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

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

df.head()
```

```
      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
```

```python
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
```

```
df.shape
```

```
(506, 14)
```

```
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
```

```
df.nunique()
```

```
CRIM       484
ZN          26
INDUS       76
CHAS         2
NOX         81
RM         446
AGE        348
DIS        412
RAD          9
TAX         66
PTRATIO     46
B          357
LSTAT      438
MEDV       229
dtype: int64
```

```
df['ZN'].unique()
```

```
array([ 18. ,   0. ,  12.5,  75. ,  21. ,  90. ,  85. , 100. ,  25. ,
        17.5,  80. ,   nan,  28. ,  45. ,  60. ,  95. ,  82.5,  30. ,
        22. ,  20. ,  40. ,  55. ,  52.5,  70. ,  34. ,  33. ,  35. ])
```

```
df['ZN'].value_counts()
```

```
ZN
0.0      360
20.0      20
80.0      14
22.0      10
12.5      10
25.0      10
45.0       6
40.0       6
30.0       5
90.0       5
95.0       4
60.0       4
33.0       4
21.0       4
55.0       3
70.0       3
75.0       3
52.5       3
35.0       2
82.5       2
28.0       2
85.0       2
17.5       1
100.0      1
34.0       1
18.0       1
Name: count, dtype: int64
```

```
# Impute values of categorical features
```

```
df['CHAS'].mode()
```

```
0    0.0
Name: CHAS, dtype: float64
```

```
df['CHAS'].fillna(df['CHAS'].mode()[0],inplace=True)
```

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

```
CRIM       20
ZN         20
INDUS      20
CHAS        0
NOX         0
RM          0
AGE        20
DIS         0
RAD         0
TAX         0
PTRATIO     0
```

```
B            0
LSTAT       20
MEDV         0
dtype: int64
```

```
df['CRIM'].skew()
```

```
5.2128426499800975
```

```
df['ZN'].skew()
```

```
2.2566126051408197
```

```
df['INDUS'].skew()
```

```
0.30372218758107833
```

```
df['LSTAT'].skew()
```

```
0.908891836957813
```

```
df['AGE'].skew()
```

```
-0.5824700575056604
```

```
df['CRIM'].fillna(df['CRIM'].median(),inplace=True)
```

```
df['ZN'].fillna(df['ZN'].median(),inplace=True)
```

```
df['INDUS'].fillna(df['INDUS'].mean(),inplace=True)
```

```
df['AGE'].fillna(df['AGE'].mean(),inplace=True)
```

```
df['LSTAT'].fillna(df['LSTAT'].median(),inplace=True)
```

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

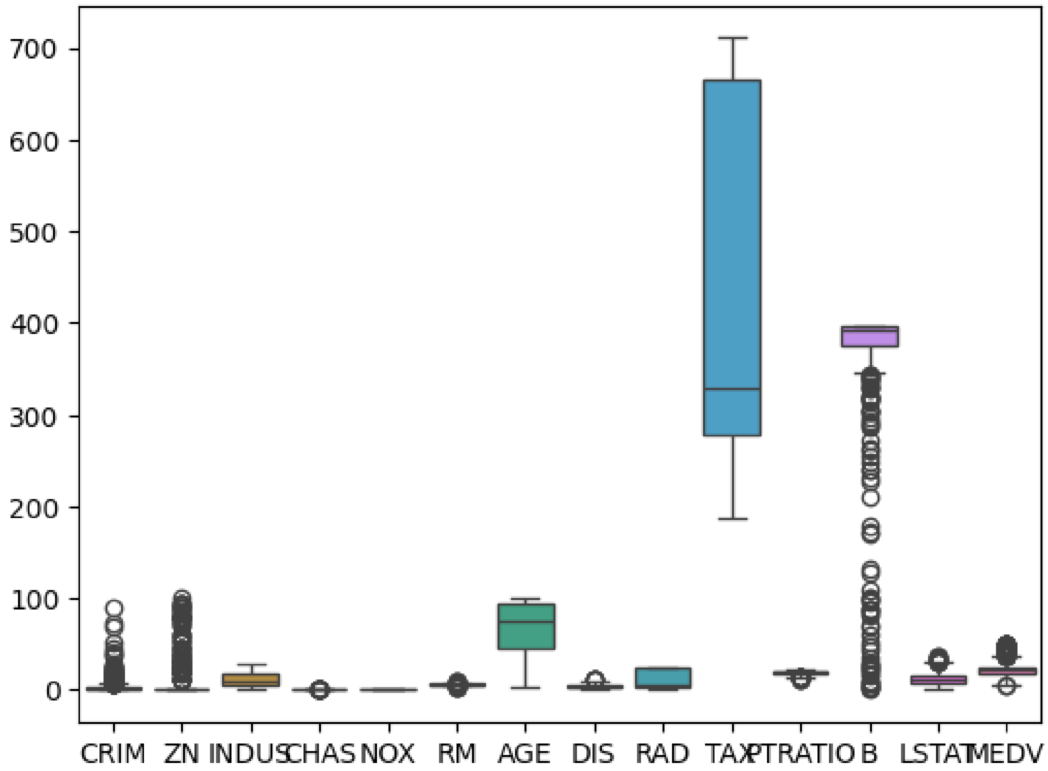
```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
```

```
MEDV       0
dtype: int64
```

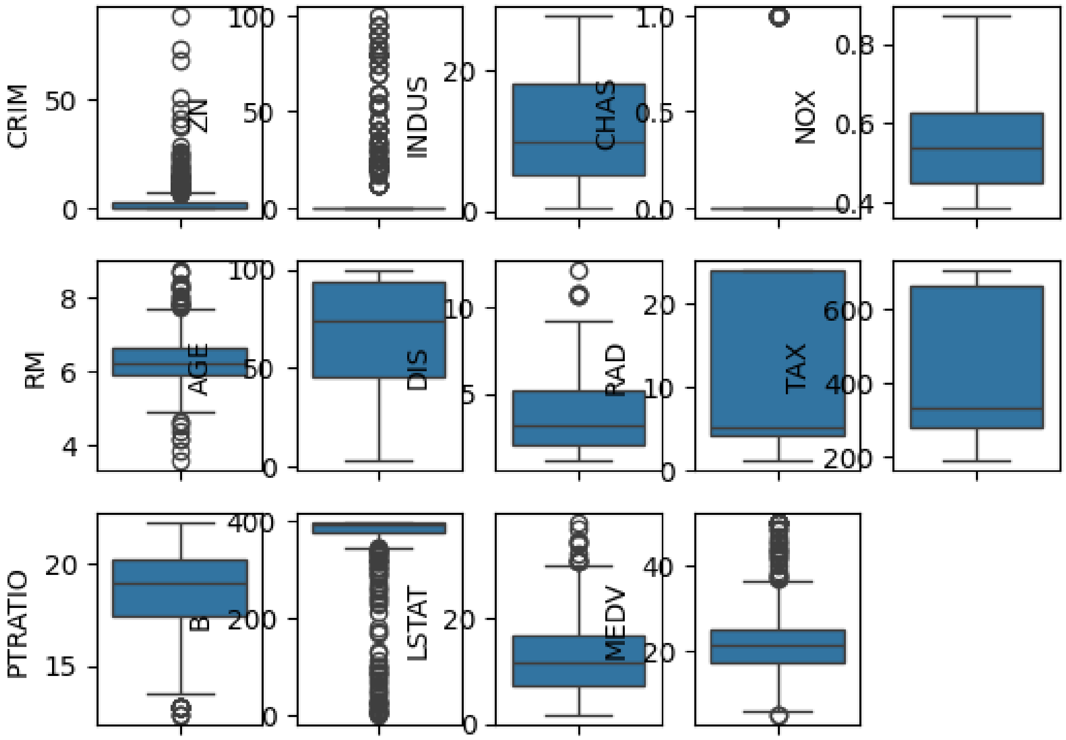
```
df.describe()
```

```
                CRIM          ZN       INDUS        CHAS         NOX
RM  \
count  506.000000  506.000000  506.000000  506.000000  506.000000
506.000000
mean     3.479140   10.768775   11.083992    0.067194    0.554695
6.284634
std      8.570832   23.025124    6.699165    0.250605    0.115878
0.702617
min      0.006320    0.000000    0.460000    0.000000    0.385000
3.561000
25%      0.083235    0.000000    5.190000    0.000000    0.449000
5.885500
50%      0.253715    0.000000    9.900000    0.000000    0.538000
6.208500
75%      2.808720    0.000000   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  506.000000  506.000000  506.000000  506.000000  506.000000
506.000000
mean    68.518519    3.795043    9.549407  408.237154   18.455534
356.674032
std     27.439466    2.105710    8.707259  168.537116    2.164946
91.294864
min      2.900000    1.129600    1.000000  187.000000   12.600000
0.320000
25%     45.925000    2.100175    4.000000  279.000000   17.400000
375.377500
50%     74.450000    3.207450    5.000000  330.000000   19.050000
391.440000
75%     93.575000    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  506.000000  506.000000
mean    12.664625   22.532806
std      7.017219    9.197104
min      1.730000    5.000000
25%      7.230000   17.025000
50%     11.430000   21.200000
```

```
75%      16.570000   25.000000
max      37.970000   50.000000

sn.boxplot(df)

<Axes: >
```



```
# hndling missing values

# chek distribution of quentitative columns

i=1
for column in df:
    plt.subplot(3,5,i)
    # subplot( i index, 3 =row, 5 column)
    sn.boxplot(df[column])
    i=i+1
plt.show()
```



```
df.corr()

                 CRIM        ZN     INDUS      CHAS       NOX        RM
AGE  \
CRIM     1.000000 -0.185359  0.392063 -0.055585  0.410971 -0.220045
0.346395
ZN      -0.185359  1.000000 -0.507800 -0.032992 -0.498619  0.312295 -
0.534831
INDUS    0.392063 -0.507800  1.000000  0.054172  0.740965 -0.381457
0.614592
CHAS    -0.055585 -0.032992  0.054172  1.000000  0.070867  0.106797
0.073549
NOX      0.410971 -0.498619  0.740965  0.070867  1.000000 -0.302188
0.711461
RM      -0.220045  0.312295 -0.381457  0.106797 -0.302188  1.000000 -
0.241351
AGE      0.346395 -0.534831  0.614592  0.073549  0.711461 -0.241351
1.000000
DIS     -0.366025  0.632428 -0.699639 -0.092318 -0.769230  0.205246 -
0.724353
RAD      0.601224 -0.300061  0.593176 -0.003339  0.611441 -0.209847
0.449989
TAX      0.560469 -0.304385  0.716062 -0.035822  0.668023 -0.292048
0.500589
PTRATIO  0.277964 -0.394622  0.384806 -0.109451  0.188933 -0.355501
0.262723
```

```
B       -0.365336  0.170125 -0.354597  0.050608 -0.380051  0.128069 -
0.265282
LSTAT    0.437417 -0.398838  0.567859 -0.047279  0.573040 -0.604323
0.576605
MEDV    -0.383895  0.362292 -0.478657  0.183844 -0.427321  0.695360 -
0.380223

              DIS       RAD       TAX   PTRATIO         B     LSTAT
MEDV
CRIM    -0.366025  0.601224  0.560469  0.277964 -0.365336  0.437417 -
0.383895
ZN       0.632428 -0.300061 -0.304385 -0.394622  0.170125 -0.398838
0.362292
INDUS   -0.699639  0.593176  0.716062  0.384806 -0.354597  0.567859 -
0.478657
CHAS    -0.092318 -0.003339 -0.035822 -0.109451  0.050608 -0.047279
0.183844
NOX     -0.769230  0.611441  0.668023  0.188933 -0.380051  0.573040 -
0.427321
RM       0.205246 -0.209847 -0.292048 -0.355501  0.128069 -0.604323
0.695360
AGE     -0.724353  0.449989  0.500589  0.262723 -0.265282  0.576605 -
0.380223
DIS      1.000000 -0.494588 -0.534432 -0.232471  0.291512 -0.483244
0.249929
RAD     -0.494588  1.000000  0.910228  0.464741 -0.444413  0.467765 -
0.381626
TAX     -0.534432  0.910228  1.000000  0.460853 -0.441808  0.524156 -
0.468536
PTRATIO -0.232471  0.464741  0.460853  1.000000 -0.177383  0.370727 -
0.507787
B        0.291512 -0.444413 -0.441808 -0.177383  1.000000 -0.370993
0.333461
LSTAT   -0.483244  0.467765  0.524156  0.370727 -0.370993  1.000000 -
0.723093
MEDV     0.249929 -0.381626 -0.468536 -0.507787  0.333461 -0.723093
1.000000
```

```
df.corr()['MEDV']
```

```
CRIM      -0.383895
ZN         0.362292
INDUS     -0.478657
CHAS       0.183844
NOX       -0.427321
RM         0.695360
AGE       -0.380223
DIS        0.249929
RAD       -0.381626
TAX       -0.468536
```

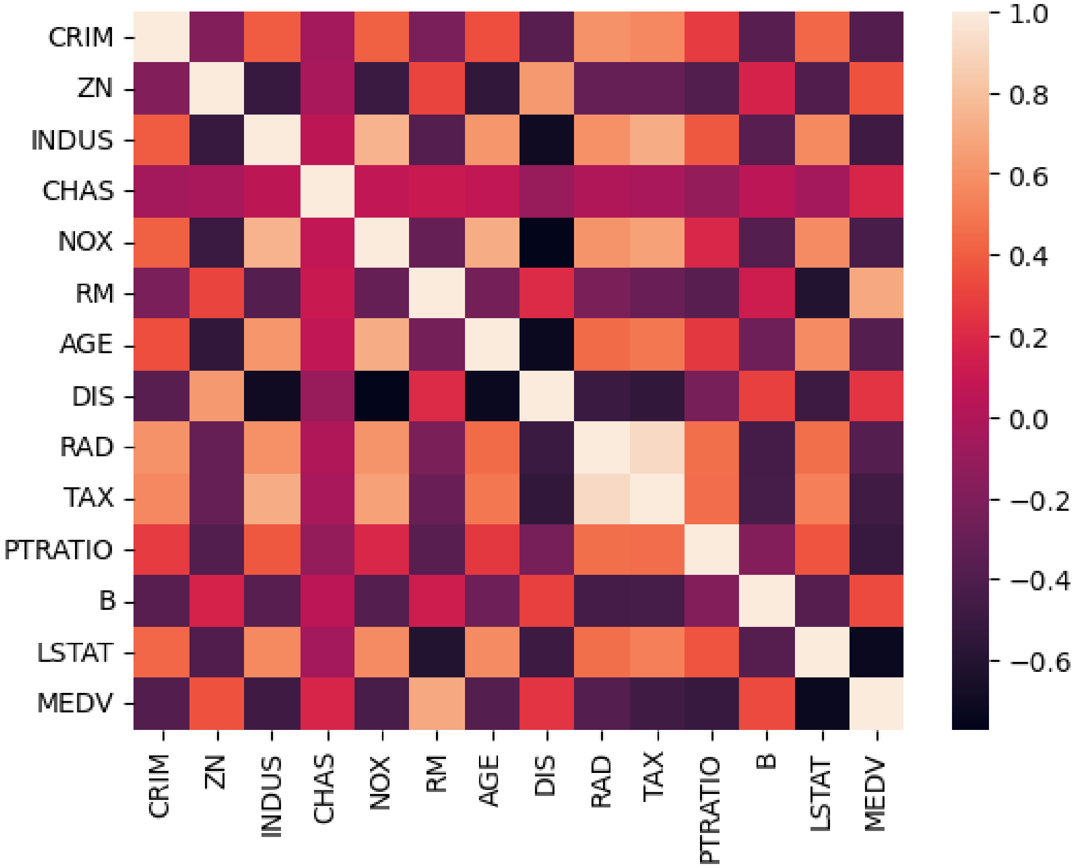
```
PTRATIO   -0.507787
B          0.333461
LSTAT     -0.723093
MEDV       1.000000
Name: MEDV, dtype: float64
```

```
df.corr()['MEDV'].sort_values()
```

```
LSTAT     -0.723093
PTRATIO   -0.507787
INDUS     -0.478657
TAX       -0.468536
NOX       -0.427321
CRIM      -0.383895
RAD       -0.381626
AGE       -0.380223
CHAS       0.183844
DIS        0.249929
B          0.333461
ZN         0.362292
RM         0.695360
MEDV       1.000000
Name: MEDV, dtype: float64
```

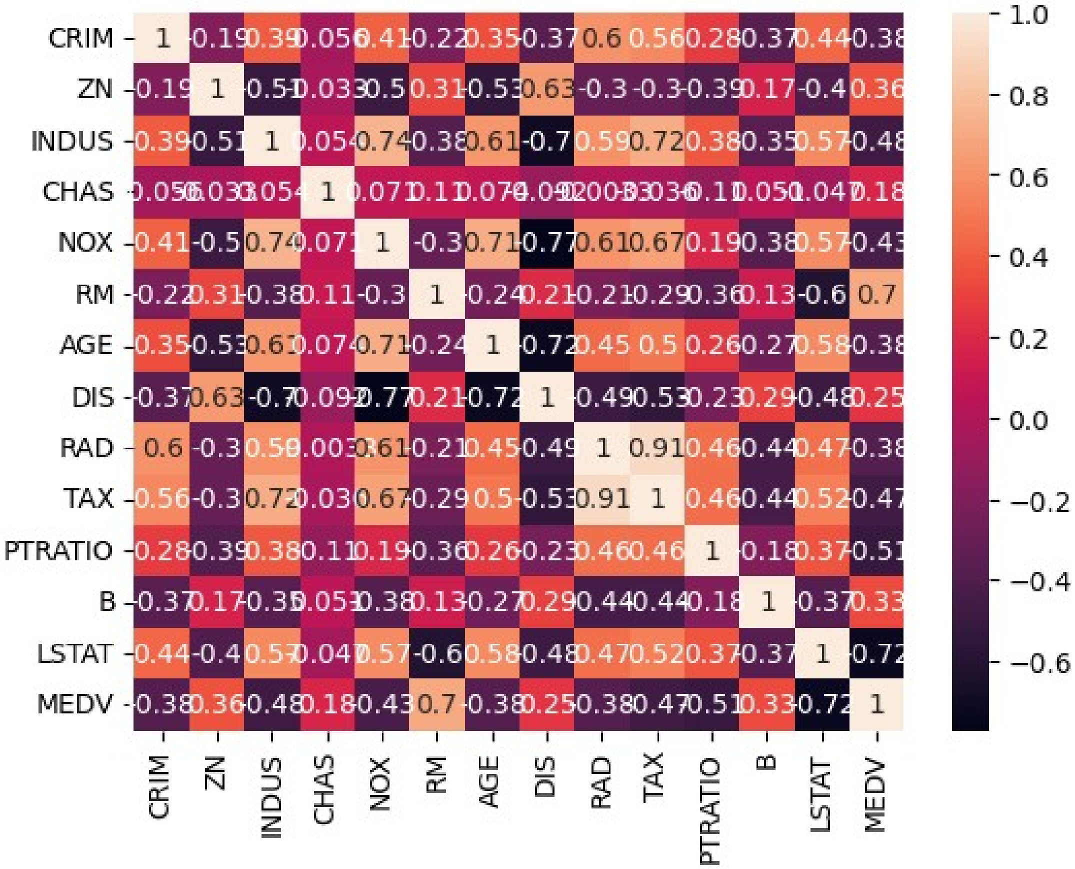
```
sn.heatmap(df.corr())
```

```
<Axes: >
```



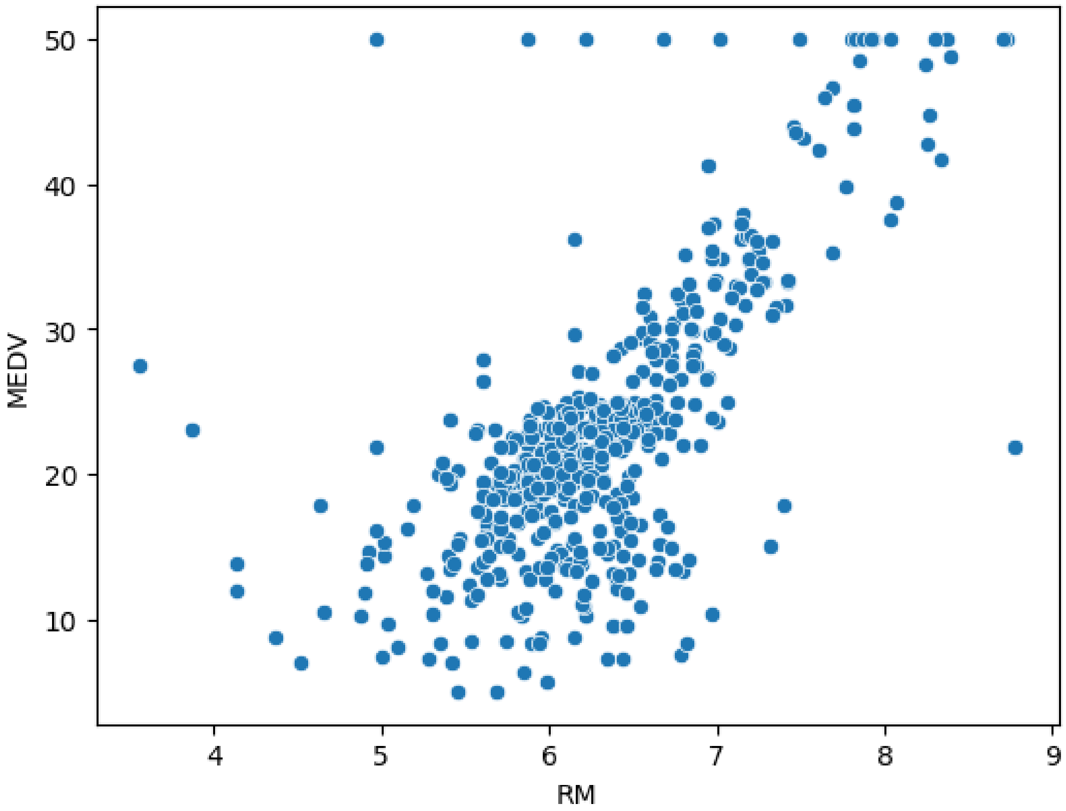
```
sn.heatmap(df.corr(),annot=True)
```

```
<Axes: >
```



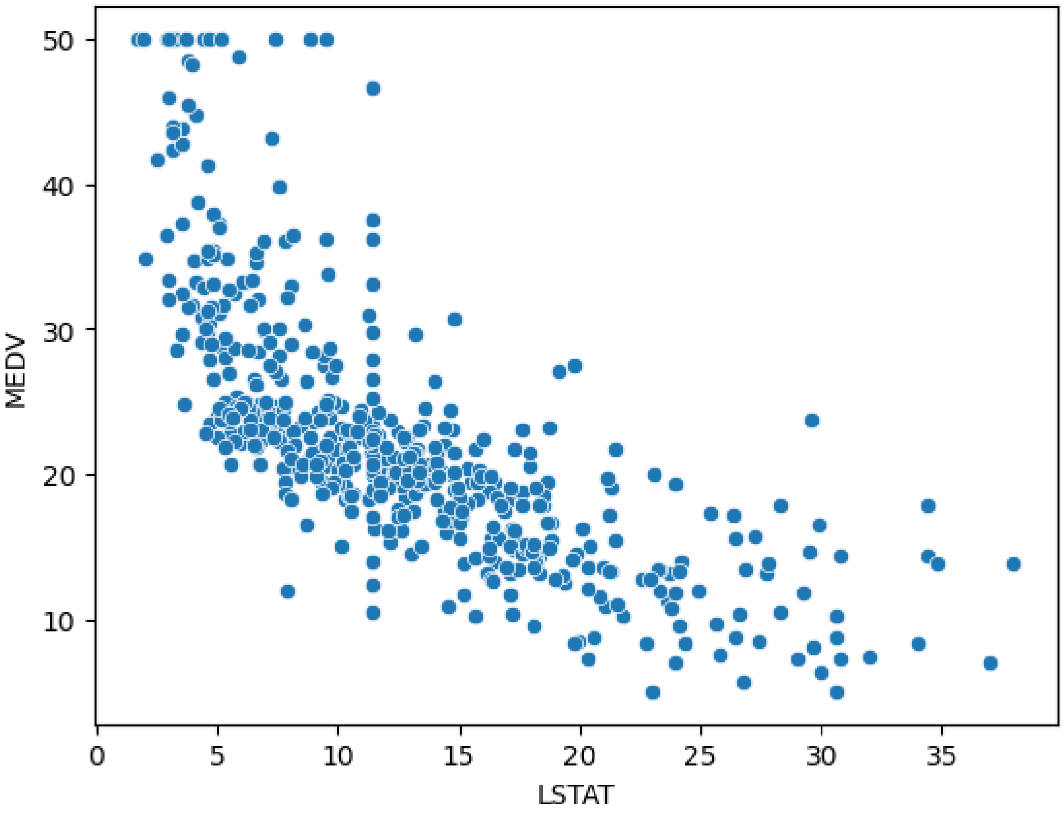
```
sn.scatterplot(x=df['RM'],y=df['MEDV'])
```

```
<Axes: xlabel='RM', ylabel='MEDV'>
```



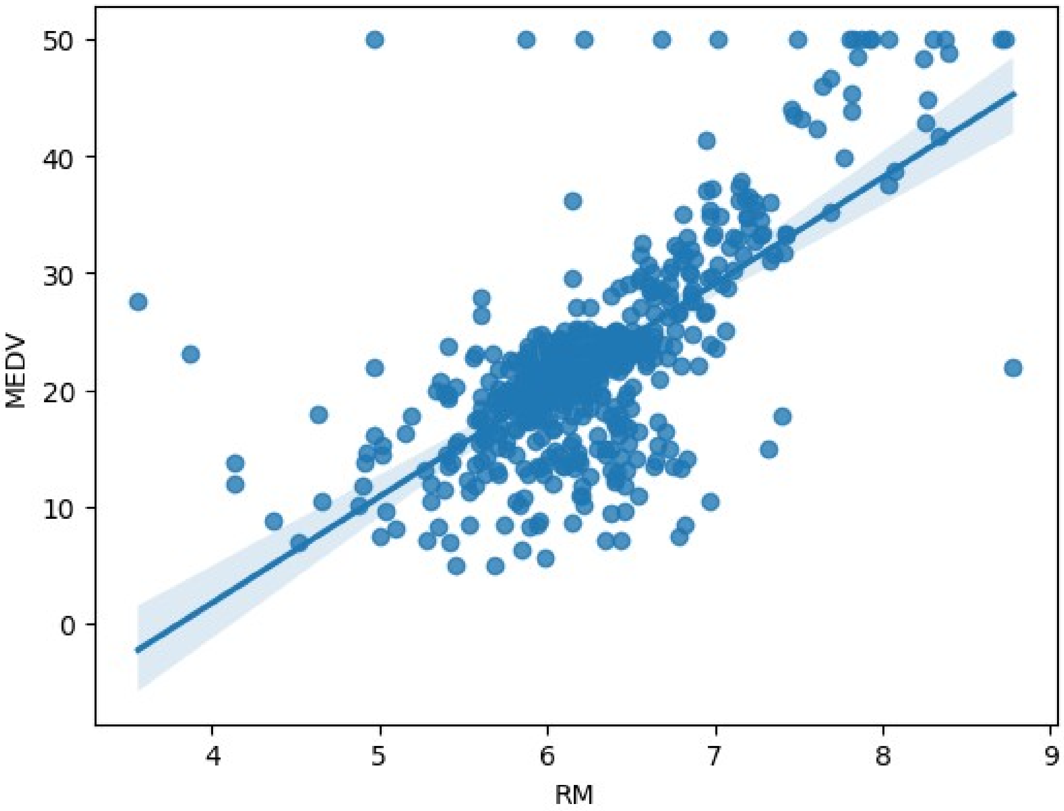
```
sn.scatterplot(x=df['LSTAT'],y=df['MEDV'])
```

```
<Axes: xlabel='LSTAT', ylabel='MEDV'>
```



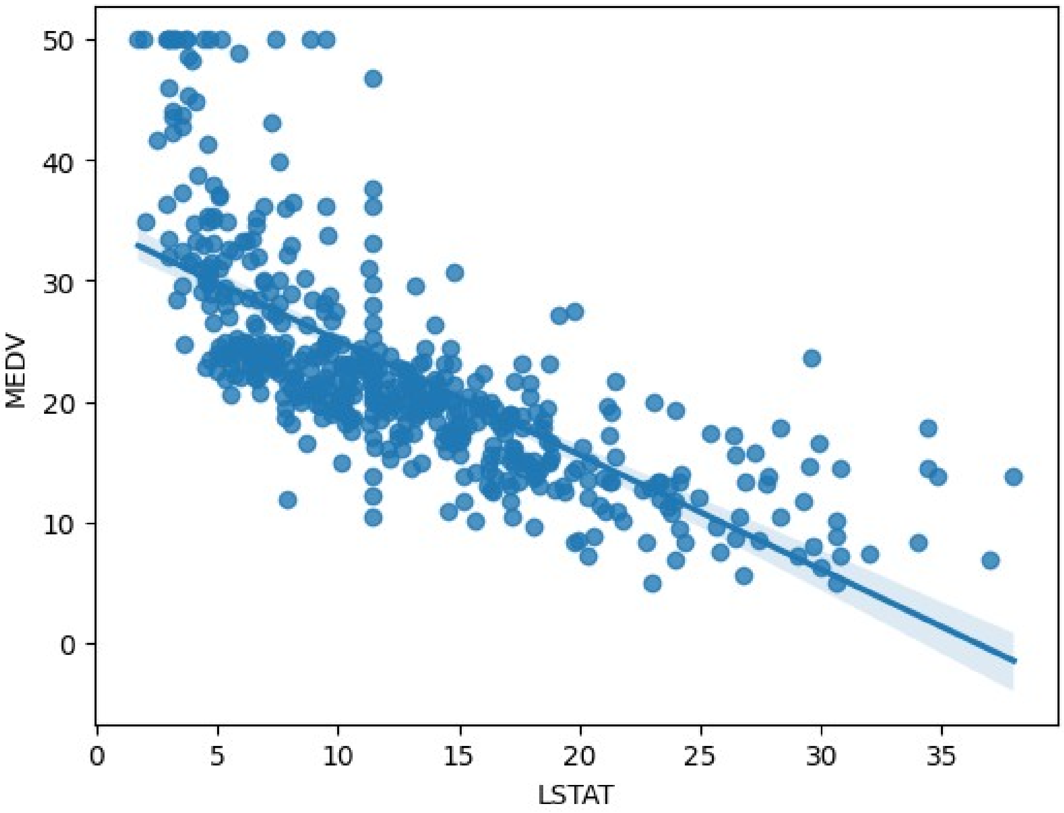
```
sn.scatterplot(x=df['PTRATIO'],y=df['MEDV'])

sn.regplot(x=df['RM'],y=df['MEDV'])
# it displays scatterplot + regression line
```

```
<Axes: xlabel='RM', ylabel='MEDV'>
```



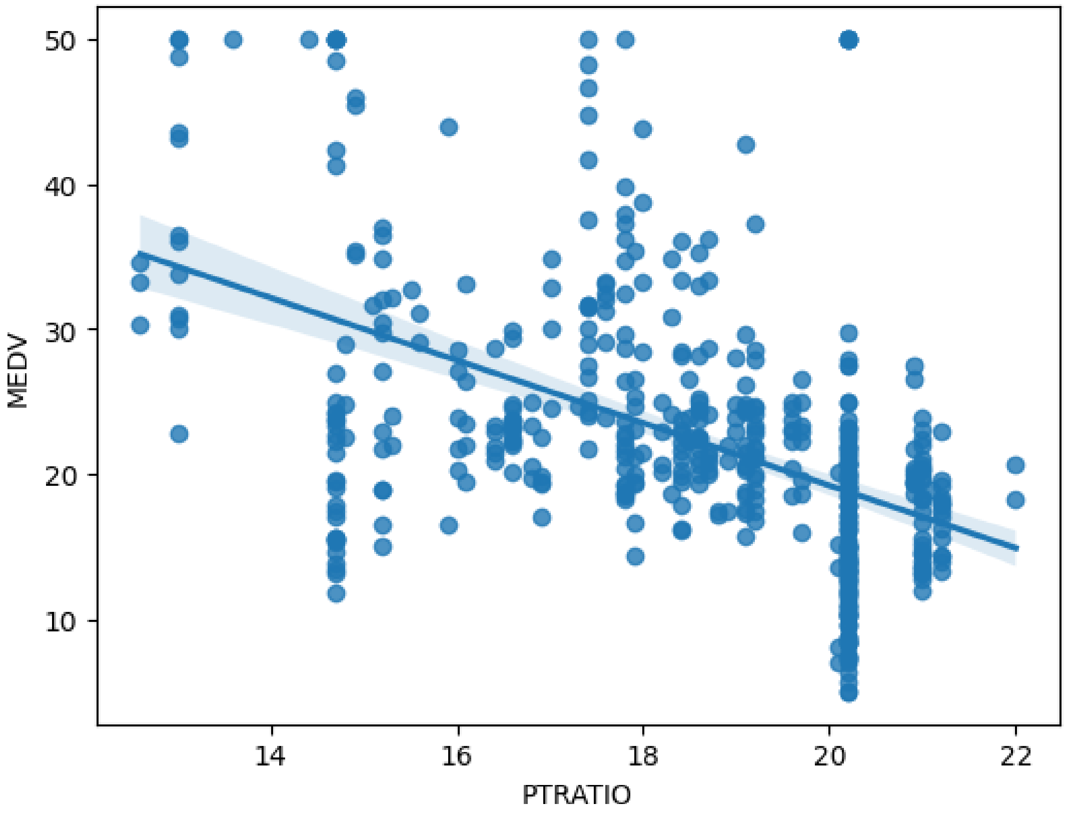
```
sn.regplot(x=df['LSTAT'],y=df['MEDV'])
```

```
<Axes: xlabel='LSTAT', ylabel='MEDV'>
```



```
sn.regplot(x=df['PTRATIO'],y=df['MEDV'])
```

```
<Axes: xlabel='PTRATIO', ylabel='MEDV'>
```



Build linear regression Model

```
from sklearn.linear_model import LinearRegression

x=df[['RM']]
y=df[['MEDV']]
# we should write in double square bracket because we want 2d data

lr=LinearRegression()
lr.fit(x,y)
lr.score(x,y)

0.48352545599133423

# R^2 score range(0-1)
# if it is between 0.5-1 it is considered best fit

x=df[['LSTAT']]
y=df[['MEDV']]
lr1=LinearRegression()
```

```
lr1.fit(x,y)
lr1.score(x,y)
```

```
0.522863589450163
```

```
x=df[['RM','LSTAT']]
y=df[['MEDV']]
lr2=LinearRegression()
lr2.fit(x,y)
lr2.score(x,y)
```

```
0.6280305701530031
```

```
x=df[['LSTAT','RM','PTRATIO']]
y=df[['MEDV']]
lr3=LinearRegression()
lr3.fit(x,y)
lr3.score(x,y)
```

```
0.6695386967800379
```

```
from sklearn.model_selection import train_test_split
```

```
x=df.iloc[:,:-1]
y=df.iloc[:,-1]
```

```
x.head()
```

```
      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
0  396.90   4.98
1  396.90   9.14
2  392.83   4.03
3  394.63   2.94
4  396.90  11.43
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,rando
m_state=34)
```

```
x_train.shape
```

```
(404, 13)

x_test.shape

(102, 13)

y_train.shape

(404,)

y_test.shape

(102,)



lreg=LinearRegression()
lreg.fit(x_train,y_train)

LinearRegression()

y_pred=lreg.predict(x_test)

df1=pd.DataFrame({"Actual":y_test, "Predicted":y_pred})
df1.head()

     Actual  Predicted
218    21.5  24.634802
370    50.0  34.029228
451    15.2  19.390038
230    24.3  24.392637
165    25.0  25.161155

from sklearn.metrics import mean_squared_error,mean_absolute_error

print('MAE: ',mean_absolute_error(y_test,y_pred))

MAE:  3.2171716291908186

print('MSE: ',mean_squared_error(y_test,y_pred))

MSE:  20.71871585814613

print('RMSE: ',np.sqrt(mean_squared_error(y_test,y_pred)))

RMSE:  4.551781613626265
```