import libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
```

```
<frozen importlib._bootstrap>:219: RuntimeWarning: numpy.ufunc size changed, may indicate binary incomp
atibility. Expected 192 from C header, got 216 from PyObject
<frozen importlib._bootstrap>:219: RuntimeWarning: numpy.ufunc size changed, may indicate binary incomp
atibility. Expected 192 from C header, got 216 from PyObject
```

Data loading

In [2]:
```
from sklearn.datasets import load_boston
boston_dataset = load_boston()
```

```
<frozen importlib._bootstrap>:219: RuntimeWarning: numpy.ufunc size changed, may indicate binary incomp
atibility. Expected 192 from C header, got 216 from PyObject
```

In [3]:
```
boston = pd.DataFrame(boston_dataset.data,
                      columns = boston_dataset.feature_names)
boston.head()
```

Out[3]:

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

In [6]:
```
# target variable

boston['MEDV'] = boston_dataset.target
```

In [7]:
```
boston.head()
```

Out[7]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
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	24.0
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	9.14	21.6
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	4.03	34.7
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	2.94	33.4
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	5.33	36.2

In [8]:
```
boston.shape
```

Out[8]: (506, 14)

In [9]:
```
1 boston.describe()
```

Out[9]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000

In [10]:
```
boston.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  MEDV     506 non-null    float64
dtypes: float64(14)
memory usage: 55.5 KB
```

In [ ]:
```

```

Univariate model (one feature, one target)

In [17]:
```
df = boston.loc[:,['LSTAT','MEDV']]
df.head()
```

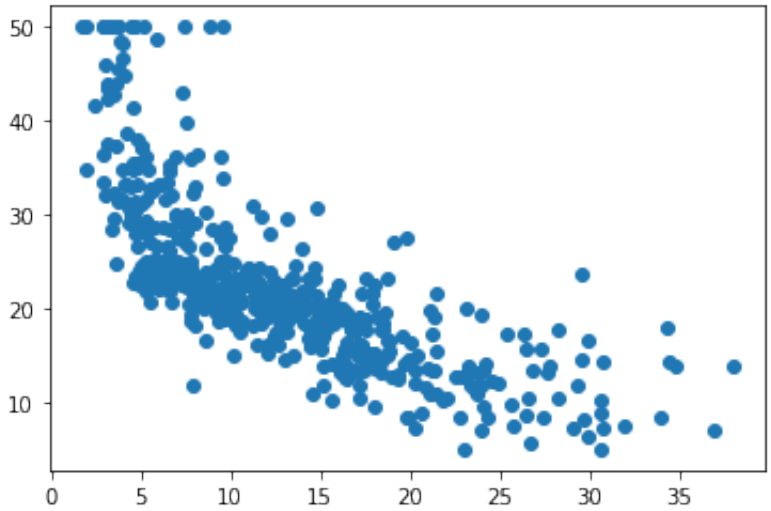
Out[17]:

	LSTAT	MEDV
0	4.98	24.0
1	9.14	21.6
2	4.03	34.7
3	2.94	33.4
4	5.33	36.2

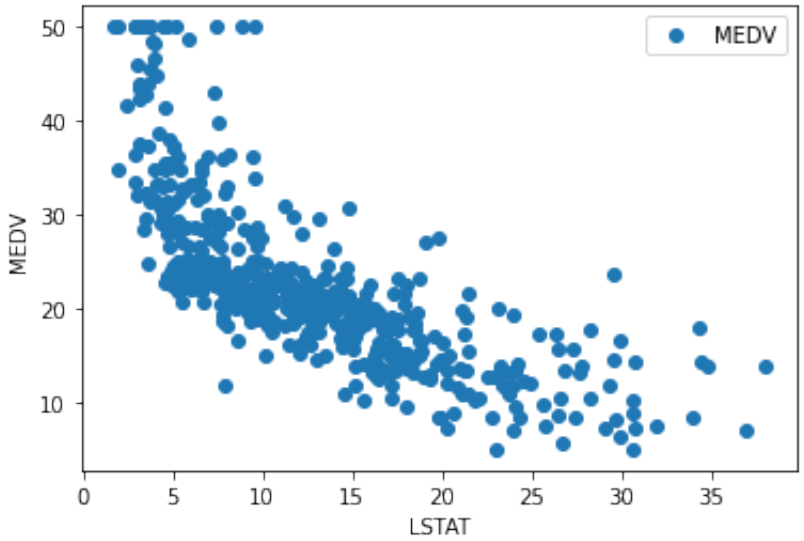
In [18]:
```
df.shape
```

Out[18]: (506, 2)

In [19]:
```
plt.scatter(df['LSTAT'],df['MEDV'])
plt.show()
```



In [21]:
```
df.plot(x = 'LSTAT',y = 'MEDV',style = 'o')
plt.xlabel('LSTAT')
plt.ylabel('MEDV')
plt.show()
```



Divide data into independent feature and dependent feature

In [31]:
```
X = pd.DataFrame(df['LSTAT'])
y = pd.DataFrame(df['MEDV'])
```

In [33]:
```
df.shape
```

Out[33]: (506, 2)

In [32]:
```
# divide data into training and testing set

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state = 1)

print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(404, 1)
(102, 1)
(404, 1)
(102, 1)
```

In [34]:
```
X_train
```

Out[34]:

	LSTAT
42	5.81
58	6.86
385	30.81
78	12.34
424	17.16
...	...
255	9.25
72	5.52
396	19.37
235	10.88
37	8.77

404 rows × 1 columns

In [35]:
```
1 y_train
```

Out[35]:

	MEDV
42	25.3
58	23.3
385	7.2
78	21.2
424	11.7
...	...
255	20.9
72	22.8
396	12.5
235	24.0
37	21.0

404 rows × 1 columns

In [37]:
```
1 X_test
```

Out[37]:

	LSTAT
307	7.53
343	7.18
47	18.80
67	8.10
362	10.19
...	...
92	8.16
224	4.14
110	13.00
426	15.69
443	18.85

102 rows × 1 columns

In [38]:
```
y_test
```

Out[38]:

	MEDV
307	28.2
343	23.9
47	16.6
67	22.0
362	20.8
...	...
92	22.9
224	44.8
110	21.7
426	10.2
443	15.4

102 rows × 1 columns

In [39]:
```
# linear regression model

regressor = LinearRegression()
regressor.fit(X_train,y_train)
```

Out[39]:
```
LinearRegression()
```

In [40]:
```
print(regressor.intercept_)
```

```
[34.33497839]
```

In [41]:
```
print(regressor.coef_)
```

```
[[-0.92441715]]
```

In [42]:
```
# prediction

y_pred = regressor.predict(X_test)
```

In [43]:
```
y_pred
```

Out[43]:
```
array([[27.37411725],
       [27.69766325],
       [16.95593597],
       [26.84719947],
       [24.91516763],
       [24.05545968],
       [29.99021779],
       [22.28057875],
       [17.76942306],
       [26.1908633 ],
       [27.17998965],
       [30.07341533],
       [21.75366098],
       [24.86894677],
       [23.50080939],
       [23.12179836],
       [12.85152382],
       [30.05492699],
       [27.46655897],
       [ 7.03693995],
       [23.70418116],
       [18.94343284],
       [25.75638724],
       [28.67754543],
       [30.0179503 ],
       [11.7884441 ],
       [15.53233356],
       [24.6008658 ],
```

```
[27.62370988],
[15.06088081],
[29.25992824],
[17.2702378 ],
[31.672657  ],
[19.13756044],
[25.9320265 ],
[21.77214932],
[17.88959729],
[29.40783498],
[12.75908211],
[20.48720948],
[27.54975651],
[28.09516263],
[27.30940805],
[12.05652507],
[17.66773717],
[13.3137324 ],
[32.56009746],
[19.22075799],
[25.26644615],
[24.50842409],
[23.55627442],
[23.96301797],
[29.51876504],
[24.07394802],
[ 6.89827738],
[28.11365097],
[ 6.6301964 ],
[28.80696383],
[20.7737788 ],
[30.68353065],
[20.4502328 ],
[28.2985344 ],
[15.92058876],
[18.02825986],
[ 7.01845161],
[29.67591595],
[32.05166803],
```

```
[26.33877004],
[24.72104003],
[23.57476276],
[28.64981292],
[ 8.19246139],
[21.39313829],
[23.26046093],
[21.30994075],
[24.70255169],
[31.59870363],
[26.72702525],
[27.50353565],
[31.04405334],
[20.10819845],
[24.81348174],
[29.35236995],
[12.82379131],
[28.80696383],
[29.74062515],
[16.48448322],
[29.63893927],
[21.30069657],
[18.65686353],
[29.12126566],
[29.60196258],
[17.63076049],
[22.84447322],
[20.91244137],
[22.06796281],
[25.39586455],
[26.79173445],
[30.50789139],
[22.31755544],
[19.83087331],
[16.90971511]])
```

In [45]:
```
len(y_pred)
```

Out[45]:
```
102
```

In [47]:
```
# Mean absolute error
print("MAE: ",metrics.mean_absolute_error(y_test,y_pred))
print("MSE: ",metrics.mean_squared_error(y_test,y_pred))
```

```
MAE:  5.078127727696938
MSE:  46.99482091954711
```

Multivariate (Multiple feature, single target)

In [53]:
```
X1 = pd.DataFrame(boston.iloc[:,:-1])
y1 = pd.DataFrame(boston.iloc[:,-1])

print("Features set : ",X1.shape)
print("Target : ",y1.shape)
```

```
Features set :  (506, 13)
Target :  (506, 1)
```

In [50]:
```
X1.head()
```

Out[50]:

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

In [51]:
```
y1.head()
```

Out[51]:

	MEDV
0	24.0
1	21.6
2	34.7
3	33.4
4	36.2

In [54]:
```
# splitting data into train and test set

X_train1, X_test1, y_train1, y_test1 = train_test_split(X1,y1,test_size=0.2,random_state = 1)

print(X_train1.shape)
print(X_test1.shape)
print(y_train1.shape)
print(y_test1.shape)
```

```
(404, 13)
(102, 13)
(404, 1)
(102, 1)
```

In [59]:
```
X_train1
```

Out[59]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
42	0.14150	0.0	6.91	0.0	0.448	6.169	6.6	5.7209	3.0	233.0	17.9	383.37	5.81
58	0.15445	25.0	5.13	0.0	0.453	6.145	29.2	7.8148	8.0	284.0	19.7	390.68	6.86
385	16.81180	0.0	18.10	0.0	0.700	5.277	98.1	1.4261	24.0	666.0	20.2	396.90	30.81
78	0.05646	0.0	12.83	0.0	0.437	6.232	53.7	5.0141	5.0	398.0	18.7	386.40	12.34
424	8.79212	0.0	18.10	0.0	0.584	5.565	70.6	2.0635	24.0	666.0	20.2	3.65	17.16
...	...	...	...	...	...	...	...	...	...	...	...	...	...
255	0.03548	80.0	3.64	0.0	0.392	5.876	19.1	9.2203	1.0	315.0	16.4	395.18	9.25
72	0.09164	0.0	10.81	0.0	0.413	6.065	7.8	5.2873	4.0	305.0	19.2	390.91	5.52
396	5.87205	0.0	18.10	0.0	0.693	6.405	96.0	1.6768	24.0	666.0	20.2	396.90	19.37
235	0.33045	0.0	6.20	0.0	0.507	6.086	61.5	3.6519	8.0	307.0	17.4	376.75	10.88
37	0.08014	0.0	5.96	0.0	0.499	5.850	41.5	3.9342	5.0	279.0	19.2	396.90	8.77

404 rows × 13 columns

In [55]:
```
# linear regression model

regressor1 = LinearRegression()
regressor1.fit(X_train1,y_train1)
```

Out[55]:
```
LinearRegression()
```

In [57]:
```
# reg coeff

regressor1.coef_
```

Out[57]:
```
array([[-1.12386867e-01,  5.80587074e-02,  1.83593559e-02,
         2.12997760e+00, -1.95811012e+01,  3.09546166e+00,
         4.45265228e-03, -1.50047624e+00,  3.05358969e-01,
        -1.11230879e-02, -9.89007562e-01,  7.32130017e-03,
        -5.44644997e-01]])
```

In [58]:
```
regressor1.intercept_
```

Out[58]:
```
array([42.93352585])
```

In [64]:
```
a = pd.DataFrame(regressor1.coef_,index = ['Coeff']).transpose() # coeff values of each feature
b = pd.DataFrame(X_train1.columns, columns = ['Features']) # features names
```

In [67]:
```
coeff_df = pd.concat([b,a],axis = 1)
coeff_df
```

Out[67]:

	Features	Coeff
0	CRIM	-0.112387
1	ZN	0.058059
2	INDUS	0.018359
3	CHAS	2.129978
4	NOX	-19.581101
5	RM	3.095462
6	AGE	0.004453
7	DIS	-1.500476
8	RAD	0.305359
9	TAX	-0.011123
10	PTRATIO	-0.989008
11	B	0.007321
12	LSTAT	-0.544645

In [69]:
```
# prediction

y_pred1 = regressor1.predict(X_test1)
len(y_pred1)
```

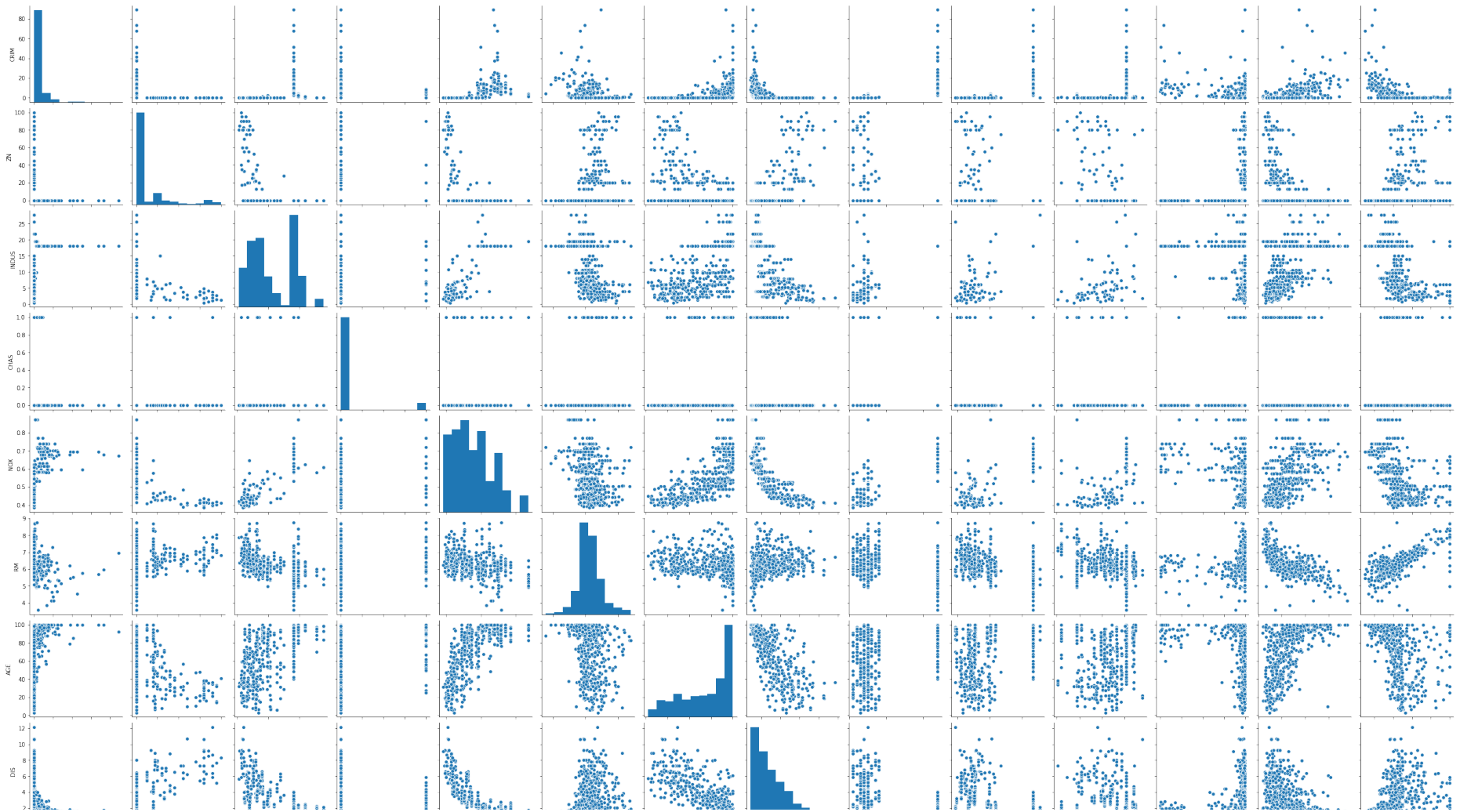
Out[69]:
```
102
```

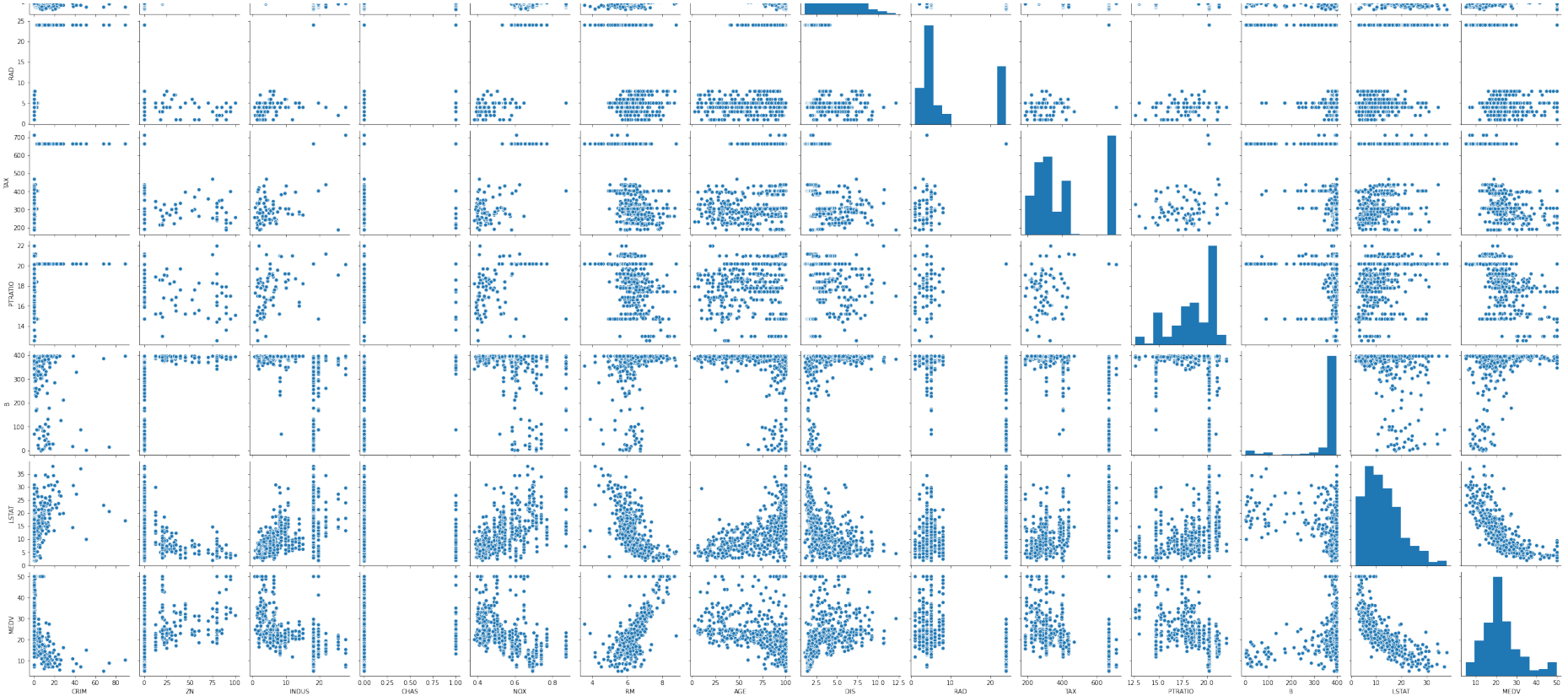
In [70]:
```
# Mean absolute error
print("MAE: ",metrics.mean_absolute_error(y_test1,y_pred1))

# Mean squared error
print("MSE: ",metrics.mean_squared_error(y_test1,y_pred1))
```

```
MAE:  3.7507121808389186
MSE:  23.380836480270375
```

In [73]:
```
sns.pairplot(data = boston)
plt.show()
```

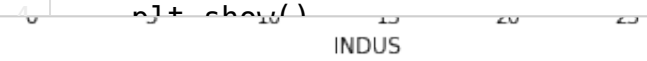




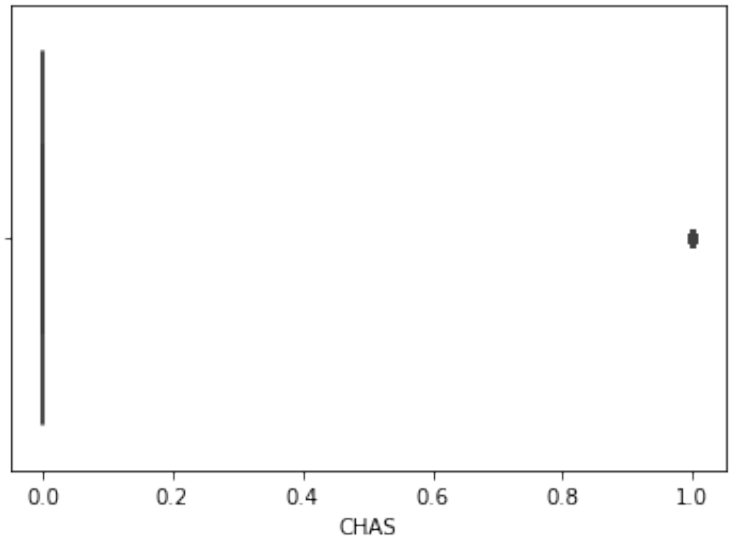
In [ ]:

```
# find null values if any
# scaling
# outlier
```

In [78]:
```
for i in X_train1.columns:
    print(i)
    sns.boxplot(boston[i])
    plt.show()
```



CHAS



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

```
1
```

In [ ]:

```
1
```

In [ ]:

```
1
```