house-price

June 24, 2024

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
path = r"C:\Users\asus\OneDrive\Desktop\items\New folder\python\New␣
 ↪folder\Boston-house-price-data.csv"

df = pd.read_csv(path)
df.head()
```

[2]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	\
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	

	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	5.33	36.2

CRIM: Per capita crime rate by town. ZN: Proportion of residential land zoned for lots over 25,000 square feet. INDUS: Proportion of non-retail business acres per town. CHAS: Charles River dummy variable (1 if tract bounds river; 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 1940. DIS: Weighted distances to five Boston employment centers. RAD: Index of accessibility to radial highways. TAX: Full-value property tax rate per $10,000. PTRATIO$: $Pupil - teacherratiobytown. B$: $1000(Bk - 0.63)^2 whereBkistheproportionofBlackresidentsbytown. LSTAT$: $Percentageoflowerstatusofthepopulation. MEDV$: $Medianvalueofowner - occupiedhomesin1000$s (this is the target variable).

[3]:
```
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     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    int64  
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    int64  
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 55.5 KB
```

[4]:
```
df.isnull().sum()
```

[4]:
```
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
```

[5]:
```
df.duplicated().sum()
```

[5]:
```
0
```

[6]:
```
df.count()/ df.shape[0] * 100
```

[6]:
```
CRIM       100.0
ZN         100.0
INDUS      100.0
CHAS       100.0
NOX        100.0
RM         100.0
AGE        100.0
DIS        100.0
RAD        100.0
TAX        100.0
PTRATIO    100.0
B          100.0
LSTAT      100.0
MEDV       100.0
dtype: float64
```

[7]:
```
df.dtypes
```

[7]:
```
CRIM       float64
ZN         float64
INDUS      float64
CHAS         int64
NOX        float64
RM         float64
AGE        float64
DIS        float64
RAD          int64
TAX        float64
PTRATIO    float64
B          float64
LSTAT      float64
MEDV       float64
dtype: object
```

[8]:
```
df.describe()
```

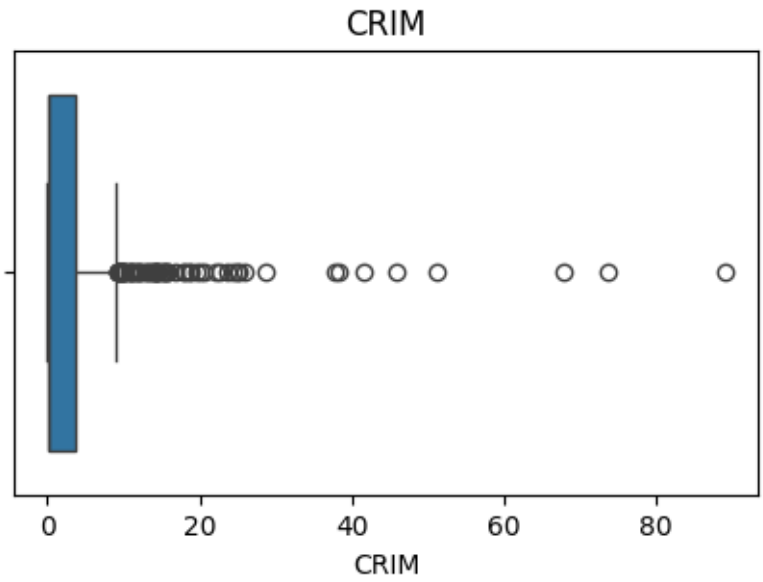
[8]:

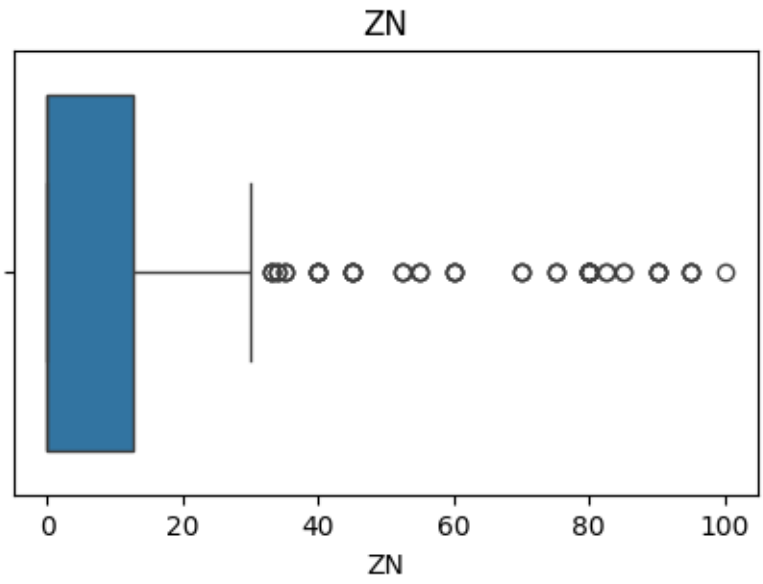
	CRIM	ZN	INDUS	CHAS	NOX	RM	\
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	

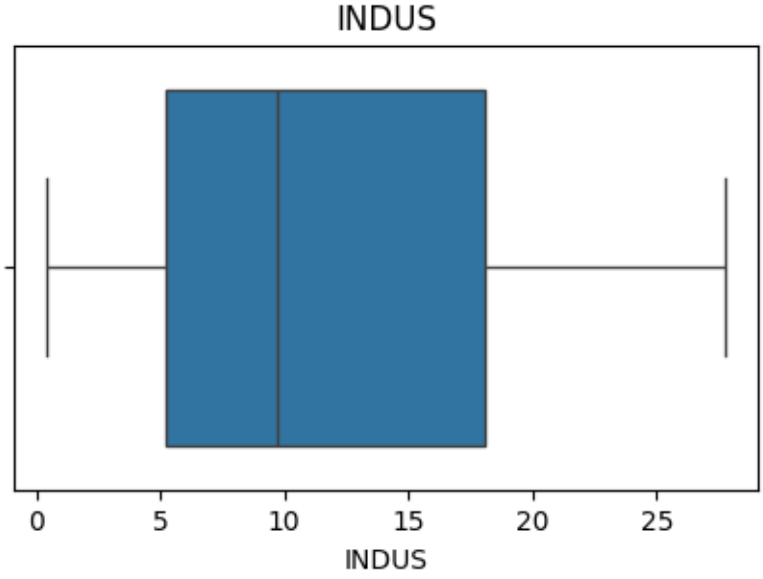
	AGE	DIS	RAD	TAX	PTRATIO	B	\

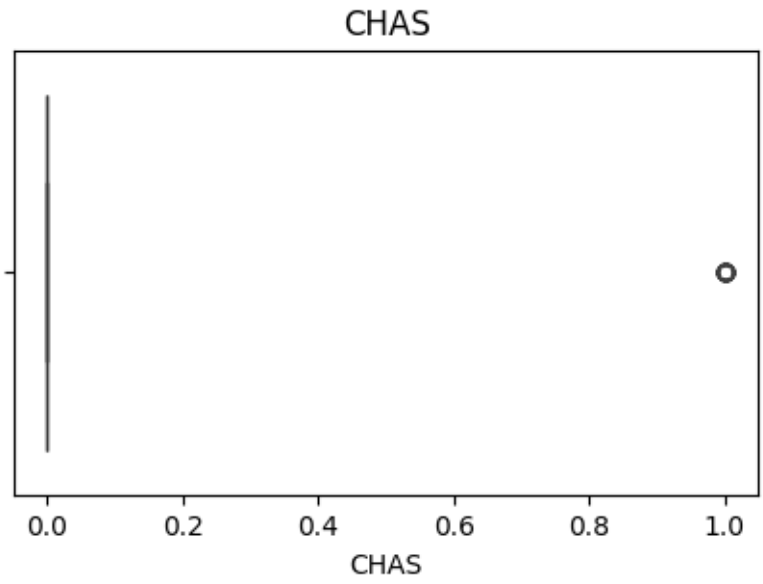
```
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean    68.574901    3.795043    9.549407  408.237154   18.455534  356.674032
std     28.148861    2.105710    8.707259  168.537116    2.164946   91.294864
min      2.900000    1.129600    1.000000  187.000000   12.600000    0.320000
25%     45.025000    2.100175    4.000000  279.000000   17.400000  375.377500
50%     77.500000    3.207450    5.000000  330.000000   19.050000  391.440000
75%     94.075000    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    12.653063   22.532806
std      7.141062    9.197104
min      1.730000    5.000000
25%      6.950000   17.025000
50%     11.360000   21.200000
75%     16.955000   25.000000
max     37.970000   50.000000
```

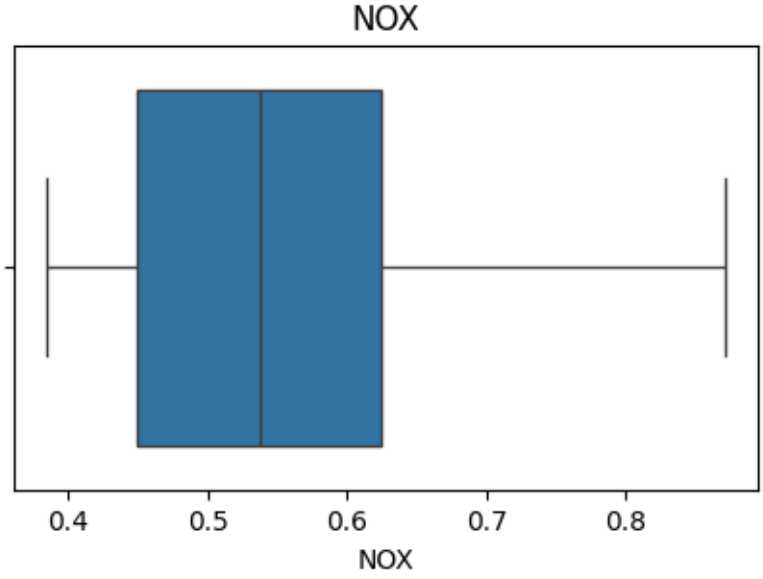
[9]:
```
#Box PLot
for column in df.columns:
    plt.figure(figsize=(5, 3))
    sns.boxplot(x=df[column])
    plt.title(column)
    plt.show()
```

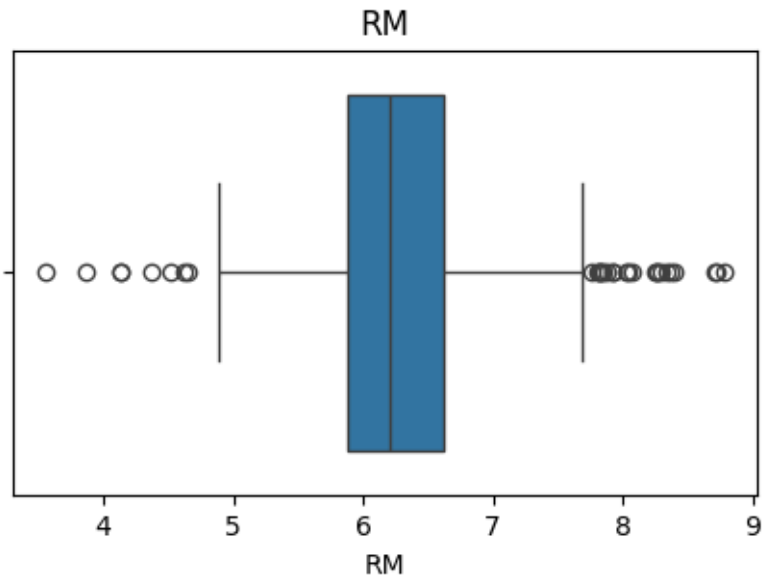


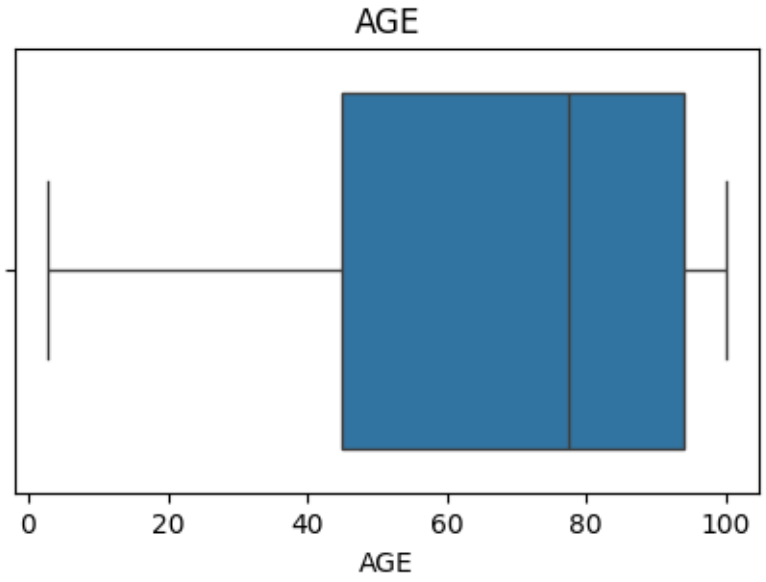

ZN
0
20
40
60
80
100
ZN

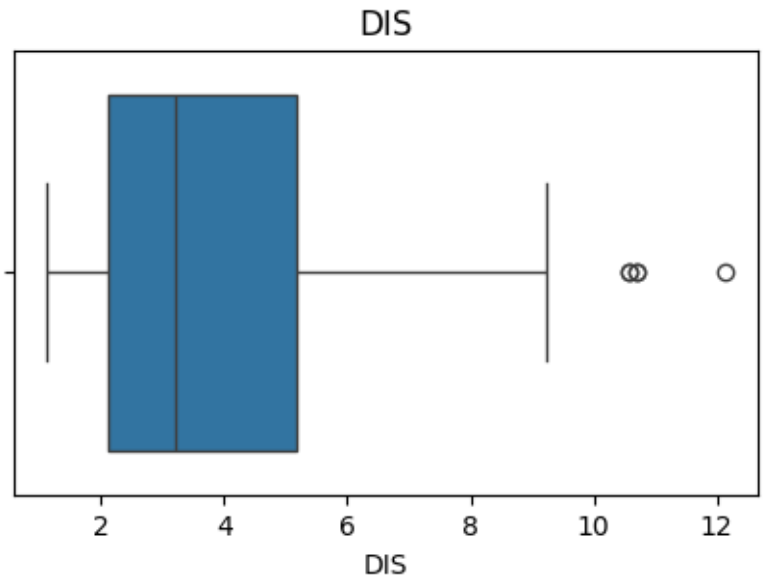

INDUS
0
5
10
15
20
25
INDUS

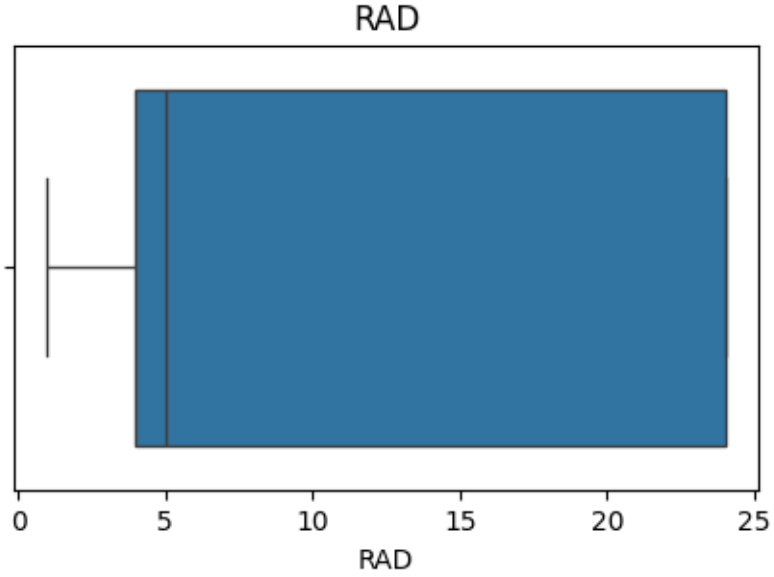

CHAS
0.0
0.2
0.4
0.6
0.8
1.0
CHAS

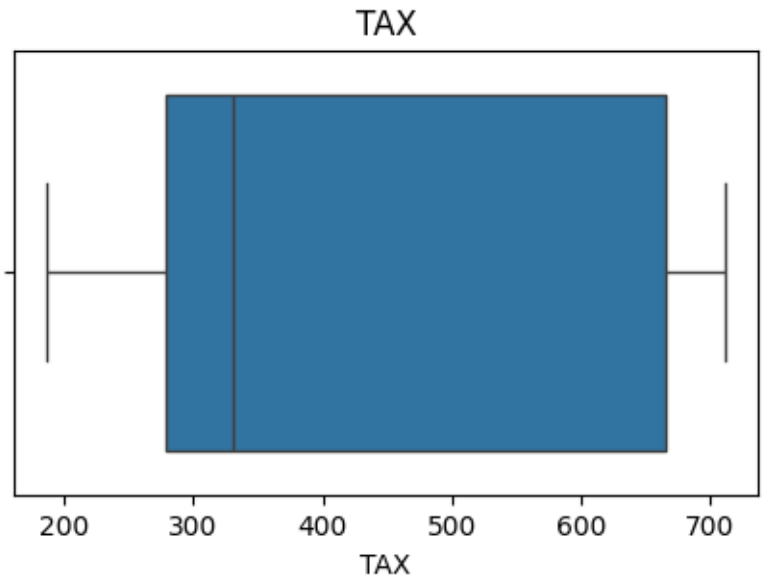

NOX
0.4
0.5
0.6
0.7
0.8
NOX

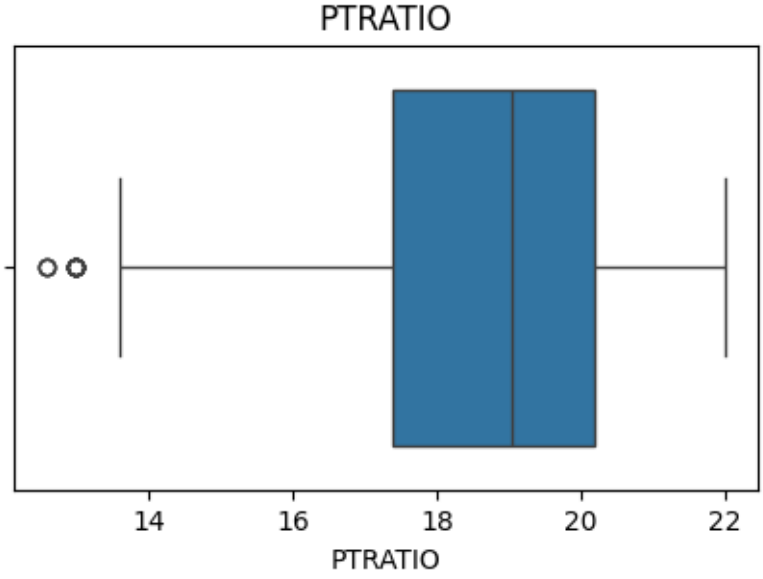

RM
4
5
6
7
8
9
RM

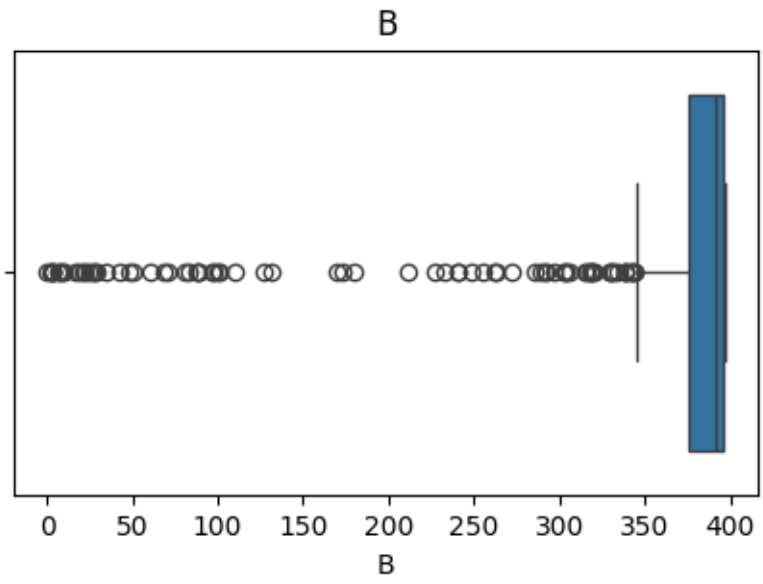

AGE
0
20
40
60
80
100
AGE

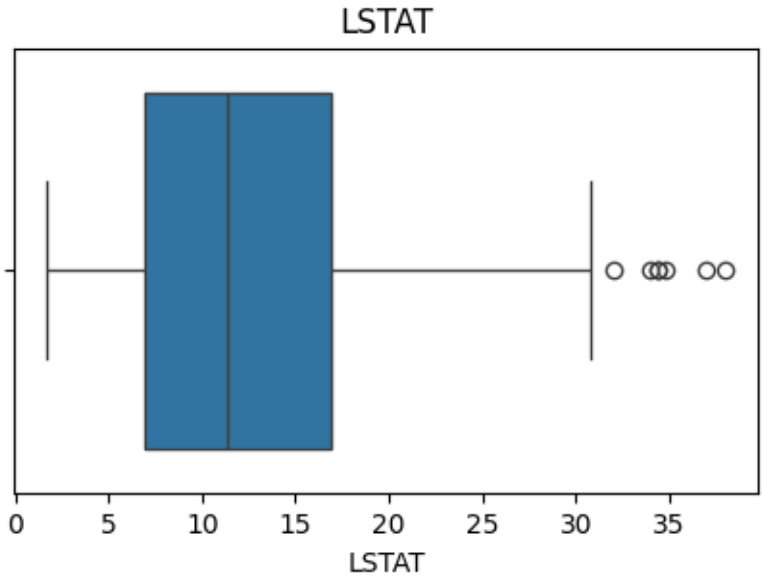

DIS
2
4
6
8
10
12
DIS

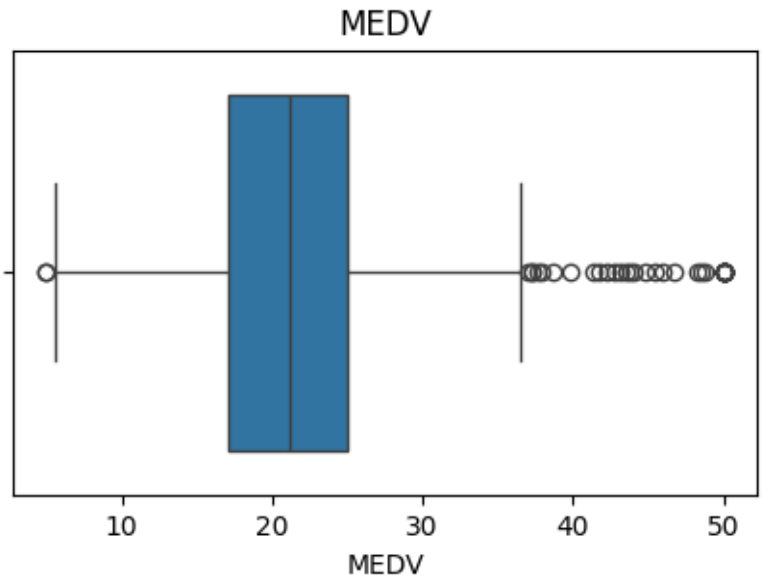

RAD
0
5
10
15
20
25
RAD


TAX
200
300
400
500
600
700
TAX

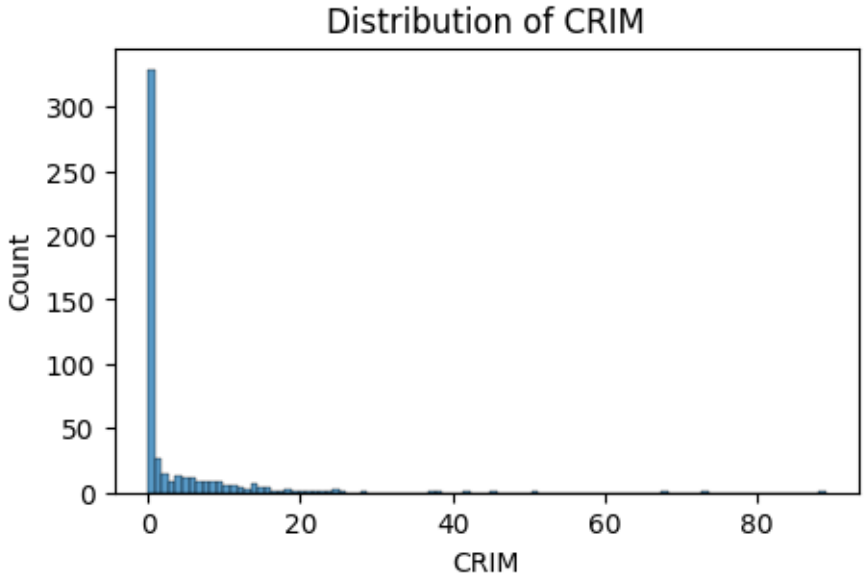

PTRATIO
14
16
18
20
22
PTRATIO

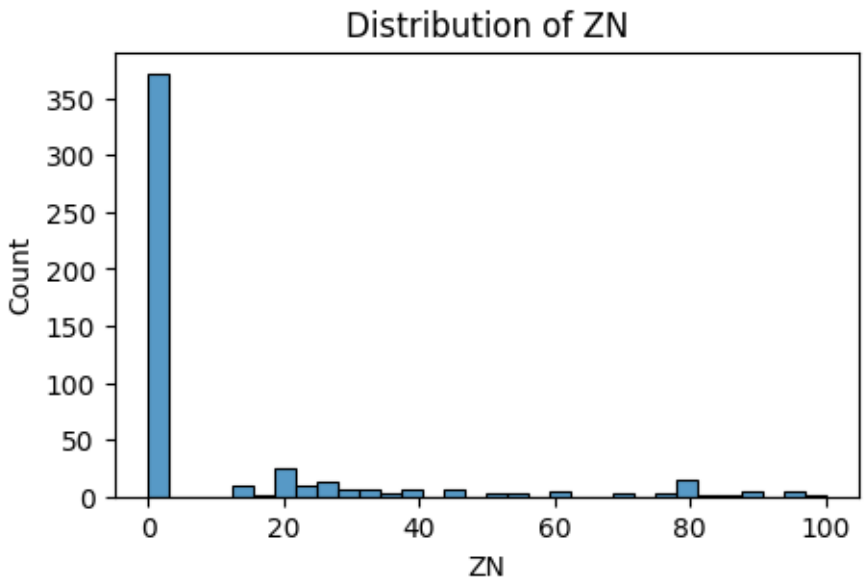

B
0
50
100
150
200
250
300
350
400
B

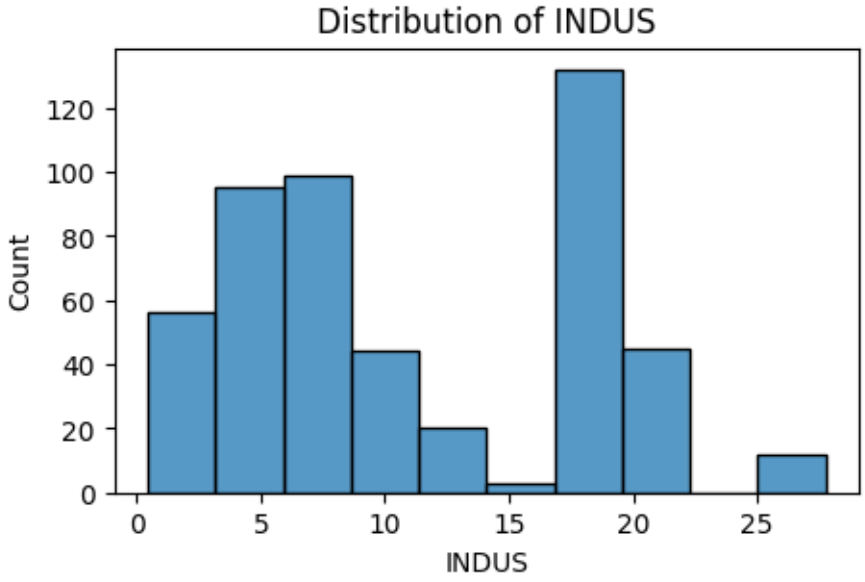

LSTAT
0
5
10
15
20
25
30
35
LSTAT

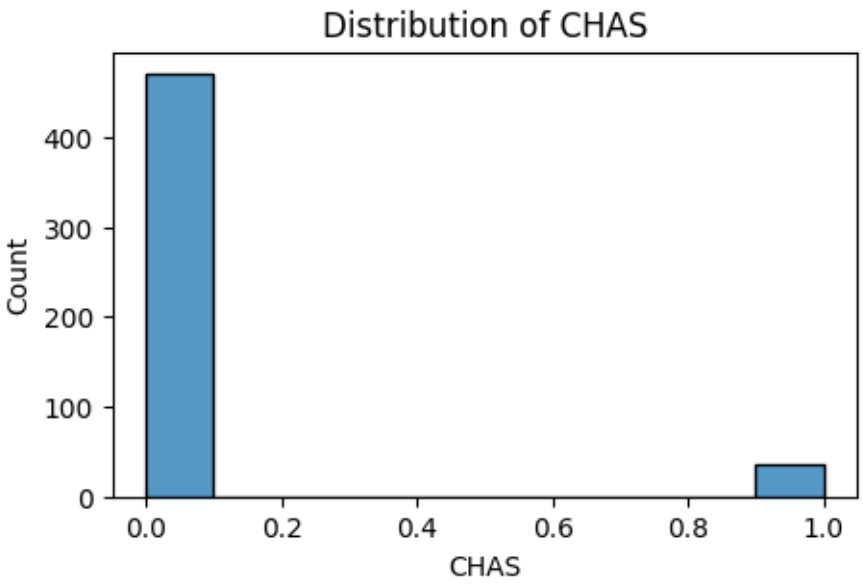


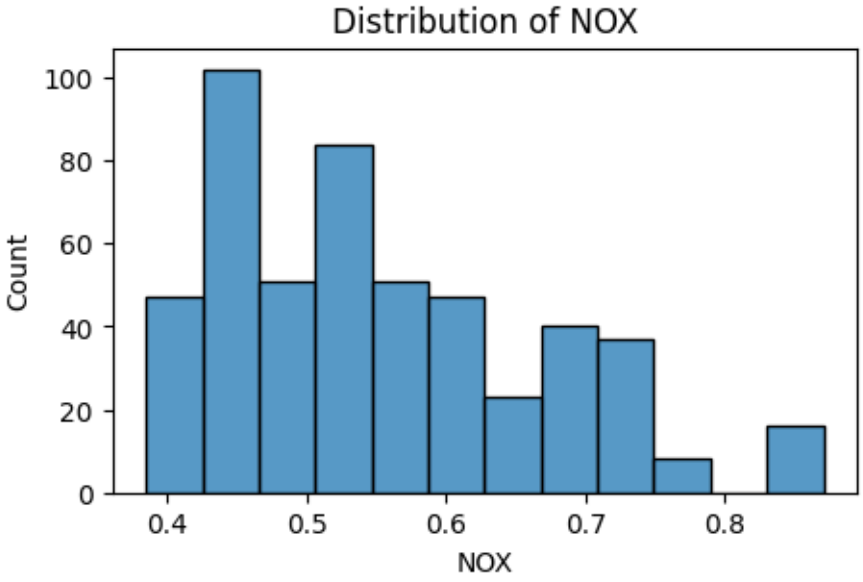
[10]:
```
# Histogram
for column in df.columns:
    plt.figure(figsize=(5, 3))
    sns.histplot(df[column])
    plt.title(f'Distribution of {column}')
    plt.show()
```

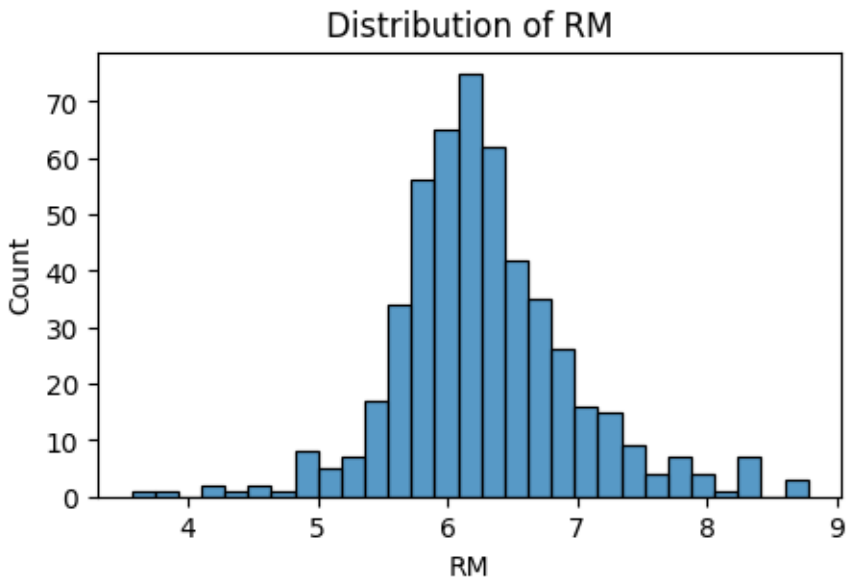


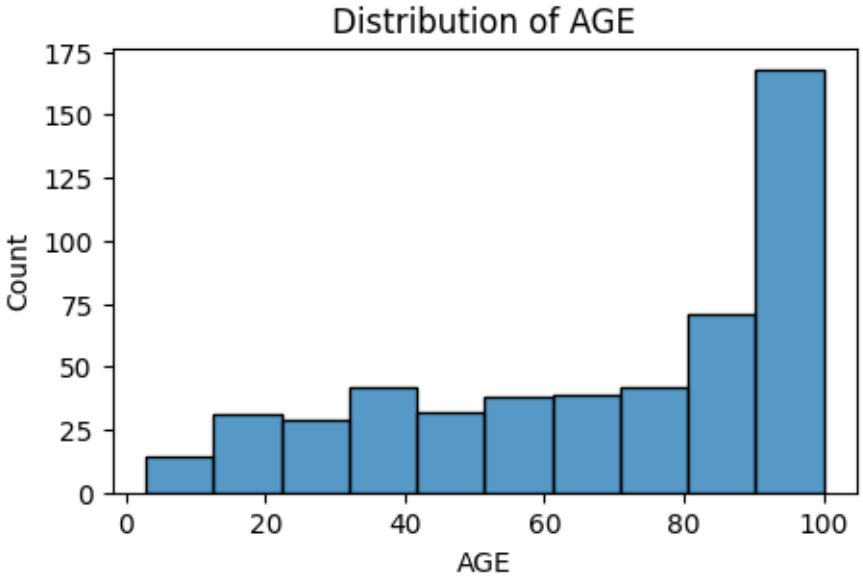

Distribution of ZN
Count
ZN
0
50
100
150
200
250
300
350
0
20
40
60
80
100

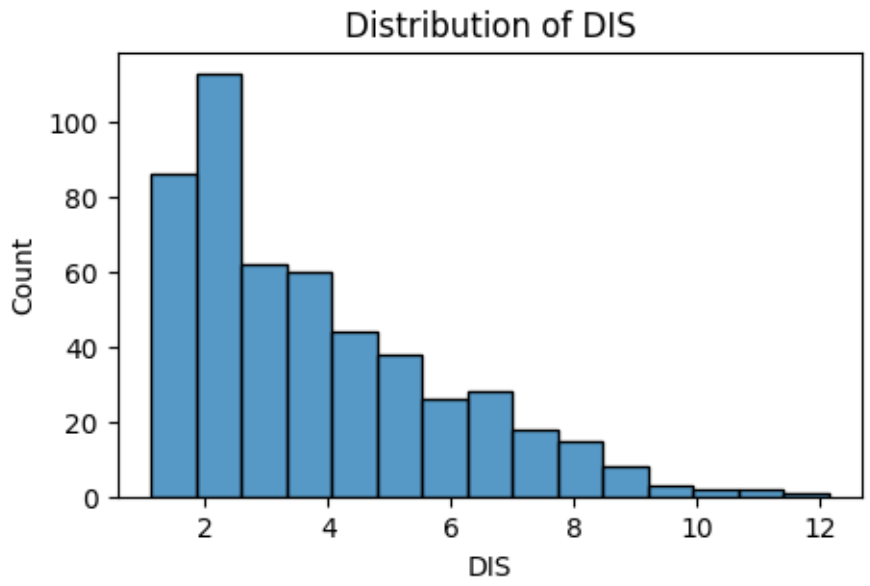

Distribution of INDUS
Count
INDUS
0
20
40
60
80
100
120
0
5
10
15
20
25

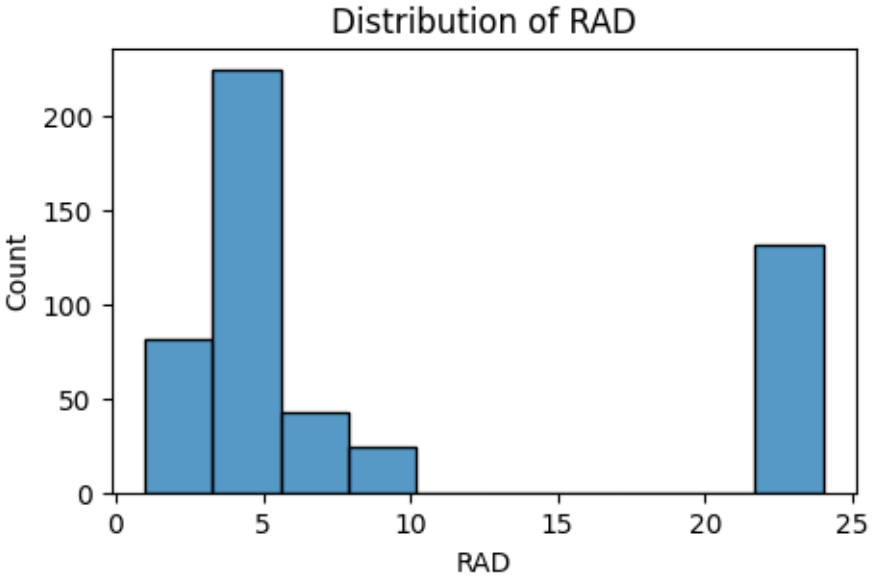

Distribution of CHAS
Count
400
300
200
100
0
0.0
0.2
0.4
0.6
0.8
1.0
CHAS

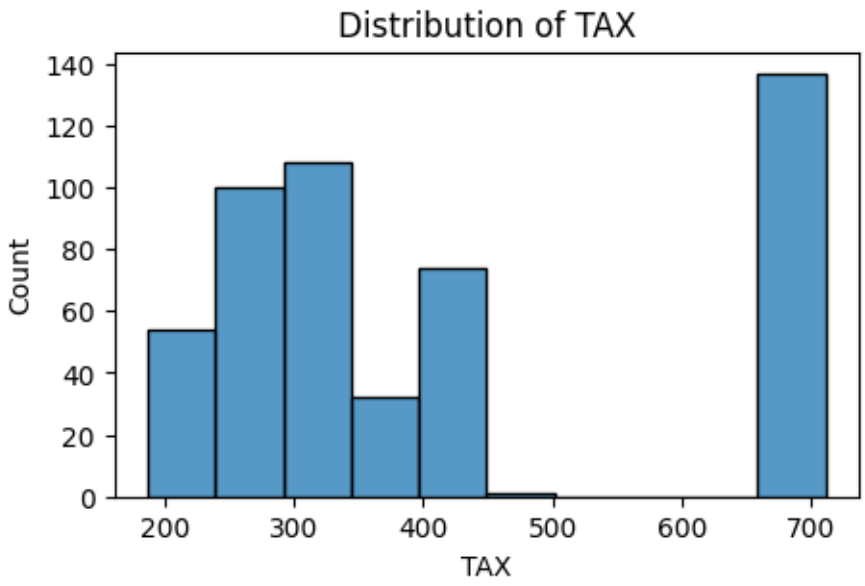

Distribution of NOX
Count
100
80
60
40
20
0
0.4
0.5
0.6
0.7
0.8
NOX

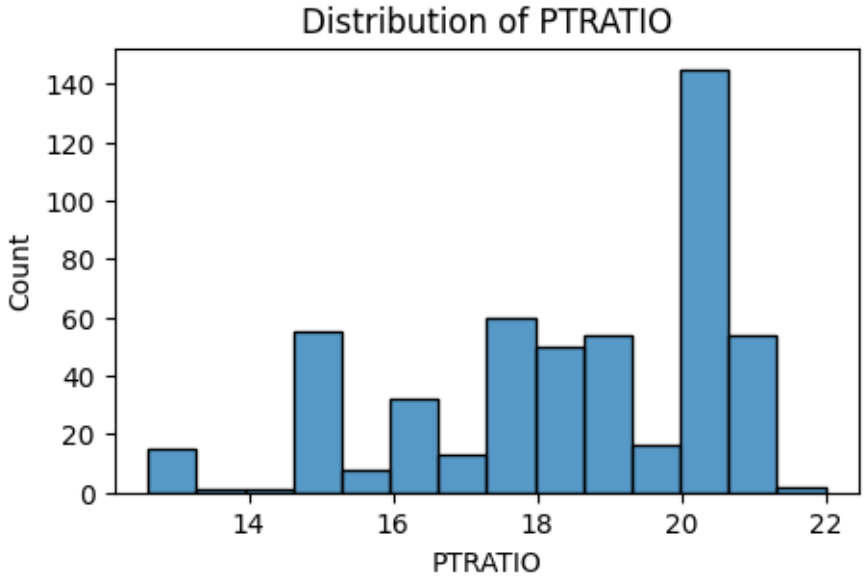

Distribution of RM
Count
RM
0
10
20
30
40
50
60
70
4
5
6
7
8
9

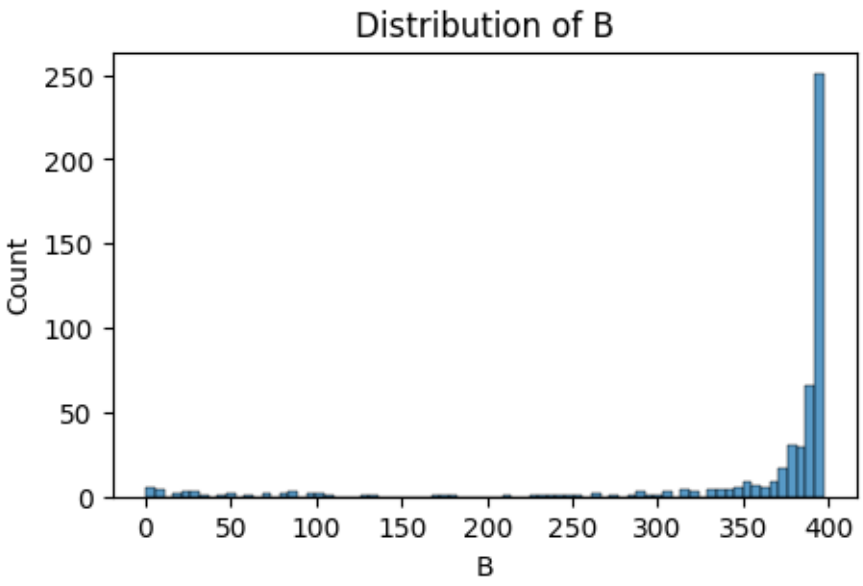

Distribution of AGE
Count
AGE
0
25
50
75
100
125
150
175
0
20
40
60
80
100

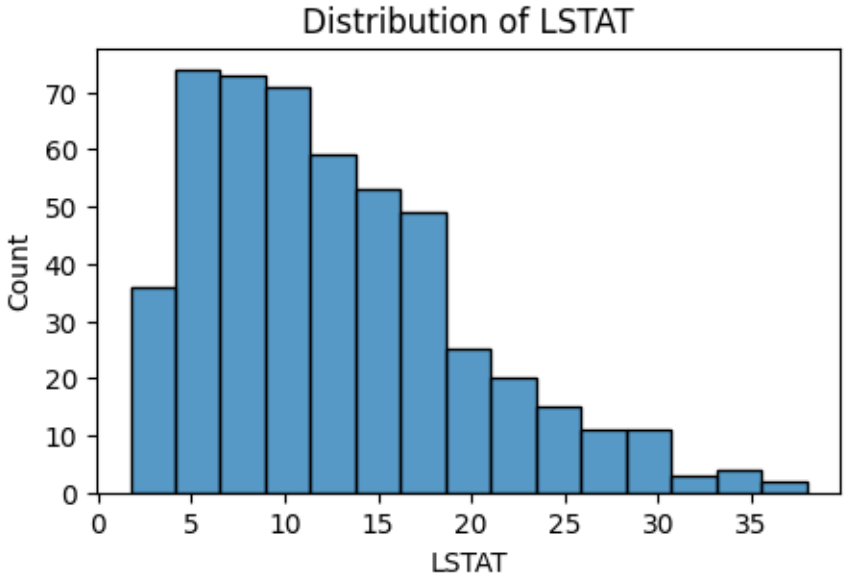

Distribution of DIS
Count
DIS
0
20
40
60
80
100
2
4
6
8
10
12

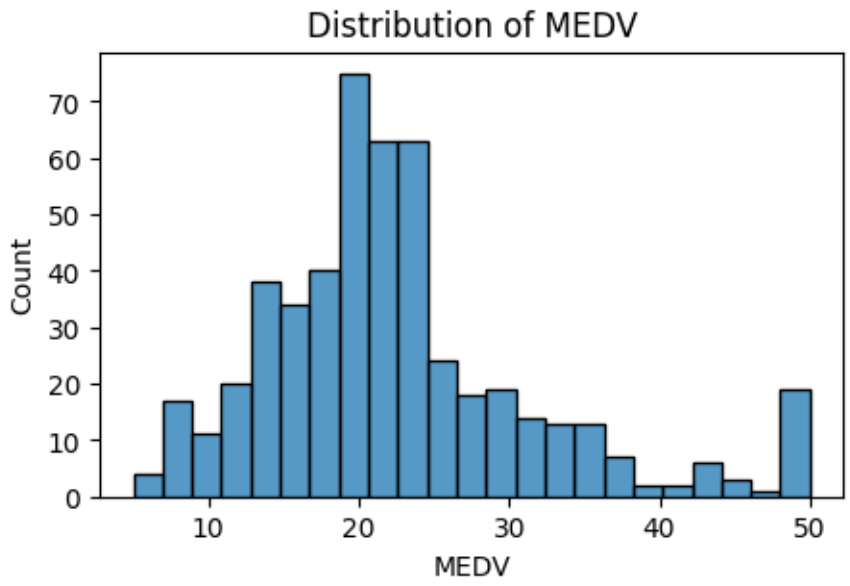

Distribution of RAD
Count
RAD
0
50
100
150
200
0
5
10
15
20
25


Distribution of TAX
Count
TAX
0
20
40
60
80
100
120
140
200
300
400
500
600
700

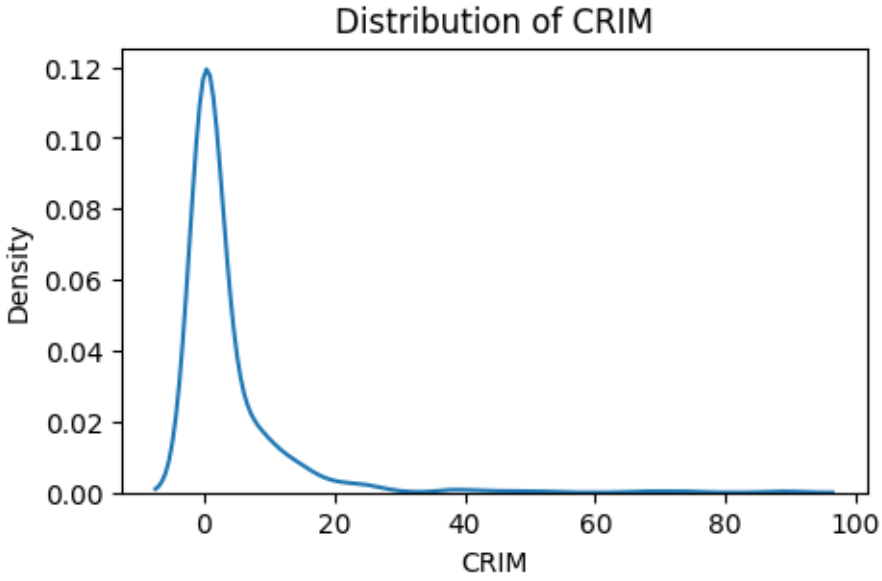

Distribution of PTRATIO
Count
PTRATIO
0
20
40
60
80
100
120
140
14
16
18
20
22

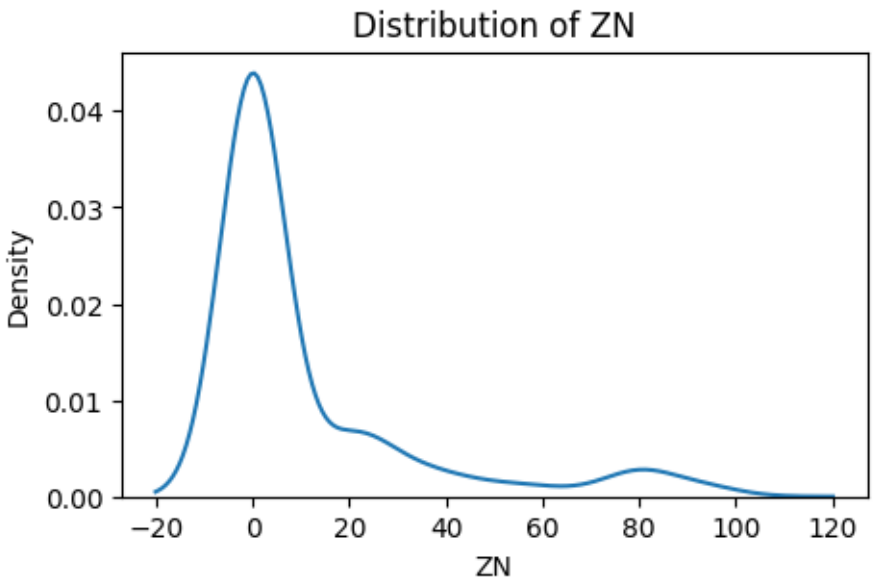

Distribution of B
Count
B
0
50
100
150
200
250
300
350
400

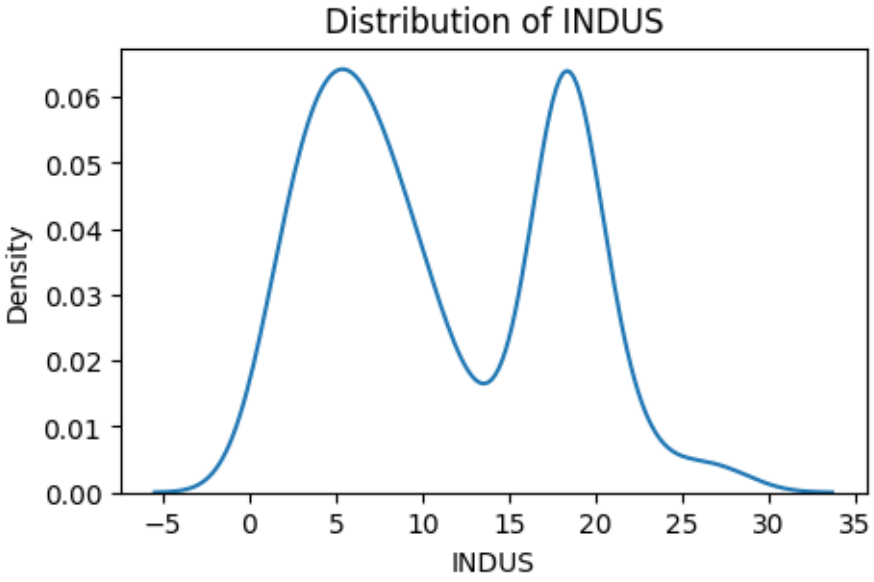

Distribution of LSTAT
Count
LSTAT
0
5
10
15
20
25
30
35

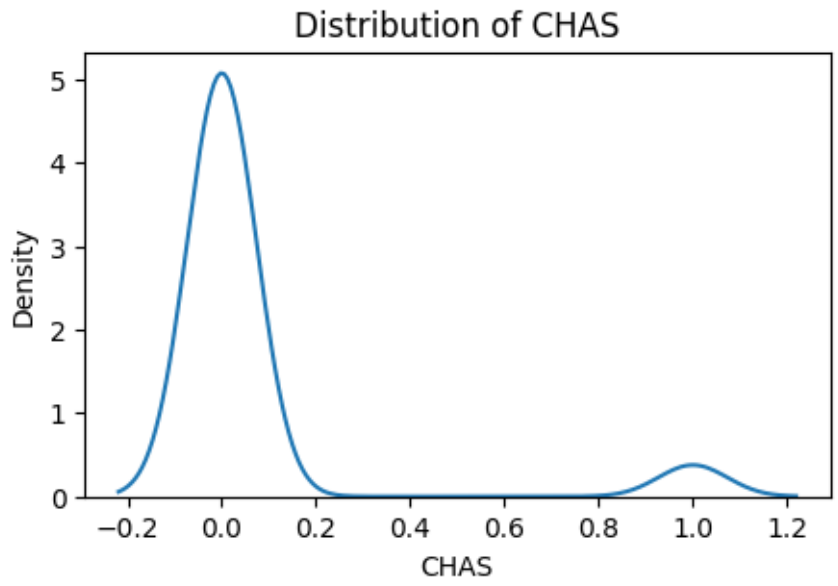


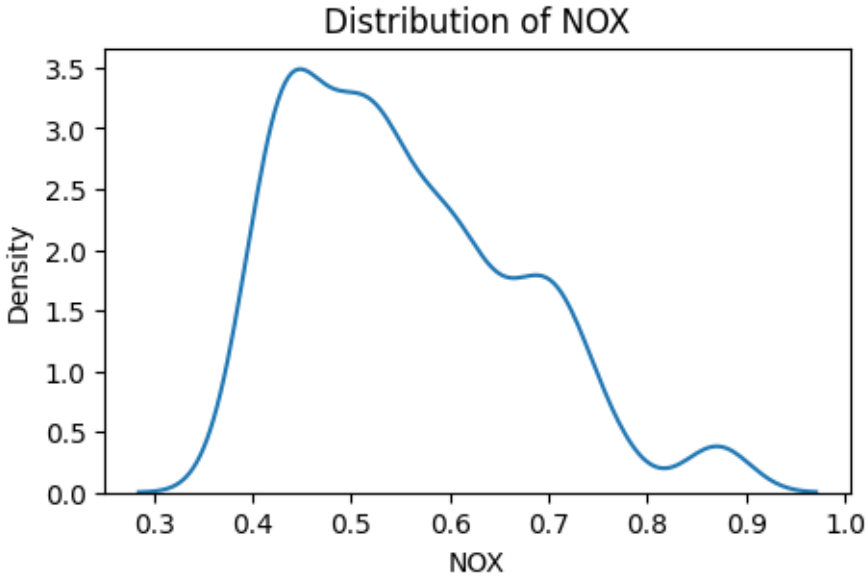
[ ]:
```
# Histogram
for column in df.columns:
    plt.figure(figsize=(5, 3))
    sns.kdeplot(df[column])
    plt.title(f'Distribution of {column}')
    plt.show()
```

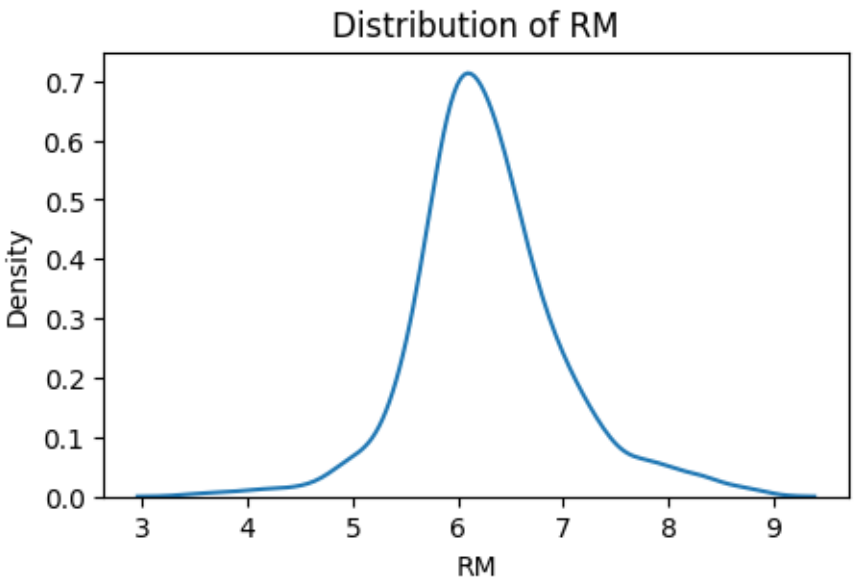



Distribution of ZN
Density
0.04
0.03
0.02
0.01
0.00
−20
0
20
40
60
80
100
120
ZN


Distribution of INDUS
Density
0.06
0.05
0.04
0.03
0.02
0.01
0.00
−5
0
5
10
15
20
25
30
35
INDUS


Distribution of CHAS
Density
CHAS
0
1
2
3
4
5
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2


Distribution of NOX
Density
NOX
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0



[ ]:
```
# Heatmap
corr_matrix = df.corr()
plt.figure(figsize=(10, 5))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.show()
```

[ ]:
```
plt.figure(figsize=(5, 3))
sns.pairplot(df)
plt.show()
```

[ ]:
```
# Handling Outliers
# Capping outliers
df_capped = df.copy()
for column in df.columns:
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    df_capped[column] = df[column].apply(lambda x: upper_bound if x >␣
 ↪upper_bound else (lower_bound if x < lower_bound else x))
print(df_capped.describe())
```

[ ]:
```
plt.figure(figsize=(12, 6))
sns.boxplot(data=df_capped)
plt.title('Boxplot after Capping Outliers')
plt.xticks(rotation=90)
```

```
plt.show()
```