In [82]:
```python
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

from sklearn.datasets import load_boston
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
```

In [44]:
```python
#loading the Dataset

boston=load_boston()
print(boston.DESCR)
```

```
Boston House Prices dataset
===========================

Notes
------
Data Set Characteristics:

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive

    :Median Value (attribute 14) is usually the target

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over 2
5,000 sq.ft.
        - INDUS    proportion of non-retail business acres per town
        - CHAS     Charles River dummy variable (= 1 if tract bounds ri
```

```
ver; 0 otherwise)
        - NOX      nitric oxides concentration (parts per 10 million)
        - RM       average number of rooms per dwelling
        - AGE      proportion of owner-occupied units built prior to 19
40
        - 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 blac
ks 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.
http://archive.ics.uci.edu/ml/datasets/Housing


This dataset was taken from the StatLib library which is maintained at
Carnegie Mellon University.

The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedoni
c
prices and the demand for clean air', J. Environ. Economics & Managemen
t,
vol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regression diagn
ostics
...', 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 pape
rs that address regression
problems.
```

```
**References**

   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influe
ntial Data and Sources of Collinearity', Wiley, 1980. 244-261.
   - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learni
ng. In Proceedings on the Tenth International Conference of Machine Lea
rning, 236-243, University of Massachusetts, Amherst. Morgan Kaufmann.
   - many more! (see http://archive.ics.uci.edu/ml/datasets/Housing)
```

In [45]:

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

features
```

Out[45]:

| | CRIM | ZN | INDUS | CHAS | NOX | RM | AGE | DIS | RAD | TAX | PTRATIO | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.00632 | 18.0 | 2.31 | 0.0 | 0.538 | 6.575 | 65.2 | 4.0900 | 1.0 | 296.0 | 15.3 | 396 |
| **1** | 0.02731 | 0.0 | 7.07 | 0.0 | 0.469 | 6.421 | 78.9 | 4.9671 | 2.0 | 242.0 | 17.8 | 396 |
| **2** | 0.02729 | 0.0 | 7.07 | 0.0 | 0.469 | 7.185 | 61.1 | 4.9671 | 2.0 | 242.0 | 17.8 | 392 |
| **3** | 0.03237 | 0.0 | 2.18 | 0.0 | 0.458 | 6.998 | 45.8 | 6.0622 | 3.0 | 222.0 | 18.7 | 394 |
| **4** | 0.06905 | 0.0 | 2.18 | 0.0 | 0.458 | 7.147 | 54.2 | 6.0622 | 3.0 | 222.0 | 18.7 | 396 |
| **5** | 0.02985 | 0.0 | 2.18 | 0.0 | 0.458 | 6.430 | 58.7 | 6.0622 | 3.0 | 222.0 | 18.7 | 394 |
| **6** | 0.08829 | 12.5 | 7.87 | 0.0 | 0.524 | 6.012 | 66.6 | 5.5605 | 5.0 | 311.0 | 15.2 | 395 |
| **7** | 0.14455 | 12.5 | 7.87 | 0.0 | 0.524 | 6.172 | 96.1 | 5.9505 | 5.0 | 311.0 | 15.2 | 396 |
| **8** | 0.21124 | 12.5 | 7.87 | 0.0 | 0.524 | 5.631 | 100.0 | 6.0821 | 5.0 | 311.0 | 15.2 | 386 |
| **9** | 0.17004 | 12.5 | 7.87 | 0.0 | 0.524 | 6.004 | 85.9 | 6.5921 | 5.0 | 311.0 | 15.2 | 386 |
| **10** | 0.22489 | 12.5 | 7.87 | 0.0 | 0.524 | 6.377 | 94.3 | 6.3467 | 5.0 | 311.0 | 15.2 | 392 |
| **11** | 0.11747 | 12.5 | 7.87 | 0.0 | 0.524 | 6.009 | 82.9 | 6.2267 | 5.0 | 311.0 | 15.2 | 396 |
| **12** | 0.09378 | 12.5 | 7.87 | 0.0 | 0.524 | 5.889 | 39.0 | 5.4509 | 5.0 | 311.0 | 15.2 | 390 |

| | CRIM | ZN | INDUS | CHAS | NOX | RM | AGE | DIS | RAD | TAX | PTRATIO | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **13** | 0.62976 | 0.0 | 8.14 | 0.0 | 0.538 | 5.949 | 61.8 | 4.7075 | 4.0 | 307.0 | 21.0 | 396 |
| **14** | 0.63796 | 0.0 | 8.14 | 0.0 | 0.538 | 6.096 | 84.5 | 4.4619 | 4.0 | 307.0 | 21.0 | 380 |
| **15** | 0.62739 | 0.0 | 8.14 | 0.0 | 0.538 | 5.834 | 56.5 | 4.4986 | 4.0 | 307.0 | 21.0 | 395 |
| **16** | 1.05393 | 0.0 | 8.14 | 0.0 | 0.538 | 5.935 | 29.3 | 4.4986 | 4.0 | 307.0 | 21.0 | 386 |
| **17** | 0.78420 | 0.0 | 8.14 | 0.0 | 0.538 | 5.990 | 81.7 | 4.2579 | 4.0 | 307.0 | 21.0 | 386 |
| **18** | 0.80271 | 0.0 | 8.14 | 0.0 | 0.538 | 5.456 | 36.6 | 3.7965 | 4.0 | 307.0 | 21.0 | 288 |
| **19** | 0.72580 | 0.0 | 8.14 | 0.0 | 0.538 | 5.727 | 69.5 | 3.7965 | 4.0 | 307.0 | 21.0 | 390 |
| **20** | 1.25179 | 0.0 | 8.14 | 0.0 | 0.538 | 5.570 | 98.1 | 3.7979 | 4.0 | 307.0 | 21.0 | 376 |
| **21** | 0.85204 | 0.0 | 8.14 | 0.0 | 0.538 | 5.965 | 89.2 | 4.0123 | 4.0 | 307.0 | 21.0 | 392 |
| **22** | 1.23247 | 0.0 | 8.14 | 0.0 | 0.538 | 6.142 | 91.7 | 3.9769 | 4.0 | 307.0 | 21.0 | 396 |
| **23** | 0.98843 | 0.0 | 8.14 | 0.0 | 0.538 | 5.813 | 100.0 | 4.0952 | 4.0 | 307.0 | 21.0 | 394 |
| **24** | 0.75026 | 0.0 | 8.14 | 0.0 | 0.538 | 5.924 | 94.1 | 4.3996 | 4.0 | 307.0 | 21.0 | 394 |
| **25** | 0.84054 | 0.0 | 8.14 | 0.0 | 0.538 | 5.599 | 85.7 | 4.4546 | 4.0 | 307.0 | 21.0 | 303 |
| **26** | 0.67191 | 0.0 | 8.14 | 0.0 | 0.538 | 5.813 | 90.3 | 4.6820 | 4.0 | 307.0 | 21.0 | 376 |
| **27** | 0.95577 | 0.0 | 8.14 | 0.0 | 0.538 | 6.047 | 88.8 | 4.4534 | 4.0 | 307.0 | 21.0 | 306 |
| **28** | 0.77299 | 0.0 | 8.14 | 0.0 | 0.538 | 6.495 | 94.4 | 4.4547 | 4.0 | 307.0 | 21.0 | 387 |
| **29** | 1.00245 | 0.0 | 8.14 | 0.0 | 0.538 | 6.674 | 87.3 | 4.2390 | 4.0 | 307.0 | 21.0 | 380 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **476** | 4.87141 | 0.0 | 18.10 | 0.0 | 0.614 | 6.484 | 93.6 | 2.3053 | 24.0 | 666.0 | 20.2 | 396 |
| **477** | 15.02340 | 0.0 | 18.10 | 0.0 | 0.614 | 5.304 | 97.3 | 2.1007 | 24.0 | 666.0 | 20.2 | 349 |
| **478** | 10.23300 | 0.0 | 18.10 | 0.0 | 0.614 | 6.185 | 96.7 | 2.1705 | 24.0 | 666.0 | 20.2 | 379 |
| **479** | 14.33370 | 0.0 | 18.10 | 0.0 | 0.614 | 6.229 | 88.0 | 1.9512 | 24.0 | 666.0 | 20.2 | 383 |

| | CRIM | ZN | INDUS | CHAS | NOX | RM | AGE | DIS | RAD | TAX | PTRATIO | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **480** | 5.82401 | 0.0 | 18.10 | 0.0 | 0.532 | 6.242 | 64.7 | 3.4242 | 24.0 | 666.0 | 20.2 | 396 |
| **481** | 5.70818 | 0.0 | 18.10 | 0.0 | 0.532 | 6.750 | 74.9 | 3.3317 | 24.0 | 666.0 | 20.2 | 393 |
| **482** | 5.73116 | 0.0 | 18.10 | 0.0 | 0.532 | 7.061 | 77.0 | 3.4106 | 24.0 | 666.0 | 20.2 | 395 |
| **483** | 2.81838 | 0.0 | 18.10 | 0.0 | 0.532 | 5.762 | 40.3 | 4.0983 | 24.0 | 666.0 | 20.2 | 392 |
| **484** | 2.37857 | 0.0 | 18.10 | 0.0 | 0.583 | 5.871 | 41.9 | 3.7240 | 24.0 | 666.0 | 20.2 | 370 |
| **485** | 3.67367 | 0.0 | 18.10 | 0.0 | 0.583 | 6.312 | 51.9 | 3.9917 | 24.0 | 666.0 | 20.2 | 388 |
| **486** | 5.69175 | 0.0 | 18.10 | 0.0 | 0.583 | 6.114 | 79.8 | 3.5459 | 24.0 | 666.0 | 20.2 | 392 |
| **487** | 4.83567 | 0.0 | 18.10 | 0.0 | 0.583 | 5.905 | 53.2 | 3.1523 | 24.0 | 666.0 | 20.2 | 388 |
| **488** | 0.15086 | 0.0 | 27.74 | 0.0 | 0.609 | 5.454 | 92.7 | 1.8209 | 4.0 | 711.0 | 20.1 | 395 |
| **489** | 0.18337 | 0.0 | 27.74 | 0.0 | 0.609 | 5.414 | 98.3 | 1.7554 | 4.0 | 711.0 | 20.1 | 344 |
| **490** | 0.20746 | 0.0 | 27.74 | 0.0 | 0.609 | 5.093 | 98.0 | 1.8226 | 4.0 | 711.0 | 20.1 | 318 |
| **491** | 0.10574 | 0.0 | 27.74 | 0.0 | 0.609 | 5.983 | 98.8 | 1.8681 | 4.0 | 711.0 | 20.1 | 390 |
| **492** | 0.11132 | 0.0 | 27.74 | 0.0 | 0.609 | 5.983 | 83.5 | 2.1099 | 4.0 | 711.0 | 20.1 | 396 |
| **493** | 0.17331 | 0.0 | 9.69 | 0.0 | 0.585 | 5.707 | 54.0 | 2.3817 | 6.0 | 391.0 | 19.2 | 396 |
| **494** | 0.27957 | 0.0 | 9.69 | 0.0 | 0.585 | 5.926 | 42.6 | 2.3817 | 6.0 | 391.0 | 19.2 | 396 |
| **495** | 0.17899 | 0.0 | 9.69 | 0.0 | 0.585 | 5.670 | 28.8 | 2.7986 | 6.0 | 391.0 | 19.2 | 393 |
| **496** | 0.28960 | 0.0 | 9.69 | 0.0 | 0.585 | 5.390 | 72.9 | 2.7986 | 6.0 | 391.0 | 19.2 | 396 |
| **497** | 0.26838 | 0.0 | 9.69 | 0.0 | 0.585 | 5.794 | 70.6 | 2.8927 | 6.0 | 391.0 | 19.2 | 396 |
| **498** | 0.23912 | 0.0 | 9.69 | 0.0 | 0.585 | 6.019 | 65.3 | 2.4091 | 6.0 | 391.0 | 19.2 | 396 |
| **499** | 0.17783 | 0.0 | 9.69 | 0.0 | 0.585 | 5.569 | 73.5 | 2.3999 | 6.0 | 391.0 | 19.2 | 395 |
| **500** | 0.22438 | 0.0 | 9.69 | 0.0 | 0.585 | 6.027 | 79.7 | 2.4982 | 6.0 | 391.0 | 19.2 | 396 |
| **501** | 0.06263 | 0.0 | 11.93 | 0.0 | 0.573 | 6.593 | 69.1 | 2.4786 | 1.0 | 273.0 | 21.0 | 391 |

| | CRIM | ZN | INDUS | CHAS | NOX | RM | AGE | DIS | RAD | TAX | PTRATIO | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **502** | 0.04527 | 0.0 | 11.93 | 0.0 | 0.573 | 6.120 | 76.7 | 2.2875 | 1.0 | 273.0 | 21.0 | 396 |
| **503** | 0.06076 | 0.0 | 11.93 | 0.0 | 0.573 | 6.976 | 91.0 | 2.1675 | 1.0 | 273.0 | 21.0 | 396 |
| **504** | 0.10959 | 0.0 | 11.93 | 0.0 | 0.573 | 6.794 | 89.3 | 2.3889 | 1.0 | 273.0 | 21.0 | 393 |
| **505** | 0.04741 | 0.0 | 11.93 | 0.0 | 0.573 | 6.030 | 80.8 | 2.5050 | 1.0 | 273.0 | 21.0 | 396 |

506 rows × 13 columns

In [46]:
```
target=pd.DataFrame(boston.target,columns=['target'])
target
```

Out[46]:

| | target |
|---|---|
| **0** | 24.0 |
| **1** | 21.6 |
| **2** | 34.7 |
| **3** | 33.4 |
| **4** | 36.2 |
| **5** | 28.7 |
| **6** | 22.9 |
| **7** | 27.1 |
| **8** | 16.5 |
| **9** | 18.9 |
| **10** | 15.0 |
| **11** | 18.9 |

| | target |
|---|---|
| **12** | 21.7 |
| **13** | 20.4 |
| **14** | 18.2 |
| **15** | 19.9 |
| **16** | 23.1 |
| **17** | 17.5 |
| **18** | 20.2 |
| **19** | 18.2 |
| **20** | 13.6 |
| **21** | 19.6 |
| **22** | 15.2 |
| **23** | 14.5 |
| **24** | 15.6 |
| **25** | 13.9 |
| **26** | 16.6 |
| **27** | 14.8 |
| **28** | 18.4 |
| **29** | 21.0 |
| **...** | ... |
| **476** | 16.7 |
| **477** | 12.0 |
| **478** | 14.6 |

| | target |
|---|---|
| **479** | 21.4 |
| **480** | 23.0 |
| **481** | 23.7 |
| **482** | 25.0 |
| **483** | 21.8 |
| **484** | 20.6 |
| **485** | 21.2 |
| **486** | 19.1 |
| **487** | 20.6 |
| **488** | 15.2 |
| **489** | 7.0 |
| **490** | 8.1 |
| **491** | 13.6 |
| **492** | 20.1 |
| **493** | 21.8 |
| **494** | 24.5 |
| **495** | 23.1 |
| **496** | 19.7 |
| **497** | 18.3 |
| **498** | 21.2 |
| **499** | 17.5 |
| **500** | 16.8 |

| | target |
|---|---|
| **501** | 22.4 |
| **502** | 20.6 |
| **503** | 23.9 |
| **504** | 22.0 |
| **505** | 11.9 |

506 rows × 1 columns

In [17]:
```
max(target['target'])
```
Out[17]: 50.0

In [18]:
```
min(target['target'])
```
Out[18]: 5.0

In [47]:
```
#Concat the features and the target into a single data frame
df = pd.concat([features,target],axis=1)
df
```
Out[47]:

| | CRIM | ZN | INDUS | CHAS | NOX | RM | AGE | DIS | RAD | TAX | PTRATIO | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.00632 | 18.0 | 2.31 | 0.0 | 0.538 | 6.575 | 65.2 | 4.0900 | 1.0 | 296.0 | 15.3 | 396 |
| **1** | 0.02731 | 0.0 | 7.07 | 0.0 | 0.469 | 6.421 | 78.9 | 4.9671 | 2.0 | 242.0 | 17.8 | 396 |
| **2** | 0.02729 | 0.0 | 7.07 | 0.0 | 0.469 | 7.185 | 61.1 | 4.9671 | 2.0 | 242.0 | 17.8 | 392 |
| **3** | 0.03237 | 0.0 | 2.18 | 0.0 | 0.458 | 6.998 | 45.8 | 6.0622 | 3.0 | 222.0 | 18.7 | 394 |
| **4** | 0.06905 | 0.0 | 2.18 | 0.0 | 0.458 | 7.147 | 54.2 | 6.0622 | 3.0 | 222.0 | 18.7 | 396 |
| **5** | 0.02985 | 0.0 | 2.18 | 0.0 | 0.458 | 6.430 | 58.7 | 6.0622 | 3.0 | 222.0 | 18.7 | 394 |

| | CRIM | ZN | INDUS | CHAS | NOX | RM | AGE | DIS | RAD | TAX | PTRATIO | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **6** | 0.08829 | 12.5 | 7.87 | 0.0 | 0.524 | 6.012 | 66.6 | 5.5605 | 5.0 | 311.0 | 15.2 | 395 |
| **7** | 0.14455 | 12.5 | 7.87 | 0.0 | 0.524 | 6.172 | 96.1 | 5.9505 | 5.0 | 311.0 | 15.2 | 396 |
| **8** | 0.21124 | 12.5 | 7.87 | 0.0 | 0.524 | 5.631 | 100.0 | 6.0821 | 5.0 | 311.0 | 15.2 | 386 |
| **9** | 0.17004 | 12.5 | 7.87 | 0.0 | 0.524 | 6.004 | 85.9 | 6.5921 | 5.0 | 311.0 | 15.2 | 386 |
| **10** | 0.22489 | 12.5 | 7.87 | 0.0 | 0.524 | 6.377 | 94.3 | 6.3467 | 5.0 | 311.0 | 15.2 | 392 |
| **11** | 0.11747 | 12.5 | 7.87 | 0.0 | 0.524 | 6.009 | 82.9 | 6.2267 | 5.0 | 311.0 | 15.2 | 396 |
| **12** | 0.09378 | 12.5 | 7.87 | 0.0 | 0.524 | 5.889 | 39.0 | 5.4509 | 5.0 | 311.0 | 15.2 | 390 |
| **13** | 0.62976 | 0.0 | 8.14 | 0.0 | 0.538 | 5.949 | 61.8 | 4.7075 | 4.0 | 307.0 | 21.0 | 396 |
| **14** | 0.63796 | 0.0 | 8.14 | 0.0 | 0.538 | 6.096 | 84.5 | 4.4619 | 4.0 | 307.0 | 21.0 | 380 |
| **15** | 0.62739 | 0.0 | 8.14 | 0.0 | 0.538 | 5.834 | 56.5 | 4.4986 | 4.0 | 307.0 | 21.0 | 395 |
| **16** | 1.05393 | 0.0 | 8.14 | 0.0 | 0.538 | 5.935 | 29.3 | 4.4986 | 4.0 | 307.0 | 21.0 | 386 |
| **17** | 0.78420 | 0.0 | 8.14 | 0.0 | 0.538 | 5.990 | 81.7 | 4.2579 | 4.0 | 307.0 | 21.0 | 386 |
| **18** | 0.80271 | 0.0 | 8.14 | 0.0 | 0.538 | 5.456 | 36.6 | 3.7965 | 4.0 | 307.0 | 21.0 | 288 |
| **19** | 0.72580 | 0.0 | 8.14 | 0.0 | 0.538 | 5.727 | 69.5 | 3.7965 | 4.0 | 307.0 | 21.0 | 390 |
| **20** | 1.25179 | 0.0 | 8.14 | 0.0 | 0.538 | 5.570 | 98.1 | 3.7979 | 4.0 | 307.0 | 21.0 | 376 |
| **21** | 0.85204 | 0.0 | 8.14 | 0.0 | 0.538 | 5.965 | 89.2 | 4.0123 | 4.0 | 307.0 | 21.0 | 392 |
| **22** | 1.23247 | 0.0 | 8.14 | 0.0 | 0.538 | 6.142 | 91.7 | 3.9769 | 4.0 | 307.0 | 21.0 | 396 |
| **23** | 0.98843 | 0.0 | 8.14 | 0.0 | 0.538 | 5.813 | 100.0 | 4.0952 | 4.0 | 307.0 | 21.0 | 394 |
| **24** | 0.75026 | 0.0 | 8.14 | 0.0 | 0.538 | 5.924 | 94.1 | 4.3996 | 4.0 | 307.0 | 21.0 | 394 |
| **25** | 0.84054 | 0.0 | 8.14 | 0.0 | 0.538 | 5.599 | 85.7 | 4.4546 | 4.0 | 307.0 | 21.0 | 303 |
| **26** | 0.67191 | 0.0 | 8.14 | 0.0 | 0.538 | 5.813 | 90.3 | 4.6820 | 4.0 | 307.0 | 21.0 | 376 |
| **27** | 0.95577 | 0.0 | 8.14 | 0.0 | 0.538 | 6.047 | 88.8 | 4.4534 | 4.0 | 307.0 | 21.0 | 306 |

| | CRIM | ZN | INDUS | CHAS | NOX | RM | AGE | DIS | RAD | TAX | PTRATIO | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **28** | 0.77299 | 0.0 | 8.14 | 0.0 | 0.538 | 6.495 | 94.4 | 4.4547 | 4.0 | 307.0 | 21.0 | 387 |
| **29** | 1.00245 | 0.0 | 8.14 | 0.0 | 0.538 | 6.674 | 87.3 | 4.2390 | 4.0 | 307.0 | 21.0 | 380 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **476** | 4.87141 | 0.0 | 18.10 | 0.0 | 0.614 | 6.484 | 93.6 | 2.3053 | 24.0 | 666.0 | 20.2 | 396 |
| **477** | 15.02340 | 0.0 | 18.10 | 0.0 | 0.614 | 5.304 | 97.3 | 2.1007 | 24.0 | 666.0 | 20.2 | 349 |
| **478** | 10.23300 | 0.0 | 18.10 | 0.0 | 0.614 | 6.185 | 96.7 | 2.1705 | 24.0 | 666.0 | 20.2 | 379 |
| **479** | 14.33370 | 0.0 | 18.10 | 0.0 | 0.614 | 6.229 | 88.0 | 1.9512 | 24.0 | 666.0 | 20.2 | 383 |
| **480** | 5.82401 | 0.0 | 18.10 | 0.0 | 0.532 | 6.242 | 64.7 | 3.4242 | 24.0 | 666.0 | 20.2 | 396 |
| **481** | 5.70818 | 0.0 | 18.10 | 0.0 | 0.532 | 6.750 | 74.9 | 3.3317 | 24.0 | 666.0 | 20.2 | 393 |
| **482** | 5.73116 | 0.0 | 18.10 | 0.0 | 0.532 | 7.061 | 77.0 | 3.4106 | 24.0 | 666.0 | 20.2 | 395 |
| **483** | 2.81838 | 0.0 | 18.10 | 0.0 | 0.532 | 5.762 | 40.3 | 4.0983 | 24.0 | 666.0 | 20.2 | 392 |
| **484** | 2.37857 | 0.0 | 18.10 | 0.0 | 0.583 | 5.871 | 41.9 | 3.7240 | 24.0 | 666.0 | 20.2 | 370 |
| **485** | 3.67367 | 0.0 | 18.10 | 0.0 | 0.583 | 6.312 | 51.9 | 3.9917 | 24.0 | 666.0 | 20.2 | 388 |
| **486** | 5.69175 | 0.0 | 18.10 | 0.0 | 0.583 | 6.114 | 79.8 | 3.5459 | 24.0 | 666.0 | 20.2 | 392 |
| **487** | 4.83567 | 0.0 | 18.10 | 0.0 | 0.583 | 5.905 | 53.2 | 3.1523 | 24.0 | 666.0 | 20.2 | 388 |
| **488** | 0.15086 | 0.0 | 27.74 | 0.0 | 0.609 | 5.454 | 92.7 | 1.8209 | 4.0 | 711.0 | 20.1 | 395 |
| **489** | 0.18337 | 0.0 | 27.74 | 0.0 | 0.609 | 5.414 | 98.3 | 1.7554 | 4.0 | 711.0 | 20.1 | 344 |
| **490** | 0.20746 | 0.0 | 27.74 | 0.0 | 0.609 | 5.093 | 98.0 | 1.8226 | 4.0 | 711.0 | 20.1 | 318 |
| **491** | 0.10574 | 0.0 | 27.74 | 0.0 | 0.609 | 5.983 | 98.8 | 1.8681 | 4.0 | 711.0 | 20.1 | 390 |
| **492** | 0.11132 | 0.0 | 27.74 | 0.0 | 0.609 | 5.983 | 83.5 | 2.1099 | 4.0 | 711.0 | 20.1 | 396 |
| **493** | 0.17331 | 0.0 | 9.69 | 0.0 | 0.585 | 5.707 | 54.0 | 2.3817 | 6.0 | 391.0 | 19.2 | 396 |
| **494** | 0.27957 | 0.0 | 9.69 | 0.0 | 0.585 | 5.926 | 42.6 | 2.3817 | 6.0 | 391.0 | 19.2 | 396 |

| | CRIM | ZN | INDUS | CHAS | NOX | RM | AGE | DIS | RAD | TAX | PTRATIO | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **495** | 0.17899 | 0.0 | 9.69 | 0.0 | 0.585 | 5.670 | 28.8 | 2.7986 | 6.0 | 391.0 | 19.2 | 393 |
| **496** | 0.28960 | 0.0 | 9.69 | 0.0 | 0.585 | 5.390 | 72.9 | 2.7986 | 6.0 | 391.0 | 19.2 | 396 |
| **497** | 0.26838 | 0.0 | 9.69 | 0.0 | 0.585 | 5.794 | 70.6 | 2.8927 | 6.0 | 391.0 | 19.2 | 396 |
| **498** | 0.23912 | 0.0 | 9.69 | 0.0 | 0.585 | 6.019 | 65.3 | 2.4091 | 6.0 | 391.0 | 19.2 | 396 |
| **499** | 0.17783 | 0.0 | 9.69 | 0.0 | 0.585 | 5.569 | 73.5 | 2.3999 | 6.0 | 391.0 | 19.2 | 395 |
| **500** | 0.22438 | 0.0 | 9.69 | 0.0 | 0.585 | 6.027 | 79.7 | 2.4982 | 6.0 | 391.0 | 19.2 | 396 |
| **501** | 0.06263 | 0.0 | 11.93 | 0.0 | 0.573 | 6.593 | 69.1 | 2.4786 | 1.0 | 273.0 | 21.0 | 391 |
| **502** | 0.04527 | 0.0 | 11.93 | 0.0 | 0.573 | 6.120 | 76.7 | 2.2875 | 1.0 | 273.0 | 21.0 | 396 |
| **503** | 0.06076 | 0.0 | 11.93 | 0.0 | 0.573 | 6.976 | 91.0 | 2.1675 | 1.0 | 273.0 | 21.0 | 396 |
| **504** | 0.10959 | 0.0 | 11.93 | 0.0 | 0.573 | 6.794 | 89.3 | 2.3889 | 1.0 | 273.0 | 21.0 | 393 |
| **505** | 0.04741 | 0.0 | 11.93 | 0.0 | 0.573 | 6.030 | 80.8 | 2.5050 | 1.0 | 273.0 | 21.0 | 396 |

506 rows × 14 columns

In [49]:

```
#describe option is used to provide a statistical description of the da
taset
#round(decimals=2) is used to set the precision

df.describe().round(decimals=2)
```

Out[49]:

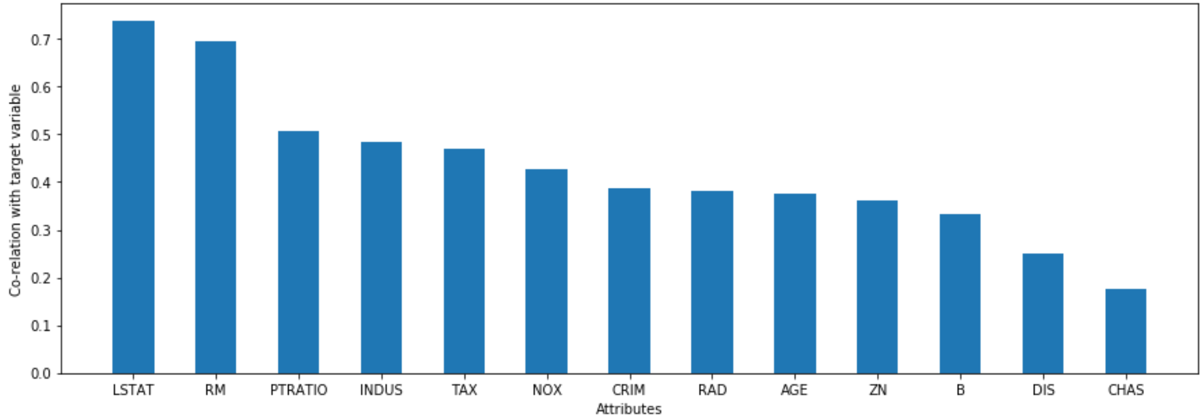
| | CRIM | ZN | INDUS | CHAS | NOX | RM | AGE | DIS | RAD | TAX | PTRA |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 506.00 | 506.00 | 506.00 | 506.00 | 506.00 | 506.00 | 506.00 | 506.00 | 506.00 | 506.00 | 506.0 |
| **mean** | 3.59 | 11.36 | 11.14 | 0.07 | 0.55 | 6.28 | 68.57 | 3.80 | 9.55 | 408.24 | 18.46 |
| **std** | 8.60 | 23.32 | 6.86 | 0.25 | 0.12 | 0.70 | 28.15 | 2.11 | 8.71 | 168.54 | 2.16 |
| **min** | 0.01 | 0.00 | 0.46 | 0.00 | 0.38 | 3.56 | 2.90 | 1.13 | 1.00 | 187.00 | 12.60 |

| | CRIM | ZN | INDUS | CHAS | NOX | RM | AGE | DIS | RAD | TAX | PTRA |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **25%** | 0.08 | 0.00 | 5.19 | 0.00 | 0.45 | 5.89 | 45.02 | 2.10 | 4.00 | 279.00 | 17.40 |
| **50%** | 0.26 | 0.00 | 9.69 | 0.00 | 0.54 | 6.21 | 77.50 | 3.21 | 5.00 | 330.00 | 19.05 |
| **75%** | 3.65 | 12.50 | 18.10 | 0.00 | 0.62 | 6.62 | 94.07 | 5.19 | 24.00 | 666.00 | 20.20 |
| **max** | 88.98 | 100.00 | 27.74 | 1.00 | 0.87 | 8.78 | 100.00 | 12.13 | 24.00 | 711.00 | 22.00 |

In [50]:

```
corr=df.corr('pearson')

#taking the abs value of the co relations using list comprehension
# this corrs has all the co relations wrt the target column
#df is a matrix.
corrs=[abs(corr[attr]['target']) for attr in list(features)]

#making a list of tuple pairs having tuples in the form of (corrs,featu
res)

l=list(zip(corrs,list(features)))

#sort the list in desc order with the co realtion value as the key for
 sorting.

l.sort(key= lambda x:x[0],reverse=True)


#We have to unzip the co realtion and features pair
#zip(*l ) is used to take the list in form of [[a,b,c],[d,e,f],[g,h,i]]
# returns a list of[[a,d,g],[b,e,h],[c,f,i]]

corrs, labels = list(zip((*l)))

#plotting the co realtions as bargraph with respect to the target
index = np.arange(len(labels)) #arrange is used to provide a range of v
alues within a given range
plt.figure(figsize=(15,5))
```

```python
plt.bar(index,corrs,width=0.5)
plt.xlabel('Attributes')
plt.ylabel('Co-relation with target variable')
plt.xticks(index,labels)
plt.show()
```




In [51]:
```python
#Data Preprocessing
#data normalisation


X=df['LSTAT'].values
Y=df['target'].values
```

In [52]:
```python
print(X[:5])
```

```
[4.98 9.14 4.03 2.94 5.33]
```

In [53]:
```python
x_scaler=MinMaxScaler()
X = x_scaler.fit_transform(X.reshape(-1,1))
X = X[:, -1]
```

```
y_scaler=MinMaxScaler()
Y=y_scaler.fit_transform(Y.reshape(-1,1))
Y = Y[:, -1]
```

In [54]:
```
#error function or cost Function
def error(m,x,c,t):
    N=x.size
    e=sum(((m*x + c)-t) ** 2)
    return e*(1/2*N)
```

In [56]:
```
#Spliting the dataset
xtrain, xtest,ytrain, ytest = train_test_split(X,Y,test_size=0.2)
#0.2 refers to the 20 % of the values in the dataset chosen randomly
```

In [69]:
```
#update function to be used inside the gradient descent

def update(m,x,c,t,learning_rate):
    grad_m=sum(2*((m*x+c)-t)*x)
    grad_c=sum(2*((m*x+c)-t))
    m=m-grad_m*learning_rate
    c=c-grad_c*learning_rate
    return m,c
```

In [70]:
```
#init_m is the initial estimate of m and init_c is the initial estimate
 of the c
#error threshold is the threshold value below which the gradient descen
t must stop
def gradient_descent(init_m,init_c,x,t,learning_rate,iterations,error_t
hreshold):
    m=init_m
    c=init_c
    error_values=list()
    mc_values=list()
    for i in range(iterations):
        e=error(m,x,c,t)
        if e < error_threshold:
            print('Error less than the threshold. Stopping Gradient Des
cent...')
```

```
            break
        error_values.append(e)
        m,c=update(m,x,c,t,learning_rate)
        mc_values.append((m,c))
    return m,c,error_values,mc_values
```

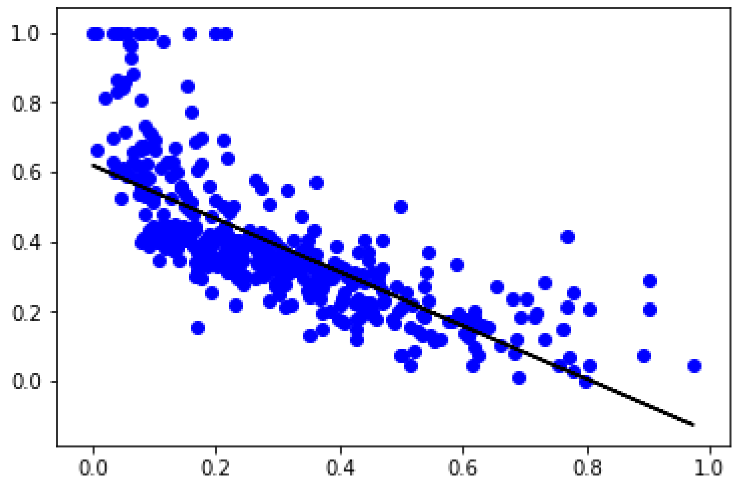
In [72]:
```
#setting the initial values

init_m=0.9
init_c=0
learning_rate=0.001
iterations=250
error_threshold=0.001

m,c,error_values,mc_values=gradient_descent(init_m,init_c,xtrain,ytrain
,learning_rate,iterations,error_threshold)
```

In [77]:
```
#visualising the training
#AS the number of iterations increases the changes made in the line are
 less noticable,this also consumes more CPU time
#take some small interval of data,slecting every 5th value


mc_values_anim=mc_values[0:250:5]
```

In [ ]:
```

```

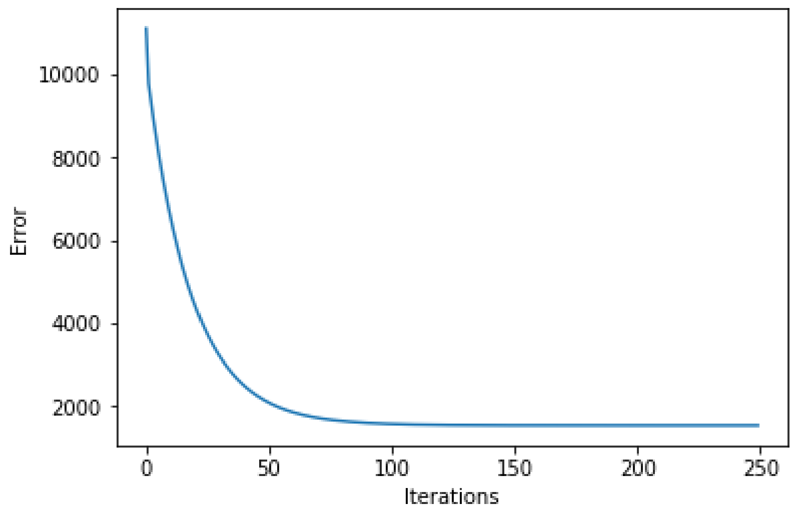
In [88]:
```
#visualsing the the egression line
plt.scatter(xtrain,ytrain,color='b')
plt.plot(xtrain,(m*xtrain+c),color='black')
```

Out[88]: [<matplotlib.lines.Line2D at 0x1c22ecb9a20>]




In [89]:
```
#plotting the error values against the no of iterations
plt.plot(np.arange(len(error_values)),error_values)
plt.ylabel('Error')
plt.xlabel('Iterations')
```

Out[89]:
```
Text(0.5,0,'Iterations')
```



In [90]:

```
#calculate the operations on the test set as a vectorized operation

predicted=(m*xtest)+c
```

In [91]:

```
#calculate MSE for the predicted value on the training set

mean_squared_error(ytest,predicted)
```

Out[91]: 0.019324299194478884

In [92]:

```
#put xtest ytest and predicted values into a single dataframe so that w
e can see the predicted values alonside the testing set

p=pd.DataFrame(list(zip(xtest,ytest,predicted)),columns=['x','target_y'
,'predicted_y'])
p.head()#returns top n rows 5 defeault
```

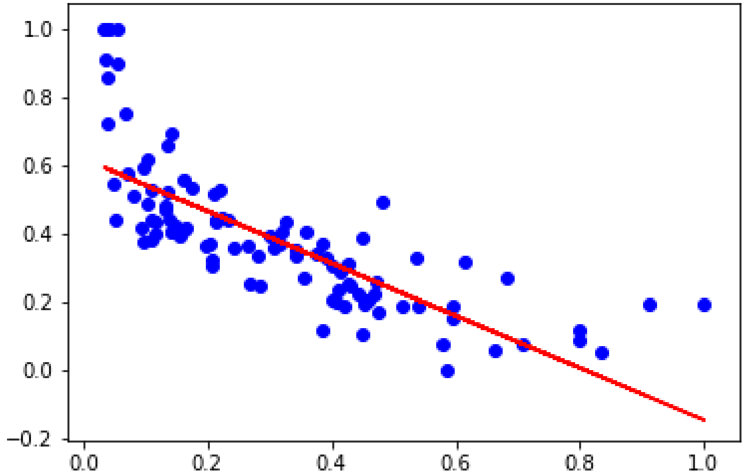
Out[92]:

| | x | target_y | predicted_y |
|---|---|---|---|
| **0** | 0.161976 | 0.557778 | 0.494405 |
| **1** | 0.042219 | 1.000000 | 0.586073 |
| **2** | 0.476821 | 0.171111 | 0.253405 |
| **3** | 0.354857 | 0.268889 | 0.346763 |
| **4** | 0.613962 | 0.317778 | 0.148429 |

In [93]:

```
plt.scatter(xtest,ytest,color='b')
plt.plot(xtest,predicted,color='r')
```

Out[93]: [<matplotlib.lines.Line2D at 0x1c22b69db38>]




In [95]:

```
#reshape to the shape required by the scaler
predicted=predicted.reshape(-1,1)
xtest=xtest.reshape(-1,1)
ytest=ytest.reshape(-1,1)

x_scaled=x_scaler.inverse_transform(xtest)
y_scaled=y_scaler.inverse_transform(ytest)
predicted_scaled=y_scaler.inverse_transform(predicted)

x_scaled=x_scaled[:,-1]
y_scaled=y_scaled[:,-1]
predicted_scaled=predicted_scaled[:,-1]

p=pd.DataFrame(list(zip(x_scaled,y_scaled,predicted_scaled)),columns=[
'x','target_y','predicted_y'])


p=p.round(decimals=2)
p.head(10)
```

Out[95]:

| | x | target_y | predicted_y |
|---|---|---|---|

| | x | target_y | predicted_y |
|---|---|---|---|
| **0** | 7.60 | 30.1 | 27.25 |
| **1** | 3.26 | 50.0 | 31.37 |
| **2** | 19.01 | 12.7 | 16.40 |
| **3** | 14.59 | 17.1 | 20.60 |
| **4** | 23.98 | 19.3 | 11.68 |
| **5** | 30.63 | 8.8 | 5.36 |
| **6** | 26.40 | 17.2 | 9.38 |
| **7** | 17.09 | 18.7 | 18.23 |
| **8** | 5.49 | 32.7 | 29.25 |
| **9** | 5.70 | 28.7 | 29.05 |