In [67]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.formula.api as smf
```

In [68]:

```
boston_data = pd.read_csv('Boston House.zip')
boston_data
```

Out[68]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90
...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273.0	21.0	391.99
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273.0	21.0	396.90
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273.0	21.0	396.90
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273.0	21.0	393.45
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273.0	21.0	396.90

506 rows × 14 columns

In [33]:

```
boston_data.shape
```

Out[33]:

(506, 14)

In [34]:

```
boston_data.head()
```

Out[34]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LS
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5

In [35]:

```
boston_data.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    int64  
 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    int64  
 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(12), int64(2)
memory usage: 55.5 KB
```

In [36]:

```
boston_data.isna().sum()
```

Out[36]:

```
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
MEDV       0
dtype: int64
```

In [41]:

```
boston_data.dtypes
```

Out[41]:

```
CRIM       float64
ZN         float64
INDUS      float64
CHAS         int64
NOX        float64
RM         float64
AGE        float64
DIS        float64
RAD          int64
TAX        float64
PTRATIO    float64
B          float64
LSTAT      float64
MEDV       float64
dtype: object
```

In [42]:

```
boston_data
```

Out[42]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90
...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273.0	21.0	391.99
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273.0	21.0	396.90
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273.0	21.0	396.90
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273.0	21.0	393.45
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273.0	21.0	396.90

506 rows × 14 columns

Have a glance at the dependent and independent variables

In [46]:

```
boston_data_2 = boston_data.loc[:,['LSTAT','MEDV']]
boston_data_2.head(5)
```

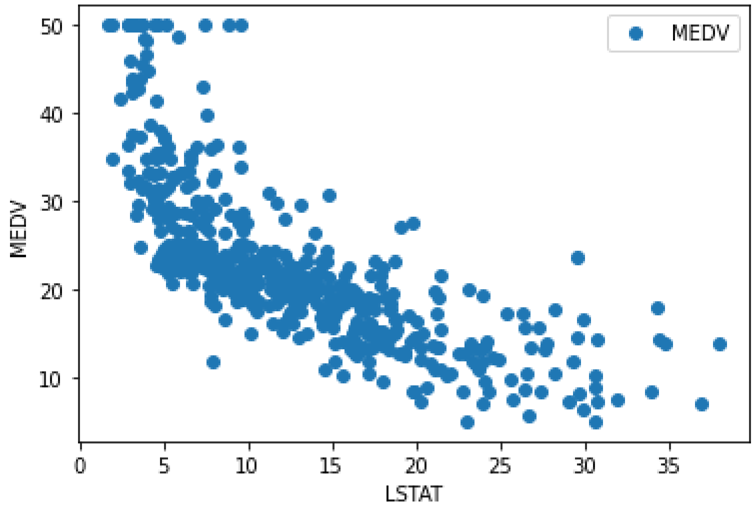
Out[46]:

	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

Visualize the change in the variables

In [47]:

```
import matplotlib.pyplot as plt
boston_data_2.plot(x='LSTAT', y='MEDV',style='o')
plt.xlabel('LSTAT')
plt.ylabel('MEDV')
plt.show()
```



In [62]:

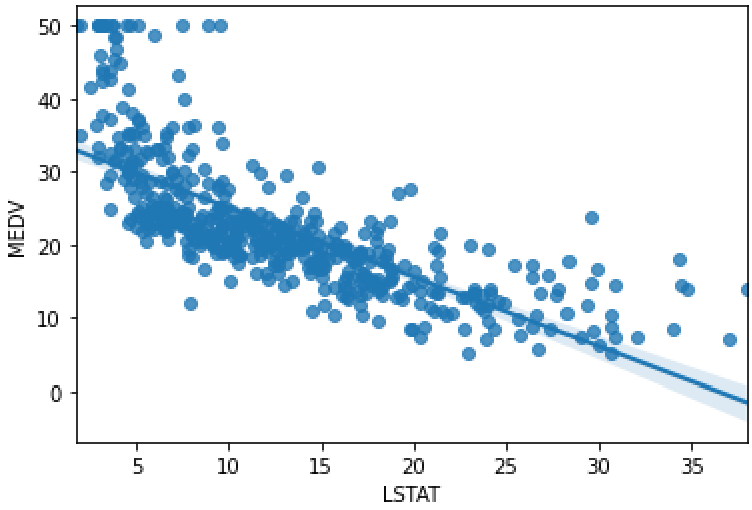
```
# correlation analysis
boston_data_2.corr()
```

Out[62]:

	LSTAT	MEDV
LSTAT	1.000000	-0.737663
MEDV	-0.737663	1.000000

In [64]:

```
sns.regplot(x='LSTAT',y='MEDV',data=boston_data_2)
plt.show()
```



Divide the data into independent and dependent variables

In [48]:

```
x = pd.DataFrame(boston_data_2['LSTAT'])
y = pd.DataFrame(boston_data_2['MEDV'])
```

Split the data into train and test sets

In [49]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=1)
```

Shape of the train and test sets

In [50]:

```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

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

Train the algorithm

In [51]:

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(x_train, y_train)
```

Out[51]:

```
LinearRegression()
```

Retrieve the intercept

In [52]:

```
print(regressor.intercept_)
```

```
[34.33497839]
```

Retrieve the slope

In [58]:

```
print(regressor.coef_)
```

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

Predicted value

In [59]:

```
y_train
```

Out[59]:

	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

Actual value

In [56]:

```
y_test
```

Out[56]:

	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 [65]:

```
data_pred = pd.DataFrame(boston_data_2,columns = ['LSTAT'])
data_pred
```

Out[65]:

	LSTAT
0	4.98
1	9.14
2	4.03
3	2.94
4	5.33
...	...
501	9.67
502	9.08
503	5.64
504	6.48
505	7.88

506 rows × 1 columns

In [ ]: