# boston-dataset-example

July 10, 2020

# 1 Boston Housing Dataset Example

Most of this code is included in Chapter 8 of Data Science for Mathematicians. We convert it to notebook form here so that you can see the output and explore it interactively online yourself.

## 1.1 Step 1: Obtain data

The Boston housing dataset is built into scikit-learn, so we can import it easily, as follows.

[1]:
```
from sklearn.datasets import load_boston
boston = load_boston()
```

But the `boston` object created this way is a conglomeration of several sub-objects and not ready to be printed in a human-readable way, so we organize it as follows.

## 1.2 Step 2: Create a feature-target dataset

We extract the features and target into separate variables and inspect their contents.

[2]:
```
import pandas as pd
features = pd.DataFrame(
    data=boston.data,
    columns=boston.feature_names)
target = boston.target
```

The features are a pandas DataFrame, the first few rows of which are shown here.

[3]:
```
features.head()
```

[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  1.0  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
```

How many rows are there, actually?

[4]:
```python
len( features )
```

[4]:
```
506
```

The target is a NumPy array that can be viewed as another column in the same dataset, as shown here.

[5]:
```python
target
```

[5]:
```
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])
```

### 1.3 Step 3: Split into training and test datasets

We will use 80% of the data for training and then test our model on the 20% held out for that purpose. Scikit-learn contains a function that will randomly split the dataset for us into training and test sets. We add the `random_state` parameter to specify a random number seed, thus guaranteeing reproducibility of the same results if you re-run this notebook later.

[6]:
```
from sklearn.model_selection import train_test_split
(X_training, X_test, y_training, y_test) = \
    train_test_split(features,target,train_size=0.8,random_state=1)
```

Let's verify that the split produced objects of the appropriate sizes.

[7]:
```
len( X_training ), len( X_test ), len( y_training ), len( y_test )
```

[7]:
```
(404, 102, 404, 102)
```

Since 404 is almost exactly 80% of the original 506, it looks like this has worked correctly.

### 1.4 Step 4: Create a model from the training dataset

We use scikit-learn's Pipeline object to compose two steps in sequence: First, select the five best features to use for prediction, and second, use those five features to fit a linear model to the training data.

[8]:
```
### Step 4: Create a model from the training dataset
from sklearn.pipeline import Pipeline
from sklearn.feature_selection import SelectKBest
from sklearn.linear_model import LinearRegression
estimator = Pipeline([
    ('select', SelectKBest(k=5)),
    ('model',  LinearRegression())
])
fit_model = estimator.fit(X=X_training , y=y_training)
```

Let's say we'd like to see which features were selected. We can ask the first step in the pipeline to show us its results.

[9]:
```
features.columns[ fit_model[0].get_support() ]
```

[9]:
```
Index(['CRIM', 'NOX', 'RM', 'AGE', 'LSTAT'], dtype='object')
```

And if we want to see the coefficients the model assigned to each of those variables, we can ask teh second step in the pipeline for its results.

[10]:
```
fit_model[1].intercept_, fit_model[1].coef_
```

[10]:
```
(2.9524367266279405,
 array([-0.09549911, -4.08891308,  4.56355544,  0.02161194, -0.61759647]))
```

The resulting model is therefore approximately the following.

$$2.952 - 0.095\text{CRIM} - 4.089\text{NOX} + 4.564\text{RM} + 0.022\text{AGE} - 0.618\text{LSTAT}$$

### 1.5 Step 5: Score the model using the test set

We compute the root mean squared error of the model on the test set.

[11]:
```
predictions = fit_model.predict(X=X_test)
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test, predictions)**0.5
```

[11]:
```
5.630885425217404
```