

조주현_20171483

기계학습기초_term Project

목차

Data pre-processing

Correlation 확인

Correlated Data plotting

Linear regression & Model Fitting

Regression 평가지표

Regression Plotting

Discussion

Data pre-processing

1 data																							
	ID	TOWN	TOWNNO	TRACT	LON	LAT	x	y	MEDV	CMEDV	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	1.0	0.0	0.0	2011.0	-70.9550	42.2550	338.73	4679.73	24.0	24.0	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	2.0	0.0	1.0	2021.0	-70.9500	42.2875	339.23	4683.33	21.6	21.6	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	3.0	0.0	1.0	2022.0	-70.9360	42.2830	340.37	4682.80	34.7	34.7	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
3	4.0	0.0	2.0	2031.0	-70.9280	42.2930	341.05	4683.89	33.4	33.4	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	5.0	0.0	2.0	2032.0	-70.9220	42.2980	341.56	4684.44	36.2	36.2	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33
...
501	502.0	0.0	91.0	1801.0	-70.9860	42.2312	336.11	4677.14	22.4	22.4	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99	9.67
502	503.0	0.0	91.0	1802.0	-70.9910	42.2275	335.69	4676.74	20.6	20.6	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90	9.08
503	504.0	0.0	91.0	1803.0	-70.9948	42.2260	335.37	4676.58	23.9	23.9	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90	5.64
504	505.0	0.0	91.0	1804.0	-70.9875	42.2240	335.97	4676.35	22.0	22.0	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45	6.48
505	506.0	0.0	91.0	1805.0	-70.9825	42.2210	336.38	4676.00	11.9	19.0	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90	7.88

506 rows × 23 columns

▶	1 data.isna().sum()	
	ID	0
	TOWN	0
	TOWNNO	0
	TRACT	0
	LON	0
	LAT	0
	x	0
	y	0
	MEDV	0
	CMEDV	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
	dtype: int64	

- 가장 먼저 data를 확인하였다. 총 23개의 feature로 이루어진 506개의 data로 이루어져 있음을 확인할 수 있다.
- 또한 결측치를 확인해 보았을 때 모든 feature의 결측치는 없는 것으로 확인되었다.

Data pre-processing

```
[ ] 1 data.columns
     2 data = data.drop(['x', 'y', 'MEDV', 'ID', 'TOWN'], axis = 1)
```

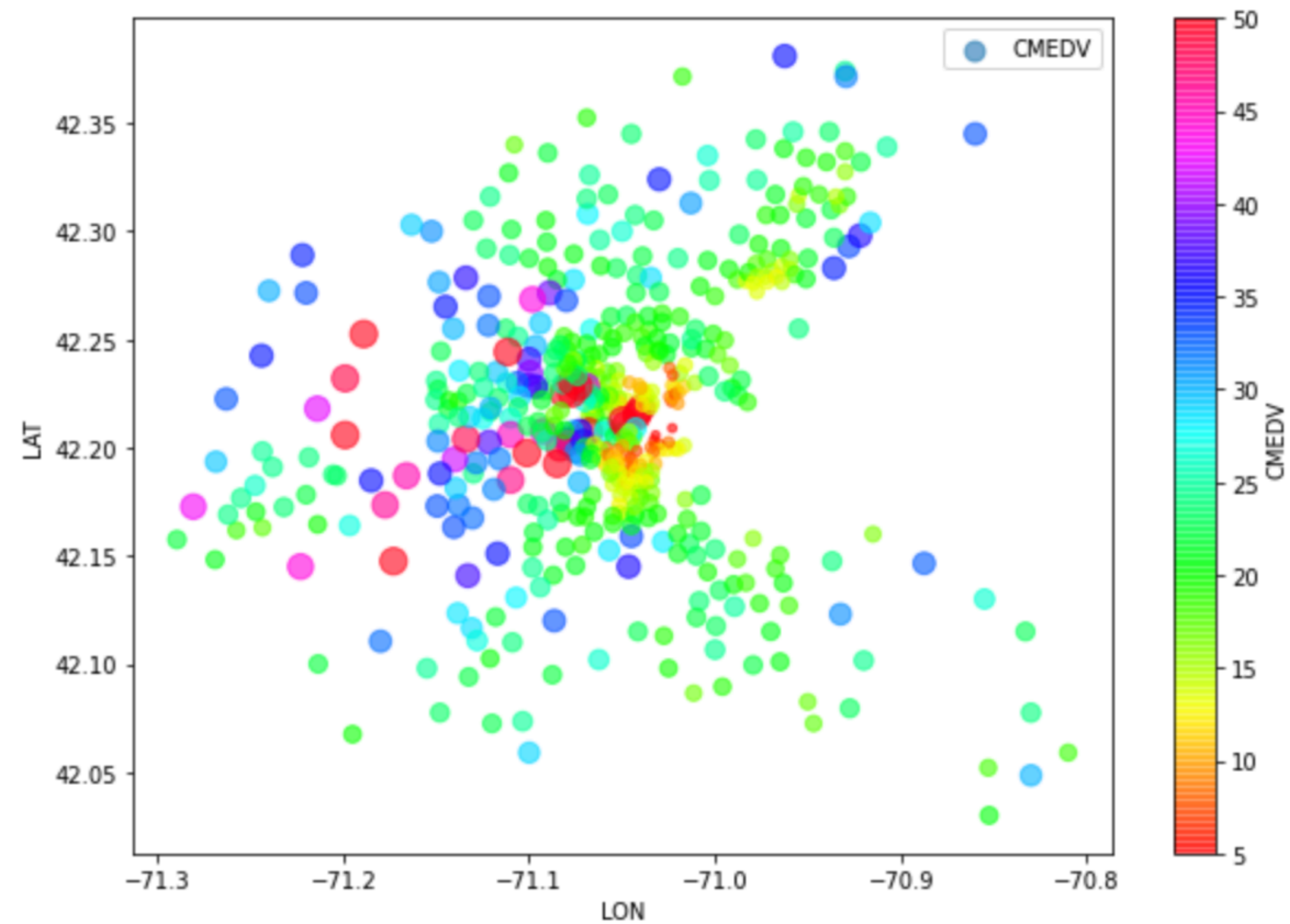
```
[ ] 1 data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 18 columns):
#   Column      Non-Null Count  Dtype
---  -
0    TOWNNO      506 non-null    float64
1    TRACT       506 non-null    float64
2    LON         506 non-null    float64
3    LAT         506 non-null    float64
4    CMEDV       506 non-null    float64
5    CRIM        506 non-null    float64
6    ZN          506 non-null    float64
7    INDUS       506 non-null    float64
8    CHAS        506 non-null    float64
9    NOX         506 non-null    float64
10   RM          506 non-null    float64
11   AGE         506 non-null    float64
12   DIS         506 non-null    float64
13   RAD         506 non-null    float64
14   TAX         506 non-null    float64
15   PTRATIO     506 non-null    float64
16   B           506 non-null    float64
17   LSTAT       506 non-null    float64
dtypes: float64(18)
memory usage: 71.3 KB
```

- 원본 data에서 삭제한 feature는 다음과 같다.
'x', 'y', 'MEDV', 'ID', 'TOWN'
- 삭제 이유는 다음과 같다.
 1. **x, y** : LON, LAT과 겹치는 feature
 2. **MEDV** : CMEDV(target값)과 동일한 feature
 3. **ID, TOWN** : ID는 순서, TOWN은 모두 0으로 동일한 값이므로 학습에 무의미하다.
- data.info() 함수 호출을 통해 정상적으로 삭제되었음을 확인할 수 있다.

Data pre-processing

```
[ ] 1 #위도와 경도를 기준으로 target 분포 확인
2 data.plot(kind='scatter',x="LON", y="LAT", alpha=0.6,
3           s = data['CMEDV']*3, label="CMEDV",figsize=(10,7),
4           c="CMEDV", cmap=plt.get_cmap("hsv"),colorbar=True, sharex=False)
5 plt.legend()
6 plt.show()
7
```

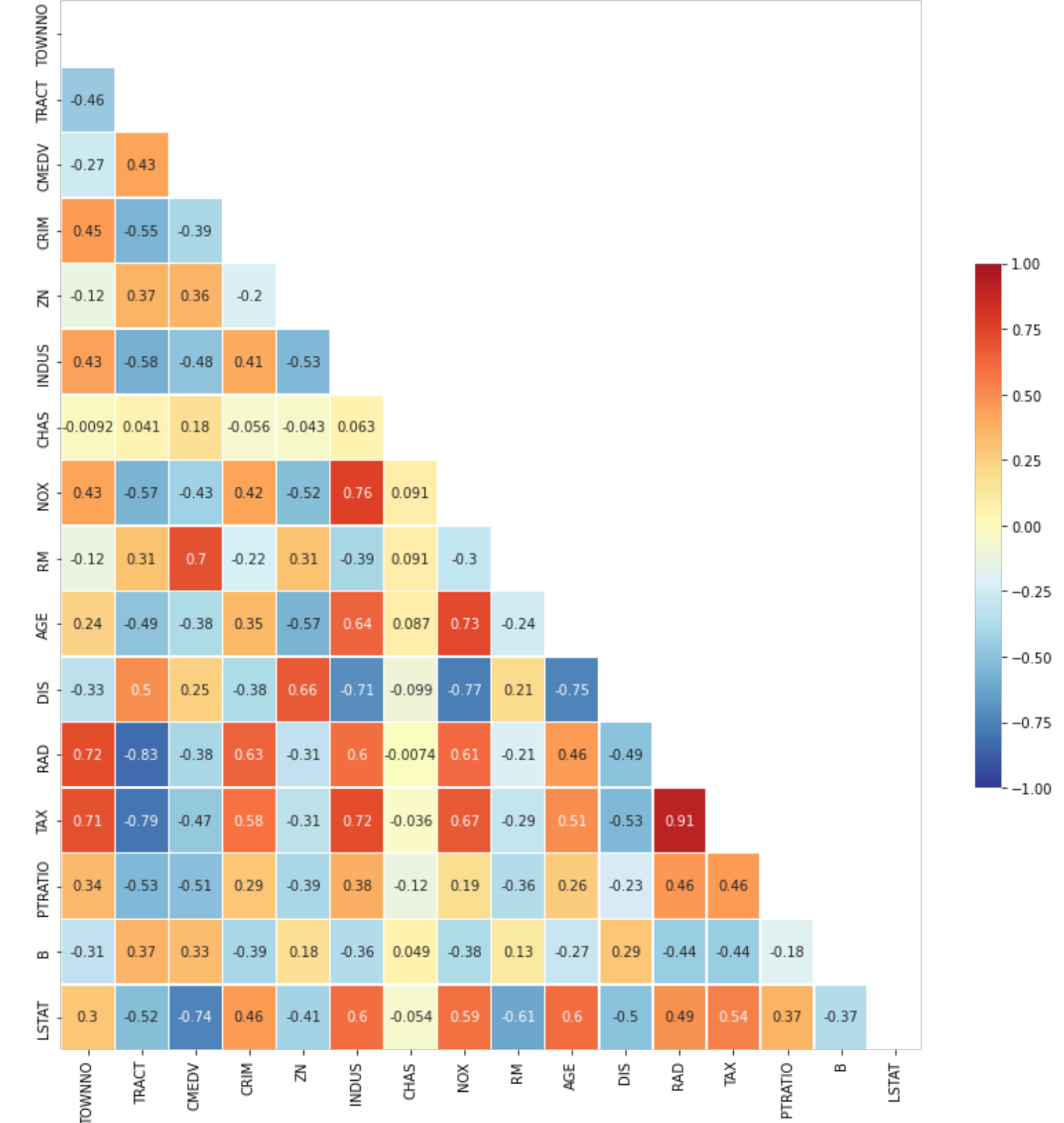


```
[ ] 1 data = data.drop(['LON', 'LAT'], axis = 1)
```

- Data의 LON, LAT을 이용해 target(CMEDV)의 분포를 확인하였다.
- Color map을 통해 CMEDV 값의 크기를 표현하였다.
- 확인 결과 위치 정보로는 유의미한 상관관계를 유추할 수 없다고 판단해 삭제하였다.

Correlation 확인

```
[ ] 1 fig, ax = plt.subplots( figsize=(14,14) )
    2
    3 corr_data = data.corr()
    4
    5 # 삼각형 마스크를 만든다(위 쪽 삼각형에 True, 아래 삼각형에 False)
    6 mask = np.zeros_like(corr_data, dtype=np.bool)
    7 mask[np.triu_indices_from(mask)] = True
    8
    9 # 히트맵을 그린다
   10 sns.heatmap(corr_data,
   11              cmap = 'RdYlBu_r',
   12              annot = True,      # 실제 값을 표시한다
   13              mask=mask,         # 표시하지 않을 마스크 부분을 지정한다
   14              linewidths=.5,    # 경계면 실선으로 구분하기
   15              cbar_kws={"shrink": .5}, # 컬러바 크기 절반으로 줄이기
   16              vmin = -1, vmax = 1  # 컬러바 범위 -1 ~ 1
   17              )
   18 plt.show()
```



- 앞서 제거하지 않은 feature들로 correlation triangle을 그렸다.
하지만 아직 feature의 수가 많아 어떤 feature의 중요도가 높은지 알아보기 힘들다.
- 따라서 **CMEVD**에 대한 **correlation**만 분석할 필요가 있다.

Correlation 확인

```
[ ] 1 corr_data = data.corr().loc[:, 'CMEDV'].abs().sort_values(ascending = False)
    2 plot_data =[]
    3 print(corr_data)
    4
    5 for i in range(6):
    6     plot_data.append(corr_data.index[i])
```

CMEDV	1.000000
LSTAT	0.740836
RM	0.696304
PTRATIO	0.505655
INDUS	0.484754
TAX	0.471979
NOX	0.429302
TRACT	0.428252
CRIM	0.389582
RAD	0.384766
AGE	0.377999
ZN	0.360386
B	0.334861
TOWNNO	0.265134
DIS	0.249315
CHAS	0.175663

Name: CMEDV, dtype: float64

```
[ ] 1 plot_data
```

```
['CMEDV', 'LSTAT', 'RM', 'PTRATIO', 'INDUS', 'TAX']
```

- 제외한 feature외에 모든 feature들의 CMEDV에 대한 correlation data를 계산하는 함수이다.
- 내림차순으로 정렬한 뒤 상위 6개의 feature를 따로 분석하기 위해 저장했다(plot_data).
- 해당 feature는 다음과 같다.
: CMEDV, LSTAT, RM, PTRATIO, INDUS, TAX
- **correlation**이 크다는 것은 상관관계가 높다는 의미이므로 **regression** 단계에서 좋은 **feature**가 될 가능성이 높다.

Correlated Data Plotting

```
[ ] 1 data = data.loc[:, plot_data]
    2 data
```

	CMEDV	LSTAT	RM	PTRATIO	INDUS	TAX
0	24.0	4.98	6.575	15.3	2.31	296.0
1	21.6	9.14	6.421	17.8	7.07	242.0
2	34.7	4.03	7.185	17.8	7.07	242.0
3	33.4	2.94	6.998	18.7	2.18	222.0
4	36.2	5.33	7.147	18.7	2.18	222.0
...
501	22.4	9.67	6.593	21.0	11.93	273.0
502	20.6	9.08	6.120	21.0	11.93	273.0
503	23.9	5.64	6.976	21.0	11.93	273.0
504	22.0	6.48	6.794	21.0	11.93	273.0
505	19.0	7.88	6.030	21.0	11.93	273.0

506 rows × 6 columns

```
[ ] 1 target = data.loc[:, 'CMEDV']
    2 data = data.drop(['CMEDV'], axis = 1)
    3 data
```

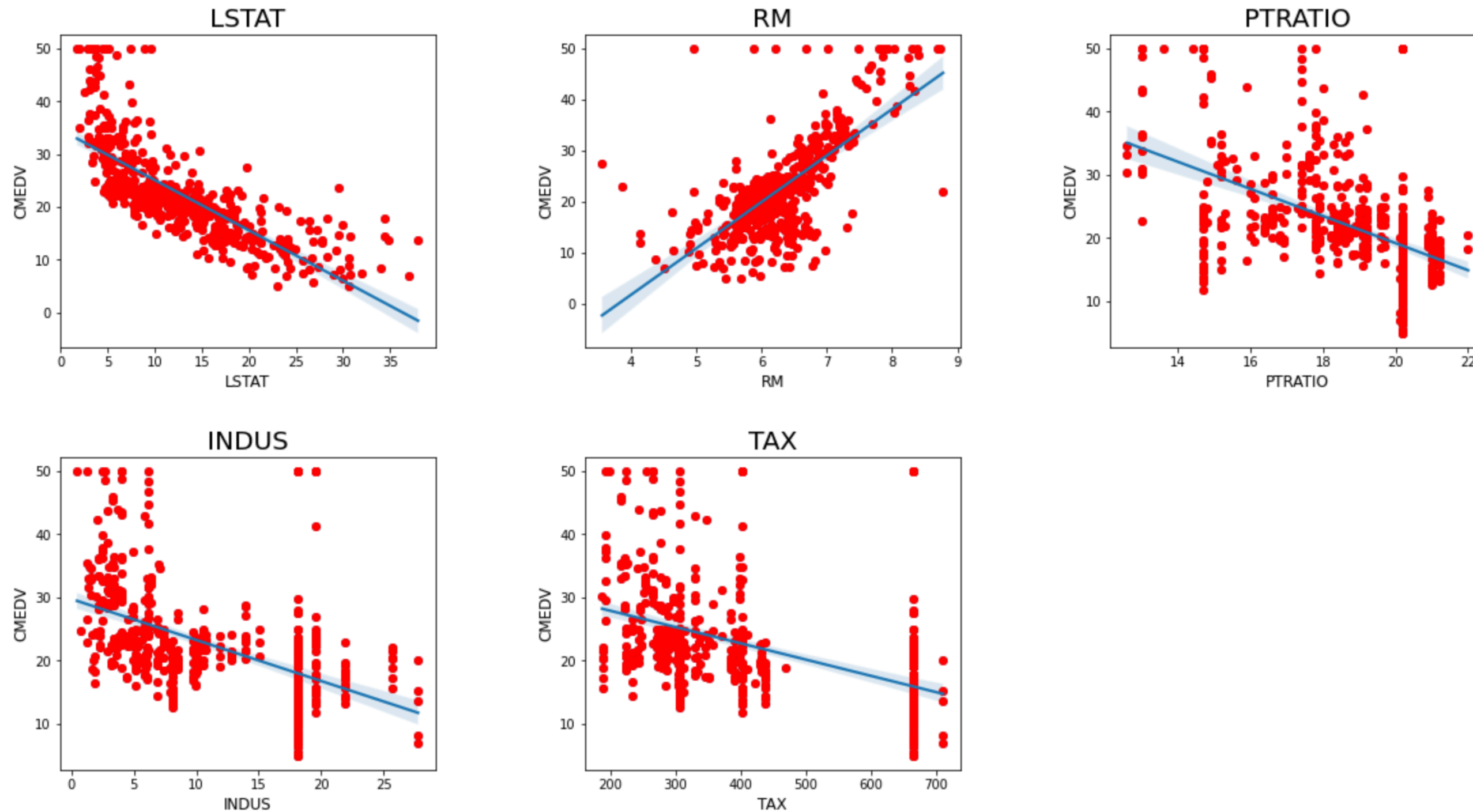
	LSTAT	RM	PTRATIO	INDUS	TAX
0	4.98	6.575	15.3	2.31	296.0
1	9.14	6.421	17.8	7.07	242.0
2	4.03	7.185	17.8	7.07	242.0
3	2.94	6.998	18.7	2.18	222.0
4	5.33	7.147	18.7	2.18	222.0
...
501	9.67	6.593	21.0	11.93	273.0
502	9.08	6.120	21.0	11.93	273.0
503	5.64	6.976	21.0	11.93	273.0
504	6.48	6.794	21.0	11.93	273.0
505	7.88	6.030	21.0	11.93	273.0

506 rows × 5 columns

- Correlation이 높은 6개의 feature를 따로 저장하였다. (CMEDV 포함)
- 그 중 CMEDV는 DataFrame target으로 따로 저장하였다.

Correlated Data Plotting

```
[ ] 1 plt.figure(figsize=(20,10))
2 plt.subplots_adjust(left=0.125, bottom=0.1, right=0.9, top=0.9, wspace=0.4, hspace=0.35)
3
4 for count in range(0, len(data.columns)):
5     plt.subplot(2, 3, count+1)
6     plt.plot(data.iloc[:,count], target, 'ro')
7     plt.xlabel(data.columns[count], fontsize = 12)
8     plt.ylabel('CMEDV', fontsize = 12)
9     sns.regplot(x=data.iloc[:,count], y=target)
10    plt.title(data.columns[count], fontsize = 20)
11
12 plt.show()
```



- 5개의 feature를 각각 가로축에, CMEDV를 세로축으로 plotting한 뒤 추세선을 그렸다.
- LSTAT과 RM은 선형적인 분포를 보이지만 나머지 PTRATIO, INDUS와 TAX는 선형적인 분포를 보인다고 하기 힘들다.
- 따라서 **Linear regression**을 이용하기 적합한 feature는 **LSTAT과 RM**이다.
- 그 외 3개의 feature는 Linear regression이 아닌 다른 model의 feature로 사용하는 것이 적합해 보인다.

다중 선형 회귀(Multiple Linear Regression)

- **Linear Regression**

- 데이터를 가장 잘 설명하는 직선을 찾아 내는 것이다.
- 가장 적합한 기울기(가중치, 계수)와 y절편(편향, bias)를 찾아내는 것이 목적이다.

- 이번 프로젝트에서는 2개의 feature(**LSTAT, RM**)을 이용하는 **단변량 다중 선형회귀**를 사용한다.

- Linear Regression은 Least Square 식을 이용해 값을 근사해 직선 함수를 얻는다.

< 다중 선형 회귀 식 >

$$\hat{y} = w_1x_1 + w_2x_2 + b$$

\hat{y} : 예측값

x : 독립변수(특성, feature등으로 불림)

w : 기울기 또는 계수(coefficient, 가중치)

b : y절편 또는 편향(offset, bias)

< Least Square 식 >

$$\sum_i (y_i - \hat{y}_i)^2$$

\hat{y} : 예측값

y : 실제값

다중 선형 회귀(Multiple Linear Regression)

$$\hat{y} = w_1x_1 + w_2x_2 + b$$

위 식을 선형방정식으로 쓰게 되면 아래와 같이 표현할 수 있다.

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{21} \\ 1 & x_{12} & x_{22} \\ \vdots & \vdots & \vdots \\ 1 & x_{1n} & x_{2n} \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} \quad (w_1, \text{과 } w_2 \text{를 계산의 편의상 } b_1, b_2 \text{로 바꾸었다.})$$

위 식을 간단히 표현하면

$$Y = Xb$$

가 됨을 알 수 있고 이 matrix에서 least square은 아래와 같다.

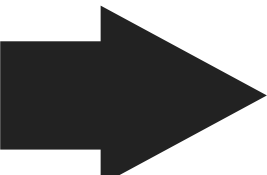
$$\text{Least Square : } X'Xb = X'Y$$

다중 선형 회귀(Multiple Linear Regression)

Least Square : $X'Xb = X'Y$

위 식을 과제에 맞추어 식을 풀어보면 아래와 같다.

$$X'X = \begin{bmatrix} n & \sum x_1 & \sum x_2 \\ \sum x_1 & \sum x_1^2 & \sum x_1 \sum x_2 \\ \sum x_2 & \sum x_1 \sum x_2 & \sum x_2^2 \end{bmatrix} \quad XY = \begin{bmatrix} \sum y \\ \sum x_1 y \\ \sum x_2 y \end{bmatrix}$$



$$\begin{bmatrix} n & \sum x_1 & \sum x_2 \\ \sum x_1 & \sum x_1^2 & \sum x_1 \sum x_2 \\ \sum x_2 & \sum x_1 \sum x_2 & \sum x_2^2 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} \sum y \\ \sum x_1 y \\ \sum x_2 y \end{bmatrix}$$

위 식을 전개해보면 $nb_0 + b_1 \sum x_1 + b_2 \sum x_2 = \sum y$ 와 같은 식이 나오는데
이는 결국 단순 선형 회귀에서 최소 제곱의 정규 방정식을 푸는 것과 같게 나온다.

Linear Regression & Model Fitting

```
[ ] 1 regression_feature = data.loc[:,['LSTAT', 'RM']]
```

```
[ ] 1 from sklearn.model_selection import train_test_split
2
3 train_feature, test_feature, train_target, test_target = train_test_split(
4     regression_feature, target, test_size = 0.2, random_state = 1)
```

```
[ ] 1 from sklearn.linear_model import LinearRegression
2 reg_model=LinearRegression()
3 reg_model.fit(train_feature, train_target)
4
5 print("회귀계수(기울기):", np.round(reg_model.coef_, 1))
6 print("상수항(절편):", np.round(reg_model.intercept_, 1))
7
8 train_predict=reg_model.predict(train_feature)
9 test_predict=reg_model.predict(test_feature)
10
11 print("예측: ", test_predict[:5])
12 print("정답: ", list(test_target[:5]))
```

회귀계수(기울기): [-0.7 4.5]

상수항(절편): 2.8

예측: [28.61741572 28.16122705 17.44059215 23.86726561 20.15532326]

정답: [28.2, 23.9, 16.6, 22.0, 20.8]

- Linear regression에 사용할 2개의 feature(LSTAT과 RM)를 저장한 뒤 LinearRegression() model에 fitting하는 함수이다.
- LSTAT에 대한 기울기는 -0.7, RM에 대한 기울기는 4.5이고, y절편은 2.8이다. 해당 식을 표현하면 아래와 같다.

$$\hat{y} = -0.7x_1 + 4.5x_2 + 2.8$$

x1 : LSTAT

x2 : RM

- 예측값과 실제값이 유사함을 볼 수 있다.

Regression 평가지표_RMSE

```
[50] 1 from sklearn.metrics import mean_squared_error
      2
      3 train_mse=mean_squared_error(train_target, train_predict)
      4 train_rmse=np.sqrt(train_mse)
      5 print("train RMSE: {:.4f}".format(train_rmse))
      6
      7 test_mse=mean_squared_error(test_target, test_predict)
      8 test_rmse=np.sqrt(test_mse)
      9 print("test RMSE: {:.4f}".format(test_rmse))
     10
```

```
train RMSE: 5.4207
test RMSE: 5.6670
```

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

- RMSE는 MSE의 제곱근값으로 큰 오차에 대해 페널티를 주어 극단적이지 않은 오차값을 보여준다.
- RMSE가 0에 가까울수록 원본에 가까운 예측값을 보인 것이므로 해당 모델의 예측값은 꽤 정확한 것으로 판단된다.

Regression 평가지표_RMSLE

```
[51] 1 from sklearn.metrics import mean_squared_log_error
      2
      *3 for i in range(len(train_predict)):
      4     if train_predict[i] < 0:
      5         train_predict[i] = 0
      6
      7 train_msle=mean_squared_log_error(train_target, train_predict)
      8 test_msle=mean_squared_log_error(test_target, test_predict)
      9
     10 train_rmsle = np.sqrt(train_msle)
     11 test_rmsle = np.sqrt(test_msle)
     12
     13 print("train RMLSE: {:.4f}".format(train_rmsle))
     14 print("test RMLSE: {:.4f}".format(test_rmsle))
     15
```

```
train RMLSE: 0.3370
test RMLSE: 0.2803
```

$$RMSLE = \sqrt{\frac{1}{N} \sum_{i=1}^N (\log(p_i + 1) - \log(a_i + 1))^2}$$

$p = Predicted, a = Actual$

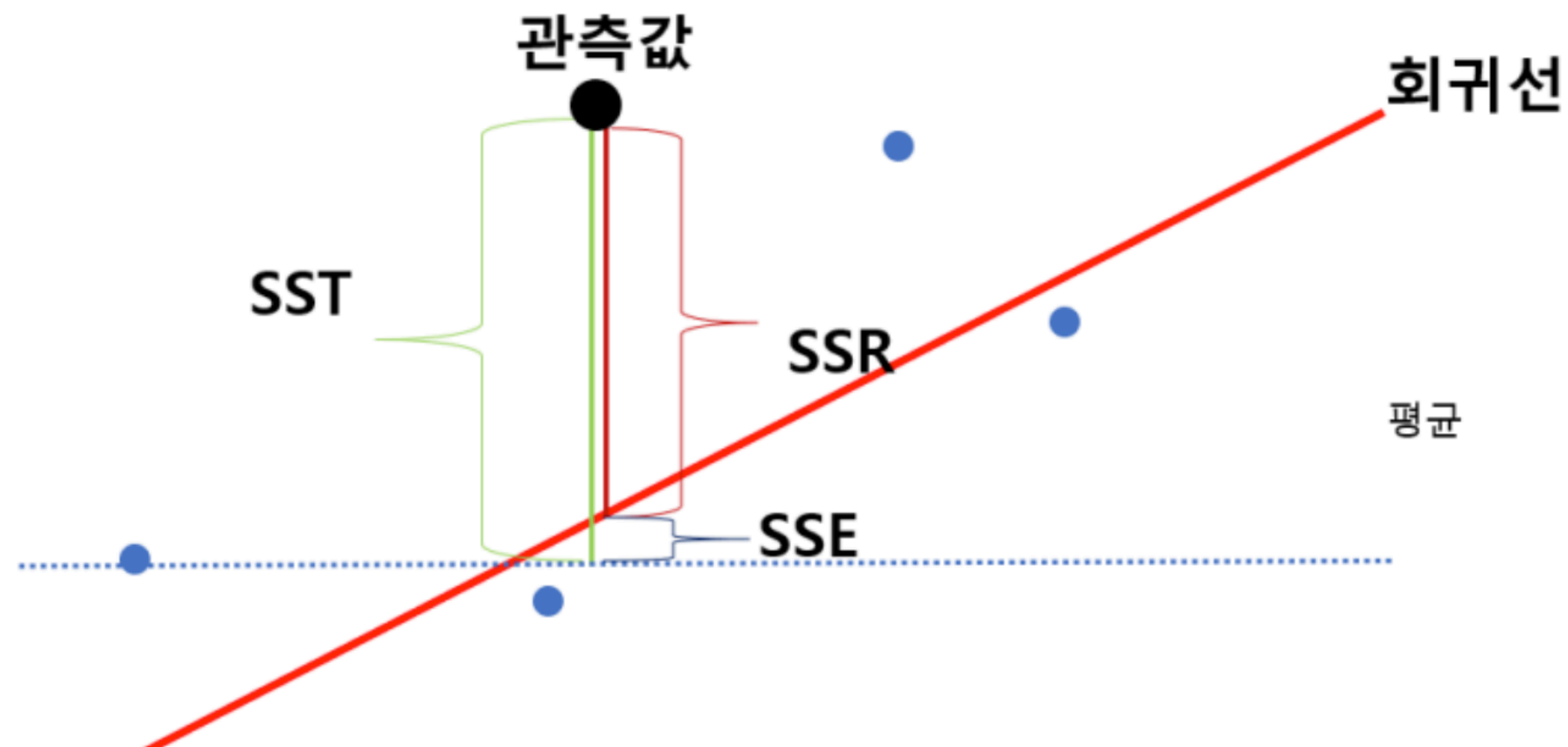
- RMSLE는 RMSE에 log를 씌운 값이다. 이 값은 RMSE와 비교해 3가지 차이점을 보인다.
 1. 아웃라이어에 강건하다(robust)
 2. 상대적인 오차를 측정해준다.
 3. Under Estimation에 큰 페널티를 부여한다.
: 예측값이 실제보다 클 때보다 작을 때 더 큰 페널티를 부여한다.
- **Under Estimation**에 큰 페널티가 부여됨을 알았을 때 **train**에서 실제보다 작게 예측한 데이터가 있음을 유추할 수 있다.

* 예측값에 음수인 값이 있어 log 계산에 있어 오류가 발생함. 평가지표에 대한 설명이 주 목적이기에 0으로 바꾼 뒤 코드를 진행함.

Regression 평가지표_R2 score

```
[56] 1 from sklearn.metrics import r2_score
      2
      3 train_r2_score = r2_score(train_target, train_predict)
      4 test_r2_score = r2_score(test_target, test_predict)
      5
      6 print("train R2 score : {:.4f}".format(train_r2_score))
      7 print("test R2 score : {:.4f}".format(test_r2_score))
      8
```

```
train R2 score : 0.6347
test R2 score : 0.6750
```



$$R^2 = \frac{SSE}{SST} = 1 - \frac{SSR}{SST}$$

SSE: Explaine Sum of Squares,
예측값에서 예측값의 평균을 뺀 값(의 제곱의 총 합)

SST: Total Sum of Squares
실제값에서 실제값의 평균을 뺀 값(의 제곱의 총 합)

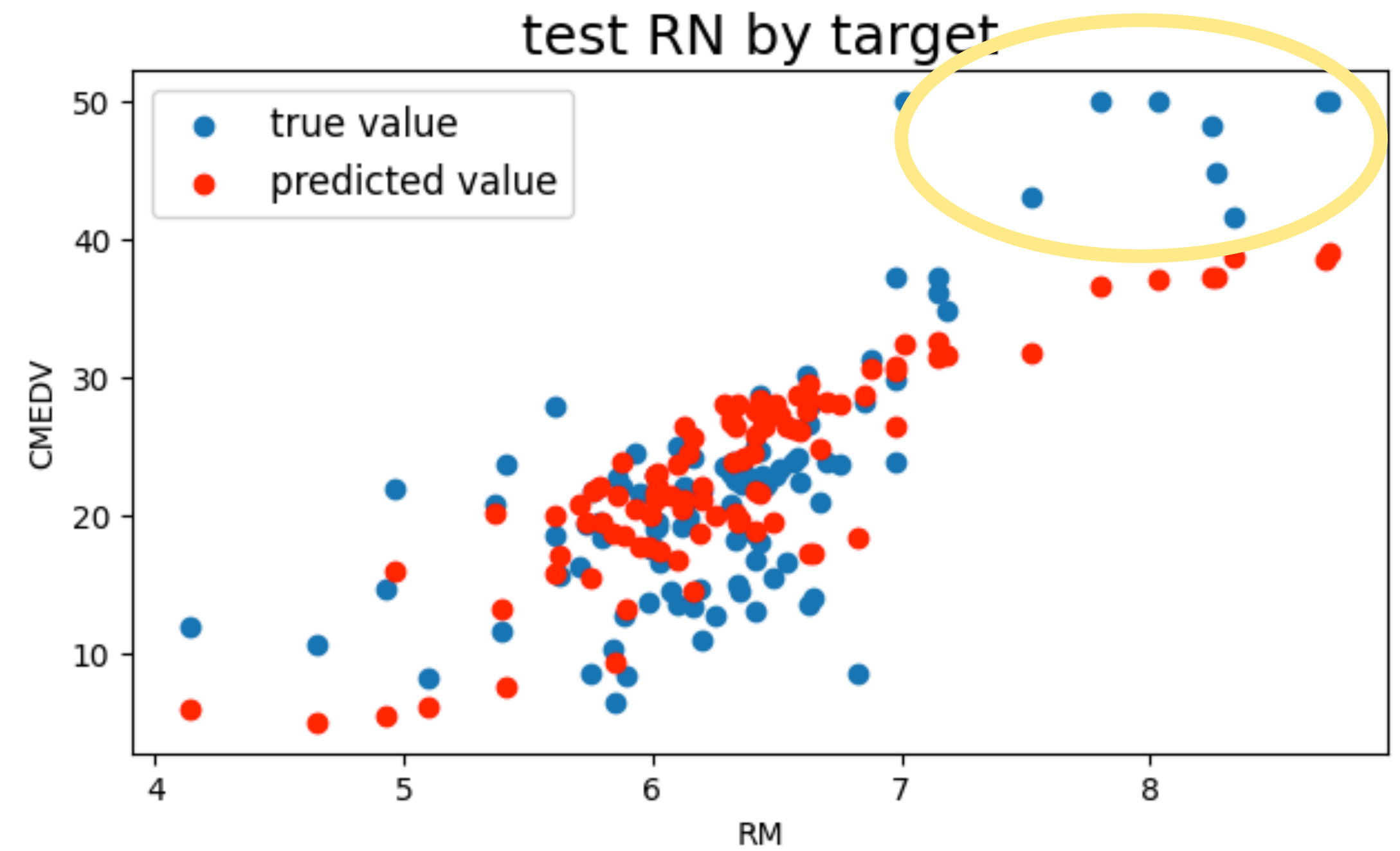
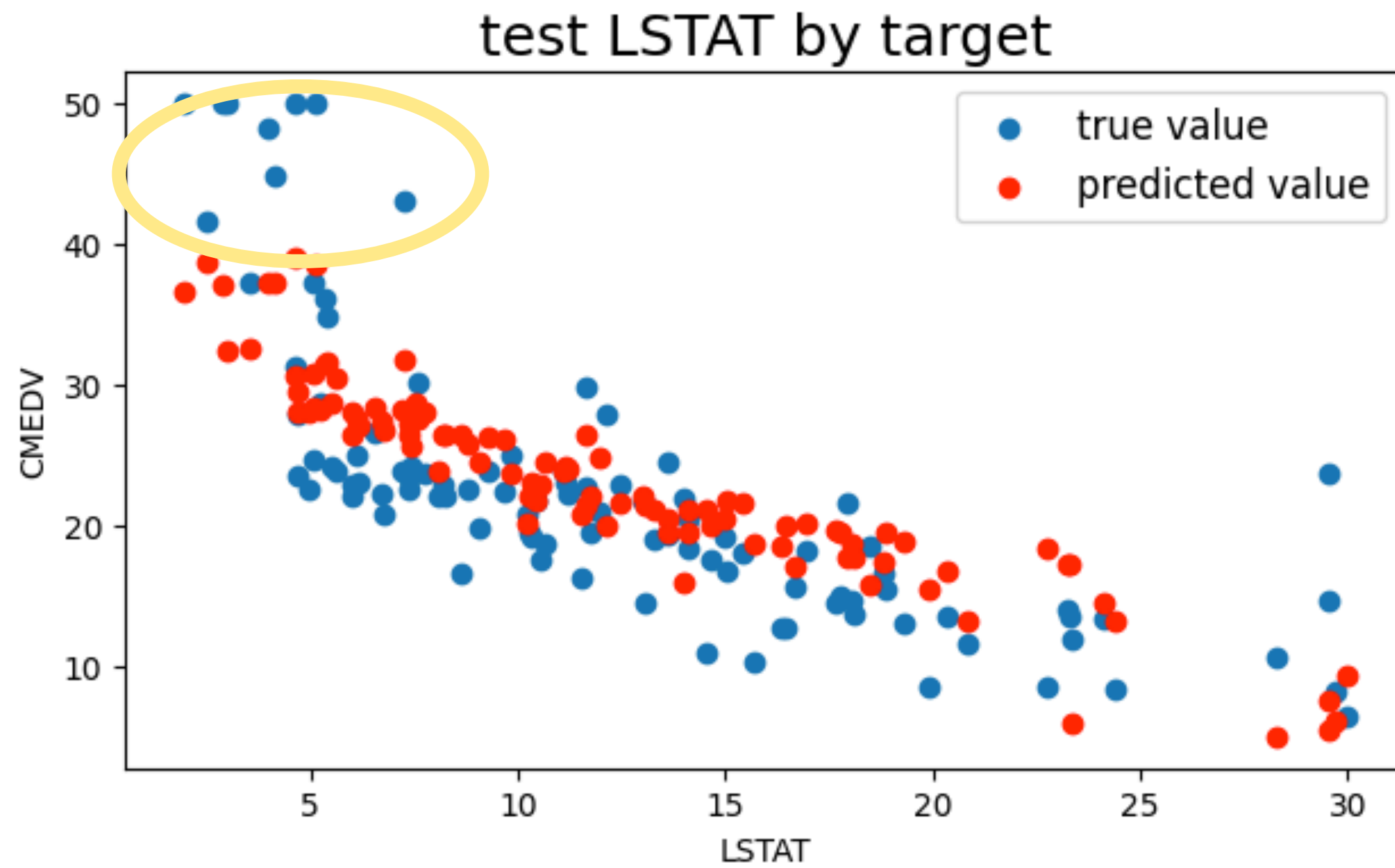
SSR: Residual Sum of Squares
오차(의 제곱의 합)

R2 score은 결정계수라고 불리며 회귀모델에서 독립변수(x)가 종속변수(y)를 얼마나 잘 설명하는지를 나타내는 값이다.

왼쪽 그림은 SSE, SST, SSR을 시각화 한 것이다.

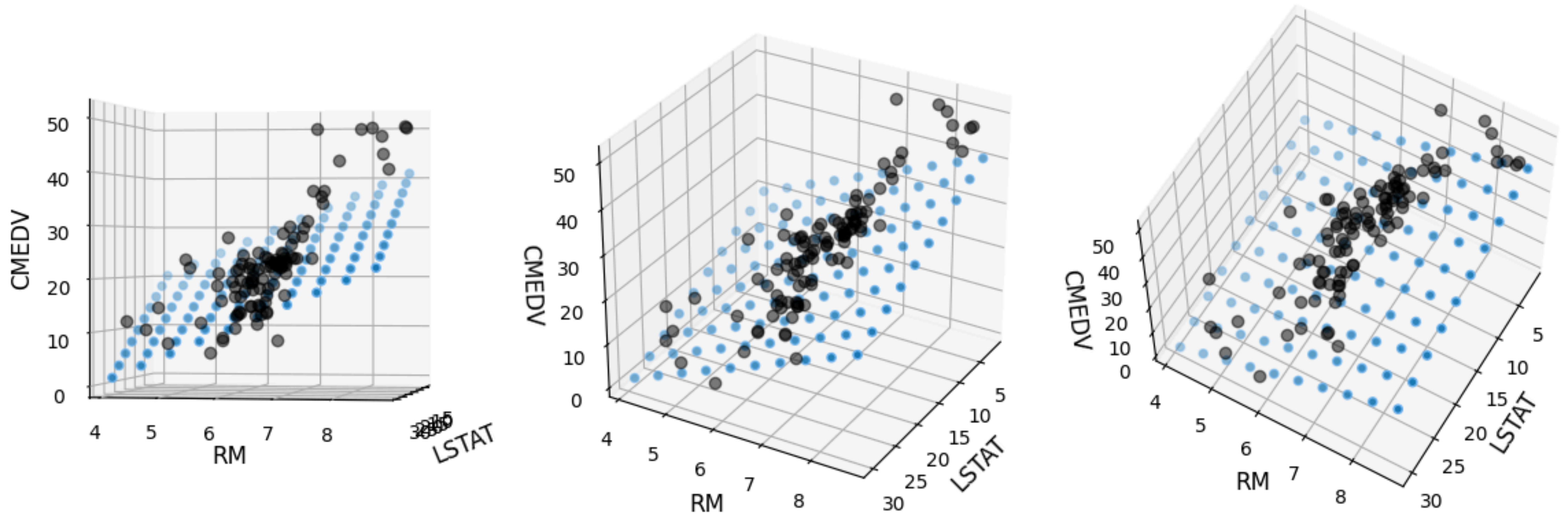
R2 score 값에 대한 척도는 분야별로 상이하기 때문에, 다른 모델을 생성해 비교하는데에 사용하는 상대적인 척도가 될 것이다.

Regression Plotting



- Target인 CMEDV를 각각 LSTAT과 RM에 대해 plotting한 그래프이다. 파란색이 실제값, 빨간색이 예측값이다.
- 대체적으로 유사한 분포를 나타내지만 LSTAT의 좌측 상단, RM의 우측 상단의 outlier의 경우는 정확하게 예측하지 못하고 있다. 이는 Linear Regression model의 한계로, outlier에 대한 예측은 약함을 알 수 있다.
- **Linear Regression**이 아닌 **Polynomial Regression** 모델을 사용하면 어느정도 해결 할 수 있다고 판단된다.

Regression Plotting



- CMEDV를 z축, RM과 LSTAT을 x, y축으로 하는 3차원 그래프를 plot 했다. 파란색으로 scatter된 면이 linear regression으로 예측한 평면, 검은색 점들이 실제 값들이다.
- 앞서 2차원 그래프에서 본 것 과 마찬가지로 **LSTAT**이 작고, **RM**이 클 때 예측값이 빛나가는 것을 확인할 수 있다.

Discussion

1. Model의 한계

회귀 분석의 가정은 다음과 같다.

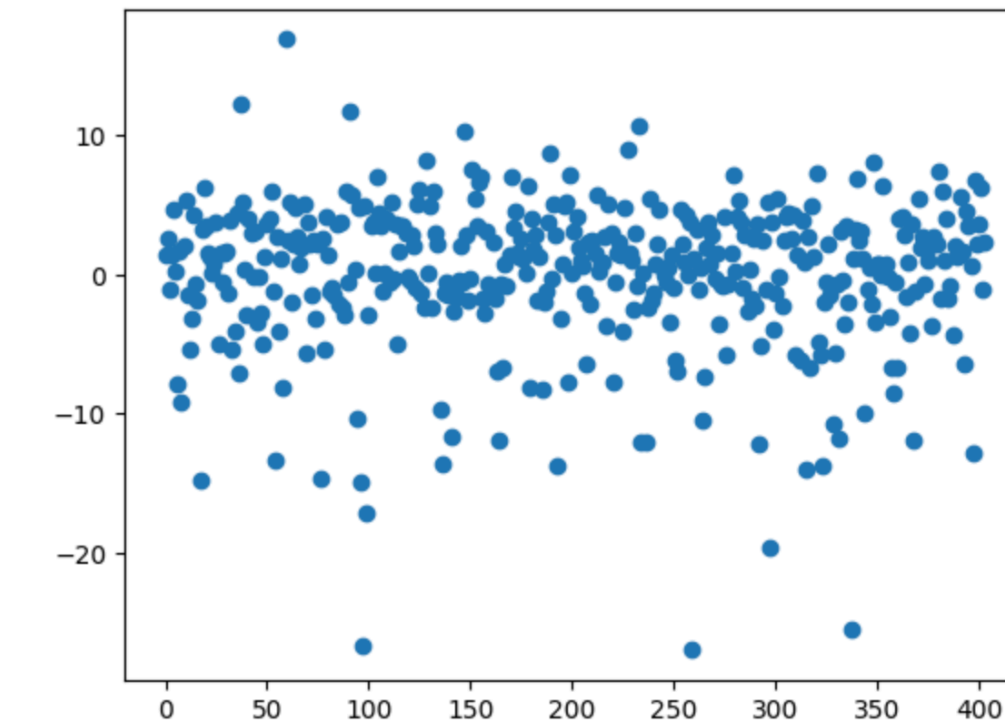
- 오차항의 평균이 0이고 분산이 일정한 정규분포를 갖는다.
- 독립변수와 종속변수는 선형관계이다.
- 오차항은 자기 상관성이 없다.
- 데이터에 아웃라이어가 없다.
- 독립변수와 오차항은 서로 독립이다.
- 독립변수 간에서는 서로 선형적으로 독립이다.

앞서 확인한 correlation triangle을 통해 LSTAT과 RM 사이의 correlation이 0.3 정도임을 확인할 수 있었다.

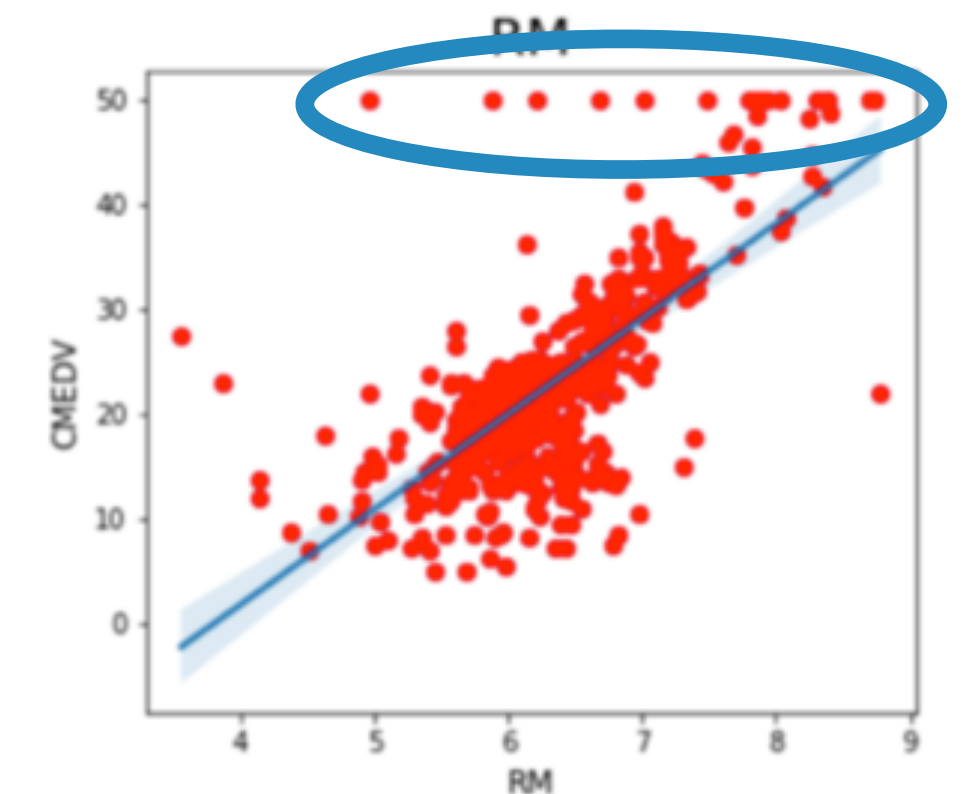
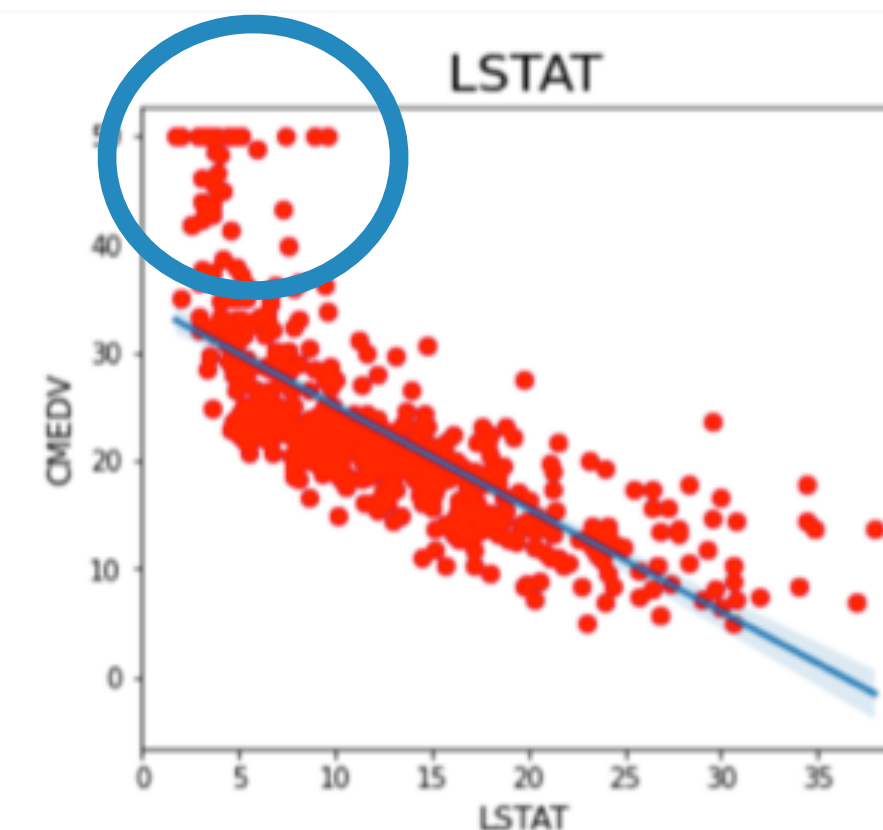
```
[39] 1 dif = train_predict - train_target

1 print("dif mean = {:.4f} \ndif variation = {:.4f}".format(dif.mean(), dif.var()))
2 plt.scatter(range(len(dif)), dif)

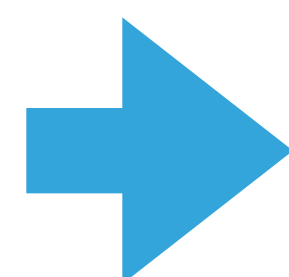
dif mean = 0.013021
dif variation = 29.456789
<matplotlib.collections.PathCollection at 0x7fbf7963ec50>
```



정규분포를 따르지 않는다.



Outlier가 존재하고, 독립변수와 종속변수가 완전히 선형적이라고할 수 없다.



위 가정을 따르지 않을 시 실제 데이터를 정확히 반영하지 못한다.
따라서 다른 방법들을 사용하는 것이 바람직하다.

Discussion

1. Model의 한계

앞서 살펴본 가정에 오류가 있을 때 시도할 수 있는 방법은 다음과 같다.

오차항의 평균이 0이고 분산이 일정한 정규분포를 갖는다

- Generalized Linear Model(GLM)
: GLM - 종속변수에 적절한 함수를 적용하는 회귀분석 방식

독립변수와 종속변수는 선형관계이다.

- Polynomial Regression, Generalized Additive Model(GAM)
: Polynomial Regression - 다항함수를 이용해 회귀하는 분석 방법
: GAM - 독립변수를 그대로 사용하지 않고, 다른 함수의 선형 결합으로 표현

데이터에 아웃라이어가 없다.

- Robust Regression, Quantile Regression
: Robust Regression - Least squares를 최소화 하는 대신
오차의 절대값의 합을 최소화 하는 기법
: Quantile Regression - 종속 변수의 평균값이 아닌 특정 분위값을 추정하는 기법

독립변수 간에서는 서로 선형적으로 독립이다.

- Principal Component Regression(PCR)
: PCR - 독립변수들의 주성분을 추출해 regression에 이용
주성분은 서로 직교하기 때문에 독립성이 보장된다.

Discussion

2. Data 전처리의 부족

표준화(Standardization)과 정규화(Normalization)

- 표준화

서로 다른 단위의 feature 비교시 평균을 0, 표준편차를 1로 변환시키는 작업

⇒ 서로 다른 척도로 측정된 변수들을 동일 선상에 놓고 비교하는 것은 일종의 편의를 낳게된다.

- 정규화

모든 변수들을 동일한 범위를 가지도록 정규화 하는 것

⇒ 표준화와 마찬가지로 변수간 값의 차이에 따른 영향력을 규제한다.

- 예시

연령과 소득을 포함하는 데이터가 있을 때 연령의 범위는 소득의 그것보다 훨씬 작을 것이다.

⇒ 소득 변수는 결과에 더 큰 영향을 끼칠 것이다. 만약 범위를 같게 한다면 동일한 영향력을 갖게 할 수 있을 것이다.

Discussion

결론

Regression이라 하면 Linear Regression만 생각하기 쉽지만, 다른 모델들도 많이 존재한다.

결국, 데이터의 특징이나, 모델링 목적에 따라 적절한 모델을 선택하는 것이 중요하다.

- 탐사 분석 과정을 통해 데이터의 특징을 파악
- 이후 모델링 결과가 정말 본인의 가정과 맞는지 확인하는 검정

하지만, 모든 가정을 따져가며 완벽한 회귀모델을 만드는 것은 불가능에 가깝다.

따라서 완벽한 모델이 아닌 목적에 맞게 사용하기에 괜찮은 정도의 모델을 만드는 것이 중요하다.

All models are wrong, but some are Useful

- George Box