

회귀분석

안화수

사이킷런(Scikit-Learn)

❖ 사이킷런 모듈 설치

1. python 에 설치

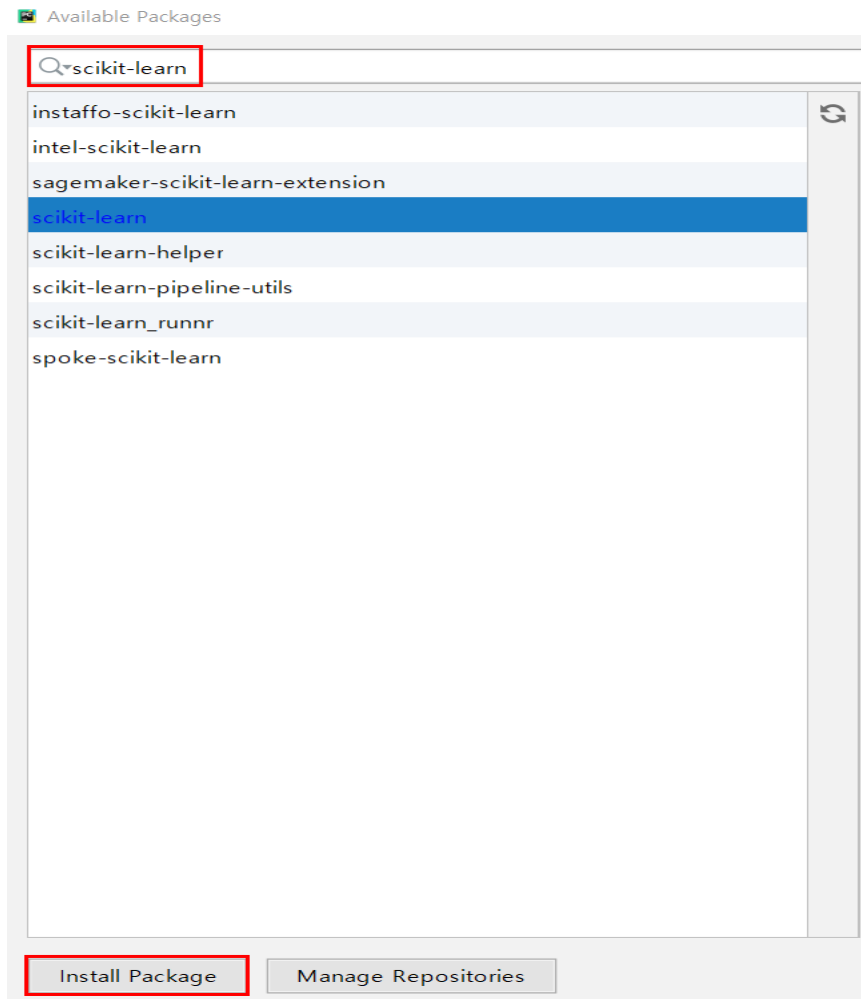
```
c:\W> pip install scikit-learn
```

2. anaconda 에 설치

```
c:\W> conda install scikit-learn
```

사이킷런(Scikit-Learn)

❖ PyCharm에 scikit-learn 설치



Scikit-Learn DataSet

❖ 사이킷런에 내장된 예제 데이터

사이킷런에는 별도의 외부 웹사이트에서 데이터 세트를 다운로드 받을 필요 없이 예제로 활용할 수 있는 간단하면서도 좋은 데이터 세트가 내장되어 있다. 이 데이터는 datasets 모듈에 있는 여러 API를 호출해 만들 수 있다.

<분류나 회귀 연습용 예제 데이터>

API명	설명
<code>datasets.load_digits()</code>	분류용, 0에서 9까지 숫자 이미지 픽셀 데이터셋
<code>datasets.load_iris()</code>	분류용, 붓꽃에 대한 피처를 가진 데이터셋
<code>datasets.load_breast_cancer()</code>	분류용, 위스콘신 유방암 피처들과 악성/음성 레이블
<code>datasets.load_boston()</code>	회귀용, 미국 보스턴 집 피처들과 가격 데이터셋
<code>datasets.load_diabetes()</code>	회귀용, 당뇨 데이터셋

회귀분석

❖ 회귀분석

머신러닝(Machine Learning)은 인공지능의 한 분야로 기계 스스로 대량의 데이터로부터 지식이나 패턴을 찾아 학습하고 예측하는 알고리즘 기법을 통칭한다.

< 예측 알고리즘 >

Types	Tasks	Algorithms
지도 학습 (Supervised Learning)	예측 (Prediction)	Linear Regression
		SVM : Support Vector Machine
		Random Forest
		KNN : K Nearest Neighbor

회귀분석

❖ 회귀 분석 분류

➤ 독립변수의 개수로 분류

1. 단순 회귀 분석(Simple Linear Regression)
독립변수가 1개인 회귀 분석 방법

$$y = ax + b$$

a : 회귀계수(기울기), b : 절편, x : 독립변수

2. 다중 회귀 분석(Multiple Linear Regression)
독립변수가 2개 이상인 회귀 분석 방법

$$y = ax_1 + bx_2 + cx_3 + d \quad \# \text{ 독립변수 : } x_1, x_2, x_3$$

단순회귀분석

❖ 회귀 분석 분류

일반적으로 소득이 증가하면 소비가 증가하는 것처럼, 어떤 변수 (독립변수 X)가 다른 변수(종속변수 Y)에 영향을 준다면 두 변수 사이에 선형 관계가 있다고 말한다. 이와 같은 선형관계를 알고 있다면 새로운 독립변수 X 값이 주어졌을 때 거기에 대응하는 종속변수 Y 값을 예측할 수 있다.

이처럼 두 변수 사이에 일대일로 대응되는 확률적, 통계적 상관성을 찾는 알고리즘을 **단순회귀분석(Simple Linear Regression)**이라고 말한다. 단순회귀분석을 대표적인 지도학습의 유형이다.

수학적으로 종속 변수 Y 와 독립 변수 X 사이의 관계를 1차함수 $Y = aX + b$ 로 나타낸다. 단순회귀분석 알고리즘은 훈련 데이터를 이용하여 직선의 기울기(a)와 직선이 y 축과 교차하는 지점인 y 절편(b)을 반복 학습을 통해서 찾는다.

일차 방정식의 계수 a (기울기)와 b (절편)를 찾는 과정이 단순회귀분석 알고리즘이다.

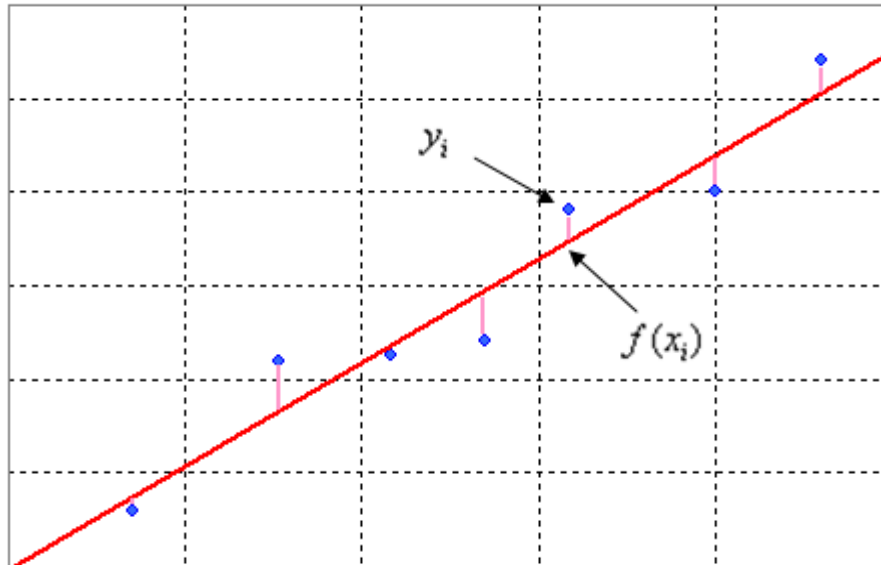
최소 제곱법

❖ 최소 제곱법(Method of Least Squares)

실제값과 예측값 차이에 대한 제곱의 합을 최소로 해서 $f(x)$ 를 구하는 방법
N회 측정한 측정값 y_1, y_2, \dots, y_n 이 어떤 다른 측정값 x_1, x_2, \dots, x_n 의 함수라고 추정할 수 있을때, 측정값 y 와 함수값 $f(x)$ 의 차이를 제곱한 것의 합

$$\sum_{i=1}^n (y_i - f(x_i))^2$$

이 최소가 되도록 하는 함수 $f(x)$ 를 구하는 것이 최소 제곱법의 원리이다.



최소 제곱법

❖ 최소 제곱법(Method of Least Squares)

➤ scikit-learn 에서 최소 제곱법 구현 방법

```
linear_model . LinearRegression()
```

단순선형회귀 예제

❖ 단순선형 회귀 예제

$y = ax + b$ 처럼 데이터를 만들어 회귀문제를 풀어 보자.

여기서는 $y = 3x - 2$ 인 경우에 최소 제곱법으로 기울기와 절편을 구해보자?

linear_orderly.py (1 / 2)

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
```

학습 데이터 생성

```
x = np.random.rand(100, 1)
```

0~1까지 난수를 100개 만든다

```
x = x * 4 - 2
```

값의 범위를 -2~2로 변경

```
y = 3 * x - 2
```

$y = 3x - 2$

단순선형회귀 예제

❖ 단순선형 회귀 예제

linear_orderly.py (2 / 2)

```
# 모델 : 최소 제곱법으로 구현
model = linear_model.LinearRegression()

# 학습
model.fit(x, y)

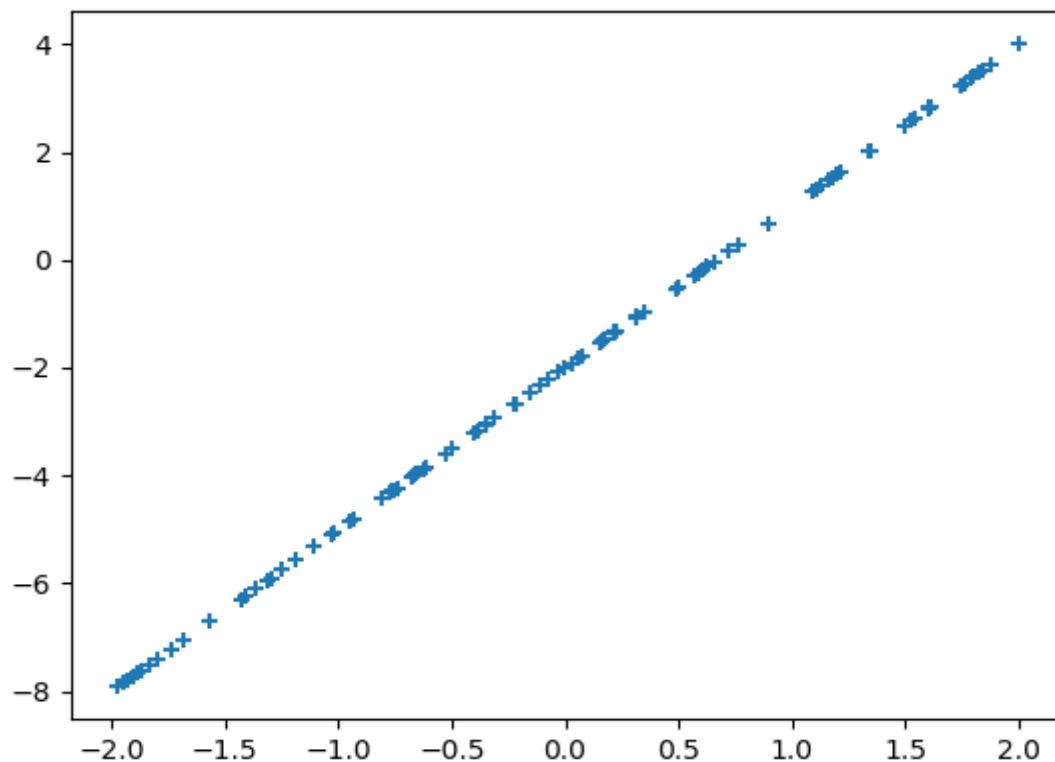
# 계수(기울기), 절편
print('기울기:', model.coef_)
print('절편:', model.intercept_)

