13. 데이터 기반 학습 및 예측 (회귀 분석)

머신 러닝 (기계 학습)

- 문제를 해결을 위해 룰을 (알고리즘) 기술 한 것을 전통 **프로그램** 이라고 함
- 풀이법을 기술하기 (설명하기) 힘들거나, 알지 못한다면 프로그래밍 할 수 없음
- 컴퓨터가 (인공지능이) 스스로 잘하는 방법을 찾아내게 할 수 있을까?
- 인공지능이 데이터를 기반으로 스스로 학습할 수 있다면?
  - 학습 가능한 파라미터에 의해 동작이 결정되는 융통성 있는 프로그램을 **모델** 이라고 부름

지도 학습 (Supervised Learning)

- 학습할 데이터와 학습 목표에 해당하는 레이블 (label) 을 함께 사용
- 데이터를 입력 받아 출력이 정답 레이블과 오차가 줄어들 수 있도록 파라미터를 조정
- 학습이 완료된 모델은 새로운 데이터를 입력 받아 정답 레이블을 예측!

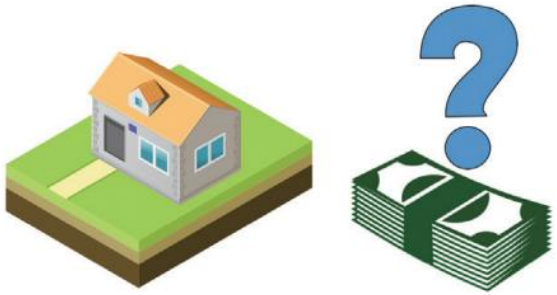
Advertisement	Sales
$90	$1000
$120	$1300
$150	$1800
$100	$1200
$130	$1380
$200	??

회귀 분석 (Regression Analysis)

- 통계학에서 회귀 분석은 독립 변인이 종속 변인에 영향을 주는지 확인하는 분석 방법
- 선형 회귀분석은 독립변수 X (설명변수) 에 대해 종속 변수 Y (반응변수) 사이의 관계를 선형적 (수학적) 함수식으로 규명한 것.
  - 규명된 함수식을 이용하면 독립변수 변화로부터 종속변수 변화를 예측할 수 있음

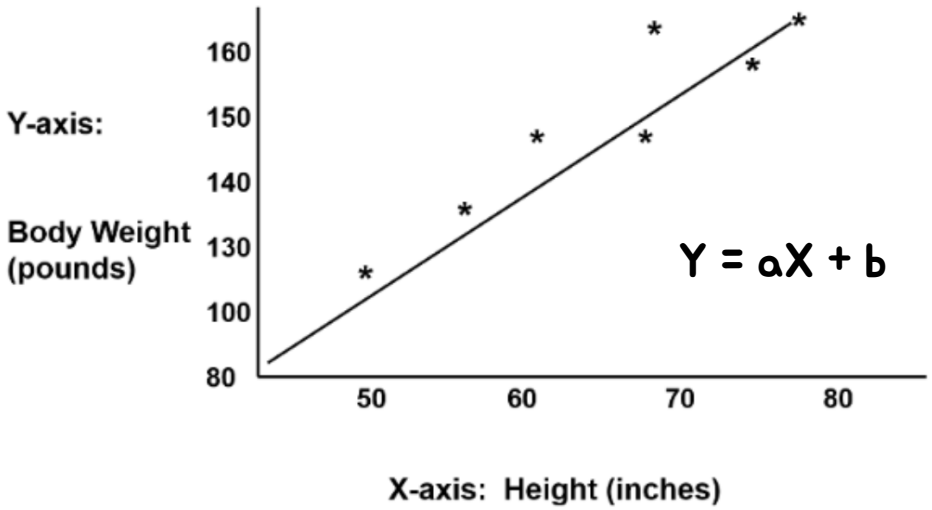
예시: 주택 면적 (독립변수) → 주택 가격 (종속 변수)

주택 면적, 일조량, 지하철 거리 (독립변수들) → 주택 가격 (종속 변수)



선형 회귀 분석 예시

- 키에 따라 몸무게가 결정된다고 하였을 때, 키 는 독립 변수, 몸무게는 종속 변수임
- 아래와 같은 선형 적인 수학적 (함수) 모델을 설계할 수 있음
  - 몸무게 = a x 키 + b (a 와 b 는 학습 가능한 파라미터)
- 선형 회귀 모델 결과인 직선과 실제 값 사이의 오차가 최소화되는 모델을 찾는 것!
  - 데이터 분포를 가장 잘 설명한다고 할 수 있음



선형 회귀 모델의 계수와 절편

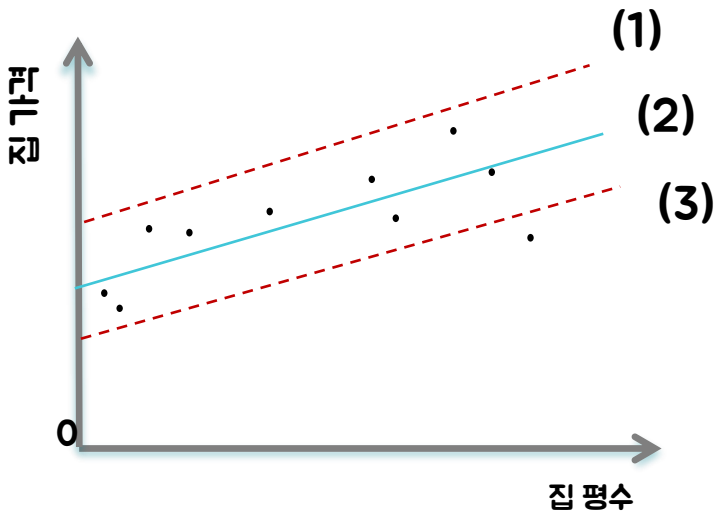
- 선형 회귀는 임의의 독립 변수 X 와 이 변수에 따른 종속 변수 Y 와의 상관관계를 모델링
- 이들 간의 관계를 학습 후 새로운 X 값에 대해 Y 를 예측하는 데 사용
- 간단한 직선의 방정식은 다음과 같음

$$Y = aX + b$$

- 여기서 a 는 직선의 기울기이고 입력 변수 X 에 곱해지는 **계수(coefficient)**
- X 와 관계없이 Y 에 영향을 주는 값 b는 **절편(intercept)**
- (절편은 $Y = aX$ 라는 회귀선을 위 또는 아래로 얼마나 평행이동 시킬지를 결정)

선형 회귀 모델의 계수와 절편

- 선형 회귀 알고리즘은 데이터를 설명하는 가장 적절한 기울기와 절편 값을 찾는 것
- 기울기와 절편의 값에 따라 다양한 직선이 존재할 수 있음
- 주어진 데이터에 가장 적은 오류를 발생시키는 직선을 찾는 것이 목표
  - 그림에서 (2) 번 직선이 가장 적은 오류를 발생시킴



선형 회귀 모델의 계수와 절편

- 2개 이상의 변수가 있는 경우로 확장 가능하며 이를 다중 회귀분석이라고 함
- 주택의 평수, 침실 수, 화장실 수, 지하철 역과의 거리 등이 주어졌을 때,
- (주택 가격) Y 는 여러 독립 변수에 종속 됨
- K 개의 독립 변수가 포함된 다중 회귀 모델은 다음과 같이 나타냄

$$Y = F(X; W) = w_0 + w_1 X_1 + w_2 X_2 + \cdots + w_k X_k$$

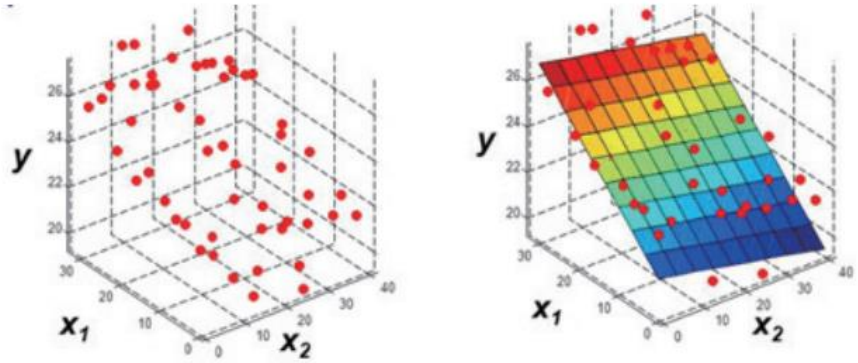
- X 와 W 는 모두 벡터 (1차원 배열) $where\ W = w_0 + w_1 + w_2 + \cdots + w_k$

선형 회귀 모델의 계수와 절편

- 2차원 공간에서 선형 회귀 모형은 직선이며,

  3차원 공간에서는 평면이고,

  3차원 이상 공간 에서는 **초평면 (hyperplane)**



- 이러한 복잡한 다중 선형 회귀 모델도 '사이킷런 (scikit-learn)' 라이브러리를 사용하면

  손쉽게 학습 시킬 수 있음!

- 설치방법: pip install sklearn

선형 회귀 분석 예시

- 사이킷런 라이브러리 sklearn 에서 LinearRegression (선형회귀분석) 모듈 선택
- 학습(입력) 데이터 X 와 목표(정답) 데이터 Y 가 아래와 같이 주어졌을 때,

  fit( ) 함수를 통해 선형 회귀 모델을 학습
  - 학습 데이터는 반드시 2차원 배열 이어야 함 (추후 다중 선형 회귀 모델에서도 동일)

```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()

X = [ [163], [179], [166], [169], [171] ]
Y = [ 54, 63, 57, 56, 58 ]

lr.fit(X, Y)
```

선형 회귀 분석 예시

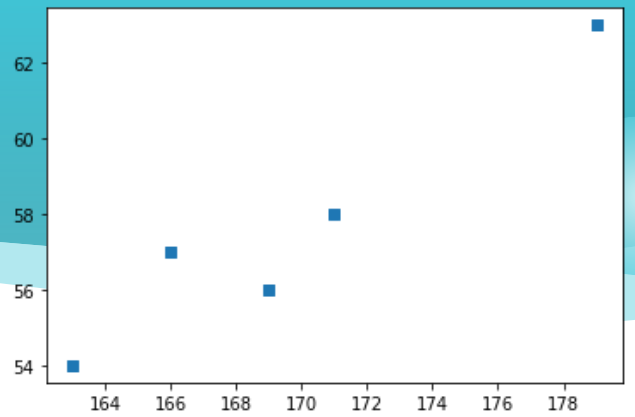
- predict() 메소드를 통해 학습된 모델을 사용하여 입력 데이터에 대한 출력 (예측) 값을 획득

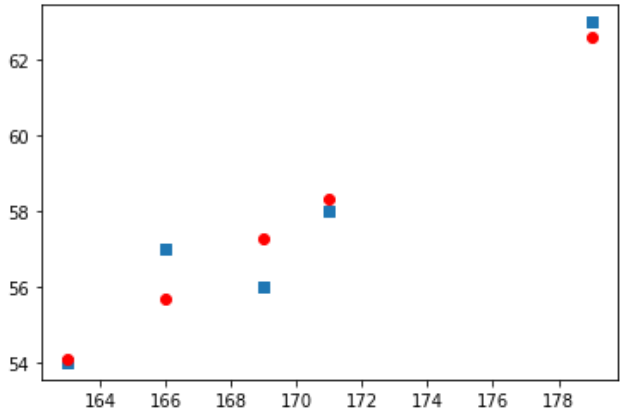
```
plt.scatter(X, Y, marker='s')
plt.show()
```

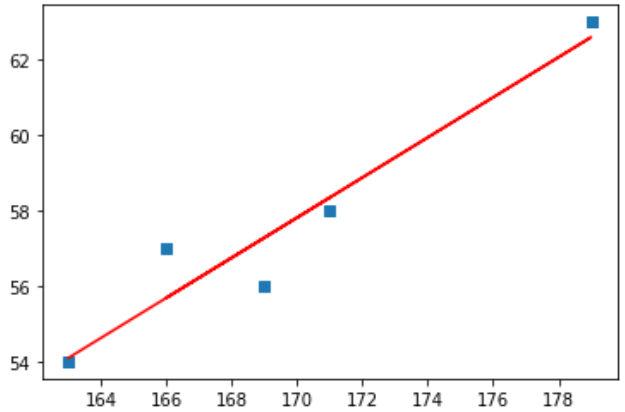
```
plt.scatter(X, Y, marker='s')
y_pred = lr.predict(X)
plt.scatter(X, y_pred, color='red')
plt.show()
```

```
plt.scatter(X, Y, marker='s')
y_pred = lr.predict(X)
plt.plot(X, y_pred, 'r')
plt.show()
```

입력 X 값과 예측한 Y 값을 시각화







선형 회귀 분석 예시

- 학습된 선형 회귀 모델이 얼마나 데이터를 잘 설명할 수 있는지 확인
  - score() 메소드는 모델의 점수 (0.0 ~ 1.0) : 얼마나 데이터를 정확히 설명하는가?
  - coef_ 속성은 학습된 선형 직선의 기울기를 나타냄
    - 기울기 0.53 이 의미하는 것은 ? X가 1 증가할 때 Y 가 0.53 증가함.

      예) 키가 1 cm 증가할 때 몸무게는 0.53 kg 증가

```
lr.score(X, Y)
print( lr.coef_, lr.intercept_ )
```

```
0.9191095132743363
[0.53125] -32.500000000000036
```

다중 선형 회귀 분석 예시

- 키와 성별을 독립변수로 입력 받는 다중 선형 회귀 모델
- 남학생은 0, 여학생은 1로 구분
- 입력 데이터는 동일하게 2차원을 구성 (예: [[키, 성별], ...])

```
X = [ [171,0], [169,0], [176,0], [168,0], [181,0], [166,0], [180,0], [175,0],
      [163,1], [162,1], [171,1], [162,1], [164,1], [162,1], [158,1], [173,1] ]

Y = [69, 65, 72, 67, 71, 65, 80, 71, 55, 51, 59, 53, 61, 56, 47, 57]
lr.fit(X, Y)
print( lr.score(X, Y) )
print( lr.coef_ )
```

```
0.8906355513939725
[ 0.66027053 -9.26509909]
```

다중 선형 회귀 분석 예시

- 학습된 모델의 계수 (기울기) 의미 :
  - 키가 1 cm 증가 시 체중 0.66 kg 증가
  - 성별에 따라 체중 9.2 kg 차이

```
print( lr.coef_ )
```

```
[ 0.66027053 -9.26509909]
```

- 168cm 남성과 168cm 여성의 체중을 예측해보자

```
test_X = [ [168, 0], [168, 1] ]
lr.predict( test_X )
```

```
[66.53357974 57.26848065]
```

도전 과제 : Boston 집 값 예측하기

- Boston housing 데이터셋
  - 1978년 보스턴 시의 506개 타운의 주택 정보 데이터
  - 총13개의 독립변수와 1개의 종속 변수 (주택 가격 중앙값) 로 구성

crim	범죄율
zn	주택지 비율
indus	비소매상업지역(농지) 토지 비율
chas	찰스 강변 이면 1, 아니면 0
nox	10ppm 당 일산화질소 농도
rm	주택 1가구당 평균 방의 수
age	1940년 이전에 건설된 비율
dis	5개 보스톤 직업센터와의 접근성 지수
rad	순환 고속 도로 접근 용이성
tax	재산세율
ptratio	학생/교사 비율
b	흑인의 비율
lstat	하위 계층의 비율
mid_price	주택가격 중앙 값(단위 $1000)

도전 과제 : Boston 집 값 예측하기

- 목표 :
  - Boston housing 데이터셋을 학습 데이터셋과 평가 데이터셋으로 분리하기
  - 학습용 데이터셋으로 학습 후 및 평가용 데이터셋을 사용하여 검증하기

    (다중 선형 회귀 모델 사용하여 주택 가격 중앙 값을 예측하기)

- sklearn.datasets 에서 데이터셋 로드

```
from sklearn.datasets import load_boston
boston = load_boston()
print( boston.keys() )
print( type(boston.data) )
print( type(boston.target) )
```

```
dict_keys(['data', 'target', 'feature_names',
'DESCR', 'filename', 'data_module'])

<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
```

