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

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-6-e1158bbdb4cf> in <module>()
      1 import numpy as np
      2 import pandas as pd
----> 3 import sklearns

ModuleNotFoundError: No module named 'sklearns'

---------------------------------------------------------------------------
NOTE: If your import is failing due to a missing package, you can
manually install dependencies using either !pip or !apt.

To view examples of installing some common dependencies, click the
"Open Examples" button below.
---------------------------------------------------------------------------
```

SEARCH STACK OVERFLOW

```
from sklearn.datasets import load_boston
df=load_boston()
```

```
df.keys()
```

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

```
print(df.DESCR)
```

```
.. _boston_dataset:

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

**Data Set Characteristics:**
```

```
        :Number of Instances: 506

        :Number of Attributes: 13 numeric/categorical predictive. Median Value (att

        :Attribute Information (in order):
            - CRIM     per capita crime rate by town
            - ZN       proportion of residential land zoned for lots over 25,000 sq
            - INDUS    proportion of non-retail business acres per town
            - CHAS     Charles River dummy variable (= 1 if tract bounds river; 0 o
            - 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 to
            - 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

    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
    problems.

    .. topic:: References

       - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influential Da
       - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learning. In P


boston=pd.DataFrame(df.data, columns=df.feature_names)
boston.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
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

```
boston['VALUE']=df.target
boston.head()


boston.isnull()


boston.isnull().sum()


from sklearn.model_selection import train_test_split

X = boston.drop('VALUE',axis=1)
Y = boston['VALUE']

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)


from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error


lin_model=LinearRegression()

lin_model.fit(X_train, Y_train)


y_train_predict=linmodel.predict(X_train)
rmse=(np.sqrt(mean_squared_error(Y_train, y_train_predict)))

print("the model performance for training set")
print('RMSE is {}'.format(rmse))
print("\n")

y_test_predict = lin_model.prdict(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))
```