# 산점도 그래프 출력
plt.scatter(x, y, marker='+')
plt.show()
```

단순선형회귀 예제

❖ 오차가 없는 $y = 3x - 2$ 예측결과 그래프

$y = ax + b$ 에서, 기울기 $a = 3$, 절편 $b = -2$ 가 구해졌음.



단순선형회귀 예제

❖ 단순선형 회귀 예제

$y = ax + b$ 처럼 데이터를 만들어 회귀문제를 풀어 보자.

여기서는 $y = 3x - 2$ 에 정규분포 난수를 더했을때, 최소 제곱법으로 기울기와 절편을 예측해 보자.

linear_rough.py (1 / 2)

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
```

```
# 학습 데이터 생성
```

```
x = np.random.rand(100, 1)
```

```
x = x * 4 - 2
```

```
y = 3 * x - 2
```

```
# 0~1까지 난수를 100개 만든다
```

```
# 값의 범위를 -2~2로 변경
```

```
#  $y = 3x - 2$ 
```

```
# 표준 정규 분포(평균 0, 표준 편차 1)의 난수를 추가함
```

```
y += np.random.randn(100, 1)
```

단순선형회귀 예제

❖ 단순선형 회귀 예제

linear_rough.py (2 / 2)

모델 : 최소 제곱법으로 구현

```
model = linear_model.LinearRegression()
```

학습

```
model.fit(x, y)
```

계수, 절편

```
print('계수:', model.coef_)
```

```
print('절편:', model.intercept_)
```

산점도 그래프 출력

```
plt.scatter(x, y, marker = '+')
```

```
plt.scatter(x, model.predict(x), marker='o')
```

```
plt.show()
```

실제값 그래프 : + 마크

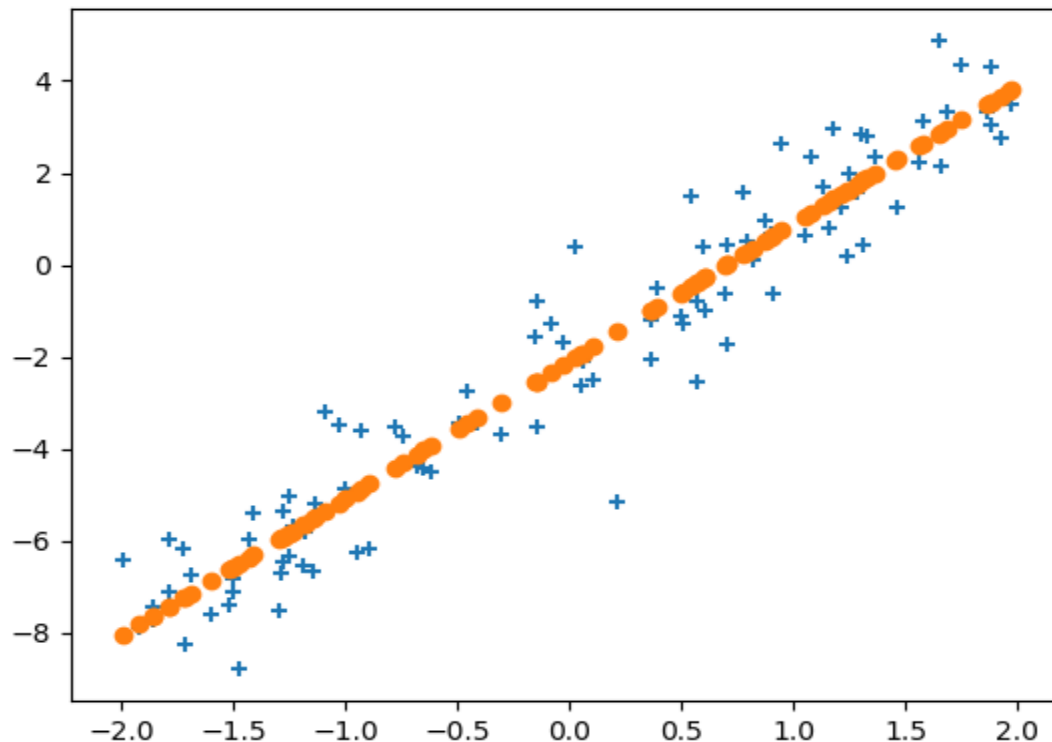
예측값 그래프 : o 마크

단순선형회귀 예제

❖ 오차가 있는 $y = 3x - 2$ 예측결과 그래프

$y = 3x - 2$ 에서, 기울기 $a = 2.89275402$,

절편 $b = -1.84078098$ 가 구해졌음.



+ : 학습데이터

o : 예측결과

결정계수

❖ 결정계수 (R^2)

✓ 회귀 결과의 타당성을 객관적으로 평가하는 지표로 결정계수를 사용한다.

✓ 회귀모형의 설명력 ($0 \leq R^2 \leq 1$)

✓ 구해진 회귀식이 “얼마나 납득할 수 있는가”를
말함

Sum of Square (제곱합)

✓ [추정치SS / 관측치SS]로 계산

$$R^2 = \frac{\sum_{i=1}^n (\hat{Y}_i - \bar{Y})^2}{\sum_{i=1}^n (Y_i - \bar{Y})^2}$$

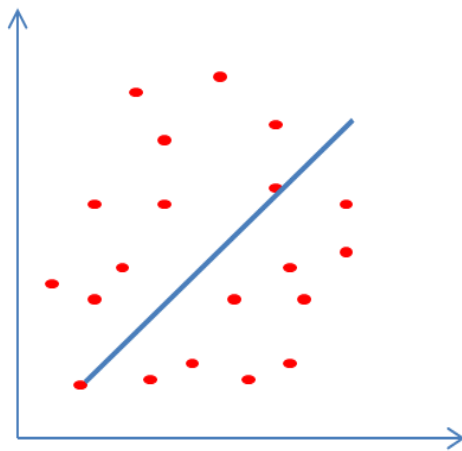
→ SS(추정치 minus 평균)
→ SS(관측치 minus 평균)

✓ 결정계수의 값이 1에 가까우면 성능이 좋은 예측 모델이라고 할 수 있다.

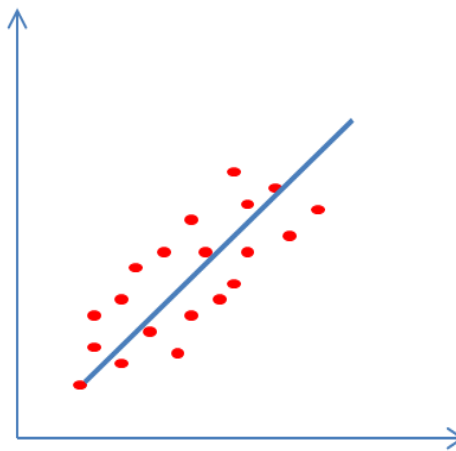
✓ scikit-learn 에서 결정계수는 `model.score()` 함수로 구할 수 있다.

결정계수

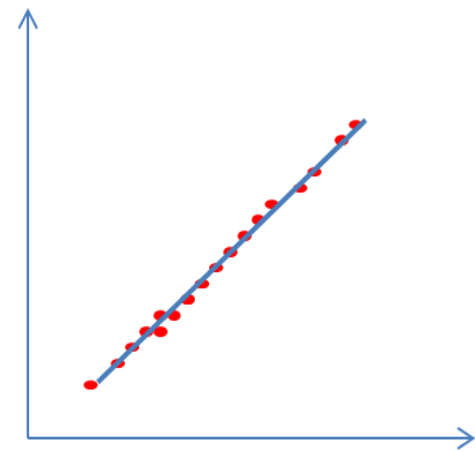
❖ 결정계수 (R^2)



$$R^2 = 0$$



$$R^2 = 0.5$$



$$R^2 = 1$$

결정계수가 0 에 가까울 수록 예측 성능이 좋지 않고, 1 에 가까울 수록 예측 성능이 좋다고 할 수 있다.

단순선형회귀 예제

❖ 단순선형 회귀 예제 : 결정계수

$y = ax + b$ 처럼 데이터를 만들어 회귀문제를 풀어 보자.

여기서는 $y = 3x - 2$ 에 정규분포 난수를 더했을때, 최소 제곱법으로 기울기와 절편을 예측해 보자.

`r_squared.py (1 / 2)`

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
```

학습 데이터 생성

```
x = np.random.rand(100, 1)
```

```
x = x * 4 - 2
```

```
y = 3 * x - 2
```

0~1까지 난수를 100개 만든다

값의 범위를 -2~2로 변경

$y = 3x - 2$

표준 정규 분포(평균 0, 표준 편차 1)의 난수를 추가함

```
y += np.random.randn(100, 1)
```

단순선형회귀 예제

❖ 단순선형 회귀 예제 : 결정계수

r_squared.py (2 / 2)

```
# 모델 : 최소 제곱법으로 구현  
model = linear_model.LinearRegression()
```

```
### 학습  
model.fit(x, y)
```

```
### 계수, 절편, 결정 계수  
print('계수:', model.coef_)  
print('절편:', model.intercept_)
```

```
r2 = model.score(x, y)  
print('결정계수:', r2)
```

```
# 산점도 그래프 출력  
plt.scatter(x, y, marker = '+')  
plt.scatter(x, model.predict(x), marker='o')  
plt.show()
```

단순선형회귀 예제

❖ 오차가 있는 $y = 3x - 2$ 예측결과

- $y = 3x - 2$ 에서, 기울기 $a = 2.89275402$,
절편 $b = -1.84078098$ 가 구해졌음
- 결정계수 : 0.9376000751123739

단순선형회귀 예제

❖ 단순선형 회귀 예제 : 결정계수

$y = ax^2 + b$ 형태의 데이터를 만들어 회귀문제를 풀어 보자.

여기서는 $y = 3x^2 - 2$ 에 정규분포 난수를 더했을때, 최소 제곱법으로 기울기와 절편을 예측해 보자.

quadratic.py (1 / 2)

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
```

학습 데이터 생성

```
x = np.random.rand(100, 1)      # 0~1까지 난수를 100개 만든다
```

```
x = x * 4 - 2                    # 값의 범위를 -2~2로 변경
```

```
y = 3 * x**2 - 2                 #  $y = 3x^2 - 2$ 
```

```
y += np.random.randn(100, 1)    # 표준 정규 분포(평균 0, 표준 편차 1)의 난수를 추가함
```

단순선형회귀 예제

❖ 단순선형 회귀 예제 : 결정계수

quadratic.py (2 / 2)

학습

```
model = linear_model.LinearRegression()  
model.fit(x**2, y)
```

x를 제공해 전달

계수, 절편, 결정 계수를 표시

```
print('계수:', model.coef_)  
print('절편:', model.intercept_)  
print('결정계수:', model.score(x**2, y))
```

산점도 그래프 출력

```
plt.scatter(x, y, marker='+')  
plt.scatter(x, model.predict(x**2), marker='o') # predict에도 x를 제공해 전달  
plt.show()
```

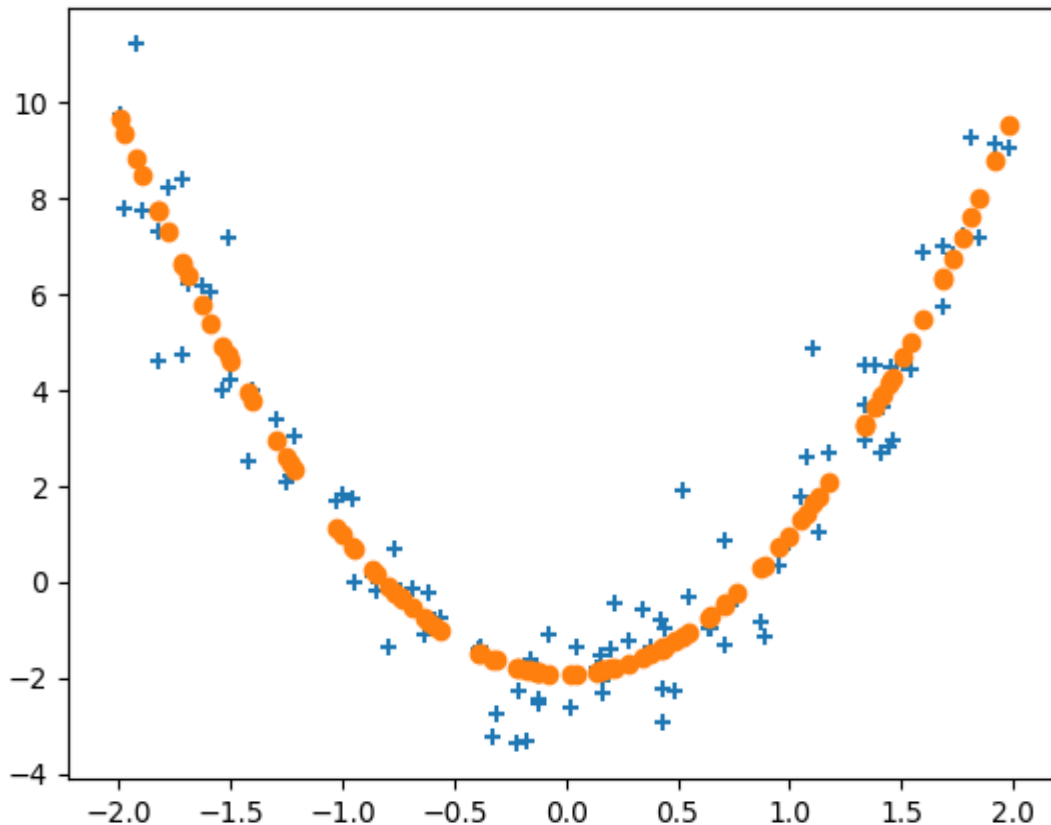
단순선형회귀 예제

❖ $y = 3x^2 - 2$ 예측결과

- $y = 3x^2 - 2$ 에서, 계수 $a = 2.90030058$,
절편 $b = -1.91436032$ 가 구해졌음.
- 결정계수 : 0.9213648484133768

단순선형회귀 예제

❖ $y = 3x^2 - 2$ 예측결과 그래프



+ : 학습데이터
o : 예측결과

다중 선형회귀 예제

❖ 다중 선형 회귀 예제

$y = ax_1 + bx_2 + c$ 형태의 데이터를 만들어 회귀문제를 풀어 보자.

여기서는 $y = 3x_1 - 2x_2 + 1$ 에 정규분포 난수를 더했을때, 최소 제곱법으로 회귀계수와 절편을 예측해 보자.

multiple.py (1 / 3)

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
```

```
# y = 3x_1 - 2x_2 + 1 학습 데이터 생성
```

```
x1 = np.random.rand(100, 1)      # 0~1까지 난수를 100개 만든다
```

```
x1 = x1 * 4 - 2                    # 값의 범위를 -2~2로 변경
```

```
x2 = np.random.rand(100, 1)      # x2에 대해서도 같게
```

```
x2 = x2 * 4 - 2
```

```
y = 3 * x1 - 2 * x2 + 1
```

```
y += np.random.randn(100, 1)    # 표준 정규 분포(평균 0, 표준 편차 1)의 난수를 추가함
```

다중 선형회귀 예제

❖ 다중 선형 회귀 예제

multiple.py (2 / 3)

x1, x2 값을 가진 행렬 생성

x1_x2 = np.c_[x1, x2]

[[x1_1, x2_1], [x1_2, x2_2], ..., [x1_100, x2_100]]

형태로 변환

모델

model = linear_model.LinearRegression()

학습

model.fit(x1_x2, y)

계수, 절편, 결정 계수를 표시

print('계수', model.coef_)

print('절편', model.intercept_)

print('결정계수', model.score(x1_x2, y))

다중 선형회귀 예제

❖ 다중 선형 회귀 예제

multiple.py (3 / 3)

산점도 그래프 표시

y_ = model.predict(x1_x2)

plt.subplot(1, 2, 1)

plt.scatter(x1, y, marker='+')

plt.scatter(x1, y_, marker='o')

plt.xlabel('x1')

plt.ylabel('y')

plt.subplot(1, 2, 2)

plt.scatter(x2, y, marker='+')

plt.scatter(x2, y_, marker='o')

plt.xlabel('x2')

plt.ylabel('y')

plt.tight_layout()

plt.show()

회귀식으로 예측

1행 2열 배치 , 1번째 subplot

1행 2열 배치 , 2번째 subplot

자동으로 레이아웃을 설정해주는 함수

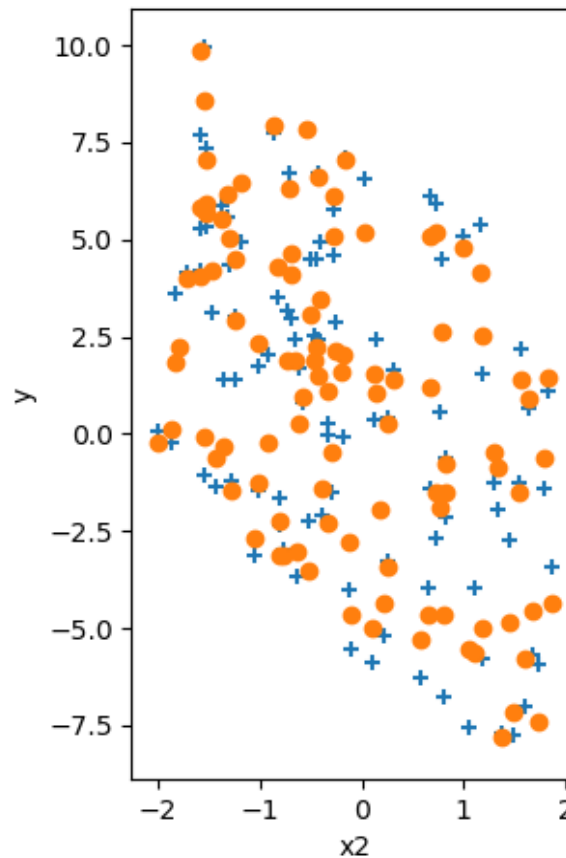
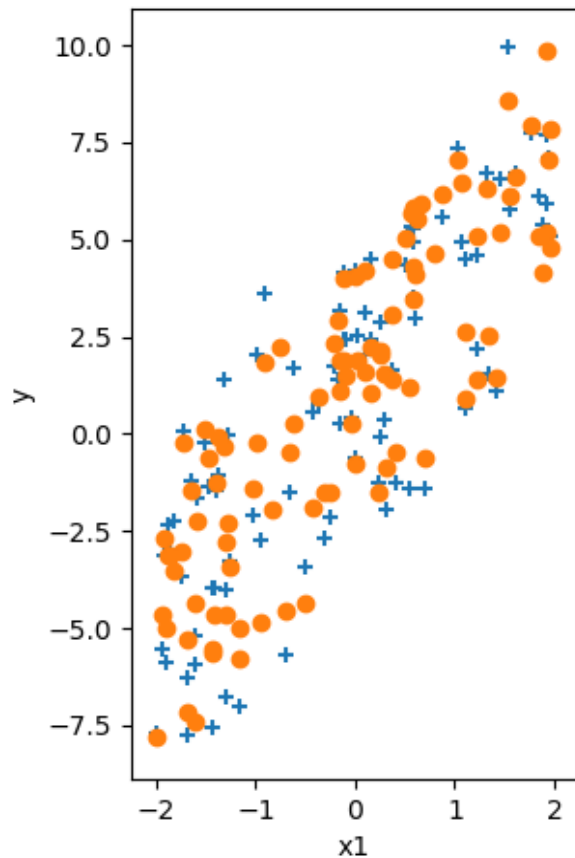
다중 선형회귀 예제

❖ $y = 3x_1 - 2x_2 + 1$ 예측결과

- $y = 3x_1 - 2x_2 + 1$ 에서
회귀식은 $y = 2.99585775x_1 - 2.01490304x_2 + 0.90307907$ 이며
회귀 계수와 절편도 원래식과 가까운 값이 구해졌다.
- 결정계수 : 0.9316382381463262 도 상당히 높은 값이 구해졌다.

다중 선형회귀 예제

❖ $y = 3x_1 - 2x_2 + 1$ 예측결과 그래프



+ : 학습데이터
o : 예측결과

붓꽃(iris) 데이터 품종 예측하기

- ❖ 사이킷런을 이용하여 붓꽃(iris) 데이터 품종 예측하기
 - 사이킷런에 내장된 3종류의 붓꽃 : **setosa, versicolor, virginica**
 - 붓꽃의 측정값 : **꽃받침의 길이 (sepal length)**

꽃받침의 폭 (sepal width)

꽃잎의 길이 (petal length)

꽃잎의 폭 (petal width)

iris DataSet

❖ 피셔의 붓꽃

iris 데이터셋은 이 데이터셋 값을 공개해 통계학적 분석을 실시한 로널드 피셔(Ronald Fisher, 1890 ~ 1962) 이름에서 따온 것이다.

이것은 영국 식물학자 에드가 앤더슨(Edgar Anderson, 1897 ~ 1969)이 세 종류의 붓꽃의 유전적 분화를 밝힐 목적으로 여러 개체의 크기를 측정한 것을 정비한 데이터 셋이다.

❖ iris 데이터셋이란 ?

iris 는 붓꽃 관련 데이터셋으로 피셔의 붓꽃 이라고도 한다.

➤ 데이터 내용

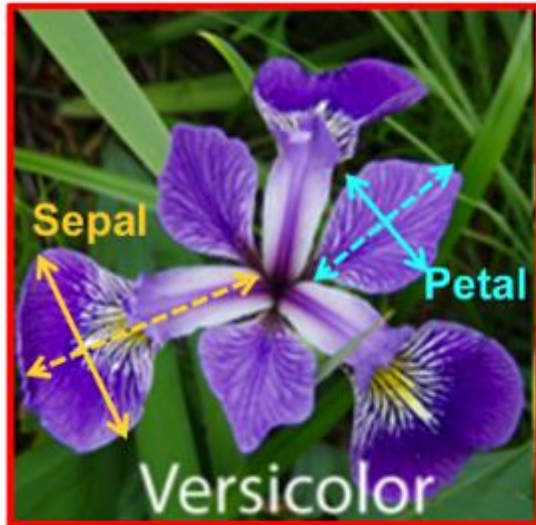
붓꽃 3종류에서 각각 50개씩의 측정 데이터

➤ 측정 데이터 내역

꽃받침의 길이, 꽃받침의 폭, 꽃잎의 길이, 꽃잎의 폭

iris DataSet

❖ iris DataSet



iris DataSet

❖ scikit-learn 의 iris DataSet

scikit-learn 에는 iris 데이터셋이 내장되어 있으며, iris 데이터를 읽을 수 있도록 API로 `sklearn.datasets.load_iris()` 를 준비해 두었다.

```
from sklearn import datasets  
iris = datasets.load_iris()
```

➤ **target_names** : 붓꽃의 품종이 등록되어 있음

```
print( iris['target_names'] )  
# ['setosa' 'versicolor' 'virginica']
```

➤ **feature_names** : 데이터 속성의 이름이 등록되어 있음

```
print( iris['feature_names'] )  
# ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']  
# [ '꽃받침의 길이', '꽃받침의 폭', '꽃잎의 길이', '꽃잎의 폭' ]
```

iris DataSet

❖ scikit-learn 의 iris DataSet

iris_datasets.py (1 / 2)

```
from sklearn import datasets
```

```
# iris 데이터 로드
```

```
iris = datasets.load_iris()
```

```
# 1. data : 붓꽃의 측정값
```

```
data = iris['data']
```

```
# print(data)
```

```
# 2. DESCR
```

```
# 피셔의 붓꽃데이터 설명 출력
```

```
print(iris['DESCR'])    # iris Data Set Characteristics
```

```
# - class:
```

```
#           - Iris-Setosa
```

```
#           - Iris-Versicolour
```

```
#           - Iris-Virginica
```

iris DataSet

❖ scikit-learn 의 iris DataSet

iris_datasets.py (2 / 2)

```
# 3. target : 붓꽃의 품종이 ID 번호로 등록되어 있음  
print(iris['target'])
```

```
# 4. target_names : 붓꽃의 품종이 등록되어 있음  
print(iris['target_names'])  
# ['setosa' 'versicolor' 'virginica']
```

```
# 5. feature_names  
print(iris['feature_names'])  
# ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']  
# [ '꽃받침의 길이', '꽃받침의 폭', '꽃잎의 길이', '꽃잎의 폭' ]
```

보스턴 주택가격 회귀분석

- ❖ **LinearRegression을 이용한 보스턴 주택 가격 회귀분석**
- **LinearRegression 클래스를 이용한 선형회귀 모델을 만들어 보자.**

