  - 05행 첫 번째 차트의 제목을 설정
  - 06행 두 번째 차트의 제목을 설정
  - 07행 구성된 차트를 나타낸다.

```
01 >>> import matplotlib.pyplot as plt
02 >>> f,ax = plt.subplots(1, 2, figsize = (10, 5))
03 >>> titanic['survived'][titanic['sex'] == 'male'].value_counts().plot.
    pie(explode = [0,0.1], autopct = '%1.1f%%', ax = ax[0], shadow = True)
    <matplotlib.axes._subplots.AxesSubplot object at 0x000001DD48E0C648>
04 >>> titanic['survived'][titanic['sex'] == 'female'].value_counts().plot.
    pie(explode = [0,0.1], autopct = '%1.1f%%', ax = ax[1], shadow = True)
    <matplotlib.axes._subplots.AxesSubplot object at 0x000001DD491C35C8>
05 >>> ax[0].set_title('Survived (Male)')
    Text(0.5, 1.0, 'Survived (Male)')
06 >>> ax[1].set_title('Survived (Female)')
    Text(0.5, 1.0, 'Survived (Female)')
07 >>> plt.show()
```

## 02. [상관 분석 + 히트맵] 타이타닉호 생존율 분석하기

- 데이터 탐색
  - 2. 차트를 그려 데이터를 시각적으로 탐색하기

- 남자 승객의 생존율: 18.9%
- 여자 승객의 생존율 74.2%

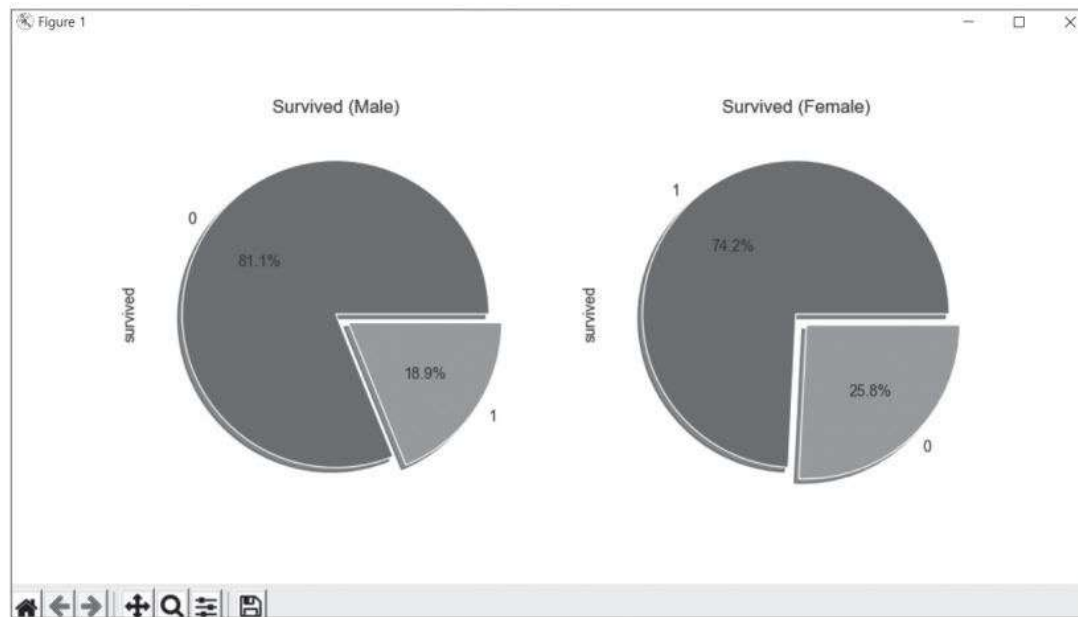


그림 7-9 성별에 따른 생존율 차트

## 02. [상관 분석 + 히트맵] 타이타닉호 생존율 분석하기

### • 데이터 탐색

#### 3. 등급별 생존자 수를 차트로 나타내기

```
01 >>> sns.countplot('pclass', hue = 'survived', data = titanic)
    <matplotlib.axes._subplots.AxesSubplot object at 0x000001DD48E03EC8>
02 >>> plt.title('Pclass vs Survived')
    Text(0.5, 1.0, 'Pclass vs Survived')
03 >>> plt.show()
```

- 01행 pclass 유형 1,2,3을 x축으로 하고 survived =0과 survived =1의 개수를 계산하여 y축으로 하는 countplot을 설정
- 02행 차트 제목을 설정
- 03행 구성한 차트를 나타냄
- 생존자(1)는 1등급에서 가장 많음
- 사망자(0)는 3등급에서 월등히 많음

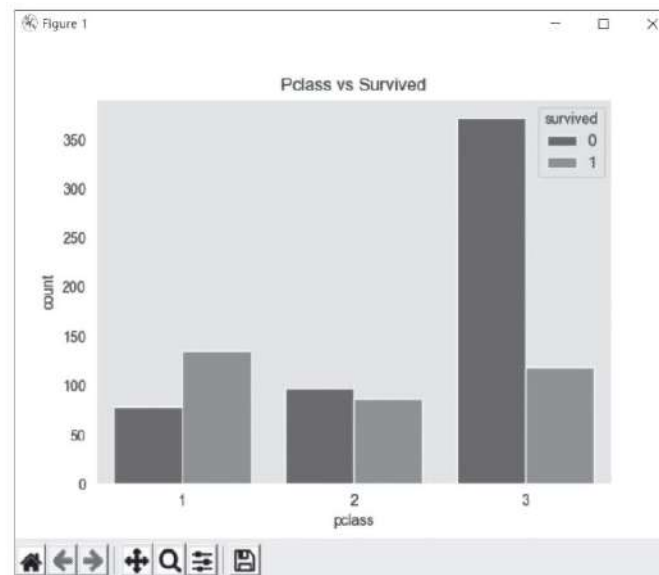


그림 7-10 객실 등급에 따른 생존자 수

## 02. [상관 분석 + 히트맵] 타이타닉호 생존율 분석하기

- 데이터 모델링

- 1. 상관 분석을 위한 상관 계수 구하고 저장하기

```
01 >>> titanic_corr = titanic.corr(method = 'pearson')
02 >>> titanic_corr
survived    pclass      age      ...    fare    adult_male    alone
survived      1.000000 -0.338481 -0.064910 ... 0.257307 -0.557080 -0.203367
pclass      -0.338481  1.000000 -0.339898 ... -0.549500  0.094035  0.135207
age          -0.064910 -0.339898  1.000000 ...  0.096688  0.247704  0.171647
sibsp       -0.035322  0.083081 -0.233296 ...  0.159651 -0.253586 -0.584471
parch        0.081629  0.018443 -0.172482 ...  0.216225 -0.349943 -0.583398
fare         0.257307 -0.549500  0.096688 ...  1.000000 -0.182024 -0.271832
adult_male-  0.557080  0.094035  0.247704 ... -0.182024  1.000000  0.404744
alone -      0.203367  0.135207  0.171647 ... -0.271832  0.404744  1.000000
[8 rows x 8 columns]
03 >>> titanic_corr.to_csv('C:/Users/kmj/My_Python/7장_data/titanic_corr.csv',
index = False)
```

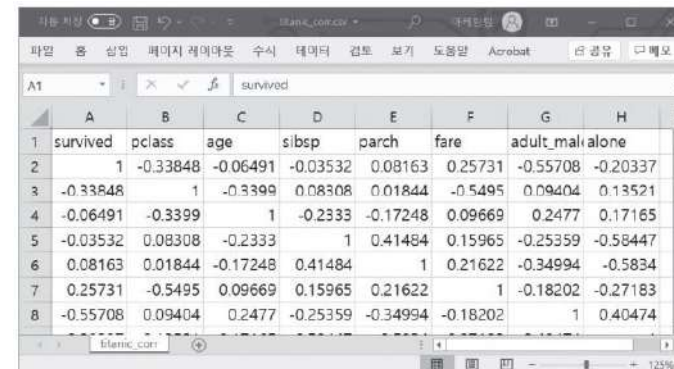
- 01행 피어슨 상관 계수를 적용하여 상관 계수를 구함
- 02행 상관 계수를 출력
- 03행 상관 계수를 CSV 파일로 저장

## 02. [상관 분석 + 히트맵] 타이타닉호 생존율 분석하기

### • 데이터 모델링

#### 2. 상관 계수 확인하기

- 남자 성인(adult\_male): 생존(survived)과 음의 상관관계
- 객실 등급(pclass): 음의 상관
- 관계, 객실 요금fare은 양의 상관관계
- 동행 없이 혼자 탑승한 경우(alone): 생존율이 떨어진다는 상관관계



	A	B	C	D	E	F	G	H
1	survived	pclass	age	sibsp	parch	fare	adult_male	alone
2	1	-0.33848	-0.06491	-0.03532	0.08163	0.25731	-0.55708	-0.20337
3	-0.33848	1	-0.3399	0.08308	0.01844	-0.5495	0.09404	0.13521
4	-0.06491	-0.3399	1	-0.2333	-0.17248	0.09669	0.2477	0.17165
5	-0.03532	0.08308	-0.2333	1	0.41484	0.15965	-0.25359	-0.58447
6	0.08163	0.01844	-0.17248	0.41484	1	0.21622	-0.34994	-0.5834
7	0.25731	-0.5495	0.09669	0.15965	0.21622	1	-0.18202	-0.27183
8	-0.55708	0.09404	0.2477	-0.25359	-0.34994	-0.18202	1	0.40474

그림 7-11 titanic\_corr.csv 파일에서 상관 계수 확인

#### 3. 특정 변수 사이의 상관 계수 구하기

```
01 >>> titanic['survived'].corr(titanic['adult_male'])  
-0.5570800422053259  
02 >>> titanic['survived'].corr(titanic['fare'])  
0.2573065223849622
```

- [01~02행] 두 변수 사이의 상관 계수 구하기
  - 01행 survived와 adult\_male 변수 사이의 상관 계수를 구함
  - 02행 survived와 fare 변수 사이의 상관 계수를 구함



## 02. [상관 분석 + 히트맵] 타이타닉호 생존율 분석하기

### • 결과 시각화

#### 1. 산점도로 상관 분석 시각화하기

```
01 >>> sns.pairplot(titanic, hue = 'survived')  
    <seaborn.axisgrid.PairGrid object at 0x000001710D852A58>  
02 >>> plt.show()
```

- [01~02행] 변수 간의 상관 분석 시각화를 위해 pairplot() 그리기
  - 01행 pairplot() 함수를 사용하여 타이타닉 데이터의 차트를 그림, hue는 종속 변수를 지정
  - 02행 pairplot을 나타냄

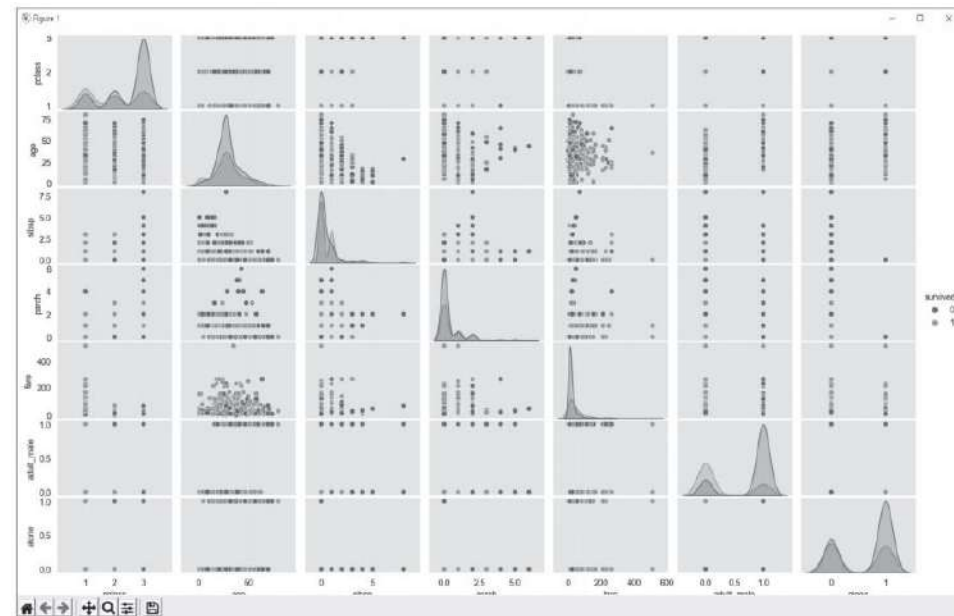


그림 7-12 pairplot() 함수를 이용한 산점도

## 02. [상관 분석 + 히트맵] 타이타닉호 생존율 분석하기

- 결과 시각화

- 2. 두 변수의 상관관계 시각화하기

```
01 >>> sns.catplot(x = 'pclass', y = 'survived', hue = 'sex', data = titanic, kind = 'point')  
      <seaborn.axisgrid.FacetGrid object at 0x000001DD44EB4B88>  
02 >>> plt.show()
```

- [01~02행] 생존자의 객실 등급과 성별 관계를 catplot()로 그리기
  - 01행 catplot() 함수를 사용하여 pclass와 survived 변수의 관계를 차트로 그림  
hue인자를 이용하여 종속 변수를 sex로 지정
  - 02행 catplot을 나타냄

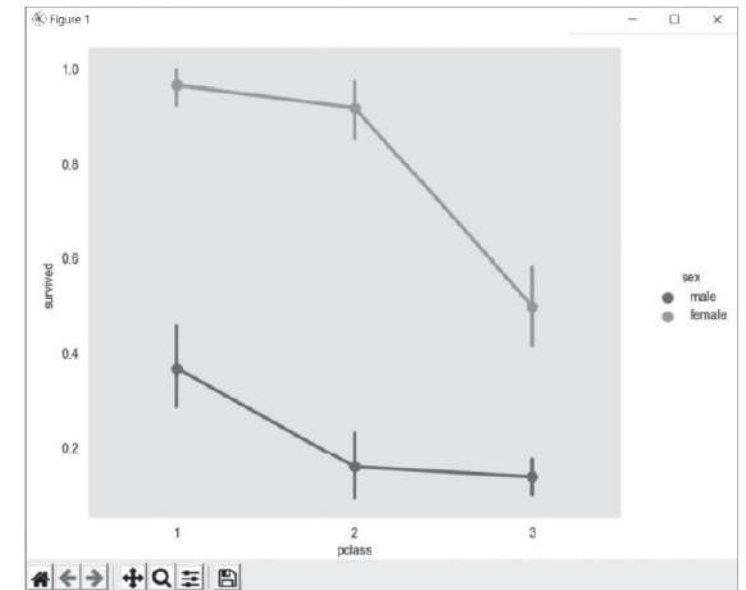


그림 7-13 객실 등급과 생존의 상관관계를 나타내는 catplot 차트

## 02. [상관 분석 + 히트맵] 타이타닉호 생존율 분석하기

- 결과 시각화
  - 2. 변수 사이의 상관 계수를 히트맵으로 시각화하기

```
01 >>> def category_age(x):  
    if x < 10:  
        return 0  
    elif x < 20:  
        return 1  
    elif x < 30:  
        return 2  
    elif x < 40:  
        return 3  
    elif x < 50:  
        return 4  
    elif x < 60:  
        return 5  
    elif x < 70:  
        return 6  
    else:  
        return 7
```


```
02 >>> titanic['age2'] = titanic['age'].apply(category_age)  
03 >>> titanic['sex'] = titanic['sex'].map({'male':1, 'female':0})  
04 >>> titanic['family'] = titanic['sibsp'] + titanic['parch'] + 1  
05 >>> titanic.to_csv('C:/Users/kmj/My_Python/7장_data/titanic3.csv', index =  
    False)  
06 >>> heatmap_data = titanic[['survived', 'sex', 'age2', 'family', 'pclass',  
    'fare']]  
07 >>> colormap = plt.cm.RdBu  
08 >>> sns.heatmap(heatmap_data.astype(float).corr(), linewidths = 0.1, vmax  
    = 1.0, square = True, cmap = colormap, linecolor = 'white', annot = True,  
    annot_kws = {"size": 10})  
    <matplotlib.axes._subplots.AxesSubplot object at 0x000001DD4C8DBF88>  
09 >>> plt.show()
```

## 02. [상관 분석 + 히트맵] 타이타닉호 생존율 분석하기

### • 결과 시각화

#### 3. 변수 사이의 상관 계수를 히트맵으로 시각화하기

- [01~02행] age를 카테고리 값으로 바꾸어 age2 변수로 추가하기
  - 01행 10살 단위로 등급을 나누어 0~7의 값으로 바꿔주는 category\_age 함수를 작성
  - 02행 category\_age 함수를 적용하여 새로운 age2 열을 만들어 추가
  - 03행 성별을 male/female에서 1/0으로 치환
  - 04행 가족의 수를 구하여 family 열을 추가
  - 05행 수정된 데이터프레임을 titanic3.csv로 저장



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_to	alive	alone	age2	family
2	0	3	1	22	1	0	7.25	S	Third	man	TRUE	C	Southamp	no	FALSE	2	2
3	1	1	0	38	1	0	71.2833	C	First	woman	FALSE	C	Cherbourg	yes	FALSE	3	2
4	1	3	0	26	0	0	7.925	S	Third	woman	FALSE	C	Southamp	yes	TRUE	2	1
5	1	1	0	35	1	0	53.1	S	First	woman	FALSE	C	Southamp	yes	FALSE	3	2
6	0	3	1	35	0	0	8.05	S	Third	man	TRUE	C	Southamp	no	TRUE	3	1
7	0	3	1	28	0	0	8.4583	Q	Third	man	TRUE	C	Queenstov	no	TRUE	2	1
8	0	1	1	54	0	0	51.8625	S	First	man	TRUE	E	Southamp	no	TRUE	5	1
9	0	3	1	2	3	1	21.075	S	Third	child	FALSE	C	Southamp	no	FALSE	0	5

그림 7-14 titanic3.csv 파일에서 치환된 내용과 추가된 내용 확인

## 02. [상관 분석 + 히트맵] 타이타닉호 생존율 분석하기

### • 결과 시각화

#### 3. 변수 사이의 상관 계수를 히트맵으로 시각화하기

- [06~09행] 상관 분석 결과를 히트맵으로 나타내기
  - 06행 히트맵에 사용할 데이터를 추출
  - 07행 히트맵에 사용할 색상맵을 지정
  - 08행 `corr()` 함수로 구한 상관 계수로 히트맵을 생성
  - 09행 생성한 히트맵을 나타냄

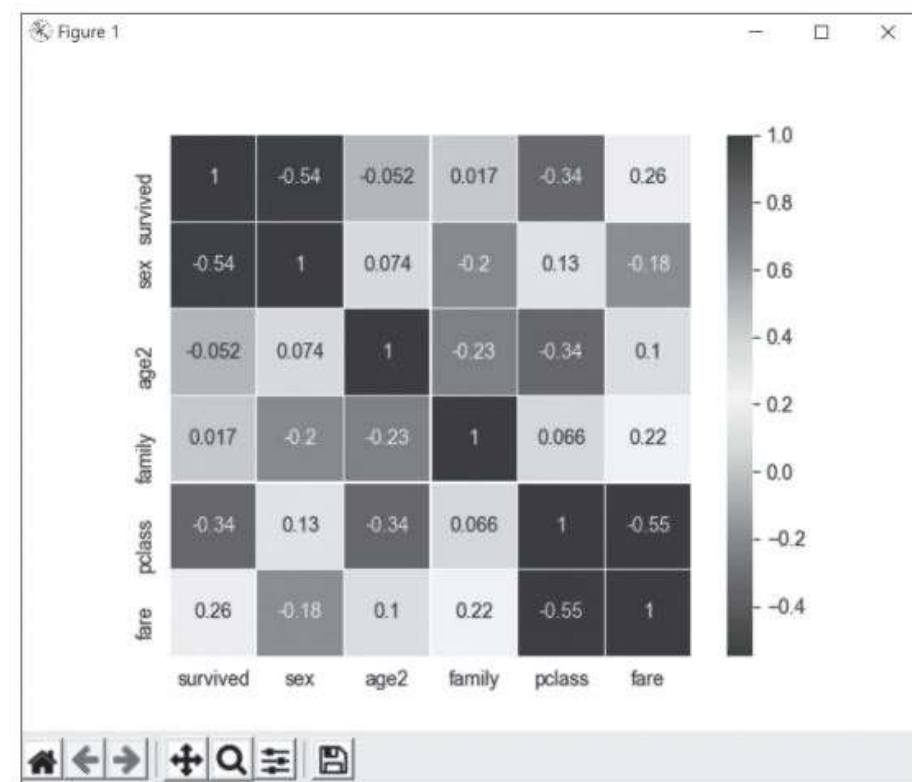


그림 7-15 상관 분석에 대한 히트맵 시각화

# 머신러닝 실습

- 수치 예측 -  
(당뇨병환자)



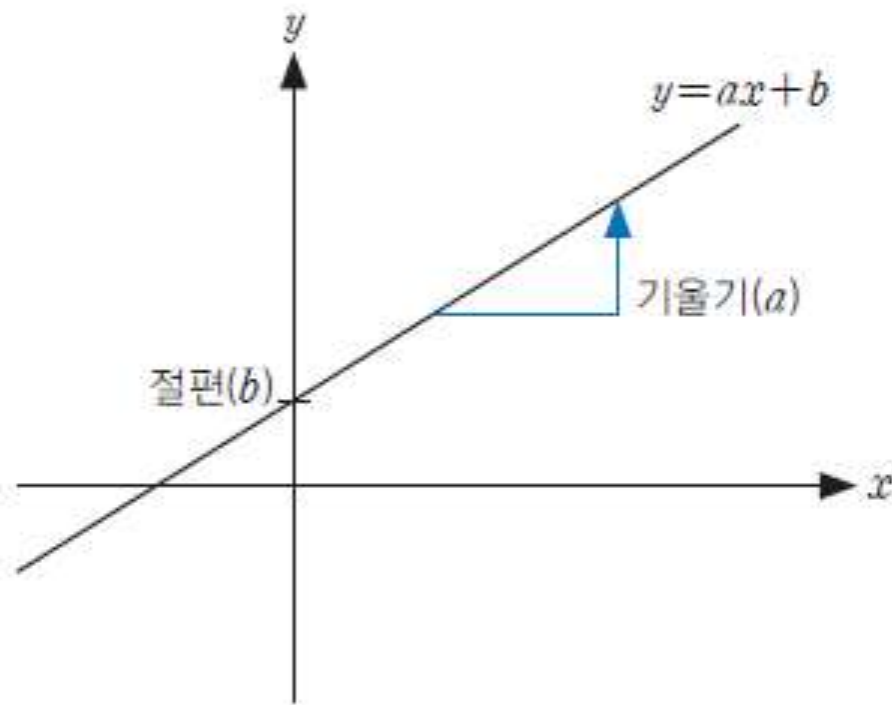
- 03-1 선형 회귀에 대해 알아보고 데이터를 준비합니다
- 03-2 경사 하강법으로 학습하는 방법을 알아봅니다
- 03-3 손실 함수와 경사 하강법의 관계를 알아봅니다
- 03-4 선형 회귀를 위한 뉴런을 만듭니다

- 03-1 선형 회귀에 대해 알아보고 데이터를 준비합니다
- 03-2 경사 하강법으로 학습하는 방법을 알아봅니다
- 03-3 손실 함수와 경사 하강법의 관계를 알아봅니다
- 03-4 선형 회귀를 위한 뉴런을 만듭니다



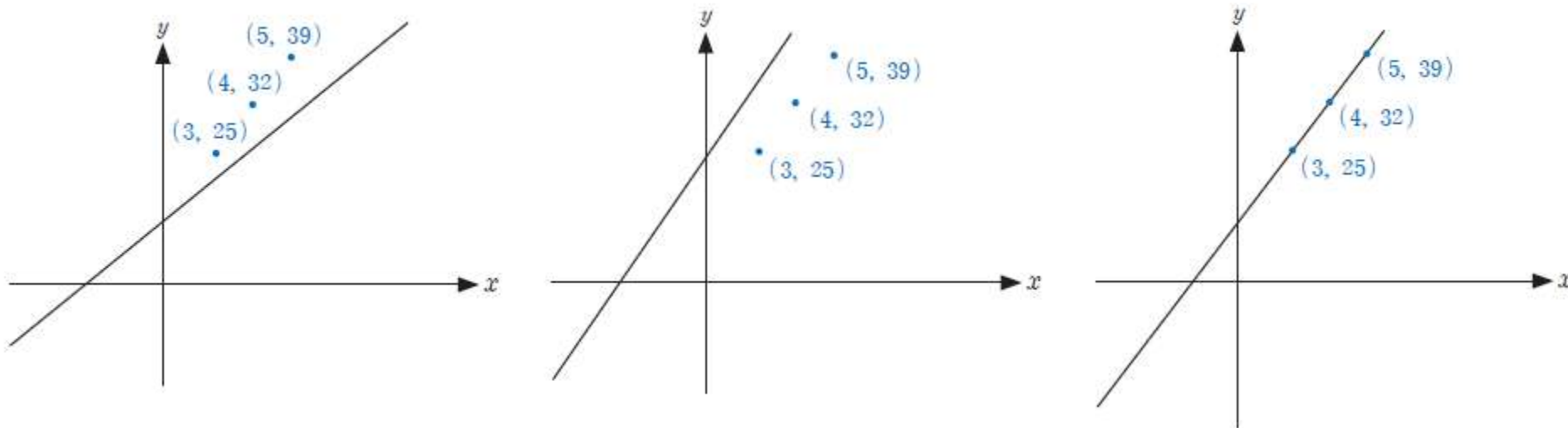
## 03-1 선형 회귀에 대해 알아보고 데이터를 준비합니다

선형 회귀는 아주 간단한 1차 함수로 표현 가능



## 03-1 선형 회귀에 대해 알아보고 데이터를 준비합니다

선형 회귀의 문제 해결 과정(기울기와 절편을 조금씩 수정하여 데이터에 맞춤)



## 03-1 선형 회귀에 대해 알아보고 데이터를 준비합니다

문제를 해결하기 위해 가장 먼저 해야 할 일?

