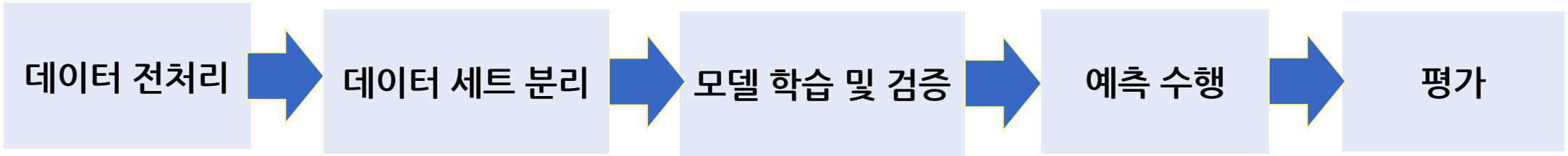
[지도 학습 : 회귀]

머신러닝 지도학습 프로세스

- 머신러닝 지도학습 프로세스



- 데이터 클린징
- 결손값 처리(Null/NaN)
- 데이터 인코딩
  (레이블 인코딩, 원-핫 인코딩)
- 피처 스케일링과 정규화
- 이상치 제거
- Feature 선택, 추출 및 가공

- 학습/테스트 데이터 분리
- train_test_split()

- 알고리즘 학습
- fit()

- 테스트 데이터로
  예측 수행
- predict()

- 예측 평가
- sklearn.metrics()

지도학습의 종류

- 지도학습의 구분 - Y(타겟 변수)의 자료 형태에 따라 분류와 회귀 2가지로 나누어짐

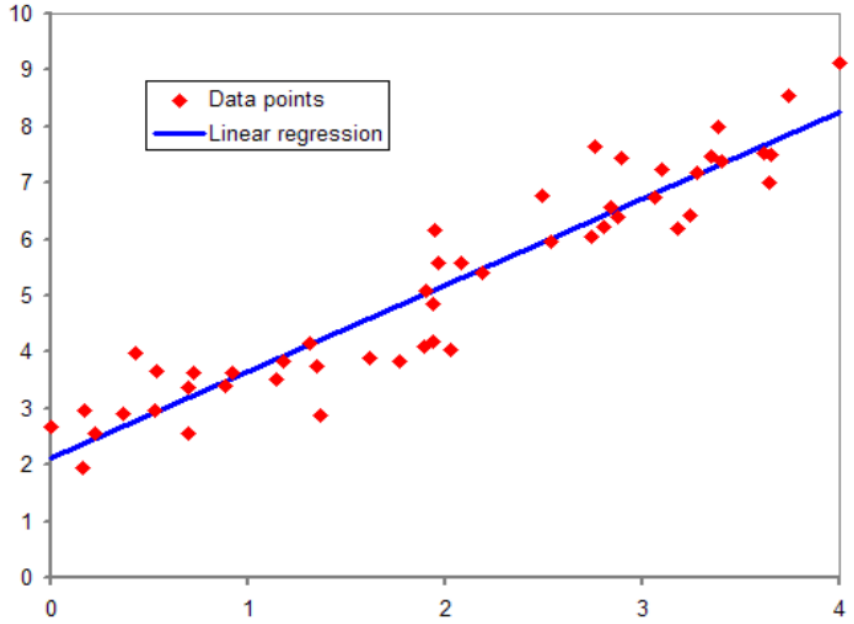
- 회귀(Regression)
  - 타겟 변수 Y가 연속형 변수(continuous variable) 인 경우
  - 연속형 변수 - 연속 범위 내에서 임의의 값을 가질 수 있는 변수, 가질 수 있는 값이 무한정

- 분류(Classification)
  - 타겟 변수 Y가 이산형 변수(discrete variable)인 경우
  - 이산형 변수 - 특정한 값만 가질 수 있는 변수, 가질 수 있는 값이 유한정

