

Madhur Jaripatke

Roll No. 48

TE A Computer

RMDSSOE, Warje, Pune

4. Data Analytics, I

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 matplotlib.pyplot as plt
```

Importing the dataset

```
In [2]: df = pd.read_csv('Datasets/HousingData.csv')
df
```

```
Out[2]:
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1	296	15.3	396
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2	242	17.8	396
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2	242	17.8	396
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3	222	18.7	396
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3	222	18.7	396
...
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1	273	21.0	396
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1	273	21.0	396
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1	273	21.0	396
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1	273	21.0	396
505	0.04741	0.0	11.93	0.0	0.573	6.030	NaN	2.5050	1	273	21.0	396

506 rows × 14 columns



Data Preprocessing

```
In [3]: df.shape
```

```
Out[3]: (506, 14)
```

```
In [4]: df.isnull().sum()
```

```
Out[4]: 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 [5]: df['CRIM'] = df['CRIM'].fillna(df['CRIM'].mean())  
df['ZN'] = df['ZN'].fillna(df['ZN'].mean())  
df['INDUS'] = df['INDUS'].fillna(df['INDUS'].mean())  
df['CHAS'] = df['CHAS'].fillna(df['CHAS'].mean())  
df['AGE'] = df['AGE'].fillna(df['NOX'].mean())  
df['LSTAT'] = df['LSTAT'].fillna(df['LSTAT'].mean())  
df
```

```
Out[5]:
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.200000	4.0900	1	296	15.3	0.63	4.97
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.900000	4.9671	2	242	17.8	0.62	4.76
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.100000	4.9671	2	242	17.8	0.62	4.76
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.800000	6.0622	3	222	18.7	0.62	4.53
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.200000	6.0622	3	222	18.7	0.62	4.53
...
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.100000	2.4786	1	273	21.0	0.62	4.03
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.700000	2.2875	1	273	21.0	0.62	4.03
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.000000	2.1675	1	273	21.0	0.62	4.03
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.300000	2.3889	1	273	21.0	0.62	4.03
505	0.04741	0.0	11.93	0.0	0.573	6.030	0.554695	2.5050	1	273	21.0	0.62	4.03

506 rows × 14 columns

```
In [6]: df.isnull().sum()
```

```
Out[6]: 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 [7]: x = df.drop('MEDV', axis=1)
x
```

Out[7]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	L
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.200000	4.0900	1	296	15.3		
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.900000	4.9671	2	242	17.8		
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.100000	4.9671	2	242	17.8		
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.800000	6.0622	3	222	18.7		
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.200000	6.0622	3	222	18.7		
...
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.100000	2.4786	1	273	21.0		
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.700000	2.2875	1	273	21.0		
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.000000	2.1675	1	273	21.0		
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.300000	2.3889	1	273	21.0		
505	0.04741	0.0	11.93	0.0	0.573	6.030	0.554695	2.5050	1	273	21.0		

506 rows × 13 columns



In [8]: `y = df['MEDV']`
`y`

Out[8]:

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

Splitting the dataset into Training set & Test set

In [9]: `from sklearn.model_selection import train_test_split`
`x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_`

In [10]: `x_train`

Out[10]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRA
220	0.358090	0.0	6.200000	1.000000	0.507	6.951	88.5	2.8617	8	307	1
71	0.158760	0.0	10.810000	0.000000	0.413	5.961	17.5	5.2873	4	305	1
240	0.113290	30.0	4.930000	0.069959	0.428	6.897	54.3	6.3361	6	300	1
6	0.088290	12.5	7.870000	0.069959	0.524	6.012	66.6	5.5605	5	311	1
417	25.940600	0.0	18.100000	0.000000	0.679	5.304	89.1	1.6475	24	666	2
...
323	0.283920	0.0	7.380000	0.000000	0.493	5.708	74.3	4.7211	5	287	1
192	3.611874	45.0	3.440000	0.000000	0.437	7.178	26.3	6.4798	5	398	1
117	0.150980	0.0	10.010000	0.000000	0.547	6.021	82.6	2.7474	6	432	1
47	0.229270	0.0	11.083992	0.000000	0.448	6.030	85.5	5.6894	3	233	1
172	0.139140	0.0	4.050000	0.000000	0.510	5.572	88.5	2.5961	5	296	1

404 rows × 13 columns



Training the Linear Regression Model

In [11]:

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x_train.values, y_train.values)
```

Out[11]:

LinearRegression ⓘ ?

LinearRegression()

In [12]:

```
y_train_pred = model.predict(x_train.values)
y_test_pred = model.predict(x_test.values)
```

In [13]:

```
y_train_pred
```

```
Out[13]: array([32.66475357, 22.54760255, 27.92272071, 23.625425 , 6.45861412,
13.93671818, 21.99813652, 29.37255546, 32.26120163, 12.96973779,
19.94017093, 21.45691431, 12.8839238 , 23.92943611, 5.97521446,
18.93574189, 9.22524241, 45.33468083, 30.74460224, 17.25459323,
17.76350511, 21.90996473, 23.29496036, 18.97171185, 34.91668639,
13.50648784, 20.87214382, 35.52772443, 19.06816761, 13.54186364,
13.76659184, 22.07218103, 14.98269752, 31.34278627, 25.39744809,
16.1870452 , 24.86653661, 9.63533679, 15.04079584, 21.9231712 ,
33.08970263, 28.30793582, 26.6815381 , 15.31770869, 31.83213513,
25.30973944, 14.15787572, 7.92149041, 27.76577059, 25.30401239,
4.99697272, 28.22349867, 16.77086549, 29.86368234, 19.11821305,
16.15376997, 18.41077744, 12.7567695 , 8.79391708, 19.05618572,
35.71550638, 32.72210532, 23.71015905, 20.18648812, 22.93000843,
26.44974296, 21.0615435 , 17.69830015, 32.39856766, 10.49365749,
19.10820204, 31.36688112, 18.96836882, 15.76882773, 18.94519156,
15.15847982, 24.11855199, 23.38207934, 17.3295206 , 13.10475782,
20.54654327, 23.99183807, 17.46003754, 25.57826811, 22.75564999,
27.96955482, 36.79911276, 16.29320092, 11.96928193, 34.98157367,
31.2956174 , 20.32883487, 39.91898815, 28.63629469, 28.52460167,
17.68554714, 26.84504736, 40.39261223, 27.46393012, 17.137448 ,
37.46198772, 35.70706734, 14.06849289, 27.49265535, 24.01488559,
24.968266 , 21.24194434, 23.22159002, 27.94396566, 29.45052215,
13.83197008, 26.09853134, 22.80277822, 15.43676479, 13.97762078,
25.50947859, 19.48687456, 30.44350731, 9.85025779, 24.27331225,
17.03316039, 16.85546978, 22.75870841, 21.67495995, 12.0451091 ,
25.08931078, 28.51617952, 22.65604474, 12.19565184, 24.93552503,
26.58866568, 25.7705037 , 23.51770572, 25.60406486, 19.33663113,
20.71038807, 36.06303681, 20.99110962, 36.30038206, 25.71105746,
20.83663618, 15.40620166, 32.04196757, 21.32944187, 28.06820878,
16.71628892, 32.72246926, 14.11501893, 1.48726866, 19.44446647,
13.70894591, 37.55581629, 16.17814528, 14.35177526, 26.62492959,
23.51386813, 17.58187046, 31.19908409, 25.40403936, 27.60077495,
24.60481019, 22.86284304, 22.41781158, 11.11394138, 20.73667755,
11.53702529, 17.62501807, 12.30439483, 27.55821134, 14.98300245,
15.86027234, 28.61576048, 14.2952606 , 21.67971867, 12.6258741 ,
16.4079762 , 23.3856092 , 21.14944384, 14.80325171, 17.41895461,
14.95913785, 26.00657216, 12.54531255, 35.34831489, 14.44057714,
43.19084197, 31.60225234, 34.73011197, 22.06549381, 15.60881839,
26.86711745, 29.08585412, 13.6020358 , 26.62936499, 36.01124274,
16.70502674, 11.53256998, 34.49979514, 36.02843502, 17.94009918,
21.23466028, 20.40463658, 24.44503261, 19.5133743 , 27.09874464,
-4.35147473, 20.83165704, 32.92028686, 35.38683579, 25.09427472,
26.70282143, 20.19959097, 21.47888434, 16.00569814, 17.82411768,
21.22008077, 27.93097558, 19.83756506, 6.96957231, 16.07007846,
32.24463684, 35.49232568, 16.35221777, 18.86660548, 22.33681004,
6.36701386, 21.42824658, 23.58756671, 15.85947528, 18.40347802,
23.21231647, 27.04107566, 25.85228541, 32.73542374, 14.68833813,
28.93776787, 24.93274252, 20.81833898, 38.5230573 , 21.98860617,
23.59465047, 22.5915317 , 12.04014312, 20.06564025, 33.38017755,
24.71481958, 17.76569636, 33.11025806, 22.08990019, 28.73314095,
32.06058555, 36.47410146, 21.66024396, 24.05946361, 23.1696524 ,
32.0392582 , 22.19777583, 18.26850685, 21.89849598, 29.33625032,
22.89607891, 21.98997098, 16.9860062 , 17.32403378, 16.95639466,
16.85650119, 16.67797165, 31.93962811, 23.06540444, 17.48840171,
19.06901983, 34.20518995, 13.92961901, 26.13376776, 16.98878425,
30.66239417, 29.94233138, 23.11218925, 20.15856534, 36.1991215 ,
20.53732926, 33.46685682, 21.36666915, 31.50394589, 30.09643429,
37.30345689, 25.71187931, 20.88627074, 29.11779833, 15.97053177,
25.91460539, 21.48266931, 29.92258458, 10.29951201, 31.2335619 ,
6.23492083, 15.21271299, 20.37088814, 35.76335679, 31.79152748,
```

```
12.22466543, 13.67981558, 21.98115843, 34.69693923, 18.75887204,
18.38805663, 14.72038213, 25.298637 , 40.95303912, 25.39285876,
42.04574424, 25.45406537, 20.88512916, 11.87425463, 15.8524752 ,
14.04805806, 18.44010864, 3.0485567 , 27.64307522, 26.43528674,
41.75366915, 21.65531543, 21.07349544, 34.06526064, 32.73156545,
9.46651233, 24.80881062, 43.77303166, 21.73707107, 17.66971023,
26.3438443 , 18.42979496, 6.36551293, 18.82126102, 35.63631667,
16.10479431, 23.92928569, 13.10217538, 24.45122275, 18.1733537 ,
17.16243484, 18.27651925, 32.98016225, 19.24753909, 29.60002147,
31.76788565, 43.45274978, 18.39829662, 15.77094546, 38.32218921,
17.48890093, 10.4567689 , 14.58291612, 25.32215453, 19.39038299,
16.3931107 , 26.4981961 , 15.21641805, 5.99624195, 18.72341049,
11.00565791, 28.43940705, 4.77977595, 28.49842529, 32.7749635 ,
22.71279076, 16.39172307, 17.70378296, 21.20658832, 33.91580167,
28.3465387 , 19.40882851, 20.33459157, 6.88538961, 29.06990391,
25.17119662, 22.45813448, 13.53193 , 24.48063603, 19.39198825,
8.76491126, 26.75622464, 15.84398325, 31.60691698, 33.593711 ,
24.99304656, 18.41963172, 30.42821065, 21.24240722, 26.09533173,
24.3152015 , 31.15075678, 24.44459048, 31.50609222, 17.48431929,
19.54265449, 18.76350478, 41.32674441, 25.65378283, 19.17369501,
33.43036639, 23.57405103, 18.03491836, 23.12408505])
```

```
In [14]: model.predict([[0.00632, 18.0, 2.31, 0.0, 0.538, 6.575, 65.2, 4.0900, 1.0, 296.0
```

```
Out[14]: array([30.66239417])
```

```
In [15]: y_test_pred
```

```
Out[15]: array([26.38244903, 22.48142477, 28.9150927 , 11.51311019, 21.55596144,
19.39569936, 20.24684687, 21.42696075, 19.34116377, 19.68860665,
4.16870305, 15.92252628, 16.82679893, 5.33336826, 39.08414873,
33.13236137, 21.86276797, 36.51777073, 31.70654122, 23.61115389,
24.93638896, 23.41076277, 20.78702452, 30.60639121, 22.78779124,
8.34225949, 17.46512364, 17.83410415, 35.88821362, 21.01664412,
17.72326132, 17.42852689, 19.15036073, 23.30738941, 30.794513 ,
19.22448935, 11.2795003 , 23.85638994, 17.65920118, 15.27836886,
26.42841366, 21.60671364, 23.86193607, 14.66954994, 23.89469709,
24.73879055, 19.95659189, 23.04414555, 10.49937272, 24.42914586,
23.54775421, 18.98321991, 24.46540719, 31.13192075, 12.69208308,
22.46866718, 21.36471519, 15.97301174, 12.04261875, 22.54049973,
18.20501757, 21.91741417, 32.62538174, 31.43041916, 17.56480551,
33.25033675, 21.22233526, 19.67388402, 19.97829012, 24.072029 ,
22.59941828, 24.13111571, 30.83037261, 28.84034313, 25.01388616,
5.68203177, 38.85648038, 24.12681527, 27.6468537 , 19.64454572,
28.67492372, 18.62889753, 17.47846418, 37.87598241, 39.37598 ,
24.26550781, 25.2577793 , 16.66424155, 25.32136054, 16.69295872,
16.37096093, 13.29262768, 24.62082096, 31.00936186, 23.93931001,
20.41960539, 0.89355306, 25.32071273, 15.40869795, 17.51079031,
25.88556368, 22.24344716])
```

```
In [16]: y_train
```

```
Out[16]: 220    26.7
          71     21.7
          240    22.0
           6     22.9
          417    10.4
          ...
          323    18.5
          192    36.4
          117    19.2
           47    16.6
          172    23.1
          Name: MEDV, Length: 404, dtype: float64
```

```
In [17]: y_test
```

```
Out[17]: 329    22.6
          371    50.0
          219    23.0
          403     8.3
           78    21.2
          ...
           56    24.7
          455    14.1
           60    18.7
          213    28.1
          108    19.8
          Name: MEDV, Length: 102, dtype: float64
```

```
In [18]: df1 = pd.DataFrame({'Actual': y_train, 'Predicted': y_train_pred})
          df2 = pd.DataFrame({'Actual': y_test, 'Predicted': y_test_pred})
```

```
In [19]: df1
```

```
Out[19]:
```

	Actual	Predicted
220	26.7	32.664754
71	21.7	22.547603
240	22.0	27.922721
6	22.9	23.625425
417	10.4	6.458614
...
323	18.5	19.173695
192	36.4	33.430366
117	19.2	23.574051
47	16.6	18.034918
172	23.1	23.124085

404 rows × 2 columns

```
In [20]: df2
```


Out[20]:

	Actual	Predicted
329	22.6	26.382449
371	50.0	22.481425
219	23.0	28.915093
403	8.3	11.513110
78	21.2	21.555961
...
56	24.7	25.320713
455	14.1	15.408698
60	18.7	17.510790
213	28.1	25.885564
108	19.8	22.243447

102 rows × 2 columns

```
In [21]: from sklearn.metrics import mean_squared_error, r2_score
mse = mean_squared_error(y_test, y_test_pred)
mse
```

Out[21]: 35.893643434646215

```
In [22]: mse = mean_squared_error(y_train, y_train_pred)
mse
```

Out[22]: 19.805844906182962

```
In [23]: r2 = r2_score(y_test, y_test_pred)
r2
```

Out[23]: 0.5592001535561669

Visualizing the Results

```
In [24]: plt.scatter(y_test, y_test_pred, c='red', marker='o', label='Test Data')
plt.scatter(y_train, y_train_pred, c='blue', marker='s', label='Train Data')
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
plt.title('Actual Prices vs Predicted Prices')
plt.legend(loc='upper left')
plt.plot()
plt.show()
```

Actual Prices vs Predicted Prices