```
lr = LinearRegression()
```

- **사이킷런에 내장된 데이터셋인 보스턴 주택가격 데이터를 이용한다.**

```
# boston 데이터셋 로드  
boston = datasets.load_boston()
```

보스턴 주택가격 회귀분석

➤ 보스턴 집값 데이터 : 인덱스 : 506행, 컬럼 : 14열

	feature_names					target
	0	1	2	...	12	13(집값)
0	0.00632	18.0	2.31	...	4.98	24.0
1	0.02731	0	7.07	...	9.14	21.6
2	0.02729	0	7.07	...	4.03	34.7
...
505	0.04741	0	11.930		7.88	11.90

보스턴 주택가격 회귀분석

❖ 보스턴 집값 데이터의 14개 컬럼 정보

0	CRIM : 인구 1인당 범죄 발생 수
1	ZN : 25,000평방 피트 이상의 주거 구역 비중
2	INDUS : 소매업 외 상업이 차지하는 면적 비율
3	CHAR : 찰스강 위치 변수 (1:강 주변, 0:이외)
4	NOX : 일산화 질소 농도
5	RM : 집의 평균 방 수
6	AGE : 1940년 이전에 지어진 비율
7	DIS : 5가지 보스턴시 고용 시설까지의 거리
8	RAD : 순환 고속도로의 접근 용이성
9	TAX : \$10,000당 부동산 세율 총계
10	PTRATIO : 지역별 학생과 교사 비율
11	B : 지역별 흑인 비율
12	LSTAT : 급여가 낮은 직업에 종사하는 인구 비율(%)
13	주택 가격(단위 : \$1,000)

보스턴 주택가격 회귀분석

boston.py (1 / 5)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_boston
```

```
# boston 데이터셋 로드
boston = load_boston()
```

```
# boston 데이터셋을 DataFrame으로 변환
bostonDF = pd.DataFrame(boston.data, columns=boston.feature_names)
print(bostonDF)                                # [506 rows x 13 columns]
```

```
# boston 데이터셋의 target 열(컬럼)은 주택 가격임.
```

```
# boston.target을 PRICE 컬럼으로 추가함
```

```
bostonDF['PRICE'] = boston.target
print('Boston 데이터셋 크기:', bostonDF.shape)
Print(bostonDF.head())
```

```
# 'PRICE' 컬럼 추가
```

```
# Boston 데이터셋 크기: (506, 14)
```

```
# 앞에서 5개의 데이터 출력
```

보스턴 주택가격 회귀분석

boston.py (2 / 5)

다음의 각 컬럼 RM, ZN, INDUS, NOX, AGE, PTRATIO, LSTAT, RAD 의 총 8개의 컬럼에
대해서 값이 증가할수록 PRICE에 어떤 영향을 미치는지 분석하고 시각화를 해보자

2개의 행과 4개의 열을 가진 subplots 로 시각화, axs는 4x2개의 ax를 가짐

```
fig, axs = plt.subplots(figsize=(16,8), ncols=4, nrows=2)
```

```
lm_features = ['RM', 'ZN', 'INDUS', 'NOX', 'AGE', 'PTRATIO', 'LSTAT', 'RAD']
```

```
for i, feature in enumerate(lm_features):
```

```
    row = int(i/4)
```

```
    col = i%4
```

```
# 시본(seaborn)의 regplot을 이용해 산점도와 선형 회귀선을 출력
```

```
sns.regplot(x=feature , y='PRICE', data=bostonDF , ax=axs[row][col])
```

```
plt.show()
```

RM(방개수)와 LSTAT(하위 계층의 비율)이 PRICE에 영향도가 가장 두드러지게 나타남.

1. RM(방개수)은 양 방향의 선형성(Positive Linearity)이 가장 크다.

방의 개수가 많을수록 가격이 증가하는 모습을 확연히 보여준다.

2. LSTAT(하위 계층의 비율)는 음 방향의 선형성(Negative Linearity)이 가장 크다.

하위 계층의 비율이 낮을수록 PRICE 가 증가하는 모습을 확연히 보여준다.

보스턴 주택가격 회귀분석

boston.py (3 / 5)

LinearRegression 클래스를 이용해서 보스턴 주택 가격의 회귀 모델을 만들어 보자
train_test_split()을 이용해 학습과 테스트 데이터셋을 분리해서 학습과 예측을 수행한다.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

```
y_target = bostonDF['PRICE'] # 'PRICE' 컬럼값을 y_target에 할당
x_data = bostonDF.drop(['PRICE'], axis=1, inplace=False)
# 데이터프레임에서 'PRICE' 컬럼 삭제한 나머지 데이터를 반환
```

```
# train data 와 test data로 분할(7:3 비율)
x_train, x_test, y_train, y_test = train_test_split(x_data, y_target, test_size=0.3,
                                                    random_state=156)
```

선형회귀 모델생성/ 모델학습 / 모델예측 / 모델평가 수행

```
lr = LinearRegression() # 모델생성
lr.fit(x_train, y_train) # 모델학습
y_preds = lr.predict(x_test) # 모델예측
print('Variance score : {0:.3f}'.format(r2_score(y_test, y_preds))) # 모델평가
# Variance score : 0.757
```

보스턴 주택가격 회귀분석

boston.py (4 / 5)

```
# LinearRegression 으로 생성한 주택가격 모델의 회귀계수(coefficients)와  
# 절편(intercept)을 구해보자  
# 회귀계수는 LinearRegression 객체의 coef_ 속성으로 구할수 있고,  
# 절편은 LinearRegression 객체의 intercept_ 속성으로 구할수 있다.
```

```
print('회귀계수값:', np.round(lr.coef_, 1))           # 소수 첫째자리  
print('절편값:', lr.intercept_)  
# 회귀계수값: [ -0.1  0.1  0.3. -19.8  3.4  0.  -1.7  0.4 -0.  -0.9  0. -0.6]  
# 절편값: 40.99559517216433
```

보스턴 주택가격 회귀분석

boston.py (5 / 5)

회귀계수를 큰 값 순으로 정렬하기 위해서 Series로 생성함.

```
coff = pd.Series(data=np.round(lr.coef_, 1), index=x_data.columns)
```

```
print(coff.sort_values(ascending=False))
```

회귀계수를 기준으로 내림차순 정렬

방개수

RM 3.4

CHAS 3.0

RAD 0.4

ZN 0.1

B 0.0

TAX -0.0

AGE 0.0

INDUS 0.0

CRIM -0.1

LSTAT -0.6

PTRATIO -0.9

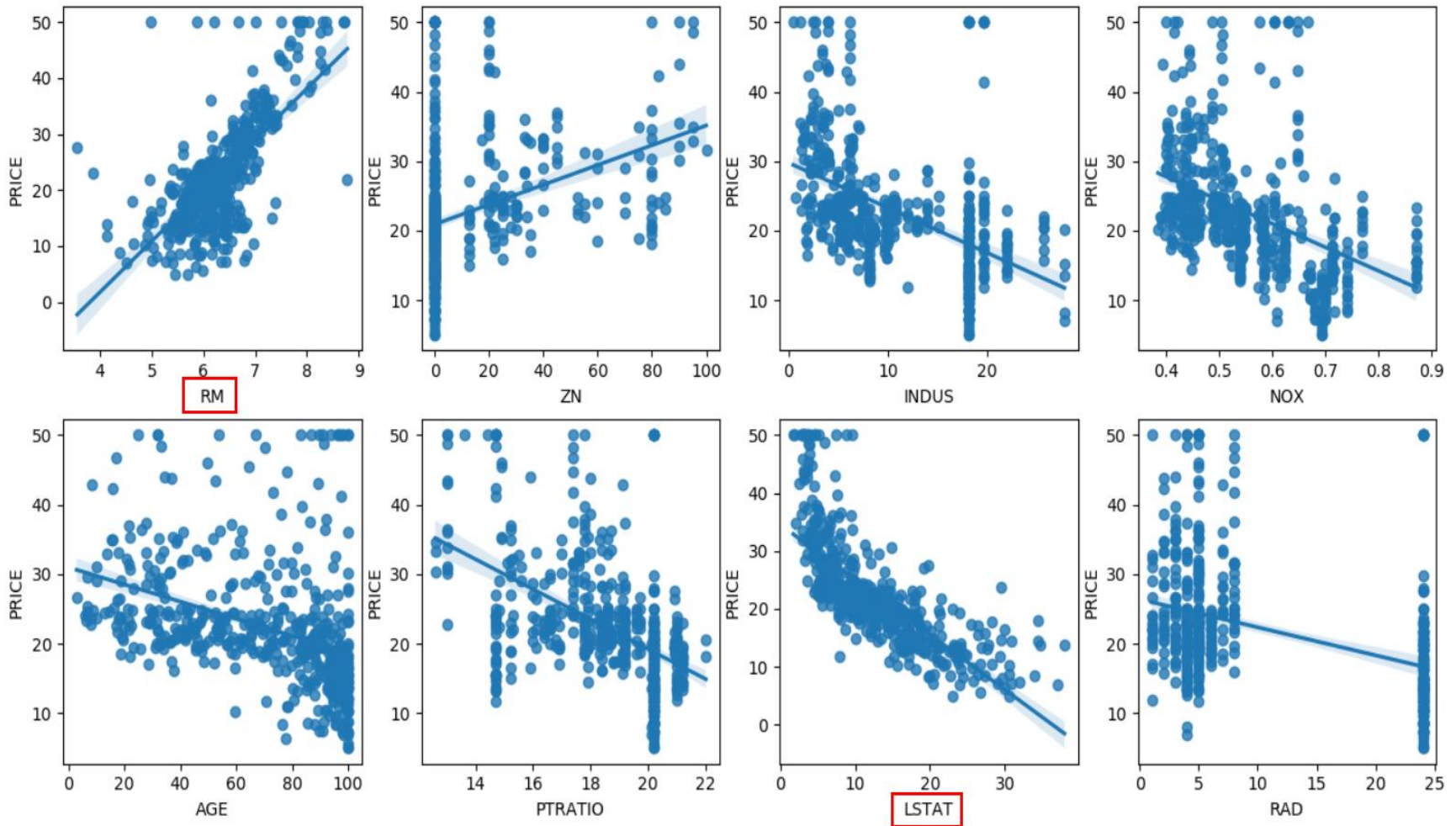
DIS -1.7

NOX -19.8

일산화질소 농도

보스톤 주택가격 회귀분석

❖ LinearRegression을 이용한 보스톤 주택 가격 회귀분석 : 결과



보스턴 주택가격 회귀분석

❖ 결과

- RM(방개수)와 LSTAT(하위 계층의 비율)이 PRICE에 영향도가 가장 두드러지게 나타남.
- RM(방개수)은 양 방향의 선형성(Positive Linearity)이 가장 크다.
방의 개수가 많을수록 가격이 증가하는 모습을 확연히 보여준다.
- LSTAT(하위 계층의 비율)는 음 방향의 선형성(Negative Linearity)이 가장 크다.
하위 계층의 비율이 낮을수록 PRICE 가 증가하는 모습을 확연히 보여준다.

회귀분석 예제

❖ 회귀분석 예제

➤ 단순회귀분석(Simple Linear Regression)

- 독립변수가 1개
- $Y = aX + b$

➤ 다항회귀분석(Polynomial Regression)

- 2차 방정식
- $Y = aX^2 + bX + c$

➤ 다중회귀분석(Multivariate Regression)

- 독립변수가 2개 이상
- $Y = a_1X_1 + a_2X_2 + c$

단순회귀분석 예제

❖ 단순회귀분석 예제

UCI(University of California, Irvine) 자동차 연비 데이터셋으로
단순회귀분석을 해보자

Step1. 데이터 준비

Step2. 데이터 탐색

Step3. 분석에 활용할 속성(feature 또는 variable) 선택 및 그래프 그리기
독립변수 : weight(중량), 종속변수 : mpg(연비)

Step4. 훈련 데이터 / 검증 데이터 분할

Step5. 모델 학습 및 모델 검증

단순회귀분석 예제

❖ Step1. 데이터 준비

UCI(University of California, Irvine) 자동차 연비 데이터셋으로
단순회귀분석을 해보자

<https://archive.ics.uci.edu/ml/datasets.php>

← → ↺ 주의 요함 | archive.ics.uci.edu/ml/datasets.php

UCI Machine Learning Repository
Center for Machine Learning and Intelligent Systems

About Citation Policy Donate a Data Set Contact

Search

Repository Web

View ALL Data Sets

Browse Through: 497 Data Sets

Default Task	Name	Data Types	Default Task	Attribute Types	# Instances	# Attributes	Year
Classification (306) Regression (112) Clustering (62) Other (55)	Abalone	Multivariate	Classification	Categorical, Integer, Real	4177	8	1995
Attribute Type	Adult	Multivariate	Classification	Categorical, Integer	48842	14	1996
Categorical (50) Numerical (331) Mixed (65)	UCI Annealing	Multivariate	Classification	Categorical, Integer, Real	798	38	
Data Type	UCI Anonymous Microsoft Web Data		Recommender-Systems	Categorical	37711	294	1998
Multivariate (383) Univariate (24) Sequential (52) Time-Series (102) Text (57) Domain Theory (23) Other (21)	Arrhythmia	Multivariate	Classification	Categorical, Integer, Real	452	279	1998
Area	Artificial Characters	Multivariate	Classification	Categorical, Integer, Real	6000	7	1992
Life Sciences (113) Physical Sciences (55) C.S. Engineering (179) Social Sciences (28) Business (32) Games (10) Other (75)	Audiology (Original)	Multivariate	Classification	Categorical	226		1987
# Attributes	Audiology (Standardized)	Multivariate	Classification	Categorical	226	69	1992
Less than 10 (122) 10 to 100 (223) Greater than 100 (90)	Auto MPG	Multivariate	Regression	Categorical, Real	398	8	1993
# Instances	Automobile	Multivariate	Regression	Categorical, Integer, Real	205	26	1987
Less than 100 (27) 100 to 1000 (169) Greater than 1000 (267)	UCI Badges	Univariate, Text	Classification		294	1	1994
Format Type	Balance Scale	Multivariate	Classification	Categorical	625	4	1994
Matrix (347) Non-Matrix (150)	Balloons	Multivariate	Classification	Categorical	16	4	


단순회귀분석 예제

❖ Step1. 데이터 준비

UCI(University of California, Irvine) 자동차 연비 데이터셋으로
단순회귀분석을 해보자

<https://archive.ics.uci.edu/ml/datasets/Auto+MPG>

← → ↻ ⓘ 주의 요함 | archive.ics.uci.edu/ml/datasets/Auto+MPG




UCI
Machine Learning Repository
Center for Machine Learning and Intelligent Systems

Auto MPG Data Set

Download: Data Folder [Data Set Description](#)

Abstract: Revised from CMU StatLib library, data concerns city-cycle fuel consumption



Data Set Characteristics:	Multivariate	Number of Instances:	398	Area:	N/A
Attribute Characteristics:	Categorical, Real	Number of Attributes:	8	Date Donated	1993-07-07
Associated Tasks:	Regression	Missing Values?	Yes	Number of Web Hits:	521691

단순회귀분석 예제






❖ Step1. 데이터 준비

UCI(University of California, Irvine) 자동차 연비 데이터셋으로
단순회귀분석을 해보자

<https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/>

← → ↻ 🔒 archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/

Index of /ml/machine-learning-databases/auto-mpg

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory		-	
 Index	1996-12-03 04:08	161	
 auto-mpg.data	1993-07-07 21:59	30K	
 auto-mpg.data-original	1993-07-07 21:59	31K	
 auto-mpg.names	1993-07-07 22:05	1.6K	

단순회귀분석 예제

❖ Step1. 데이터 준비

simple_linear_regression.py (1)

기본 라이브러리 불러오기

```
import numpy as np
```

```
import pandas as pd
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

CSV 파일을 읽어와서 데이터프레임으로 변환

```
df = pd.read_csv('auto-mpg.csv', header=None)
```

```
print(df) # 컬럼 번호로 출력 ( 0 ~ 8 )
```

단순회귀분석 예제

❖ Step1. 데이터 준비

simple_linear_regression.py (2)

열 이름 설정

```
df.columns = ['mpg','cylinders','displacement','horsepower','weight',  
              'acceleration','model year','origin','name']
```

데이터 살펴보기

```
print(df.head())          # 앞에서 5개의 데이터 출력  
print('\n')
```

IPython 디스플레이 설정 - 출력할 열의 개수 늘리기

```
pd.set_option('display.max_columns', 10)  
print(df.head())  
print('\n')
```

단순회귀분석 예제

❖ Step1. 데이터 준비 : 결과

IPython 디스플레이 설정 - 출력할 열의 개수 늘리기

```
pd.set_option('display.max_columns', 10)
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model	year	W
0	18.0	8	307.0	130.0	3504.0	12.0		70	
1	15.0	8	350.0	165.0	3693.0	11.5		70	
2	18.0	8	318.0	150.0	3436.0	11.0		70	
3	16.0	8	304.0	150.0	3433.0	12.0		70	
4	17.0	8	302.0	140.0	3449.0	10.5		70	

	origin	name
0	1	chevrolet chevelle malibu
1	1	buick skylark 320
2	1	plymouth satellite
3	1	amc rebel sst
4	1	ford torino

단순회귀분석 예제

❖ Step2. 데이터 탐색

simple_linear_regression.py (3)

데이터 자료형 확인

```
print(df.info())  
print('\n')
```

horsepower컬럼 object(문자형)

데이터 통계 요약정보 확인

```
print(df.describe())  
print('\n')
```

horsepower컬럼 object(문자형) 출력안됨

horsepower 열의 고유값 확인 : ['130.0' '165.0' '150.0' '140.0' ...]

```
print(df['horsepower'].unique())  
print('\n')
```

horsepower 열의 고유값 확인

horsepower 열의 자료형 object(문자형)

horsepower 열의 자료형 변경 (문자형 -> 실수형)

```
df['horsepower'].replace('?', np.nan, inplace=True)  
df.dropna(subset=['horsepower'], axis=0, inplace=True)  
df['horsepower'] = df['horsepower'].astype('float')
```

'?'을 np.nan으로 변경

누락데이터 행을 삭제

문자형을 실수형으로 변환

```
print(df.describe())
```

문자형에서 실수형으로 변환된 horsepower컬럼 출력됨

단순회귀분석 예제

❖ Step3. 분석에 활용할 속성(feature 또는 variable) 선택 및 그래프 그리기

simple_linear_regression.py (4)

분석에 활용할 열(속성)을 선택 (연비, 실린더, 출력, 중량)

```
ndf = df[['mpg', 'cylinders', 'horsepower', 'weight']]
```

```
print(ndf.head())
```

앞에서 5개의 데이터 출력

독립변수(x) weight(중량)와 종속 변수(y)인 mpg(연비) 간의 선형관계를 산점도 그래프로 확인

1.matplotlib으로 산점도 그리기

```
ndf.plot(kind='scatter', x='weight', y='mpg', c='coral', s=10, figsize=(10, 5))
```

```
plt.show()
```

```
plt.close()
```

2.seaborn으로 산점도 그리기

```
fig = plt.figure(figsize=(10, 5))
```

```
ax1 = fig.add_subplot(1, 2, 1)
```

```
ax2 = fig.add_subplot(1, 2, 2)
```

```
sns.regplot(x='weight', y='mpg', data=ndf, ax=ax1, fit_reg=True)
```

```
sns.regplot(x='weight', y='mpg', data=ndf, ax=ax2, fit_reg=False)
```

```
plt.show()
```

```
plt.close()
```

1행 2열 첫번째 그래프

1행 2열 두번째 그래프

회귀선 표시

회귀선 미표시

단순회귀분석 예제

- ❖ Step3. 분석에 활용할 속성(feature 또는 variable) 선택 및 그래프 그리기

simple_linear_regression.py (5)

3.seaborn 조인트 그래프 - 산점도, 히스토그램

```
sns.jointplot(x='weight', y='mpg', data=ndf)
```

회귀선 없음

```
sns.jointplot(x='weight', y='mpg', kind='reg', data=ndf)
```

회귀선 표시

```
plt.show()
```

```
plt.close()
```

4.seaborn pariplot으로 두 변수 간의 모든 경우의 수 그리기

```
sns.pairplot(ndf)
```

```
plt.show()
```

```
plt.close()
```


단순회귀분석 예제

❖ Step3. 분석에 활용할 속성(feature 또는 variable) 선택 및 그래프 그리기 : 결과

분석에 활용할 열(속성)을 선택 (연비, 실린더, 출력, 중량)

```
ndf = df[['mpg', 'cylinders', 'horsepower', 'weight']]
```

```
print(ndf.head())
```

앞에서 5개의 데이터 출력

	mpg	cylinders	horsepower	weight
0	18.0	8	130.0	3504.0
1	15.0	8	165.0	3693.0
2	18.0	8	150.0	3436.0
3	16.0	8	150.0	3433.0
4	17.0	8	140.0	3449.0

단순회귀분석 예제

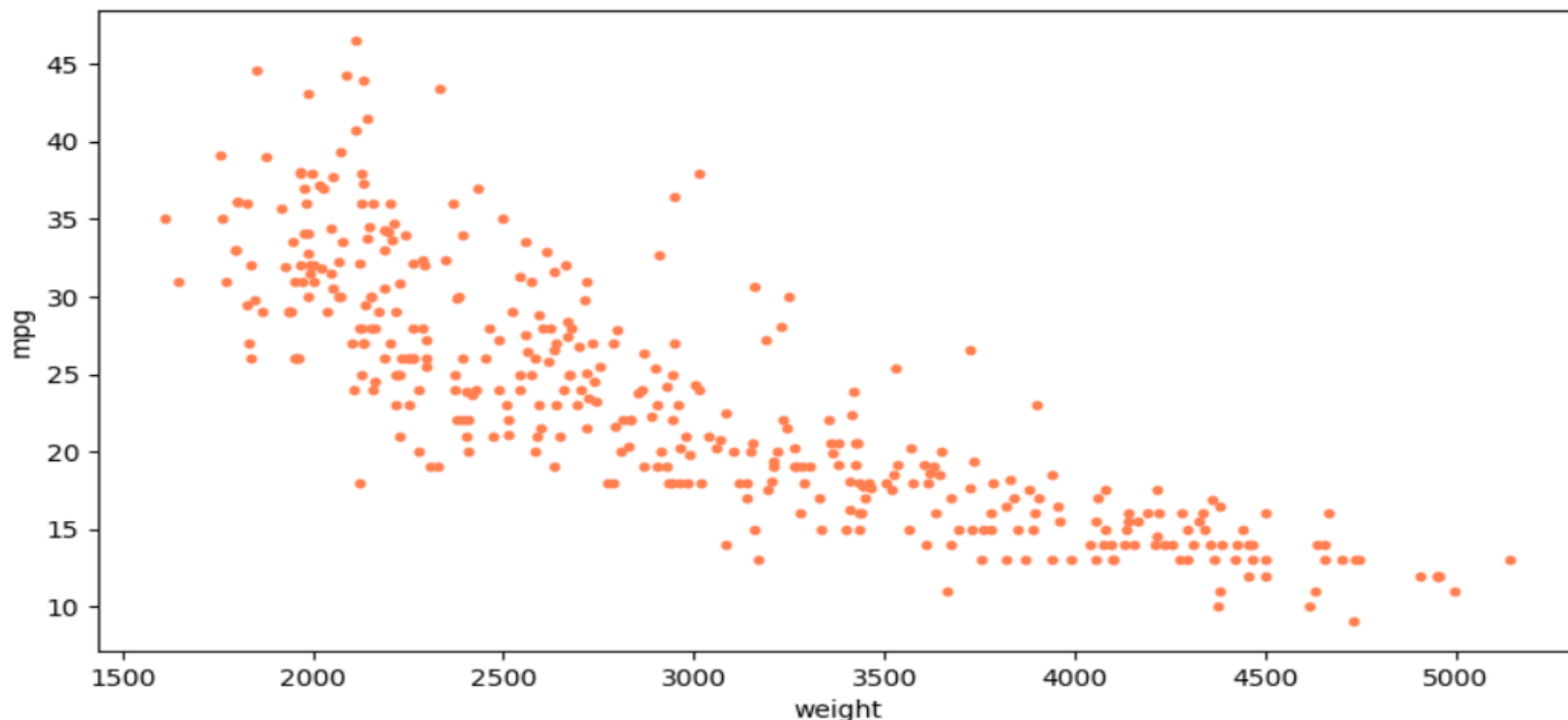
❖ Step3. 분석에 활용할 속성(feature 또는 variable) 선택 및 그래프 그리기 : 결과

독립변수(x) weight(중량)와 종속 변수(y)인 mpg(연비) 간의 선형관계를 산점도 그래프

1.matplotlib으로 산점도 그리기

```
ndf.plot(kind='scatter', x='weight', y='mpg', c='coral', s=10, figsize=(10, 5))
```

weight(중량)이 적을수록 mpg(연비)가 높은 것을 알 수 있다.



단순회귀분석 예제

❖ Step3. 분석에 활용할 속성(feature 또는 variable) 선택 및 그래프 그리기 : 결과

2.seaborn으로 산점도 그리기

```
fig = plt.figure(figsize=(10, 5))
```

```
ax1 = fig.add_subplot(1, 2, 1)
```

1행 2열 첫번째 그래프

```
ax2 = fig.add_subplot(1, 2, 2)
```

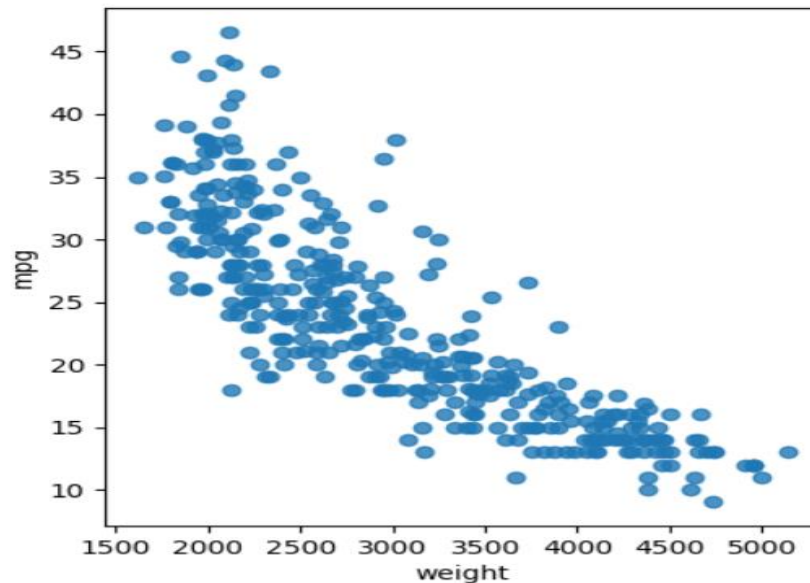
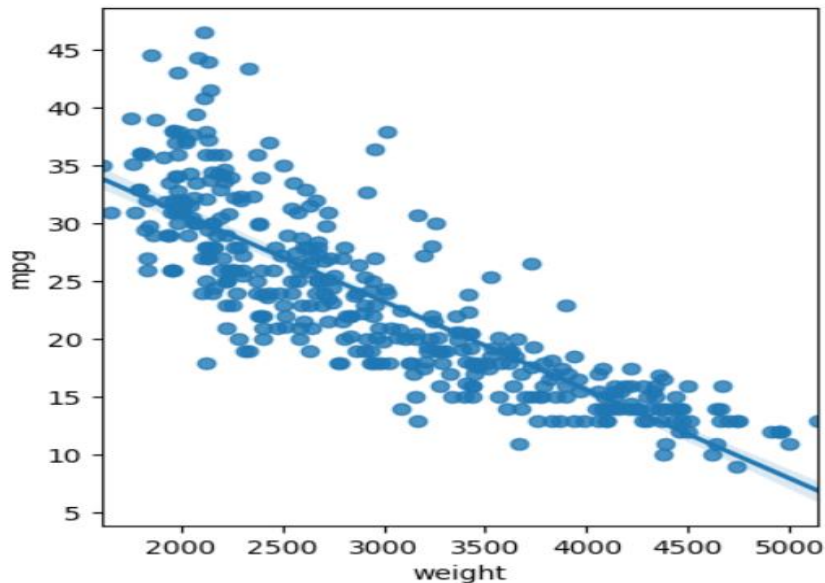
1행 2열 두번째 그래프

```
sns.regplot(x='weight', y='mpg', data=ndf, ax=ax1, fit_reg=True)
```

회귀선 표시

```
sns.regplot(x='weight', y='mpg', data=ndf, ax=ax2, fit_reg=False)
```

#회귀선 미표시



단순회귀분석 예제

❖ Step3. 분석에 활용할 속성(feature 또는 variable) 선택 및 그래프 그리기 : 결과

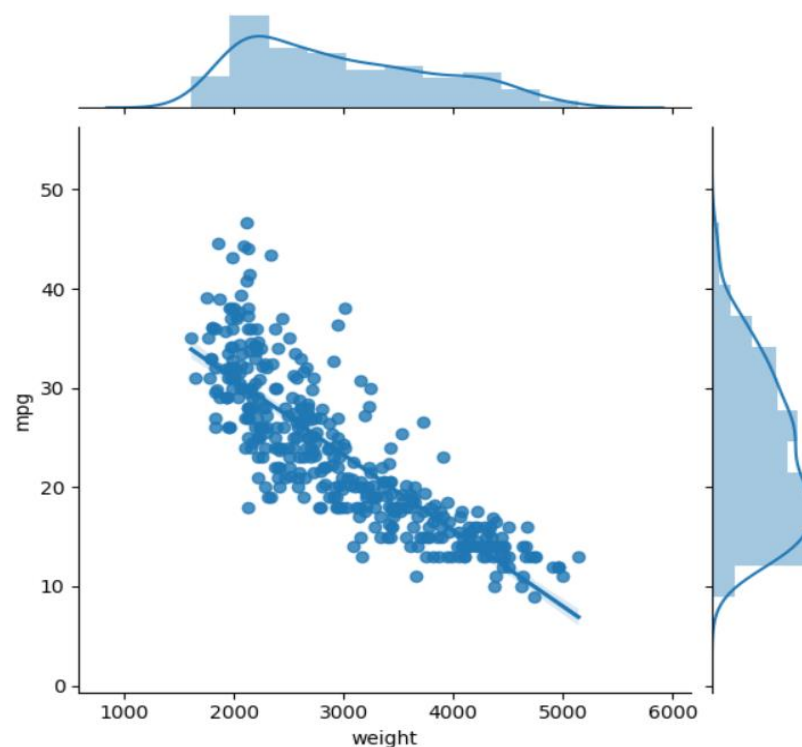
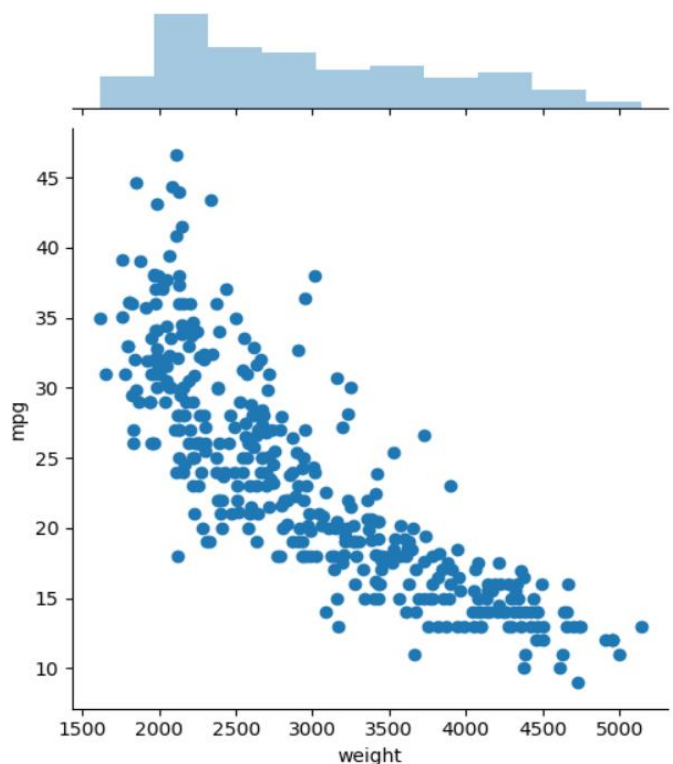
3.seaborn 조인트 그래프 - 산점도, 히스토그램

```
sns.jointplot(x='weight', y='mpg', data=ndf)
```

회귀선 없음

```
sns.jointplot(x='weight', y='mpg', kind='reg', data=ndf)
```

회귀선 표시



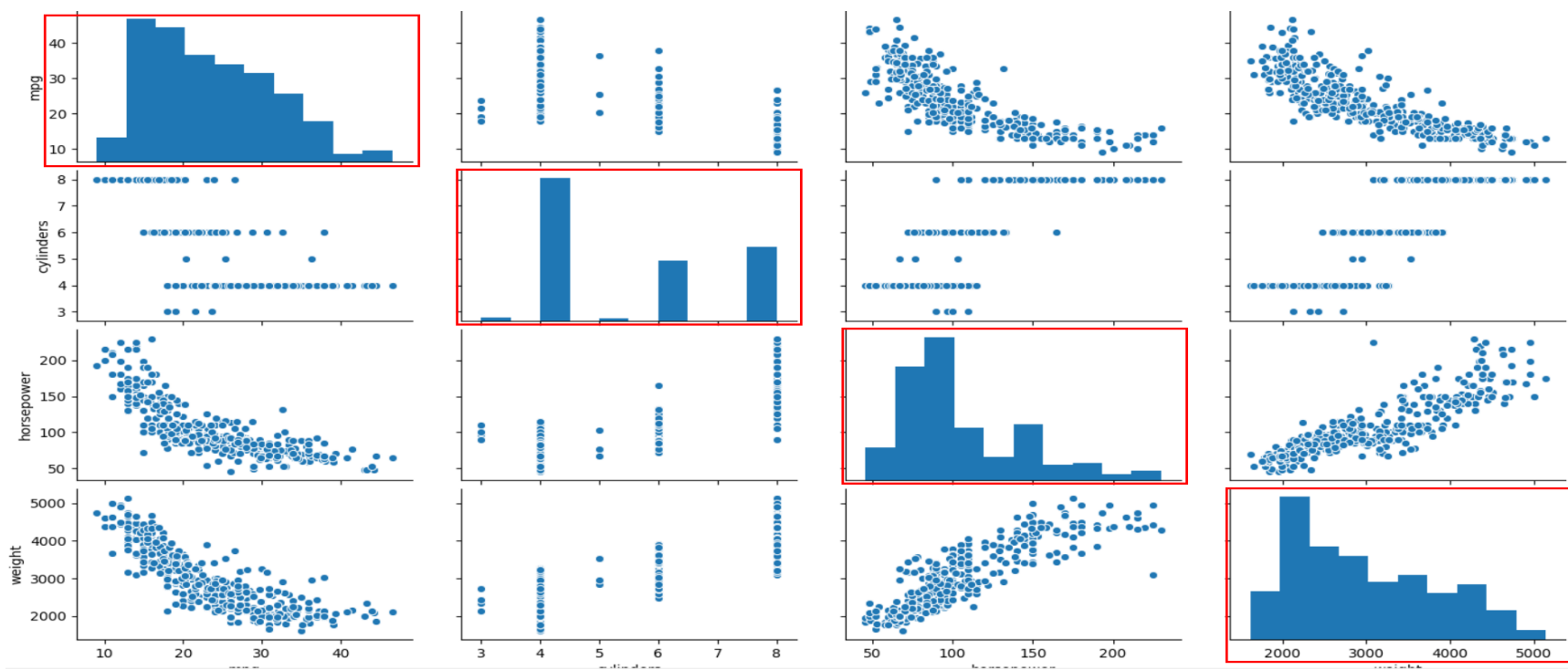
단순회귀분석 예제

❖ Step3. 분석에 활용할 속성(feature 또는 variable) 선택 및 그래프 그리기 : 결과

4.seaborn pariplot으로 두 변수 간의 모든 경우의 수 그리기

`sns.pairplot(ndf)`

같은 변수끼리 짝을 이루는 대각선 방향은 히스토그램을 그리고 서로 다른 변수간에는 산점도로 출력된다.



단순회귀분석 예제

❖ Step4. 훈련 데이터 / 검증 데이터 분할

앞에서 그린 산점도에서 'mpg' 열과 선형관계를 보이는 'weight' 열을 독립변수 x 로 선택한다.

다음은 두 변수 간의 회귀 방정식을 구하기 위해서 훈련 데이터와 검증 데이터로 나눠서 모델을 구축한다.

다음 예제에서는 'weight' 열을 독립 변수 x 로 선택하고, 데이터를 7:3 으로 각각 분할한다.

훈련 데이터 274개, 검증 데이터 118개로 분할한다.

weight – 독립변수(x)

mpg – 종속변수(y)

단순회귀분석 예제

❖ Step4. 훈련 데이터 / 검증 데이터 분할

simple_linear_regression.py (6)

속성(변수) 선택

x=ndf[['weight']]

y=ndf['mpg']

독립 변수 : x

종속 변수 : y

train data 와 test data로 분할(7:3 비율)

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x,

y,

test_size=0.3,

random_state=10)

독립 변수

종속 변수

검증 30%

랜덤 추출 값

print('train data 개수: ', len(x_train))

print('test data 개수: ', len(x_test))

단순회귀분석 예제

❖ Step4. 훈련 데이터 / 검증 데이터 분할 : 결과

train data 개수: 274 # 70 %

test data 개수: 118 # 30 %

단순회귀분석 예제

❖ Step5. 모델 학습 및 모델 검증

sklearn 라이브러리를 이용하여 선형회귀분석 모델을 만든다.

`LinearRegression()` 함수로 회귀분석 모델 객체를 생성한다.

생성된 모델을 이용하여 학습 데이터(train data)를 학습 시키기 위해서는 `fit()` 함수를 사용한다.

모델 객체에 `fit()` 함수를 적용하고 훈련 데이터(`x_train, y_train`)를 전달하면 모델이 학습을 통해 회귀 방정식의 계수 a (기울기), b (절편)를 찾는다.

학습을 마친 모델의 예측 능력을 평가하기 위해서 검증 데이터를 `score()` 함수에 전달하여 모델의 결정계수(R^2 -제곱)를 구한다.

결정 계수의 값이 1에 가까우면 모델의 예측 능력이 좋다고 평가한다.

단순회귀분석 예제

❖ Step5. 모델 학습 및 모델 검증

simple_linear_regression.py (7)

sklearn 라이브러리에서 선형회귀분석 모듈 가져오기

```
from sklearn.linear_model import LinearRegression
```

단순회귀분석 모델 객체 생성

```
lr = LinearRegression()
```

train data를 가지고 모델 학습

```
lr.fit(x_train, y_train)
```

학습을 마친 모델에 test data를 적용하여 결정계수(R-제곱) 계산

```
r_square = lr.score(x_test, y_test)
```

```
print(r_square)
```

```
print('\n')
```

결정계수 : 0.6822458558299325

단순회귀분석 예제

❖ Step5. 모델 학습 및 모델 검증

simple_linear_regression.py (8)

회귀식의 기울기

```
print('기울기 a: ', lr.coef_)  
print('\n')
```

[-0.00775343]

회귀식의 y절편

```
print('y절편 b', lr.intercept_)  
print('\n')
```

46.7103662572801

모델에 전체 x데이터를 입력하여 예측한 값 y_hat을 실제 값 y와 비교
y_hat = lr.predict(x)

예측값 구함

```
plt.figure(figsize=(10, 5))
```

```
ax1 = sns.distplot(y, hist=False, label="y")
```

실제 값

```
ax2 = sns.distplot(y_hat, hist=False, label="y_hat", ax=ax1)
```

예측한 값

```
plt.show()
```

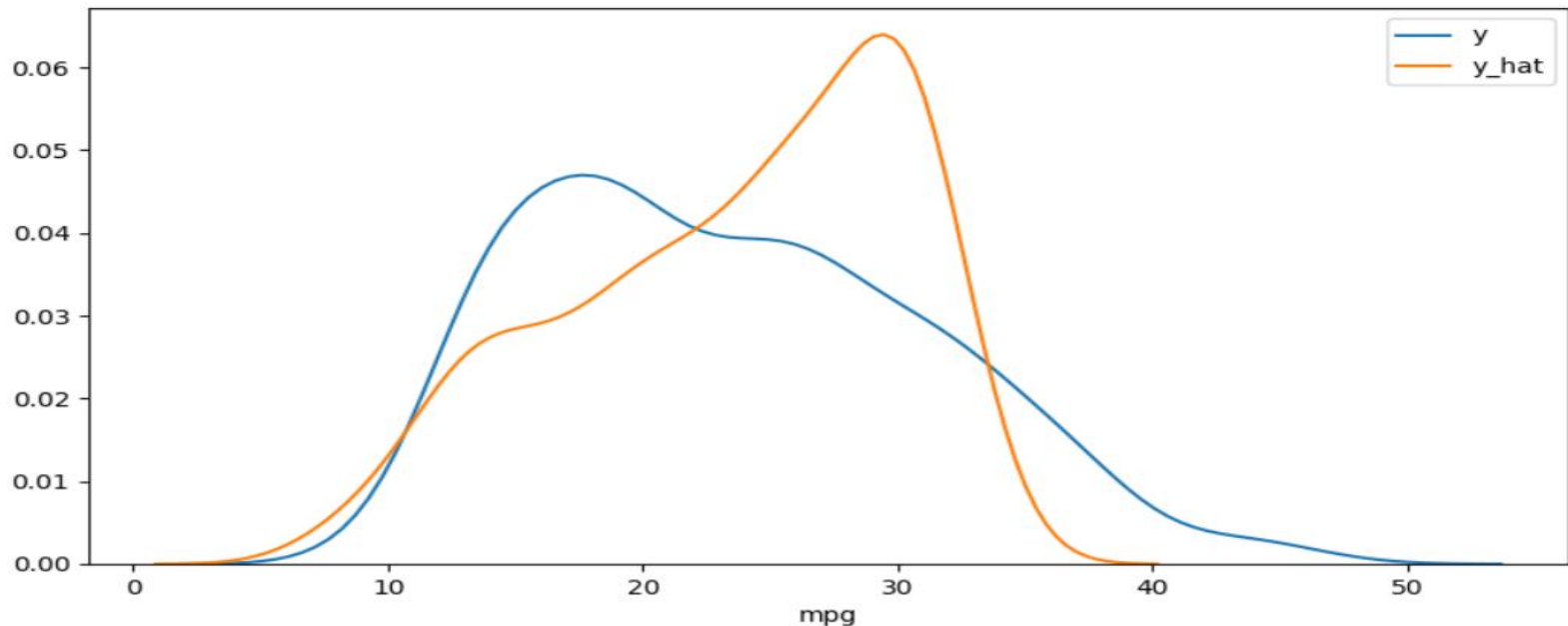
```
plt.close()
```

단순회귀분석 예제

❖ Step5. 모델 학습 및 모델 검증 : 결과

<모델이 예측한 값($y_{\hat{}}$)와 실제 값(y) 비교 결과>

출력된 결과를 보면 실제 값은 왼쪽으로 편향되어 있고, 예측값은 반대로 오른쪽으로 편중되는 경향을 보인다. 따라서 독립변수(weight)와 종속변수(mpg) 사이에 선형관계가 있지만, 모델의 오차를 더 줄일 필요가 있어 보인다.



다항회귀분석 예제

❖ 다항회귀분석(Polynomial Regression)

단순회귀분석은 두 변수 간의 관계를 직선 형태로 설명하는 알고리즘이다. 독립변수 x 와 종속변수 y 사이에 선형의 상관관계가 있지만, 직선보다는 곡선으로 설명하는 것이 적합할 때는 단순회귀분석은 부적합하다. 이를 경우에 다항 함수를 사용하면 보다 복잡한 곡선 형태의 회귀선을 표현할 수 있다.

다항회귀분석(Polynomial Regression)은 2차함수 이상의 다항 함수를 이용하여 두 변수 사이의 선형관계를 설명하는 알고리즘이다.

예를 들면, 2차함수는 종속변수 y 와 독립변수 x 사이의 관계를 $y = ax^2 + bx + c$ 로 표시하여 설명한다.

다항회귀분석 모형은 학습을 통해서 3개의 계수 a, b, c 를 찾아서 모델을 만든다.

다항회귀분석 예제

❖ 다항회귀분석 예제

UCI 자동차 연비 데이터셋으로 다항회귀분석을 해보자

Step1. 데이터 준비

Step2. 데이터 탐색

Step3. 분석에 활용할 속성(feature 또는 variable) 선택
독립변수 : weight(중량), 종속변수 : mpg(연비)

Step4. 훈련 데이터 / 검증 데이터 분할

Step5. 모델 학습 및 모델 검증

다항회귀분석 예제

❖ Step1. 데이터 준비

polynomial_regression.py (1)

기본 라이브러리 불러오기

```
import pandas as pd
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

CSV 파일을 읽어와서 데이터프레임으로 변환

```
df = pd.read_csv('auto-mpg.csv', header=None)
```

열 이름 지정

```
df.columns = ['mpg', 'cylinders', 'displacement', 'horsepower', 'weight',  
              'acceleration', 'model year', 'origin', 'name']
```

다항회귀분석 예제

❖ Step2. 데이터 탐색

polynomial_regression.py (2)

horsepower 열의 자료형 변경 (문자형 -> 실수형)

df['horsepower'].replace('?', np.nan, inplace=True) # '?'을 np.nan으로 변경

df.dropna(subset=['horsepower'], axis=0, inplace=True) # 누락데이터 행을 삭제

df['horsepower'] = df['horsepower'].astype('float') # 문자형을 실수형으로 변환

print(df.describe())

데이터 통계 요약정보 확인

print('\n')

다항회귀분석 예제

- ❖ Step3. 분석에 활용할 속성(feature 또는 variable) 선택
- ❖ Step4. 훈련 데이터 / 검증 데이터 분할

polynomial_regression.py (3)

분석에 활용할 열(속성)을 선택 (연비, 실린더, 출력, 중량)

```
ndf = df[['mpg', 'cylinders', 'horsepower', 'weight']]
```

ndf 데이터를 train data 와 test data로 구분(7:3 비율)

```
x=ndf[['weight']] #독립 변수 x
```

```
y=ndf['mpg'] #종속 변수 y
```

train data 와 test data로 구분(7:3 비율)

```
from sklearn.model_selection import train_test_split
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=10)
```

```
print('훈련 데이터: ', x_train.shape) # 훈련 데이터: (274, 1)
```

```
print('검증 데이터: ', x_test.shape) # 검증 데이터: (118, 1)
```

```
print('\n')
```

다항회귀분석 예제

❖ Step5. 모델 학습 및 모델 검증

polynomial_regression.py (4)

sklearn 라이브러리에서 필요한 모듈 가져오기

from sklearn.linear_model import LinearRegression # 선형회귀분석

from sklearn.preprocessing import PolynomialFeatures # 다항식 변환

다항식 변환

poly = PolynomialFeatures(degree=2)

2차항 적용

x_train_poly=poly.fit_transform(x_train)

x_train 데이터를 2차항으로 변환

print(x_train_poly) # x_train의 1개의 열이 x_train_poly 에서는 3개의 열로 늘어난다.

print('원 데이터: ', x_train.shape)

원 데이터: (274, 1)

print('2차항 변환 데이터: ', x_train_poly.shape)

2차항 변환 데이터: (274, 3)

print('\n') # x_train의 1개의 열이 x_train_poly 에서는 3개의 열로 늘어난다.

다항회귀분석 예제

❖ Step5. 모델 학습 및 모델 검증

polynomial_regression.py (5)

train data를 가지고 모델 학습

```
pr = LinearRegression()
```

```
pr.fit(x_train_poly, y_train)
```

모델 만들기

모델 학습

학습을 마친 모델에 test data를 적용하여 결정계수(R-제곱) 계산

```
x_test_poly = poly.fit_transform(x_test)
```

```
r_square = pr.score(x_test_poly, y_test)
```

```
print(r_square)
```

```
print('\n')
```

x_test 데이터를 2차항으로 변환

결정계수 구하기

결정계수 : 0.7087009262975481

train data의 산점도와 test data로 예측한 회귀선을 그래프로 출력

```
y_hat_test = pr.predict(x_test_poly)
```

test data로 예측하기

다항회귀분석 예제

❖ Step5. 모델 학습 및 모델 검증

polynomial_regression.py (6)

train data의 산점도와 test data로 예측한 회귀선을 그래프로 출력

```
fig = plt.figure(figsize=(10, 5))
```

```
ax = fig.add_subplot(1, 1, 1)
```

```
ax.plot(x_train, y_train, 'o', label='Train Data')
```

```
ax.plot(x_test, y_hat_test, 'r+', label='Predicted Value')
```

```
ax.legend(loc='best')
```

```
plt.xlabel('weight')
```

```
plt.ylabel('mpg')
```

```
plt.show()
```

```
plt.close()
```

그래프 크기 설정

train data의 산점도

모델이 학습한 회귀선

범례 설정

x축 라벨

y축 라벨

다항회귀분석 예제

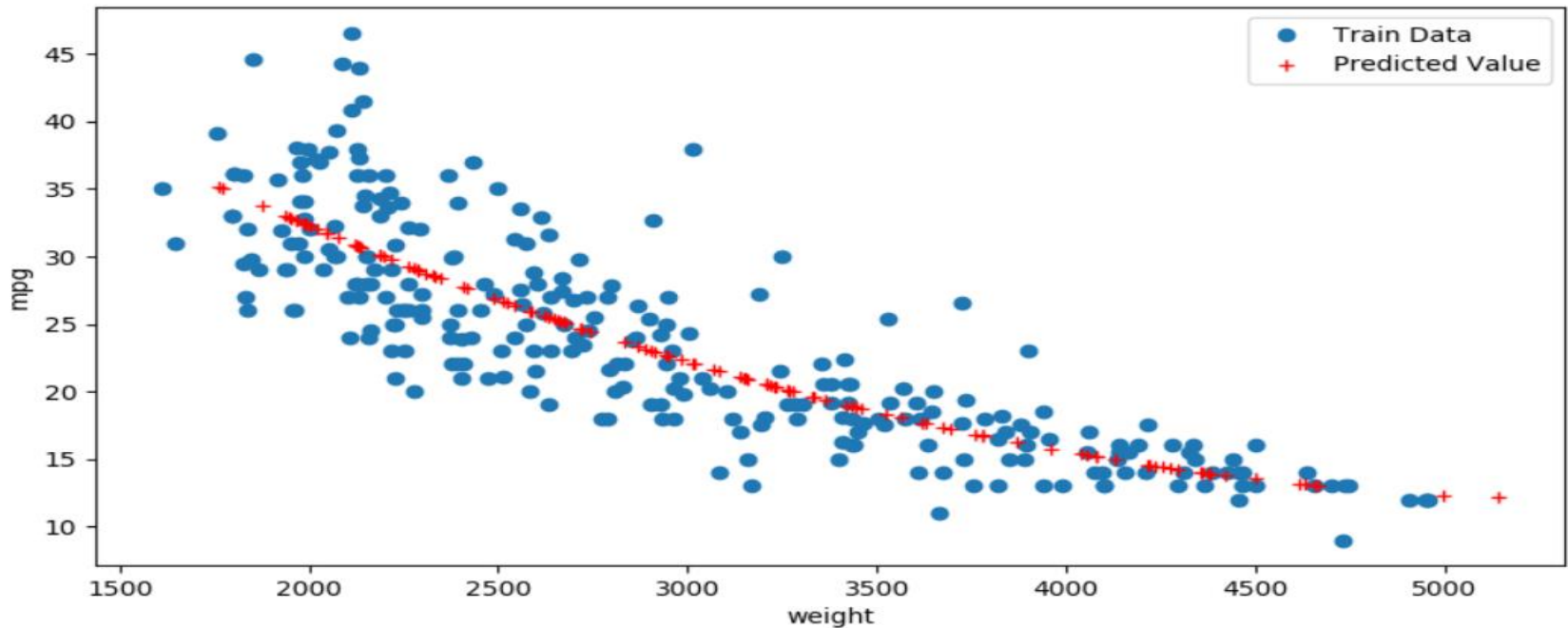
❖ Step5. 모델 학습 및 모델 검증 : 결과

단순회귀분석 결정계수 : 0.6822458558299325

다항회귀분석 결정계수 : 0.7087009262975481

단순회귀분석을 했을때 보다 다항회귀분석을 했을때 결정계수값이 높아진 것에서 알수 있듯이 직선보다 곡선으로 만들어진 회귀선이 데이터 패턴을 더욱 더 잘 설명한다고 할 수 있다.

train data의 산점도와 test data로 예측한 회귀선을 그래프로 출력 : 곡선모양의 회귀선



다항회귀분석 예제

❖ Step5. 모델 학습 및 모델 검증

polynomial_regression.py (7)

실제값 y와 예측값 y_hat 의 분포 차이 비교

모델에 전체 x데이터를 입력하여 예측한 값 y_hat을 실제 값 y와 비교

x_ploy = poly.fit_transform(x)

x 데이터를 2차항으로 변환

y_hat = pr.predict(x_ploy)

예측값 구하기

displot() 함수 : 히스토그램 + 커널밀도함수

plt.figure(figsize=(10, 5))

그래프 크기 설정

ax1 = sns.distplot(y, hist=False, label="y")

실제값

ax2 = sns.distplot(y_hat, hist=False, label="y_hat", ax=ax1)

예측값

plt.show()

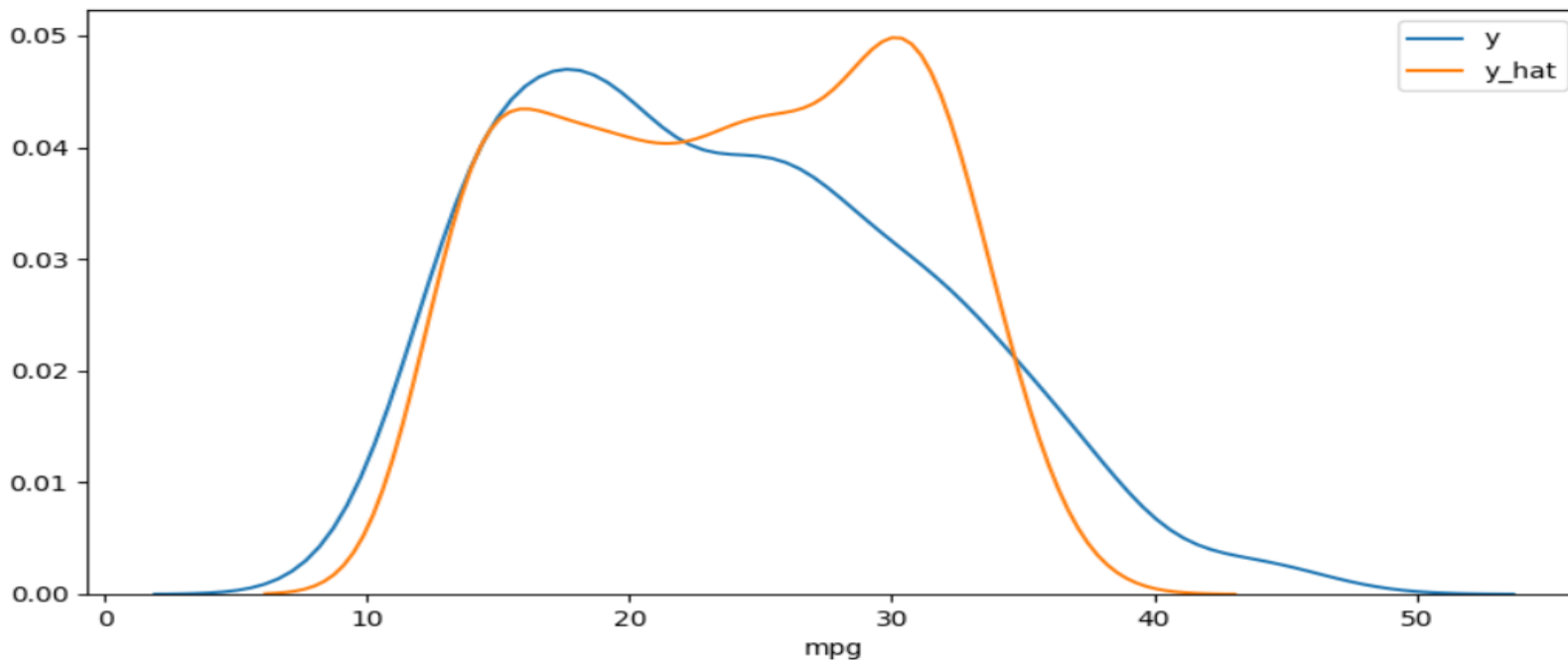
plt.close()

다항회귀분석 예제

❖ Step5. 모델 학습 및 모델 검증 : 결과

단순회귀분석의 결과와 비교할 때 데이터가 어느 한쪽으로 편향되는 경향이 상당히 감소한 것을 알 수 있다. 다항회귀분석이 더 적합한 모델이라고 볼 수 있다.

실제값 y 와 예측값 y_{hat} 의 분포 차이 비교



다중회귀분석 예제

❖ 다중회귀분석(Multivariate Regression)

단순회귀분석은 소득이 증가하면 소비도 증가하는 것처럼 종속변수에 y 에 영향을 주는 독립변수가 x 가 하나인 경우를 말한다. 하지만 소비에 영향을 주는 독립변수에는 소득 외에도 자녀의 수, 거주지, 직업 등 다른 요인이 있을 수 있다.

이처럼 여러 개의 독립변수가 종속 변수에 영향을 주고 선형 관계를 갖는 경우에 다중회귀분석(Multivariate Regression)을 사용한다.

수학적으로 종속변수 y 와 독립변수 x 간의 관계를 $Y = b + a_1X_1 + a_2X_2 + \dots + a_nX_n$ 와 같은 함수식으로 표현한다.

다중회귀분석 알고리즘은 각 독립 변수의 계수(a_1, a_2, \dots, a_n) 와 상수항(b)에 적절한 값들을 찾아서 모델을 완성한다.

모델의 예측값인 종속 변수에 대한 실제 데이터를 알고 있는 상태에서 학습하기 때문에 지도학습으로 분류된다.

다중회귀분석 예제

❖ 다중회귀분석 예제

UCI 자동차 연비 데이터셋으로 다중회귀분석을 해보자

Step1. 데이터 준비

Step2. 데이터 탐색

Step3. 분석에 활용할 속성(feature 또는 variable) 선택

독립변수 : cylinders(실린더), horsepower(마력), weight(중량)

종속변수 : mpg(연비)

Step4. 훈련 데이터/ 검증 데이터 분할

Step5. 모델 학습 및 모델 검증

다중회귀분석 예제

❖ Step1. 데이터 준비

multivariate_regression.py (1)

#기본 라이브러리 불러오기

```
import pandas as pd
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

CSV 파일을 읽어와서 데이터프레임으로 변환

```
df = pd.read_csv('auto-mpg.csv', header=None)
```

열 이름 지정

```
df.columns = ['mpg','cylinders','displacement','horsepower','weight',  
              'acceleration','model year','origin','name']
```

다중회귀분석 예제

❖ Step2. 데이터 탐색

multivariate_regression.py (2)

horsepower 열의 자료형 변경 (문자형 -> 실수형)

df['horsepower'].replace('?', np.nan, inplace=True)

df.dropna(subset=['horsepower'], axis=0, inplace=True)

df['horsepower'] = df['horsepower'].astype('float')

print(df.describe())

print('\n')

'?'을 np.nan으로 변경

누락데이터 행을 삭제

문자형을 실수형으로 변환

데이터 통계 요약정보 확인

다중회귀분석 예제

- ❖ Step3. 분석에 활용할 속성(feature 또는 variable) 선택
- ❖ Step4. 훈련 데이터 / 검증 데이터 분할

multivariate_regression.py (3)

분석에 활용할 열(속성)을 선택 (연비, 실린더, 출력, 중량)

```
ndf = df[['mpg', 'cylinders', 'horsepower', 'weight']]
```

독립변수와 종속변수 선택

```
x=ndf[['cylinders', 'horsepower', 'weight']]
```

독립 변수 : cylinders, horsepower, weight

```
y=ndf['mpg']
```

종속 변수 : mpg

train data 와 test data로 구분(7:3 비율)

```
from sklearn.model_selection import train_test_split
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=10)
```

```
print('훈련 데이터: ', x_train.shape)
```

훈련 데이터: (274, 3)

```
print('검증 데이터: ', x_test.shape)
```

검증 데이터: (118, 3)

```
print('\n')
```

다중회귀분석 예제

❖ Step5. 모델 학습 및 모델 검증

multivariate_regression.py (4)

```
# sklearn 라이브러리에서 선형회귀분석 모듈 가져오기  
from sklearn.linear_model import LinearRegression
```

```
# 단순회귀분석 모델 객체 생성  
lr = LinearRegression()
```

모델 만들기

```
# train data를 가지고 모델 학습  
lr.fit(x_train, y_train)
```

모델 학습

```
# 모델 평가
```

```
# 모델 학습이 완료된 다음에 검증데이터(x_test, y_test)를 사용하여 모델의 평가지표인  
# 결정계수(R-제곱) 계산
```

```
r_square = lr.score(x_test, y_test)  
print(r_square)  
print('\n')
```

결정계수 : 0.6939048496695599

다중회귀분석 예제

❖ Step5. 모델 학습 및 모델 검증

multivariate_regression.py (5)

회귀식의 기울기 : 독립변수 3개의 기울기가 리스트로 리턴됨

```
print('X 변수의 계수 a: ', lr.coef_)          # a : [-0.60691288 -0.03714088 -0.00522268]
print('\n')
```

회귀식의 y절편

```
print('상수항 b', lr.intercept_)            # b : 46.414351269634025
print('\n')
```

모델이 예측한 값과 실제값을 비교

```
y_hat = lr.predict(x_test)                  # 예측값 구하기
print(y_hat)
```

```
plt.figure(figsize=(10, 5))
```

```
ax1 = sns.distplot(y_test, hist=False, label="y_test")      # 실제값
```

```
ax2 = sns.distplot(y_hat, hist=False, label="y_hat", ax=ax1) # 예측값
```

```
plt.show()
```

```
plt.close()
```

다중회귀분석 예제

❖ Step5. 모델 학습 및 모델 검증 : 결과

다중회귀분석 결정계수 : 0.6939048496695599

다중회귀분석을 했을때 모델의 성능 지표인 결정계수값은 0.6939048496695599 으로 비교적 양호한 수준이다.

단순회귀분석의 결과와 비교할 때 데이터가 어느 한쪽으로 편향되는 경향은 그대로 남아 있지만 그래프의 첨도(뾰족한 정도)가 약간 누그러진 것을 볼 수 있다.

모델이 예측한값($y_{\hat{}}$)과 실제값(y_{test}) 분포 비교

