Name:Chaitanya more

Roll_no:- 01 "B"

Batch :-TB1-B2

Pratical no : 4

Create a Linear Regression Model using Python/R to predict home prices using Boston Housing Dataset (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

In [1]:
```
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
```

In [2]:
```
boston=pd.read_csv("boston.csv")
boston.head()
```

Out[2]:

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	1	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	2	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	3	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	4	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	5	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

In [3]:
```
x=boston.drop(columns=["medv"],axis=1)
y=boston.medv
```

In [4]:
```
x.head()
```

Out[4]:

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat
0	1	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98
1	2	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14
2	3	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03
3	4	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94
4	5	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33

In [5]:
```
x.shape,y.shape
```

Out[5]:
```
((506, 14), (506,))
```

In [6]:
```
x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 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
dtypes: float64(10), int64(4)
memory usage: 55.5 KB
```

In [7]:
```
x.describe()
```

Out[7]:

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax
count	506.000000	506.000000	506.000000	506.000000	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	68.574901	3.795043	9.549407	408.237154
std	146.213884	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116
min	1.000000	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000
25%	127.250000	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000
50%	253.500000	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000
75%	379.750000	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000
max	506.000000	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000

In [8]:
```
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 [9]:
```
y.describe()
```

Out[9]:
```
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 [10]:
```
x.isnull().sum()
```

Out[10]:
```
Unnamed: 0    0
crim          0
zn            0
indus         0
chas          0
nox           0
rm            0
age           0
dis           0
rad           0
tax           0
ptratio       0
black         0
lstat         0
dtype: int64
```

In [11]:
```
y.isnull().sum()
```

Out[11]:
```
0
```

In [12]:
```
df = x
df["target"] = y
df.head()
```

Out[12]:

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	target
0	1	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	2	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	3	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	4	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	5	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

In [13]:
```
df=df[['rm','lstat','target']]
```

In [14]:
```
x=df[['rm','lstat']]
y=df['target']
```

In [15]:
```
scaler=StandardScaler()
```

In [16]:
```
x=scaler.fit_transform(x)
```

In [17]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,shuffle=True)
```

In [18]:
```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

Out[18]:
```
((354, 2), (152, 2), (354,), (152,))
```

In [19]:
```
model=LinearRegression()
```

In [20]:
```
model.fit(x_train,y_train)
```

Out[20]:
```
▾ LinearRegression
LinearRegression()
```

In [21]:
```
y_pred=model.predict(x_test)
```

In [22]:
```
mean_absolute_error(y_test,y_pred)
```

Out[22]:
```
3.6810892592904376
```

In [23]:
```
mean_squared_error(y_test,y_pred)
```

Out[23]:
```
25.17064159568953
```

In [24]:
```
df.head()
```

Out[24]:

	rm	lstat	target
0	6.575	4.98	24.0
1	6.421	9.14	21.6
2	7.185	4.03	34.7
3	6.998	2.94	33.4
4	7.147	5.33	36.2

In [ ]:

In [ ]:

In [ ]: