Practical No. 4

Name: Prathamesh Pawar | Roll No: B-23

Create a Linear Regression Model using Python/R to predict home prices using Boston Housing Dataset (https://www.kaggle.com/c/boston-housing (https://www.kaggle.com/c/boston-housing)). The Boston Housing dataset contains information about various houses in Boston through different parameters. There are 506 samples and 14 feature variables in this dataset.

The objective is to predict the value of prices of the house using the given features.

```
The Boston Housing Dataset

The Boston Housing Dataset is a derived from information collected by the U.S. Census
Service concerning housing in the area of Boston MA. The following describes the dataset
columns:

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 river; 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 1940
DIS - weighted distances to five Boston employment centres
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 blacks by town
LSTAT - % lower status of the population
MEDV - Median value of owner-occupied homes in $1000's
```

In [1]:

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

In [2]:

```python
# Importing DataSet and take a look at Data
Boston = pd.read_csv("Boston.csv")
Boston.head()
```

Out[2]:

	Unnamed: 0	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO
0	1	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3
1	2	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8
2	3	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8
3	4	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7
4	5	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7

In [12]:

```
Boston.info()
Boston.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  506 non-null    int64  
 1   CRIM        506 non-null    float64
 2   ZN          506 non-null    float64
 3   INDUS       506 non-null    float64
 4   CHAS        506 non-null    int64  
 5   NOX         506 non-null    float64
 6   RM          506 non-null    float64
 7   AGE         506 non-null    float64
 8   DIS         506 non-null    float64
 9   RAD         506 non-null    int64  
 10  TAX         506 non-null    int64  
 11  PTRATIO     506 non-null    float64
 12  BLACK       506 non-null    float64
 13  LSTAT       506 non-null    float64
 14  MEDV        506 non-null    float64
dtypes: float64(11), int64(4)
memory usage: 59.4 KB
```

Out[12]:

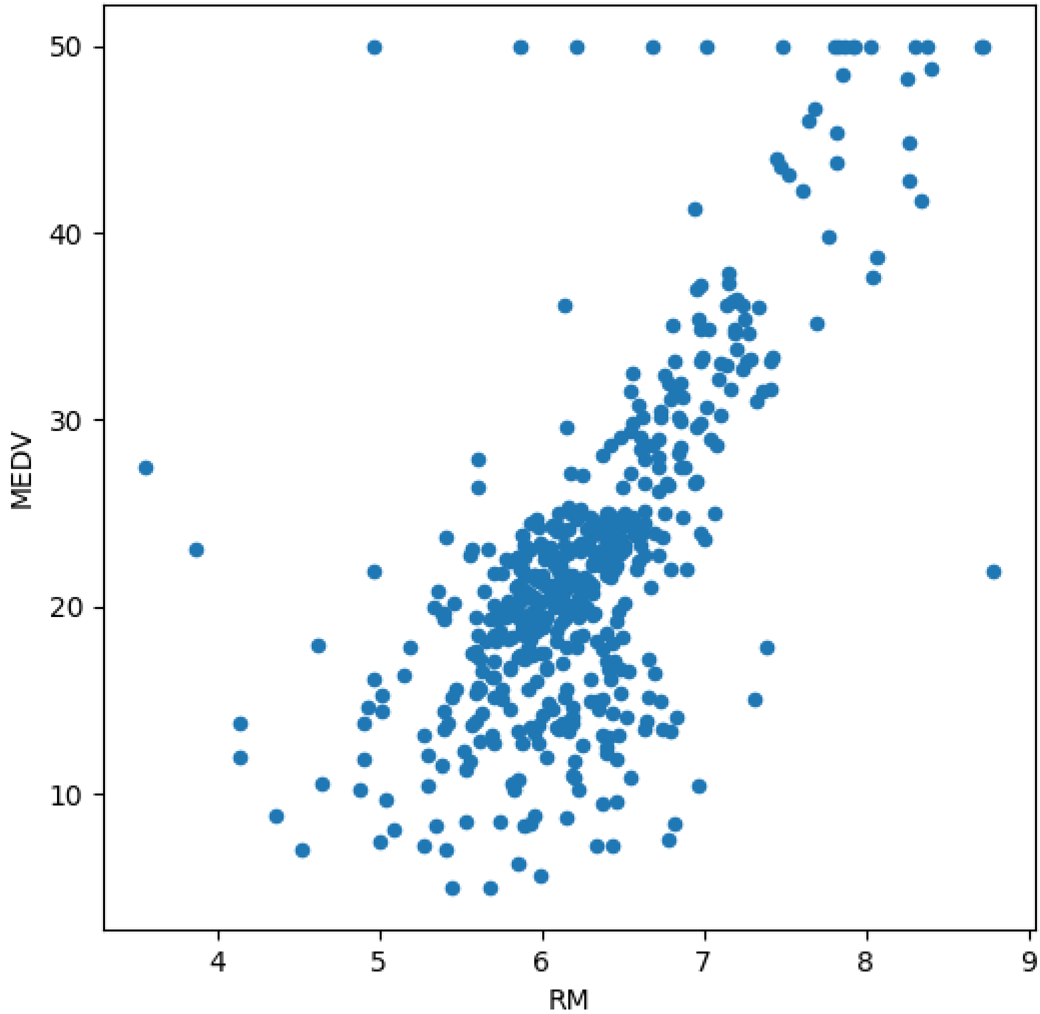
	Unnamed: 0	CRIM	ZN	INDUS	CHAS	NOX	RM
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	253.500000	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634
std	146.213884	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617
min	1.000000	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000
25%	127.250000	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500
50%	253.500000	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500
75%	379.750000	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500
max	506.000000	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000

In [4]:

```
Boston.plot.scatter('RM', 'MEDV', figsize=(6, 6))
```

Out[4]:

```
<Axes: xlabel='RM', ylabel='MEDV'>
```



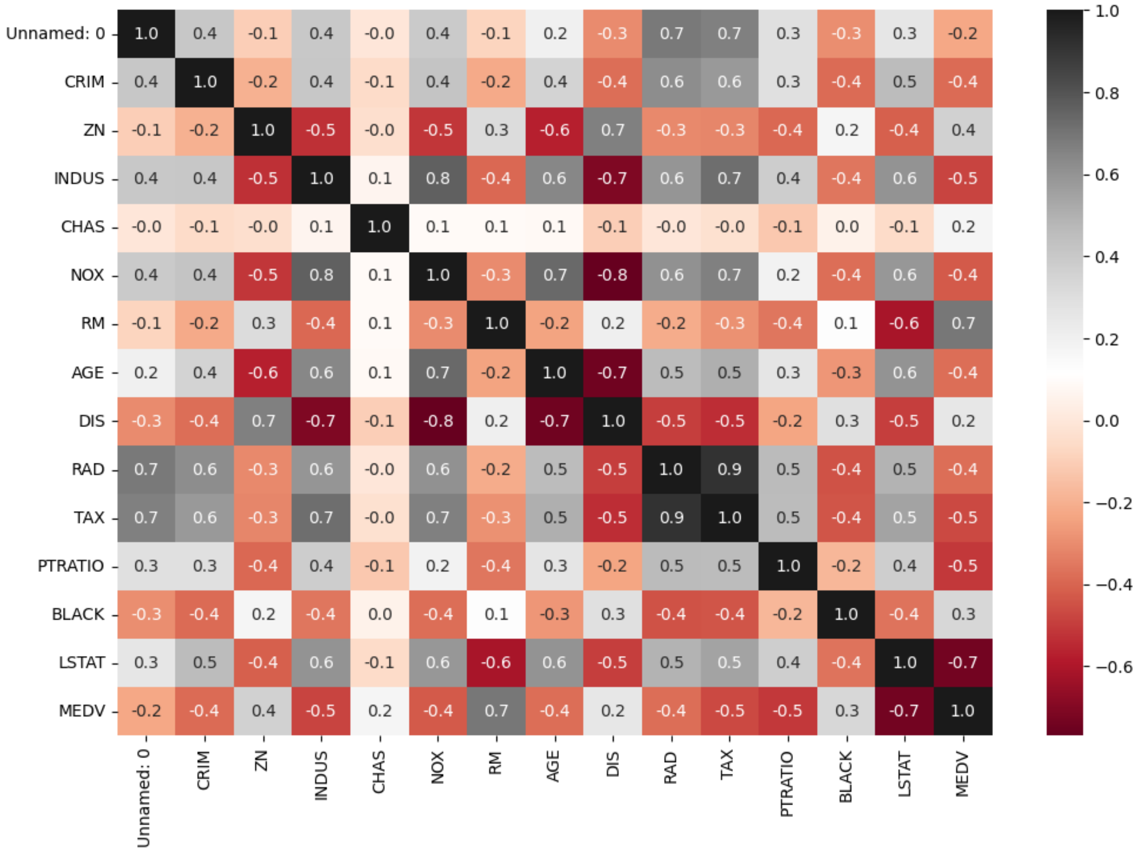
In this plot its clearly to see a linear pattern. Wheter more average number of rooms per dwelling, more expensive the median value is.