지도학습 : 회귀

- **회귀(Regression)이란?**
  - 유전적 특성을 연구하던 영국의 통계학자 갈톤(Galton)이 수행한 연구에서 유래
    - 아버지와 자식의 키 관계 연구 → "사람의 키는 평균 키로 회귀하려는 경향을 가진다는 자연의 법칙이 있다"
  - 여러 개의 독립변수와 한 개의 종속변수 간의 상관관계를 모델링 하는 기법
  - 시간에 따라 변화하는 데이터나 어떤 영향, 가설적 실험, 인과 관계의 모델링 등의 통계적 예측에 사용

- **회귀의 종류(독립변수의 개수에 따라 구분)**
  - 단순 회귀분석(simple regression analysis)
    - 하나의 종속변수와 하나의 독립변수 사이의 관계를 분석
  - 다중 회귀분석(multiple regression analysis)
    - 하나의 종속변수와 여러 독립변수 사이의 관계를 규명



- **회귀 모델 평가(Evaluation)**
  - R^2(결정계수), Adjusted R^2(수정결정계수)
  - RMSE

지도학습 : 회귀

참고) 규제(Regularization) : 일반적인 선형 회귀의 과적합의 문제를 해결하기 위해서 회귀계수에 패널티 값을 적용하는 것

- 회귀의 종류
  - 일반 선형 회귀
    - 예측값과 실제값의 RSS(Residual Sum of Squares)를 최소화 할 수 있도록 회귀계수를 최적화하며, 규제(Regularization)을 적용하지 않은 모델
  - 릿지(Ridge)
    - 선형회귀에 L2 규제를 추가한 회귀모델, L2는 규제는 상대적으로 큰 회귀 계수 값의 예측 영향도를 감소시키기 위해서 회귀 계수 값을 더 작게 만드는 규제 모델
  - 라쏘(Rasso)
    - 선형회귀애 L1 규제를 적용한 회귀모델, L1 규제는 예측 영향력이 작은 피처의 회귀 계수를 0으로 만들어 회귀 예측 시 피처가 선택되지 않도록 하는 방법, 피처 선택 기능이라고도 불림
  - 엘라스틱넷(ElasticNet)
    - L2, L1 규제를 함께 결합한 모델, 주로 피처가 많은 데이터 세트에서 적용되며, L1 규제로 피처의 개수를 줄이고 L2 규제로 회귀계수의 값의 크기를 조정
  - 로지스틱 회귀(Logistic Regression)
    - 분류에서 사용되는 선형 모델, 이진 분류에서 매우 강력한 분류 알고리즘
    - 텍스트 분류에도 뛰어난 예측 성능을 보임

지도학습 : 회귀

선형회귀(Linear Regression)

- 선형 회귀는 종속 변수(Y)와 한 개 이상의 독립 변수(X)와의 선형 상관 관계를 모델링하는 회귀 분석 기법
  1) 단순 선형회귀 : $y = wx + b$ w : 계수(가중치) , b : 절편(편향)
  2) 다중 선형회귀: $y = w_0x_0 + w_1x_1 + w_2x_2 + ... w_nx_n + b$

- 선형 회귀의 비용 함수

$$Cost_{lr} = \sum_i (y_i - \hat{y}_i)^2$$
$$\hat{y}_i = b + wx_i$$

  - 실제 참값과 회귀 모델이 출력한 예측값 사이의 잔차의 제곱의 합을 최소화하는 w(계수)를 구하는 것이 목적 → Least Square, 최소 제곱법

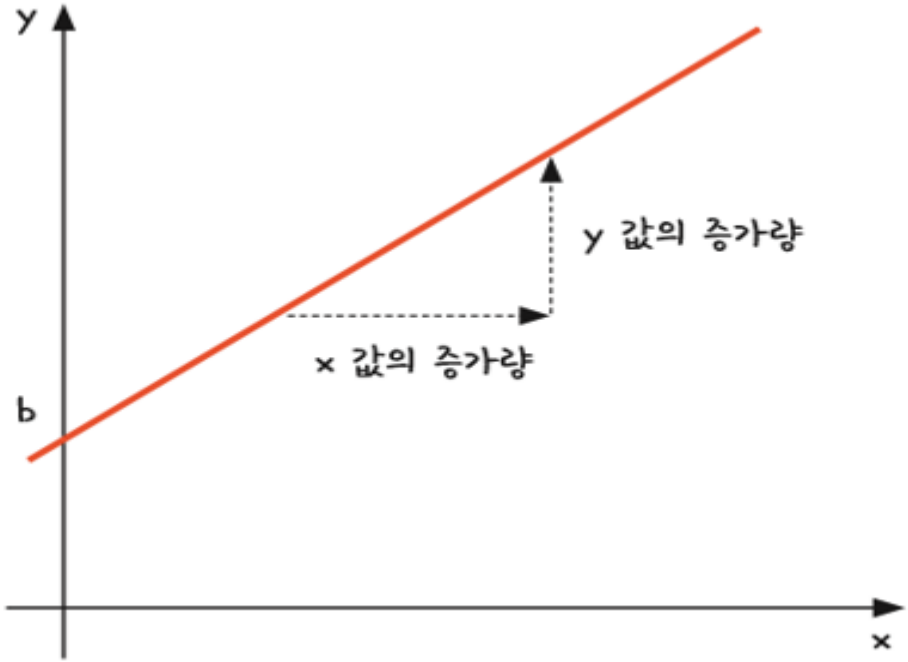
모델 평가(Evaluation)

- R^2
- 학습한 회귀 모델이 얼마나 데이터를 잘 표현하는지에 대한 정도를 나타내는 통계적인 척도이며,
  $0 < R^2 < 1$ 범위의 값을 갖는다. 1에 가까울수록 회귀모델이 데이터를 잘 표현한다는 것을 의미

지도학습 : 회귀

- 단순 선형 회귀(Linear Regression)
  - 독립변수도 하나이고 종속변수도 하나인 선형회귀

- 일차 함수 그래프
  - A는 직선의 기울기, 즉 $\frac{y\text{값의 증가량}}{x\text{값의 증가량}}$ 이고, b는 y축을 지나는 값인 'y 절편'이 됨



$$y = ax + b$$

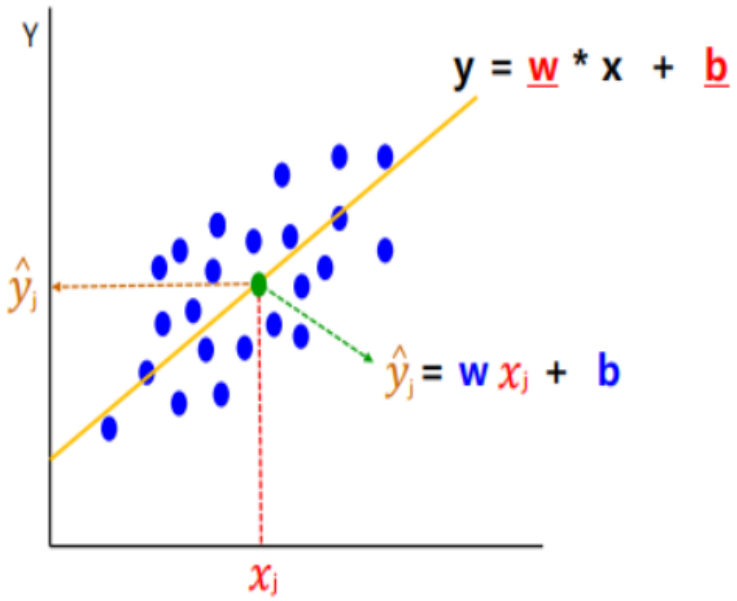
지도학습 : 회귀

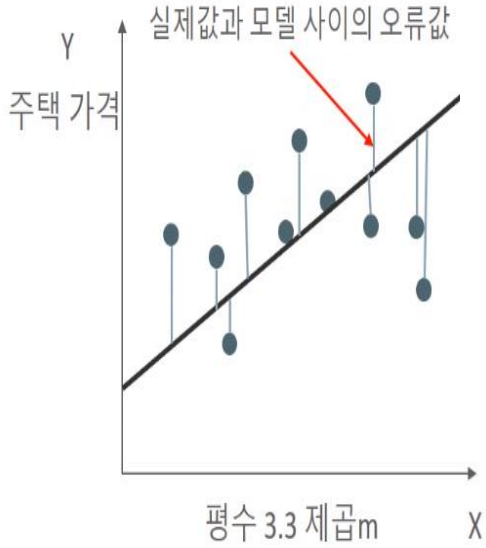
- 주택 가격 예측 : 주택 시세가 평수로만 결정된다고 가정

Tabular Form

	Xi(평수)	Yi(시세)
	2	100
	5	150
	10	190
	20	250
New Data	x_j	$\hat{y}_j(?)$
	.	.
	.	.
	Xn	Yn

Graphical Form





- 선형 회귀(Linear Regression)
  - 다중 선형 회귀에서 피처 간의 상관관계가 매우 높은 경우 분산이 매우 커져서 오류에 매우 민감해짐
    → 다중 공선성(multi-collinerarity) → 상관관계가 높은 피처 중 중요한 피처만 남기고 제거 및 규제
  - 사이킷런의 LinearRegression 클래스로 구현

지도학습 : 회귀

- 사이킷런의 LinearRegression 클래스
  - 예측 값과 실제 값의 RSS(Residual Sum of Squares)를 최소화해 OLS(Ordinary Least Squares) 추정 방식으로 구현한 클래스
  - fit() 메서드 - 테스트 데이터 x와 레이블 y 배열을 입력 받아 회귀 계수(Coefficients)인 W를 coef_ 속성에 저장

  - Class sklearn.linear_model.LinearRegression(fit_intercept=True, normalize=False, copy_X=True, n_jobs=1)

  - 입력파라메터
    - - fit_intercept : 불린값으로 디폴트는 True, intercept(절편) 값을 계산할 것인지를 지정
      만일 False로 지정하면 절편이 사용되지 않고 0으로 지정
  - 속성
    - coef_ : fit() 메서드를 수행했을 때 회귀 계수가 배열 형태로 저장되는 속성
      shape는 (target 값 개수, 피처 개수)
    - Intercept_ : intercept(절편) 값

지도학습 : 회귀

- 회귀 모델 평가 시 사용하는 지표
  - Residuals(잔차) : 실제 값과 예측 값의 차이(오차)

$$\text{Residual Error} = y - \hat{y}$$

  - Mean Squared Error(MSE, 평균제곱오차) : 오차를 제곱의 합으로 계산

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y})^2$$

  - Root Mean Squared Error(RMSE) : MSE에 루트를 씌워 실제 값과 유사한 값으로 변경

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y})^2}$$

  - R^2
    - 분산 기법으로 예측 성능을 평가
    - 실제 값의 분산 대비 예측 값의 분산 비율을 지표로 하며, 1에 가까울 수록 예측 정확도가 높음

$$R^2 = \frac{\text{예측값 분산}}{\text{실제값 분산}}$$

지도학습 : 선형 회귀

패키지 불러오기

```
# 패키지 불러오기
from sklearn import datasets
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
from sklearn import linear_model
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

보스톤 주택 데이타셋 불러오기

```
# 보스톤 주택 데이타셋 불러오기
from sklearn.datasets import load_boston
boston = load_boston()
```

```
data = boston.data    #data
label = boston.target #label
columns = boston.feature_names
```

```
data = pd.DataFrame(data, columns = columns)
data.head(3)
```

	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

지도학습 : 선형 회귀

```
1  # boston 데이타 살펴보기
2  print(boston.DESCR)
```

```
.. _boston_dataset:

Boston house prices dataset
---------------------------

**Data Set Characteristics:**
```

- CRIM : 지역별 범죄 발생률
- ZN : 25,000평방피트를 초과하는 거주 지역의 비율
- INDUS : 비상업 지역 넓이 비율
- CHAS : 찰스강에 대한 더미 변수(강의 경계에 위치한 경우는 1, 아니면 0)
- NOX : 일산화질소 농도
- RM : 거주할 수 있는 방의 개수
- AGE : 1940년 이전에 건축된 소유 주택의 비율
- DIS : 5개 주요 고용센터까지의 가중 거리
- RAD : 고속도로 접근 용이도
- TAX : 10,000당 재산세율
- PTRATIO : 지역의 교사와 학생수 비율
- B : 지역의 흑인 거주 비율
- LSTAT : 하위 계층의 비율
- MEDV : 본인 소유의 주택 가격(중앙값)

지도학습 : 선형 회귀

```
# 보스톤 데이터세트의 target 배열은 주택가격임, 이를 PRICE 컬럼으로 데이터프레임에 추가
data['PRICE'] = boston.target
print('보스톤 데이터세트 크기 : ', data.shape)
data.head(2)
```

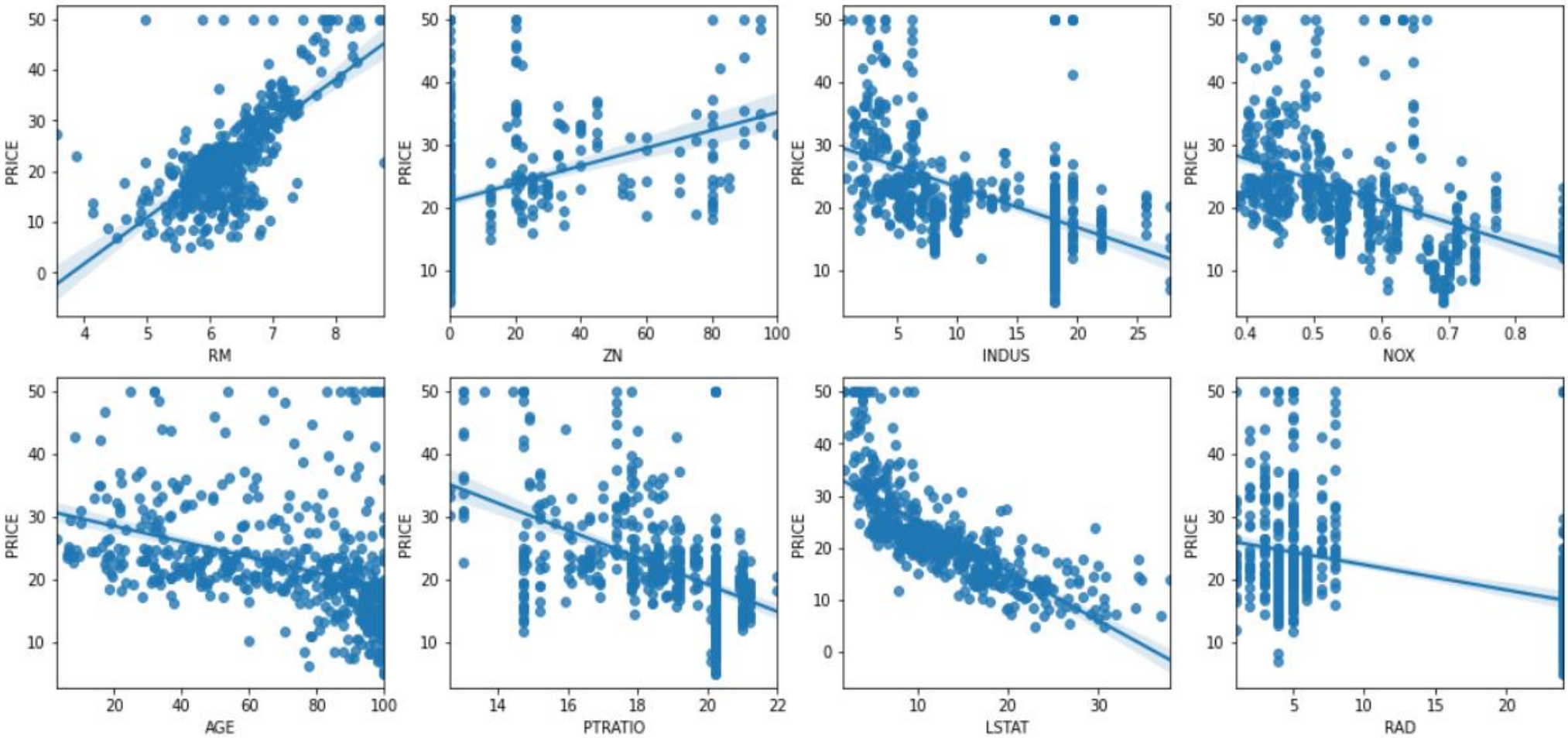
보스톤 데이터세트 크기 : (506, 14)

	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	396.9	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	396.9	9.14	21.6

각 컬럼별로 주택 가격에 미치는 영향 시각화

```
# 2개의 행과 4개의 열을 가진 subplots을 이용, axsms 4x2=8개
fig, axs = plt.subplots(figsize=(18,8),ncols=4, nrows=2)
lm_features = ['RM', 'ZN', 'INDUS', 'NOX', 'AGE', 'PTRATIO', 'LSTAT', 'RAD']
for i, feature in enumerate(lm_features):
    row = i//4
    col = i%4
    # seaborn의 regplot 이용하여 산점도와 선형회귀직선을 함께 표현
    sns.regplot(x=feature, y='PRICE', data=data, ax=axs[row][col])
```

지도학습 : 선형 회귀



- RM(방의 개수) :양방향의 선형성(Positive Linearity)가 가장 큼, RM이 클수록 PRICE가 증가
- LSTAT(하위계층의 비율) : 음방향의 선형성(Negative Linearity)가 가장 큼, LSTAT가 적을수록 PRICE 증가

지도학습 : 선형 회귀

선형회귀

```
# 보스톤 주택 데이터셋 불러오기
from sklearn.datasets import load_boston
boston = load_boston()
```

```
data = boston.data    #data
label = boston.target #label
columns = boston.feature_names
```

```
data = pd.DataFrame(data, columns = columns)
data.head(2)
```

	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	396.9	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	396.9	9.14

데이타셋 나누기 - train, test

```
# 데이타셋 나누기
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(data, label, test_size=0.2, random_state=2020)

# 크기 확인
x_train.shape, x_test.shape
```

```
((404, 13), (102, 13))
```

지도학습 : 선형 회귀

1. 단순선형회귀모델

x(설명변수): RM, y(종속변수) : PRICE

1) 모델 불러오기 및 정의하기

```
from sklearn.linear_model import LinearRegression

sim_lr = LinearRegression()
```

2) 모델 학습하기(훈련 데이타)

```
sim_lr.fit(x_train['RM'].values.reshape((-1, 1)), y_train)
```

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

3) 결과 예측하기(테스트 데이타)

```
y_pred = sim_lr.predict(x_test['RM'].values.reshape((-1,1)))
```

```
# 예측 값
y_pred[:5]
```

```
array([27.95733816, 22.25345529, 18.84570418, 28.1304592 , 23.62020038])
```

```
# 정답
y_test[:5]
```

```
array([27.5, 20.5,  6.3, 24.8, 23.1])
```

지도학습 : 선형 회귀

4) 모델 평가 : R2, RMSE

```
from sklearn.metrics import r2_score

print('보스톤 주택가격 예측 단순선형회귀, R2 : {:.4f}'.format(r2_score(y_test, y_pred)))
```

보스톤 주택가격 예측 단순선형회귀, R2 : 0.3983

```
from sklearn.metrics import mean_squared_error, r2_score

mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)

print('MSE : {0:.3f}, RMSE : {1:.3f}'.format(mse, rmse))
```

MSE : 51.340, RMSE : 7.165

단순 회귀 모델의 계수 w, 절편 b 살펴보기

```
print('단순 선형 회귀, 계수(w) : {:.1f}, 절편(b) : {:.4f}'.format(np.round(sim_lr.coef_[0],1), sim_lr.intercept_))
```

단순 선형 회귀, 계수(w) : 9.1, 절편(b) : -34.4756

단순 선형 회귀식

- PRICE = -34.4756 + 9.1*RM

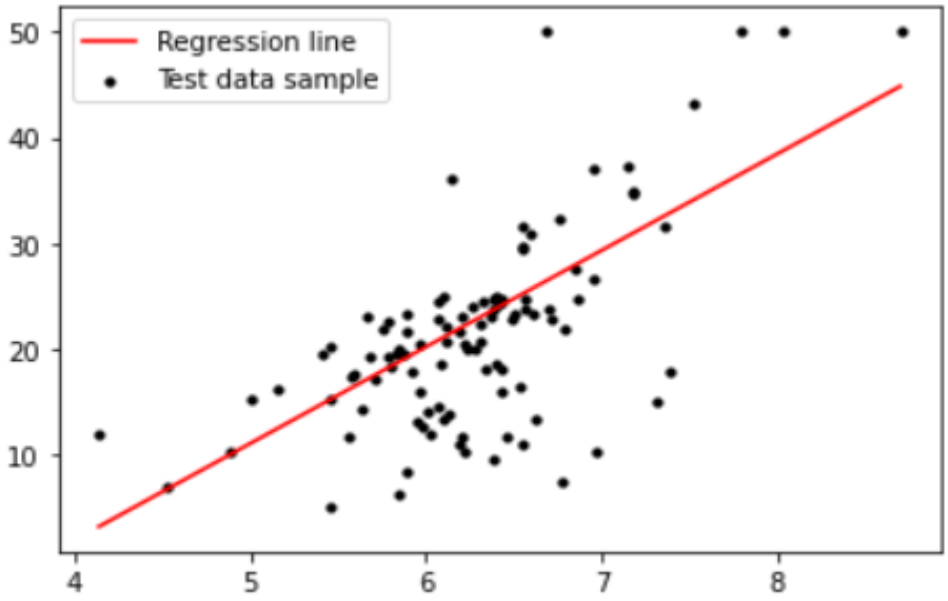
지도학습 : 선형 회귀

단순선형회귀 그래프 그리기

```
line_x = np.linspace(np.min(x_test['RM']), np.max(x_test['RM']), 10)
line_y = sim_lr.predict(line_x.reshape((-1, 1)))

plt.scatter(x_test['RM'], y_test, s=10, c='black')
plt.plot(line_x, line_y, c = 'red')
plt.legend(['Regression line', 'Test data sample'], loc='upper left')

plt.show()
```



지도학습 : 선형 회귀

2. 다중선형회귀(Multiple Linear Regression)

패키지 불러오기

```
# 패키지 불러오기
from sklearn import datasets
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
from sklearn import linear_model
```

보스톤 주택 데이타셋 불러오기

```
# 보스톤 주택 데이타셋 불러오기
from sklearn.datasets import load_boston
boston = load_boston()
```

```
data = boston.data     #data
label = boston.target  #label
columns = boston.feature_names
```

```
data = pd.DataFrame(data, columns = columns)
data.head(2)
```

	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	396.9	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	396.9	9.14

train, test 데이타셋 나누기

```
# 데이타셋 나누기
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(data, label, test_size=0.2, random_state=2020)

# 크기 확인
x_train.shape, x_test.shape
```

```
((404, 13), (102, 13))
```

지도학습 : 선형 회귀

다중선형회귀모델 : x : 설명변수 전체, y : PRICE

1) 모델 불러오기 및 정의하기

```
from sklearn.linear_model import LinearRegression

mul_lr = LinearRegression()
```

2) 모델 학습하기(훈련 데이타)

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

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

3) 결과 예측하기(테스트 데이타)

```
y_pred = mul_lr.predict(x_test)
```

4) 모델 평가 : R2, RMSE

```
from sklearn.metrics import r2_score

print('보스톤 주택가격 예측 다중선형회귀, R2 : {:.4f}'.format(r2_score(y_test, y_pred)))
```

```
보스톤 주택가격 예측 다중선형회귀, R2 : 0.7683
```

```
from sklearn.metrics import mean_squared_error, r2_score

mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)

print('MSE : {0:.3f}, RMSE : {1:.3f}'.format(mse, rmse))
```

```
MSE : 19.774, RMSE : 4.447
```

지도학습 : 선형 회귀

다중 회귀 모델의 계수 w, 절편 b 살펴보기

```
# 회귀 계수와 절편 확인하기 --> 회귀식
print('다중 선형 회귀(LinearRegression), 계수(w) : {}, 절편(b) : {:.4f}'.format(np.round(mul_lr.coef_,1),  mul_lr.intercept_))
```

```
다중 선형 회귀(LinearRegression), 계수(w) : [ -0.1   0.    0.    2.2 -16.9   3.9   0.   -1.4   0.3  -0.   -0.9   0.
  -0.5], 절편(b) : 34.6673
```

```
# 회귀 계수가 큰 값 순으로 정렬학 이해 Series로 생성, 인덱스 컬럼에 유의
coeff = pd.Series(data=np.round(mul_lr.coef_,1), index=data.columns)
coeff.sort_values(ascending=False)
```

```
RM          3.9
CHAS        2.2
RAD         0.3
B           0.0
TAX        -0.0
AGE         0.0
INDUS       0.0
ZN          0.0
CRIM       -0.1
LSTAT      -0.5
PTRATIO    -0.9
DIS        -1.4
NOX       -16.9
dtype: float64
```

최종 다중선형회귀식

- 다중 선형 회귀식(LinearRegression) : PRICE = 34.6673- 3.9*rm* + 2.2CHAS+...

지도학습 : 선형 회귀

K-Fold 검증

```
1  from sklearn.model_selection import cross_val_score
2
3  y_target = boston.target
4  x_data = boston.data
5  lr = LinearRegression()
6
7  # cross_val_score()로 5-폴드 세트로 MSE를 구한 뒤 이를 기반으로 RMSE 구함
8  neg_mse_scores = cross_val_score(lr, x_data, y_target, scoring="neg_mean_squared_error", cv=5)
9  rmse_scores = np.sqrt(-1*neg_mse_scores)
10 arg_rmse = np.mean(rmse_scores)
11
12 # cross_val_score(scoring="neg_mean_squared_error")로 반환된 값은 모두 음수
13 print('5-folds의 개별 Negative MSE scores : ', np.round(neg_mse_scores,2))
14 print('5-folds의 개별 Negative RMSE scores : ', np.round(rmse_scores,2))
15 print('5-folds의 평균 RMSE : ', np.round(arg_rmse))
```

```
5-folds의 개별 Negative MSE scores :  [-12.46 -26.05 -33.07 -80.76 -33.31]
5-folds의 개별 Negative RMSE scores :  [3.53 5.1  5.75 8.99 5.77]
5-folds의 평균 RMSE :  6.0
```

- RMSE가 낮을 수록 좋은 회귀모델