```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Finding Price

```
df = pd.read_csv("HousingData.csv")

df.head(),df.tail()
```

```
(       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  296
15.3
 1  0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671    2  242
17.8
 2  0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671    2  242
17.8
 3  0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622    3  222
18.7
 4  0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622    3  222
18.7

         B  LSTAT  MEDV
 0  396.90   4.98  24.0
 1  396.90   9.14  21.6
 2  392.83   4.03  34.7
 3  394.63   2.94  33.4
 4  396.90    NaN  36.2  ,
          CRIM   ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD  TAX
PTRATIO  \
 501  0.06263  0.0  11.93   0.0  0.573  6.593  69.1  2.4786    1  273
21.0
 502  0.04527  0.0  11.93   0.0  0.573  6.120  76.7  2.2875    1  273
21.0
 503  0.06076  0.0  11.93   0.0  0.573  6.976  91.0  2.1675    1  273
21.0
 504  0.10959  0.0  11.93   0.0  0.573  6.794  89.3  2.3889    1  273
21.0
 505  0.04741  0.0  11.93   0.0  0.573  6.030   NaN  2.5050    1  273
21.0

           B  LSTAT  MEDV
 501  391.99    NaN  22.4
 502  396.90   9.08  20.6
 503  396.90   5.64  23.9
```

```
504  393.45   6.48  22.0
505  396.90   7.88  11.9  )
```

Checking for dulicate Data

```python
df.duplicated().sum()
```

```
0
```

Data Inforamtion

```python
df.describe()
```

```
             CRIM          ZN       INDUS        CHAS         NOX
RM  \
count  486.000000  486.000000  486.000000  486.000000  506.000000
506.000000
mean     3.611874   11.211934   11.083992    0.069959    0.554695
6.284634
std      8.720192   23.388876    6.835896    0.255340    0.115878
0.702617
min      0.006320    0.000000    0.460000    0.000000    0.385000
3.561000
25%      0.081900    0.000000    5.190000    0.000000    0.449000
5.885500
50%      0.253715    0.000000    9.690000    0.000000    0.538000
6.208500
75%      3.560263   12.500000   18.100000    0.000000    0.624000
6.623500
max     88.976200  100.000000   27.740000    1.000000    0.871000
8.780000

              AGE         DIS         RAD         TAX     PTRATIO
B  \
count  486.000000  506.000000  506.000000  506.000000  506.000000
506.000000
mean    68.518519    3.795043    9.549407  408.237154   18.455534
356.674032
std     27.999513    2.105710    8.707259  168.537116    2.164946
91.294864
min      2.900000    1.129600    1.000000  187.000000   12.600000
0.320000
25%     45.175000    2.100175    4.000000  279.000000   17.400000
375.377500
50%     76.800000    3.207450    5.000000  330.000000   19.050000
391.440000
```

```
75%      93.975000    5.188425   24.000000  666.000000   20.200000
396.225000
max     100.000000   12.126500   24.000000  711.000000   22.000000
396.900000

            LSTAT        MEDV
count  486.000000  506.000000
mean    12.715432   22.532806
std      7.155871    9.197104
min      1.730000    5.000000
25%      7.125000   17.025000
50%     11.430000   21.200000
75%     16.955000   25.000000
max     37.970000   50.000000
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   CRIM     486 non-null    float64
 1   ZN       486 non-null    float64
 2   INDUS    486 non-null    float64
 3   CHAS     486 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      486 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    int64
 9   TAX      506 non-null    int64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    486 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 55.5 KB
```

```
a = df.isnull().mean()
```

```
df.isnull().sum()
```

```
CRIM       20
ZN         20
INDUS      20
CHAS       20
NOX         0
RM          0
AGE        20
```

```
DIS          0
RAD          0
TAX          0
PTRATIO      0
B            0
LSTAT       20
MEDV         0
dtype: int64
```

```
# 3 percent in each column is missing
df.isnull().mean()*100
```

```
CRIM       3.952569
ZN         3.952569
INDUS      3.952569
CHAS       3.952569
NOX        0.000000
RM         0.000000
AGE        3.952569
DIS        0.000000
RAD        0.000000
TAX        0.000000
PTRATIO    0.000000
B          0.000000
LSTAT      3.952569
MEDV       0.000000
dtype: float64
```

```
#Droping missing data

df.dropna(inplace = True)

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 394 entries, 0 to 504
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     394 non-null    float64
 1   ZN       394 non-null    float64
 2   INDUS    394 non-null    float64
 3   CHAS     394 non-null    float64
 4   NOX      394 non-null    float64
 5   RM       394 non-null    float64
 6   AGE      394 non-null    float64
 7   DIS      394 non-null    float64
 8   RAD      394 non-null    int64  
 9   TAX      394 non-null    int64  
 10  PTRATIO  394 non-null    float64
 11  B        394 non-null    float64
```

```
 12  LSTAT    394 non-null    float64
 13  MEDV     394 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 46.2 KB
```

```
df.describe()
```

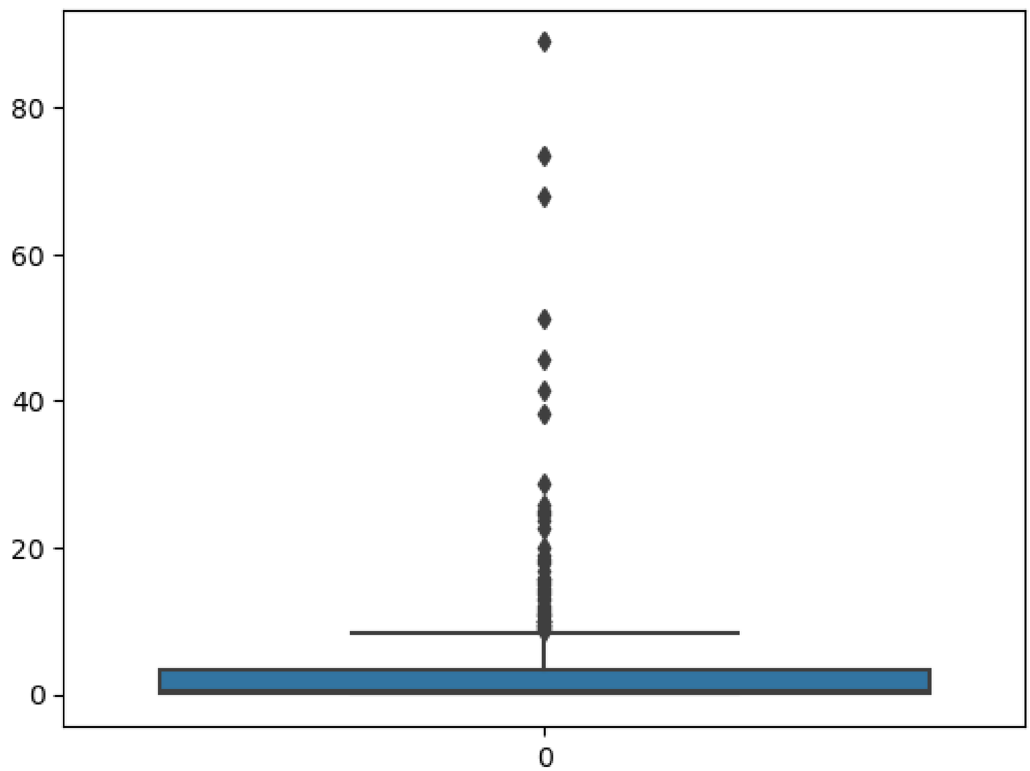
```
             CRIM          ZN       INDUS        CHAS         NOX
RM  \
count  394.000000  394.000000  394.000000  394.000000  394.000000
394.000000
mean     3.690136   11.460660   11.000863    0.068528    0.553215
6.280015
std      9.202423   23.954082    6.908364    0.252971    0.113112
0.697985
min      0.006320    0.000000    0.460000    0.000000    0.389000
3.561000
25%      0.081955    0.000000    5.130000    0.000000    0.453000
5.879250
50%      0.268880    0.000000    8.560000    0.000000    0.538000
6.201500
75%      3.435973   12.500000   18.100000    0.000000    0.624000
6.605500
max     88.976200  100.000000   27.740000    1.000000    0.871000
8.780000

              AGE         DIS         RAD         TAX     PTRATIO
B  \
count  394.000000  394.000000  394.000000  394.000000  394.000000
394.000000
mean    68.932741    3.805268    9.403553  406.431472   18.537563
358.490939
std     27.888705    2.098571    8.633451  168.312419    2.166460
89.283295
min      2.900000    1.129600    1.000000  187.000000   12.600000
2.600000
25%     45.475000    2.110100    4.000000  280.250000   17.400000
376.707500
50%     77.700000    3.199200    5.000000  330.000000   19.100000
392.190000
75%     94.250000    5.116700   24.000000  666.000000   20.200000
396.900000
max    100.000000   12.126500   24.000000  711.000000   22.000000
396.900000