도전 과제 : 데이터 확인

- 데이터프레임 타입 데이터로 생성
  - 데이터 컬럼 별 개수, 타입 확인
  - 중복 데이터, 결측 데이터 여부 확인

```
dataset = pd.DataFrame(boston.data, columns=boston.feature_names)
dataset['mid_price'] = boston.target
print(dataset)
print( dataset.info() )
print( dataset.duplicated().sum() )
print( dataset.isna().sum() )
```

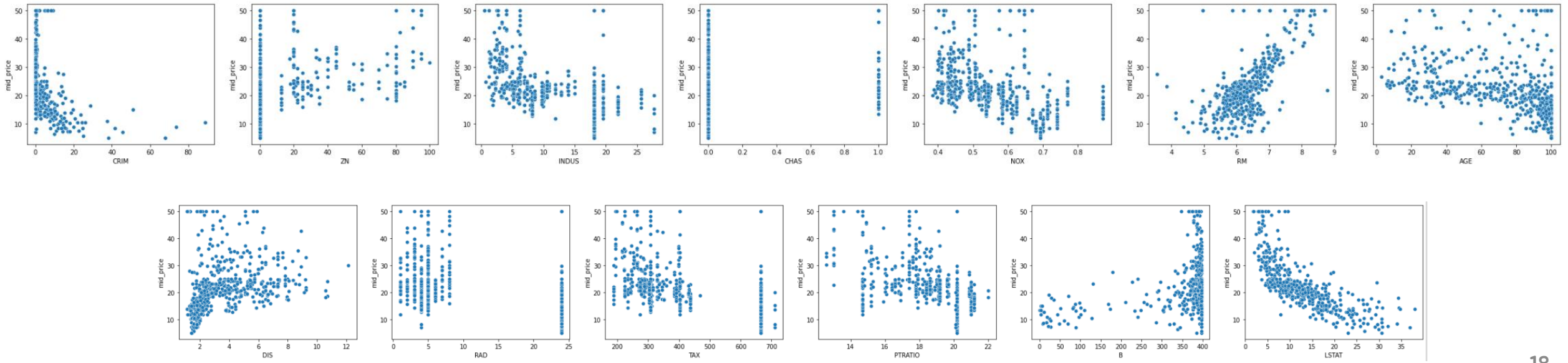
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   CRIM        506 non-null    float64
 1   ZN          506 non-null    float64
 2   INDUS       506 non-null    float64
 3   CHAS        506 non-null    float64
 4   NOX         506 non-null    float64
 5   RM          506 non-null    float64
 6   AGE         506 non-null    float64
 7   DIS         506 non-null    float64
 8   RAD         506 non-null    float64
 9   TAX         506 non-null    float64
 10  PTRATIO     506 non-null    float64
 11  B           506 non-null    float64
 12  LSTAT       506 non-null    float64
 13  mid_price   506 non-null    float64
```

```
0
CRIM          0
ZN            0
INDUS         0
CHAS          0
NOX           0
RM            0
AGE           0
DIS           0
RAD           0
TAX           0
PTRATIO       0
B             0
LSTAT         0
mid_price     0
```

도전 과제 : 시각화

- 시각화를 통한 데이터 탐색

```
plt.figure ( figsize = (80, 4))
i=1
for c in dataset.columns:
  if c != 'mid_price':
    plt.subplot(1, len(dataset.columns)-1, i )
    i = i+1
    sns.scatterplot(x=c, y='mid_price', data=dataset)
plt.show()
```

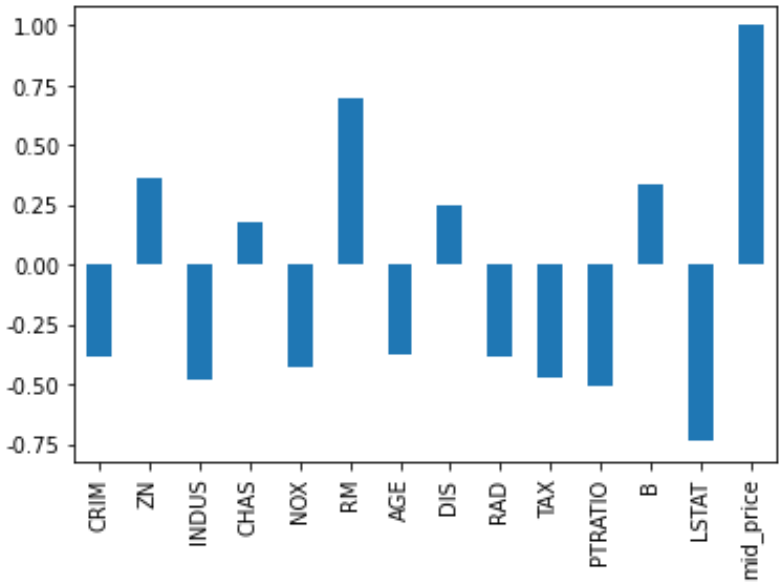


도전 과제 : 상관계수 확인

- 전체 상관계수 계산 및 mid_price 와의 상관계수 값 확인

```
res = dataset.corr()
print(res.loc['mid_price'])
print()
res.loc['mid_price'].plot.bar()
plt.show()
```

```
CRIM        -0.388305
ZN           0.360445
INDUS       -0.483725
CHAS         0.175260
NOX         -0.427321
RM           0.695360
AGE         -0.376955
DIS          0.249929
RAD         -0.381626
TAX         -0.468536
PTRATIO     -0.507787
B            0.333461
LSTAT       -0.737663
mid_price    1.000000
Name: mid_price, dtype: float64
```



도전 과제 : 학습용 & 평가용 데이터셋 분리

- 데이터셋 일부를 학습 된 모델의 성능 검증을 위해 평가용으로 분리
  - 학습용 & 평가용 데이터셋에서
    모델 입력으로는 정답(종속)변수에
    해당하는 'mid_price' 는 제외

```
# 데이터셋 약 20% 를 평가용으로 사용
test_set = dataset.iloc[ : 100 ]
train_set = dataset.iloc[ 100 :  ]

# 입력 데이터에서는 'mid_price' 열을 제외
# 정답 (레이블) 데이터로는 'mid_price' 열만 사용
train_X = train_set.drop( ['mid_price'], axis=1)
train_Y = train_set['mid_price']

test_X = test_set.drop( ['mid_price'], axis=1)
test_Y = test_set['mid_price']
```

도전 과제 : 다중 선형 회귀 모델 학습

- 사이킷런 라이브러리 LenearRegression 를 사용하여 학습용 데이터셋을 사용하여 다중선형 회귀 모델을 학습 시킴

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit( train_X, train_Y )
```

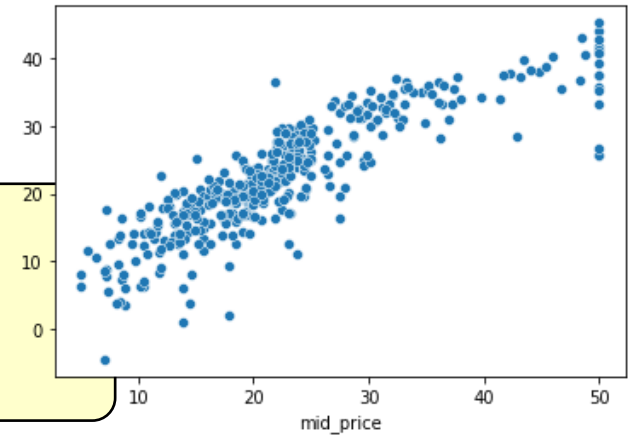
- 학습된 모델 점수를 확인

```
print( lr.score( train_X, train_Y ) )
```

```
0.7465346879380911
```

- 학습된 모델로 예측한 주택 가격 중앙 값을 시각화

```
pred = lr.predict( train_X )
sns.scatterplot( x=train_Y, y=pred )
plt.show()
```



도전 과제 : 학습 된 모델 평가

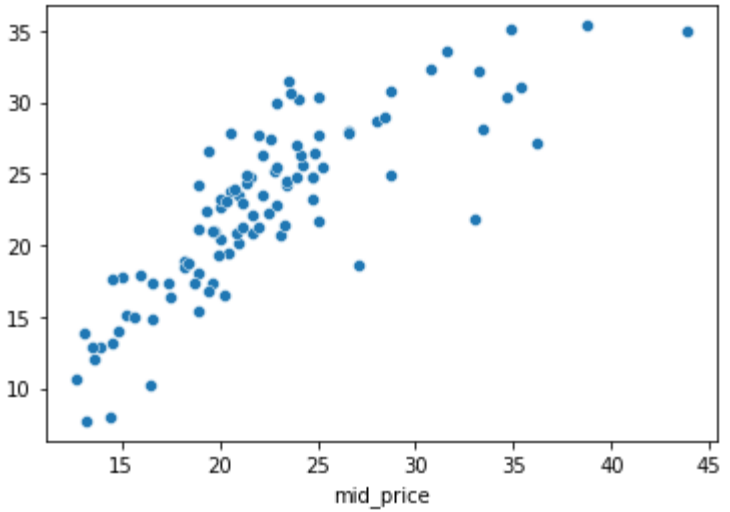
- 평가용 데이터셋으로 학습된 모델 성능을 검증

```
print( lr.score( test_X, test_Y ) )
```

```
0.6377771855934954
```

- 평가용 입력 데이터에 대한 예측 결과와 정답을 비교

```
pred = lr.predict( test_X )
sns.scatterplot( x=test_Y, y=pred)
plt.show()
```



도전 과제 : 학습 된 모델 해석해보기

- 학습된 다중 선형 회귀모델의 계수 (기울기) 값을 확인

```
for i in range( len(train_X.columns) ) :
  print( train_X.columns[i], ":\t",  lr.coef_[i] * 1000 )
```

```
CRIM :    -109.12483869914173
ZN :      58.89962982722602
INDUS :   53.21431135979127
CHAS :    2531.364351313172
NOX :     -21159.335829429976
RM :      3282.285403572024
AGE :     -4.043961150022982
DIS :     -1783.2976866029417
RAD :     318.7746940849711
TAX :     -12.518088641898517
PTRATIO :         -1011.8621860050716
B :       9.287793522139841
LSTAT :   -581.7562680136233
```

```
#print( type(lr.coef_) ) # <class 'numpy.ndarray'>
coef = pd.Series(data=lr.coef_, index=train_data.columns) * 1000
print( coef.sort_values(ascending=False) )
```

```
RM          3282.285404
CHAS        2531.364351
RAD          318.774694
ZN            58.899630
INDUS         53.214311
B              9.287794
AGE           -4.043961
TAX          -12.518089
CRIM        -109.124839
LSTAT       -581.756268
PTRATIO    -1011.862186
DIS        -1783.297687
NOX       -21159.335829
dtype: float64
```

- RM (방 개수가) 1 증가할 때 주택 가격은 $3,282 증가함
- NOX (질소 수치가) 0.1 증가할 때 주택 가격은 $2,116 낮아짐

도전 과제 : 모델 튜닝 해보기

- 독립 변수 선택을 다르게 적용해보기

```
dataset2 = dataset.drop( ['ZN',  'INDUS', 'B', 'AGE',  'TAX'], axis=1 )
train_set = dataset2.iloc[ 100 :  ]
test_set = dataset2.iloc[ : 100 ]

train_X = train_set.drop( ['mid_price'], axis=1)
train_Y = train_set['mid_price']
test_X = test_set.drop( ['mid_price'], axis=1)
test_Y = test_set['mid_price']
```

```
lr.fit(train_X, train_Y)
print( lr.score(train_X, train_Y) )
```

```
0.7276877334642399
```

```
lr.score(test_X, test_Y)
```

```
0.6429009228353522
```

맺음말

- 주어진 데이터를 이해하고, 분석하기 위해 선형회귀 모델을 배워 봄
- 학습된 모델을 사용하여 새로운 데이터를 입력 받아 종속 변수 값을 예측
- 선형 회귀 모델의 계수 (기울기) 를 사용하면 각 독립변수의 영향을 이해하는데 도움
- 학습에 사용할 변수 선정에 따라 모델 성능이 달라짐
- 본 과목에서는 간단한 선형회귀 모델만 소개하였으며 보다 깊게 배우고 싶은 학생은

  "기계학습 (Machine Learning)" 과목 수강을 추천!