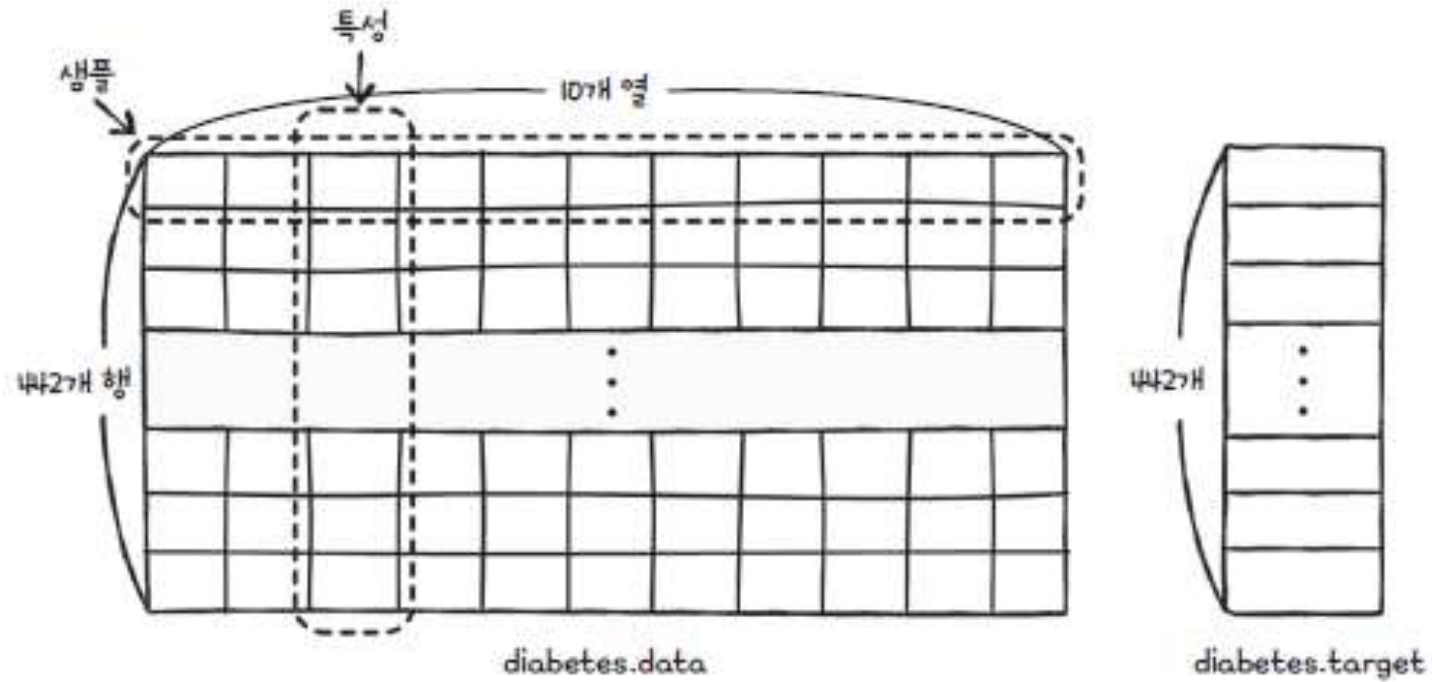
충분한 양의 입력 데이터와 타겟 데이터를 준비(당뇨병 데이터)!

```
from sklearn.datasets import load_diabetes  
diabetes = load_diabetes( )
```

## 03-1 선형 회귀에 대해 알아보고 데이터를 준비합니다

shape 속성으로 데이터 살펴보기

```
print(diabetes.data.shape, diabetes.target.shape)  
(442, 10) (442,)
```



## 03-1 선형 회귀에 대해 알아보고 데이터를 준비합니다

data 속성에 있는 입력, 타깃 데이터 자세히 살펴보기

```
diabetes.data[0:3]
```

```
array([[ 0.03807591,  0.05068012,  0.06169621,  0.02187235, -0.0442235 ,  
        -0.03482076, -0.04340085, -0.00259226,  0.01990842, -0.01764613],  
       [-0.00188202, -0.04464164, -0.05147406, -0.02632783, -0.00844872,  
        -0.01916334,  0.07441156, -0.03949338, -0.06832974, -0.09220405],  
       [ 0.08529891,  0.05068012,  0.04445121, -0.00567061, -0.04559945,  
        -0.03419447, -0.03235593, -0.00259226,  0.00286377, -0.02593034]])
```

네 번째 특성의 값입니다.

첫 번째 샘플입니다.

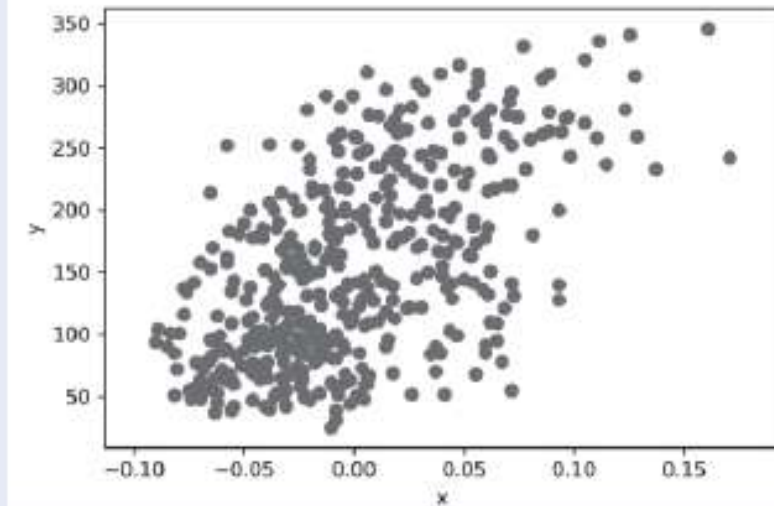
```
diabetes.target[:3]
```

```
array([151., 75., 141.])
```

## 03-1 선형 회귀에 대해 알아보고 데이터를 준비합니다

data 속성에 있는 데이터 시각화하기

```
import matplotlib.pyplot as plt
plt.scatter(diabetes.data[:, 2], diabetes.target)
plt.xlabel('x')
plt.ylabel('y')
plt.show( )
```



## 03-1 선형 회귀에 대해 알아보고 데이터를 준비합니다

훈련 데이터 준비하기

```
x = diabetes.data[:, 2]  
y = diabetes.target
```

- 03-1 선형 회귀에 대해 알아보고 데이터를 준비합니다
- **03-2 경사 하강법으로 학습하는 방법을 알아봅니다**
- 03-3 손실 함수와 경사 하강법의 관계를 알아봅니다
- 03-4 선형 회귀를 위한 뉴런을 만듭니다



## 03-2 경사 하강법으로 학습하는 방법을 알아봅니다

### 선형 회귀와 경사 하강법의 관계

#### 선형 회귀의 목표

입력 데이터( $x$ )와 타겟 데이터( $y$ )를 통해 기울기( $a$ )와 절편( $b$ )을 찾는 것

#### 경사 하강법

모델이 데이터를 잘 표현할 수 있도록 기울기(변화율)를 사용하여 모델을 조금씩 조정하는 최적화 알고리즘

### 예측값과 변화율

#### 예측값( $\hat{y}$ )

모델을 통해 예측한 값(ex.  $y=7x+4$ 라는 모델의  $x$ 에 7을 넣으면 53이 예측 됨)

$$\hat{y} = \omega x + b$$

#### 변화율

모델의 가중치와 절편을 **효율적**으로 업데이트시키기 위한 값

### 예측값으로 올바른 모델 찾기

#### 훈련 데이터에 잘와 맞는 $\omega$ 와 $\mathbf{b}$ 찾기

1. 무작위로  $\omega$ 와  $\mathbf{b}$ 를 정한다.(무작위로 모델 만들기)
2.  $x$ 에서 샘플 하나를 선택하여  $y_{\text{hat}}$ 을 계산한다.(무작위로 모델 예측하기)
3.  $y_{\text{hat}}$ 과 선택한 샘플의 진짜  $y$ 를 비교한다(예측한 값과 진짜 정답 비교하기, 틀릴 확률 99%)
4.  $y_{\text{hat}}$ 과  $y$ 가 더 가까워지도록  $\omega$ 와  $\mathbf{b}$ 를 조정한다.(모델 조정하기)
5. 모든 샘플을 처리할 때까지 다시 2~4를 반복한다.

### 예측값으로 올바른 모델 찾기 - 1

1.  $\omega$ 와  $b$ 를 초기화하기

$\omega = 1.0$

$b = 1.0$

2. 훈련 데이터의 첫번째 샘플 데이터로  $y_{\text{hat}}$  얻기

$y_{\text{hat}} = x[0] * w + b$

`print(y_hat)`

1.0616962422

### 예측값으로 올바른 모델 찾기 - 2

3. 타겟과 예측 데이터 비교하기

```
print(y[0])
```

151.0

4. w값 조절해 예측값 바꾸기(0.1만 증가)

```
W_inc = w + 0.1
```

```
Y_hat_inc = x[0] * w_inc + b
```

```
print(y_hat_inc)
```

1.068293475892

### 예측값으로 올바른 모델 찾기 - 3

5. w값 조정한 후 예측값 증가 정도 확인하기

```
w_rate = (y_hat_inc - y_hat) / (w_inc - w)
```

```
print(w_rate)
```

```
0.06169626571349857423
```



x[0]에 대한 w의 변화율

### 예측값으로 올바른 모델 찾기 - 정리

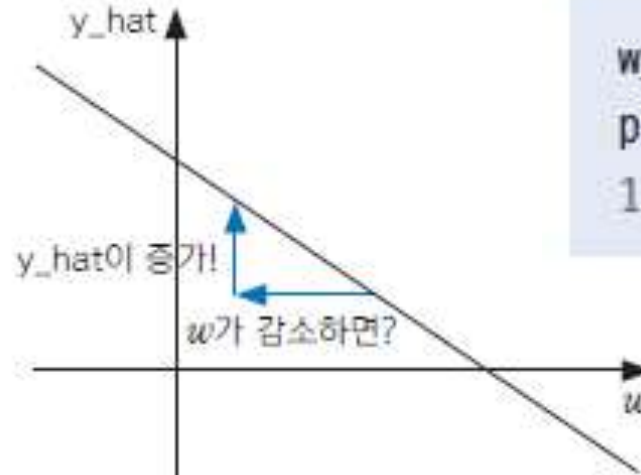
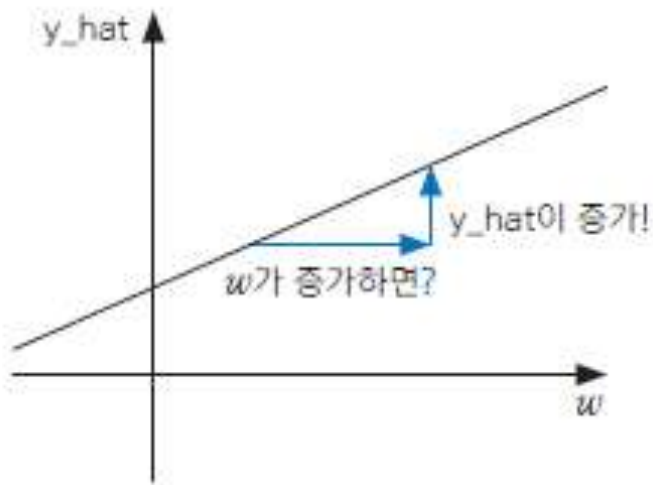
변화율은 결국 훈련데이터의 첫번째 샘플인  $x[0]$ 인 것을 알 수 있음

$$\begin{aligned}w\_rate &= \frac{y\_hat\_inc - y\_hat}{w\_inc - w} = \frac{(x[0] * w\_inc + b) - (x[0] * w + b)}{w\_inc - w} \\&= \frac{(x[0] * (w + 0.1) - w)}{(w + 0.1) - w} = x[0]\end{aligned}$$

1.  $y\_hat$ 의 값은  $y$ 보다 작으므로  $y\_hat$ 의 값을 증가시켜야 함
2. 변화율은 양수이므로  $w$ 값을 증가시키면  $y\_hat$ 값을 증가시킬수 있음(예, 0.1정도)
3. 그런데 만약 변화율이 음수일때  $y\_hat$ 의 값을 증가시켜야 한다면 어떻게 해야 할까?
4.  $w$ 값을 감소시키면 됨(예, 0.1정도)
5. 이방법은 변화율이 양수일때와 음수일때를 구분해야하므로 번거로움

## 03-2 경사 하강법으로 학습하는 방법을 알아봅시다

변화율은 단순한 방법으로 가중치를 업데이트할 수 있게 만들어 줍니다



```
w_new = w + w_rate  
print(w_new)  
1.0616962065186888
```

변화율이 0보다 크거나 작은 경우 모두 가중치에 변화율을 더하기(+)만 하면 OK



## 03-2 경사 하강법으로 학습하는 방법을 알아봅니다

### 변화율로 절편 업데이트 하기

절편 b에 대한 변화율 구한 후 변화율로 b를 업데이트

```
b_inc = b + 0.1
```

```
y_hat_inc = x[0] * w + b_inc
```

```
print(y_hat_inc)
```

```
1.161698574738234
```

```
b_rate = (y_hat_inc - y_hat) / (b_inc - b)
```

```
print(b_rate)
```

```
1.0
```

$$\begin{aligned} b\_rate &= \frac{y\_hat\_inc - y\_hat}{b\_inc - b} = \frac{(x[0] * w + b\_inc) - (x[0] * w + b)}{b\_inc - b} \\ &= \frac{(b + 0.1) - b}{(b + 0.1) - b} = 1 \end{aligned}$$

```
b_new = b + 1
```

```
print(b_new)
```

```
1.0
```


## 03-2 경사 하강법으로 학습하는 방법을 알아봅니다

가중치 변화율과 절편 변화율 공식에 의해  
가중치 변화율 ==  $x[0]$ , 절편 변화율 == 1임을 알 수 있음

## 03-2 경사 하강법으로 학습하는 방법을 알아봅시다


가중치와 절편을 더 적절하게 업데이트하려면?  
y-hat과 y의 차이를 변화율에 곱하면 된다!

```
err = y[0] - y_hat
w_new = w + w_rate * err
b_new = b + 1 * err
print(w_new, b_new)
```



10.250624555904514 150.9383037934813

```
y_hat = x[1] * w_new + b_new
err = y[1] - y_hat
w_rate = x[1]
w_new = w_new + w_rate * err
b_new = b_new + 1 * err
print(w_new, b_new)
```



14.132317616381767 75.52764127612664

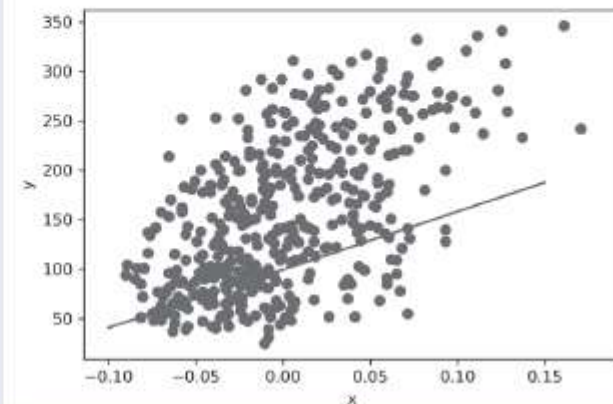
## 03-2 경사 하강법으로 학습하는 방법을 알아봅니다

전체 샘플에 가중치, 절편 업데이트(erro를 변화율에 곱하여)한 결과  
이를 '1 에포크동안 훈련시켰다'라고 함

```
for x_i, y_i in zip(x, y):  
    y_hat = x_i * w + b  
    err = y_i - y_hat  
    w_rate = x_i  
    w = w + w_rate * err  
    b = b + 1 * err  
print(w, b)
```

```
587.8654539985689 99.40935564531424
```

```
plt.scatter(x, y)  
pt1 = (-0.1, -0.1 * w + b)  
pt2 = (0.15, 0.15 * w + b)  
plt.plot([pt1[0], pt2[0]], [pt1[1], pt2[1]])  
plt.xlabel('x')  
plt.ylabel('y')  
plt.show()
```

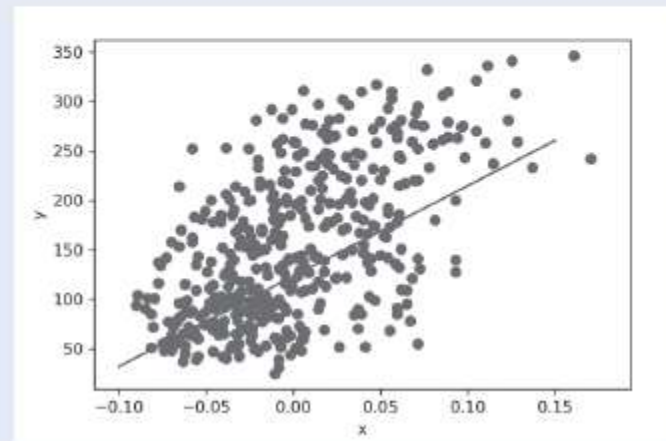


## 03-2 경사 하강법으로 학습하는 방법을 알아봅시다

100 에포크 동안 훈련시킨 결과

```
for i in range(1, 100):  
    for x_i, y_i in zip(x, y):  
        y_hat = x_i * w + b  
        err = y_i - y_hat  
        w_rate = x_i  
        w = w + w_rate * err  
        b = b + 1 * err  
print(w, b)  
913.5973364345905 123.39414383177204
```

```
plt.scatter(x, y)  
pt1 = (-0.1, -0.1 * w + b)  
pt2 = (0.15, 0.15 * w + b)  
plt.plot([pt1[0], pt2[0]], [pt1[1], pt2[1]])  
plt.xlabel('x')  
plt.ylabel('y')  
plt.show()
```

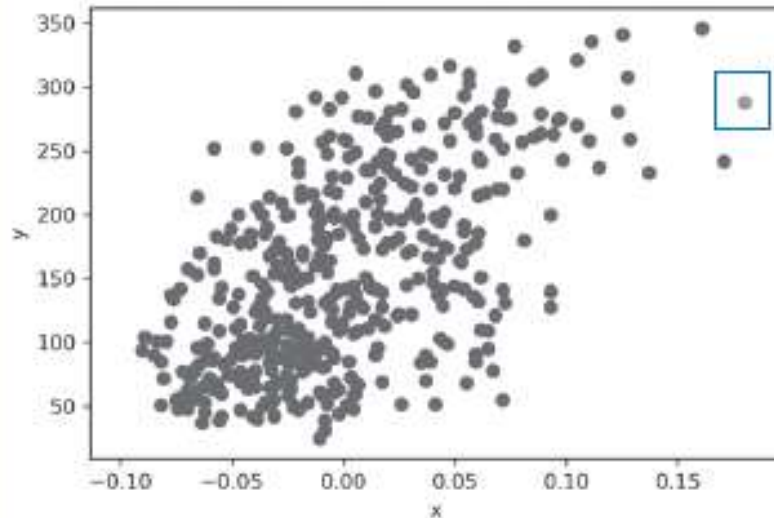


## 03-2 경사 하강법으로 학습하는 방법을 알아봅니다

100 에포크 동안 훈련시킨 모델 + 모델로 예측값 출력

$$\hat{y} = 913.6x + 123.4$$

```
x_new = 0.18  
y_pred = x_new * w + b  
print(y_pred)  
287.8416643899983
```



- 03-1 선형 회귀에 대해 알아보고 데이터를 준비합니다
- 03-2 경사 하강법으로 학습하는 방법을 알아봅니다
- **03-3 손실 함수와 경사 하강법의 관계를 알아봅니다**
- 03-4 선형 회귀를 위한 뉴런을 만듭니다

## 03-3 손실 함수와 경사 하강법의 관계를 알아보니다

경사 하강법의 기술적 표현은 '어떤 손실 함수(loss function)가 정의되었을 때 손실 함수의 값이 최소가 되는 지점을 찾아가는 방법'  
제공 오차는 많은 손실 함수 중 1가지 함수

$$SE = (y - \hat{y})^2$$



## 03-3 손실 함수와 경사 하강법의 관계를 알아봅시다

경사 하강법에서는 손실 함수를 가중치나 절편에 대해 편미분한 다음, 변화율에서 빼는 방법을 사용함

$$\frac{\partial SE}{\partial w} = \frac{\partial}{\partial w} (y - \hat{y})^2 = 2(y - \hat{y}) \left( -\frac{\partial}{\partial w} \hat{y} \right) = 2(y - \hat{y})(-x) = -2(y - \hat{y})x \quad (\text{손실 함수(제곱 오차)를 가중치에 대해 편미분})$$

$$\frac{\partial SE}{\partial b} = \frac{\partial}{\partial b} \frac{1}{2} (y - \hat{y})^2 = (y - \hat{y}) \left( -\frac{\partial}{\partial b} \hat{y} \right) = (y - \hat{y})(-1) = -(y - \hat{y})1 \quad (\text{손실 함수(제곱 오차)를 절편에 대해 편미분})$$

$$w = w - \frac{\partial SE}{\partial w} = w + (y - \hat{y})x \quad (\text{가중치에서 편미분 결과를 뺌})$$

$$b = b - \frac{\partial SE}{\partial b} = b + (y - \hat{y}) \quad (\text{절편에서 편미분 결과를 뺌})$$

## 03-3 손실 함수와 경사 하강법의 관계를 알아봅시다

경사 하강법에서는 손실 함수를 가중치나 절편에 대해 편미분한 다음, 변화율에서 빼는 방법을 사용함

$$\frac{\partial SE}{\partial w} = \frac{\partial}{\partial w} (y - \hat{y})^2 = 2(y - \hat{y}) \left( -\frac{\partial}{\partial w} \hat{y} \right) = 2(y - \hat{y})(-x) = -2(y - \hat{y})x \quad (\text{손실 함수(제공 오차)를 가중치에 대해 편미분})$$

$$\frac{\partial SE}{\partial b} = \frac{\partial}{\partial b} \frac{1}{2} (y - \hat{y})^2 = (y - \hat{y}) \left( -\frac{\partial}{\partial b} \hat{y} \right) = (y - \hat{y})(-1) = -(y - \hat{y})1 \quad (\text{손실 함수(제공 오차)를 절편에 대해 편미분})$$

$$w = w - \frac{\partial SE}{\partial w} = w + (y - \hat{y})x \quad (\text{가중치에서 편미분 결과를 뺌})$$

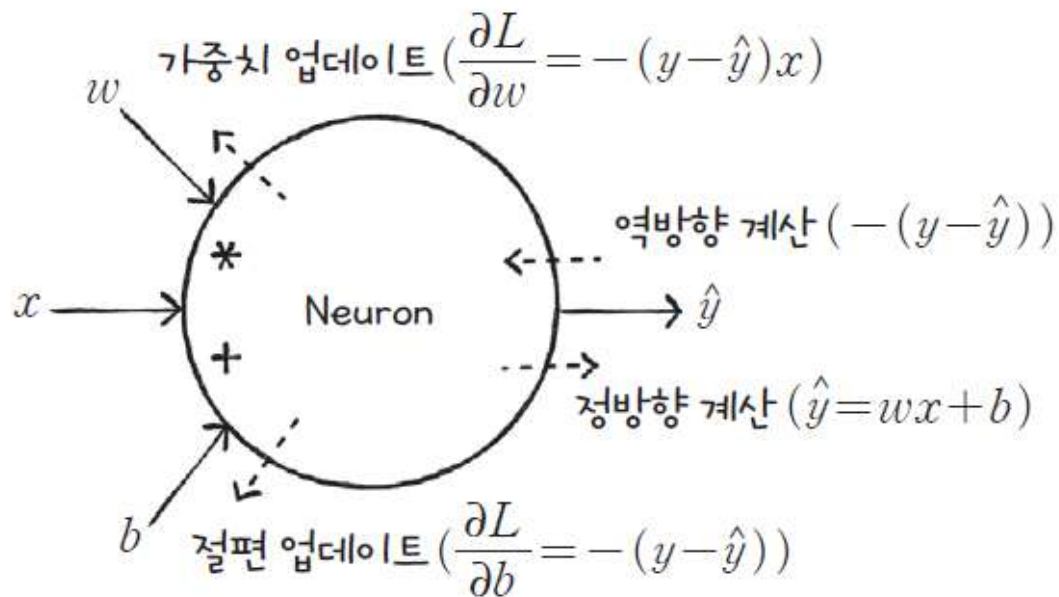
$$b = b - \frac{\partial SE}{\partial b} = b + (y - \hat{y}) \quad (\text{절편에서 편미분 결과를 뺌})$$

- 03-1 선형 회귀에 대해 알아보고 데이터를 준비합니다
- 03-2 경사 하강법으로 학습하는 방법을 알아봅니다
- 03-3 손실 함수와 경사 하강법의 관계를 알아봅니다
- **03-4 선형 회귀를 위한 뉴런을 만듭니다**

## 03-4 선형 회귀를 위한 뉴런을 만듭니다

정방향 계산 ==  $\hat{y}$ 을 구함

역방향 계산 == 그래디언트(변화율)을 구함



## 03-4 선형 회귀를 위한 뉴런을 만듭니다

```
class Neuron:

    def __init__(self):
        self.w = 1.0    # 가중치를 초기화합니다
        self.b = 1.0    # 절편을 초기화합니다

    def forpass(self, x):
        y_hat = x * self.w + self.b    # 직선 방정식을 계산합니다
        return y_hat

    def backprop(self, x, err):
        w_grad = x * err    # 가중치에 대한 그래디언트를 계산합니다
        b_grad = 1 * err    # 절편에 대한 그래디언트를 계산합니다
        return w_grad, b_grad

    def fit(self, x, y, epochs=100):
        for i in range(epochs):    # 에포크만큼 반복합니다
            for x_i, y_i in zip(x, y):    # 모든 샘플에 대해 반복합니다
                y_hat = self.forpass(x_i)    # 정방향 계산
                err = -(y_i - y_hat)    # 오차 계산
                w_grad, b_grad = self.backprop(x_i, err)    # 역방향 계산
                self.w -= w_grad    # 가중치 업데이트
                self.b -= b_grad    # 절편 업데이트
```

```
neuron = Neuron()
neuron.fit(x, y)
```

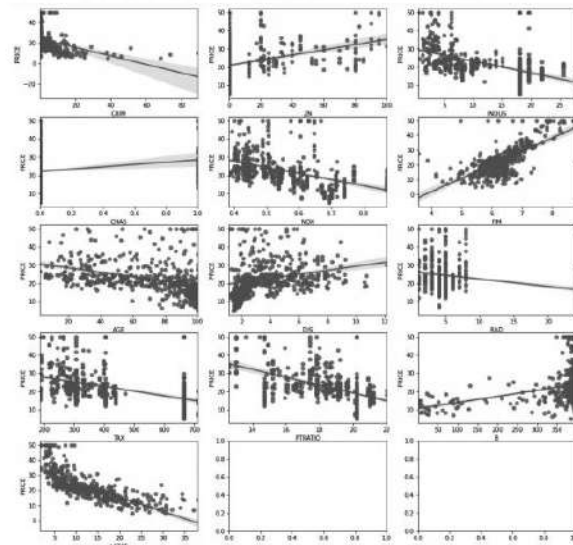
```
plt.scatter(x, y)
pt1 = (-0.1, -0.1 * neuron.w + neuron.b)
pt2 = (0.15, 0.15 * neuron.w + neuron.b)
plt.plot([pt1[0], pt2[0]], [pt1[1], pt2[1]])
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

# 머신러닝 실습 — 수치 예측 (보스톤 주택가격)

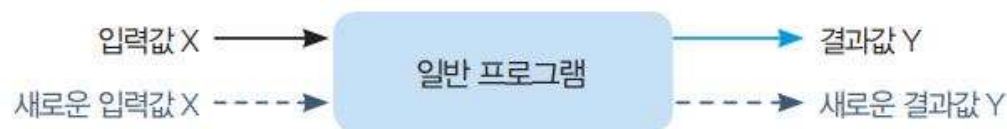


환경에 따른 주택 가격 예측하기	
목표	보스턴 주택 가격 데이터에 머신러닝 기반의 회귀 분석을 수행하여 주택 가격에 영향을 미치는 환경 변수를 확인하고, 그에 따른 주택 가격을 예측한다.
핵심 개념	머신러닝, 머신러닝 프로세스, 지도 학습, 사이킷런, 사이킷런의 내장 데이터셋, 분석 평가 지표
데이터 수집	보스턴 주택 가격 데이터: 사이킷런 내장 데이터셋
데이터 준비 및 탐색	1. 사이킷런 데이터셋 확인: <code>boston.DESCR</code> 2. 사이킷런 데이터셋에 지정된 X 피처와 타겟 피처 결합
분석 모델 구축	사이킷런의 선형 회귀 모델 구축
결과 시각화	

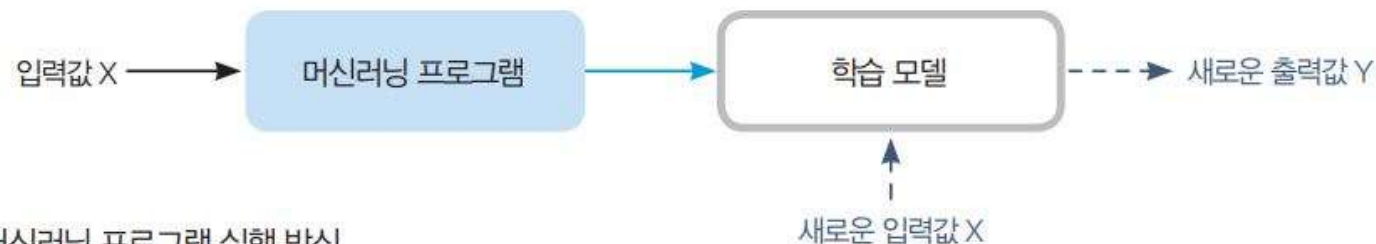
데이터가 주택 가격에 미치는 영향을 산점도와 선형 회귀 그래프로 시각화



보스턴 주택 가격 데이터에 머신러닝 기반의 회귀 분석을 수행  
주택 가격에 영향을 미치는 변수를 확인하고 그 값에 따른 주택 가격을 예측



(a) 일반 프로그램 실행 방식



(b) 머신러닝 프로그램 실행 방식

그림 10-1 일반 프로그램과 머신러닝 프로그램 실행 방식 비교



- 핵심 개념 이해

- 머신러닝 프로세스

- 데이터 수집 → 데이터 전처리 및 훈련/테스트 데이터 분할 → 모델 구축 및 학습 → 모델 평가 → 예측

- 지도 학습

- 학습을 하기 위한 훈련 데이터에 입력과 출력을 같이 제공
    - 문제(입력)에 대한 답(출력, 결과값)을 아는 상태에서 학습하는 방식
    - 입력: 예측 변수, 속성, 특징
    - 출력: 반응 변수, 목표 변수, 클래스, 레이블

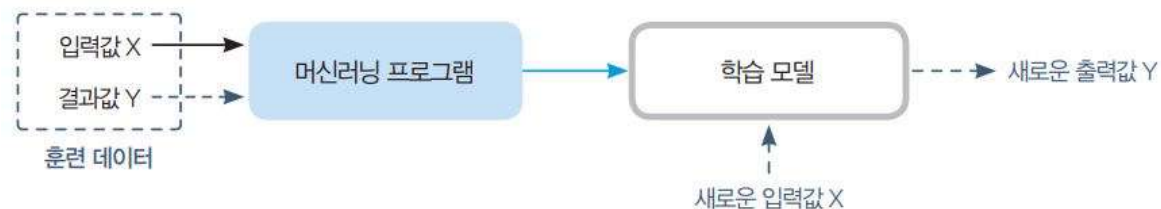


그림 10-2 머신러닝의 지도 학습 방식

- 사이킷런

- 파이썬으로 머신러닝을 수행하기 위한 쉽고 효율적인 개발 라이브러리를 제공
    - 보스턴 주택 가격 데이터, 붓꽃 데이터 등과 같은 머신러닝 분석용 데이터셋 을 제공
    - 전체 n개의 컬럼 중 앞에서 (n-1)개의 컬럼은 독립 변수 X를 의미
    - 마지막 컬럼 은 종속 변수 Y이며, 데이터셋 객체의 target 배열로 관리

COLAB에서 '10장\_주택가격분석'으로 노트북 페이지를 추가하고 입력

In [1]:	<pre>!pip install sklearn</pre>
In [2]:	<pre>import <b>numpy</b> as <b>np</b> import <b>pandas</b> as <b>pd</b>  from <b>sklearn.datasets</b> import <b>load_boston</b> boston = <b>load_boston()</b></pre>

# 선형회귀 실습

- 데이터 수집, 준비 및 탐색
  - 데이터가 이미 정리된 상태이므로 데이터셋 구성을 확인

In [3]:	print(boston.DESCR)																																																																																										
In [4]:	boston_df = pd.DataFrame(boston.data, columns = boston.feature_names) boston_df.head()																																																																																										
Out[4]:	<table><tr><th></th><th>CRIM</th><th>ZN</th><th>INDUS</th><th>CHAS</th><th>NOX</th><th>RM</th><th>AGE</th><th>DIS</th><th>RAD</th><th>TAX</th><th>PTRATIO</th><th>B</th><th>LSTAT</th></tr><tr><td>0</td><td>0.00632</td><td>18.0</td><td>2.31</td><td>0.0</td><td>0.538</td><td>6.575</td><td>65.2</td><td>4.0900</td><td>1.0</td><td>296.0</td><td>15.3</td><td>396.90</td><td>4.98</td></tr><tr><td>1</td><td>0.02731</td><td>0.0</td><td>7.07</td><td>0.0</td><td>0.469</td><td>6.421</td><td>78.9</td><td>4.9671</td><td>2.0</td><td>242.0</td><td>17.8</td><td>396.90</td><td>9.14</td></tr><tr><td>2</td><td>0.02729</td><td>0.0</td><td>7.07</td><td>0.0</td><td>0.469</td><td>7.185</td><td>61.1</td><td>4.9671</td><td>2.0</td><td>242.0</td><td>17.8</td><td>392.83</td><td>4.03</td></tr><tr><td>3</td><td>0.03237</td><td>0.0</td><td>2.18</td><td>0.0</td><td>0.458</td><td>6.998</td><td>45.8</td><td>6.0622</td><td>3.0</td><td>222.0</td><td>18.7</td><td>394.63</td><td>2.94</td></tr><tr><td>4</td><td>0.06905</td><td>0.0</td><td>2.18</td><td>0.0</td><td>0.458</td><td>7.147</td><td>54.2</td><td>6.0622</td><td>3.0</td><td>222.0</td><td>18.7</td><td>396.90</td><td>5.33</td></tr></table>		CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33						
	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT																																																																														
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98																																																																														
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14																																																																														
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03																																																																														
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94																																																																														
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33																																																																														
In [5]:	boston_df['PRICE'] = boston.target boston_df.head()																																																																																										
Out[5]:	<table><tr><th></th><th>CRIM</th><th>ZN</th><th>INDUS</th><th>CHAS</th><th>NOX</th><th>RM</th><th>AGE</th><th>DIS</th><th>RAD</th><th>TAX</th><th>PTRATIO</th><th>B</th><th>LSTAT</th><th>PRICE</th></tr><tr><td>0</td><td>0.00632</td><td>18.0</td><td>2.31</td><td>0.0</td><td>0.538</td><td>6.575</td><td>65.2</td><td>4.0900</td><td>1.0</td><td>296.0</td><td>15.3</td><td>396.90</td><td>4.98</td><td>24.0</td></tr><tr><td>1</td><td>0.02731</td><td>0.0</td><td>7.07</td><td>0.0</td><td>0.469</td><td>6.421</td><td>78.9</td><td>4.9671</td><td>2.0</td><td>242.0</td><td>17.8</td><td>396.90</td><td>9.14</td><td>21.6</td></tr><tr><td>2</td><td>0.02729</td><td>0.0</td><td>7.07</td><td>0.0</td><td>0.469</td><td>7.185</td><td>61.1</td><td>4.9671</td><td>2.0</td><td>242.0</td><td>17.8</td><td>392.83</td><td>4.03</td><td>34.7</td></tr><tr><td>3</td><td>0.03237</td><td>0.0</td><td>2.18</td><td>0.0</td><td>0.458</td><td>6.998</td><td>45.8</td><td>6.0622</td><td>3.0</td><td>222.0</td><td>18.7</td><td>394.63</td><td>2.94</td><td>33.4</td></tr><tr><td>4</td><td>0.06905</td><td>0.0</td><td>2.18</td><td>0.0</td><td>0.458</td><td>7.147</td><td>54.2</td><td>6.0622</td><td>3.0</td><td>222.0</td><td>18.7</td><td>396.90</td><td>5.33</td><td>36.2</td></tr></table>		CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	PRICE	0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0	1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6	2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7	3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4	4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2
	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	PRICE																																																																													
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0																																																																													
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6																																																																													
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7																																																																													
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4																																																																													
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2																																																																													

In [3]: 데이터셋에 대한 설명 `boston.DESCR` 을 확인

In [4]: 데이터셋 객체의 `data` 배열 `boston.data`, 즉 독립 변수 X가 되는 피쳐들을 `DataFrame` 자료형으로 변환하여 `boston_df`를 생성  
boston\_df의 데이터 5개를 확인 `boston_df.head()`

In [5]: 데이터셋 객체의 `target` 배열 `boston.target`, 즉 종속 변수인 주택 가격('PRICE') 컬럼을 `boston_df`에 추가  
boston\_df의 데이터 5개를 확인 `boston_df.head()`

- 데이터 수집, 준비 및 탐색
  - 2. 데이터가 이미 정리된 상태이므로 데이터셋 구성을 확인

In [6]:	print('보스턴 주택 가격 데이터셋 크기: ', boston_df.shape)
Out[6]:	보스턴 주택 가격 데이터셋 크기: (506, 14)
In [7]:	boston_df.info()
Out[7]:	<class 'pandas.core.frame.DataFrame'> RangeIndex: 506 entries, 0 to 505 Data columns (total 14 columns): CRIM      506 non-null float64 ZN        506 non-null float64 INDUS     506 non-null float64 CHAS      506 non-null float64 NOX       506 non-null float64 RM        506 non-null float64 AGE       506 non-null float64 DIS       506 non-null float64 RAD       506 non-null float64 TAX       506 non-null float64 PTRATIO   506 non-null float64 B          506 non-null float64 LSTAT     506 non-null float64 PRICE     506 non-null float64 dtypes: float64(14) memory usage: 55.4KB

In [6]: 데이터셋의 형태 `boston_df.shape`, 즉 행의 개수(데이터 개수)와 열의 개수(변수 개수)를 확인  
행의 개수가 506이므로 데이터가 506개 있으며, 열의 개수가 14이므로 변수가 14개 있음  
변수 중에서 13개는 독립 변수 X가 되고, 마지막 변수 'PRICE'는 종속 변수 Y가 됨

In [7]: `boston_df`에 대한 정보를 확인 `boston_df.info()`

## • 14개의 독립 변수(피처)의 의미

- CRIM: 지역별 범죄 발생률
- ZN: 25,000평방피트를 초과하는 거주 지역 비율
- INDUS: 비산업 지역의 넓이 비율
- CHAS: 찰스강의 더미변수(1은 강에 인접, 0은 강에 인접 아님)
- NOX: 일산화질소 농도
- RM: 거주할 수 있는 방 개수
- AGE: 1940년 이전에 건축된 주택 비율
- DIS: 5개 주요 고용센터까지 가중 거리
- RAD: 고속도로 접근 용이도
- TAX: 10,000달러당 재산세 비율
- PTRATIO: 지역의 교사와 학생 수 비율
- B: 지역의 흑인 거주 비율
- LSTAT: 하위 계층의 비율
- PRICE(MEDV): 본인 소유 주택 가격의 중앙값

# 선형회귀 실습

- 분석 모델 구축, 결과 분석 및 시각화

1. 선형 회귀를 이용해 분석 모델 구축하기

1. 사이킷런의 선형 분석 모델 패키지 `sklearn.linear_model` 에서 선형 회귀 `LinearRegression` 를 이용하여 분석 모델을 구축

In [8]:	<pre>from sklearn.linear_model import <b>LinearRegression</b> from sklearn.model_selection import <b>train_test_split</b> from sklearn.metrics import <b>mean_squared_error, r2_score</b></pre>
In [9]:	<pre>#X, Y 분할하기 Y = boston_df['PRICE'] X = boston_df.<b>drop</b>(['PRICE'], axis = 1, inplace = False)</pre>
In [10]:	<pre>#훈련용 데이터와 평가용 데이터 분할하기 X_train, X_test, Y_train, Y_test = <b>train_test_split</b>(X, Y, test_size = 0.3, random_state = 156)</pre>

In [8]: 사이킷런을 사용하여 머신러닝 회귀 분석을 하기 위한 `LinearRegression`과 데이터셋 분리 작업을 위한 `train_test_split`, 성능 측정을 위한 평가 지표인 `mean_squared_error`, `r2_score`를 임포트

In [9]: PRICE 피처를 회귀식의 종속 변수 Y로 설정하고 PRICE를 제외 `drop()` 한 나머지 피처를 독립 변수 X로 설정

In [10]: X와 Y 데이터 506개를 학습 데이터와 평가 데이터로 7:3 비율로 분할 `test_size=0.3`

# 선형회귀 실습

- 분석 모델 구축, 결과 분석 및 시각화

1. 선형 회귀를 이용해 분석 모델 구축하기

1. 사이킷런의 선형 분석 모델 패키지 `sklearn.linear_model` 에서 선형 회귀 `LinearRegression` 를 이용하여 분석 모델을 구축

In [11]:	#선형 회귀 분석 : 모델 생성 <code>lr = LinearRegression()</code>
In [12]:	#선형 회귀 분석 : 모델 훈련 <code>lr.fit(X_train, Y_train)</code>
Out[12]:	<code>LinearRegression()</code>
In [13]:	#선형 회귀 분석 : 평가 데이터에 대한 예측 수행 -> 예측 결과 <code>Y_predict</code> 구하기 <code>Y_predict = lr.predict(X_test)</code>

In [11]: 선형 회귀 분석 모델 객체 `lr`을 생성

In [12]: 학습 데이터 `X_train` 와 `Y_train` 를 가지고 학습을 수행 `fit()`.

In [13]: 평가 데이터 `X_test` 를 가지고 예측을 수행하여 `predict()` 예측값 `Y_predict` 를 구함

# 선형회귀 실습

- 분석 모델 구축, 결과 분석 및 시각화

1. 선형 회귀를 이용해 분석 모델 구축하기

2. 선형 회귀 분석 모델을 평가 지표를 통해 평가하고 회귀 계수를 확인하여 피처의 영향을 분석

In [14]:	<pre>mse = mean_squared_error(Y_test, Y_predict) rmse = np.sqrt(mse) print('MSE : {0:.3f}, RMSE : {1:.3f}'.format(mse, rmse)) print('R^2(Variance score) : {0:.3f}'.format(r2_score(Y_test, Y_predict)))</pre>
Out[14]:	<pre>MSE : 17.297, RMSE : 4.159 R^2(Variance score) : 0.757</pre>
In [15]:	<pre>print('Y 절편 값: ', lr.intercept_) print('회귀 계수 값: ', np.round(lr.coef_, 1))</pre>
Out[15]:	<pre>Y 절편 값: 40.995595172164336 회귀 계수 값: [-0.1 0.1 0. 3. -19.8 3.4 0. -1.7 0.4 -0. -0.9 0. -0.6]</pre>

In [14]: 회귀 분석은 지도 학습이므로 평가 데이터 X에 대한 결과값  $Y_{test}$ 를 이미 알고 있는 상태에서 평가 데이터  $Y_{test}$ 와

In [13]에서 구한 예측 결과  $Y_{predict}$ 의 오차를 계산하여 모델을 평가. 평가 지표 MSE를 구하고 mean\_squared\_error()  
구한 값의 제곱근을 계산하여 np.sqrt(mse) 평가 지표 RMSE를 구함 그리고 평가 지표 R2 을 구함 r2\_score()

In [15]: 선형 회귀의 Y절편 lr.intercept\_과 각 피처의 회귀 계수 lr.coef\_를 확인

# 선형회귀 실습

- 분석 모델 구축, 결과 분석 및 시각화
  - 선형 회귀를 이용해 분석 모델 구축하기
  - 선형 회귀 분석 모델을 평가 지표를 통해 평가하고 회귀 계수를 확인하여 피처의 영향을 분석

In [16]:	coef = pd.Series(data = np.round(lr.coef_, 2), index = X.columns) coef.sort_values(ascending = False)	
Out[16]:	RM      3.35 CHAS    3.05 RAD     0.36 ZN      0.07 INDUS   0.03 B       0.01 AGE     0.01 TAX     -0.01 CRIM    -0.11 LSTAT   -0.57 PTRATIO -0.92 DIS     -1.74 NOX     -19.80 dtype: float64	

In [16]: 회귀 모델에서 구한 회귀 계수 값 `lr.coef_` 과 피처 이름 `X.columns` 을 묶어서 Series 자료 형으로 만들고, 회귀 계수 값을 기준으로 내림차순으로 정렬하여 `ascending=False` 확인 `sort_values()`

회귀 모델 결과를 토대로 보스톤 주택 가격에 대한 회귀식

$$Y_{\text{PRICE}} = -0.11X_{\text{CRIM}} + 0.07X_{\text{ZN}} + 0.03X_{\text{INDUS}} + 3.05X_{\text{CHAS}} - 19.80X_{\text{NOX}} + 3.35X_{\text{RM}} + 0.01X_{\text{AGE}} - 1.74X_{\text{DIS}} + 0.36X_{\text{RAD}} - 0.01X_{\text{TAX}} - 0.92X_{\text{PTRATIO}} + 0.01X_{\text{B}} - 0.57X_{\text{LSTAT}} + 41.00$$



# 선형회귀 실습

- 회귀 분석 결과를 산점도 + 선형 회귀 그래프로 시각화하기
- 2. 선형 회귀를 이용해 분석 모델 구축하기

In [17]:	<pre>import matplotlib.pyplot as plt import seaborn as sns</pre>
In [18]:	<pre>fig, axs = plt.subplots(figsize = (16, 16), ncols = 3, nrows = 5)  x_features = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT']  for i, feature in enumerate(x_features):     row = int(i/3)     col = i%3     sns.regplot(x = feature, y = 'PRICE', data = boston_df, ax = axs[row][col])</pre>

In [17]: 시각화에 필요한 모듈을 импорт

In [18]: 독립 변수인 13개 피처와 종속 변수인 주택 가격, PRICE와의 회귀 관계를 보여주는 13개 그래프를 subplots()를

사용하여 5행 3열 구조로 모아서 나타냄

aborn의 regplot()은 산점도 그래프와 선형 회귀 그래프를 함께 그려줌

# 선형회귀 실습

- 회귀 분석 결과를 산점도 + 선형 회귀 그래프로 시각화하기
- 2. 선형 회귀를 이용해 분석 모델 구축하기

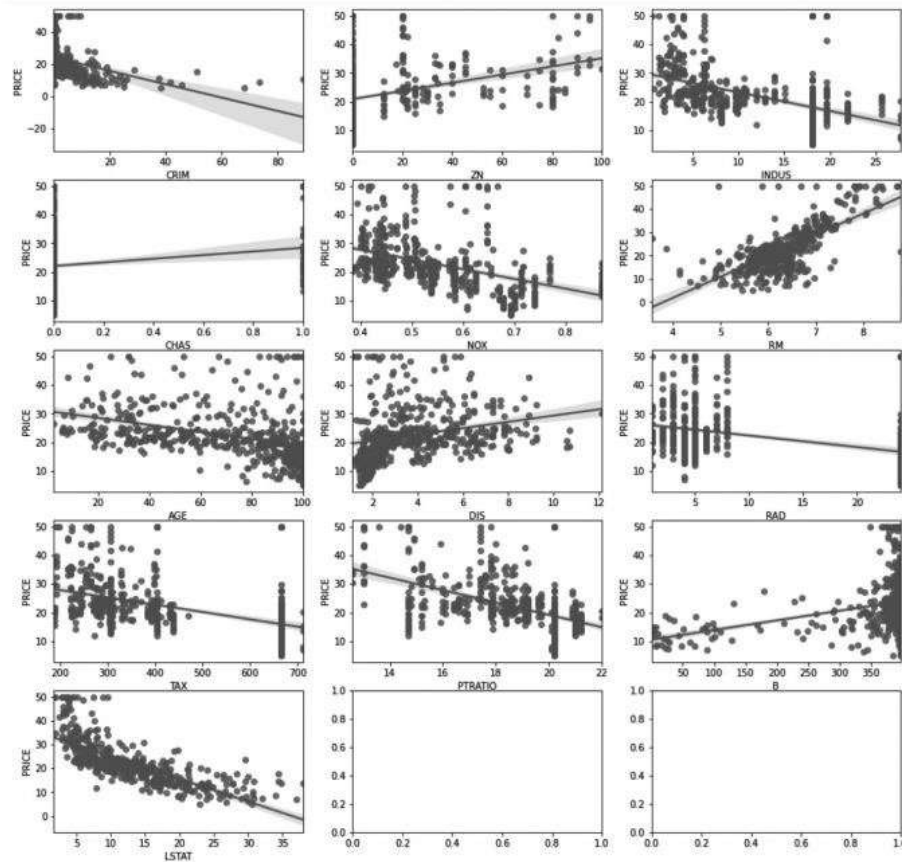


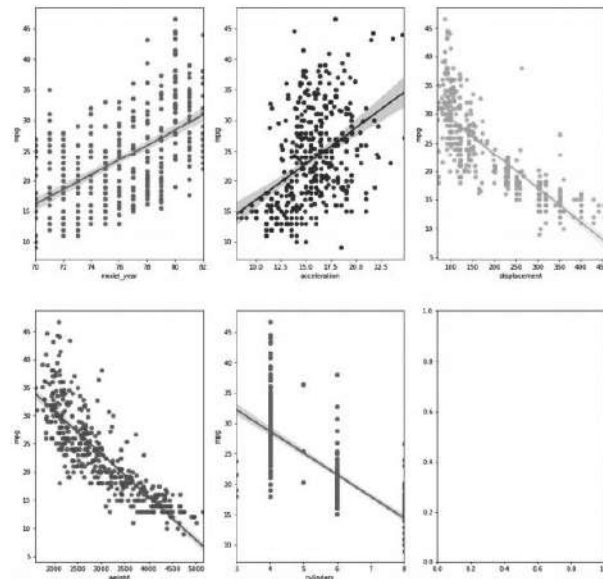
그림 10-3 13개 피처와 주택 가격의 회귀 관계를 나타낸 산점도/선형 회귀 그래프

# 머신러닝 실습 — 수치 예측 (자동차 연비 예측)



항목에 따른 자동차 연비 예측하기	
목표	자동차 연비 데이터에 머신러닝 기반의 회귀 분석을 수행하여 연비에 영향을 미치는 항목을 확인하고 그에 따른 자동차 연비를 예측한다.
핵심 개념	머신러닝, 머신러닝 프로세스, 지도 학습, 사이킷런, 사이킷런의 내장 데이터셋, 분석 평가 지표
데이터 수집	자동차 연비 데이터: UCI Machine Learning Repository에서 다운로드
데이터 준비 및 탐색	1. 필요 없는 컬럼 제거 2. X 변수와 Y 변수 확인
분석 모델 구축	사이킷런의 선형 회귀 모델 구축
결과 시각화	

X 변수와 Y 변수에 대한 산점도 그래프와 선형 회귀 그래프



- 목표설정
  - 목표: 자동차 연비 데이터에 머신러닝 기반의 회귀 분석을 수행  
연비에 영향을 미치는 항목을 확인하고, 그에 따른 자동차 연비를 예측
- 핵심 개념 이해
  - 1절의 프로젝트와 동일한 개념에 대한 이해가 필요

- 데이터 수집

1. 자동차 연비 데이터 다운로드하기

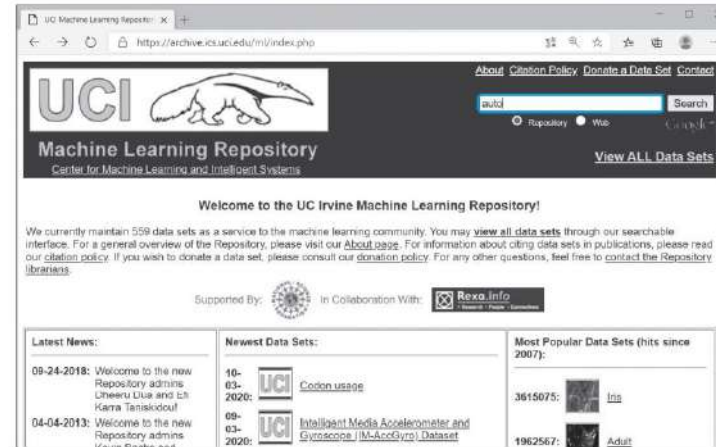


그림 10-4 UCI Machine Learning Repository 사이트에서 'auto' 검색

2. 검색 결과 목록에서 'Auto MPG Data Set - UCI Machine Learning Repository' 클릭

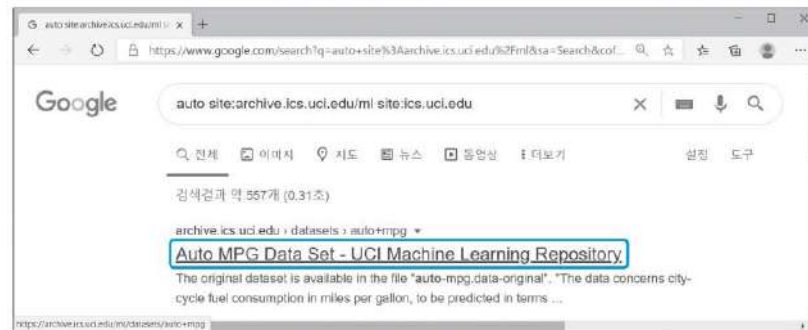
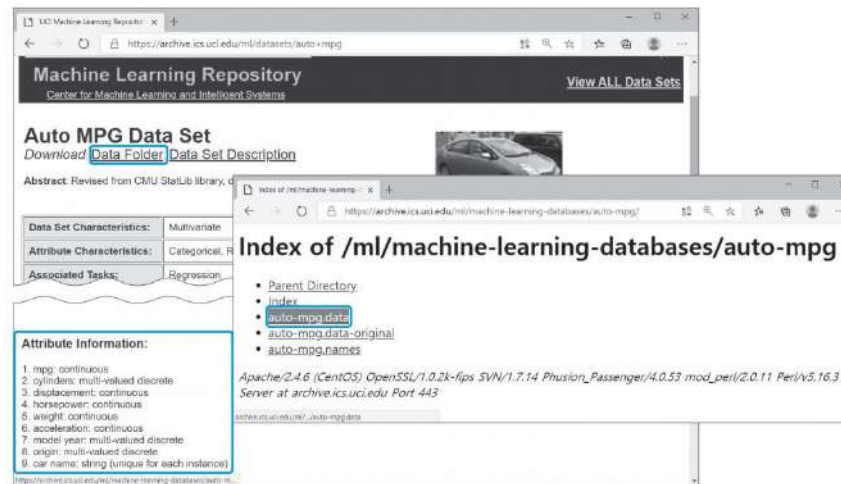


그림 10-5 검색 목록에서 다운로드할 데이터셋 선택

# 선형회귀 실습

## • 데이터 수집

3. Data Folder를 클릭하여 'auto-mpg.data'를 다운로드



4. CSV 파일로 변경하기

그림 10-6 데이터셋 다운로드

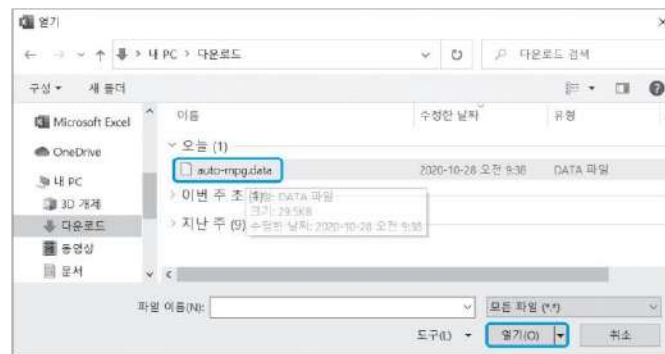


그림 10-7 CSV 파일로 변경하기 1 - 엑셀 프로그램에서 파일 열기

# 선형회귀 실습

## • 데이터 수집

5. 1단계는 버튼을 클릭, 2단계에서는 [구분 기호]로 '공백'을 선택하고 버튼을 클릭



6. 텍스트 마법사 3단계에서 데이터 미리 보기를 확인하고 버튼을 클릭

그림 10-8 CSV 파일로 변경하기 2 - 텍스트 마법사로 파일 정리

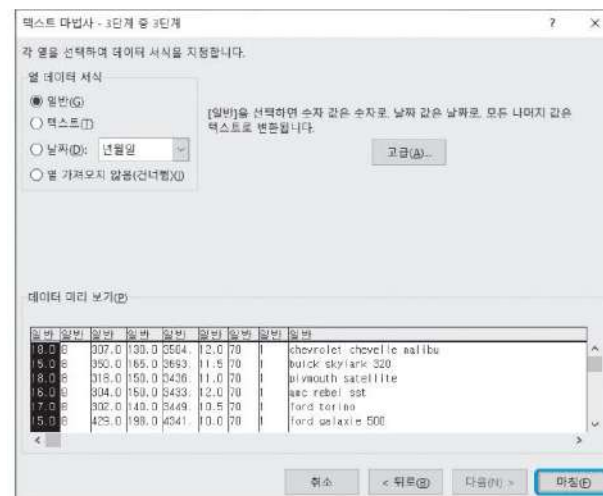


그림 10-9 CSV 파일로 변경하기 3 - 미리보기로 확인 후 텍스트 마법사 종료

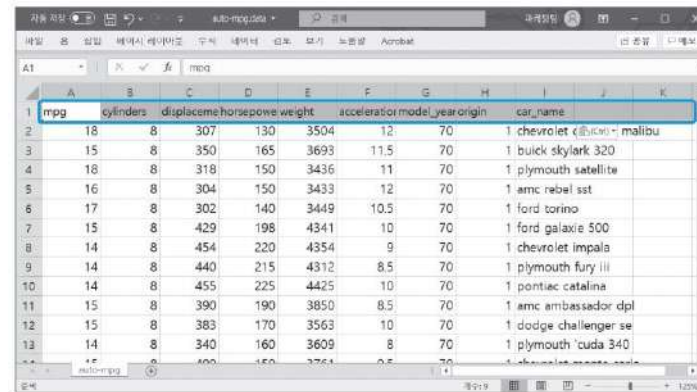


# 선형회귀 실습

## • 데이터 수집

### 7. 항목을 구분하기 위해 열 이름을 추가

- 행을 삽입하고 열 이름으로 mpg, cylinders, displacement, horsepower, weight, acceleration, model\_year, origin, car\_name을 각각 입력



	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name
1	18	8	307	130	3504	12	70	1	chevrolet chevelle malibu
2	15	8	350	165	3693	11.5	70	1	buick skylark 320
3	18	8	318	150	3436	11	70	1	plymouth satellite
4	16	8	304	150	3433	12	70	1	amc rebel sst
5	17	8	302	140	3449	10.5	70	1	ford torino
6	15	8	429	198	4341	10	70	1	ford galaxie 500
7	14	8	454	220	4354	9	70	1	chevrolet impala
8	14	8	440	215	4312	8.5	70	1	plymouth fury iii
9	14	8	455	225	4425	10	70	1	pontiac catalina
10	15	8	390	190	3850	8.5	70	1	amc ambassador dpl
11	15	8	383	170	3563	10	70	1	dodge challenger se
12	14	8	340	160	3609	8	70	1	plymouth cuda 340
13	14	8	360	160	3761	9.5	70	1	chevrolet monte carlo

그림 10-10 CSV 파일로 변경하기 4 - 항목 이름 추가

### 8. My\_Python 폴더에 10장\_data 폴더를 만들고 파일을 'auto-mpg.csv'로 저장

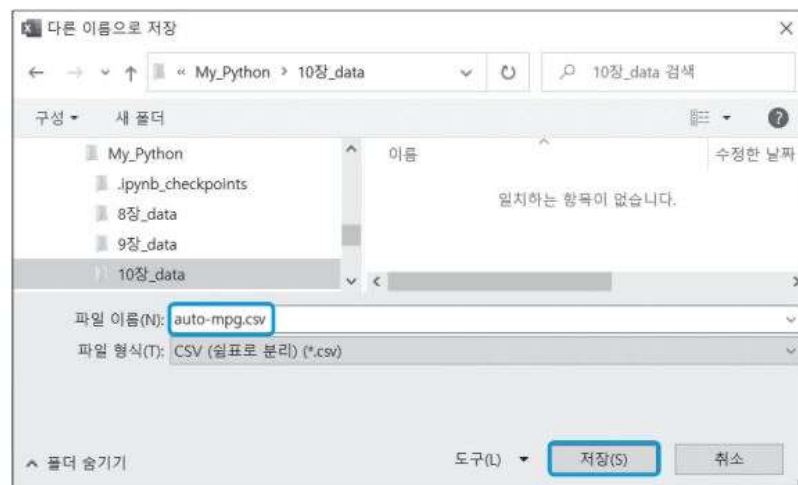


그림 10-11 CSV 파일로 변경하기 4 - CSV 파일 저장

# 선형회귀 실습

- 데이터 준비 및 탐색
  - 분석에 필요 없는 컬럼을 제거하고 데이터셋의 내용을 확인

In [1]:	<pre>import numpy as np import pandas as pd data_df = pd.read_csv('./10장_data/auto-mpg.csv', header = 0, engine = 'python')</pre>																																																												
In [2]:	<pre>print('데이터셋 크기: ', data_df.shape) data_df.head()</pre>																																																												
Out[2]:	<div>데이터셋 크기: (398, 9)</div> <table><tr><th></th><th>mpg</th><th>cylinders</th><th>displacement</th><th>horsepower</th><th>weight</th><th>acceleration</th><th>model_year</th><th>origin</th><th>car_name</th></tr><tr><td>0</td><td>18.0</td><td>8</td><td>307.0</td><td>130</td><td>3504</td><td>12.0</td><td>70</td><td>1</td><td>chevrolet chevelle malibu</td></tr><tr><td>1</td><td>15.0</td><td>8</td><td>350.0</td><td>165</td><td>3693</td><td>11.5</td><td>70</td><td>1</td><td>buick skylark 320</td></tr><tr><td>2</td><td>18.0</td><td>8</td><td>318.0</td><td>150</td><td>3436</td><td>11.0</td><td>70</td><td>1</td><td>plymouth satellite</td></tr><tr><td>3</td><td>16.0</td><td>8</td><td>304.0</td><td>150</td><td>3433</td><td>12.0</td><td>70</td><td>1</td><td>amc rebel sst</td></tr><tr><td>4</td><td>17.0</td><td>8</td><td>302.0</td><td>140</td><td>3449</td><td>10.5</td><td>70</td><td>1</td><td>ford torino</td></tr></table>		mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name	0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu	1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320	2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite	3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst	4	17.0	8	302.0	140	3449	10.5	70	1	ford torino
	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name																																																				
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu																																																				
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320																																																				
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite																																																				
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst																																																				
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino																																																				
In [3]:	<pre>data_df = data_df.drop(['car_name', 'origin', 'horsepower'], axis = 1, inplace = False) data_df.head()</pre>																																																												
Out[3]:	<table><tr><th></th><th>mpg</th><th>cylinders</th><th>displacement</th><th>weight</th><th>acceleration</th><th>model_year</th></tr><tr><td>0</td><td>18.0</td><td>8</td><td>307.0</td><td>3504</td><td>12.0</td><td>70</td></tr><tr><td>1</td><td>15.0</td><td>8</td><td>350.0</td><td>3693</td><td>11.5</td><td>70</td></tr><tr><td>2</td><td>18.0</td><td>8</td><td>318.0</td><td>3436</td><td>11.0</td><td>70</td></tr><tr><td>3</td><td>16.0</td><td>8</td><td>304.0</td><td>3433</td><td>12.0</td><td>70</td></tr><tr><td>4</td><td>17.0</td><td>8</td><td>302.0</td><td>3449</td><td>10.5</td><td>70</td></tr></table>		mpg	cylinders	displacement	weight	acceleration	model_year	0	18.0	8	307.0	3504	12.0	70	1	15.0	8	350.0	3693	11.5	70	2	18.0	8	318.0	3436	11.0	70	3	16.0	8	304.0	3433	12.0	70	4	17.0	8	302.0	3449	10.5	70																		
	mpg	cylinders	displacement	weight	acceleration	model_year																																																							
0	18.0	8	307.0	3504	12.0	70																																																							
1	15.0	8	350.0	3693	11.5	70																																																							
2	18.0	8	318.0	3436	11.0	70																																																							
3	16.0	8	304.0	3433	12.0	70																																																							
4	17.0	8	302.0	3449	10.5	70																																																							

In [2]: 데이터셋의 형태 `data_df.shape` 를 확인해보면, 398행과 9열로 구성되어 있음  
398개 데이터에 9개 컬럼이 있으므로 파일 내용이 DataFrame으로 잘 저장되었다는 것을 알 수 있음  
데이터 5개를 출력하여 내용을 확인 `data_df.head()` .

In [3] 피쳐 중에서 `car_name`, `origin`, `horsepower`는 분석에 사용하지 않으므로 제거 `data_df.drop()` 후 확인 `data_df.head()` .

# 선형회귀 실습

- 데이터 준비 및 탐색
  - 분석에 필요 없는 컬럼을 제거하고 데이터셋의 내용을 확인

In [4]:	print('데이터셋 크기: ', data_df.shape)
Out[4]:	데이터셋 크기: (398, 6)
In [5]:	data_df.info()
Out[5]:	<class 'pandas.core.frame.DataFrame'> RangeIndex: 398 entries, 0 to 397 Data columns (total 6 columns): mpg          398 non-null float64 cylinders    398 non-null int64 displacement 398 non-null float64 weight       398 non-null int64 acceleration 398 non-null float64 year         398 non-null int64 dtypes: float64(3), int64(3) memory usage: 18.7 KB

In [4]: 분석에 사용할 데이터셋의 형태 `data_df.shape` 를 확인

In [5]: 분석에 사용할 데이터셋의 정보 `data_df.info()` 를 확인

# 선형회귀 실습

- 분석 모델 구축, 결과 분석 및 시각화

1. 선형 회귀 분석 모델 구축하기

1. 자동차 연비 예측을 위해 다음과 같이 선형 회귀 분석 모델을 구축

In [6]:	<pre>from sklearn.linear_model import <b>LinearRegression</b> from sklearn.model_selection import <b>train_test_split</b> from sklearn.metrics import <b>mean_squared_error, r2_score</b></pre>
In [7]:	<pre><i>#X, Y 분할하기</i> Y = data_df['mpg'] X = data_df.<b>drop</b>(['mpg'], axis = 1, inplace = False)</pre>
In [8]:	<pre><i>#훈련용 데이터와 평가용 데이터 분할하기</i> X_train, X_test, Y_train, Y_test = <b>train_test_split</b>(X, Y, test_size = 0.3, random_state = 0)</pre>

In [6]: 사이킷런을 사용하여 머신러닝 선형 회귀 분석을 하기 위한 LinearRegression과 데이터셋 분리 작업을 위한 train\_test\_split, 성능 측정을 위한 평가 지표인 mean\_squared\_error, r2\_score를 импорт

In [7]: 자동차 연비를 예측하는 것이 프로젝트의 목표이므로, mpg 피처를 회귀식의 종속 변수 Y로 설정하고, mpg를 제외한 나머지 피처를 독립 변수 X로 설정

In [8]: 데이터를 7:3 비율 test\_size=0.3로 분할하여 train\_test\_split( ) 학습 데이터와 평가 데이터로 설정

# 선형회귀 실습

- 분석 모델 구축, 결과 분석 및 시각화

1. 선형 회귀 분석 모델 구축하기

1. 자동차 연비 예측을 위해 다음과 같이 선형 회귀 분석 모델을 구축

In [9]:	<i>#선형 회귀 분석 : 모델 생성</i> <code>lr = LinearRegression()</code>
In [10]	<i>#선형 회귀 분석 : 모델 훈련</i> <code>lr.fit(X_train, Y_train)</code>
Out[10]:	LinearRegression()
In [11]:	<i>#선형 회귀 분석 : 평가 데이터에 대한 예측 수행 -&gt; 예측 결과 Y_predict 구하기</i> <code>Y_predict = lr.predict(X_test)</code>

In [9]: 선형 회귀 분석 모델 객체인 lr을 생성

In [10]: 학습 데이터  $X_{train}$ 와  $Y_{train}$ 를 가지고 학습을 수행 `fit()`

In [11]: 평가 데이터  $X_{test}$ 로 예측을 수행하여 `predict()` 예측값  $Y_{predict}$ 를 구함

# 선형회귀 실습

- 분석 모델 구축, 결과 분석 및 시각화

1. 선형 회귀 분석 모델 구축하기

2. 평가 지표를 통해 선형 회귀 분석 모델을 평가하고 회귀 계수를 확인하여 자동차 연비에 끼치는 피처의 영향을 분석

In [12]:	<pre>mse = mean_squared_error(Y_test, Y_predict) rmse = np.sqrt(mse) print('MSE : {0:.3f}, RMSE : {1:.3f}'.format(mse, rmse)) print('R^2(Variance score) : {0:.3f}'.format(r2_score(Y_test, Y_predict)))</pre>
Out[12]:	<pre>MSE : 12.278, RMSE : 3.504 R^2(Variance score) : 0.808</pre>
In [13]	<pre>print('Y 절편 값: ', np.round(lr.intercept_, 2)) print('회귀 계수 값: ', np.round(lr.coef_, 2))</pre>
Out[13]:	<pre>Y 절편 값: -17.55 회귀 계수 값: [-0.14 0.01 -0.01 0.2 0.76]</pre>

In [12]: 회귀 분석은 지도 학습이므로 평가 데이터 X에 대한  $Y_{test}$ 를 이미 알고 있음

평가 데이터의 결과값  $Y_{test}$ 과 예측 결과값  $Y_{predict}$ 의 오차를 계산하여 모델을 평가하는데, `mean_squared_error()`를 이용하여 평가 지표 MSE를 구하고 구한 값의 제곱근을 계산하여 평가 지표 RMSE를 구한다. 그리고 `r2_score()`를 이용하여 평가 지표 R2를 구함

In [13]: 선형 회귀의 Y절편  $lr.intercept_$ 과 각 피처의 회귀 계수  $lr.coef_$ 를 확인

# 선형회귀 실습

- 분석 모델 구축, 결과 분석 및 시각화

1. 선형 회귀 분석 모델 구축하기

2. 평가 지표를 통해 선형 회귀 분석 모델을 평가하고 회귀 계수를 확인하여 자동차 연비에 끼치는 피처의 영향을 분석

In [14]:	<pre>coef = pd.Series(data = np.round(lr.coef_, 2), index = X.columns) coef.sort_values(ascending = False)</pre>
Out[14]:	<pre>model_year    0.76 acceleration   0.20 displacement   0.01 weight        -0.01 cylinders     -0.14 dtype: float64</pre>

In [14]: 회귀 모델에서 구한 회귀 계수 값 `lr.coef_` 과 피처 이름 `X.columns` 을 묶어서 Series 자료 형으로 만들고, 회귀 계수 값을 기준으로 내림차순 `ascending = False` 으로 정렬 `sort_values()` 하여 회귀 계수 값이 큰 항목을 확인

회귀 모델 결과로 자동차 연비를 예측하는 회귀식

$$Y_{\text{mpg}} = -0.14X_{\text{cylinders}} + 0.01X_{\text{displacement}} - 0.01X_{\text{weight}} + 0.20X_{\text{acceleration}} + 0.76X_{\text{model\_year}} - 17.55$$

- 분석 모델 구축, 결과 분석 및 시각화

- 2. 선형 회귀 분석 모델 구축하기

- 1. 평가 지표를 통해 선형 회귀 분석 모델을 평가하고 회귀 계수를 확인하여 자동차 연비에 끼치는 피처의 영향을 분석

In [15]:	<pre>import matplotlib.pyplot as plt import seaborn as sns</pre>
In [16]:	<pre>fig, axs = plt.subplots(figsize = (16, 16), ncols = 3, nrows = 2) x_features = ['model_year', 'acceleration', 'displacement', 'weight', 'cylinders'] plot_color = ['r', 'b', 'y', 'g', 'r'] for i, feature in enumerate(x_features):     row = int(i/3)     col = i%3     sns.regplot(x = feature, y = 'mpg', data = data_df, ax = axs[row][col], color = plot_color[i])</pre>

In [15]: 시각화에 필요한 모듈을 импорт

In [16]: subplots()를 사용하여 독립 변수인 5개 피처 `['model_year', 'acceleration', 'displacement', 'weight', 'cylinders']`와 종속 변수인 연비 mpg와의 회귀 관계를 보여주는 5개 그래프를 2행 3열 구조로 나타낸



# 선형회귀 실습

- 분석 모델 구축, 결과 분석 및 시각화

- 2. 선형 회귀 분석 모델 구축하기

- 1. 평가 지표를 통해 선형 회귀 분석 모델을 평가하고 회귀 계수를 확인하여 자동차 연비에 끼치는 피처의 영향을 분석

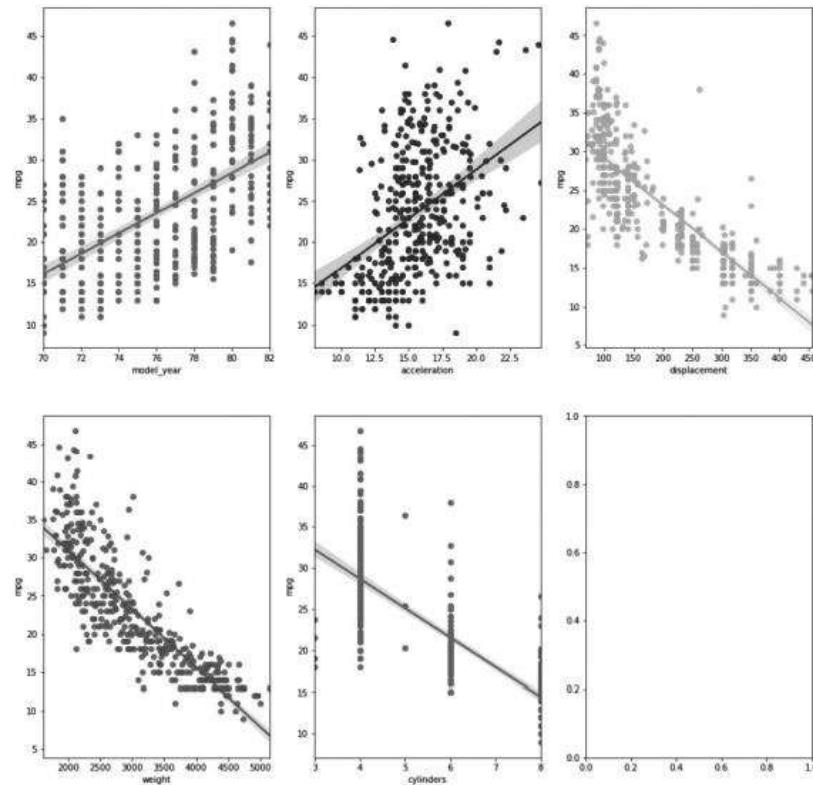


그림 10-12 5개 피처와 연비의 회귀 관계를 보여주는 산점도 + 선형 회귀 그래프

# 선형회귀 실습

- 분석 모델 구축, 결과 분석 및 시각화

- 2. 선형 회귀 분석 모델 구축하기

- 2. 완성된 자동차 연비 예측 모델을 사용하여 임의의 데이터를 입력하면 연비를 예측할 수 있음

In [17]:	<pre>print("연비를 예측하고 싶은 차의 정보를 입력해주세요.") cylinders_1 = int(input("cylinders : ")) displacement_1 = int(input("displacement : ")) weight_1 = int(input("weight : ")) acceleration_1 = int(input("acceleration : ")) model_year_1 = int(input("model_year : "))</pre>
Out[17]:	<div><div>연비를 예측하고 싶은 차의 정보를 입력해주세요. cylinders : 8 displacement : 350 weight : 3200 acceleration : 22 model_year : 99</div><div>키보드로 값을 입력한 후 [Enter] 누르기</div></div>
In [18]:	<pre>mpg_predict = lr.predict([[cylinders_1, displacement_1, weight_1, acceleration_1, model_year_1]])</pre>
In [19]:	<pre>print("이 자동차의 예상 연비(MPG)는 %.2f입니다." %mpg_predict)</pre>
Out[19]:	이 자동차의 예상 연비(MPG)는 41.32입니다

In [17]: 5개 항목(독립 변수)을 입력하면 변수에 저장

In [18]: 변수를 회귀 모델에 적용하여 예측 결과값을 구함