회귀 분석

충북대학교 소프트웨어학과
류관희

목 차

01

개요

- 회귀분석
- 회귀모델
- 선형회귀 모델

02

선형회귀 모델 적용

- 주택 가격 예측
- 선형회귀 모델의 평가 방법
- 당뇨병 진행도 예측

03

선형회귀 모델

- Ridge 모델 이용 주택가격예측
- Lasso 모델 이용 주택가격예측
- 선형모델 이용 주식가격예측

학습목표

충북대학교
SW중심대학사업단

이번 파트에서는 회귀 분석을 위해 요구되는 다음과 같은 내용을 공부한다.

- 회귀분석
- 회귀모델
- 선형회귀 모델

회귀분석

- 회귀분석은 데이터 요소간의 복잡한 관계를 모델링하기 위해 사용됨
  - 모집단과 개별 항목이 측정된 특성에 의해 어떻게 변화되는지 보여줌
    - 여러영역 응용-economics, sociology, psychology, physics, and ecology.
  - 사건과 결과의 원인관계를 규명
    - clinical drug trials, engineering safety tests, or marketing research.
  - 미래 행위 예측을 위해 사용되는 패턴을 식별
    - predicting insurance claims, natural disaster damage, election results, and crime rates.

회귀 모델

- 회귀 모델 함수 $\hat{y} = w[0] \times x[0] + w[1] \times x[1] + \cdots + w[p] \times x[p] + b$

  x[0] ~ x[p] : 하나의 데이터에 대한 독립 변수

  w, b : 모델이 학습할 파라미터

  $\hat{y}$: 종속 변수, 예측 값

- 회귀 분석의 여러 가지 기법

  1. 최소제곱법(OLS) – Linear Regression
  2. Ridge
  3. Lasso
  4. ElasticNet(리지와 라소의 결합)
  5. 다항식

선형회귀 모델

- 선형 회귀식

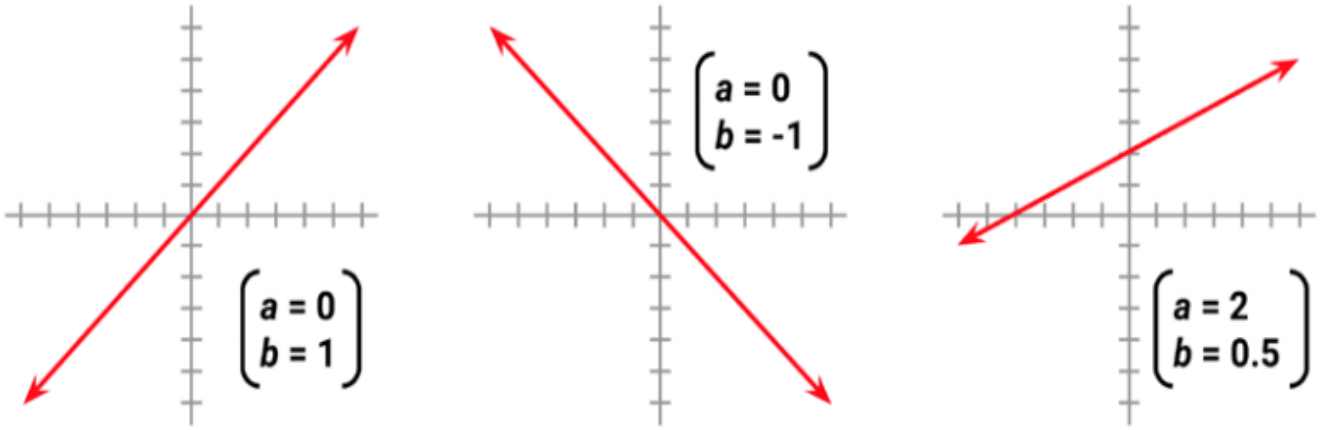
  $y = a + bx.$

  - y : 종속변수 dependent variable
  - x : 독립변수 independent variable

- 기울기 slope (b):
  - 양의 기울기
  - 음의 기울기

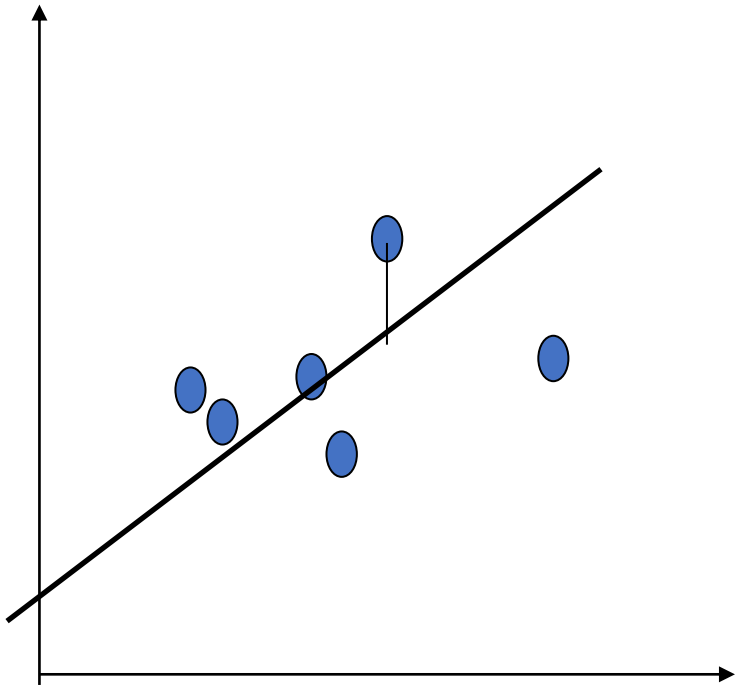
- 절편 (a) intercept

선형회귀 모델



선형 회귀 모델:OLS

- OLS(Ordinary least squares)
- Y= a+b x
  - 제곱 에러 최소화하는 a,b를 구하자
    - 에러: 예측 y 값과 실제 y 값과의 차이
  - 잔차(residual): 에러의 합



선형 회귀 모델:OLS

- 가장 간단하고 오래된 회귀 선형 알고리즘

- 예측과 훈련 세트에 있는 타깃 y사이의 평균제곱오차를 최소화 하는 w와 b를 찾는 방법

- 평균제곱오차 : 예측값과 타깃값의 차를 제곱하여 더한 후 샘플 개수로 나눈 것

- 장점 : 매개변수가 없음
  단점 : 매개변수가 없어서 모델의 복잡도 제어 불가

- 기울기 w(=가중치, 계수) : lr 객체의 coef_에 저장
  절편 b(=편향) : lr 객체의 intercept_에 저장

선형 회귀 모델 예제

$$J(\Theta) = \frac{1}{n}\sum_{i=1}^{n}(f_\Theta(\mathbf{x}_i) - y_i)^2$$

```
import numpy as np

x_train = np.array([1., 2., 3., 4., 5., 6.])
y_train = np.array([9., 12., 15., 18., 21., 24.])
```

```
n_data = len(x_train)  # 6
epochs = 5000
learning_rate = 0.01
W = 0.0
b = 0.0
```

선형 회귀 모델 예제


```
for i in range(epochs):
    hypothesis = x_train * W + b
    cost = np.sum((hypothesis - y_train) ** 2) / n_data
    gradient_w = np.sum((W * x_train - y_train + b) * 2 * x_train) / n_data
    gradient_b = np.sum((W * x_train - y_train + b) * 2) / n_data

    W -= learning_rate * gradient_w
    b -= learning_rate * gradient_b

    if i % 100 == 0:
        print('Epoch ({:10d}/{:10d}) cost: {:10f}, W: {:10f}, b:{:10f}'.format(i, epochs, cost, W, b))
```

```
print('W: {:10f}'.format(W))
print('b: {:10f}'.format(b))
print('result : ')
print(x_train * W + b)
```

```
W:   3.000000
b:   6.000000
result :
[ 8.99999996 11.99999997 14.99999998 18.         21.00000001 24.00000002]
```

문제풀이

- 회귀 모델이란 무엇인가?
- 선형회귀 모델을 설명하시오.

요약

- 회귀 모델을 공부하였음
- 선형회귀 모델을 공부하였음

01

개요

- 회귀분석
- 회귀모델
- 선형회귀 모델

02

선형회귀 모델 적용

- 주택 가격 예측
- 선형회귀 모델의 평가 방법
- 당뇨병 진행도 예측

03

선형회귀 모델

- Ridge 모델 이용 주택가격예측
- Lasso 모델 이용 주택가격예측
- 선형모델 이용 주식가격예측

학습목표

이번 파트에서는 회귀 분석을 위해 요구되는 다음과 같은 내용을 공부한다.

- 주택 가격 예측
- 선형회귀 모델의 평가 방법
- 당뇨병 진행도 예측

회귀모델응용-보스턴주택가격예측

- 보스턴 주택 가격 (1978발표): 주택가격에 영향을 미치는 요소

CRIM	자치시(town) 별 1인당 범죄율
ZN	25,000 평방피트를 초과하는 거주지역의 비율
INDUS	비소매상업지역이 점유하고 있는 토지의 비율
CHAS	찰스강에 대한 더미변수(강의 경계에 위치한 경우는 1, 아니면 0)
NOX	10ppm 당 농축 일산화질소
RM	주택 1가구당 평균 방의 개수
AGE	1940년 이전에 건축된 소유주택의 비율
DIS	5개의 보스턴 직업센터까지의 접근성 지수
RAD	방사형 도로까지의 접근성 지수
TAX	10,000 달러 당 재산세율
PTRATIO	자치시(town)별 학생/교사 비율
B	1000(Bk-0.63)^2, 여기서 Bk는 자치시별 흑인의 비율을 말함.
LSTAT	모집단의 하위계층의 비율(%)

Price 집 값 ← 종속변수

회귀모델응용-보스턴주택가격예측


```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

plt.style.use('seaborn')
sns.set(font_scale=2.5) # 이 두줄은 본 필자가 항상 쓰는 방법입니다.
import missingno as msno

#ignore warnings
import warnings
warnings.filterwarnings('ignore')

%matplotlib inline
```

```
import matplotlib as mpl
import matplotlib.font_manager as fm
# 그래프에서 마이너스 폰트 깨지는 문제에 대한 대처
mpl.rcParams['axes.unicode_minus'] = False
#plt.rcParams["font.family"] = 'Nanum Brush Script OTF'
#plt.rcParams["font.size"] = 20
plt.rcParams["figure.figsize"] = (20,10)

plt.rc('font', family='NanumGothic') # For Windows
plt.rcParams['font.family'] = 'Malgun Gothic'
plt.rcParams.update({'font.size': 15})
print(plt.rcParams['font.family'])
```

회귀모델응용-보스턴주택가격예측


```python
from sklearn.datasets import load_boston #scikit-learn의 datasets에서 sample data import

boston = load_boston() # boston dataset load
```

```python
df = pd.DataFrame(data=boston.data, columns=boston.feature_names)
df['price'] = boston.target
print(df)
```

```
        CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD    TAX  #
0    0.00632  18.0   2.31   0.0  0.538  6.575  65.2  4.0900  1.0  296.0
1    0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671  2.0  242.0
2    0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671  2.0  242.0
3    0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622  3.0  222.0
4    0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622  3.0  222.0
..       ...   ...    ...   ...    ...    ...   ...     ...  ...    ...
501  0.06263   0.0  11.93   0.0  0.573  6.593  69.1  2.4786  1.0  273.0
502  0.04527   0.0  11.93   0.0  0.573  6.120  76.7  2.2875  1.0  273.0
503  0.06076   0.0  11.93   0.0  0.573  6.976  91.0  2.1675  1.0  273.0
504  0.10959   0.0  11.93   0.0  0.573  6.794  89.3  2.3889  1.0  273.0
505  0.04741   0.0  11.93   0.0  0.573  6.030  80.8  2.5050  1.0  273.0

     PTRATIO       B  LSTAT  price
0       15.3  396.90   4.98   24.0
1       17.8  396.90   9.14   21.6
2       17.8  392.83   4.03   34.7
3       18.7  394.63   2.94   33.4
4       18.7  396.90   5.33   36.2
..       ...     ...    ...    ...
501     21.0  391.99   9.67   22.4
502     21.0  396.90   9.08   20.6
503     21.0  396.90   5.64   23.9
504     21.0  393.45   6.48   22.0
505     21.0  396.90   7.88   11.9

[506 rows x 14 columns]
```

보스턴 집값의 분포

```
1 print(df.isnull().sum())
```

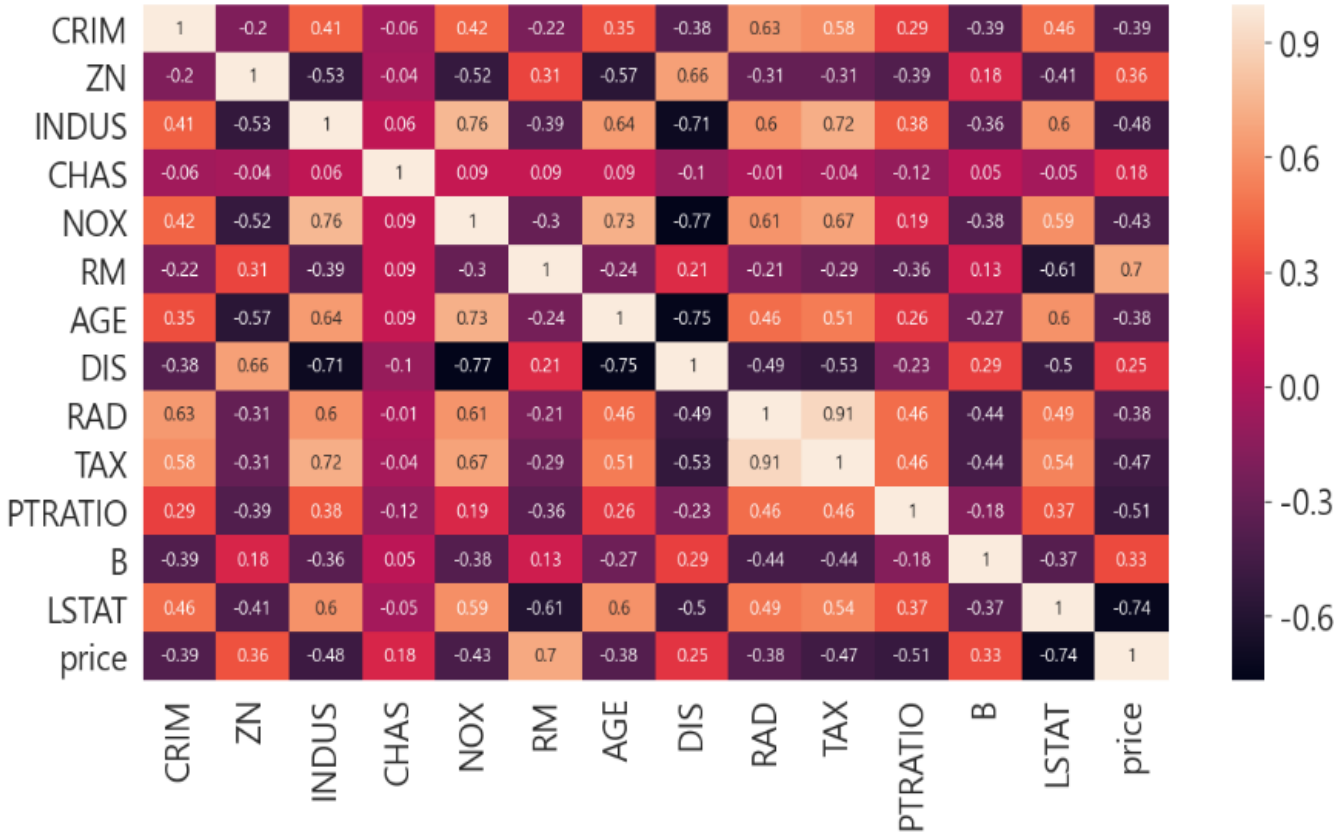
```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
price      0
dtype: int64
```

```
1 bhp_data.iloc[:,-1].value_counts()
```

```
50.0    16
25.0     8
23.1     7
21.7     7
22.0     7
        ..
12.8     1
29.9     1
9.6      1
36.1     1
13.0     1
Name: MEDV, Length: 229, dtype: int64
```

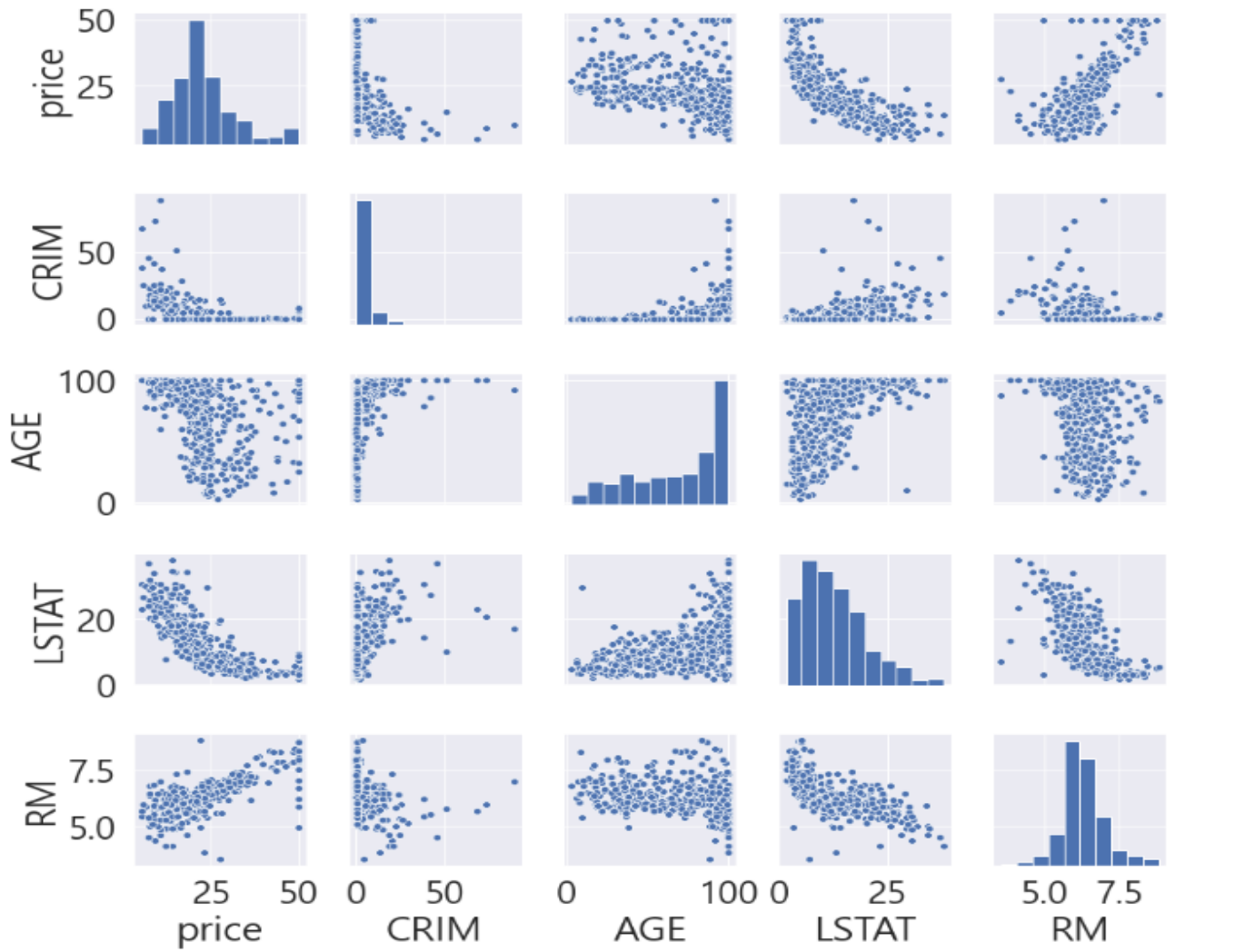
보스턴 집값의 분포

```
correlation_matrix = df.corr().round(2)
sns.heatmap(data=correlation_matrix, annot=True)
plt.show()
```



보스턴 집값의 분포

```
1 cols = ["price","CRIM", "AGE", "LSTAT", "RM"]
2 sns.pairplot(df[cols])
3 plt.show()
```



선형 회귀 모델의 평가 방법

- MAE(Mean Absolute Error) : 실제값과 예측값의 차이를 절댓값으로 변환해 평균한 수치

$$MAE = \frac{\sum |y - \hat{y}|}{n}$$

- MSE(Mean Sqaured Error) : 실제값과 예측값의 차이를 제곱하여 평균한 수치

$$MSE = \frac{\sum (y - \hat{y})^2}{n}$$

mean_squared_error(y_train, y_train_pred)

- RMSE(Root Mean Squared Error) : MSE에 루트(제곱근)을 씌운 수치. 실제 오류 평균보다 더 커지는 특성이 있으므로 이를 방지해준다.

$$RMSE = \sqrt{\frac{\sum (y - \hat{y})^2}{n}}$$

선형 회귀 모델의 평가 방법

MAPE(Mean Absolute Percentage Error)

$$MAPE = \frac{\sum \left| \frac{y - \hat{y}}{y} \right|}{n} * 100\%$$

- RMSLE(Root Mean Sqaured Log Error) : RMSE에 Log(로그를) 취한 수치. 결정값이 클수록 오류값도 커지기 때문에 일부 큰 오류값들로 인해 전체 오류값이 커지는 것을 방지해준다.

$$RMSLE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(log(\hat{y}_i + 1) - log(y_i + 1))^2}$$

- R2 Score(결정계수) : 분산 기반으로 예측 성능을 평가한다. 실제 값의 분산 대비 예측값의 분산 비율을 지표로 하며, 1에 가까울수록 예측 정확도가 높다.

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

$\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$ $\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \mu_y\right)^2$ **r2_square(y_train, y_train_pred)**

선형모델 이용 주택가격예측


```python
import numpy as np
import pandas as pd
from sklearn import datasets
#
from sklearn import model_selection
from sklearn.linear_model import LinearRegression
from sklearn import metrics

from sklearn import datasets
dataset = datasets.fetch_california_housing()
x_data = dataset.data
y_data = dataset.target

x_train, x_test, y_train, y_test = model_selection.train_test_split(x_data, y_data, test_size=0.3)

estimator = LinearRegression()

estimator.fit(x_train, y_train)

y_predict = estimator.predict(x_train)
score = metrics.r2_score(y_train, y_predict)
print(score) #1.0

y_predict = estimator.predict(x_test)
score = metrics.r2_score(y_test, y_predict)
print(score) #1.0
```

```
0.5996813750167989
0.6187842970105374
```

선형회귀모델-당뇨병 진행도예측

당뇨병 진행도 예측: 442명 당뇨병 환자 대상으로 한 검사 결과

- age: 나이
- sex: 성별
- bmi: BMI(Body mass index)지수
- bp: 평균혈압
- s1~s6: 6종류의 혈액검사수치

```
1 from sklearn.datasets import load_diabetes
2
3 diabetes = load_diabetes()
4 df = pd.DataFrame(diabetes.data, columns=diabetes.feature_names)
5 df["target"] = diabetes.target
6 df
```

	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6	target
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.019908	-0.017646	151.0
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.068330	-0.092204	75.0
2	0.085299	0.050680	0.044451	-0.005671	-0.045599	-0.034194	-0.032356	-0.002592	0.002864	-0.025930	141.0
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.022692	-0.009362	206.0
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.031991	-0.046641	135.0
...	...	...	...	...	...	...	...	...	...	...	...
437	0.041708	0.050680	0.019662	0.059744	-0.005697	-0.002566	-0.028674	-0.002592	0.031193	0.007207	178.0
438	-0.005515	0.050680	-0.015906	-0.067642	0.049341	0.079165	-0.028674	0.034309	-0.018118	0.044485	104.0
439	0.041708	0.050680	-0.015906	0.017282	-0.037344	-0.013840	-0.024993	-0.011080	-0.046879	0.015491	132.0
440	-0.045472	-0.044642	0.039062	0.001215	0.016318	0.015283	-0.028674	0.026560	0.044528	-0.025930	220.0
441	-0.045472	-0.044642	-0.073030	-0.081414	0.083740	0.027809	0.173816	-0.039493	-0.004220	0.003064	57.0

442 rows × 11 columns

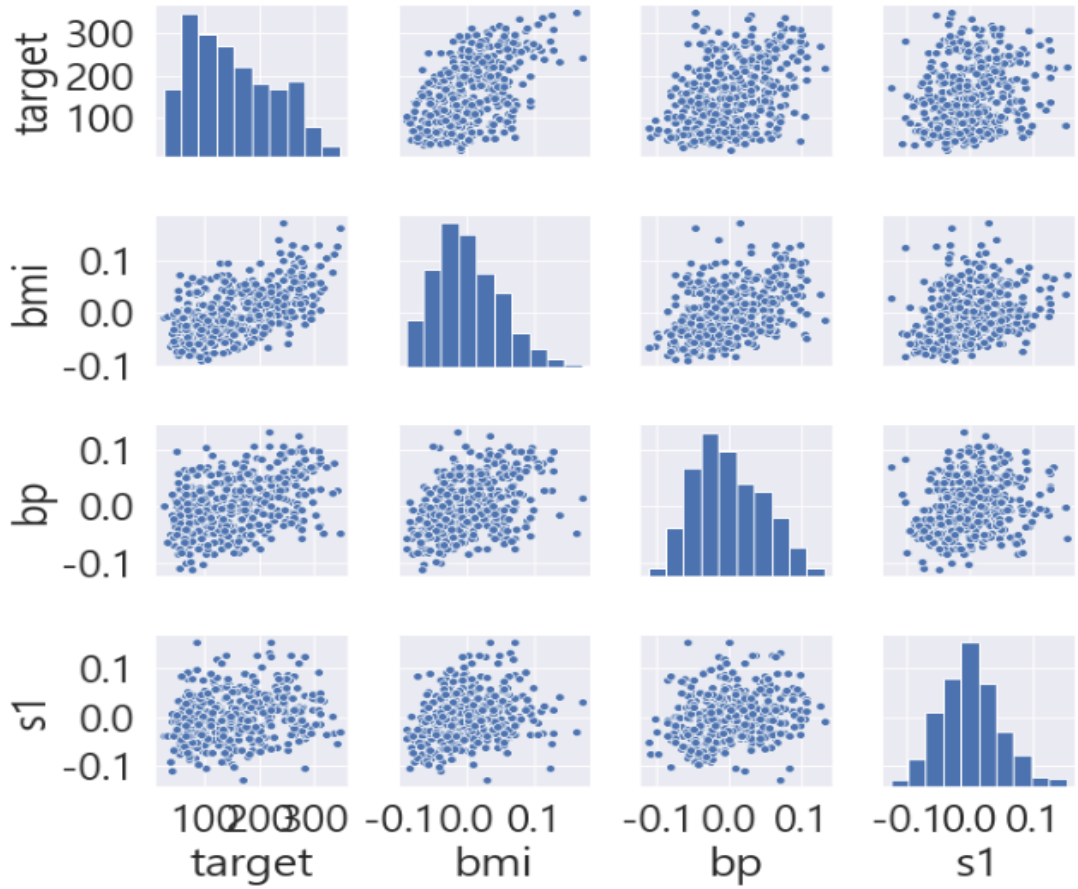
선형회귀모델-상관관계


```
1 df.corr()
```

	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6	target
age	1.000000	0.173737	0.185085	0.335427	0.260061	0.219243	-0.075181	0.203841	0.270777	0.301731	0.187889
sex	0.173737	1.000000	0.088161	0.241013	0.035277	0.142637	-0.379090	0.332115	0.149918	0.208133	0.043062
bmi	0.185085	0.088161	1.000000	0.395415	0.249777	0.261170	-0.366811	0.413807	0.446159	0.388680	0.586450
bp	0.335427	0.241013	0.395415	1.000000	0.242470	0.185558	-0.178761	0.257653	0.393478	0.390429	0.441484
s1	0.260061	0.035277	0.249777	0.242470	1.000000	0.896663	0.051519	0.542207	0.515501	0.325717	0.212022
s2	0.219243	0.142637	0.261170	0.185558	0.896663	1.000000	-0.196455	0.659817	0.318353	0.290600	0.174054
s3	-0.075181	-0.379090	-0.366811	-0.178761	0.051519	-0.196455	1.000000	-0.738493	-0.398577	-0.273697	-0.394789
s4	0.203841	0.332115	0.413807	0.257653	0.542207	0.659817	-0.738493	1.000000	0.617857	0.417212	0.430453
s5	0.270777	0.149918	0.446159	0.393478	0.515501	0.318353	-0.398577	0.617857	1.000000	0.464670	0.565883
s6	0.301731	0.208133	0.388680	0.390429	0.325717	0.290600	-0.273697	0.417212	0.464670	1.000000	0.382483
target	0.187889	0.043062	0.586450	0.441484	0.212022	0.174054	-0.394789	0.430453	0.565883	0.382483	1.000000

선형회귀모델-pairplot

```
1 sns.pairplot(df[["target", "bmi", "bp", "s1"]])
2 plt.show()
```



선형회귀모델-pairplot

```
1  import statsmodels.api as sm;
```

```
1  df['intercept'] = 1
2
3  lm = sm.OLS(df['target'], df[['intercept', 'bmi']])
4  results = lm.fit()
5  results.summary()
```

OLS Regression Results

Dep. Variable:	target	**R-squared:**	0.344
Model:	OLS	**Adj. R-squared:**	0.342
Method:	Least Squares	**F-statistic:**	230.7
Date:	Wed, 13 Jan 2021	**Prob (F-statistic):**	3.47e-42
Time:	15:10:50	**Log-Likelihood:**	-2454.0
No. Observations:	442	**AIC:**	4912.
Df Residuals:	440	**BIC:**	4920.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
intercept	152.1335	2.974	51.162	0.000	146.289	157.978
bmi	949.4353	62.515	15.187	0.000	826.570	1072.301

Omnibus:	11.674	**Durbin-Watson:**	1.848
Prob(Omnibus):	0.003	**Jarque-Bera (JB):**	7.310
Skew:	0.156	**Prob(JB):**	0.0259
Kurtosis:	2.453	**Cond. No.**	21.0

target
=152.1335 + 949.4393 * bmi

선형회귀모델-pairplot


```
1 df['intercept'] = 1
2
3 lm = sm.OLS(df['target'], df[['intercept', 'bp']])
4 results = lm.fit()
5 results.summary()
```

OLS Regression Results

Dep. Variable:	target	R-squared:	0.195
Model:	OLS	Adj. R-squared:	0.193
Method:	Least Squares	F-statistic:	106.5
Date:	Wed, 13 Jan 2021	Prob (F-statistic):	1.65e-22
Time:	15:12:23	Log-Likelihood:	-2499.3
No. Observations:	442	AIC:	5003.
Df Residuals:	440	BIC:	5011.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
intercept	152.1335	3.294	46.185	0.000	145.660	158.607
bp	714.7416	69.252	10.321	0.000	578.636	850.847

Omnibus:	10.258	Durbin-Watson:	2.058
Prob(Omnibus):	0.006	Jarque-Bera (JB):	9.969
Skew:	0.329	Prob(JB):	0.00684
Kurtosis:	2.670	Cond. No.	21.0

target
=152.1335 + 714.7416 * bp

문제풀이

- 주택 가격 예측을 위한 선형 회귀 모델을 적용 사례를 설명하시오.
- 당뇨병 지행도 예측을 위한 선형 회귀 모델을 적용 사례를 설명하시오.
- 선형 회귀 모델에 대한 평가 방법을 설명하시오.

요약

- 주택 가격 예측을 위한 선형 회귀 모델을 적용 사례를 공부하였음.
- 당뇨병 지행도 예측을 위한 선형 회귀 모델을 적용 사례를 공부하였음.
- 선형 회귀 모델에 대한 평가 방법을 공부하였음.

01

개요

- 회귀분석
- 회귀모델
- 선형회귀 모델

02

선형회귀 모델 적용

- 주택 가격 예측
- 선형회귀 모델의 평가 방법
- 당뇨병 진행도 예측

03

선형회귀 모델

- Ridge 모델 이용 주택가격예측
- Lasso 모델 이용 주택가격예측
- 선형모델 이용 주식가격예측

학습목표

충북대학교
SW중심대학사업단

이번 파트에서는 회귀 분석을 위해 요구되는 다음과 같은 내용을 공부한다.

- Ridge 모델 이용 주택가격예측
- Lasso 모델 이용 주택가격예측
- 선형모델 이용 주식가격예측

선형회귀분석: Ridge모델

- 가장 널리 쓰이는 알고리즘

- OLS 모델과 예측함수는 같지만 모든 특성이 출력에 주는 영향을 최소화하여 기울기를 작게 만듬. (가중치 w의 모든 원소를 0에 가깝게 만드는 방법)

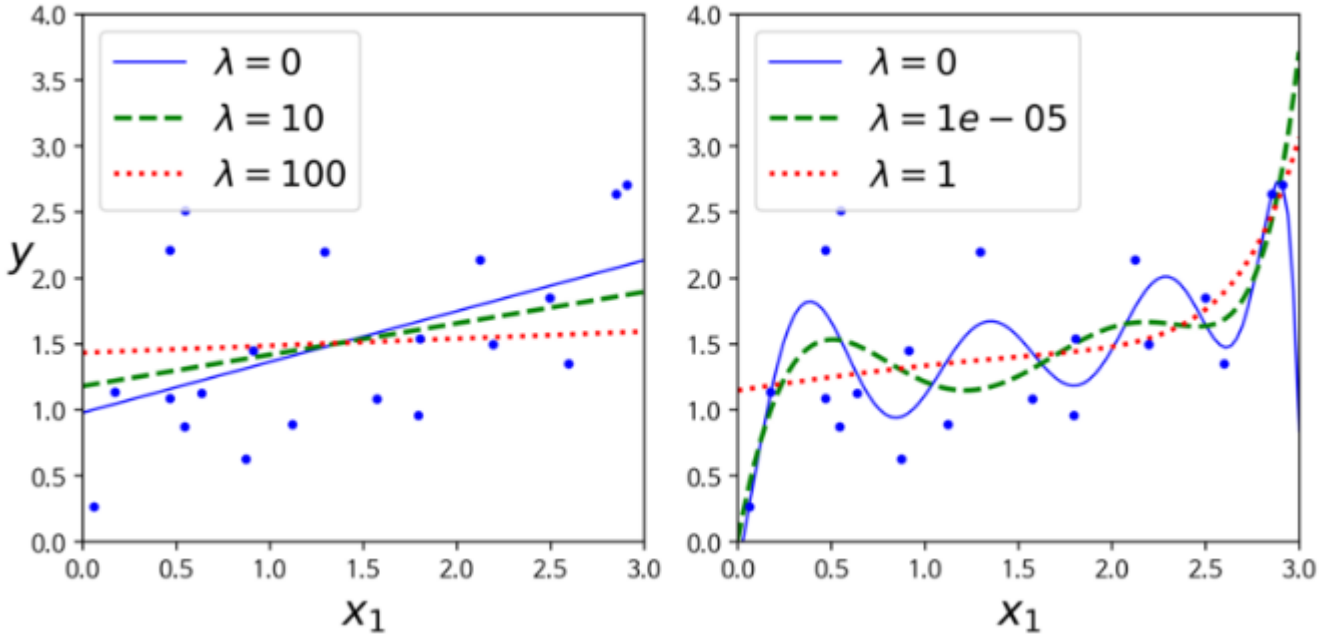
- 리지 모델은 L2 규제 방식 활용
  (규제 : 모델 복잡도가 높아지지 않도록 모델을 강제로 제한함)

- 매개변수 alpha : 값이 커지면 복잡도가 낮아지고 작아지면 복잡도가 높아짐
  값이 너무 작아지면 OLS 모델과 거의 유사해짐

선형회귀분석: Ridge모델

$$J(w)_{Ridge} = \sum_{i=1}^{n} \left(y^{(i)} - \hat{y}^{(i)}\right)^2 + \lambda ||w||_2^2$$

$$L2 : \lambda ||w||_2^2 = \lambda \sum_{j=1}^{m} w_j^2$$



Ridge 모델 이용 주택가격 예측


```
1  import mglearn
2  from sklearn.linear_model import Ridge
3  from sklearn import model_selection
4
5  X, y =mglearn.datasets.load_extended_boston()
6  print(X)
7  print(y)
```

```
[[0.00000000e+00 1.80000000e-01 6.78152493e-02 ... 1.00000000e+00
  8.96799117e-02 8.04248656e-03]
 [2.35922539e-04 0.00000000e+00 2.42302053e-01 ... 1.00000000e+00
  2.04470199e-01 4.18080621e-02]
 [2.35697744e-04 0.00000000e+00 2.42302053e-01 ... 9.79579831e-01
  6.28144504e-02 4.02790570e-03]
 ...
 [6.11892474e-04 0.00000000e+00 4.20454545e-01 ... 1.00000000e+00
  1.07891832e-01 1.16406475e-02]
 [1.16072990e-03 0.00000000e+00 4.20454545e-01 ... 9.82676920e-01
  1.29930407e-01 1.71795127e-02]
 [4.61841693e-04 0.00000000e+00 4.20454545e-01 ... 1.00000000e+00
  1.69701987e-01 2.87987643e-02]]
[24.  21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 15.  18.9 21.7 20.4
 18.2 19.9 23.1 17.5 20.2 18.2 13.6 19.6 15.2 14.5 15.6 13.9 16.6 14.8
 18.4 21.  12.7 14.5 13.2 13.1 13.5 18.9 20.  21.  24.7 30.8 34.9 26.6
 25.3 24.7 21.2 19.3 20.  16.6 14.4 19.4 19.7 20.5 25.  23.4 18.9 35.4
 24.7 31.6 23.3 19.6 18.7 16.  22.2 25.  33.  23.5 19.4 22.  17.4 20.9
```

Ridge 모델 이용 주택가격 예측


```
1  import mglearn
2  from sklearn.linear_model import Ridge
3  from sklearn import model_selection
4
5  X, y =mglearn.datasets.load_extended_boston()
6
7  X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, random_state=0)
8
9  ridge = Ridge().fit(X_train, y_train)
10
11 print("훈련 세트 점수 : {:.2f}".format(ridge.score(X_train,y_train)))
12
13 print("테스트 세트 점수 : {:.2f}".format(ridge.score(X_test,y_test)))
```

```
C:₩ProgramData₩Anaconda3₩lib₩site-packages₩sklearn₩externals₩six.py:31: DeprecationWarning: The mod
ill be removed in version 0.23 since we've dropped support for Python 2.7. Please rely on the offic
oject/six/).
  "(https://pypi.org/project/six/).", DeprecationWarning)
C:₩ProgramData₩Anaconda3₩lib₩site-packages₩sklearn₩externals₩joblib₩__init__.py:15: DeprecationWarn
ated in 0.21 and will be removed in 0.23. Please import this functionality directly from joblib, wh
joblib. If this warning is raised when loading pickled models, you may need to re-serialize those m
  warnings.warn(msg, category=DeprecationWarning)
```

```
훈련 세트 점수 : 0.89
테스트 세트 점수 : 0.75
```

Ridge 모델 이용 주택가격 예측


```
# alpha=10인 Ridge 모델
ridge10 = Ridge(alpha=10).fit(X_train, y_train)
print("훈련 세트 점수 : {:.2f}".format(ridge10.score(X_train,y_train)))
print("테스트 세트 점수 : {:.2f}".format(ridge10.score(X_test,y_test)))
```

```
훈련 세트 점수 : 0.79
테스트 세트 점수 : 0.64
```

```
# alpha=0.1인 Ridge 모델
ridge10 = Ridge(alpha=0.1).fit(X_train, y_train)
print("훈련 세트 점수 : {:.2f}".format(ridge10.score(X_train,y_train)))
print("테스트 세트 점수 : {:.2f}".format(ridge10.score(X_test,y_test)))
```

```
훈련 세트 점수 : 0.93
테스트 세트 점수 : 0.77
```

선형회귀분석: Rasso 모델

- Ridge 회귀와 같이 가중치 w를 0에 가깝게 만드는 모델

- 라소 모델은 L1 규제 방식을 활용하며 어떤 가중치는 0이 되기도 함
  즉, 완전히 제외되는 특성이 생길 수 있음

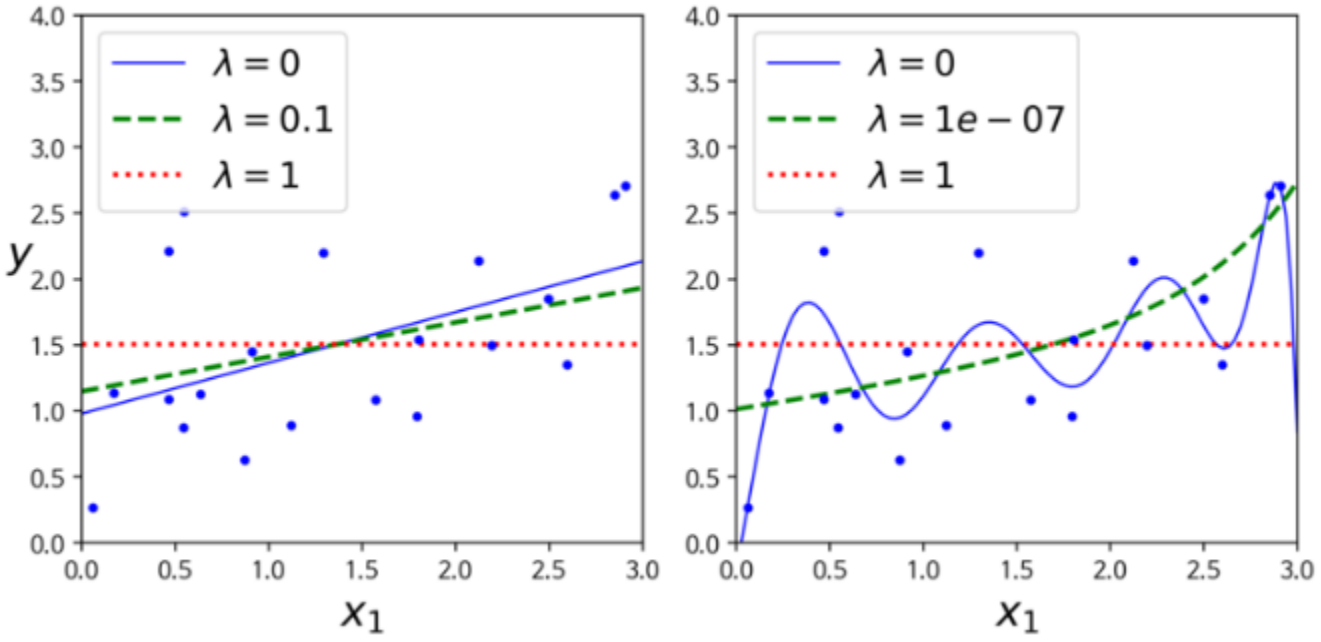
- 특성 선택이 자동으로 이루어지므로 모델을 이해하기 쉬워지고 가장 중요한 특성을 알 수 있음

- 매개변수 alpha : 값이 커지면 복잡도가 낮아지고 작아지면 복잡도가 높아짐
  값이 너무 작아지면 OLS 모델과 거의 유사해짐

선형회귀분석: Rasso 모델

$$J(w)_{LASSO} = \sum_{i=1}^{n} \left(y^{(i)} - \hat{y}^{(i)}\right)^2 + \lambda ||w||_1$$

$$L1{:}\lambda ||w||_1 = \lambda \sum_{j=1}^{m} |w_j|$$



Lasso 모델 이용 주택가격 예측

```
1  from sklearn.linear_model import Lasso
2  lasso = Lasso().fit(X_train, y_train)
3  print("훈련 세트 점수 : {:.2f}".format(lasso.score(X_train,y_train)))
4  print("테스트 세트 점수 : {:.2f}".format(lasso.score(X_test,y_test)))
5  print("사용한 특성의 개수:", np.sum(lasso.coef_ !=0))
```

```
훈련 세트 점수 : 0.29
테스트 세트 점수 : 0.21
사용한 특성의 개수: 4
```

Lasso 모델 이용 주택가격 예측


```
# alpha=0.01일 때 ('max_iter' 증가시키지 않으면 경고 발생)

lasso001= Lasso(alpha=0.01, max_iter=100000).fit(X_train,y_train)
print("훈련 세트 점수: {:.2f}".format(lasso001.score(X_train,y_train)))
print("테스트 세트 점수 : {:.2f}".format(lasso001.score(X_test,y_test)))
print("사용한 특성의 개수:", np.sum(lasso001.coef_ !=0))
```

훈련 세트 점수 : 0.90
테스트 세트 점수 : 0.77
사용한 특성의 개수 : 33

Lasso 모델 이용 주택가격 예측


```
## alpha=0.0001일 때 ('max_iter' 증가시키지 않으면 경고 발생)

lasso00001= Lasso(alpha=0.0001, max_iter=100000).fit(X_train,y_train)
print("훈련 세트 점수: {:.2f}".format(lasso00001.score(X_train,y_train)))
print("테스트 세트 점수 : {:.2f}".format(lasso00001.score(X_test,y_test)))
print("사용한 특성의 개수:", np.sum(lasso00001.coef_ !=0))
```

훈련 세트 점수 : 0.95
테스트 세트 점수 : 0.64
사용한 특성의 개수 : 96

선형 모델이용 주식 가격 예측

Year	Month	Interest_Rate	Unemployment_Rate	Stock_Index_Price
2017	12	2.75	5.3	1464
2017	11	2.5	5.3	1394
2017	10	2.5	5.3	1357
2017	9	2.5	5.3	1293
2017	8	2.5	5.4	1256
2017	7	2.5	5.6	1254
2017	6	2.5	5.5	1234
2017	5	2.25	5.5	1195
2017	4	2.25	5.5	1159
2017	3	2.25	5.6	1167
2017	2	2	5.7	1130
2017	1	2	5.9	1075
2016	12	2	6	1047
2016	11	1.75	5.9	965
2016	10	1.75	5.8	943
2016	9	1.75	6.1	958
2016	8	1.75	6.2	971
2016	7	1.75	6.1	949
2016	6	1.75	6.1	884
2016	5	1.75	6.1	866
2016	4	1.75	5.9	876
2016	3	1.75	6.2	822
2016	2	1.75	6.2	704
2016	1	1.75	6.1	719

선형 모델이용 주식 가격 예측


```
from pandas import DataFrame
import matplotlib.pyplot as plt
```

```
from pandas import DataFrame Stock_Market =
{'Year': [2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,
2017,2017,2016,2016,2016,2016,2016,2016,2016,2016,2016,2016,2016,2016],
'Month': [12, 11,10,9,8,7,6,5,4,3,2,1,12,11,10,9,8,7,6,5,4,3,2,1],
'Interest_Rate': [2.75,2.5,2.5,2.5,2.5,2.5,2.5,2.25,2.25,2.25,2,2,2,
1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75],
'Unemployment_Rate': [5.3,5.3,5.3,5.3,5.4,5.6,5.5,5.5,5.5,5.6,
5.7,5.9,6,5.9,5.8,6.1,6.2,6.1,6.1,6.1,5.9,6.2,6.2,6.1],
'Stock_Index_Price': [1464,1394,1357,1293,1256,1254,1234,1195,1159,
1167,1130,1075,1047,965,943,958,971,949,884,866,876,822,704,719] }


df =
DataFrame(Stock_Market,columns=['Year','Month','Interest_Rate','Unemployment_
Rate','Stock_Index_Price'])
print (df)
```

선형 모델이용 주식 가격 예측


```
plt.scatter(df['Interest_Rate'], df['Stock_Index_Price'],
color='red')
plt.title('Stock Index Price Vs Interest Rate', fontsize=14)
plt.xlabel('Interest Rate', fontsize=14)
plt.ylabel('Stock Index Price', fontsize=14)
plt.grid(True)
plt.show()

plt.scatter(df['Unemployment_Rate'], df['Stock_Index_Price'],
color='green')
plt.title('Stock Index Price Vs Unemployment Rate',
fontsize=14)
plt.xlabel('Unemployment Rate', fontsize=14)
plt.ylabel('Stock Index Price', fontsize=14)
plt.grid(True) plt.show()
```

선형 모델이용 주식 가격 예측


```
X = df[['Interest_Rate','Unemployment_Rate']]
Y== df['Stock_Index_Price']
```

```
# with sklearn
regr =
linear_model.LinearRegression()

regr.fit(X, Y)

print('Intercept: ₩n',
regr.intercept_)

print('Coefficients: ₩n',
regr.coef_)
```

Stock_Index_Price = (Intercept) + (Interest_Rate coef)*X_1 + (Unemployment_Rate coef)*X_2

Stock_Index_Price = (1798.4040) + (345.5401)*X_1 + (-250.1466)*X_2

선형 모델이용 주식 가격 예측


```
X = df[['Interest_Rate','Unemployment_Rate']]
Y== df['Stock_Index_Price']
```

```
# prediction with sklearn
New_Interest_Rate = 2.75
New_Unemployment_Rate = 5.3
print ('Predicted Stock Index Price: ₩n',
regr.predict([[New_Interest_Rate ,New_Unemployment_Rate]]))

# with statsmodels
X = sm.add_constant(X) # adding a constant
model = sm.OLS(Y, X).fit()
predictions = model.predict(X)
print_model = model.summary()

print(print_model)
```

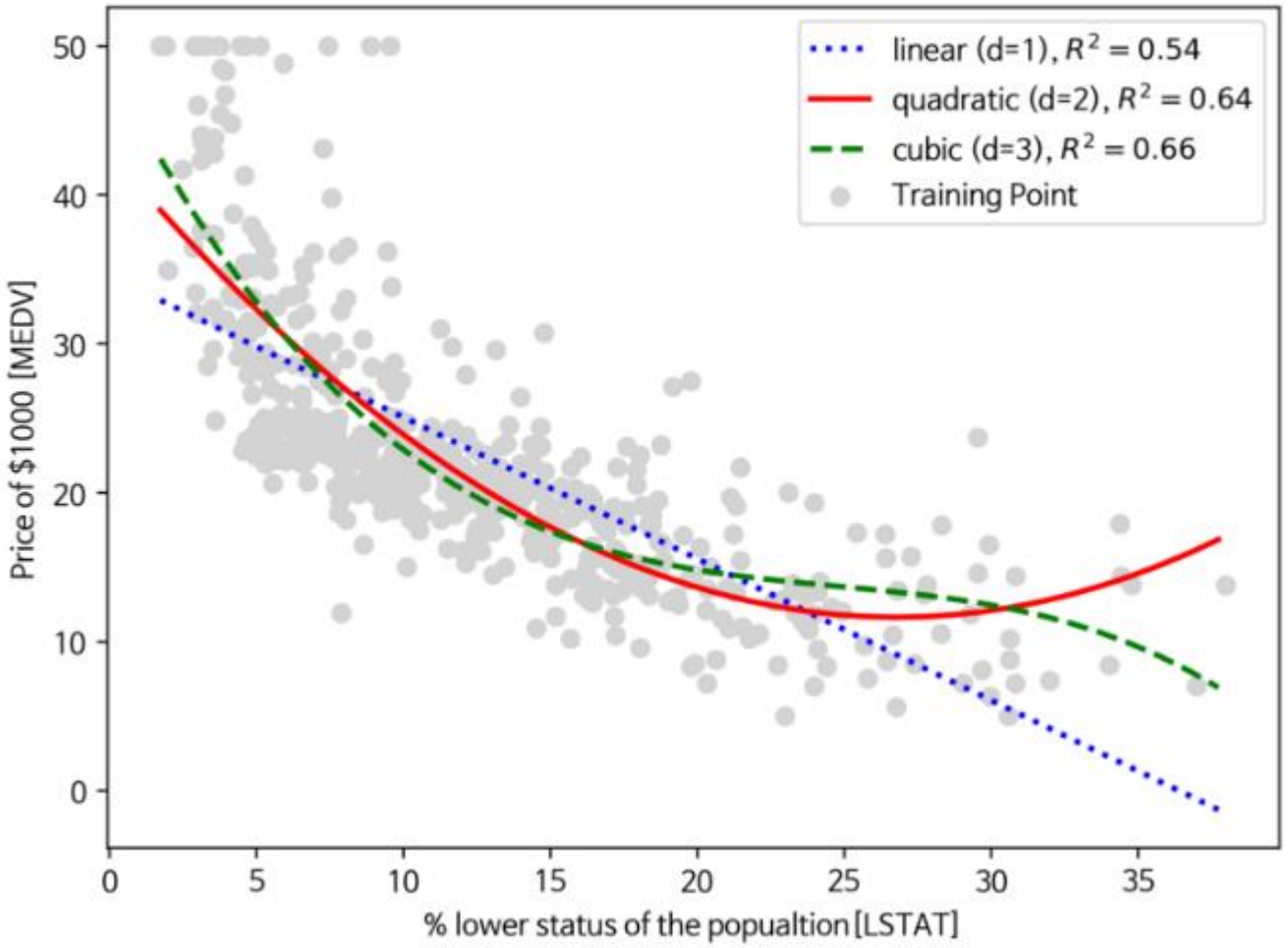
Stock_Index_Price = (1798.4040) + (345.5401)*(2.75) + (-250.1466)*(5.3) = 1422.86

다항회귀 분석 모델

$$y = w_0 + w_1 x + w_2 x^2 + \ldots + w_n x^n$$

```
x = df[['LSTAT']].values
y = df[['MEDV']].values

model = LinearRegression()

quadratic = PolynomialFeatures(degree=2)
cubic = PolynomialFeatures(degree=3)
x_quad = quadratic.fit_transform(x)
x_cubic = cubic.fit_transform(x)

x_fit = np.arange(x.min(), x.max(), 1)[:, np.newaxis]
model = model.fit(x, y)
y_fit = model.predict(x_fit)
r2 = r2_score(y, model.predict(x))

model = model.fit(x_quad,y)
y_quad_fit = model.predict(quadratic.fit_transform(x_fit))
quadratic_r2 = r2_score(y, model.predict(x_quad))

model = model.fit(x_cubic,y)
y_cubic_fit = model.predict(cubic.fit_transform(x_fit))
cubic_r2 = r2_score(y, model.predict(x_cubic))

plt.scatter(x, y, label="Training Point", color="lightgray")
plt.plot(x_fit, y_fit, label="linear (d=1), $R^2=%.2f$" % r2, color="blue", lw=2, lin
plt.plot(x_fit, y_quad_fit, label="quadratic (d=2), $R^2=%.2f$" % quadratic_r2, color
plt.plot(x_fit, y_cubic_fit, label="cubic (d=3), $R^2=%.2f$" % cubic_r2, color="green
plt.xlabel("% lower status of the popualtion[LSTAT]")
plt.ylabel("Price of $1000 [MEDV]")
plt.legend(loc="upper right")
save_fig("Multi Regression")
plt.show()
```

다항회귀 분석 모델


linear (d=1), $R^2 = 0.54$
quadratic (d=2), $R^2 = 0.64$
cubic (d=3), $R^2 = 0.66$
Training Point
Price of $1000 [MEDV]
% lower status of the popualtion [LSTAT]
0
10
20
30
40
50
0
5
10
15
20
25
30
35

문제풀이

- Linear Regression, Ridge. Lasso 모델의 차이점을 설명하시오

- 다항 회귀 분석 모델을 설명하시오.

요약

- 다음과 같은 선형 회귀 모델을 공부하였음
  - Ridge
  - Lasso
  - 다항 회귀 모델