머신러닝응용 제03강

Basic Methods for Regression 2

첨단공학부 김동하교수

제03강 Basic Methods for Regression 2

1	모형의 과적합에 대해 학습한다.
2	주성분 회귀분석에 대해 학습한다.
3	벌점화 회귀분석에 대해 학습한다.

핵심 단어

- 과적합
- 주성분 회귀분석
- Ridge 회귀분석
- Lasso 회귀분석

03강. Basic Methods for Regression 1

01. 모형의 과적합

1) 모형의 과적합이란

과적합 또는 과대적합 (Overfitting)

- 모형이 학습데이터를 과하게 학습하는 것을 의미.
- $Y_i = f^*(X_i) + \epsilon_i, i = 1, \cdots, n$
- 학습시키는 모형은 관계 f^* 만을 학습하길 원함.
- 데이터의 정보를 과하게 학습할 경우 오차항 ϵ_i의 정보도 모형에 녹아들게 됨. -> 과적합
- 과적합할 경우 모형의 예측 성능 저하.

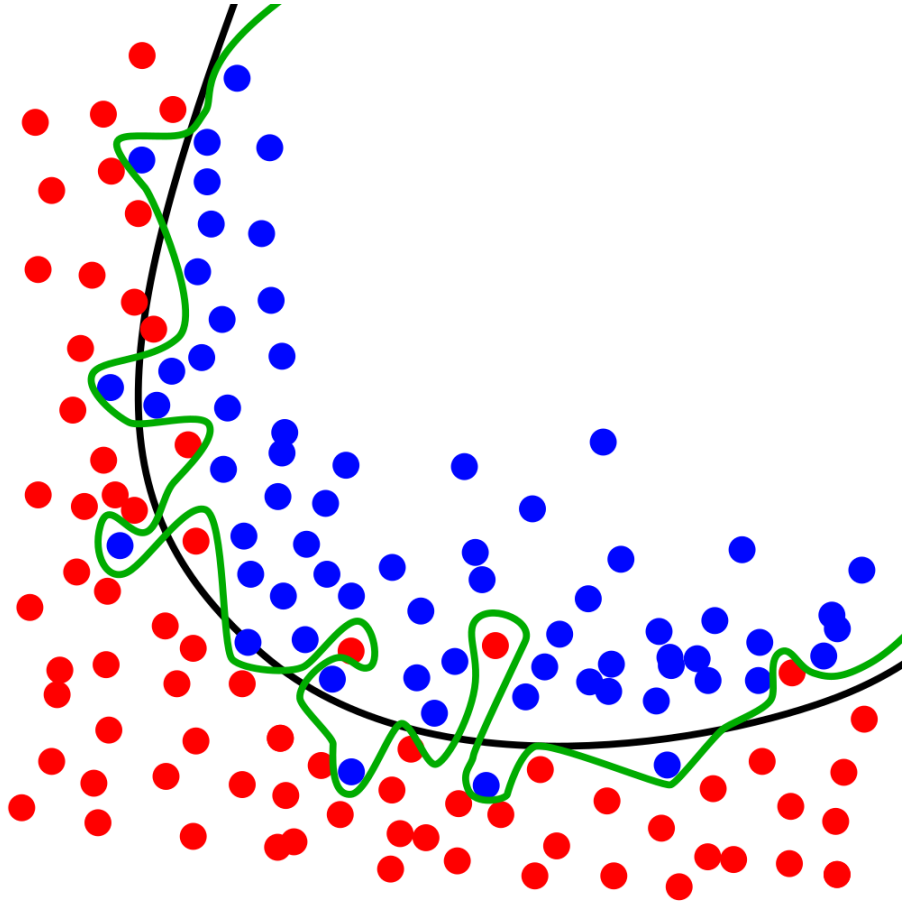
1) 모형의 과적합이란

과소적합 (Underfitting)

- 반대로, 모델이 너무 단순해서 데이터의 구조 및 패턴을 정확히 반영하지 못하는 경우도 생길 수 있음.
  - 과소적합 (Underfitting)
- 과소적합이 일어날 경우에도 모형의 성능이 저하.

1) 모형의 과적합이란

- 과적합 또는 과대적합 (Overfitting)



출처: https://ko.wikipedia.org/wiki/%EA%B3%BC%EC%A0%81%ED%95%A9

2) 과적합 방지를 위한 해결책

선형모형에서 과적합을 피하려면?

- 입력데이터에서 중요한 정보만을 추출한 후 선형모형을 적합 -> 주성분 회귀분석

- 선형모형에서 모수를 적합할 때 추가 제약조건을 부여 -> 벌점화 회귀분석
  - Ridge Regression
  - Lasso Regression

03강. Basic Methods for Regression 1

02. 주성분 회귀분석

1) 차원축소기법

- 분석 대상이 되는 변수의 수를 줄이고, 주요한 정보만을 추출하기 위한 방법론

- 주로 고차원 자료 분석에서 사용함.

1) 차원축소기법

- **변수 선택 (Feature selection)**
  - 기존 변수 중 중요한 일부 변수만을 빼내는 기법.
- **변수 변환 (Feature transformation)**
  - 기존 변수를 조합해 새로운 변수를 만드는 기법.

2) 주성분분석

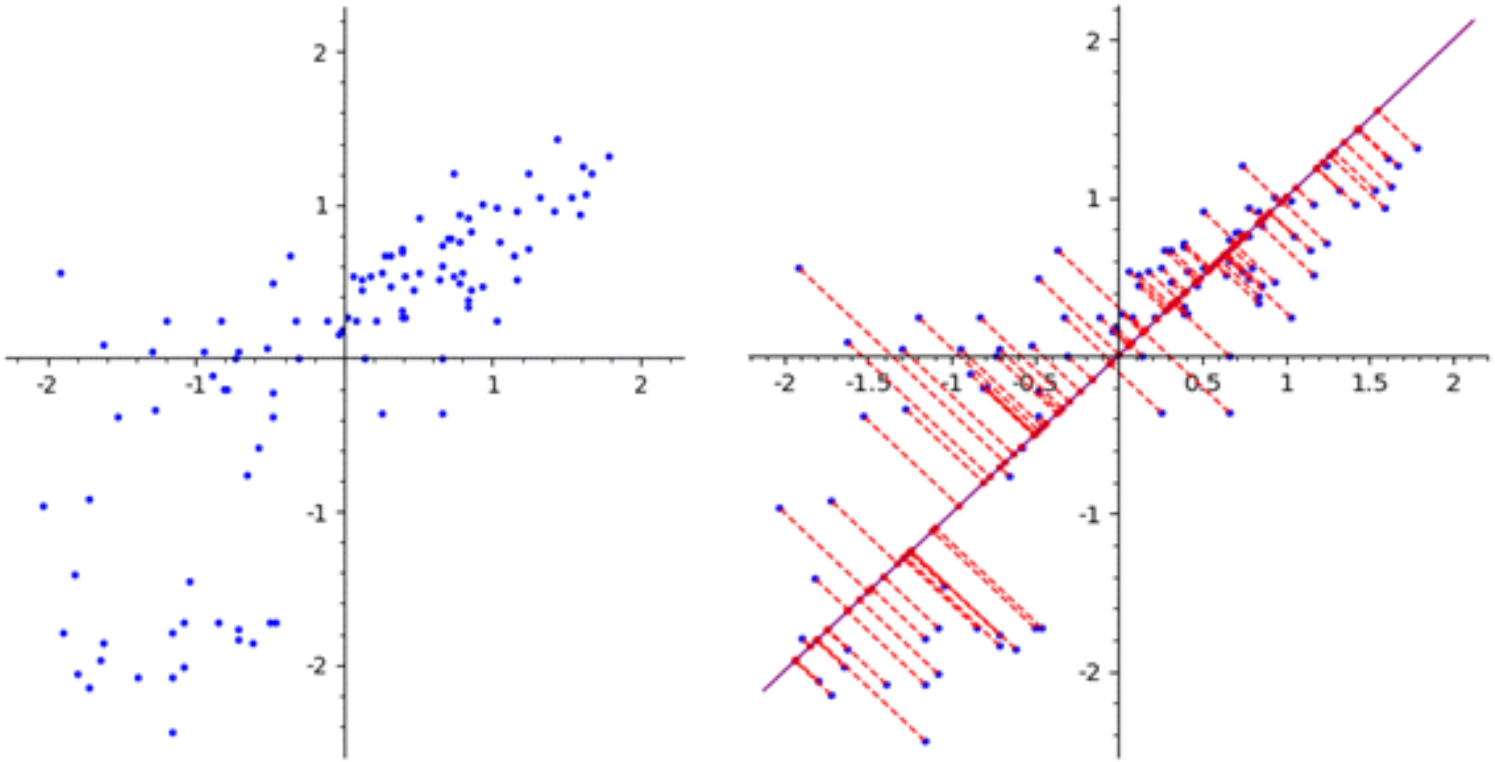
Principal Component Analysis (PCA)

