```
import numpy as np
import pandas as pd
import sklearn
from sklearn.datasets import load_boston
df = load_boston()
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning
 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)
df.keys() #return all the keys of the dataset dictionary
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename', 'data_module'])
boston = pd.DataFrame(df.data, columns=df.feature_names)
boston.head()
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 65.2 4.0900 1.0 296.0 15.3 396.90
1 0.02731 0.0 7.07 0.0 0.469 6.421 78.9 4.9671 2.0 242.0 17.8 396.90
2 0.02729 0.0 7.07 0.0 0.469 7.185 61.1 4.9671 2.0 242.0 17.8 392.83
3 0.03237 0.0 2.18 0.0 0.458 6.998 45.8 6.0622 3.0 222.0 18.7 394.63
boston['MEDV'] = df.target
boston.head()
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 65.2 4.0900 1.0 296.0 15.3 396.90
1 0.02731 0.0 7.07 0.0 0.469 6.421 78.9 4.9671 2.0 242.0 17.8 396.90
2 0.02729 0.0 7.07 0.0 0.469 7.185 61.1 4.9671 2.0 242.0 17.8 392.83
3 0.03237 0.0 2.18 0.0 0.458 6.998 45.8 6.0622 3.0 222.0 18.7 394.63
4 0.06905 0.0 2.18 0.0 0.458 7.147 54.2 6.0622 3.0 222.0 18.7 396.90
boston.isnull()
CRIM ZN INDUS CHAS NOX RM AGE DIS RAD TAX PTRATIO B
0 False False False False False False False False False False False False
1 False False False False False False False False False False False False
2 False False False False False False False False False False False False
3 False False False False False False False False False False False False
4 False False False False False False False False False False False False
... ... ... ... ... ... ... ... ... ... ... ... ...
501 False False False False False False False False False False False False
502 False False False False False False False False False False False False
503 False False False False False False False False False False False False
```

```
504 False False False False False False False False False False False False False
505 False False False False False False False False False False False False False
boston.isnull().sum()
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
from sklearn.model_selection import train_test_split
X = boston.drop('MEDV', axis=1)
Y = boston['MEDV']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.15, random_state=5

print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
(430, 13)
(76, 13)
(430,)
(76,)
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
lin_model = LinearRegression()
lin_model.fit(X_train, Y_train)
LinearRegression()
y_train_predict = lin_model.predict(X_train)
rmse = (np.sqrt(mean_squared_error(Y_train,y_train_predict)))
print("The model performance of training set")
print('RMSE is {}'.format(rmse))
print("\n")
y_test_predict = lin_model.predict(X_test)
rmse = (np.sqrt(mean_squared_error(Y_test,y_test_predict)))
print("The model performance for testing set")
print('RMSE is {}'.format(rmse))
The model performance of training set
RMSE is 4.710901797319796
The model performance for testing set
RMSE is 4.687543527902972
```

```
File "<ipython-input-1-acf7ba32ca90>", line 6
  usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87:
```