# atharva-bhide-assignment-4

April 10, 2024

## 1 Import libraries

[3]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_boston
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
```

[4]:

```
boston = load_boston()
boston.keys()
```

[4]:

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

[5]:

```
x = pd.DataFrame(boston.data, columns=boston.feature_names)
y = pd.DataFrame(boston.target, columns=['MEDV'])
```

[6]:

```
x.head()
```

[6]:

```
      CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD    TAX  \
0  0.00632  18.0   2.31   0.0  0.538  6.575  65.2  4.0900  1.0  296.0
1  0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671  2.0  242.0
2  0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671  2.0  242.0
3  0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622  3.0  222.0
4  0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622  3.0  222.0

   PTRATIO       B  LSTAT
0     15.3  396.90   4.98
1     17.8  396.90   9.14
2     17.8  392.83   4.03
```

```
3     18.7  394.63   2.94
4     18.7  396.90   5.33
```

[7]:
```
x.shape, y.shape
```

[7]:
```
((506, 13), (506, 1))
```

## 2 Basic stats

[8]:
```
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    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
dtypes: float64(13)
memory usage: 51.5 KB
```

[9]:
```
x.describe()
```

[9]:
```
             CRIM          ZN       INDUS        CHAS         NOX          RM  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean     3.613524   11.363636   11.136779    0.069170    0.554695    6.284634
std      8.601545   23.322453    6.860353    0.253994    0.115878    0.702617
min      0.006320    0.000000    0.460000    0.000000    0.385000    3.561000
25%      0.082045    0.000000    5.190000    0.000000    0.449000    5.885500
50%      0.256510    0.000000    9.690000    0.000000    0.538000    6.208500
75%      3.677083   12.500000   18.100000    0.000000    0.624000    6.623500
max     88.976200  100.000000   27.740000    1.000000    0.871000    8.780000

              AGE         DIS         RAD         TAX     PTRATIO           B  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
```

```
mean    68.574901    3.795043    9.549407  408.237154   18.455534  356.674032
std     28.148861    2.105710    8.707259  168.537116    2.164946   91.294864
min      2.900000    1.129600    1.000000  187.000000   12.600000    0.320000
25%     45.025000    2.100175    4.000000  279.000000   17.400000  375.377500
50%     77.500000    3.207450    5.000000  330.000000   19.050000  391.440000
75%     94.075000    5.188425   24.000000  666.000000   20.200000  396.225000
max    100.000000   12.126500   24.000000  711.000000   22.000000  396.900000

            LSTAT
count  506.000000
mean    12.653063
std      7.141062
min      1.730000
25%      6.950000
50%     11.360000
75%     16.955000
max     37.970000
```

[10]:
```
y.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   MEDV    506 non-null    float64
dtypes: float64(1)
memory usage: 4.1 KB
```

[11]:
```
y.describe()
```

[11]:
```
             MEDV
count  506.000000
mean    22.532806
std      9.197104
min      5.000000
25%     17.025000
50%     21.200000
75%     25.000000
max     50.000000
```

[12]:
```
x.isnull().sum()
```

[12]:
```
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
```

[13]:
```
y.isnull().sum()
```

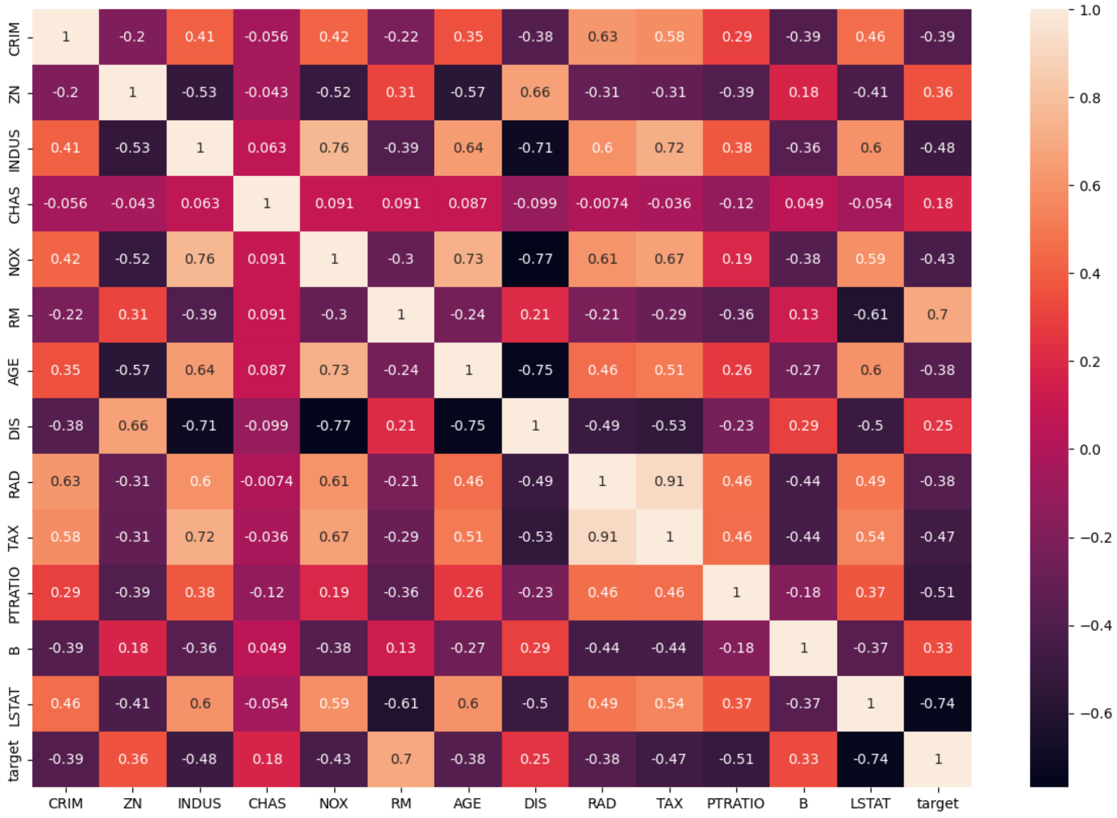
[13]:
```
MEDV    0
dtype: int64
```

[14]:
```
df = x
df["target"] = y
df.head()
```

[14]:
```
      CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD    TAX  \
0  0.00632  18.0   2.31   0.0  0.538  6.575  65.2  4.0900  1.0  296.0
1  0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671  2.0  242.0
2  0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671  2.0  242.0
3  0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622  3.0  222.0
4  0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622  3.0  222.0

   PTRATIO       B  LSTAT  target
0     15.3  396.90   4.98    24.0
1     17.8  396.90   9.14    21.6
2     17.8  392.83   4.03    34.7
3     18.7  394.63   2.94    33.4
4     18.7  396.90   5.33    36.2
```

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

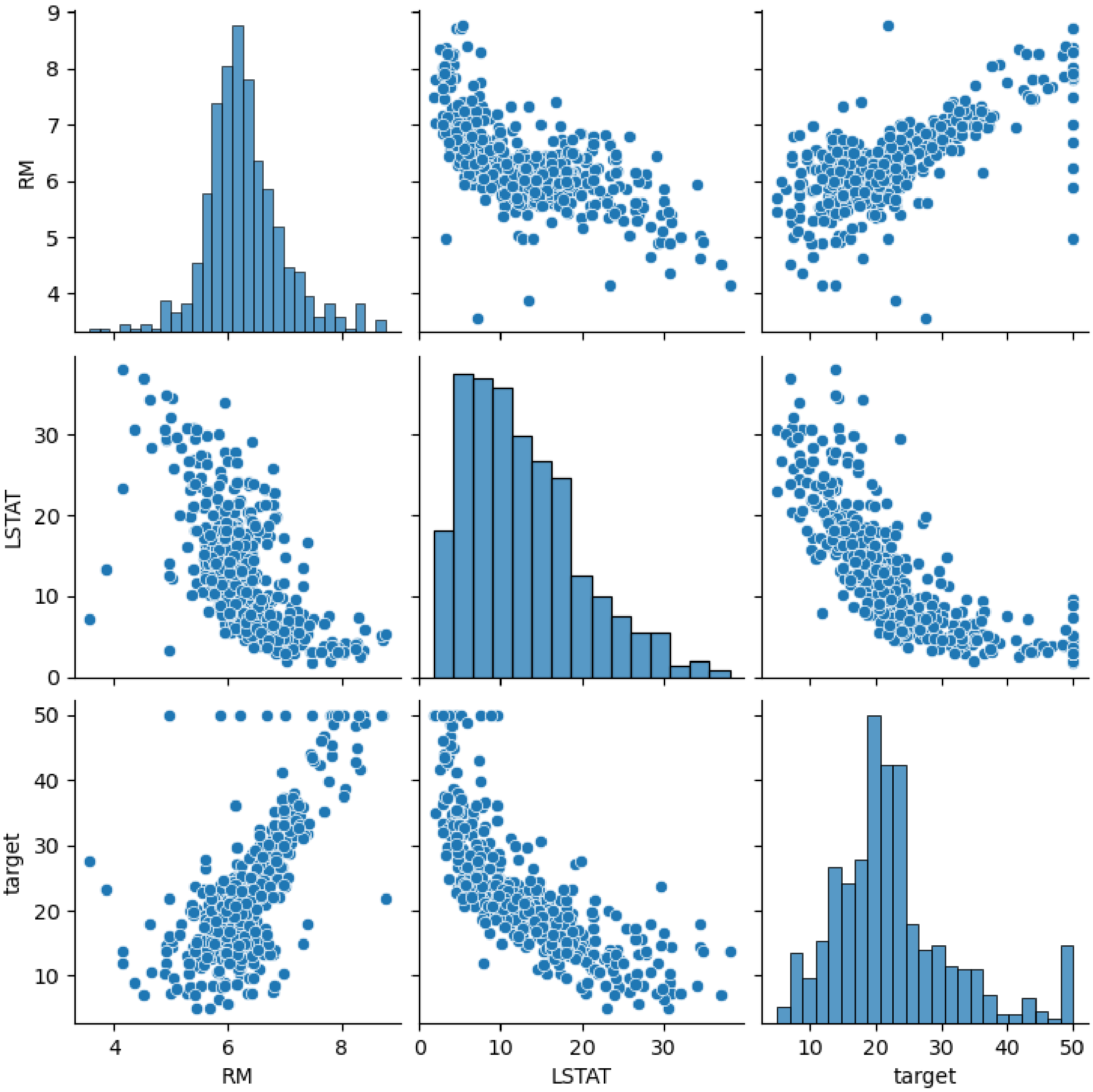


| | CRIM | ZN | INDUS | CHAS | NOX | RM | AGE | DIS | RAD | TAX | PTRATIO | B | LSTAT | target |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CRIM | 1 | -0.2 | 0.41 | -0.056 | 0.42 | -0.22 | 0.35 | -0.38 | 0.63 | 0.58 | 0.29 | -0.39 | 0.46 | -0.39 |
| ZN | -0.2 | 1 | -0.53 | -0.043 | -0.52 | 0.31 | -0.57 | 0.66 | -0.31 | -0.31 | -0.39 | 0.18 | -0.41 | 0.36 |
| INDUS | 0.41 | -0.53 | 1 | 0.063 | 0.76 | -0.39 | 0.64 | -0.71 | 0.6 | 0.72 | 0.38 | -0.36 | 0.6 | -0.48 |
| CHAS | -0.056 | -0.043 | 0.063 | 1 | 0.091 | 0.091 | 0.087 | -0.099 | -0.0074 | -0.036 | -0.12 | 0.049 | -0.054 | 0.18 |
| NOX | 0.42 | -0.52 | 0.76 | 0.091 | 1 | -0.3 | 0.73 | -0.77 | 0.61 | 0.67 | 0.19 | -0.38 | 0.59 | -0.43 |
| RM | -0.22 | 0.31 | -0.39 | 0.091 | -0.3 | 1 | -0.24 | 0.21 | -0.21 | -0.29 | -0.36 | 0.13 | -0.61 | 0.7 |
| AGE | 0.35 | -0.57 | 0.64 | 0.087 | 0.73 | -0.24 | 1 | -0.75 | 0.46 | 0.51 | 0.26 | -0.27 | 0.6 | -0.38 |
| DIS | -0.38 | 0.66 | -0.71 | -0.099 | -0.77 | 0.21 | -0.75 | 1 | -0.49 | -0.53 | -0.23 | 0.29 | -0.5 | 0.25 |
| RAD | 0.63 | -0.31 | 0.6 | -0.0074 | 0.61 | -0.21 | 0.46 | -0.49 | 1 | 0.91 | 0.46 | -0.44 | 0.49 | -0.38 |
| TAX | 0.58 | -0.31 | 0.72 | -0.036 | 0.67 | -0.29 | 0.51 | -0.53 | 0.91 | 1 | 0.46 | -0.44 | 0.54 | -0.47 |
| PTRATIO | 0.29 | -0.39 | 0.38 | -0.12 | 0.19 | -0.36 | 0.26 | -0.23 | 0.46 | 0.46 | 1 | -0.18 | 0.37 | -0.51 |
| B | -0.39 | 0.18 | -0.36 | 0.049 | -0.38 | 0.13 | -0.27 | 0.29 | -0.44 | -0.44 | -0.18 | 1 | -0.37 | 0.33 |
| LSTAT | 0.46 | -0.41 | 0.6 | -0.054 | 0.59 | -0.61 | 0.6 | -0.5 | 0.49 | 0.54 | 0.37 | -0.37 | 1 | -0.74 |
| target | -0.39 | 0.36 | -0.48 | 0.18 | -0.43 | 0.7 | -0.38 | 0.25 | -0.38 | -0.47 | -0.51 | 0.33 | -0.74 | 1 |

### 2.0.1 Considering only ‘RM’ and ‘LSTAT’ by considering correlation and multi-collinearity of other features

[51]:
```
df = df[['RM', 'LSTAT', 'target']]
```

[52]:
```
sns.pairplot(df)
plt.show()
```



[63]:

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

# 3 Scale the data

[64]:

```
scaler = StandardScaler()
```

[65]:

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

## 4 Split the data

[67]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, shuffle=True)
```

[68]:
```
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

[68]:
```
((354, 2), (152, 2), (354,), (152,))
```

## 5 Linear Regression Modelling

[69]:
```
model = LinearRegression(n_jobs=-1)
```

[70]:
```
model.fit(x_train, y_train)
```

[70]:
```
LinearRegression(n_jobs=-1)
```

## 6 Make predictions

[71]:
```
y_pred = model.predict(x_test)
```

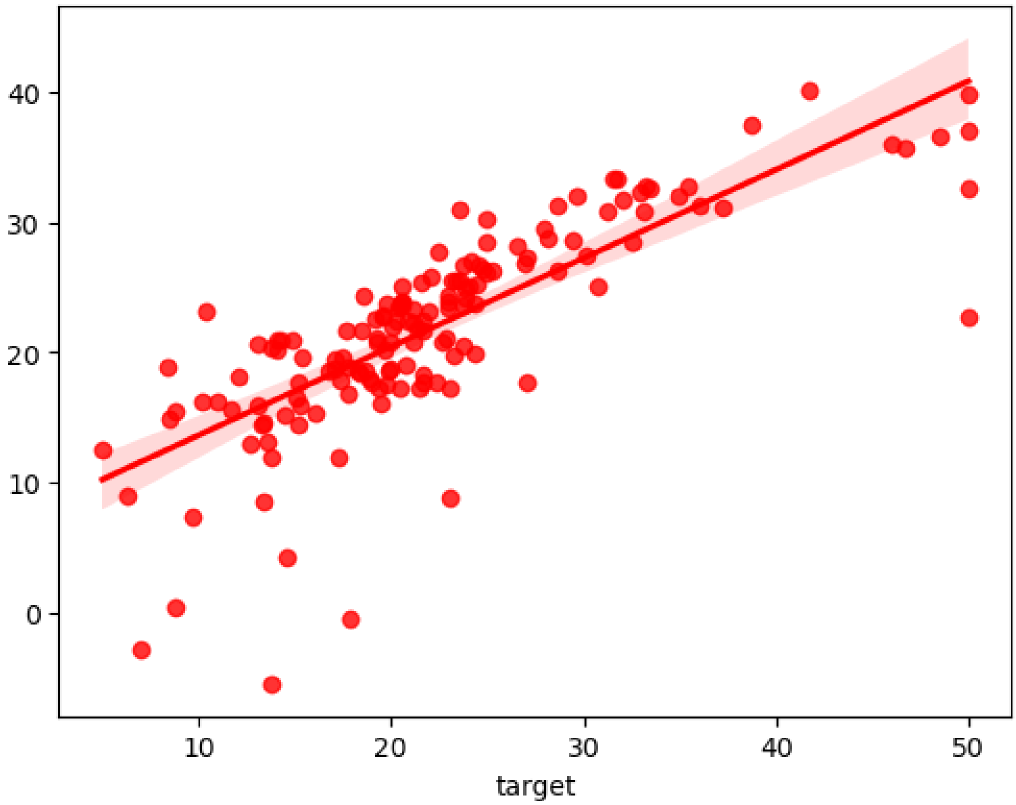
[72]:
```
mean_absolute_error(y_test, y_pred)
```

[72]:
```
3.701010266760501
```

[73]:
```
mean_squared_error(y_test, y_pred)
```

[73]:
```
30.5001478179898
```

[74]:
```
sns.regplot(y_test, y_pred, color='red')
plt.show()
```


40
30
20
10
0
10
20
30
40
50
target