- 대표적인 차원 축소 기법 중 하나 (변수 변환에 기초)

- 기존 변수들의 선형 변환을 통해 데이터를 잘 설명하는 새로운 변수들을 찾고, 이를 이용해 데이터의 차원을 축소.

2) 주성분분석

Principal Component Analysis (PCA)



출처: http://matrix.skku.ac.kr/math4ai-intro/W12/

3) 주성분 개수 결정

최적의 주성분 개수 선정

- 주성분 중에서 데이터의 주요 정보를 갖고 있는 최적의 주성분 개수를 구해야 함.

- Scree Plot을 이용해 결정

3) 주성분 개수 결정

Scree Plot

- PCA 분석 결과를 이용해 고유값-주성분의 분산 변화를 보는 그래프.

- 분산 변화율이 완만해지는 주성분의 수를 선정.

4) 주성분 회귀분석

- 주성분분석 + 선형회귀모형
  - 설명 변수 데이터에 주성분분석을 적용하여 설명 변수의 차원을 축소
  - 주성분분석을 통해 얻은 축소 데이터를 이용해 종속변수를 예측하는 선형회귀모형 적합

4) 주성분 회귀분석

- 주성분회귀분석의 장단점
  - 과적합을 막을 수 있기 때문에 더 좋은 예측 성능을 기대할 수 있음.
  - 다중 공선성 문제를 해결할 수 있음.
  - 다만, 모형의 해석이 어려움.
    - 각각의 주성분은 기존 설명 변수들의 선형결합된 형태이기 때문.

5) Python을 이용한 실습

Fat dataset

- 252명 남성에 대한 나이, 몸무게, 키 등의 신체 정보와 비만도를 측정한 자료.

- 나머지 정보를 이용해 비만도 (brozek)를 예측하는 모형을 구축해보자.

5) Python을 이용한 실습

Fat dataset 불러오기

```
data_file = "./data/fat.csv"
fat = pd.read_csv(data_file)
print(fat.shape)
fat.head()
```

```
(252, 18)
```

	brozek	siri	density	age	weight	height	adipos	free	neck
0	12.6	12.3	1.0708	23	154.25	67.75	23.7	134.9	36.2
1	6.9	6.1	1.0853	22	173.25	72.25	23.4	161.3	38.5
2	24.6	25.3	1.0414	22	154.00	66.25	24.7	116.0	34.0
3	10.9	10.4	1.0751	26	184.75	72.25	24.9	164.7	37.4
4	27.8	28.7	1.0340	24	184.25	71.25	25.6	133.1	34.4

5) Python을 이용한 실습

◆주성분분석 적용

- 반드시 설명 변수 데이터를 표준화하는 작업이 선행.

```
xfat_st = preprocessing.StandardScaler().fit_transform(xfat)
feature_names = ['siri', 'density', 'age', 'weight', 'height', 'adipos', 'free', 'neck',
       'chest', 'abdom', 'hip', 'thigh', 'knee', 'ankle', 'biceps', 'forearm',
       'wrist']
xfat_st = pd.DataFrame(xfat_st, columns=feature_names)
xfat_st.head()
```

	siri	density	age	weight	height	adipos	free	neck	chest	abdom
0	-0.820246	0.801647	-1.740073	-0.841246	-0.656205	-0.477058	-0.484401	-0.738665	-0.918048	-0.683533
1	-1.562573	1.565061	-1.819583	-0.193462	0.574790	-0.559456	0.966512	0.209365	-0.858621	-0.887963
2	0.736245	-0.746240	-1.819583	-0.849769	-1.066536	-0.202398	-1.523123	-1.645475	-0.597144	-0.432643

5) Python을 이용한 실습

◆주성분분석 결과

```
pca = PCA(n_components = 10)
pca_components = pca.fit_transform(xfat_st)
pca_xfat = pd.DataFrame(data=pca_components,columns=
                        ['pc1','pc2','pc3','pc4','pc5',
                         'pc6','pc7','pc8','pc9','pc10'])
pca_xfat.head()
```

```
(252, 10)
```

	pc1	pc2	pc3	pc4	pc5	pc6	pc7
0	-2.555110	-0.642961	1.847360	0.353975	-0.215243	-0.259544	-0.056970
1	-1.434682	-2.932980	0.672117	0.377292	-0.034448	-0.200036	-0.586670
2	-2.157678	1.387028	2.872635	-1.240407	1.526779	0.591437	0.039899

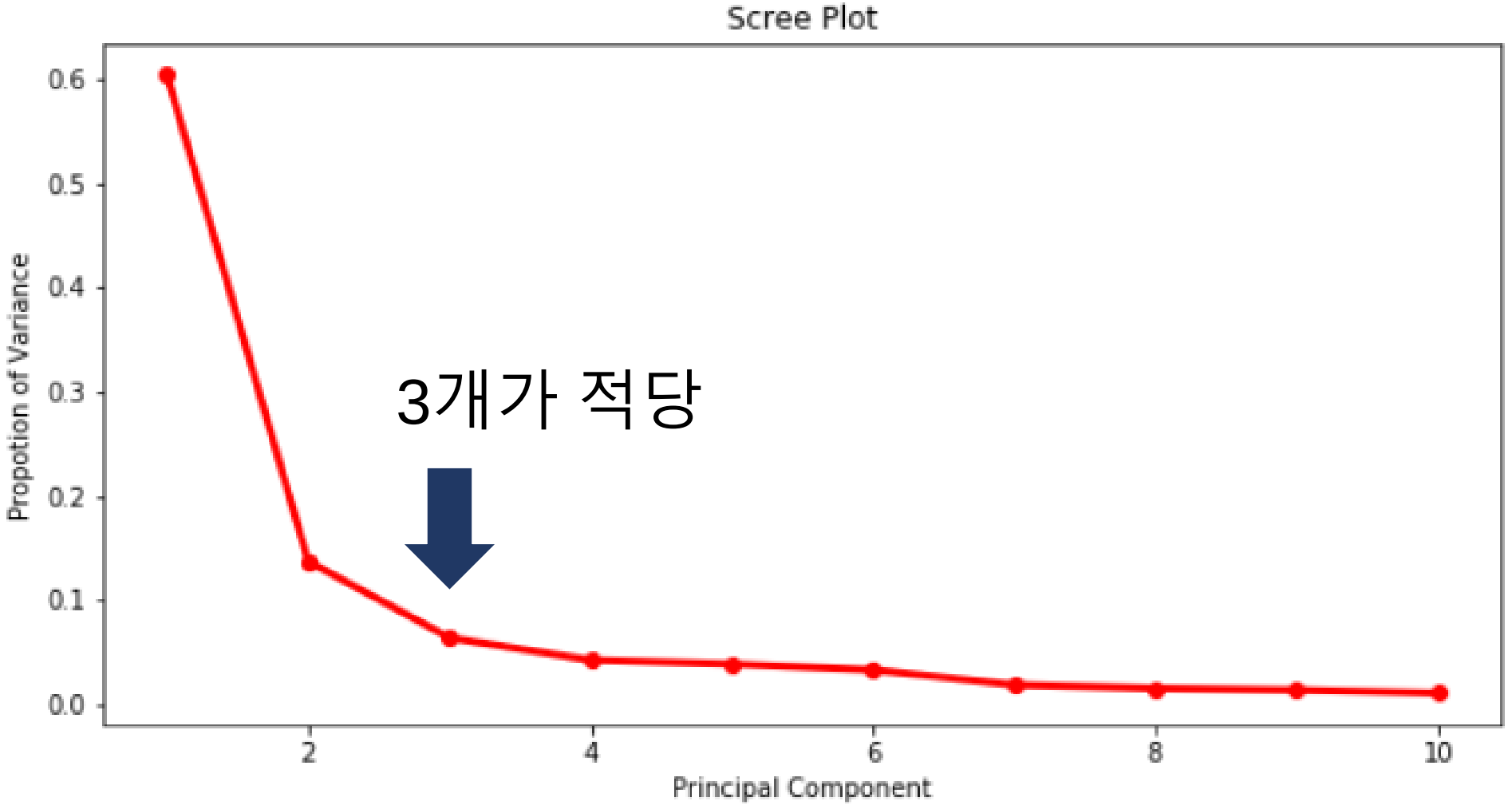
5) Python을 이용한 실습

◆ 최적의 주성분 수 계산 (Scree Plot)

```
fig = plt.figure(figsize = (10, 5))
sing_vals = np.arange(10) + 1
plt.plot(sing_vals, pca.explained_variance_ratio_,
         'ro-', linewidth = 3)
plt.title('Scree Plot')
plt.xlabel('Principal Component')
plt.ylabel('Propotion of Variance')
```

5) Python을 이용한 실습

◆ 최적의 주성분 수 계산 (Scree Plot)



5) Python을 이용한 실습

- 3개의 주성분 정보를 이용해 비만도를 예측하는 선형회귀모형 적합
  - brozek ~ pc1 + pc2 + pc3

```
pca_fat = pd.concat([yfat,pca_xfat[['pc1', 'pc2', 'pc3']]], axis=1)
## 선형모형 적합
pcalmfit = smf.ols(formula='brozek~pc1+pc2+pc3', data=pca_fat).fit()
print(pcalmfit.summary())
```

5) Python을 이용한 실습

- 3개의 주성분 정보를 이용해 비만도를 예측하는 선형회귀모형 적합

	coef	std err	t	P>|t|
Intercept	18.9385	0.154	122.910	0.000
pc1	1.6608	0.048	34.525	0.000
pc2	3.2789	0.101	32.524	0.000
pc3	0.7015	0.148	4.748	0.000

03강. Basic Methods for Regression 1

03. 벌점화 회귀분석

1) 벌점화 회귀분석

벌점 함수를 이용한 회귀분석

- 일반적인 선형회귀 추정 방법은 제곱손실함수를 최소화하는 최소제곱법.
- 데이터에 내재되어 있는 오차 및 다중 공선성에 민감하게 반응할 수 있음.
- 이를 해결하기 위해 모수가 상대적으로 둔감하게 반응하도록 벌점화 함수를 추가.

1) 벌점화 회귀분석

벌점 함수를 이용한 회귀분석

- 제곱손실함수 + 벌점화함수 = 벌점화된 손실함수

- 벌점화된 손실함수를 최소화하는 모수를 추정
  -> 벌점화 회귀분석

2) 벌점화된 손실함수

- **제곱손실함수**

$$\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \beta_0 - \beta_1 x_{i1} - \cdots - \beta_p x_{ip}\right)^2$$

- **벌점함수**

$$pen(\beta_1, \cdots, \beta_p)$$

2) 벌점화된 손실함수

벌점화된 손실함수

$$\frac{1}{n}\sum_{i=1}^{n}\left(y_i-\beta_0-\beta_1 x_{i1}-\cdots-\beta_p x_{ip}\right)^2+\boxed{\lambda}\times\boxed{pen(\beta_1,\cdots,\beta_p)}$$

벌점함수의 종류

- 여러가지 벌점 함수가 존재.
- 두 가지 벌점 함수에 대해 학습할 것.

2) 벌점화된 손실함수

벌점함수의 종류

- $pen(\beta_1, \cdots, \beta_p) = \sum_{j=1}^{p} \beta_j^2$
  - Ridge regression (능형 회귀) - L_2 벌점함수 사용

- $pen(\beta_1, \cdots, \beta_p) = \sum_{j=1}^{p} |\beta_j|$
  - Lasso regression (라쏘 회귀) - L_1 벌점함수 사용

2) 벌점화된 손실함수

- 둘 모두 벌점화 함수를 이용해서 모형의 성능을 향상.

- Lasso의 경우 모수 추정량이 정확히 0이 되는 모수가 존재.
  - 변수 선택의 기능

2) 벌점화된 손실함수

- **조율모수:** $\lambda > 0$
  - 모수에 부여하는 벌점의 크기를 결정하는 모수
  - 조율모수의 크기에 따라 다른 모수가 추정됨.
  - $\lambda = 0$ -> 일반적인 최소제곱법
  - $\lambda = \infty$ -> 설명변수의 모든 기울기는 0으로 추정

- **최적의 조율모수를 선택해야 함.**
  - 검증자료 또는 **교차검증법**을 이용.

2) 벌점화된 손실함수

교차검증법

- 데이터를 훈련용과 검증용으로 교차하여 선택하는 방법.
- 다양한 조율모수에 대해서 교차검증을 이용한 손실 점수를 계산하고, 이를 최소로 하는 최적의 조율모수를 선택.
- 선택된 조율모수를 이용해 최종 모형을 학습.

2) 벌점화된 손실함수

◆ 교차검증법

출처: https://heytech.tistory.com/113

3) Python을 이용한 실습

Ridge regression

- 최적의 조율 모수 선택

```
alphas = np.logspace(2, -10, 200)
rmse = []
for alpha in alphas:
    neg_mse_scores = cross_val_score(Ridge(alpha), xfat_st, yfat,
                                     scoring='neg_mean_squared_error', cv=3)
    rmse_scores = np.sqrt(-1 * neg_mse_scores)
    rmse.append(np.mean(rmse_scores))
```

3) Python을 이용한 실습

Ridge regression

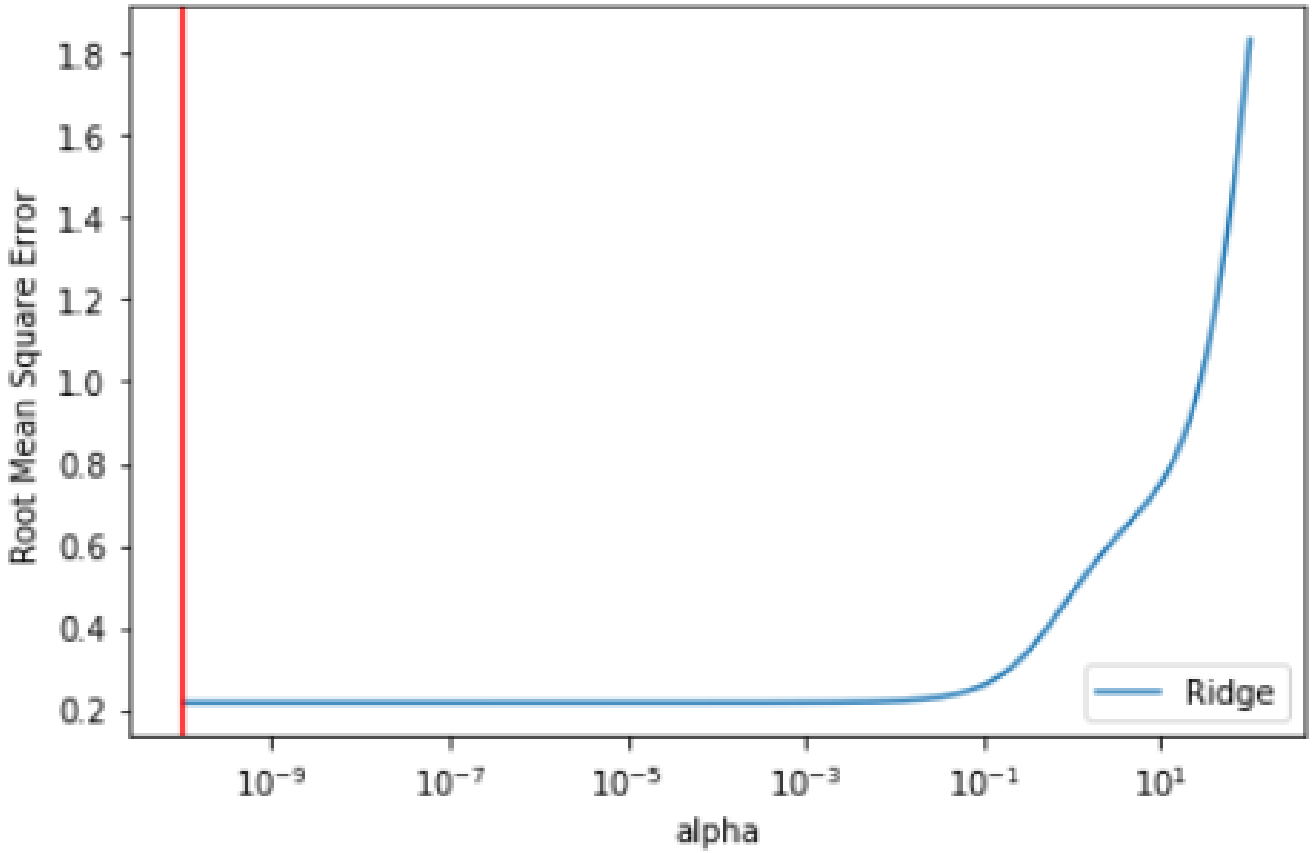
- 최적의 조율 모수 선택

```
best_alpha = alphas[list(rmse==min(rmse)).index(True)]
plt.plot(alphas, rmse, label=Ridge.__name__)
plt.axvline(best_alpha, color = 'r')
plt.legend(loc = 'lower right')
plt.xscale("log")
plt.xlabel('alpha')
plt.ylabel('Root Mean Square Error')
plt.tight_layout()
plt.show()
```

3) Python을 이용한 실습

Ridge regression

- 최적의 조율 모수 선택



3) Python을 이용한 실습

Ridge regression

- 최적의 조율 모수를 이용해서 최종 모형 학습

```
optimal_ridge = Ridge(best_alpha)
optimal_ridge.fit(xfat_st, yfat)
print(optimal_ridge.intercept_)
print(optimal_ridge.coef_)
```

```
18.938492063492067
[ 7.42009341e+00 -1.87004321e-01 -6.62524123e-03  2.48887207e-01
 -1.99545282e-03 -5.57956160e-02 -1.77200971e-01  1.21347867e-03
  1.80509937e-02  1.55655884e-02 -3.18267718e-02  8.22217091e-02
 -6.06869824e-02  4.70068114e-03 -4.43648926e-02  3.02468282e-02
  3.04227118e-02]
```

3) Python을 이용한 실습

Lasso regression

- 최적의 조율 모수 선택

```
alphas = np.logspace(1, -5, 200)
rmse = []
for alpha in alphas:
    neg_mse_scores = cross_val_score(Lasso(alpha, max_iter=10000), xfat_st, yfat,
                                     scoring='neg_mean_squared_error', cv=3)
    rmse_scores = np.sqrt(-1 * neg_mse_scores)
    rmse.append(np.mean(rmse_scores))
```

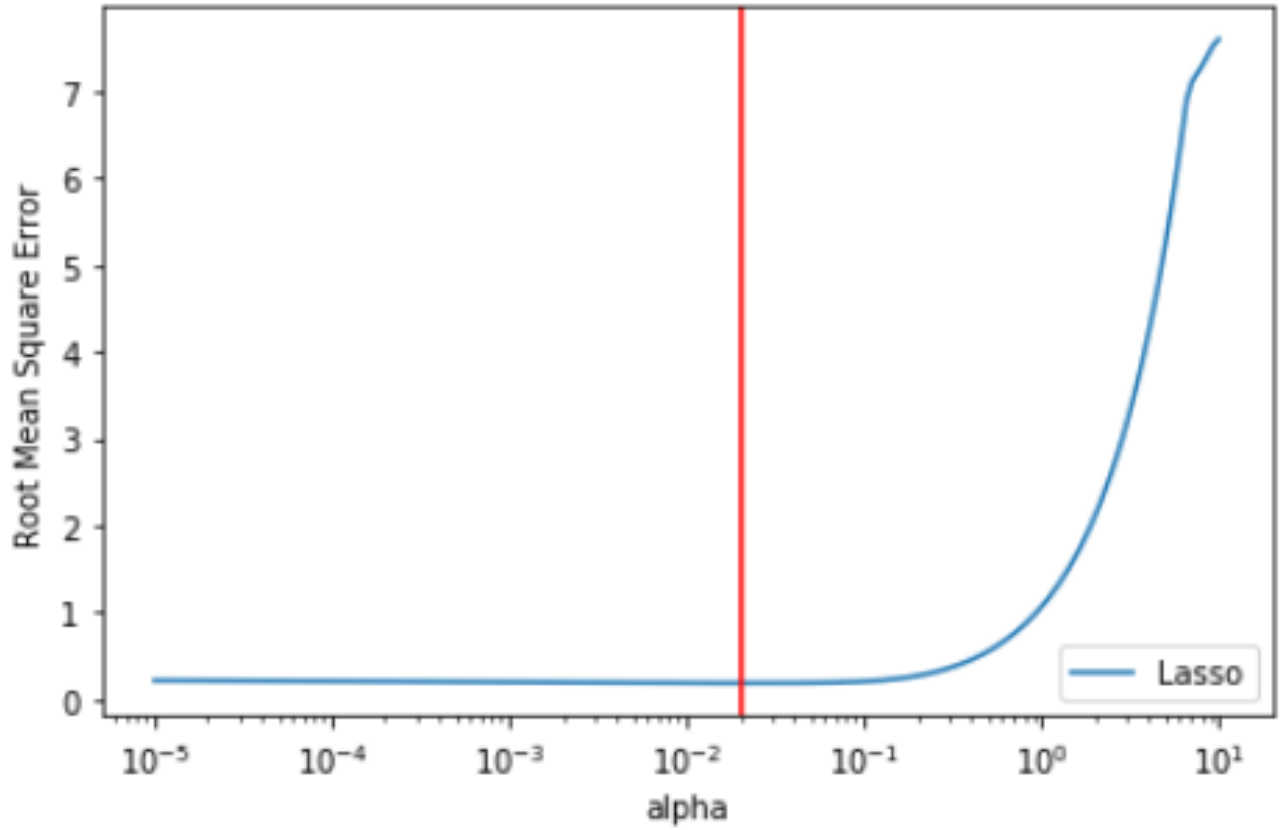
3) Python을 이용한 실습

- Lasso regression
  - 최적의 조율 모수 선택

```
best_alpha = alphas[list(rmse==min(rmse)).index(True)]
plt.plot(alphas, rmse, label=Lasso.__name__)
plt.axvline(best_alpha, color = 'r')
plt.legend(loc = 'lower right')
plt.xscale("log")
plt.xlabel('alpha')
plt.ylabel('Root Mean Square Error')
plt.tight_layout()
plt.show()
```

3) Python을 이용한 실습

- Lasso regression
  - 최적의 조율 모수 선택



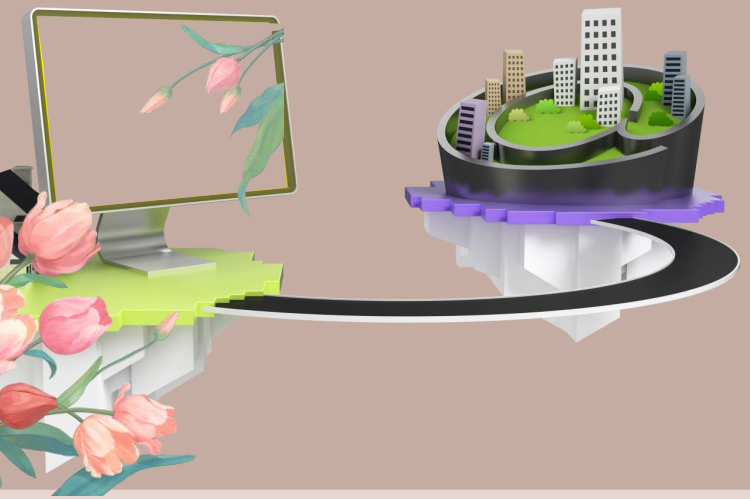
3) Python을 이용한 실습

Lasso regression

- 최적의 조율 모수를 이용해서 최종 모형 학습

```
optimal_lasso = Lasso(best_alpha)
optimal_lasso.fit(xfat_st, yfat)
print(optimal_lasso.intercept_)
print(optimal_lasso.coef_)
```

```
18.938492063492067
[ 7.57372300e+00 -1.37753930e-01 -0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00  0.00000000e+00  6.41373707e-03
  0.00000000e+00  0.00000000e+00  0.00000000e+00  2.68691580e-04
  0.00000000e+00]
```



다음시간안내

제04강

Basic Methods for Classification