In [3]:

```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

In [4]:

```
#loading dataset
from sklearn.datasets import load_boston
boston=load_boston()
```

In [5]:

```
boston.keys()
```

Out[5]:

```
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename'])
```

In [6]:

```
print(boston.DESCR)
```

```
.. _boston_dataset:

Boston house prices dataset
---------------------------

**Data Set Characteristics:**

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive. Median V
alue (attribute 14) is usually the target.

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over
25,000 sq.ft.
        - INDUS    proportion of non-retail business acres per town
        - CHAS     Charles River dummy variable (= 1 if tract bounds r
iver; 0 otherwise)
        - NOX      nitric oxides concentration (parts per 10 million)
        - RM       average number of rooms per dwelling
        - AGE      proportion of owner-occupied units built prior to 1
940
        - DIS      weighted distances to five Boston employment centre
s
        - RAD      index of accessibility to radial highways
        - TAX      full-value property-tax rate per $10,000
        - PTRATIO  pupil-teacher ratio by town
        - B        1000(Bk - 0.63)^2 where Bk is the proportion of bla
cks by town
        - LSTAT    % lower status of the population
        - MEDV     Median value of owner-occupied homes in $1000's

    :Missing Attribute Values: None

    :Creator: Harrison, D. and Rubinfeld, D.L.

This is a copy of UCI ML housing dataset.
https://archive.ics.uci.edu/ml/machine-learning-databases/housing/ (ht
tps://archive.ics.uci.edu/ml/machine-learning-databases/housing/)


This dataset was taken from the StatLib library which is maintained at
Carnegie Mellon University.

The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedon
ic
prices and the demand for clean air', J. Environ. Economics & Manageme
nt,
vol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regression diag
nostics
...', Wiley, 1980.   N.B. Various transformations are used in the tabl
e on
pages 244-261 of the latter.

The Boston house-price data has been used in many machine learning pap
ers that address regression
```

```
problems.

.. topic:: References

   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influ
ential Data and Sources of Collinearity', Wiley, 1980. 244-261.
   - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learn
ing. In Proceedings on the Tenth International Conference of Machine L
earning, 236-243, University of Massachusetts, Amherst. Morgan Kaufman
n.
```

In [7]:

```
print(boston.feature_names)
```

```
['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATI
O'
 'B' 'LSTAT']
```

In [8]:

```
bs=pd.DataFrame(boston.data)
```

In [9]:

```
bs.head()
```

Out[9]:

	0	1	2	3	4	5	6	7	8	9	10	11	12
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 [10]:

```
bs.columns=boston.feature_names
```

In [11]:

```
bs.head()
```

Out[11]:

	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	1.0	296.0	15.3	396.90	
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	
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	
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	
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	

In [12]:

```
bs['PRICE']=boston.target
```

In [13]:

```
bs['PRICE']
```

Out[13]:

```
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: PRICE, Length: 506, dtype: float64
```

In [14]:

```
bs.head()
```

Out[14]:

	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	1.0	296.0	15.3	396.90	
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	
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	
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	
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	

In [15]:

```
bs.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
```

In [16]:

```
bs.isnull().sum()
```

Out[16]:

```
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
PRICE      0
dtype: int64
```

In [17]:

```
bs.describe()
```

Out[17]:

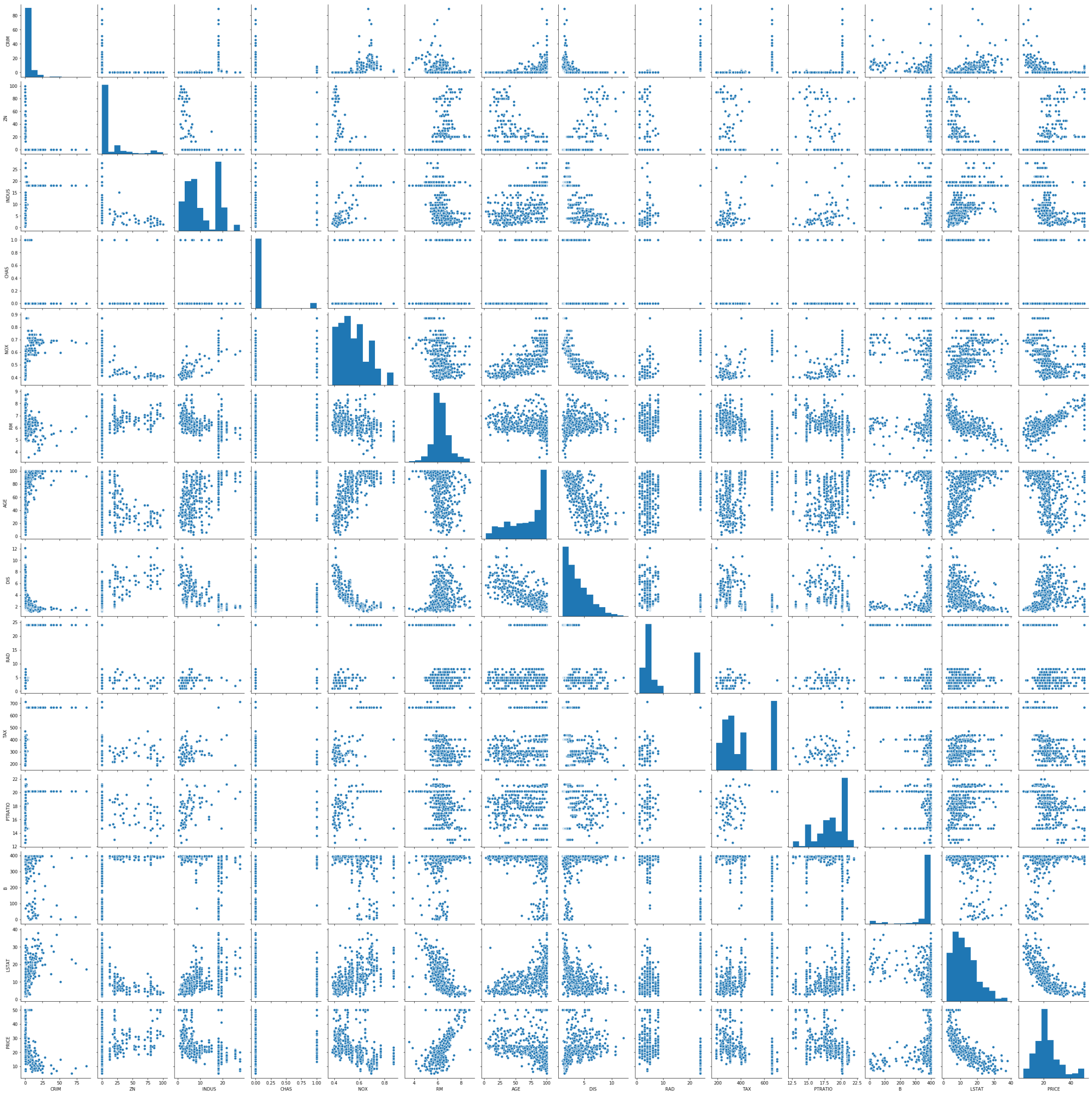
	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	50
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	1

In [18]:

```
sns.pairplot(bs)                    #all features are mapped with other features
```

Out[18]:

<seaborn.axisgrid.PairGrid at 0x20e5892dec8>

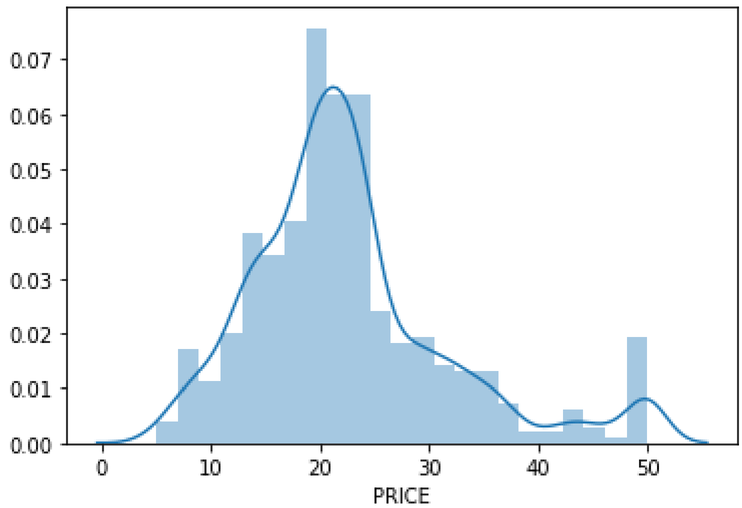


In [31]:

```
sns.distplot(bs['PRICE'])                    #visualize the distribution of price
```

Out[31]:

<matplotlib.axes._subplots.AxesSubplot at 0x20e63cf58c8>

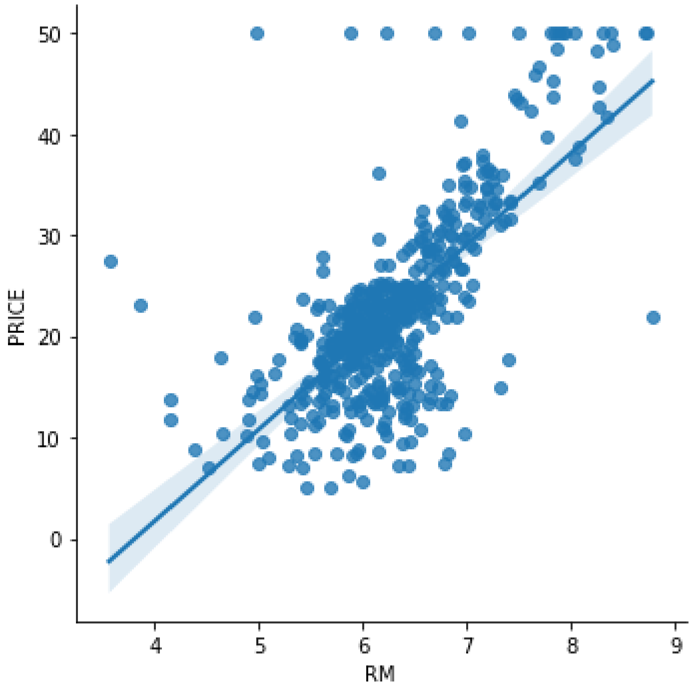


In [22]:

```
sns.lmplot(x='RM',y='PRICE',data=bs)
```

Out[22]:

<seaborn.axisgrid.FacetGrid at 0x20e63c58e08>



Training a linear regression model

In [24]:

```
X=bs.drop('PRICE',axis=1)
Y=bs['PRICE']
```

Train Test Split

In [25]:

```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.3,random_state=4)
```

Creating and training model

In [26]:

```
lr=LinearRegression()
```

In [27]:

```
lr.fit(X_train,Y_train)
```

Out[27]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normali
ze=False)
```

In [28]:

```
Y_pred=lr.predict(X_test)
```

In [29]:

```
Y_pred
```

Out[29]:

```
array([11.07380893, 26.47910329, 17.34489869, 19.1948608 , 36.3617073
5,
       24.77095832, 31.00051311, 19.94226881, 19.22375105, 24.4299843
5,
       28.31512637, 28.40796034, 19.27427968, 33.82295207, 21.2859648
7,
       15.11171444, 20.97688767, 11.28556596, 11.8611348 , 13.8844438
7,
        5.37422679, 17.55278177, 20.58171204, 22.59849951, 16.0754426
5,
       20.45924503, 19.1068775 , 14.37832191, 21.23235601, 17.5218656
4,
       14.40725559, 23.68483414, 33.7410661 , 22.02733357, 17.6213914
7,
       19.97241153, 30.24069397, 34.69718954, 23.85821534, 24.3071509
3,
       36.13378112, 31.97532293, 19.626175  , 31.61097971, 34.5812780
9,
       25.62718797, 39.95041812, 17.60880538, 19.90319708, 23.4041750
1,
       33.70182396, 25.62491083, 18.25559302, 27.27317174, 13.4637787
1,
       23.43470656, 24.43721849, 33.52056736, 16.99896935, 37.9446440
4,
       15.94567818, 19.32528916, 31.84088262, 15.25081303, 38.4034478
9,
       27.45372884, 34.36154312,  9.37353936, 19.42580066, 21.9921845
9,
       22.79983394, 22.50810313, 22.30918714, 27.84395887, 16.4081834
5,
       22.55507669, 16.5147332 , 25.11106836, 13.76991927, 19.7865639
9,
       22.10247463, 20.26663237, 28.15165586, 19.52050766, 30.3325436
4,
       22.79109999, 29.2663436 , 19.43113706, 24.7968264 , 37.4627564
8,
       31.05503576, 41.3372879 , 18.46365381, 36.67964528, 19.4084240
5,
       23.61810063, 27.93475362, 24.41825213,  9.4599059 , 20.6808867
7,
        8.99426788, 28.4492398 , 31.88237066, 14.04302958, 24.8347909
,
       19.94124425, 36.90271393, 31.06556982, 33.91883403, 28.6459153
6,
       31.1007263 , 22.82363163, 11.58125942, 29.46902405, 37.0606610
6,
       23.01945872, 41.79865192, 18.44334162,  3.433324  , 18.5748566
3,
       22.21257489, 16.71192648, 28.00473344, 28.42374739, 19.6417452
,
       18.76090758, 35.37631447, 13.12349548, 14.73923539, 18.1620233
3,
       38.26604753, 15.97821613, 41.91544265, 30.44631625, 28.6584808
9,
       24.19590457, 12.06559683, 26.01408744, 23.25012698, 18.9250685
```

```
7,
       17.05016777, 17.50245392, 20.89247338, 24.62630514,  1.8216755
8,
       23.03969555, 19.35693345, 17.89193065, 38.43943954, 19.7075262
,
       31.67181183, 19.0130913 ])
```

In [ ]: