Boston House Price

- 'boston.csv' (506 * 14)
  - CRIM: 범죄 발생률
  - ZN: 25,000평방피트를 초과하는 거주 지역의 비율
  - INDUS: 비소매 상업 지역 비율
  - CHAS: 찰스강 인접 여부(강주변 1, 아니면 0)
  - NOX: 일산화질소 농도
  - RM: 거주 목적 방 개수
  - AGE: 1940년 이전 건축된 주택 비율
  - DIS: 보스턴 5대 고용지역까지 거리
  - RAD: 고속도로 접근성
  - TAX: 10,000달러당 재산세율
  - PTRATIO: 교사와 학생 수 비율
  - B: 흑인 거주 비율
  - LSTAT: 하위 계층 비율
  - PRICE: 주택 가격 -> 'y'

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

I. Data Load

- 'boston.csv' Github에서 읽어오기

```
import pandas as pd

url = 'https://raw.githubusercontent.com/rusita-ai/pyData/master/boston.csv'
DF = pd.read_csv(url)

DF.info()
```

```
<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  PRICE    506 non-null    float64
dtypes: float64(14)
memory usage: 55.5 KB
```

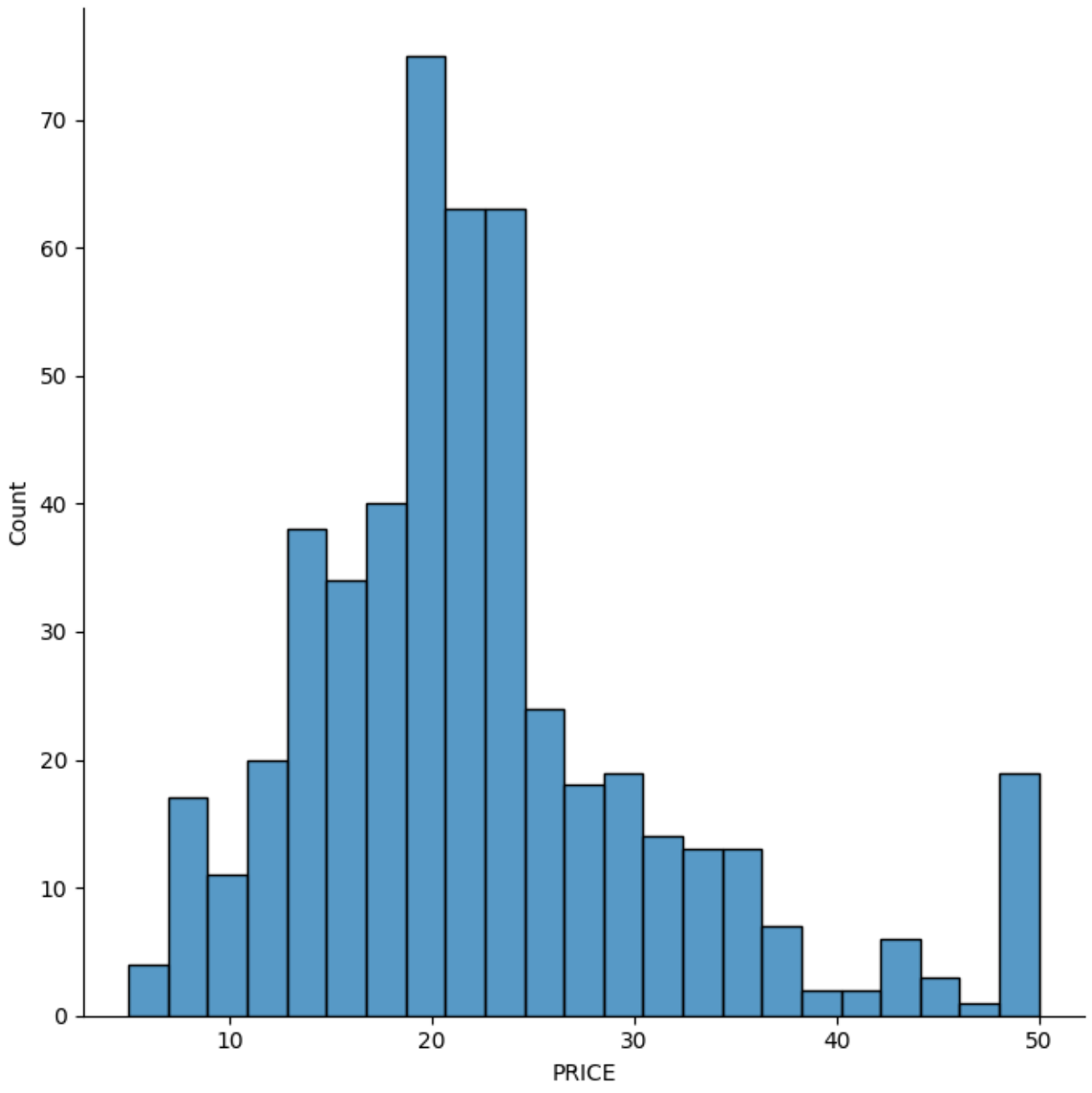
```
DF.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTA
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	4.9
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	9.1
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	4.0
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.9
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.3

II. 탐색적 데이터 분석(EDA)

1) 'PRICE' 분포

```
import matplotlib.pyplot as plt
import seaborn as sns

sns.displot(x = 'PRICE',
            data = DF,
            height = 7)
plt.show()
```



- 'PRICE' 평균

```
DF['PRICE'].mean()
```

```
22.532806324110677
```

2) 상관계수 ('r')

- 전체 데이터

```
DF.corr()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	D
CRIM	1.000000	-0.200469	0.406583	-0.055892	0.420972	-0.219247	0.352734	-0.3796
ZN	-0.200469	1.000000	-0.533828	-0.042697	-0.516604	0.311991	-0.569537	0.6644
INDUS	0.406583	-0.533828	1.000000	0.062938	0.763651	-0.391676	0.644779	-0.7080
CHAS	-0.055892	-0.042697	0.062938	1.000000	0.091203	0.091251	0.086518	-0.0991
NOX	0.420972	-0.516604	0.763651	0.091203	1.000000	-0.302188	0.731470	-0.7692
RM	-0.219247	0.311991	-0.391676	0.091251	-0.302188	1.000000	-0.240265	0.2052
AGE	0.352734	-0.569537	0.644779	0.086518	0.731470	-0.240265	1.000000	-0.7478
DIS	-0.379670	0.664408	-0.708027	-0.099176	-0.769230	0.205246	-0.747881	1.0000
RAD	0.625505	-0.311948	0.595129	-0.007368	0.611441	-0.209847	0.456022	-0.4945
TAX	0.582764	-0.314563	0.720760	-0.035587	0.668023	-0.292048	0.506456	-0.5344
B	0.385064	-0.175520	0.356977	0.048788	0.380051	-0.128069	0.273534	-0.2915

- HeatMap Visualization

```
plt.figure(figsize = (12, 12))
sns.heatmap(DF.corr(),
            annot = True,
            cmap = 'Blues',
            fmt = '.2f')
plt.show()
```

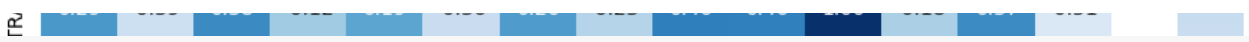
3) 연속형 변수 상관계수 ('r')

- 'PRICE'와 나머지 변수 간 상관계수 내림차순 정렬

```
DF.corr().loc[:'LSTAT', 'PRICE'].abs().sort_values(ascending = False)
```

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

- 6위까지 추출



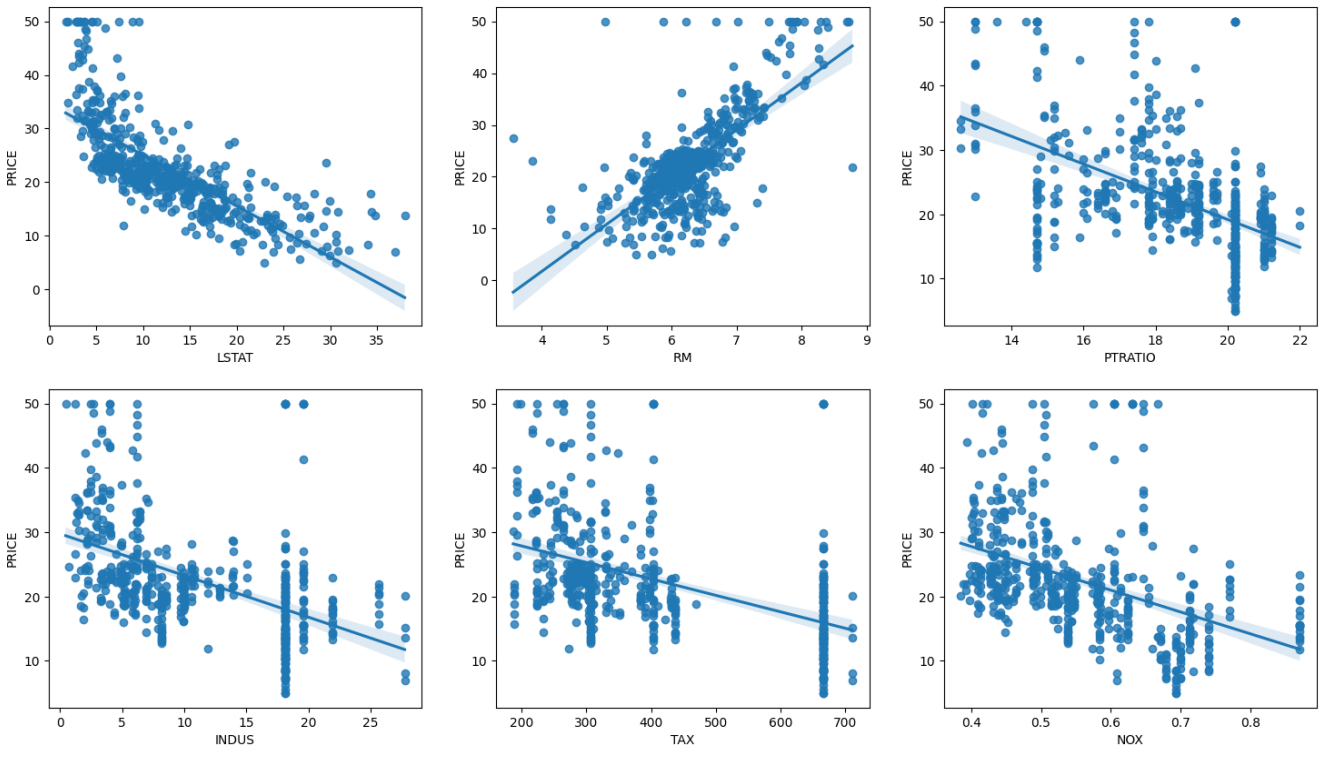
```
DF_reg = DF.loc[:, ['PRICE', 'LSTAT', 'RM', 'PTRATIO', 'INDUS', 'TAX', 'NOX']].copy()

DF_reg.head()
```

	PRICE	LSTAT	RM	PTRATIO	INDUS	TAX	NOX
0	24.0	4.98	6.575	15.3	2.31	296.0	0.538
1	21.6	9.14	6.421	17.8	7.07	242.0	0.469
2	34.7	4.03	7.185	17.8	7.07	242.0	0.469
3	33.4	2.94	6.998	18.7	2.18	222.0	0.458
4	36.2	5.33	7.147	18.7	2.18	222.0	0.458

4) 'PRICE'와 6위까지 변수간 시각화

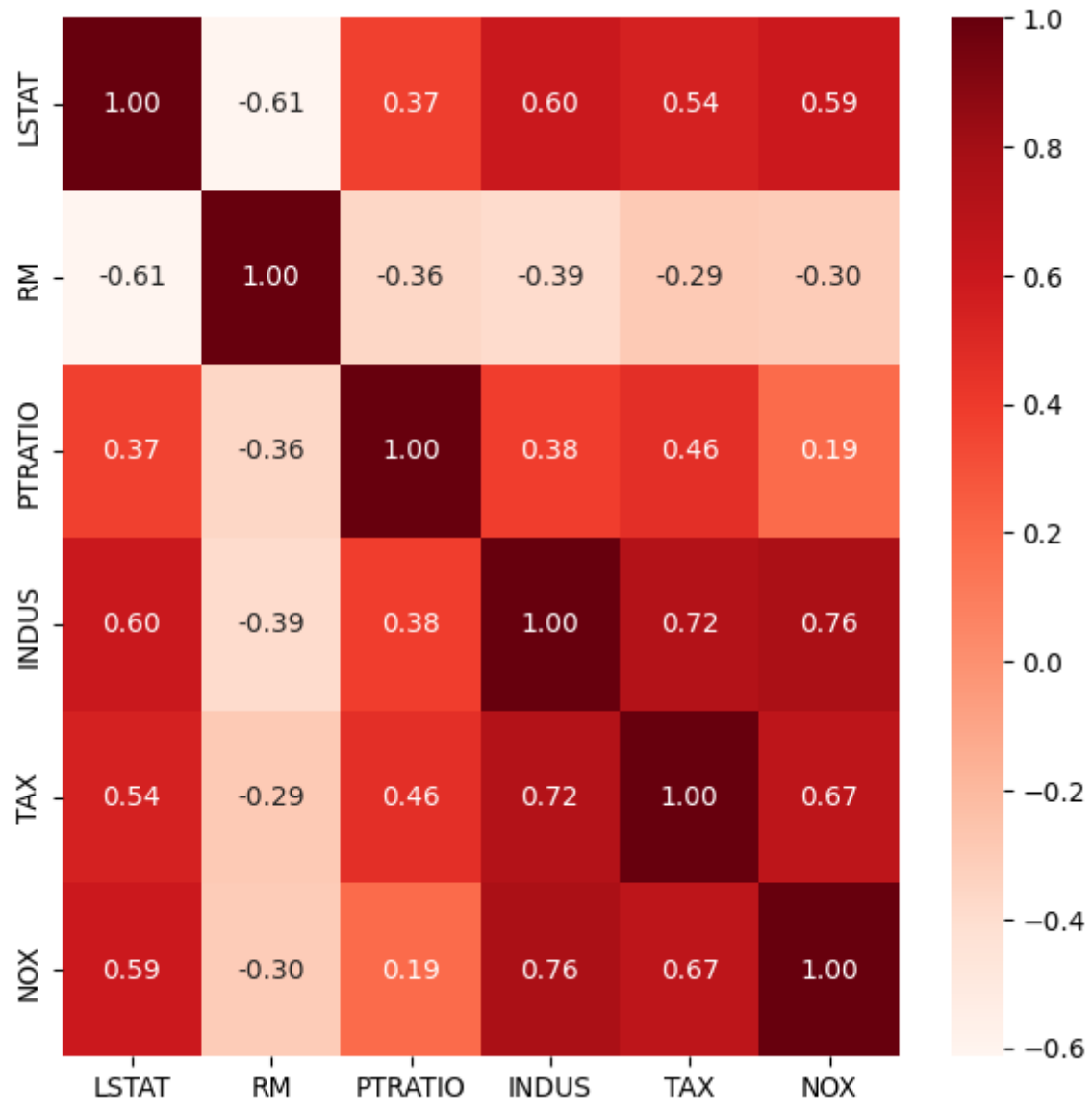
- 회귀모델 시각화

```
plt.figure(figsize = (18, 10))

for index, col_names in enumerate(DF_reg.columns[1:]):
  ax1 = plt.subplot(2, 3, index + 1)
  sns.regplot(x = col_names,
              y = DF_reg.columns[0],
              data = DF_reg,
              ax = ax1)

plt.show()
```



- 연속형 변수간 상관관계 ('r') 확인

```
plt.figure(figsize = (7, 7))
sns.heatmap(DF_reg.loc[:,'LSTAT':].corr(),
            annot = True,
            cmap = 'Reds',
            fmt = '.2f')
plt.show()
```



# # #

End Of Document

# # #