```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,r2_score
from sklearn.preprocessing import StandardScaler

import warnings
warnings.filterwarnings('ignore')

data=pd.read_csv('Boston housing dataset.csv')

data.head
```

```
<bound method NDFrame.head of         CRIM    ZN  INDUS  CHAS    NOX
RM   AGE     DIS  RAD  TAX  \
0    0.00632  18.0   2.31   0.0  0.538  6.575  65.2  4.0900    1  296

1    0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671    2  242

2    0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671    2  242

3    0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622    3  222

4    0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622    3  222

..       ...   ...    ...   ...    ...    ...   ...     ...  ...  ...

501  0.06263   0.0  11.93   0.0  0.573  6.593  69.1  2.4786    1  273

502  0.04527   0.0  11.93   0.0  0.573  6.120  76.7  2.2875    1  273

503  0.06076   0.0  11.93   0.0  0.573  6.976  91.0  2.1675    1  273

504  0.10959   0.0  11.93   0.0  0.573  6.794  89.3  2.3889    1  273

505  0.04741   0.0  11.93   0.0  0.573  6.030   NaN  2.5050    1  273


     PTRATIO       B  LSTAT  MEDV
0       15.3  396.90   4.98  24.0
1       17.8  396.90   9.14  21.6
2       17.8  392.83   4.03  34.7
3       18.7  394.63   2.94  33.4
4       18.7  396.90    NaN  36.2
..       ...     ...    ...   ...
501     21.0  391.99    NaN  22.4
502     21.0  396.90   9.08  20.6
503     21.0  396.90   5.64  23.9
```

```
504     21.0  393.45   6.48  22.0
505     21.0  396.90   7.88  11.9

[506 rows x 14 columns]>
```

```
data.shape
```

```
(506, 14)
```

```
data.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
```

```
data.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
```

```
data.CHAS.unique()
```

```
array([ 0., nan,  1.])
```

```
data.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. ])
```

```
data.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
```

```
data.duplicated().sum()
```

```
0
```

```
np.int64(0)
```

```
0
```

```
df=data.copy()
```

```
df['CRIM'].fillna(df['CRIM'].mean(),inplace=True)
df['ZN'].fillna(df['ZN'].mean(),inplace=True)
df['CHAS'].fillna(df['CHAS'].mode()[0],inplace=True)
df['INDUS'].fillna(df['INDUS'].mean(),inplace=True)
df['AGE'].fillna(df['AGE'].median(),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.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  11.43  36.2
```

```
df['CHAS']=df['CHAS'].astype('int')
```

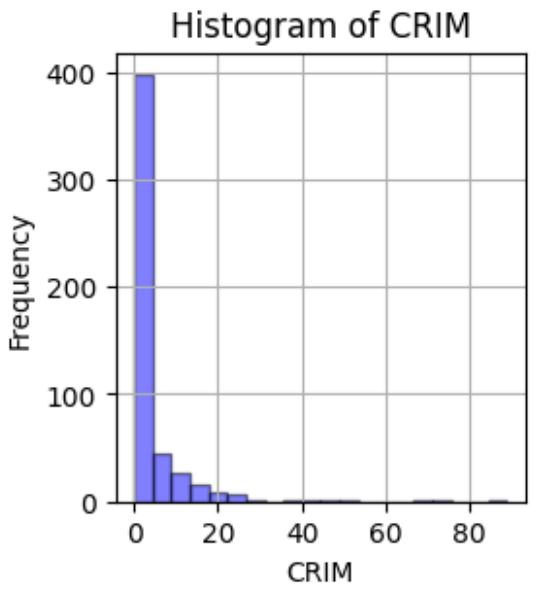
```
df.describe().T
```

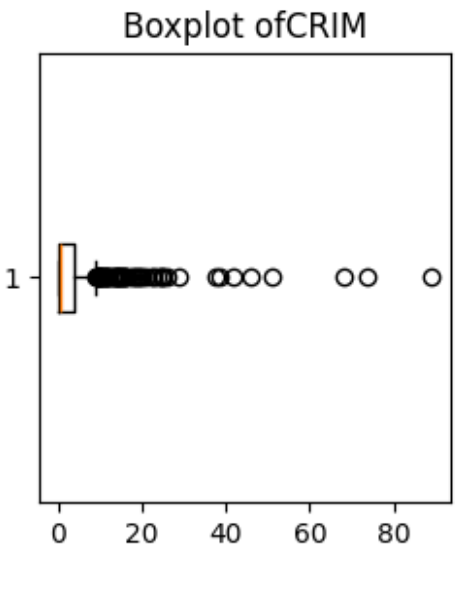
```
         count        mean         std        min         25%
50%  \
CRIM     506.0    3.611874    8.545770    0.00632    0.083235
0.29025
ZN       506.0   11.211934   22.921051    0.00000    0.000000
0.00000
INDUS    506.0   11.083992    6.699165    0.46000    5.190000
9.90000
CHAS     506.0    0.067194    0.250605    0.00000    0.000000
0.00000
NOX      506.0    0.554695    0.115878    0.38500    0.449000
0.53800
RM       506.0    6.284634    0.702617    3.56100    5.885500
```

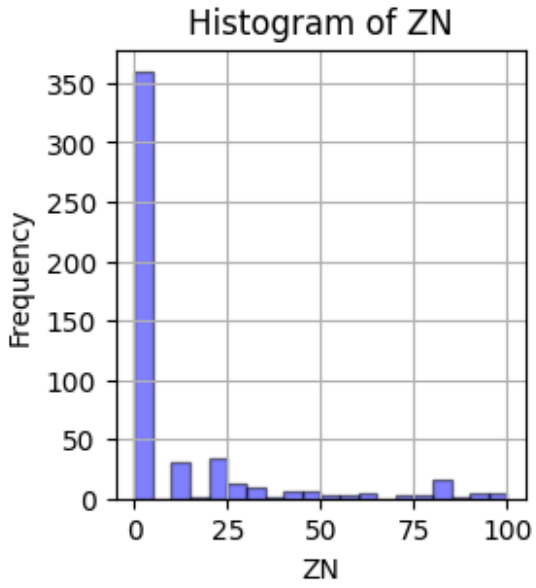
```
6.20850
AGE      506.0   68.845850   27.486962    2.90000   45.925000
76.80000
DIS      506.0    3.795043    2.105710    1.12960    2.100175
3.20745
RAD      506.0    9.549407    8.707259    1.00000    4.000000
5.00000
TAX      506.0  408.237154  168.537116  187.00000  279.000000
330.00000
PTRATIO  506.0   18.455534    2.164946   12.60000   17.400000
19.05000
B        506.0  356.674032   91.294864    0.32000  375.377500
391.44000
LSTAT    506.0   12.664625    7.017219    1.73000    7.230000
11.43000
MEDV     506.0   22.532806    9.197104    5.00000   17.025000
21.20000
```

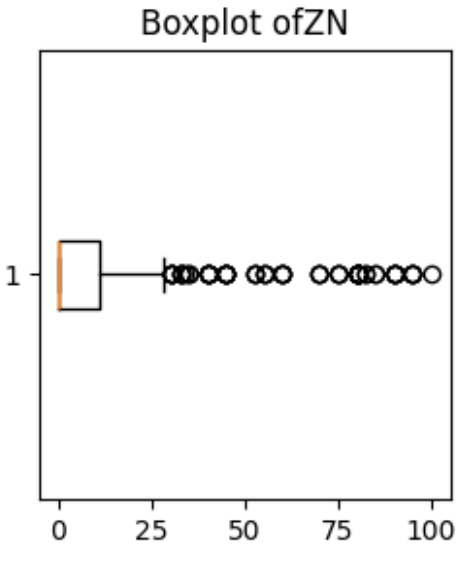
```
                75%       max
CRIM       3.611874   88.9762
ZN        11.211934  100.0000
INDUS     18.100000   27.7400
CHAS       0.000000    1.0000
NOX        0.624000    0.8710
RM         6.623500    8.7800
AGE       93.575000  100.0000
DIS        5.188425   12.1265
RAD       24.000000   24.0000
TAX      666.000000  711.0000
PTRATIO   20.200000   22.0000
B        396.225000  396.9000
LSTAT     16.570000   37.9700
MEDV      25.000000   50.0000
```

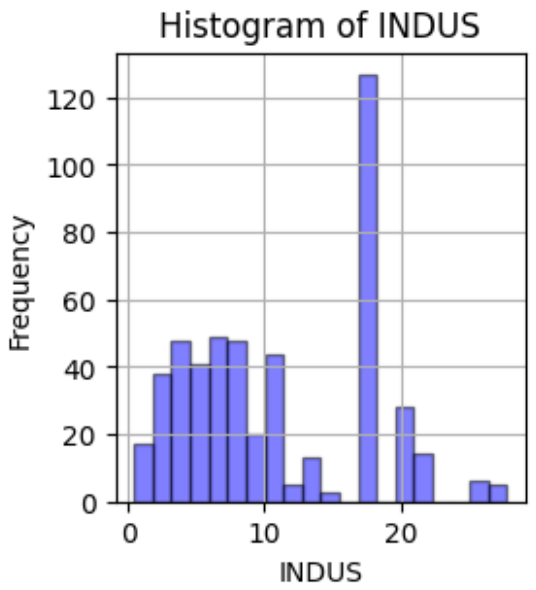
```
for i in df.columns:
    plt.figure(figsize=(6,3))
    plt.subplot(1,2,1)
    df[i].hist(bins=20,alpha=0.5,color='b',edgecolor='black')
    plt.title(f'Histogram of {i}')
    plt.xlabel(i)
    plt.ylabel('Frequency')
    plt.subplot(1,2,2)
    plt.boxplot(df[i],vert=False)
    plt.title(f'Boxplot of{i}')
    plt.show()
```

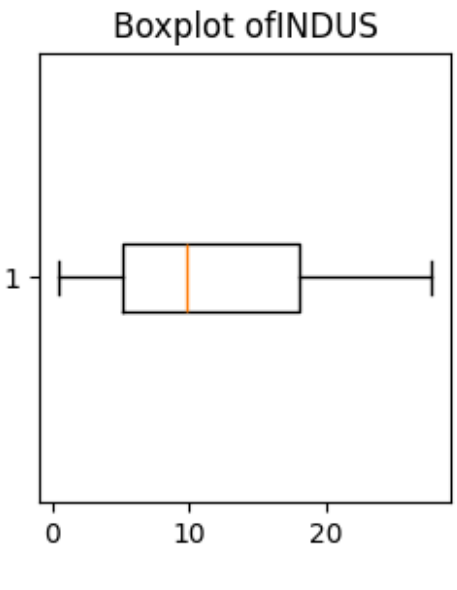

Histogram of CRIM
Frequency
CRIM
0
100
200
300
400
0
20
40
60
80

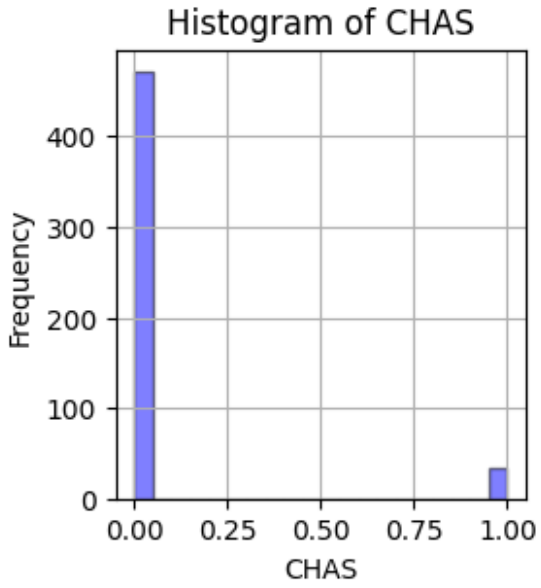

Boxplot ofCRIM
1
0
20
40
60
80

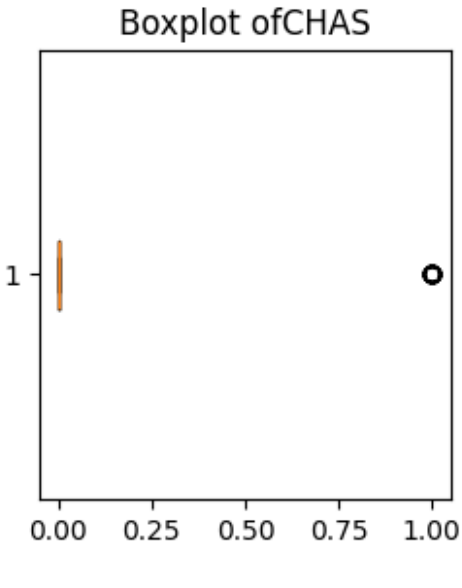

Histogram of ZN
Frequency
ZN
0
50
100
150
200
250
300
350
0
25
50
75
100

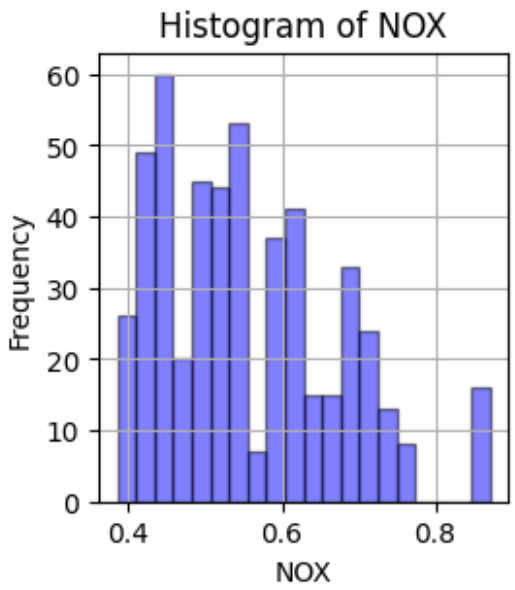

Boxplot ofZN
1
0
25
50
75
100

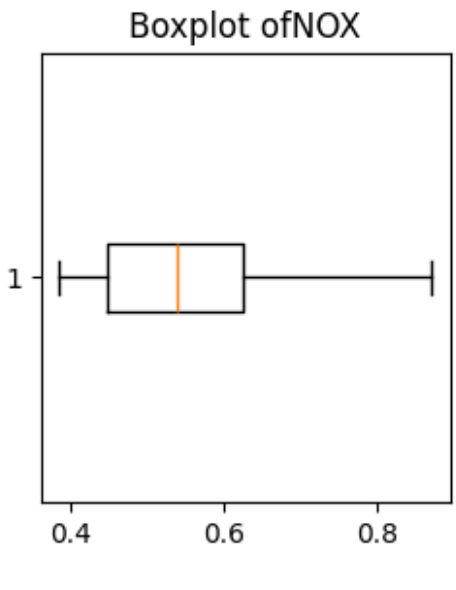

Histogram of INDUS
Frequency
INDUS
0
20
40
60
80
100
120
0
10
20

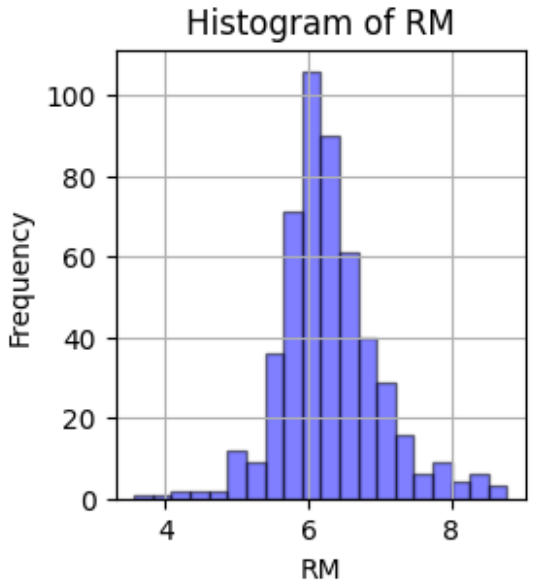

Boxplot ofINDUS
1
0
10
20

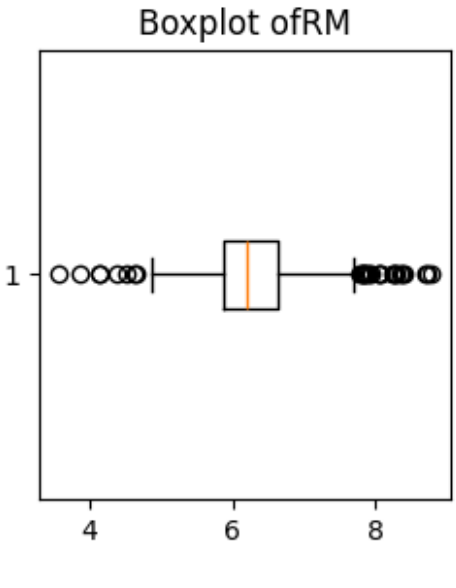

Histogram of CHAS
Frequency
CHAS
0
100
200
300
400
0.00
0.25
0.50
0.75
1.00

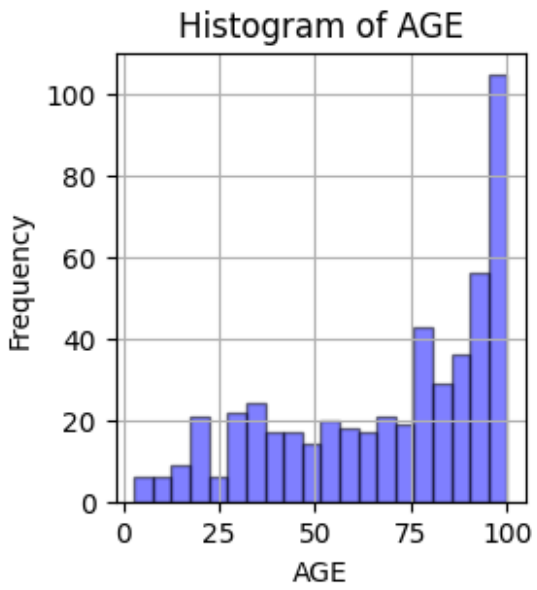

Boxplot ofCHAS
1
0.00
0.25
0.50
0.75
1.00

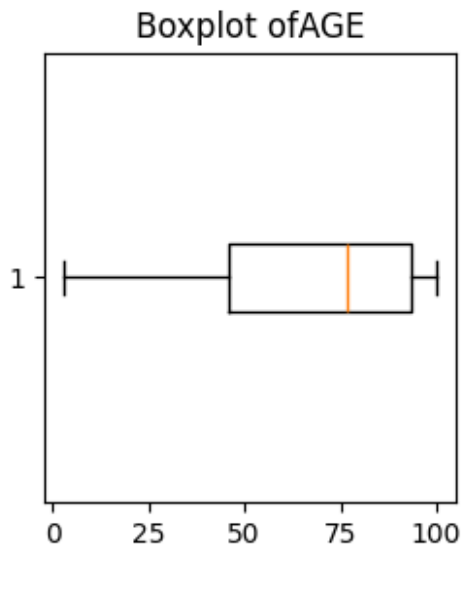

Histogram of NOX
Frequency
NOX
0
10
20
30
40
50
60
0.4
0.6
0.8

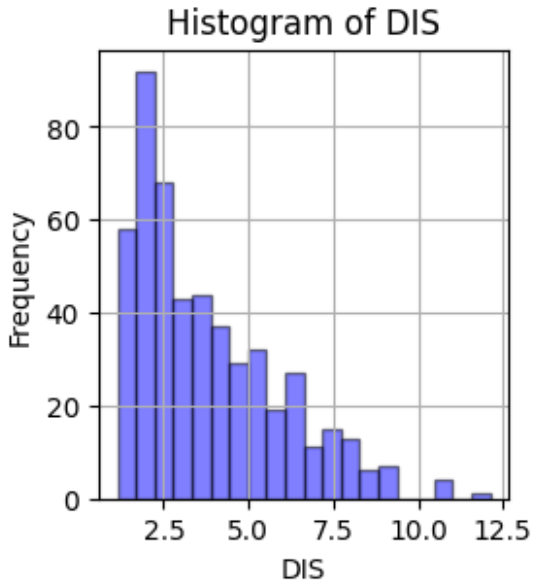

Boxplot ofNOX
1
0.4
0.6
0.8

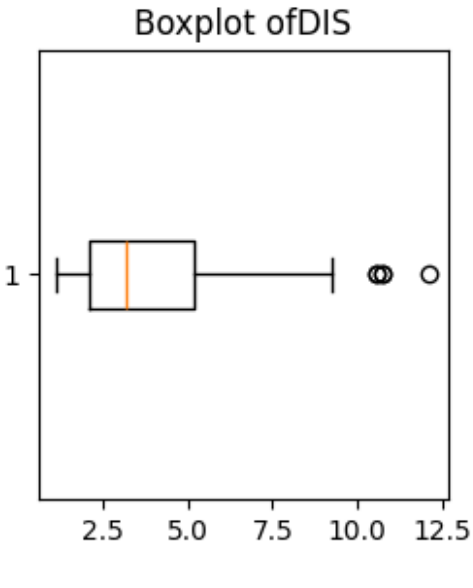

Histogram of RM
Frequency
RM
0
20
40
60
80
100
4
6
8

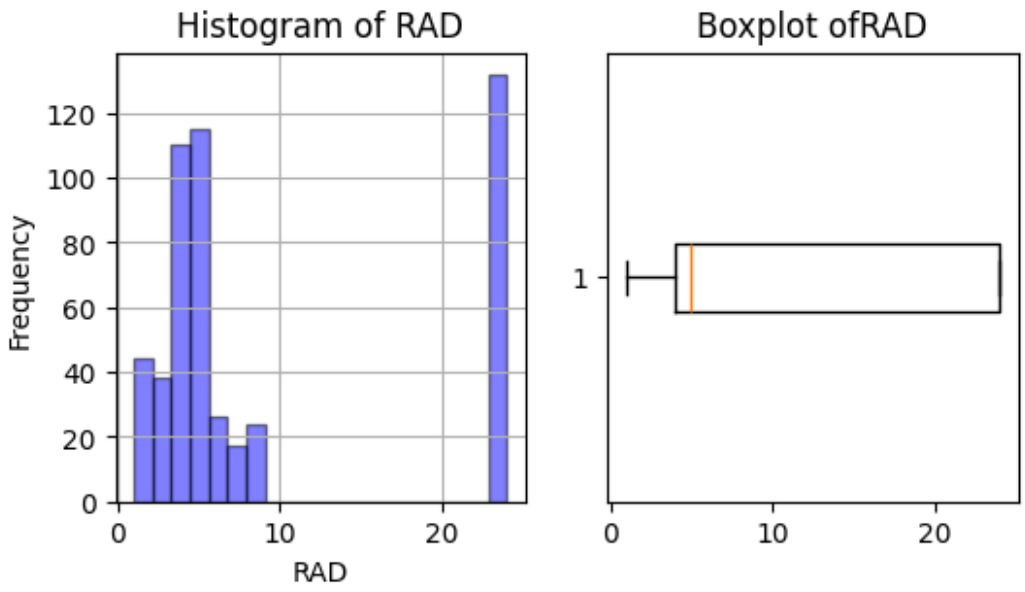

Boxplot ofRM
1
4
6
8


Histogram of AGE
Frequency
AGE
0
20
40
60
80
100
0
25
50
75
100

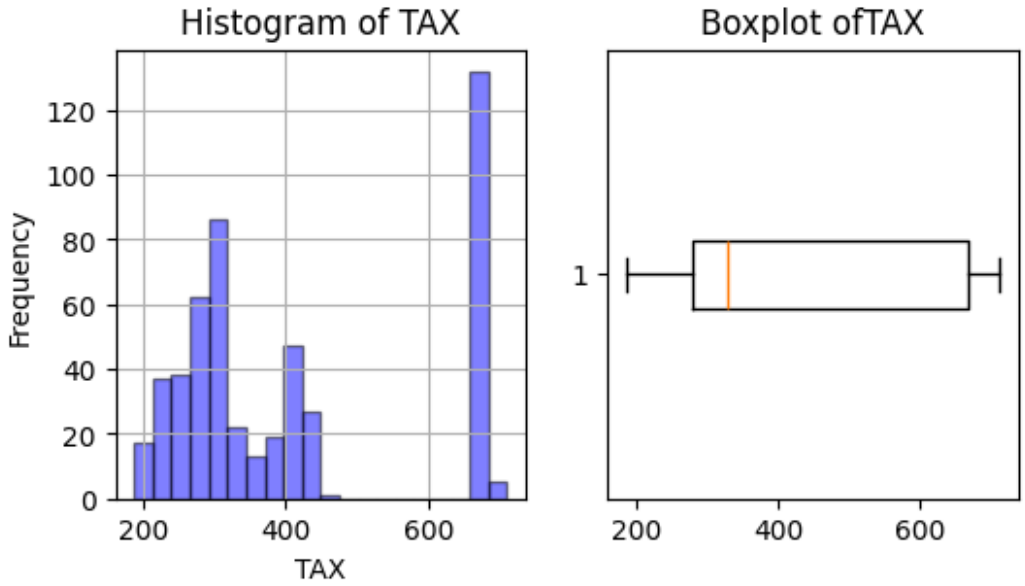

Boxplot ofAGE
1
0
25
50
75
100


Histogram of DIS
Frequency
DIS
0
20
40
60
80
2.5
5.0
7.5
10.0
12.5

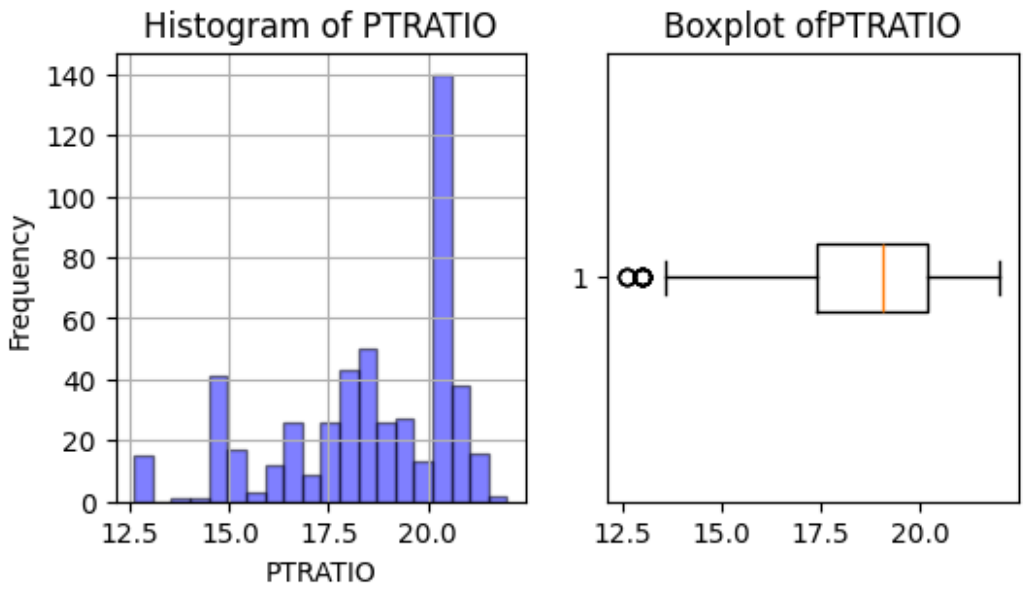

Boxplot ofDIS
1
2.5
5.0
7.5
10.0
12.5


Histogram of RAD
Frequency
RAD
0
20
40
60
80
100
120
0
10
20

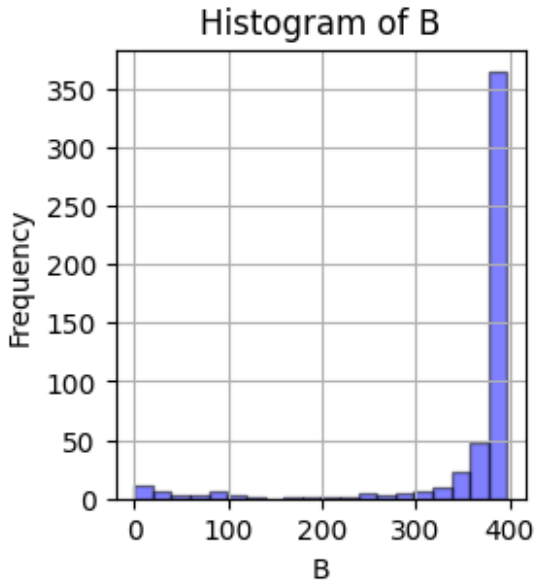
Boxplot ofRAD
1
0
10
20

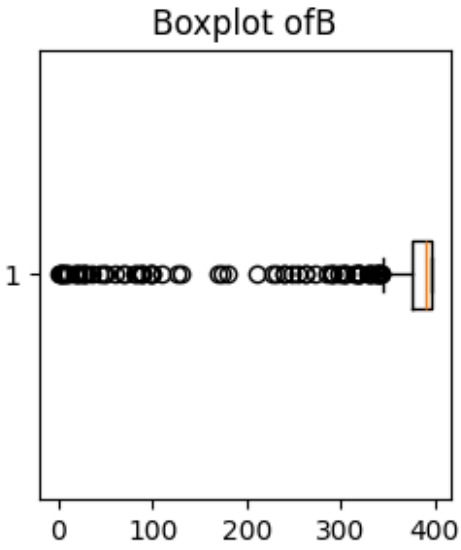

Histogram of TAX
Frequency
TAX
0
20
40
60
80
100
120
200
400
600

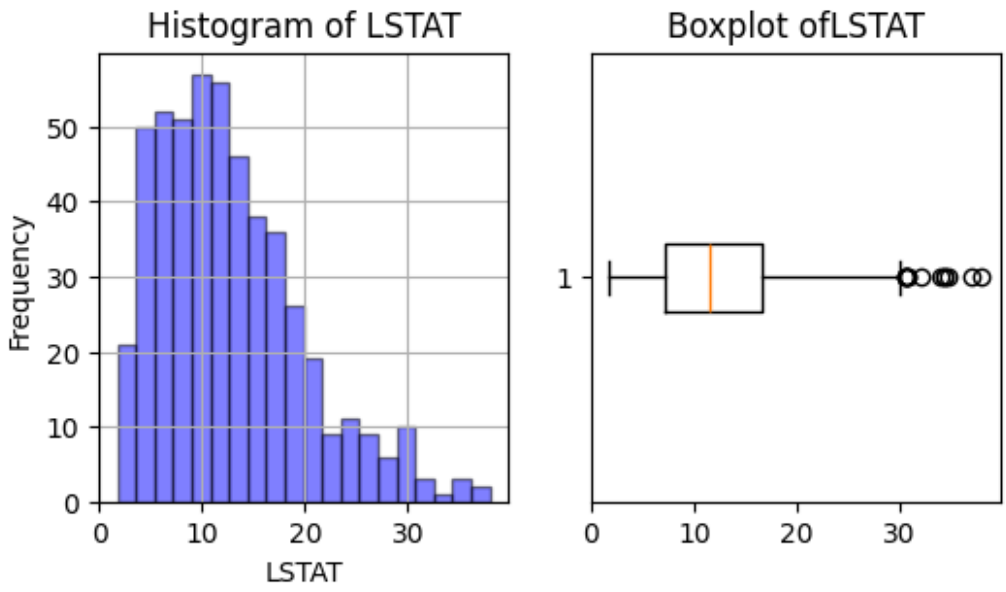
Boxplot ofTAX
1
200
400
600

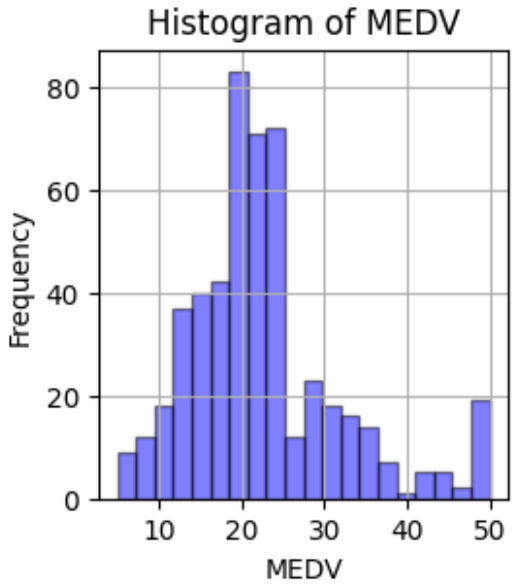

Histogram of PTRATIO
Frequency
PTRATIO
0
20
40
60
80
100
120
140
12.5
15.0
17.5
20.0

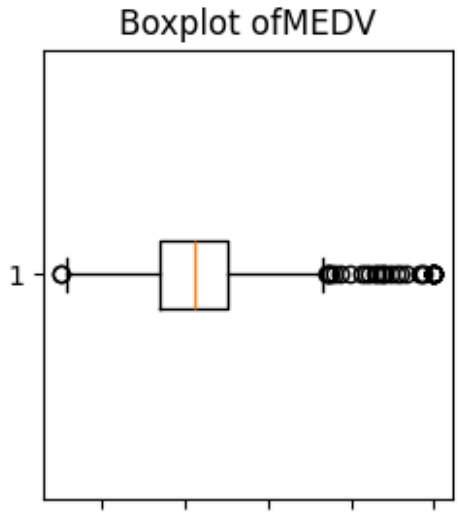
Boxplot ofPTRATIO
1
12.5
15.0
17.5
20.0


Histogram of B
Frequency
B
0
50
100
150
200
250
300
350
0
100
200
300
400


Boxplot ofB
1
0
100
200
300
400

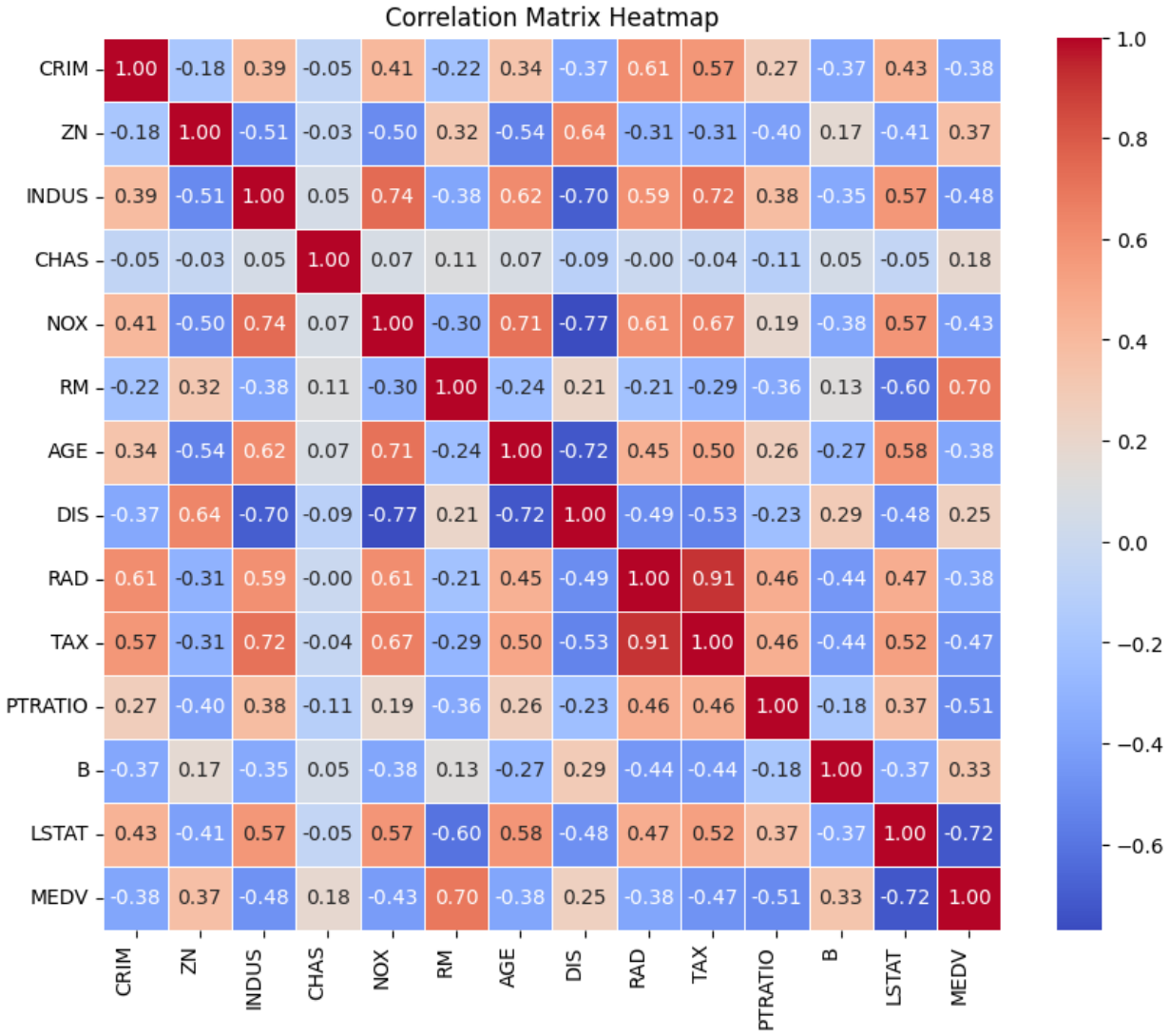







```
corr=df.corr(method='pearson')

plt.figure(figsize=(10,8))
sns.heatmap(corr,annot=True,cmap="coolwarm",fmt=".2f",linewidths=0.5)
plt.xticks(rotation=90,ha='right')
plt.yticks(rotation=0)
plt.title("Correlation Matrix Heatmap")
plt.show()
```




```
X=df.drop('MEDV',axis=1)
y=df['MEDV']

scale=StandardScaler()
X_scaled=scale.fit_transform(X)

X_train,X_test,y_train,y_test=train_test_split(X_scaled,y,test_size=0.
2,random_state=42)

model=LinearRegression()

model.fit(X_train,y_train)

LinearRegression()

y_pred=model.predict(X_test)
y_pred
```

```
array([28.99719439, 36.56606809, 14.51022803, 25.02572187,
18.42885474,
       23.02785726, 17.95437605, 14.5769479 , 22.14430832,
20.84584632,
       25.15283588, 18.55925182, -5.69168071, 21.71242445,
19.06845707,
       25.94275348, 19.70991322,  5.85916505, 40.9608103 ,
17.21528576,
       25.36124981, 30.26007975, 11.78589412, 23.48106943,
17.35338161,
       15.13896898, 21.61919056, 14.51459386, 23.17246824,
19.40914754,
       22.56164985, 25.21208496, 25.88782605, 16.68297496,
16.44747174,
       16.65894826, 31.10314158, 20.25199803, 24.38567686,
23.09800032,
       14.47721796, 32.36053979, 43.01157914, 17.61473728,
27.60723089,
       16.43366912, 14.25719607, 26.0854729 , 19.75853278,
30.15142187,
       21.01932313, 33.72128781, 16.39180467, 26.36438908,
39.75793372,
       22.02419633, 18.39453126, 32.81854401, 25.370573  ,
12.82224665,
       22.76128341, 30.73955199, 31.34386371, 16.27681305,
20.36945226,
       17.23156773, 20.15406451, 26.15613066, 30.92791361,
11.42177654,
       20.89590447, 26.58633798, 11.01176073, 12.76831709,
23.73870867,
        6.37180464, 21.6922679 , 41.74800223, 18.64423785,
8.82325704,
       20.96406016, 13.20179007, 20.99146149,  9.17404063,
23.0011185 ,
       32.41062673, 18.99778065, 25.56204885, 28.67383635,
19.76918944,
       25.94842754,  5.77674362, 19.514431  , 15.22571165,
10.87671123,
       20.08359505, 23.77725749,  0.05985008, 13.56333825,
16.1215622 ,
       22.74200442, 24.36218289])

mse=mean_squared_error(y_test,y_pred)

rmse=np.sqrt(mse)

r2=r2_score(y_test,y_pred)
```

```
print(f'Mean Squared Error:{mse}')
print(f'Root Mean Squared Error:{rmse}')
print(f'R-Squared:{r2}')
```

```
Mean Squared Error:24.944071172175573
Root Mean Squared Error:4.99440398567993
R-Squared:0.6598556613717497
```