            LSTAT        MEDV
count  394.000000  394.000000
mean    12.769112   22.359645
std      7.308430    9.142979
min      1.730000    5.000000
```

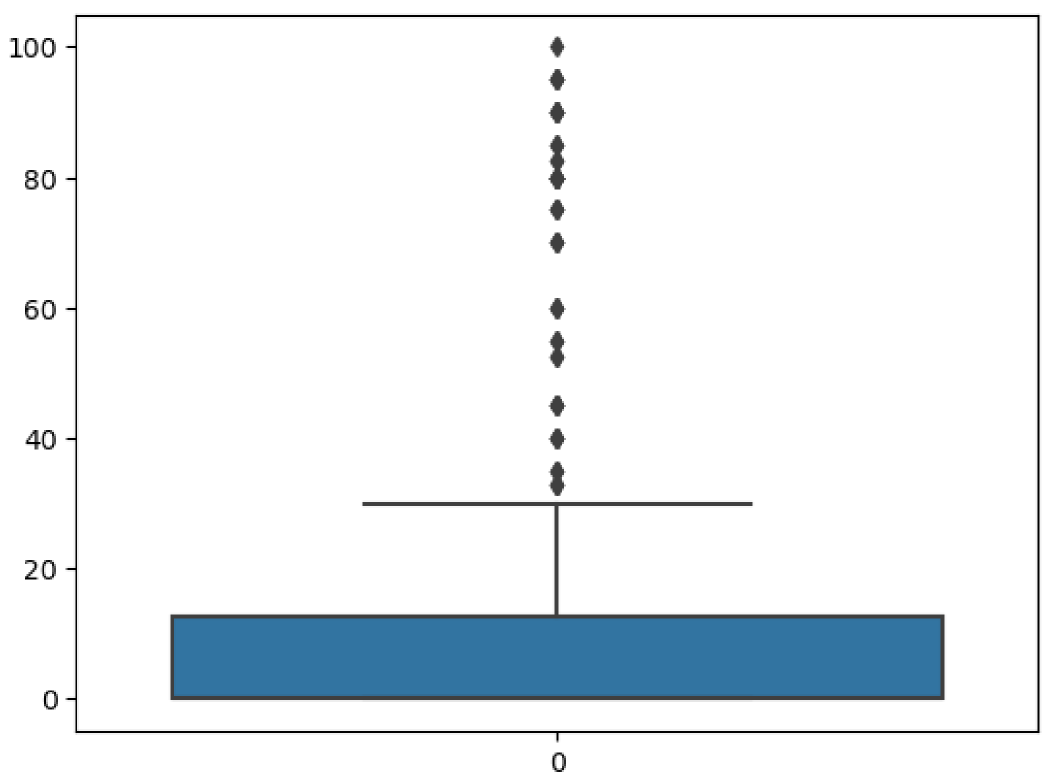
```
25%       7.125000   16.800000
50%      11.300000   21.050000
75%      17.117500   25.000000
max      37.970000   50.000000
```

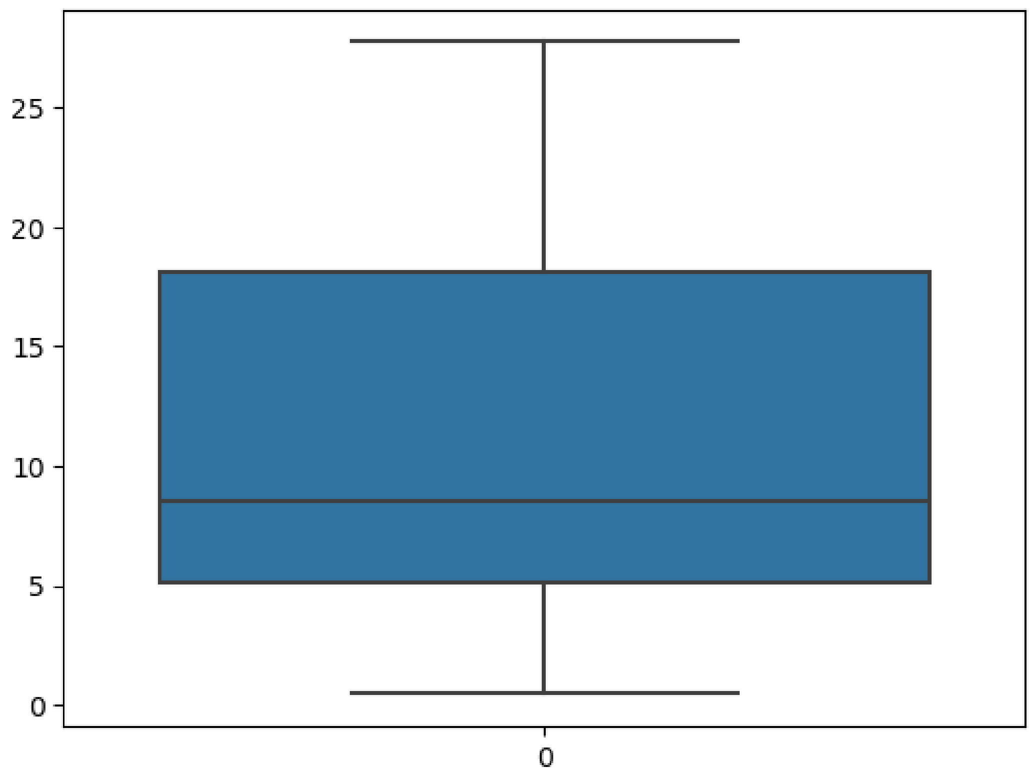
```
df.skew()
```

```
CRIM       5.256934
ZN         2.258275
INDUS      0.358792
CHAS       3.428643
NOX        0.703377
RM         0.487558
AGE       -0.594880
DIS        1.032625
RAD        1.050144
TAX        0.692876
PTRATIO   -0.884475
B         -2.987695
LSTAT      0.942665
MEDV       1.065946
dtype: float64
```

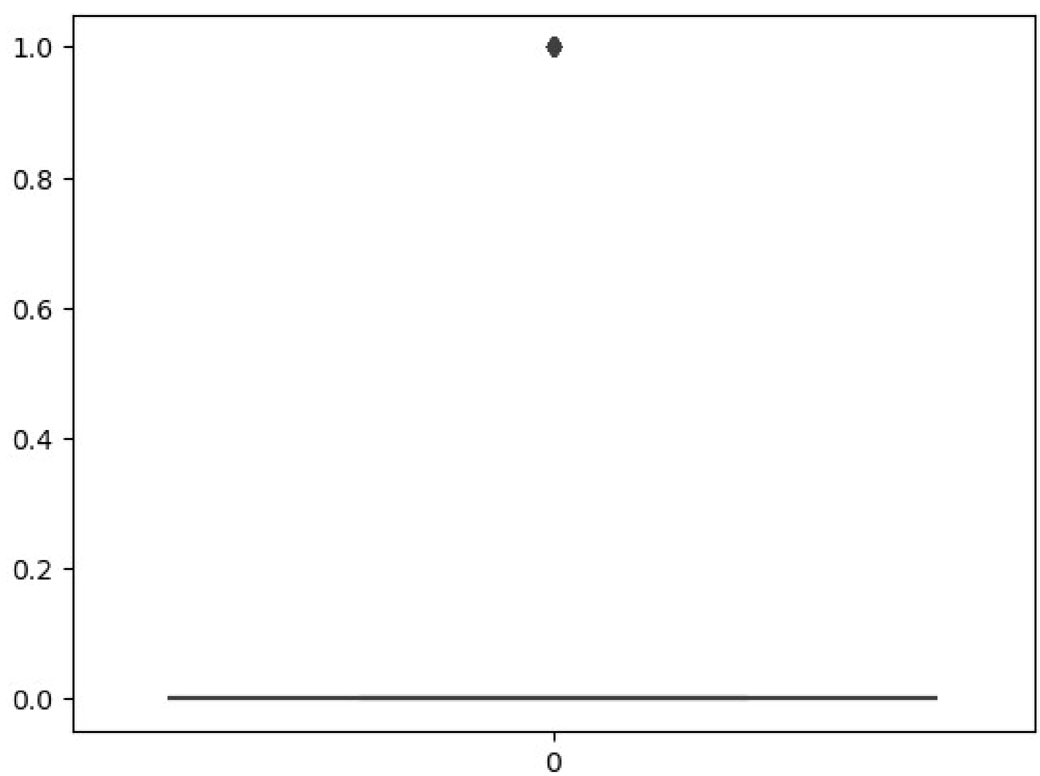
```
def boxplots(data_frame):
    for i in data_frame:
        sns.boxplot(df[i])
        plt.show()

# boxplot of each column
boxplots(df)
```


80
60
40
20
0
0

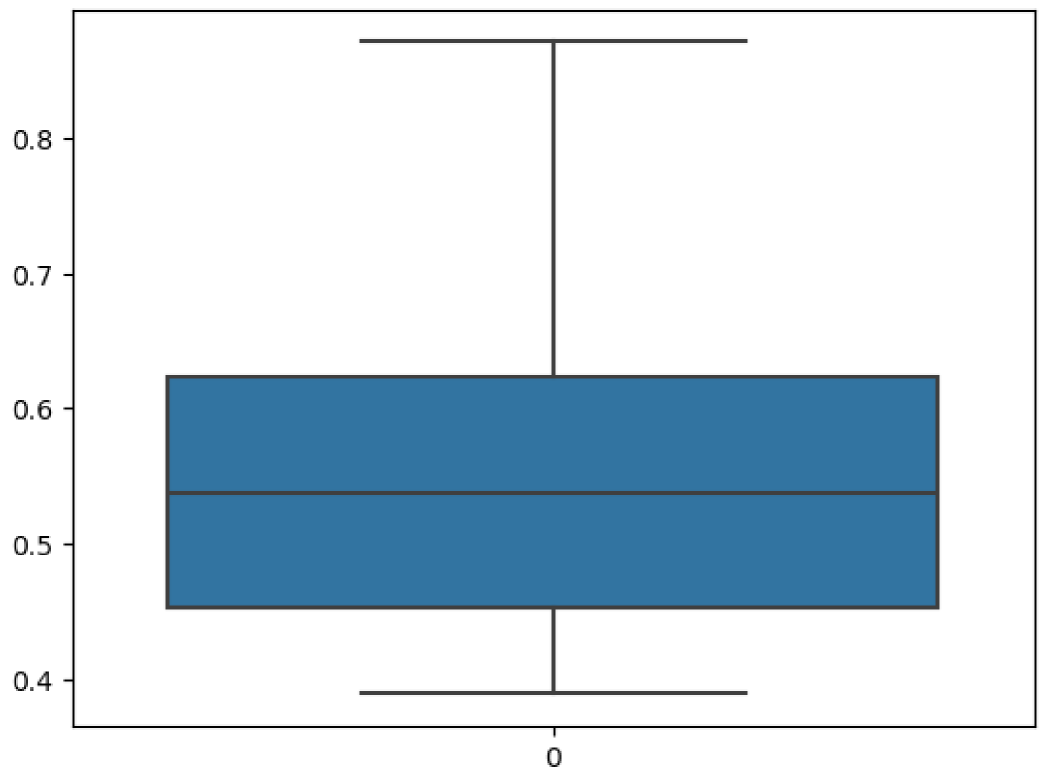

100
80
60
40
20
0
0



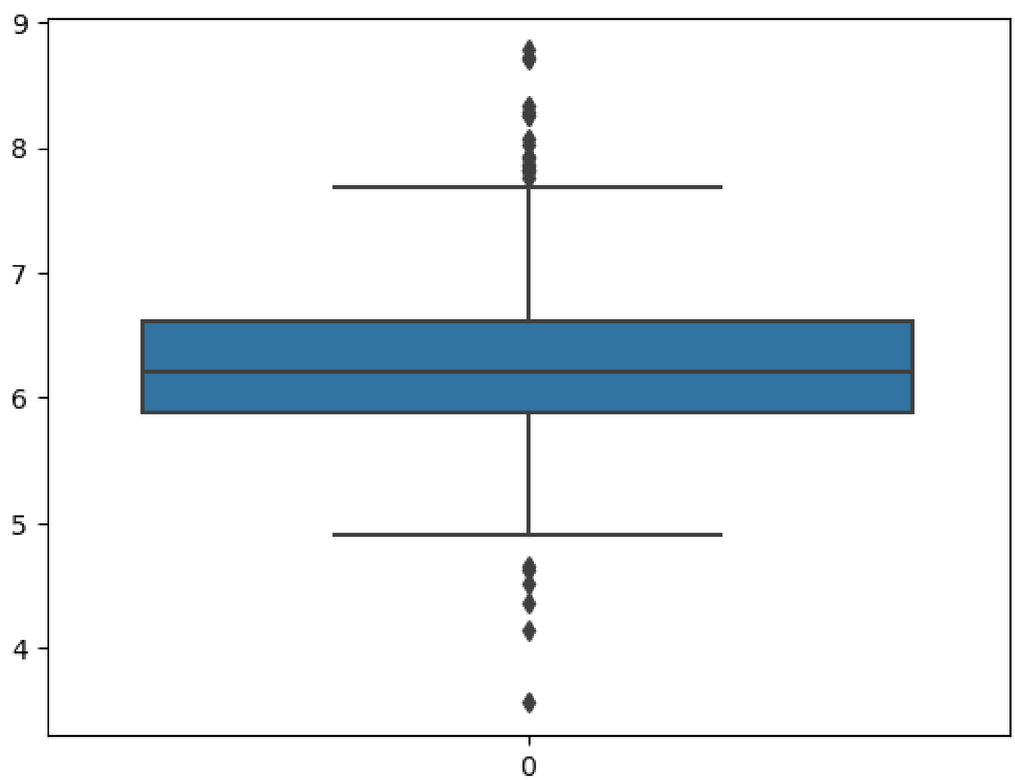
0
5
10
15
20
25
0



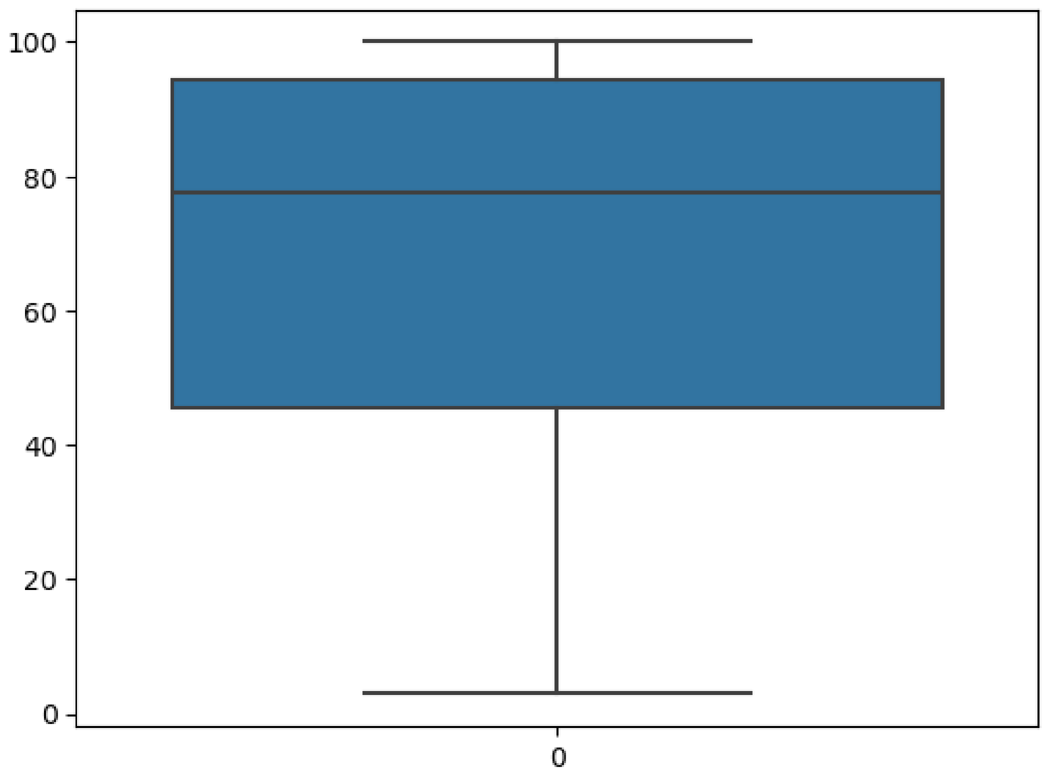
0.0
0.2
0.4
0.6
0.8
1.0
0

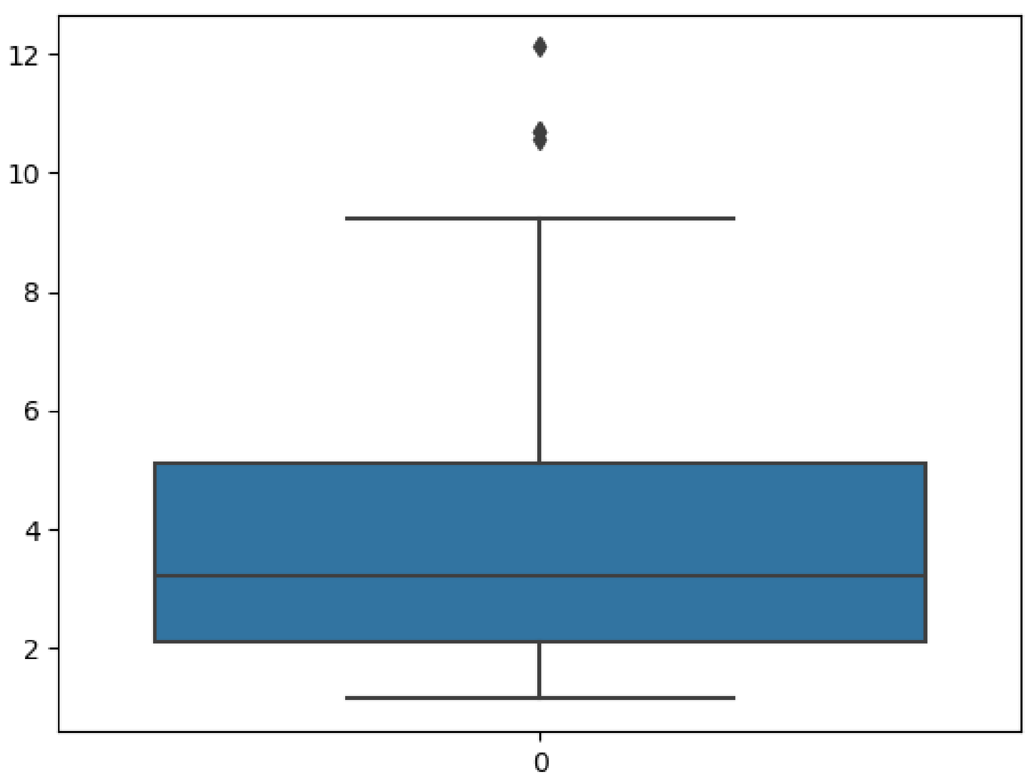


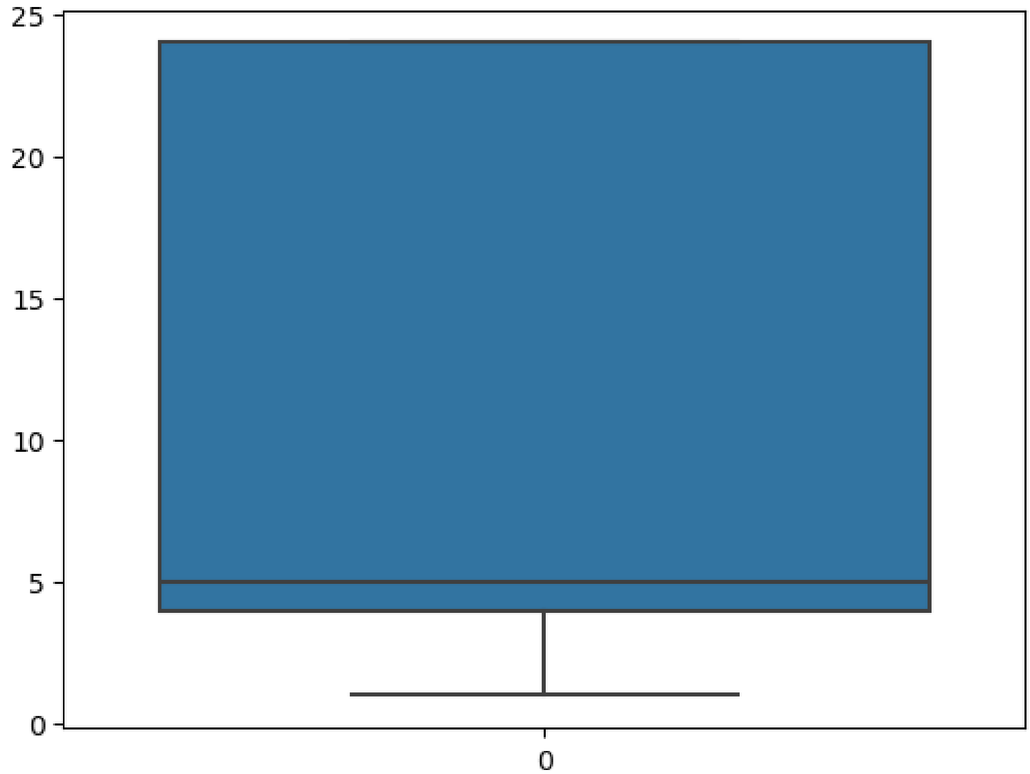
0.8
0.7
0.6
0.5
0.4
0

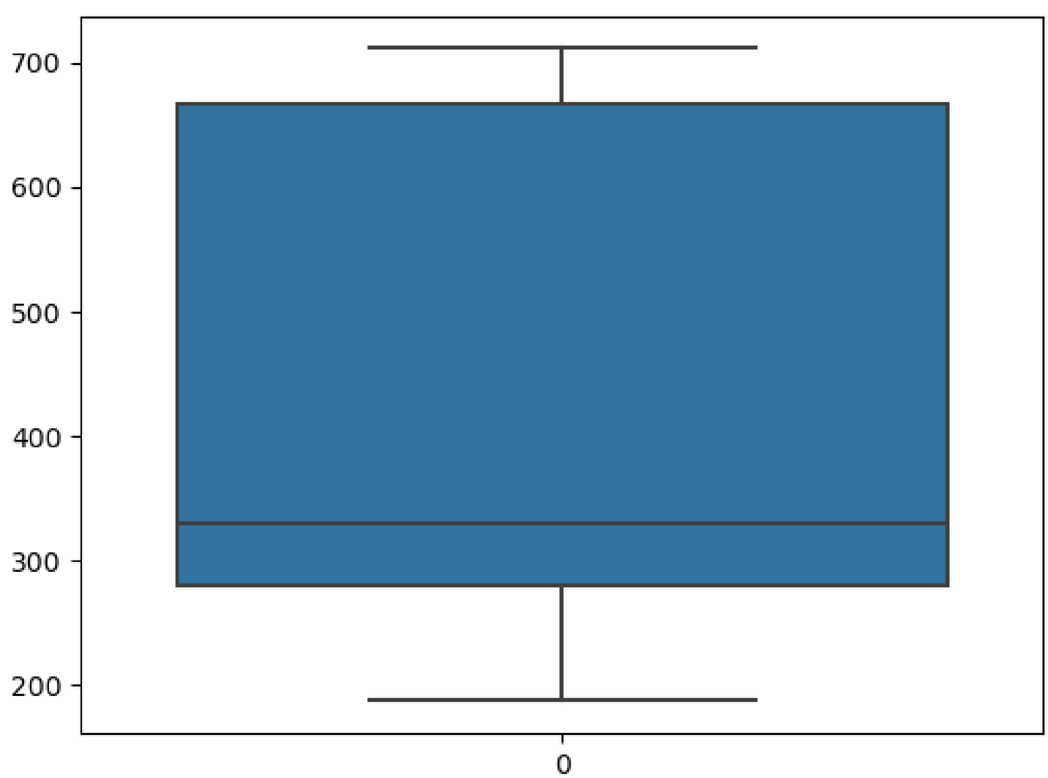


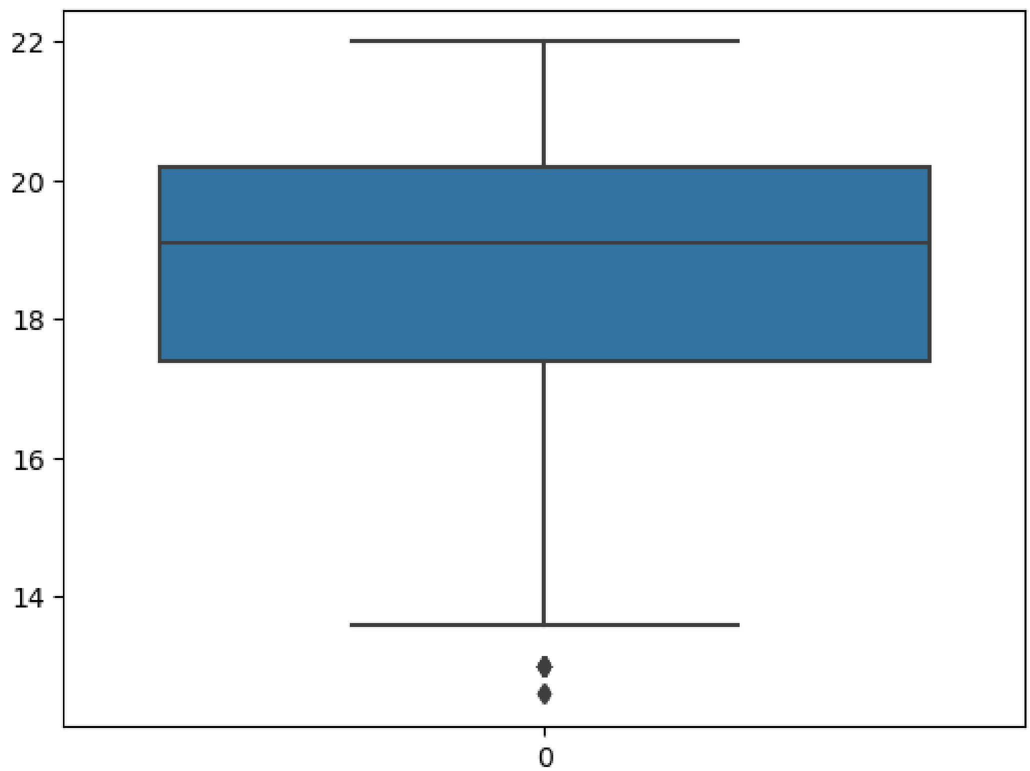
9
8
7
6
5
4
0

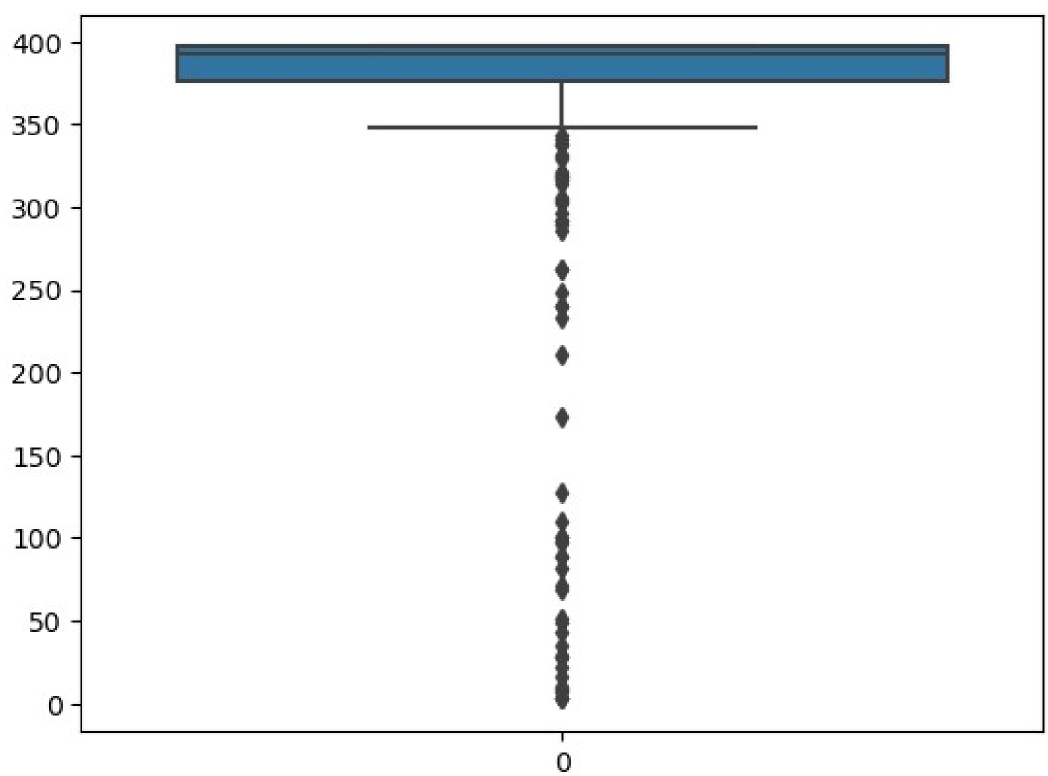

100
80
60
40
20
0
0

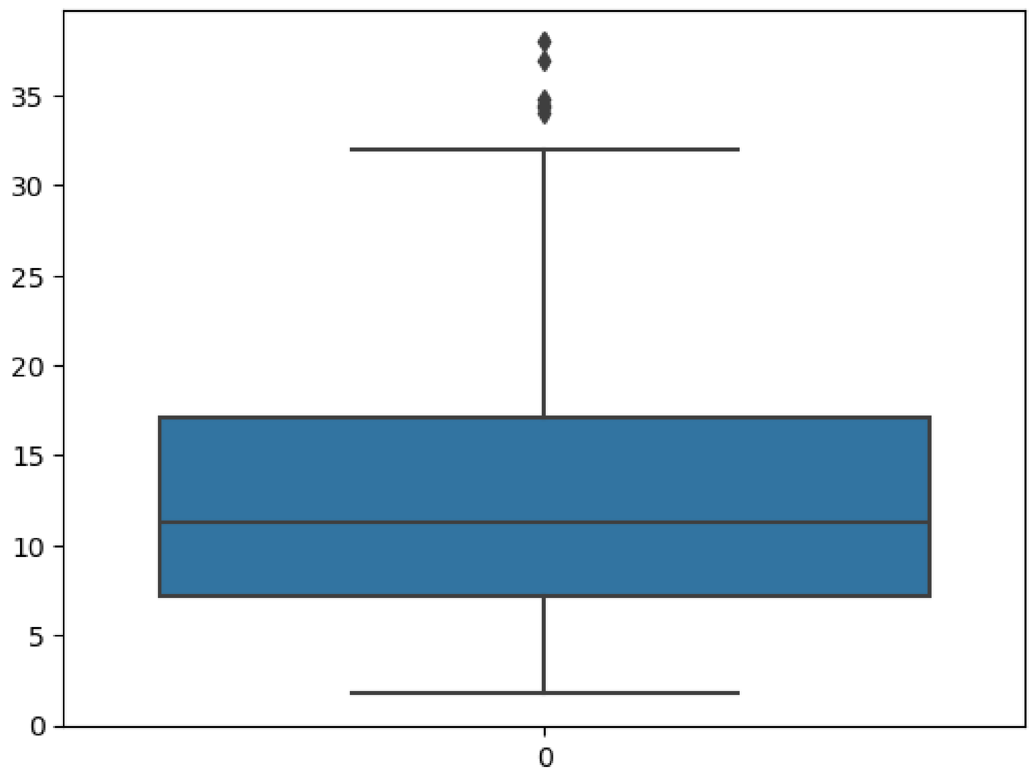

12
10
8
6
4
2
0


25
20
15
10
5
0
0

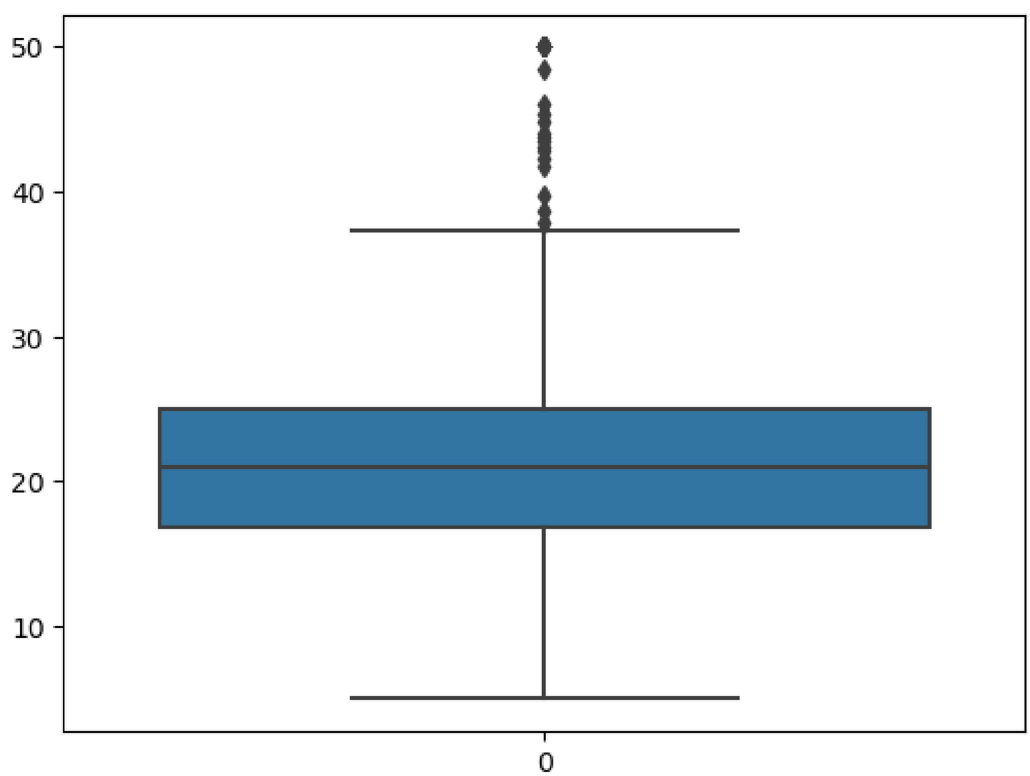

700
600
500
400
300
200
0


22
20
18
16
14
0


400
350
300
250
200
150
100
50
0
0



35
30
25
20
15
10
5
0
0



50
40
30
20
10
0

```python
# correlation
corr_matrix= df.corr().round(2)

corr_matrix[corr_matrix>(0.7)]
```

```
         CRIM   ZN  INDUS  CHAS   NOX    RM   AGE  DIS  RAD   TAX
PTRATIO  \
CRIM      1.0  NaN    NaN   NaN   NaN   NaN   NaN  NaN  NaN   NaN
NaN
ZN        NaN  1.0    NaN   NaN   NaN   NaN   NaN  NaN  NaN   NaN
NaN
INDUS     NaN  NaN   1.00   NaN  0.76   NaN   NaN  NaN  NaN  0.73
NaN
CHAS      NaN  NaN    NaN   1.0   NaN   NaN   NaN  NaN  NaN   NaN
NaN
NOX       NaN  NaN   0.76   NaN  1.00   NaN  0.73  NaN  NaN   NaN
NaN
RM        NaN  NaN    NaN   NaN   NaN  1.00   NaN  NaN  NaN   NaN
NaN
AGE       NaN  NaN    NaN   NaN  0.73   NaN  1.00  NaN  NaN   NaN
NaN
DIS       NaN  NaN    NaN   NaN   NaN   NaN   NaN  1.0  NaN   NaN
NaN
RAD       NaN  NaN    NaN   NaN   NaN   NaN   NaN  NaN  1.0  0.90
NaN
TAX       NaN  NaN   0.73   NaN   NaN   NaN   NaN  NaN  0.9  1.00
NaN
PTRATIO   NaN  NaN    NaN   NaN   NaN   NaN   NaN  NaN  NaN   NaN
1.0
B         NaN  NaN    NaN   NaN   NaN   NaN   NaN  NaN  NaN   NaN
NaN
LSTAT     NaN  NaN    NaN   NaN   NaN   NaN   NaN  NaN  NaN   NaN
NaN
MEDV      NaN  NaN    NaN   NaN   NaN  0.72   NaN  NaN  NaN   NaN
NaN

           B  LSTAT  MEDV
CRIM     NaN    NaN   NaN
ZN       NaN    NaN   NaN
INDUS    NaN    NaN   NaN
CHAS     NaN    NaN   NaN
NOX      NaN    NaN   NaN
RM       NaN    NaN  0.72
AGE      NaN    NaN   NaN
DIS      NaN    NaN   NaN
RAD      NaN    NaN   NaN
TAX      NaN    NaN   NaN
PTRATIO  NaN    NaN   NaN
B        1.0    NaN   NaN
```

```
LSTAT    NaN    1.0   NaN
MEDV     NaN    NaN  1.00
```

Outlier Treatment

```
# Outliers = crime,zn,rm,b

df["CRIM"].describe()

count    394.000000
mean       3.690136
std        9.202423
min        0.006320
25%        0.081955
50%        0.268880
75%        3.435973
max       88.976200
Name: CRIM, dtype: float64

df["CRIM"].describe()
Q1 = 0.08
Q3 = 3.67
IQR = Q3 - Q1
Upperlimit = Q3 + 1.5*IQR
lowerlimit = Q1 - 1.5 * IQR
df["CRIM"] =
np.where(df["CRIM"]>Upperlimit,Upperlimit,np.where(df["CRIM"]<lowerlim
it,lowerlimit,df["CRIM"]))

df["ZN"].describe()

count    394.000000
mean      11.460660
std       23.954082
min        0.000000
25%        0.000000
50%        0.000000
75%       12.500000
max      100.000000
Name: ZN, dtype: float64

df["ZN"].describe()
Q1 = 0.00
Q3 = 12.50
IQR = Q3 - Q1
Upperlimit = Q3 + 1.5*IQR
lowerlimit = Q1 - 1.5 * IQR
df["ZN"] =
```

```
np.where(df["ZN"]>Upperlimit,Upperlimit,np.where(df["ZN"]<lowerlimit,l
owerlimit,df["ZN"]))

df["RM"].describe()

count    394.000000
mean       6.280015
std        0.697985
min        3.561000
25%        5.879250
50%        6.201500
75%        6.605500
max        8.780000
Name: RM, dtype: float64

Q1 = 5.87
Q3 = 6.60
IQR = Q3 - Q1
Upperlimit = Q3 + 1.5*IQR
lowerlimit = Q1 - 1.5 * IQR
df["RM"] =
np.where(df["RM"]>Upperlimit,Upperlimit,np.where(df["RM"]<lowerlimit,l
owerlimit,df["RM"]))

df["B"].describe()

count    394.000000
mean     358.490939
std       89.283295
min        2.600000
25%      376.707500
50%      392.190000
75%      396.900000
max      396.900000
Name: B, dtype: float64

Q1 = 376.70
Q3 = 396.90
IQR = Q3 - Q1
Upperlimit = Q3 + 1.5*IQR
lowerlimit = Q1 - 1.5 * IQR
df["B"] =
np.where(df["B"]>Upperlimit,Upperlimit,np.where(df["B"]<lowerlimit,low
erlimit,df["B"]))
```

FEATURE SCALING

```
from sklearn.preprocessing import StandardScaler
```

```python
# splitting data into dependent and independent variable
x  = df.drop(["MEDV"], axis = 1)
y = df["MEDV"]

scaler = StandardScaler()
x_scaler = scaler.fit_transform(x)  # Scale the features
x_scaler = pd.DataFrame(x_scaler, columns=x.columns)

pd.DataFrame(x_scaler, columns = x.columns)
```

```
         CRIM        ZN     INDUS      CHAS       NOX        RM
AGE  \
0   -0.662816  0.948062 -1.259620 -0.271237 -0.134687  0.489673 -
0.134014
1   -0.656404 -0.573816 -0.569724 -0.271237 -0.745475  0.243887
0.357849
2   -0.656410 -0.573816 -0.569724 -0.271237 -0.745475  1.463243 -
0.281214
3   -0.654858 -0.573816 -1.278462 -0.271237 -0.842847  1.164788 -
0.830521
4   -0.655628 -0.573816 -1.278462 -0.271237 -0.842847  0.258251 -
0.367380
..        ...       ...       ...       ...       ...       ...
...
389 -0.610422 -0.573816 -0.189991 -0.271237  0.281356 -1.115919
0.163976
390 -0.596202 -0.573816 -0.189991 -0.271237  0.281356 -0.384944
0.386570
391 -0.650917 -0.573816  0.134666 -0.271237  0.175132 -0.236514
0.278863
392 -0.646185 -0.573816  0.134666 -0.271237  0.175132  1.129676
0.792268
393 -0.631268 -0.573816  0.134666 -0.271237  0.175132  0.839201
0.731233

          DIS       RAD       TAX   PTRATIO         B     LSTAT
0    0.135851 -0.974609 -0.656944 -1.496303  0.769546 -1.067126
1    0.554334 -0.858633 -0.978184 -0.340879  0.769546 -0.497196
2    0.554334 -0.858633 -0.978184 -0.340879  0.546744 -1.197278
3    1.076829 -0.742657 -1.097162  0.075073  0.645281 -1.346610
4    1.076829 -0.742657 -1.097162  0.075073  0.617362 -1.035615
..        ...       ...       ...       ...       ...       ...
389 -0.670530 -0.394730 -0.091800  0.306158  0.707687  0.319337
390 -0.623629 -0.394730 -0.091800  0.306158  0.769546  0.213845
391 -0.724158 -0.974609 -0.793769  1.138063  0.769546 -0.505417
392 -0.781413 -0.974609 -0.793769  1.138063  0.769546 -0.976704
393 -0.675778 -0.974609 -0.793769  1.138063  0.580685 -0.861622

[394 rows x 13 columns]
```

```
x_scaler.describe().round(2)
```

```
         CRIM      ZN   INDUS    CHAS     NOX      RM     AGE     DIS
RAD  \
count  394.00  394.00  394.00  394.00  394.00  394.00  394.00  394.00
394.00
mean    -0.00   -0.00   -0.00   -0.00    0.00   -0.00   -0.00    0.00
0.00
std      1.00    1.00    1.00    1.00    1.00    1.00    1.00    1.00
1.00
min     -0.66   -0.57   -1.53   -0.27   -1.45   -2.38   -2.37   -1.28
-0.97
25%     -0.64   -0.57   -0.85   -0.27   -0.89   -0.62   -0.84   -0.81
-0.63
50%     -0.58   -0.57   -0.35   -0.27   -0.13   -0.11    0.31   -0.29
-0.51
75%      0.38    0.48    1.03   -0.27    0.63    0.54    0.91    0.63
1.69
max      2.10    2.07    2.43    3.69    2.81    2.28    1.12    3.97
1.69

          TAX  PTRATIO       B   LSTAT
count  394.00   394.00  394.00  394.00
mean     0.00     0.00    0.00   -0.00
std      1.00     1.00    1.00    1.00
min     -1.31    -2.74   -1.99   -1.51
25%     -0.75    -0.53   -0.34   -0.77
50%     -0.45     0.26    0.51   -0.20
75%      1.54     0.77    0.77    0.60
max      1.81     1.60    0.77    3.45
```

```
# VIF- Variance inflation factor > 5 - multicollinearity

x_scaler.shape

(394, 13)

from statsmodels.stats.outliers_influence import
variance_inflation_factor

variable = x_scaler

vif = pd.DataFrame()

vif["variance inflation factor"] =
[variance_inflation_factor(variable,i) for i in
range(variable.shape[1])]

vif["Features"] = x.columns

vif
```

```
    variance inflation factor Features
0                    9.673299     CRIM
1                    2.444219       ZN
2                    4.078226    INDUS
3                    1.070390     CHAS
4                    4.513045      NOX
5                    2.221245       RM
6                    3.157609      AGE
7                    3.875598      DIS
8                   12.484269      RAD
9                    8.283109      TAX
10                   1.906799  PTRATIO
11                   1.330136        B
12                   3.498310    LSTAT
```

```
# rad has the highest correlation factor
x_scaler = x_scaler.drop("RAD", axis = 1)

x_scaler.columns
```

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

```
vif = pd.DataFrame()
variable = x_scaler
vif["variance inflation factor"] =
[variance_inflation_factor(variable,i) for i in
range(variable.shape[1])]
vif["Features"] = x.columns
```

```
----------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
Cell In[43], line 4
      2 variable = x_scaler
      3 vif["variance inflation factor"] =
[variance_inflation_factor(variable,i) for i in
range(variable.shape[1])]
----> 4 vif["Features"] = x.columns

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\
frame.py:4091, in DataFrame.__setitem__(self, key, value)
   4088     self._setitem_array([key], value)
   4089 else:
   4090     # set column
-> 4091     self._set_item(key, value)

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\
```

```
frame.py:4300, in DataFrame._set_item(self, key, value)
   4290 def _set_item(self, key, value) -> None:
   4291     """
   4292     Add series to DataFrame in specified column.
   4293 
   (...)
   4298     ensure homogeneity.
   4299     """
-> 4300     value, refs = self._sanitize_column(value)
   4302     if (
   4303         key in self.columns
   4304         and value.ndim == 1
   4305         and not isinstance(value.dtype, ExtensionDtype)
   4306     ):
   4307         # broadcast across multiple columns if necessary
   4308         if not self.columns.is_unique or
isinstance(self.columns, MultiIndex):

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\
frame.py:5039, in DataFrame._sanitize_column(self, value)
   5036     return _reindex_for_setitem(value, self.index)
   5038 if is_list_like(value):
-> 5039     com.require_length_match(value, self.index)
   5040 return sanitize_array(value, self.index, copy=True,
allow_2d=True), None

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\
common.py:561, in require_length_match(data, index)
    557 """
    558 Check the length of data matches the length of the index.
    559 """
    560 if len(data) != len(index):
--> 561     raise ValueError(
    562         "Length of values "
    563         f"({len(data)}) "
    564         "does not match length of index "
    565         f"({len(index)})"
    566     )

ValueError: Length of values (13) does not match length of index (12)

vif["Featurs"] = x_scaler.columns

vif
```

EDA (EXPLORATORY DATA ANALYSIS)

```
import  dtale
dtale.show(df)

<IPython.lib.display.IFrame at 0x14d77dd5390>
```

Splitting the data into Training and Testing model

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size =
0.2,random_state= 43)
```

Building linear regression model

```
from statsmodels.regression.linear_model import OLS
import statsmodels.regression.linear_model as smf

regression = smf.OLS(endog = y_train, exog= x_train).fit()
regression.summary()

<class 'statsmodels.iolib.summary.Summary'>
"""
                                 OLS Regression Results

=======================================================================
=================
Dep. Variable:                   MEDV   R-squared (uncentered):
0.961
Model:                            OLS   Adj. R-squared (uncentered):
0.960
Method:                 Least Squares   F-statistic:
577.1
Date:                Wed, 18 Dec 2024   Prob (F-statistic):
1.64e-204
Time:                        14:30:38   Log-Likelihood:
-938.49
No. Observations:                 315   AIC:
1903.
Df Residuals:                     302   BIC:
1952.
Df Model:                          13
```

```
Covariance Type:            nonrobust

==============================================================================
========
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
--------
CRIM          -0.3428      0.254     -1.348      0.179      -0.843
0.158
ZN             0.0505      0.036      1.404      0.161      -0.020
0.121
INDUS          0.0033      0.080      0.041      0.967      -0.155
0.161
CHAS           4.0215      1.119      3.595      0.000       1.820
6.223
NOX           -6.8214      4.764     -1.432      0.153     -16.196
2.553
RM             5.5628      0.546     10.186      0.000       4.488
6.638
AGE           -0.0262      0.017     -1.510      0.132      -0.060
0.008
DIS           -0.9999      0.233     -4.289      0.000      -1.459
-0.541
RAD            0.2131      0.109      1.958      0.051      -0.001
0.427
TAX           -0.0084      0.005     -1.815      0.071      -0.017
0.001
PTRATIO       -0.7687      0.165     -4.661      0.000      -1.093
-0.444
B              0.0460      0.012      3.689      0.000       0.021
0.071
LSTAT         -0.3949      0.066     -6.018      0.000      -0.524
-0.266
==============================================================================
========
Omnibus:                      157.385   Durbin-Watson:
2.094
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
1113.041
Skew:                           1.946   Prob(JB):
2.02e-242
Kurtosis:                      11.346   Cond. No.
1.00e+04
==============================================================================
========

Notes:
[1] R² is computed without centering (uncentered) since the model does
```

```
not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[3] The condition number is large,  1e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

Approach- 2 Linear regression

```
from sklearn.linear_model import LinearRegression
reg_model = LinearRegression()
reg_model.fit(x_train , y_train)
```

```
LinearRegression()
```

prediction

```
y_pred =reg_model.predict(x_test)
```

Evaluation metrics

```
from sklearn.metrics import r2_score

print("Accuracy:", r2_score(y_test, y_pred))
```

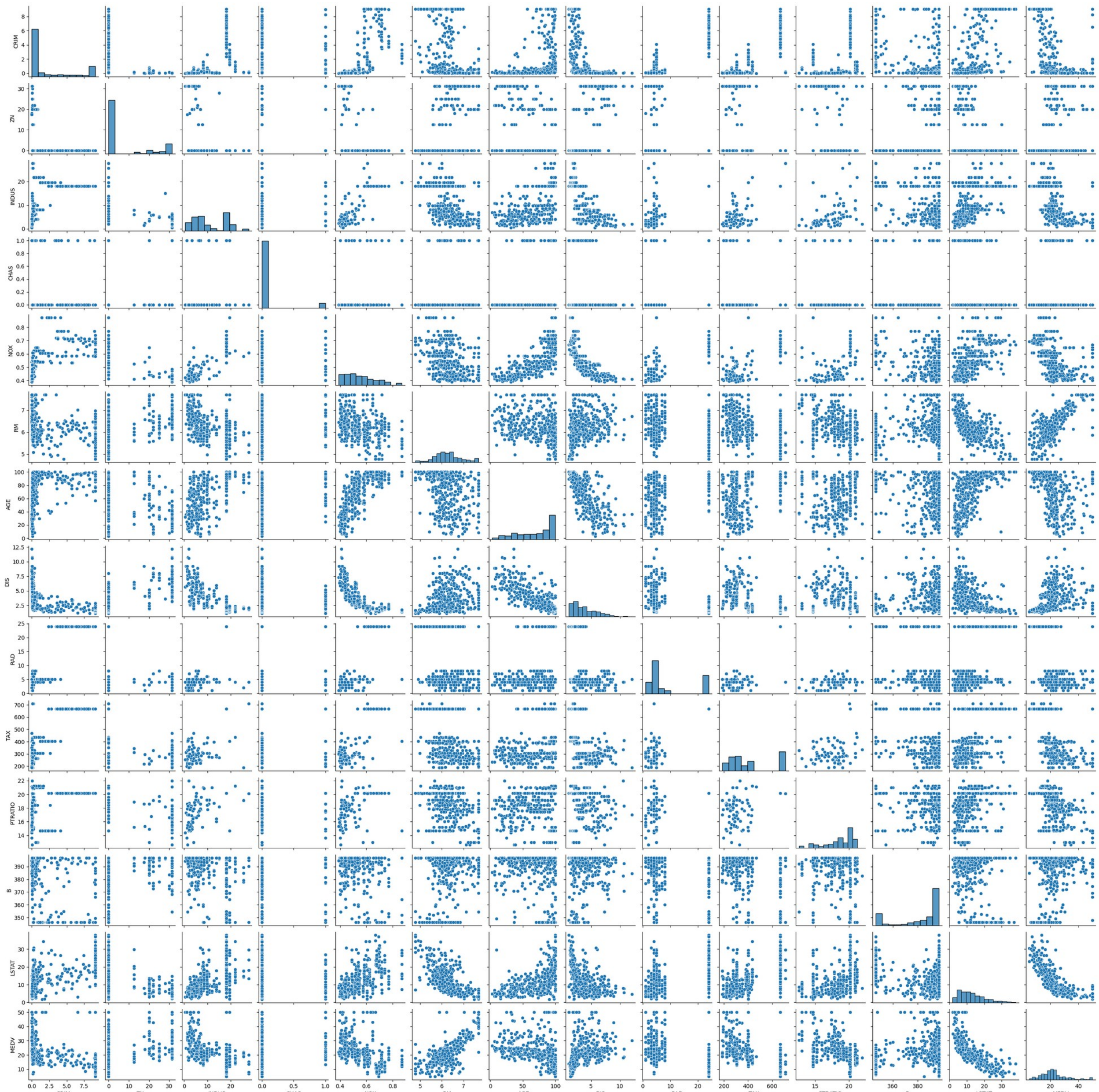
```
Accuracy: 0.7974958808527665
```

```
sns.pairplot(df,height = 2)
plt.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.

C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.

C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
```

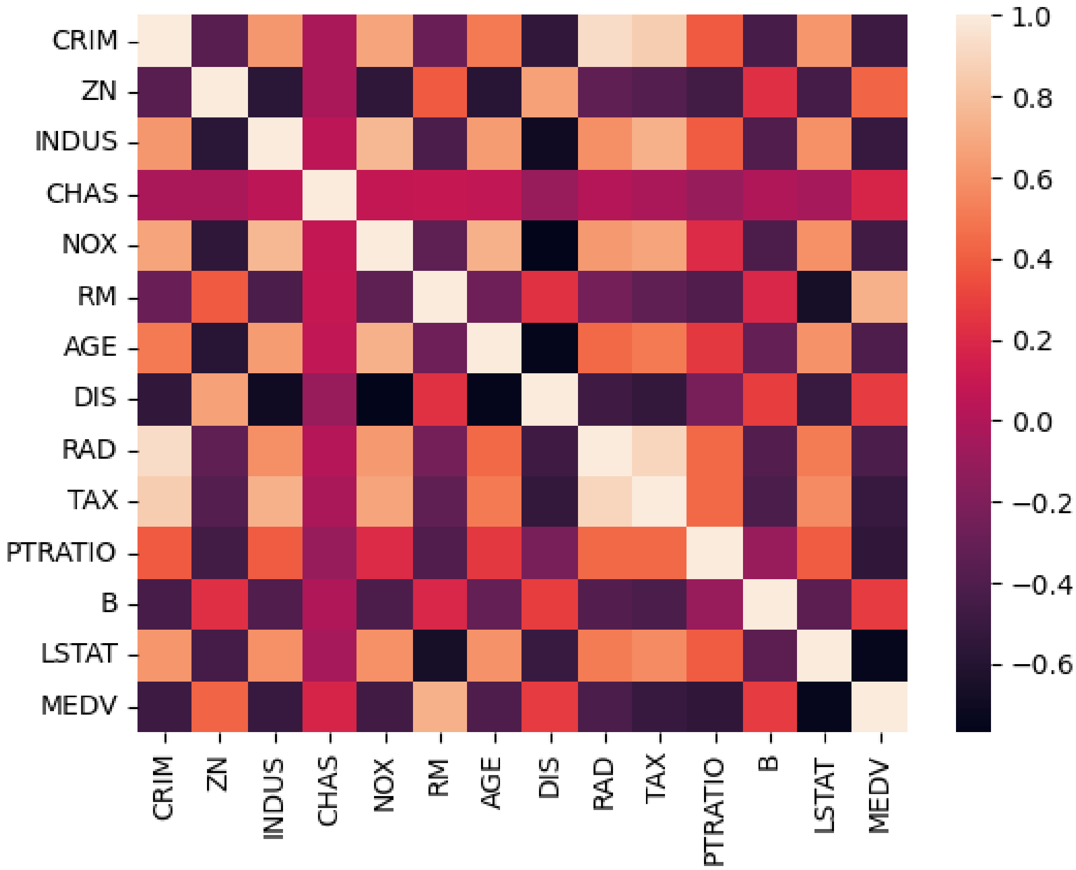
```
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.

C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.

C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.

C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.

C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.

C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.

C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.

C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.

C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future
```

```
version. Convert inf values to NaN before operating instead.

C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.

C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.

C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
```




```
sns.heatmap(df.corr())
```

```
<Axes: >
```




```
             CRIM        ZN     INDUS      CHAS       NOX        RM
AGE  \
CRIM     1.000000 -0.361225  0.623422 -0.020456  0.676892 -0.290585
0.509373
ZN      -0.361225  1.000000 -0.573423 -0.025051 -0.547805  0.389992 -
0.577199
INDUS    0.623422 -0.573423  1.000000  0.049820  0.762737 -0.420265
0.642387
CHAS    -0.020456 -0.025051  0.049820  1.000000  0.076661  0.088389
0.072644
NOX      0.676892 -0.547805  0.762737  0.076661  1.000000 -0.335521
0.732540
RM      -0.290585  0.389992 -0.420265  0.088389 -0.335521  1.000000 -
0.265917
AGE      0.509373 -0.577199  0.642387  0.072644  0.732540 -0.265917
1.000000
DIS     -0.538438  0.663585 -0.696569 -0.095037 -0.768137  0.236080 -
0.753547
RAD      0.925917 -0.331259  0.591944  0.014102  0.628170 -0.240004
0.443585
TAX      0.860415 -0.378232  0.734204 -0.026513  0.679824 -0.327562
```

```
0.504472
PTRATIO  0.389845 -0.457177  0.395691 -0.104995  0.210216 -0.397525
0.264968
B       -0.434958  0.227525 -0.399533 -0.007114 -0.410601  0.187988 -
0.307924
LSTAT    0.616377 -0.449960  0.598156 -0.037113  0.593655 -0.658038
0.601137
MEDV    -0.479865  0.424049 -0.510829  0.173701 -0.459054  0.731312 -
0.407470

              DIS       RAD       TAX   PTRATIO         B     LSTAT
MEDV
CRIM    -0.538438  0.925917  0.860415  0.389845 -0.434958  0.616377 -
0.479865
ZN       0.663585 -0.331259 -0.378232 -0.457177  0.227525 -0.449960
0.424049
INDUS   -0.696569  0.591944  0.734204  0.395691 -0.399533  0.598156 -
0.510829
CHAS    -0.095037  0.014102 -0.026513 -0.104995 -0.007114 -0.037113
0.173701
NOX     -0.768137  0.628170  0.679824  0.210216 -0.410601  0.593655 -
0.459054
RM       0.236080 -0.240004 -0.327562 -0.397525  0.187988 -0.658038
0.731312
AGE     -0.753547  0.443585  0.504472  0.264968 -0.307924  0.601137 -
0.407470
DIS      1.000000 -0.477075 -0.529603 -0.228840  0.287118 -0.505036
0.279547
RAD     -0.477075  1.000000  0.900000  0.441949 -0.384599  0.510868 -
0.416638
TAX     -0.529603  0.900000  1.000000  0.446961 -0.420094  0.572218 -
0.508864
PTRATIO -0.228840  0.441949  0.446961  1.000000 -0.093122  0.395006 -
0.543809
B        0.287118 -0.384599 -0.420094 -0.093122  1.000000 -0.340145
0.280402
LSTAT   -0.505036  0.510868  0.572218  0.395006 -0.340145  1.000000 -
0.743450
MEDV     0.279547 -0.416638 -0.508864 -0.543809  0.280402 -0.743450
1.000000
```