

Linear Regression (Boston House Pricing)

March 22, 2022

```
[9]: from sklearn import linear_model
from sklearn import datasets
import pandas as pd
data = datasets.load_boston() ## loads Boston dataset from datasets librar
print(data['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 $1000(B_k - 0.63)^2$ where B_k is the proportion of black people
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.

/home/niteesh/.local/lib/python3.10/site-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function load_boston is deprecated; `load_boston` is deprecated in 1.0 and will be removed in 1.2.

The Boston housing prices dataset has an ethical problem. You can refer to the documentation of this function for further details.

The scikit-learn maintainers therefore strongly discourage the use of this dataset unless the purpose of the code is to study and educate about ethical issues in data science and machine learning.

In this special case, you can fetch the dataset from the original source::

```
import pandas as pd
import numpy as np
```

```

data_url = "http://lib.stat.cmu.edu/datasets/boston"
raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)
data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
target = raw_df.values[1::2, 2]

```

Alternative datasets include the California housing dataset (i.e. `:func:`~sklearn.datasets.fetch_california_housing``) and the Ames housing dataset. You can load the datasets as follows::

```

from sklearn.datasets import fetch_california_housing
housing = fetch_california_housing()

```

for the California housing dataset and::

```

from sklearn.datasets import fetch_openml
housing = fetch_openml(name="house_prices", as_frame=True)

```

for the Ames housing dataset.

```
warnings.warn(msg, category=FutureWarning)
```

```

[10]: # define the data/predictors as the pre-set feature names
df = pd.DataFrame(data.data, columns=data.feature_names)

# Put the target (housing value -- MEDV) in another DataFrame
target = pd.DataFrame(data.target, columns=["MEDV"])

```

```

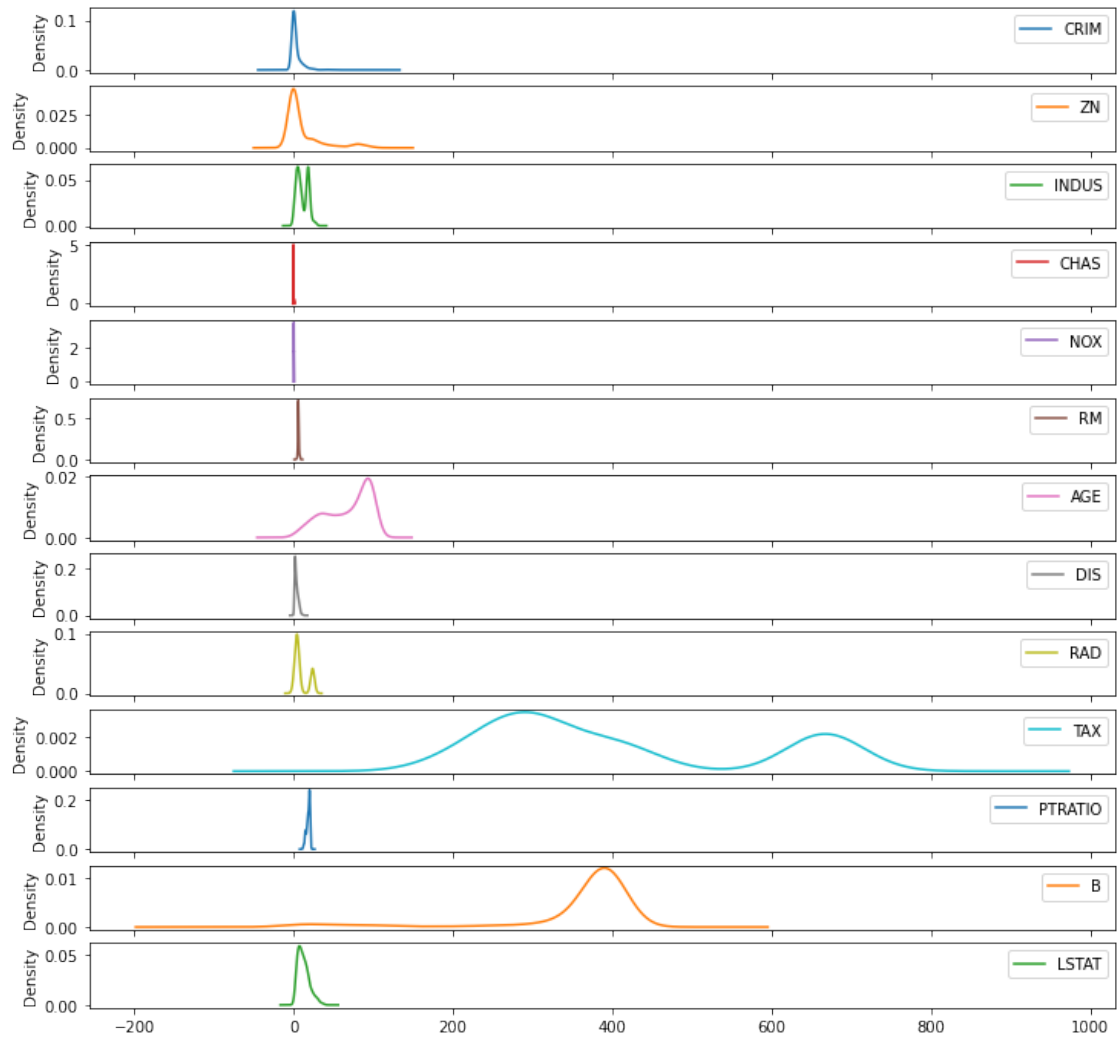
[11]: df.plot(kind='kde', subplots=True, figsize=(12,12))

```

```

[11]: array([<AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>,
<AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>,
<AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>,
<AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>,
<AxesSubplot:ylabel='Density'>], dtype=object)

```



```
[12]: df.corr()
```

```
[12]:
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	\
CRIM	1.000000	-0.200469	0.406583	-0.055892	0.420972	-0.219247	0.352734	
ZN	-0.200469	1.000000	-0.533828	-0.042697	-0.516604	0.311991	-0.569537	
INDUS	0.406583	-0.533828	1.000000	0.062938	0.763651	-0.391676	0.644779	
CHAS	-0.055892	-0.042697	0.062938	1.000000	0.091203	0.091251	0.086518	
NOX	0.420972	-0.516604	0.763651	0.091203	1.000000	-0.302188	0.731470	
RM	-0.219247	0.311991	-0.391676	0.091251	-0.302188	1.000000	-0.240265	
AGE	0.352734	-0.569537	0.644779	0.086518	0.731470	-0.240265	1.000000	
DIS	-0.379670	0.664408	-0.708027	-0.099176	-0.769230	0.205246	-0.747881	
RAD	0.625505	-0.311948	0.595129	-0.007368	0.611441	-0.209847	0.456022	
TAX	0.582764	-0.314563	0.720760	-0.035587	0.668023	-0.292048	0.506456	
PTRATIO	0.289946	-0.391679	0.383248	-0.121515	0.188933	-0.355501	0.261515	
B	-0.385064	0.175520	-0.356977	0.048788	-0.380051	0.128069	-0.273534	
LSTAT								

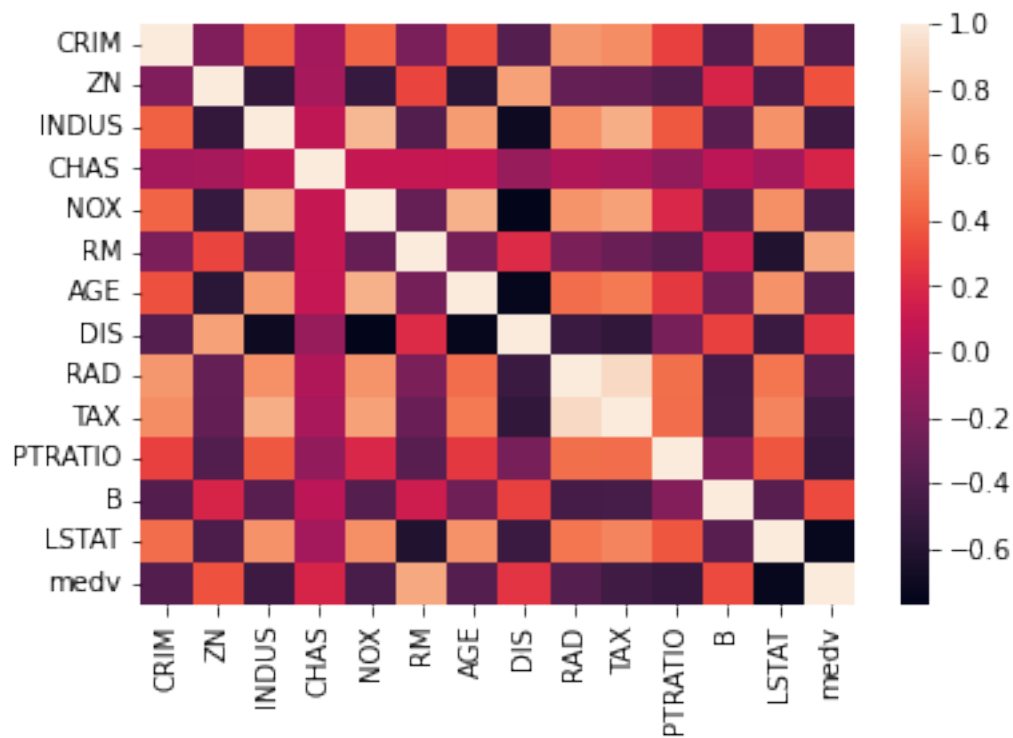
LSTAT	0.455621	-0.412995	0.603800	-0.053929	0.590879	-0.613808	0.602339
	DIS	RAD	TAX	PTRATIO	B	LSTAT	
CRIM	-0.379670	0.625505	0.582764	0.289946	-0.385064	0.455621	
ZN	0.664408	-0.311948	-0.314563	-0.391679	0.175520	-0.412995	
INDUS	-0.708027	0.595129	0.720760	0.383248	-0.356977	0.603800	
CHAS	-0.099176	-0.007368	-0.035587	-0.121515	0.048788	-0.053929	
NOX	-0.769230	0.611441	0.668023	0.188933	-0.380051	0.590879	
RM	0.205246	-0.209847	-0.292048	-0.355501	0.128069	-0.613808	
AGE	-0.747881	0.456022	0.506456	0.261515	-0.273534	0.602339	
DIS	1.000000	-0.494588	-0.534432	-0.232471	0.291512	-0.496996	
RAD	-0.494588	1.000000	0.910228	0.464741	-0.444413	0.488676	
TAX	-0.534432	0.910228	1.000000	0.460853	-0.441808	0.543993	
PTRATIO	-0.232471	0.464741	0.460853	1.000000	-0.177383	0.374044	
B	0.291512	-0.444413	-0.441808	-0.177383	1.000000	-0.366087	
LSTAT	-0.496996	0.488676	0.543993	0.374044	-0.366087	1.000000	

```
[13]: df['medv'] = target
```

```
[14]: import seaborn as sns
```

```
sns.heatmap(df.corr())
```

```
[14]: <AxesSubplot:>
```



```
[ ]:
```

```
[15]: target
```

```
[15]:      MEDV
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
```

```
[506 rows x 1 columns]
```

```
[33]:
```

```
[33]: pandas.core.frame.DataFrame
```

```
[34]: d = pd.DataFrame(df,target['MEDV'])
      d.to_csv("Boston House Prices dataset.csv", index = False)
```

```
[35]: X = df
      y = target["MEDV"]
```

```
[36]: lm = linear_model.LinearRegression()
      model = lm.fit(X,y)
```

```
[37]: predictions = lm.predict(X)
      predictions
```

```
[37]: array([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])

```

```
[38]: lm.score(X,y)
```

```
[38]: 1.0
```

```
[39]: lm.coef_
```

```
[39]: array([-8.54790267e-17, -2.49800181e-16, -5.85252333e-16, -1.44037004e-14,
-5.88240665e-16, 1.13667078e-15, -1.84314369e-16, -4.45715513e-16,
-2.58907479e-16, -1.04083409e-17, 7.64633582e-17, -2.98372438e-16,
-4.51461785e-16, 1.00000000e+00])
```

[]:

[]: