```
import sklearn
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

```
from sklearn.datasets import load_boston
```

```
house_price = load_boston()
df = pd.DataFrame(house_price.data,
                  columns = house_price.feature_names)
```

```
#checking if any data is empty
check=df.isnull()
check.sum()
#output shows no empty values
```

```
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
dtype: int64
```

```
df['PRICE'] = house_price.target
df.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	

```
print(house_price.DESCR)
```

```
.. _boston_dataset:

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

**Data Set Characteristics:**

    :Number of Instances: 506

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

    :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 otherwi
        - 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 Mellc

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 addres
problems.

.. topic:: References

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

```
X= df.drop(axis=1,columns='PRICE')
Y= df['PRICE']
X.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	

Task 1: splitting the data

```
import sklearn.model_selection
X_train, X_test, Y_train, Y_test = sklearn.model_selection.train_test_split(X, Y, test_siz
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(354, 13)
(152, 13)
(354,)
(152,)
```

```
from sklearn.linear_model import LinearRegression

OLS_model = LinearRegression(fit_intercept=True,normalize=False,copy_X=True)
OLS_model.fit(X_train, Y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
b_coef= OLS_model.coef_
print(b_coef[12])
```

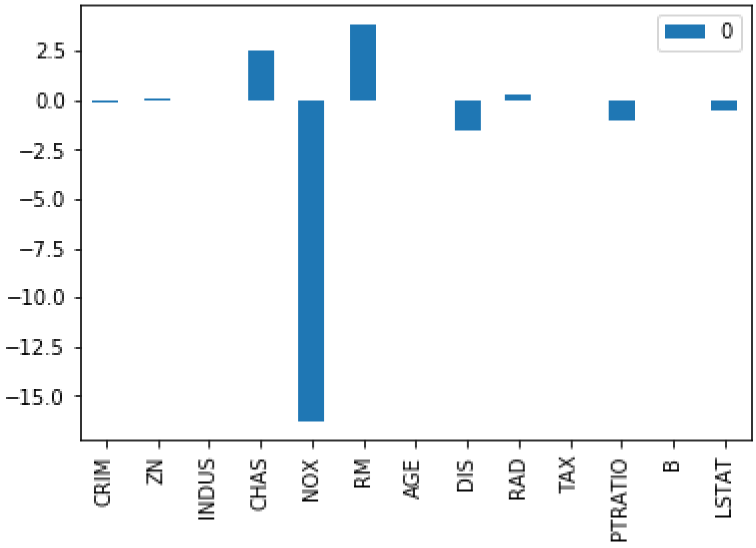
```
-0.4867380656449212
```

```
B= pd.DataFrame(data=b_coef,index=X.columns)
X.columns
```

```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT'],
      dtype='object')
```

```
B.plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe9ede588d0>
```



Task2

```
print(X.shape)
```

```
(506, 13)
```

```
df3 = pd.DataFrame(columns=['room',
                   'residential zone',
                   'highway access'
                    ,'crime rate'
                    ,'tax'],index= range(200))
df3.head()
```

	room	residential zone	highway access	crime rate	tax
0	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN

```
X_train, X_test, Y_train, Y_test = sklearn.model_selection.train_test_split(X, Y, test_siz
```

```
from sklearn.linear_model import Ridge
reg_coef_L = pd.DataFrame(columns=['crime rate',
                                   'room',
                   'residential zone',
                   'highway access'
                    ,'tax'])
for lamda in range(200):
  Ridge_model =Ridge(fit_intercept=True,normalize=True,copy_X=True,alpha=lamda)
```

```
Ridge_model.fit(X_train, Y_train)
df2 = pd.DataFrame([[Ridge_model.coef_[0],Ridge_model.coef_[5],Ridge_model.coef_[1],Ridg
                                'room',
                'residential zone',
                'highway access'
                 ,'tax'])
reg_coef_L=reg_coef_L.append(df2,ignore_index=True)
```

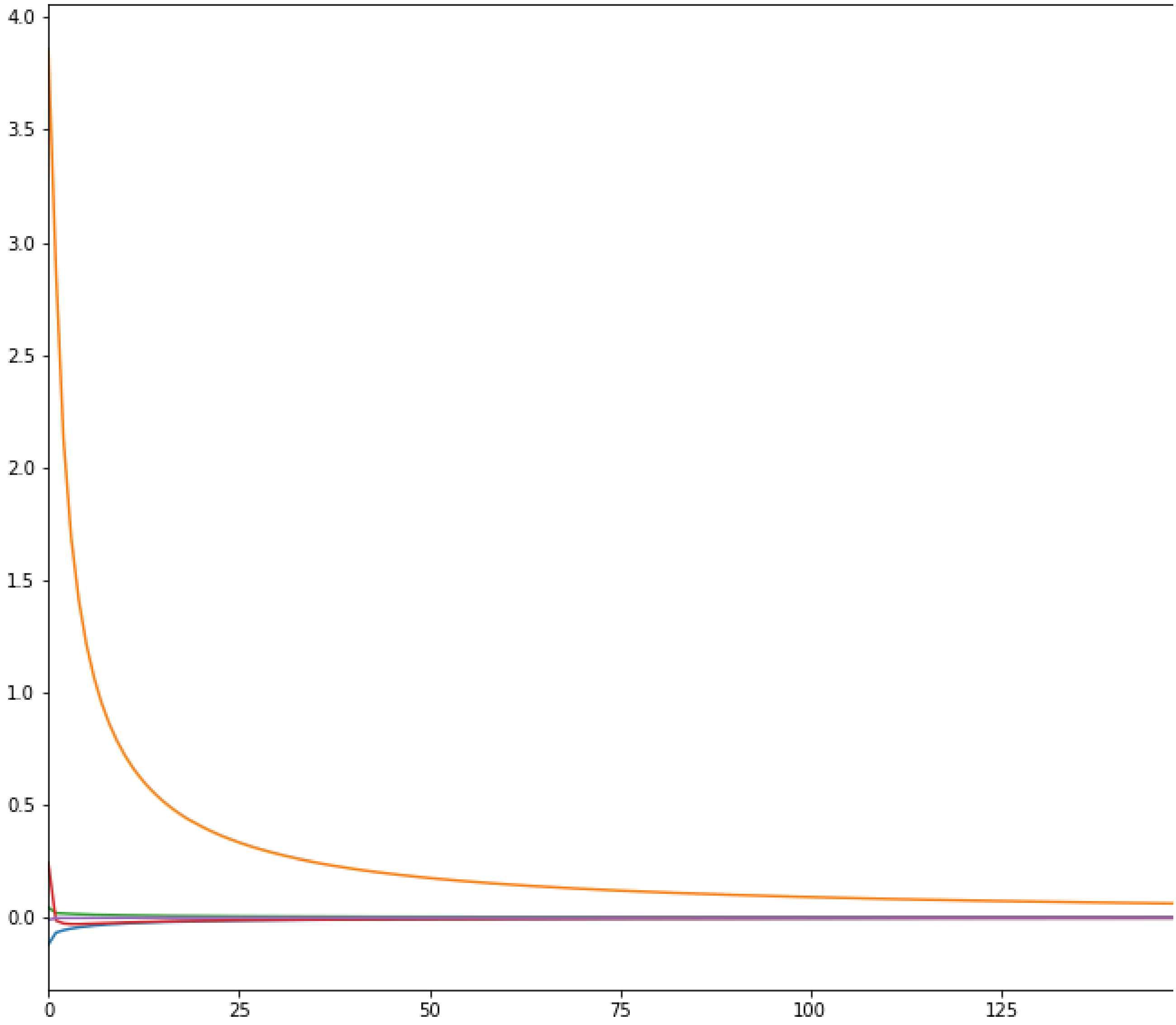
```
reg_coef_L
```

	crime rate	room	residential zone	highway access	tax
0	-0.121310	3.859068	0.044466	0.242143	-0.011072
1	-0.068001	2.880140	0.019307	-0.013911	-0.002939
2	-0.056939	2.122684	0.017274	-0.027019	-0.002759
3	-0.049720	1.691319	0.015636	-0.029878	-0.002554
4	-0.044315	1.411524	0.014244	-0.029921	-0.002356
...	...	...	...	...	...
195	-0.002133	0.046972	0.000748	-0.002046	-0.000129
196	-0.002123	0.046737	0.000744	-0.002036	-0.000128
197	-0.002112	0.046503	0.000740	-0.002026	-0.000128
198	-0.002102	0.046272	0.000737	-0.002016	-0.000127
199	-0.002092	0.046043	0.000733	-0.002007	-0.000127

200 rows × 5 columns

```
reg_coef_L.plot.line(figsize=(15,10))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe9eded99e8>
```



Task 3

```
df4 = pd.DataFrame(columns=['room',
                   'residential zone',
                   'highway access'
                    ,'crime rate'
                    ,'tax'],index= range(200))
df4.head()
```

room residential zone highway access crime rate tax

```
X_train, X_test, Y_train, Y_test = sklearn.model_selection.train_test_split(X, Y, test_siz

from sklearn.linear_model import Lasso
reg_coef_Lasso = pd.DataFrame(columns=['crime rate',
                                   'room',
                   'residential zone',
                   'highway access'
                    ,'tax'])
Lasso_model =Lasso(fit_intercept=True,normalize=False,copy_X=True,alpha=4)
Lasso_model.fit(X_train, Y_train)
print(Lasso_model.coef_)
```

```
[-0.          0.03965108 -0.          0.          0.          0.
  0.03706659 -0.          0.         -0.00849255 -0.21079256  0.00464294
 -0.76847208]
```

```
for lamda in range(2,200):
  Lasso_model =Lasso(fit_intercept=True,normalize=False,copy_X=True,alpha=lamda)
  Lasso_model.fit(X_train, Y_train)
  print(Lasso_model.coef_)
  df5 = pd.DataFrame([[Lasso_model.coef_[0],Lasso_model.coef_[5],Lasso_model.coef_[1],Lass
                                   'room',
                   'residential zone',
                   'highway access'
                    ,'tax'])

  reg_coef_Lasso=reg_coef_Lasso.append(df5,ignore_index=True)
```

```
 [-0.          0.         -0.          0.         -0.          0.
  -0.          0.         -0.         -0.02005173 -0.          0.
  -0.        ]
 [-0.          0.         -0.          0.         -0.          0.
  -0.          0.         -0.         -0.02001727 -0.          0.
  -0.        ]
 [-0.          0.         -0.          0.         -0.          0.
  -0.          0.         -0.         -0.01998281 -0.          0.
  -0.        ]
 [-0.          0.         -0.          0.         -0.          0.
  -0.          0.         -0.         -0.01994835 -0.          0.
  -0.        ]
 [-0.          0.         -0.          0.         -0.          0.
  -0.          0.         -0.         -0.01991389 -0.          0.
  -0.        ]
 [-0.          0.         -0.          0.         -0.          0.
  -0.          0.         -0.         -0.01987943 -0.          0.
  -0.        ]
 [-0.          0.         -0.          0.         -0.          0.
  -0.          0.         -0.         -0.01984497 -0.          0.
  -0.        ]
 [-0.          0.         -0.          0.         -0.          0.
  -0.          0.         -0.         -0.01981051 -0.          0.
  -0.        ]
 [-0.          0.         -0.          0.         -0.          0.
  -0.          0.         -0.         -0.01977605 -0.          0.
  -0.        ]
 [-0.          0.         -0.          0.         -0.          0.
  -0.          0.         -0.         -0.01974159 -0.          0.
  -0.        ]
 [-0.          0.         -0.          0.         -0.          0.
  -0.          0.         -0.         -0.01970713 -0.          0.
  -0.        ]
 [-0.          0.         -0.          0.         -0.          0.
  -0.          0.         -0.         -0.01967267 -0.          0.
  -0.        ]
 [-0.          0.         -0.          0.         -0.          0.
  -0.          0.         -0.         -0.01963822 -0.          0.
  -0.        ]
 [-0.          0.         -0.          0.         -0.          0.
  -0.          0.         -0.         -0.01960376 -0.          0.
  -0.        ]
 [-0.         0.        -0.         0.        -0.         0.
  -0.         0.        -0.        -0.0195693 -0.         0.
  -0.        ]
 [-0.          0.         -0.          0.         -0.          0.
  -0.          0.         -0.         -0.01953484 -0.          0.
  -0.        ]
 [-0.          0.         -0.          0.         -0.          0.
  -0.          0.         -0.         -0.01950038 -0.          0.
  -0.        ]
 [-0.          0.         -0.          0.         -0.          0.
  -0.          0.         -0.         -0.01946592 -0.          0.
  -0.        ]
 [-0.          0.         -0.          0.         -0.          0.
  -0.          0.         -0.         -0.01943146 -0.          0.
  -0.        ]
 [-0.        0.       -0.        0.       -0.        0.       -0.
   0.       -0.       -0.019397 -0.        0.       -0.      ]
 [-0.          0.         -0.          0.         -0.          0.
  -0.          0.         -0.         -0.01936254 -0.          0.
```

```
 -0.        ]
[-0.         0.         -0.          0.         -0.          0.
 -0.         0.         -0.         -0.01932808 -0.          0.
 -0.        ]
[-0.         0.         -0.          0.         -0.          0.
 -0.         0.         -0.         -0.01929362 -0.          0.
 -0.        ]
[-0.         0.         -0.          0.         -0.          0.
 -0.         0.         -0.         -0.01925916 -0.          0.
 -0.        ]
[-0.        0.        -0.         0.        -0.         0.
 -0.        0.        -0.        -0.0192247 -0.         0.
 -0.       ]
[-0.         0.         -0.          0.         -0.          0.
 -0.         0.         -0.         -0.01919024 -0.          0.
 -0.        ]
[-0.         0.         -0.          0.         -0.          0.
 -0.         0.         -0.         -0.01915578 -0.          0.
 -0.        ]
```

```
reg_coef_Lasso
```

	crime rate	room	residential zone	highway access	tax
0	-0.018627	0.0	0.034923	0.109626	-0.010408
1	-0.000000	0.0	0.034382	0.008605	-0.007361
2	-0.000000	0.0	0.039651	0.000000	-0.008493
3	-0.000000	0.0	0.043481	0.000000	-0.009773
4	-0.000000	0.0	0.040648	0.000000	-0.009987
...	...	...	...	...	...
193	-0.000000	0.0	0.000000	-0.000000	-0.019294
194	-0.000000	0.0	0.000000	-0.000000	-0.019259
195	-0.000000	0.0	0.000000	-0.000000	-0.019225
196	-0.000000	0.0	0.000000	-0.000000	-0.019190
197	-0.000000	0.0	0.000000	-0.000000	-0.019156

198 rows × 5 columns

```
reg_coef_Lasso.plot.line(figsize=(15,10))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe9edcea4e0>
```