linear_regression

July 22, 2020

1 Linear regression exercises

1.1 Prepare modules and data.

[1]:
```
# Import the necessary modules.
from sklearn.datasets import load_boston
from pandas import DataFrame
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d.axes3d import Axes3D
```

[2]:
```
# Get the Boston housing dataset.
boston = load_boston()
```

[3]:
```
# Check the key of the data.
boston.keys()
```

[3]:
```
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename'])
```

[4]:
```
# Check the description of the data.
print(boston['DESCR'])
```

```
.. _boston_dataset:

Boston house prices dataset
---------------------------

**Data Set Characteristics:**

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive. Median Value
(attribute 14) is usually the target.

    :Attribute Information (in order):
        - 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

    :Missing Attribute Values: None

    :Creator: Harrison, D. and Rubinfeld, D.L.

This is a copy of UCI ML housing dataset.
https://archive.ics.uci.edu/ml/machine-learning-databases/housing/


This dataset was taken from the StatLib library which is maintained at Carnegie
Mellon University.

The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic
prices and the demand for clean air', J. Environ. Economics & Management,
vol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regression diagnostics
…', Wiley, 1980.   N.B. Various transformations are used in the table on
pages 244-261 of the latter.

The Boston house-price data has been used in many machine learning papers that
address regression
problems.

.. topic:: References

   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influential
Data and Sources of Collinearity', Wiley, 1980. 244-261.
   - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learning. In
Proceedings on the Tenth International Conference of Machine Learning, 236-243,
University of Massachusetts, Amherst. Morgan Kaufmann.
```

[5]:
```
# Check the contents of the feature_names variable.
print(boston['feature_names'])
```

```
['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO'
 'B' 'LSTAT']
```

[6]:
```
# Check the contents of the explanatory variables.
print(boston['data'])
```

```
[[6.3200e-03 1.8000e+01 2.3100e+00 … 1.5300e+01 3.9690e+02 4.9800e+00]
 [2.7310e-02 0.0000e+00 7.0700e+00 … 1.7800e+01 3.9690e+02 9.1400e+00]
 [2.7290e-02 0.0000e+00 7.0700e+00 … 1.7800e+01 3.9283e+02 4.0300e+00]
 …
 [6.0760e-02 0.0000e+00 1.1930e+01 … 2.1000e+01 3.9690e+02 5.6400e+00]
 [1.0959e-01 0.0000e+00 1.1930e+01 … 2.1000e+01 3.9345e+02 6.4800e+00]
 [4.7410e-02 0.0000e+00 1.1930e+01 … 2.1000e+01 3.9690e+02 7.8800e+00]]
```

[7]:
```
# Check the contents of the target variable.
print(boston['target'])
```

```
[24.  21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 15.  18.9 21.7 20.4
 18.2 19.9 23.1 17.5 20.2 18.2 13.6 19.6 15.2 14.5 15.6 13.9 16.6 14.8
 18.4 21.  12.7 14.5 13.2 13.1 13.5 18.9 20.  21.  24.7 30.8 34.9 26.6
 25.3 24.7 21.2 19.3 20.  16.6 14.4 19.4 19.7 20.5 25.  23.4 18.9 35.4
 24.7 31.6 23.3 19.6 18.7 16.  22.2 25.  33.  23.5 19.4 22.  17.4 20.9
 24.2 21.7 22.8 23.4 24.1 21.4 20.  20.8 21.2 20.3 28.  23.9 24.8 22.9
 23.9 26.6 22.5 22.2 23.6 28.7 22.6 22.  22.9 25.  20.6 28.4 21.4 38.7
 43.8 33.2 27.5 26.5 18.6 19.3 20.1 19.5 19.5 20.4 19.8 19.4 21.7 22.8
 18.8 18.7 18.5 18.3 21.2 19.2 20.4 19.3 22.  20.3 20.5 17.3 18.8 21.4
 15.7 16.2 18.  14.3 19.2 19.6 23.  18.4 15.6 18.1 17.4 17.1 13.3 17.8
 14.  14.4 13.4 15.6 11.8 13.8 15.6 14.6 17.8 15.4 21.5 19.6 15.3 19.4
 17.  15.6 13.1 41.3 24.3 23.3 27.  50.  50.  50.  22.7 25.  50.  23.8
 23.8 22.3 17.4 19.1 23.1 23.6 22.6 29.4 23.2 24.6 29.9 37.2 39.8 36.2
 37.9 32.5 26.4 29.6 50.  32.  29.8 34.9 37.  30.5 36.4 31.1 29.1 50.
 33.3 30.3 34.6 34.9 32.9 24.1 42.3 48.5 50.  22.6 24.4 22.5 24.4 20.
 21.7 19.3 22.4 28.1 23.7 25.  23.3 28.7 21.5 23.  26.7 21.7 27.5 30.1
 44.8 50.  37.6 31.6 46.7 31.5 24.3 31.7 41.7 48.3 29.  24.  25.1 31.5
 23.7 23.3 22.  20.1 22.2 23.7 17.6 18.5 24.3 20.5 24.5 26.2 24.4 24.8
 29.6 42.8 21.9 20.9 44.  50.  36.  30.1 33.8 43.1 48.8 31.  36.5 22.8
 30.7 50.  43.5 20.7 21.1 25.2 24.4 35.2 32.4 32.  33.2 33.1 29.1 35.1
 45.4 35.4 46.  50.  32.2 22.  20.1 23.2 22.3 24.8 28.5 37.3 27.9 23.9
 21.7 28.6 27.1 20.3 22.5 29.  24.8 22.  26.4 33.1 36.1 28.4 33.4 28.2
 22.8 20.3 16.1 22.1 19.4 21.6 23.8 16.2 17.8 19.8 23.1 21.  23.8 23.1
 20.4 18.5 25.  24.6 23.  22.2 19.3 22.6 19.8 17.1 19.4 22.2 20.7 21.1
 19.5 18.5 20.6 19.  18.7 32.7 16.5 23.9 31.2 17.5 17.2 23.1 24.5 26.6
 22.9 24.1 18.6 30.1 18.2 20.6 17.8 21.7 22.7 22.6 25.  19.9 20.8 16.8
 21.9 27.5 21.9 23.1 50.  50.  50.  50.  50.  13.8 13.8 15.  13.9 13.3
 13.1 10.2 10.4 10.9 11.3 12.3  8.8  7.2 10.5  7.4 10.2 11.5 15.1 23.2
  9.7 13.8 12.7 13.1 12.5  8.5  5.   6.3  5.6  7.2 12.1  8.3  8.5  5.
 11.9 27.9 17.2 27.5 15.  17.2 17.9 16.3  7.   7.2  7.5 10.4  8.8  8.4
 16.7 14.2 20.8 13.4 11.7  8.3 10.2 10.9 11.   9.5 14.5 14.1 16.1 14.3
```

```
 11.7 13.4  9.6  8.7  8.4 12.8 10.5 17.1 18.4 15.4 10.8 11.8 14.9 12.6
 14.1 13.  13.4 15.2 16.1 17.8 14.9 14.1 12.7 13.5 14.9 20.  16.4 17.7
 19.5 20.2 21.4 19.9 19.  19.1 19.1 20.1 19.9 19.6 23.2 29.8 13.8 13.3
 16.7 12.  14.6 21.4 23.  23.7 25.  21.8 20.6 21.2 19.1 20.6 15.2  7.
  8.1 13.6 20.1 21.8 24.5 23.1 19.7 18.3 21.2 17.5 16.8 22.4 20.6 23.9
 22.  11.9]
```

1.2 Creating a data frame.

[8]:
```
# Import a module to draw a three-dimensional graph.
from mpl_toolkits.mplot3d.axes3d import Axes3D
```

[9]:
```
#  Converts an explanatory variable to a DataFrame.
df = DataFrame(data=boston.data, columns = boston.feature_names)
```

[10]:
```
# Adds the target variable to the DataFrame.
df['PRICE'] = np.array(boston.target)
```

[11]:
```
# Display the first five lines of the data frame.
df.head(5)
```

[11]:

	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	PRICE
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

1.3 linear regression analysis

[12]:
```
# Display data (number of rooms) by specifying a column.
df[['RM']].head()
```

[12]:

	RM
0	6.575
1	6.421
2	7.185
3	6.998
4	7.147

[13]:
```
# explanatory variables
data = df.loc[:, ['RM']].values
data[0:5]
```

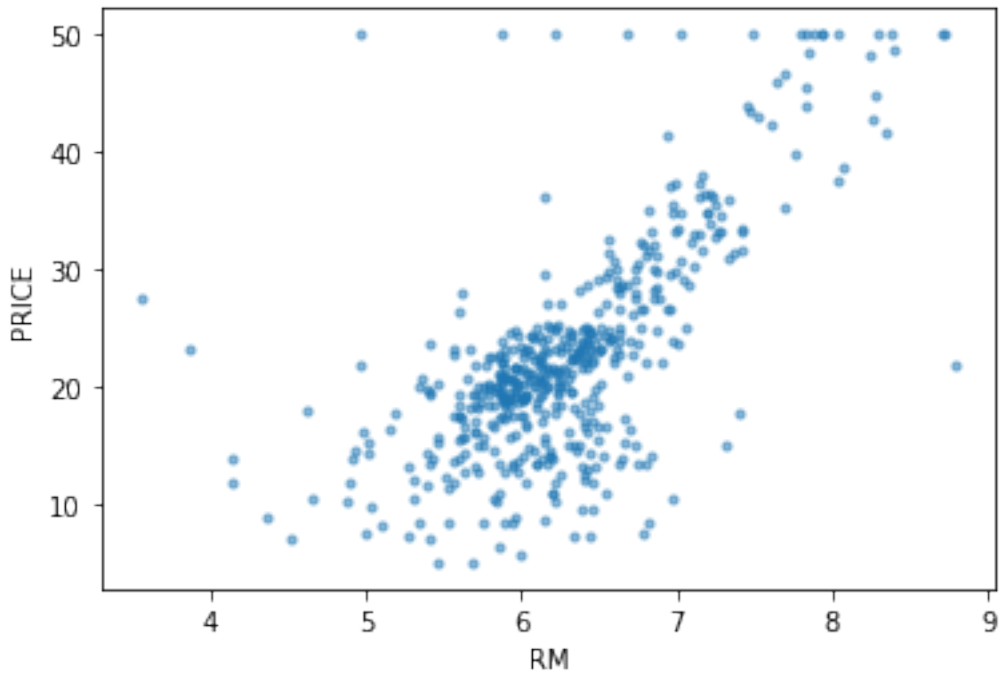
[13]:
```
array([[6.575],
       [6.421],
       [7.185],
       [6.998],
       [7.147]])
```

[14]:
```
# target variable
target = df.loc[:, 'PRICE'].values
target[:5]
```

[14]:
```
array([24. , 21.6, 34.7, 33.4, 36.2])
```

[15]:
```
# Check the data on the scatterplot.
plt.xlabel("RM")
plt.ylabel("PRICE")
plt.scatter(data, target, s=10, alpha=0.5, linewidths="1")
```

[15]:
```
<matplotlib.collections.PathCollection at 0x7fb8c91c7fd0>
```



[16]:
```
## Importing LinearRegression from a sklearn module
from sklearn.linear_model import LinearRegression
```

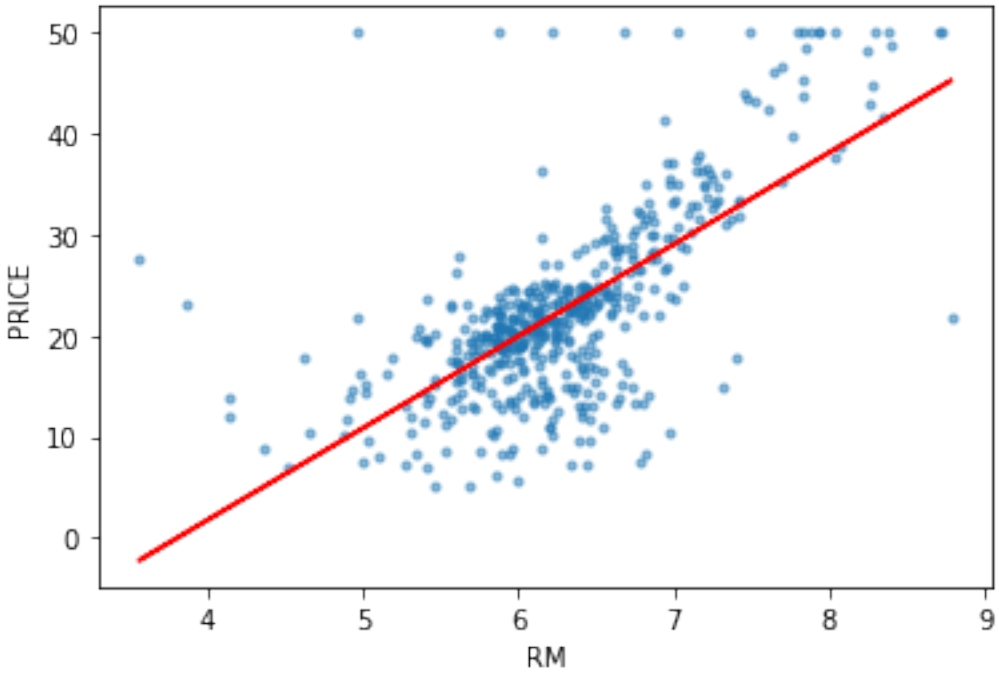
[17]:
```
# Create a LinearRegression object.
model = LinearRegression()
```

[18]:
```
# Estimate the parameters with the fit function.
model.fit(data, target)
```

[18]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

[19]:
```
# Draws a line in the model.
plt.xlabel("RM")
plt.ylabel("PRICE")
plt.scatter(data, target, s=10, alpha=0.5, linewidths="1")
plt.plot(data, model.predict(data), color = 'red')
```

[19]:
```
[<matplotlib.lines.Line2D at 0x7fb8c8d1e390>]
```



[20]:
```
# Projections (price for 6 rooms)
model.predict([[6]])
```

[20]:
```
array([19.94203311])
```

1.4 Multiple regression analysis (2 variables)

[21]:
```
#  Check the data by specifying columns.
df[['CRIM', 'RM']].head()
```

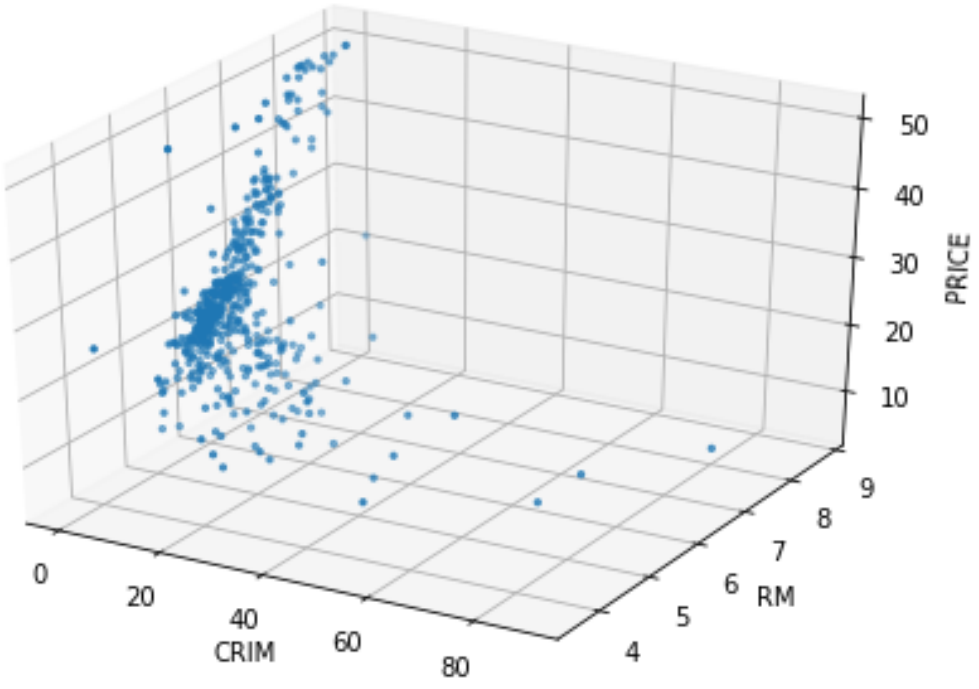
[21]:

	CRIM	RM
0	0.00632	6.575
1	0.02731	6.421
2	0.02729	7.185
3	0.03237	6.998
4	0.06905	7.147

[22]:
```
# explanatory variables
data2 = df.loc[:, ['CRIM', 'RM']].values
# target variable
target2 = df.loc[:, 'PRICE'].values
```

[23]:
```
# Plot the data.
x = df['CRIM']
y = df['RM']
z = df['PRICE']

# Generate figure
fig = plt.figure()

# Set ax to figure
ax = Axes3D(fig)

ax.set_xlabel("CRIM")
ax.set_ylabel("RM")
ax.set_zlabel("PRICE")

ax.scatter(x, y, z, s=5, marker="o")
```

[23]:
```
<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x7fb8c8d58490>
```



[24]:
```
# Create a LinearRegression object.
model2 = LinearRegression()
```

[25]:
```
# Estimate the parameters with the fit function.
model2.fit(data2, target2)
```

[25]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

[26]:
```
# Projections (4 rooms and a crime rate of 0.3)
model2.predict([[0.3, 4]])
```

[26]:
```
array([4.24007956])
```