Q13 and Q15 are subjective answer type questions, Answer them briefly.

1. Explain the term regularization? Ans-Regularization is the technique through which we can handle problem of under fitting and over fitting, where the model should learn all the data in where all the proper wat in predication.
   There are three techniques of regularization- L1 ---Lasso Regression L2 ---Ridge Regression Elastic net

L1 ---Lasso Regression L1 is do This model will internally controls the coefficient, it will whether the particular column, data or variable is working perfectly for the output or not, if so internally controls all the coefficient values of the particular variable to be thinking of where the coefficient column of variable as good input to the output way or not. it thinks that this not working that the output it will make them coefficient value or it will take that variable totally removed from the database. This all activities do in internally who are not controlling any think

L2 ---Ridge Regression It will try to reduce the gab between the different coefficient, it is finally tunning the values of coefficient or variables to be up or down to be help of alpha parameters Try to control the different between the coefficient values Try to minimize the different between coefficient values Elastic net Elastic net is the combination of loss regression & ridge regression properties present here. Elastic net do try to delete the variable, also it will try to minimize the different between the coefficient values or different variables both of thinks control with the help of elastic net

In [ ]:

1. Which particular algorithms are used for regularization? Ans- there are three main regularization techniques, lasso regression, ridge regression and elastic net so use these all three regularizations for algorithm. EX I am writing algorithm in below

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
```

In [2]:
```
from sklearn.datasets import load_boston
```

In [9]:
```
boston = load_boston()
```

```
E:\Users\Ashwini\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Funct
```

```
ion 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)
```

In [10]:

```
boston
```

Out[10]:

```
{'data': array([[6.3200e-03, 1.8000e+01, 2.3100e+00, ..., 1.5300e+01, 3.9690e+02,
         4.9800e+00],
        [2.7310e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.9690e+02,
         9.1400e+00],
        [2.7290e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.9283e+02,
         4.0300e+00],
        ...,
        [6.0760e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9690e+02,
         5.6400e+00],
        [1.0959e-01, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9345e+02,
         6.4800e+00],
        [4.7410e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9690e+02,
         7.8800e+00]]),
 'target': 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]),
 'feature_names': array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RA
D',
        'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='<U7'),
 'DESCR': ".. _boston_dataset:\n\nBoston house prices dataset\n----------------------
--\n\n**Data Set Characteristics:**  \n\n    :Number of Instances: 506 \n\n    :Number o
f Attributes: 13 numeric/categorical predictive. Median Value (attribute 14) is usually
the target.\n\n    :Attribute Information (in order):\n        - CRIM     per capita cri
me rate by town\n        - ZN       proportion of residential land zoned for lots over 2
5,000 sq.ft.\n        - INDUS    proportion of non-retail business acres per town\n
- CHAS     Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)\n
- NOX      nitric oxides concentration (parts per 10 million)\n        - RM       averag
e number of rooms per dwelling\n        - AGE      proportion of owner-occupied units bu
ilt prior to 1940\n        - DIS      weighted distances to five Boston employment centr
es\n        - RAD      index of accessibility to radial highways\n        - TAX      ful
l-value property-tax rate per $10,000\n        - PTRATIO  pupil-teacher ratio by town\n
- B        1000(Bk - 0.63)^2 where Bk is the proportion of black people by town\n
- LSTAT    % lower status of the population\n        - MEDV     Median value of owner-oc
cupied homes in $1000's\n\n    :Missing Attribute Values: None\n\n    :Creator: Harriso
n, D. and Rubinfeld, D.L.\n\nThis is a copy of UCI ML housing dataset.\nhttps://archive.
ics.uci.edu/ml/machine-learning-databases/housing/\n\n\nThis dataset was taken from the
StatLib library which is maintained at Carnegie Mellon University.\n\nThe Boston house-p
rice data of Harrison, D. and Rubinfeld, D.L. 'Hedonic\nprices and the demand for clean
air', J. Environ. Economics & Management,\nvol.5, 81-102, 1978.   Used in Belsley, Kuh &
Welsch, 'Regression diagnostics\n...', Wiley, 1980.   N.B. Various transformations are u
sed in the table on\npages 244-261 of the latter.\n\nThe Boston house-price data has bee
n used in many machine learning papers that address regression\nproblems.   \n     \n..
topic:: References\n\n   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying I
nfluential Data and Sources of Collinearity', Wiley, 1980. 244-261.\n   - Quinlan,R. (19
93). Combining Instance-Based and Model-Based Learning. In Proceedings on the Tenth Inte
rnational Conference of Machine Learning, 236-243, University of Massachusetts, Amherst.
```

```
Morgan Kaufmann.\n",
 'filename': 'boston_house_prices.csv',
 'data_module': 'sklearn.datasets.data'}
```

In [11]:
```
boston.keys()
```

Out[11]:
```
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename', 'data_module'])
```

In [12]:
```
boston.data
```

Out[12]:
```
array([[6.3200e-03, 1.8000e+01, 2.3100e+00, ..., 1.5300e+01, 3.9690e+02,
        4.9800e+00],
       [2.7310e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.9690e+02,
        9.1400e+00],
       [2.7290e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.9283e+02,
        4.0300e+00],
       ...,
       [6.0760e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9690e+02,
        5.6400e+00],
       [1.0959e-01, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9345e+02,
        6.4800e+00],
       [4.7410e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9690e+02,
        7.8800e+00]])
```

In [13]:
```
boston.target
```

Out[13]:
```
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])
```

In [14]:
```
boston.feature_names
```

Out[14]:
```
array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD',
       'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='<U7')
```

In [15]:
```
boston.DESCR
```

Out[15]:
```
".. _boston_dataset:\n\nBoston house prices dataset\n---------------------------\n\n**Da
ta Set Characteristics:**  \n\n    :Number of Instances: 506 \n\n    :Number of Attribut
es: 13 numeric/categorical predictive. Median Value (attribute 14) is usually the targe
t.\n\n    :Attribute Information (in order):\n        - CRIM     per capita crime rate b
y town\n        - ZN       proportion of residential land zoned for lots over 25,000 sq.
ft.\n        - INDUS    proportion of non-retail business acres per town\n        - CHAS
Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)\n        - NOX
nitric oxides concentration (parts per 10 million)\n        - RM       average number of
rooms per dwelling\n        - AGE      proportion of owner-occupied units built prior to
1940\n        - DIS      weighted distances to five Boston employment centres\n        -
RAD      index of accessibility to radial highways\n        - TAX      full-value proper
ty-tax rate per $10,000\n        - PTRATIO  pupil-teacher ratio by town\n        - B
1000(Bk - 0.63)^2 where Bk is the proportion of black people by town\n        - LSTAT
% lower status of the population\n        - MEDV     Median value of owner-occupied home
s in $1000's\n\n    :Missing Attribute Values: None\n\n    :Creator: Harrison, D. and Ru
binfeld, D.L.\n\nThis is a copy of UCI ML housing dataset.\nhttps://archive.ics.uci.edu/
ml/machine-learning-databases/housing/\n\n\nThis dataset was taken from the StatLib libr
ary which is maintained at Carnegie Mellon University.\n\nThe Boston house-price data of
Harrison, D. and Rubinfeld, D.L. 'Hedonic\nprices and the demand for clean air', J. Envi
ron. Economics & Management,\nvol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Reg
ression diagnostics\n...', Wiley, 1980.   N.B. Various transformations are used in the t
able on\npages 244-261 of the latter.\n\nThe Boston house-price data has been used in ma
ny machine learning papers that address regression\nproblems.   \n     \n.. topic:: Refe
rences\n\n   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influential D
ata and Sources of Collinearity', Wiley, 1980. 244-261.\n   - Quinlan,R. (1993). Combini
ng Instance-Based and Model-Based Learning. In Proceedings on the Tenth International Co
nference of Machine Learning, 236-243, University of Massachusetts, Amherst. Morgan Kauf
mann.\n"
```

In [16]:
```
x = boston.data
```

In [17]:
```
y = boston.target
```

In [19]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.33,random_state=47)
```

In [20]:
```
x_train.shape
```

Out[20]:
```
(339, 13)
```

In [21]:
```
y_train.shape
```

Out[21]:
```
(339,)
```

In [22]:
```
x_test.shape
```

Out[22]:
```
(167, 13)
```

In [23]:
```
y_test.shape
```

Out[23]:
```
(167,)
```

In [24]:
```
lm = LinearRegression()
```

In [25]:
```
lm.fit(x_train,y_train)
```

Out[25]:
```
LinearRegression()
```

In [26]:
```
lm.score(x_train,y_train)
```

Out[26]:
```
0.7665529829880268
```

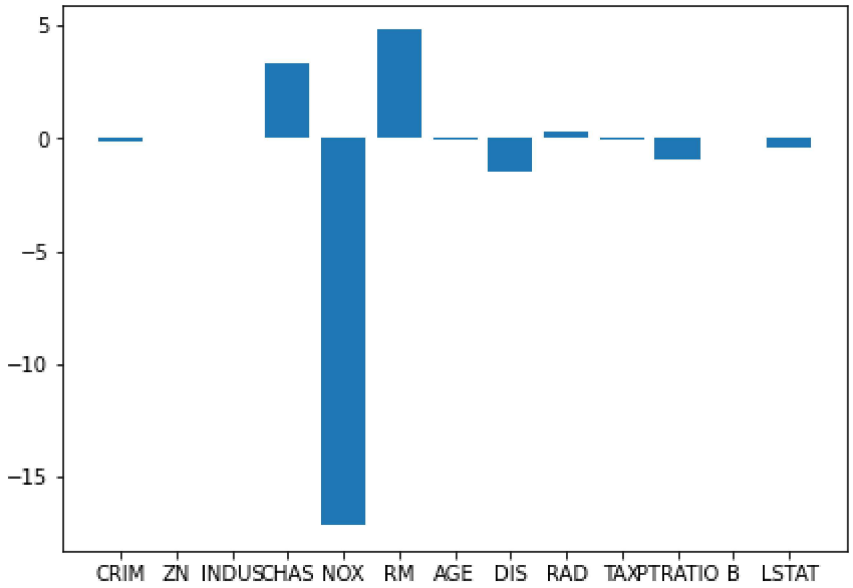
In [27]:
```
lm.coef_
```

Out[27]:
```
array([-1.10843155e-01,  3.66995321e-02,  8.74460015e-03,  3.32218225e+00,
       -1.71846638e+01,  4.81511727e+00, -1.38022222e-02, -1.44146494e+00,
        2.63392383e-01, -1.17496784e-02, -9.25130518e-01,  9.07087108e-03,
       -4.12146398e-01])
```

In [28]:
```
boston.feature_names
```

Out[28]:
```
array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD',
       'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='<U7')
```

In [29]:
```
plt.figure(figsize=(7,5))
plt.bar(boston.feature_names,lm.coef_)
plt.show()
```



In [ ]:

```
```

In [30]:

```
from sklearn.linear_model import Lasso,Ridge
```

In [31]:

```
#  x_train,x_test,y_train,y_test
```

In [32]:

```
#will reduce the coefficient to zero (those features are not informative)
#alpha values could be ------> .0001,.001,.01,.1,1,10------->higher values reduce all c
#Default value of alpha =1.0
# alpha  =.01
ls=Lasso(alpha=0.0001)
#ls=lasso(alpha=1.0)  #default
ls.fit(x_train,y_train)
ls.score(x_train,y_train)
```

Out[32]:

```
0.7665529400345414
```

In [33]:

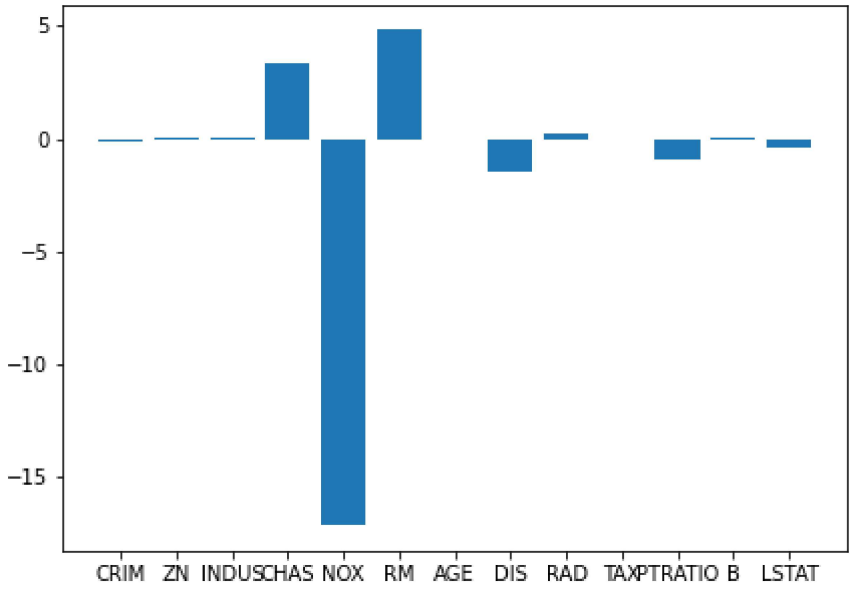
```
ls.coef_
```

Out[33]:

```
array([-1.10824693e-01,  3.67066154e-02,  8.58729659e-03,  3.32038457e+00,
       -1.71509902e+01,  4.81528448e+00, -1.38318082e-02, -1.44099502e+00,
        2.63304303e-01, -1.17526095e-02, -9.24709393e-01,  9.07325563e-03,
       -4.12162696e-01])
```

In [34]:

```
plt.figure(figsize=(7,5))
plt.bar(boston.feature_names,ls.coef_)
plt.show()
```



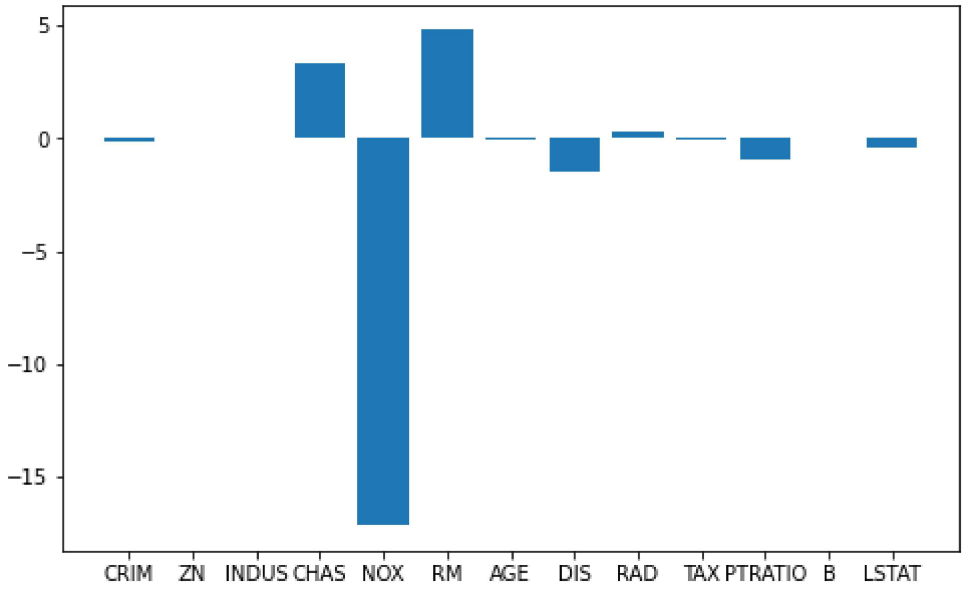
In [35]:
```
#try to minimize the coefficient variance

rd=Ridge(alpha=0.0001)
#rd=Ridge()
rd.fit(x_train,y_train)
rd.score(x_train,y_train)
```

Out[35]:
```
0.7665529828886577
```

In [36]:
```
rd.coef_
```

Out[36]:
```
array([-1.10842268e-01,  3.66998800e-02,  8.73686616e-03,  3.32215005e+00,
       -1.71830063e+01,  4.81513570e+00, -1.38038984e-02, -1.44144307e+00,
        2.63388003e-01, -1.17498140e-02, -9.25109364e-01,  9.07099128e-03,
       -4.12146425e-01])
```

In [37]:
```
plt.figure(figsize=(8,5))
plt.bar(boston.feature_names,rd.coef_)
plt.show()
```



In [38]:
```
#elasticNet is a combination of both Lasso and Ridge

from sklearn.linear_model import ElasticNet
enr=ElasticNet(alpha=0.0001)
#enr=ElasticNet()
enr.fit(x_train,y_train)
enrpred=enr.predict(x_test)
print(enr.score(x_train,y_train))
enr.coef_
```

```
0.7665498739655398
```

Out[38]:
```
array([-1.10686350e-01,  3.67611422e-02,  7.37727002e-03,  3.31588047e+00,
       -1.68916472e+01,  4.81825513e+00, -1.40961471e-02, -1.43758689e+00,
        2.62619134e-01, -1.17737697e-02, -9.21396641e-01,  9.09208744e-03,
       -4.12160129e-01])
```

In [ ]:
```

```

In [39]:
```
#regularization algorithm
from sklearn.svm import SVR


svr = SVR(kernel = "linear")
svr.fit(x_train,y_train)
svr.score(x_train,y_train)
pred_y=svr.predict(x_test)
pred_y


svr = SVR(kernel = "poly")
svr.fit(x_train,y_train)
svr.score(x_train,y_train)
pred_y=svr.predict(x_test)
pred_y

svr = SVR(kernel = "rbf")
svr.fit(x_train,y_train)
svr.score(x_train,y_train)
```

```
pred_y=svr.predict(x_test)
pred_y
```

Out[39]:
```
array([23.28744134, 24.56631358, 18.46788571, 24.71247389, 23.63365497,
       19.37654012, 15.42079077, 23.08463582, 19.91760361, 16.07283145,
       23.99245456, 24.20352244, 19.20781723, 15.93298859, 19.71862163,
       20.32705259, 23.38813836, 21.63917456, 15.21690356, 22.08296183,
       24.61322324, 22.41408217, 22.43797101, 15.68228779, 23.60405219,
       13.1356742 , 22.96661344, 21.28691871, 15.72338131, 23.80585188,
       21.7286332 , 23.53659785, 16.12229577, 15.53022173, 22.35804844,
       22.64810536, 13.04619512, 24.69576529, 22.59646615, 22.71348407,
       23.46490829, 22.29987701, 20.00163233, 22.43112248, 22.1864765 ,
       15.4996442 , 13.33854066, 22.76908036, 22.89178134, 24.17857538,
       22.87320299, 23.24431762, 14.96564362, 22.59120107, 15.50525114,
       22.97791384, 22.49893981, 15.60462996, 15.28462021, 21.73752155,
       13.09239213, 22.62645151, 14.99894616, 19.47772653, 25.02670173,
       20.59807296, 22.49473978, 21.00555132, 23.33452147, 19.76263747,
       22.66032871, 16.86749603, 20.88001221, 20.44467842, 21.7517663 ,
       15.17113117, 22.25716028, 23.59719194, 19.43193719, 14.73281927,
       14.52129835, 24.79847311, 23.22956696, 15.7477263 , 23.1717765 ,
       15.02485424, 23.52651507, 22.78012264, 15.83363992, 20.72355851,
       15.66151263, 19.65173122, 20.37771416, 22.4104981 , 22.92873072,
       15.6026823 , 24.25745714, 22.40668655, 15.60406066, 15.74375243,
       23.15853788, 23.37517576, 20.64318507, 13.12323859, 19.41568829,
       24.52131116, 21.12956108, 22.22027919, 20.04067278, 20.56263218,
       21.83094495, 13.41236346, 21.8215765 , 20.95319778, 13.33143978,
       14.92895223, 19.44186223, 22.68895254, 24.69775341, 22.94587147,
       15.57329803, 20.05769155, 23.6717599 , 23.01366232, 12.91709013,
       22.09215304, 23.8602335 , 15.38950823, 22.51119289, 19.44116095,
       20.77751926, 13.03960665, 22.18567484, 22.92945119, 17.42898608,
       24.16714061, 15.80907607, 20.69880401, 22.14948992, 15.64596766,
       15.48525816, 22.92498743, 15.57677123, 23.49944313, 15.65071167,
       15.35646683, 23.17267577, 15.61378222, 24.27983555, 22.96694058,
       23.2683893 , 23.29045227, 22.95302614, 15.58752741, 15.52939018,
       20.68401664, 23.97967482, 14.78639397, 23.07229527, 19.72900774,
       22.46358208, 16.12854607, 15.70785925, 20.03988635, 24.48279605,
       25.12568281, 24.29313999])
```

In [40]:
```
from sklearn.svm import SVR

kernellist=['linear','poly','rbf']
for i in kernellist:
    sv=SVR(kernel=i)
    sv.fit(x_train,y_train)
    print(sv.score(x_train,y_train))
```

```
0.7390428007452163
0.21984205036201276
0.22924996554545518
```

1. Explain the term error present in linear regression equation?

An error term is a residual variable produced by a statistical or mathematical model, which is created when the model does not fully represent the actual relationship between the independent variables and the dependent variables. As a result of this incomplete relationship, the error term is the amount at which the equation may differ during empirical analysis.

The error term is also known as the residual, disturbance, or remainder term, and is variously represented in models by the letters e, ε, or u.

An error term represents the margin of error within a statistical model; it refers to the sum of the deviations within the regression line, which provides an explanation for the difference between the theoretical value of the model and the actual observed results. The regression line is used as a point of analysis when attempting to determine the correlation between one independent variable and one dependent variable.

Error Term Use in a Formula: $Y=\alpha X+B\rho+\epsilon$

where: α,B=Constant parameters X,ρ=Independent variables ϵ=Error term

Linear regression is a form of analysis that relates to current trends experienced by a particular security or index by providing a relationship between a dependent and independent variables, such as the price of a security and the passage of time, resulting in a trend line that can be used as a predictive model.

for example: Within a linear regression model tracking a stock's price over time, the error term is the difference between the expected price at a particular time and the price that was actually observed

In [ ]: