In [2]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.preprocessing import StandardScaler
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
```

In [3]:

```python
data = pd.read_csv('HousingData.csv')
```

In [4]:

```python
data
```

Out[4]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LS
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	N
...	...	...	...	...	...	...	...	...	...	...	...	...	
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	N
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	9
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	5
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	6
505	0.04741	0.0	11.93	0.0	0.573	6.030	NaN	2.5050	1	273	21.0	396.90	7

506 rows × 14 columns

In [5]:

```python
data.head()
```

Out[5]:

	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	296	15.3	396.90	4.9
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.1
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.0
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.9
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	Na

In [6]:
```
data.head(10)
```

Out[6]:
```
      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  296     15.3  396.90   4.9
1  0.02731  0.0   7.07   0.0  0.469  6.421   78.9  4.9671    2  242     17.8  396.90   9.1
2  0.02729  0.0   7.07   0.0  0.469  7.185   61.1  4.9671    2  242     17.8  392.83   4.0
3  0.03237  0.0   2.18   0.0  0.458  6.998   45.8  6.0622    3  222     18.7  394.63   2.9
4  0.06905  0.0   2.18   0.0  0.458  7.147   54.2  6.0622    3  222     18.7  396.90   Na
5  0.02985  0.0   2.18   0.0  0.458  6.430   58.7  6.0622    3  222     18.7  394.12   5.2
6  0.08829 12.5   7.87   NaN  0.524  6.012   66.6  5.5605    5  311     15.2  395.60  12.4
7  0.14455 12.5   7.87   0.0  0.524  6.172   96.1  5.9505    5  311     15.2  396.90  19.1
8  0.21124 12.5   7.87   0.0  0.524  5.631  100.0  6.0821    5  311     15.2  386.63  29.9
9  0.17004 12.5   7.87   NaN  0.524  6.004   85.9  6.5921    5  311     15.2  386.71  17.1
```

In [7]:
```
data.tail()
```

Out[7]:
```
        CRIM   ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD  TAX  PTRATIO       B  LST
501  0.06263  0.0  11.93   0.0  0.573  6.593  69.1  2.4786    1  273     21.0  391.99   Na
502  0.04527  0.0  11.93   0.0  0.573  6.120  76.7  2.2875    1  273     21.0  396.90   9.
503  0.06076  0.0  11.93   0.0  0.573  6.976  91.0  2.1675    1  273     21.0  396.90   5.
504  0.10959  0.0  11.93   0.0  0.573  6.794  89.3  2.3889    1  273     21.0  393.45   6.
505  0.04741  0.0  11.93   0.0  0.573  6.030   NaN  2.5050    1  273     21.0  396.90   7.
```

In [8]:
```
data.tail(10)
```

Out[8]:
```
        CRIM   ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD  TAX  PTRATIO       B  LST
496  0.28960  0.0   9.69   0.0  0.585  5.390  72.9  2.7986    6  391     19.2  396.90  21.
497  0.26838  0.0   9.69   0.0  0.585  5.794  70.6  2.8927    6  391     19.2  396.90  14.
498  0.23912  0.0   9.69   0.0  0.585  6.019  65.3  2.4091    6  391     19.2  396.90  12.
499  0.17783  0.0   9.69   0.0  0.585  5.569  73.5  2.3999    6  391     19.2  395.77  15.
500  0.22438  0.0   9.69   0.0  0.585  6.027  79.7  2.4982    6  391     19.2  396.90  14.
501  0.06263  0.0  11.93   0.0  0.573  6.593  69.1  2.4786    1  273     21.0  391.99   Na
502  0.04527  0.0  11.93   0.0  0.573  6.120  76.7  2.2875    1  273     21.0  396.90   9.
503  0.06076  0.0  11.93   0.0  0.573  6.976  91.0  2.1675    1  273     21.0  396.90   5.
504  0.10959  0.0  11.93   0.0  0.573  6.794  89.3  2.3889    1  273     21.0  393.45   6.
505  0.04741  0.0  11.93   0.0  0.573  6.030   NaN  2.5050    1  273     21.0  396.90   7.
```

Loading [MathJax]/extensions/Safe.js

In [9]:

```
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     486 non-null    float64
 1   ZN       486 non-null    float64
 2   INDUS    486 non-null    float64
 3   CHAS     486 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      486 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    int64  
 9   TAX      506 non-null    int64  
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    486 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 55.5 KB
```

In [10]:

```
data.columns
```

Out[10]:

```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TA
X',
       'PTRATIO', 'B', 'LSTAT', 'MEDV'],
      dtype='object')
```

In [11]:

```
data.describe()
```

Out[11]:

	CRIM	**ZN**	**INDUS**	**CHAS**	**NOX**	**RM**	**AGE**	
count	486.000000	486.000000	486.000000	486.000000	506.000000	506.000000	486.000000	506
mean	3.611874	11.211934	11.083992	0.069959	0.554695	6.284634	68.518519	3
std	8.720192	23.388876	6.835896	0.255340	0.115878	0.702617	27.999513	2
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1
25%	0.081900	0.000000	5.190000	0.000000	0.449000	5.885500	45.175000	2
50%	0.253715	0.000000	9.690000	0.000000	0.538000	6.208500	76.800000	3
75%	3.560263	12.500000	18.100000	0.000000	0.624000	6.623500	93.975000	5
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12

In [12]:

```
data.shape
```

Out[12]:

```
(506, 14)
```

Loading [MathJax]/extensions/Safe.js

In [13]:
```
data.index
```

Out[13]:
```
RangeIndex(start=0, stop=506, step=1)
```

In [14]:
```
data.isnull()
```

Out[14]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTA
0	False	False	False	False	False	False	False	False	False	False	False	False	Fals
1	False	False	False	False	False	False	False	False	False	False	False	False	Fals
2	False	False	False	False	False	False	False	False	False	False	False	False	Fals
3	False	False	False	False	False	False	False	False	False	False	False	False	Fals
4	False	False	False	False	False	False	False	False	False	False	False	False	Tru
...	...	...	...	...	...	...	...	...	...	...	...	...	
501	False	False	False	False	False	False	False	False	False	False	False	False	Tru
502	False	False	False	False	False	False	False	False	False	False	False	False	Fals
503	False	False	False	False	False	False	False	False	False	False	False	False	Fals
504	False	False	False	False	False	False	False	False	False	False	False	False	Fals
505	False	False	False	False	False	False	True	False	False	False	False	False	Fals

506 rows × 14 columns

In [16]:
```
data.isnull().any()
```

Out[16]:
```
CRIM        True
ZN          True
INDUS       True
CHAS        True
NOX        False
RM         False
AGE         True
DIS        False
RAD        False
TAX        False
PTRATIO    False
B          False
LSTAT       True
MEDV       False
dtype: bool
```

In [17]:

```
data.isnull().sum()
```

Out[17]:

```
CRIM       20
ZN         20
INDUS      20
CHAS       20
NOX         0
RM          0
AGE        20
DIS         0
RAD         0
TAX         0
PTRATIO     0
B           0
LSTAT      20
MEDV        0
dtype: int64
```

In [18]:

```
data['CRIM'].fillna(data['CRIM'].mean(),inplace=True)
data['ZN'].fillna(data['ZN'].mean(),inplace=True)
data['INDUS'].fillna(data['INDUS'].mean(),inplace=True)
data['CHAS'].fillna(data['CHAS'].mean(),inplace=True)
data['LSTAT'].fillna(data['LSTAT'].mean(),inplace=True)
data['AGE'].fillna(data['AGE'].mean(),inplace=True)
```

In [19]:

```
data.isnull().sum()
```

Out[19]:

```
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 [20]:

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

In [21]:
```
x
```

Out[21]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.200000	4.0900	1	296	15.3	396.90
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.900000	4.9671	2	242	17.8	396.90
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.100000	4.9671	2	242	17.8	392.83
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.800000	6.0622	3	222	18.7	394.63
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.200000	6.0622	3	222	18.7	396.90
...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.100000	2.4786	1	273	21.0	391.99
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.700000	2.2875	1	273	21.0	396.90
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.000000	2.1675	1	273	21.0	396.90
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.300000	2.3889	1	273	21.0	393.45
505	0.04741	0.0	11.93	0.0	0.573	6.030	68.518519	2.5050	1	273	21.0	396.90

506 rows × 13 columns

In [22]:
```
y
```

Out[22]:
```
0      24.0
1      21.6
2      34.7
3      33.4
4      36.2
       ...
501    22.4
502    20.6
503    23.9
504    22.0
505    11.9
Name: MEDV, Length: 506, dtype: float64
```

In [23]:
```
x.head()
```

Out[23]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LS
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	12.71

Loading [MathJax]/extensions/Safe.js

In [24]:
```
x.shape, y.shape
```

Out[24]:
```
((506, 13), (506,))
```

Basic Stats

In [26]:
```
x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 13 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    int64  
 9   TAX      506 non-null    int64  
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
dtypes: float64(11), int64(2)
memory usage: 51.5 KB
```

In [27]:
```
x.describe()
```

Out[27]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506
mean	3.611874	11.211934	11.083992	0.069959	0.554695	6.284634	68.518519	3
std	8.545770	22.921051	6.699165	0.250233	0.115878	0.702617	27.439466	2
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1
25%	0.083235	0.000000	5.190000	0.000000	0.449000	5.885500	45.925000	2
50%	0.290250	0.000000	9.900000	0.000000	0.538000	6.208500	74.450000	3
75%	3.611874	11.211934	18.100000	0.000000	0.624000	6.623500	93.575000	5
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12

In [29]:

```
y.info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 506 entries, 0 to 505
Series name: MEDV
Non-Null Count  Dtype
--------------  -----
506 non-null    float64
dtypes: float64(1)
memory usage: 4.1 KB
```

In [30]:

```
y.describe()
```

Out[30]:

```
count    506.000000
mean      22.532806
std        9.197104
min        5.000000
25%       17.025000
50%       21.200000
75%       25.000000
max       50.000000
Name: MEDV, dtype: float64
```

In [31]:

```
x.isnull().sum()
```

Out[31]:

```
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
```

In [32]:

```
y.isnull().sum()
```

Out[32]:

```
0
```

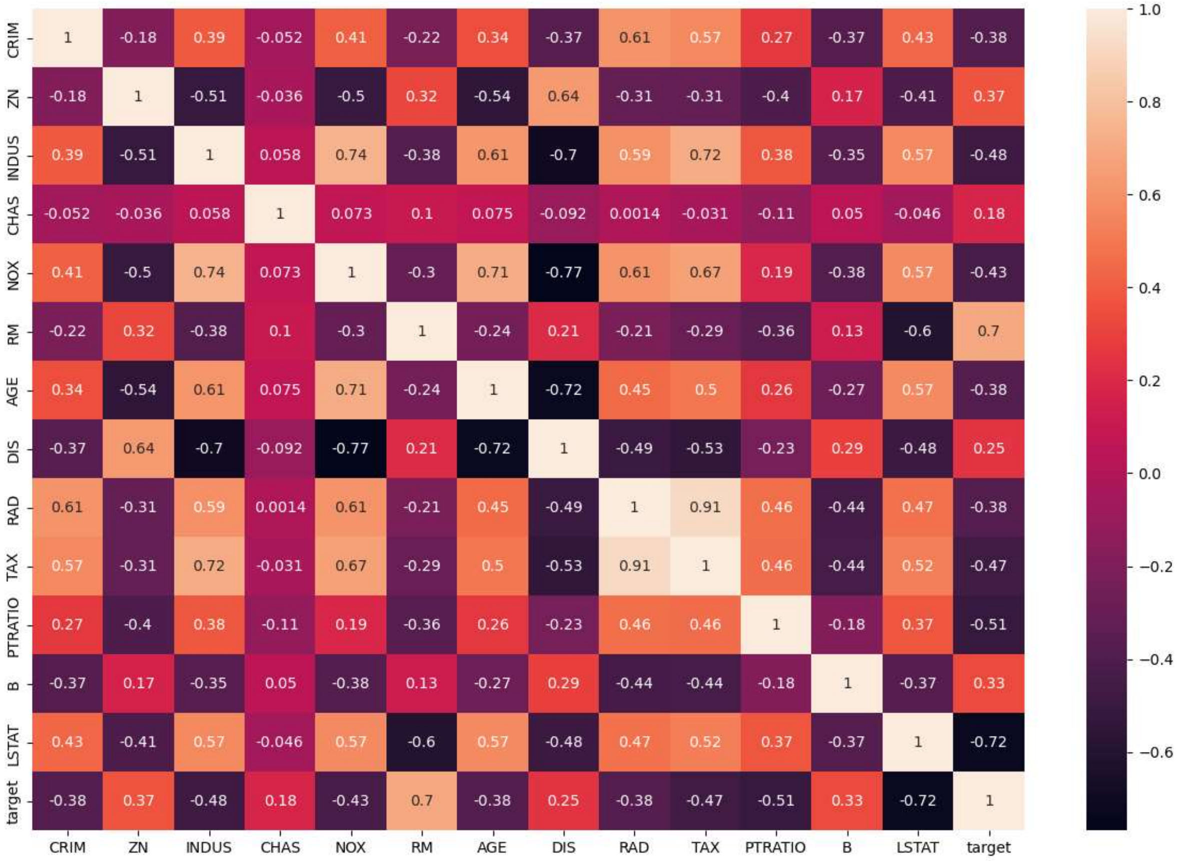
Loading [MathJax]/extensions/Safe.js

In [33]:
```
df = x
df["target"] = y
df.head()
```

Out[33]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LS
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	12.71

In [34]:
```
plt.figure(figsize=(15,10))
sns.heatmap(df.corr(), annot=True)
plt.show()
```

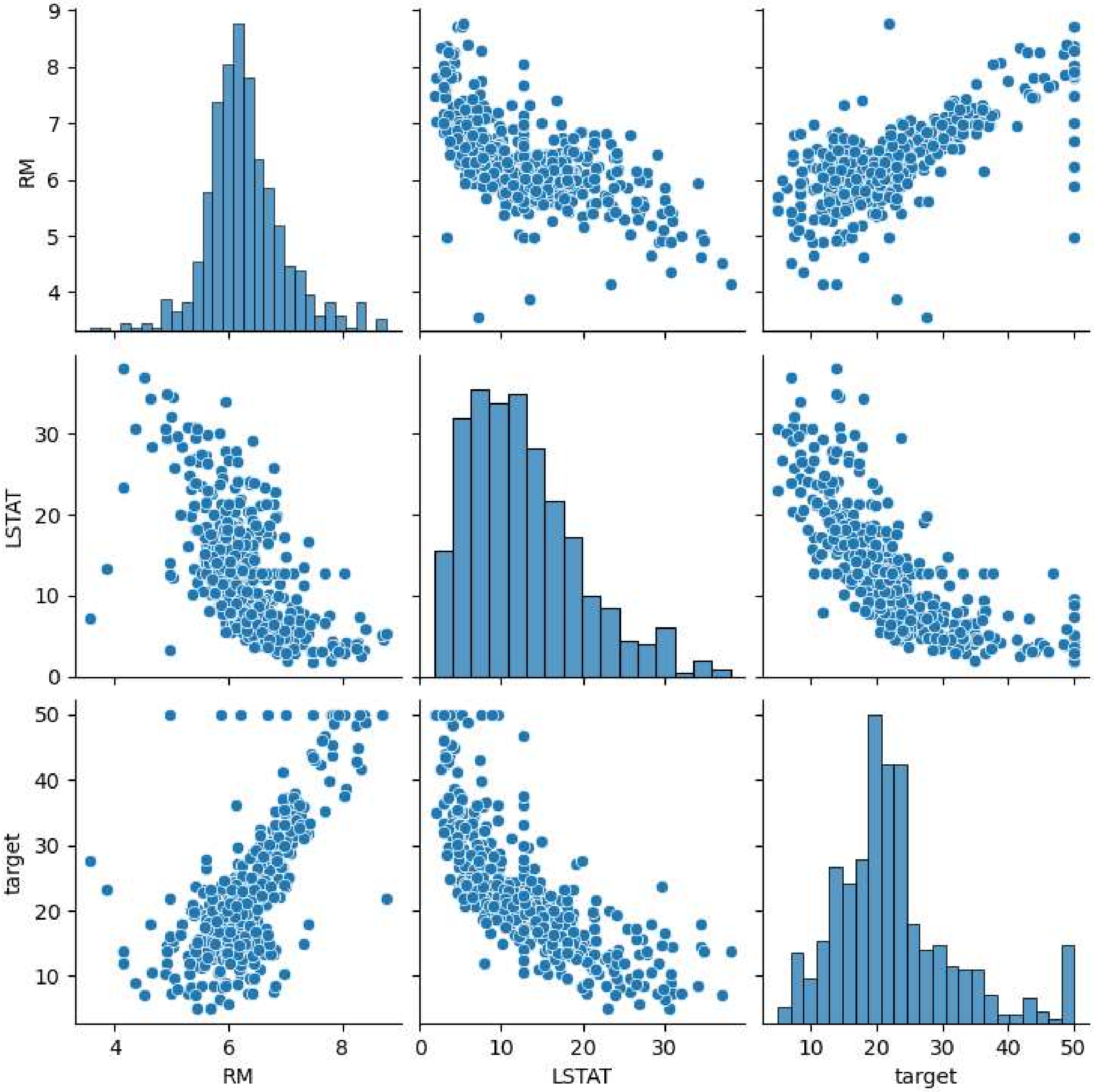


Considering only 'RM' and 'LSTAT' by considering correlation and multi-collinearity of other features

In [35]:
```
df = df[['RM', 'LSTAT', 'target']]
```

In [36]:

```
sns.pairplot(df)
plt.show()
```



In [37]:

```
x = df[['RM', 'LSTAT']]
y = df['target']
```

Scale The Data

In [38]:

```
scaler = StandardScaler()
```

In [39]:

```
x = scaler.fit_transform(x)
```

Split The Data

In [40]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, shuff
```

In [41]:
```
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

Out[41]:
```
((354, 2), (152, 2), (354,), (152,))
```

Linear Regression Modelling

In [42]:
```
model = LinearRegression(n_jobs=-1)
```

In [43]:
```
model.fit(x_train, y_train)
```

Out[43]:
```
LinearRegression(n_jobs=-1)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Make Predictions

In [44]:
```
y_pred = model.predict(x_test)
```

In [45]:
```
mean_absolute_error(y_test, y_pred)
```

Out[45]:
```
3.6220319170331003
```

In [46]:
```
mean_squared_error(y_test, y_pred)
```

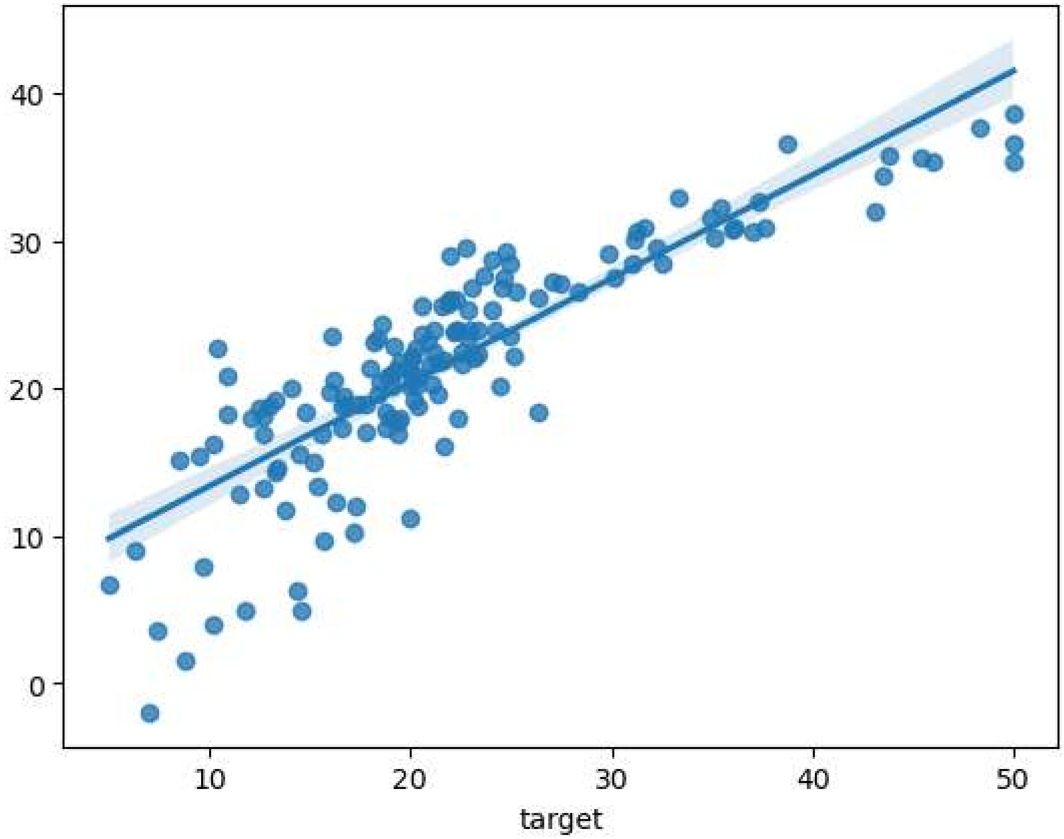
Out[46]:
```
22.510564423852326
```

In [51]:

```
sns.regplot(x = y_test, y = y_pred, ci= 95)
```

Out[51]:

```
<Axes: xlabel='target'>
```



In [ ]:

```

```

In [ ]:

```

```

Loading [MathJax]/extensions/Safe.js