practical-04

April 26, 2024

```
[42]: import numpy as np
      import pandas as pd
      from sklearn.model_selection import train_test_split
      from sklearn.linear_model import LinearRegression
      from sklearn.metrics import mean_squared_error, r2_score
```

```
[3]: data = pd.read_csv("C:/Users/gugal/Desktop/THIRD 2/PRACTICALS/DS/CODES/DATASETS/
      ↪HousingData.csv")
     data
```

```
[3]:         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  296
     1    0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671    2  242
     2    0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671    2  242
     3    0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622    3  222
     4    0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622    3  222
     ..       …    …     …     …     …    …      …   …    …
     501  0.06263   0.0  11.93   0.0  0.573  6.593  69.1  2.4786    1  273
     502  0.04527   0.0  11.93   0.0  0.573  6.120  76.7  2.2875    1  273
     503  0.06076   0.0  11.93   0.0  0.573  6.976  91.0  2.1675    1  273
     504  0.10959   0.0  11.93   0.0  0.573  6.794  89.3  2.3889    1  273
     505  0.04741   0.0  11.93   0.0  0.573  6.030   NaN  2.5050    1  273

          PTRATIO       B  LSTAT  MEDV
     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    NaN  36.2
     ..        …       …     …    …
     501     21.0  391.99    NaN  22.4
     502     21.0  396.90   9.08  20.6
     503     21.0  396.90   5.64  23.9
     504     21.0  393.45   6.48  22.0
     505     21.0  396.90   7.88  11.9

     [506 rows x 14 columns]
```

[8]:
```
data.head()
```

[8]:
```
      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
1  0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671    2  242     17.8
2  0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671    2  242     17.8
3  0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622    3  222     18.7
4  0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622    3  222     18.7

        B  LSTAT  MEDV
0  396.90   4.98  24.0
1  396.90   9.14  21.6
2  392.83   4.03  34.7
3  394.63   2.94  33.4
4  396.90    NaN  36.2
```

[4]:
```
data.isnull().sum()
```

[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
```

[22]:
```
data.fillna(data.mean(), inplace=True)
```

[23]:
```
data
```

[23]:
```
        CRIM    ZN  INDUS  CHAS    NOX     RM        AGE     DIS  RAD  TAX  \
0    0.00632  18.0   2.31   0.0  0.538  6.575  65.200000  4.0900    1  296
1    0.02731   0.0   7.07   0.0  0.469  6.421  78.900000  4.9671    2  242
2    0.02729   0.0   7.07   0.0  0.469  7.185  61.100000  4.9671    2  242
3    0.03237   0.0   2.18   0.0  0.458  6.998  45.800000  6.0622    3  222
4    0.06905   0.0   2.18   0.0  0.458  7.147  54.200000  6.0622    3  222
..       …     …     …     …      …      …          …       …   …    …
501  0.06263   0.0  11.93   0.0  0.573  6.593  69.100000  2.4786    1  273
502  0.04527   0.0  11.93   0.0  0.573  6.120  76.700000  2.2875    1  273
```

```
503  0.06076   0.0  11.93   0.0  0.573  6.976  91.000000  2.1675    1  273
504  0.10959   0.0  11.93   0.0  0.573  6.794  89.300000  2.3889    1  273
505  0.04741   0.0  11.93   0.0  0.573  6.030  68.518519  2.5050    1  273

     PTRATIO       B      LSTAT  MEDV
0       15.3  396.90   4.980000  24.0
1       17.8  396.90   9.140000  21.6
2       17.8  392.83   4.030000  34.7
3       18.7  394.63   2.940000  33.4
4       18.7  396.90  12.715432  36.2
..       …       …         …    …
501     21.0  391.99  12.715432  22.4
502     21.0  396.90   9.080000  20.6
503     21.0  396.90   5.640000  23.9
504     21.0  393.45   6.480000  22.0
505     21.0  396.90   7.880000  11.9

[506 rows x 14 columns]
```

[24]:
```
data.isnull().sum()
```

[24]:
```
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
```

[25]:
```
data.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    int64
 9   TAX      506 non-null    int64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 55.5 KB
```

[26]:
```
data.columns
```

[26]:
```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT', 'MEDV'],
      dtype='object')
```

[27]:
```
data.describe()
```

[27]:
```
             CRIM          ZN       INDUS        CHAS         NOX          RM  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean     3.611874   11.211934   11.083992    0.069959    0.554695    6.284634
std      8.545770   22.921051    6.699165    0.250233    0.115878    0.702617
min      0.006320    0.000000    0.460000    0.000000    0.385000    3.561000
25%      0.083235    0.000000    5.190000    0.000000    0.449000    5.885500
50%      0.290250    0.000000    9.900000    0.000000    0.538000    6.208500
75%      3.611874   11.211934   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    68.518519    3.795043    9.549407  408.237154   18.455534  356.674032
std     27.439466    2.105710    8.707259  168.537116    2.164946   91.294864
min      2.900000    1.129600    1.000000  187.000000   12.600000    0.320000
25%     45.925000    2.100175    4.000000  279.000000   17.400000  375.377500
50%     74.450000    3.207450    5.000000  330.000000   19.050000  391.440000
75%     93.575000    5.188425   24.000000  666.000000   20.200000  396.225000
max    100.000000   12.126500   24.000000  711.000000   22.000000  396.900000

            LSTAT        MEDV
count  506.000000  506.000000
mean    12.715432   22.532806
std      7.012739    9.197104
min      1.730000    5.000000
```

```
25%      7.230000   17.025000
50%     11.995000   21.200000
75%     16.570000   25.000000
max     37.970000   50.000000
```

[28]:
```
X = data.drop('MEDV', axis=1)
Y = data['MEDV']
```

[29]:
```
X.columns
```

[29]:
```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT'],
      dtype='object')
```

[30]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,␣
 ↪random_state=42)
```

[31]:
```
print(len(Y_test))
```

```
102
```

[32]:
```
model = LinearRegression()
model.fit(X_train,Y_train)
```

[32]:
```
LinearRegression()
```

[37]:
```
prediction = model.predict(X_test)
```

[41]:
```
mse = mean_squared_error(Y_test,prediction)
mse
```

[41]:
```
25.017672023842703
```

[43]:
```
rmse = np.sqrt(mse)
rmse
```

[43]:
```
5.001766890194174
```

[45]:
```
r2 = r2_score(Y_test,prediction)
r2
```

[45]:
```
0.658852019550814
```