In [5]:

```
plt.subplots(figsize=(12,8))
sns.heatmap(Boston.corr(), cmap = 'RdGy', annot = True, fmt = '.1f')
```

Out[5]:

```
<Axes: >
```




```
At this heatmap plot, we can do our analysis better than the pairplot.
Lets focus at the last line, where y = MEDV:
When shades of Red/Orange: the more red the color is on X axis, smaller the MEDV.
Negative correlation
When light colors: those variables at axis x and y, they dont have any relation. Zero
correlation
When shades of Gray/BLACK : the more BLACK the color is on X axis, more higher the value
medv is. Positive correlation
```

Trainning Linear Regression Model

Define X and Y

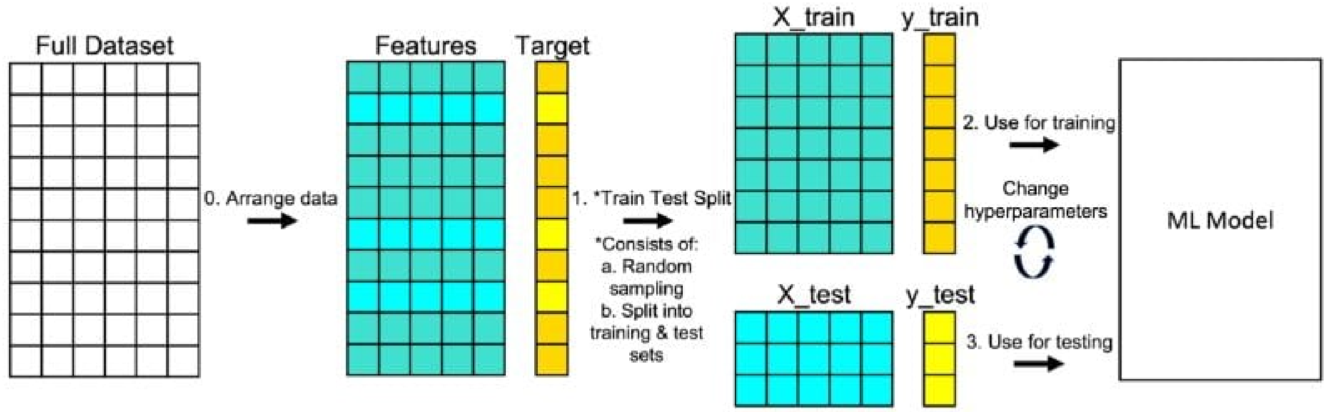
- X: Varibles named as predictors, independent variables, features.
- Y: Variable named as response or dependent variable

In [6]:

```
X = Boston[['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX','PTRAT
Y = Boston['MEDV']
```

Import sklearn librarys:

```
train_test_split, to split our data in two DF, one for build a model and other to
validate.
LinearRegression, to apply the linear regression.
```



In [7]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [8]:

```
# Split DataSet
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3,random_state=0)
```

In [14]:

```
print(f'Train Dataset Size - X: {X_train.shape}, Y: {Y_train.shape}')
print(f'Test Dataset Size - X: {X_test.shape}, Y: {Y_test.shape}')
```

```
Train Dataset Size - X: (354, 13), Y: (354,)
Test Dataset Size - X: (152, 13), Y: (152,)
```

In [16]:

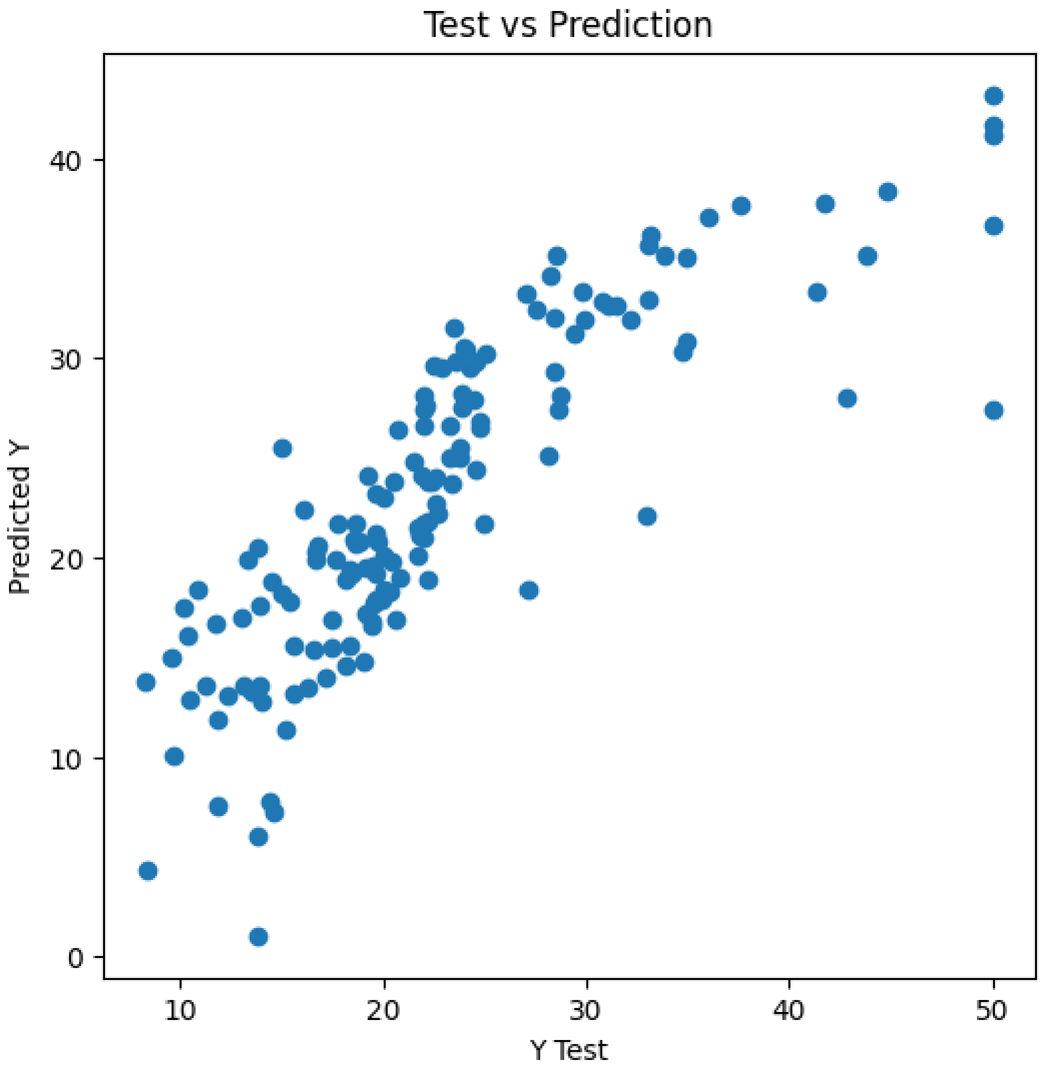
```
# Model Building
lm = LinearRegression()
lm.fit(X_train,Y_train)
predictions = lm.predict(X_test)
```

In [22]:

```
# Model Visualization
plt.figure(figsize=(6, 6))
plt.scatter(Y_test, predictions)
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
plt.title('Test vs Prediction')
```

Out[22]:

```
Text(0.5, 1.0, 'Test vs Prediction')
```

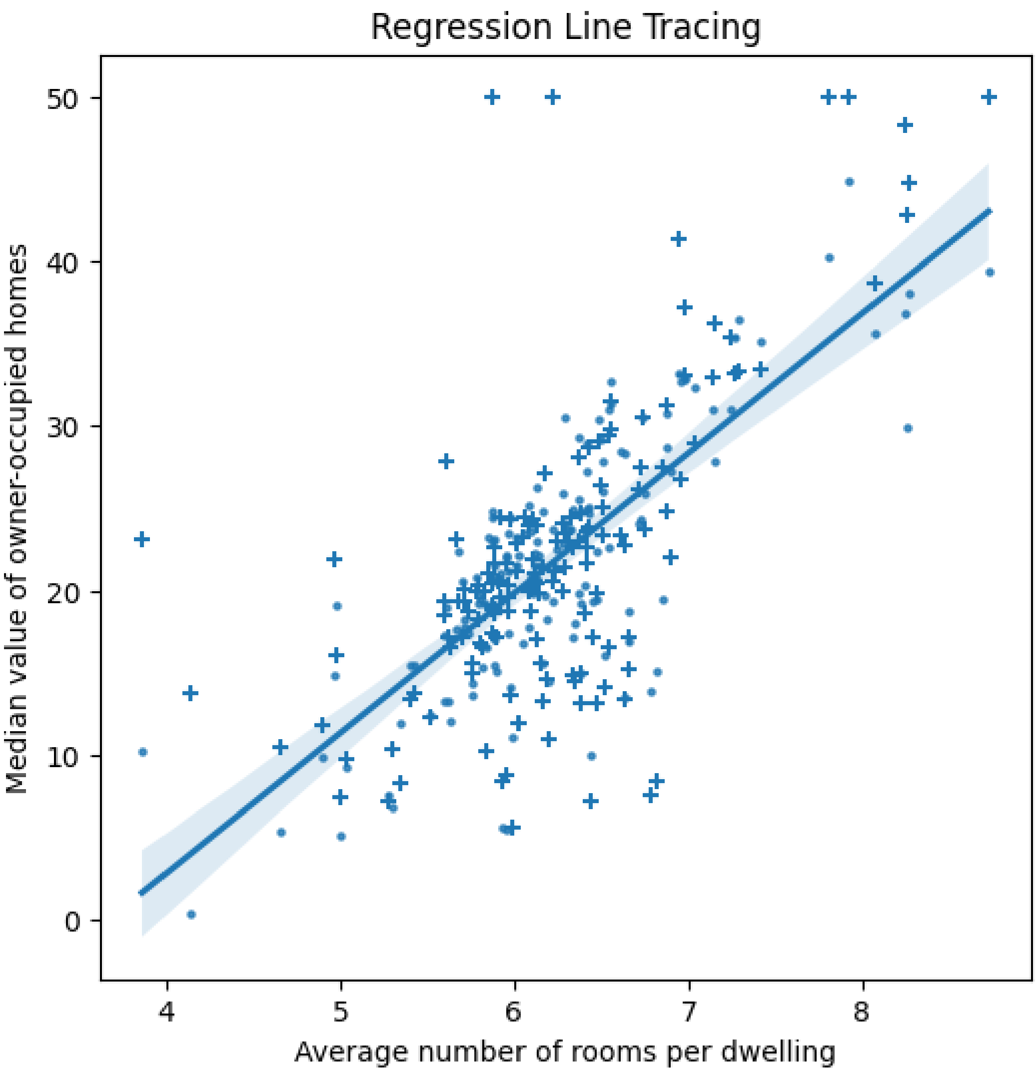


In [21]:

```
plt.figure(figsize=(6, 6))
sns.regplot(x = X_test['RM'], y = predictions, scatter_kws={'s':5})
plt.scatter(X_test['RM'], Y_test, marker = '+')
plt.xlabel('Average number of rooms per dwelling')
plt.ylabel('Median value of owner-occupied homes')
plt.title('Regression Line Tracing')
```

Out[21]:

```
Text(0.5, 1.0, 'Regression Line Tracing')
```



In [23]:

```
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(Y_test, predictions))
print('Mean Square Error:', metrics.mean_squared_error(Y_test, predictions))
print('Root Mean Square Error:', np.sqrt(metrics.mean_squared_error(Y_test, predictions)))
```

```
Mean Absolute Error: 3.609904060381827
Mean Square Error: 27.19596576688351
Root Mean Square Error: 5.2149751453754325
```

In [13]:

```
# Model Coefficients
coefficients = pd.DataFrame(lm.coef_.round(2), X.columns)
coefficients.columns = ['coefficients']
coefficients
```

Out[13]:

	coefficients
CRIM	-0.12
ZN	0.04
INDUS	0.01
CHAS	2.51
NOX	-16.23
RM	3.86
AGE	-0.01
DIS	-1.50
RAD	0.24
TAX	-0.01
PTRATIO	-1.02
BLACK	0.01
LSTAT	-0.49