# Regression Project: Boston House Price Prediction

# Marks: 60

Welcome to the project on regression. We will use the **Boston house price dataset** for this project.

---

## Objective

---

The problem at hand is to **predict the housing prices of a town or a suburb based on the features of the locality provided to us**. In the process, we need to **identify the most important features affecting the price of the house**. We need to employ techniques of data preprocessing and build a linear regression model that predicts the prices for the unseen data.

---

## Dataset

---

Each record in the database describes a house in Boston suburb or town. The data was drawn from the Boston Standard Metropolitan Statistical Area (SMSA) in 1970. Detailed attribute information can be found below:

Attribute Information:

- **CRIM:** Per capita crime rate by town
- **ZN:** Proportion of residential land zoned for lots over 25,000 sq.ft.
- **INDUS:** Proportion of non-retail business acres per town
- **CHAS:** Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
- **NOX:** Nitric Oxide concentration (parts per 10 million)
- **RM:** The average number of rooms per dwelling
- **AGE:** Proportion of owner-occupied units built before 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 dollars
- **PTRATIO:** Pupil-teacher ratio by town
- **LSTAT:** % lower status of the population
- **MEDV:** Median value of owner-occupied homes in 1000 dollars

# Importing the necessary libraries

In [46]:
```
# Import Libraries for data manupulation
import pandas as pd
import numpy as np

# Import libraries for data visualization
import matplotlib.pyplot as plt
import seaborn as sns

# Import libraries for data preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler

# Import libraries for data preparation
from sklearn.model_selection import train_test_split

#Import libraries for Model Bulding
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
from statsmodels.formula.api import ols
from sklearn.metrics import mean_squared_error, r2_score

import warnings
warnings.filterwarnings('ignore')
```

## Loading the dataset

In [47]:
```
df = pd.read_csv('Boston.csv')
```

# Data Overview

- Observations
- Sanity checks

In [48]:
```
df.head()
```

Out[48]:

| | CRIM | ZN | INDUS | CHAS | NOX | RM | AGE | DIS | RAD | TAX | PTRATIO | LSTAT | MEDV |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.00632 | 18.0 | 2.31 | 0 | 0.538 | 6.575 | 65.2 | 4.0900 | 1 | 296 | 15.3 | 4.98 | 24.0 |
| **1** | 0.02731 | 0.0 | 7.07 | 0 | 0.469 | 6.421 | 78.9 | 4.9671 | 2 | 242 | 17.8 | 9.14 | 21.6 |
| **2** | 0.02729 | 0.0 | 7.07 | 0 | 0.469 | 7.185 | 61.1 | 4.9671 | 2 | 242 | 17.8 | 4.03 | 34.7 |
| **3** | 0.03237 | 0.0 | 2.18 | 0 | 0.458 | 6.998 | 45.8 | 6.0622 | 3 | 222 | 18.7 | 2.94 | 33.4 |
| **4** | 0.06905 | 0.0 | 2.18 | 0 | 0.458 | 7.147 | 54.2 | 6.0622 | 3 | 222 | 18.7 | 5.33 | 36.2 |

In [49]:
```
df.shape
```
Out[49]:
```
(506, 13)
```

In [50]:
```
df.info()
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 13 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    int64  
 10  PTRATIO  506 non-null    float64
 11  LSTAT    506 non-null    float64
 12  MEDV     506 non-null    float64
dtypes: float64(10), int64(3)
memory usage: 51.5 KB
```

In [51]:
```
df.isnull().sum()
```
Out[51]:
```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
LSTAT      0
MEDV       0
dtype: int64
```

In [52]:
```
df.duplicated().sum()
```
Out[52]:
```
0
```

In [53]:
```
df.describe().T
```

Out[53]:

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **CRIM** | 506.0 | 3.613524 | 8.601545 | 0.00632 | 0.082045 | 0.25651 | 3.677083 | 88.9762 |
| **ZN** | 506.0 | 11.363636 | 23.322453 | 0.00000 | 0.000000 | 0.00000 | 12.500000 | 100.0000 |
| **INDUS** | 506.0 | 11.136779 | 6.860353 | 0.46000 | 5.190000 | 9.69000 | 18.100000 | 27.7400 |
| **CHAS** | 506.0 | 0.069170 | 0.253994 | 0.00000 | 0.000000 | 0.00000 | 0.000000 | 1.0000 |
| **NOX** | 506.0 | 0.554695 | 0.115878 | 0.38500 | 0.449000 | 0.53800 | 0.624000 | 0.8710 |
| **RM** | 506.0 | 6.284634 | 0.702617 | 3.56100 | 5.885500 | 6.20850 | 6.623500 | 8.7800 |
| **AGE** | 506.0 | 68.574901 | 28.148861 | 2.90000 | 45.025000 | 77.50000 | 94.075000 | 100.0000 |
| **DIS** | 506.0 | 3.795043 | 2.105710 | 1.12960 | 2.100175 | 3.20745 | 5.188425 | 12.1265 |
| **RAD** | 506.0 | 9.549407 | 8.707259 | 1.00000 | 4.000000 | 5.00000 | 24.000000 | 24.0000 |
| **TAX** | 506.0 | 408.237154 | 168.537116 | 187.00000 | 279.000000 | 330.00000 | 666.000000 | 711.0000 |
| **PTRATIO** | 506.0 | 18.455534 | 2.164946 | 12.60000 | 17.400000 | 19.05000 | 20.200000 | 22.0000 |
| **LSTAT** | 506.0 | 12.653063 | 7.141062 | 1.73000 | 6.950000 | 11.36000 | 16.955000 | 37.9700 |
| **MEDV** | 506.0 | 22.532806 | 9.197104 | 5.00000 | 17.025000 | 21.20000 | 25.000000 | 50.0000 |

- There are 506 Rows and 13 Columns in the dataset.
- There are no missing values.
- There are no duplicate values.
- All the Columns are numeric datatype.
- There is outlier in Crime rate columns because there is huge difference between mean=3.61 and median=0.25. It also have a large range going from 0 to 89.
- Column ZN,AGE and TAX has a large Spread with Stand deviation of 23, 28, and 168 respectively.
- ZN also have a outlier as it has huge difference between mean=11.36 and median =0

## Exploratory Data Analysis (EDA)

- EDA is an important part of any project involving data.
- It is important to investigate and understand the data better before building a model with it.
- A few questions have been mentioned below which will help you approach the analysis in the right manner and generate insights from the data.
- A thorough analysis of the data, in addition to the questions mentioned below, should be done.

**Questions:**

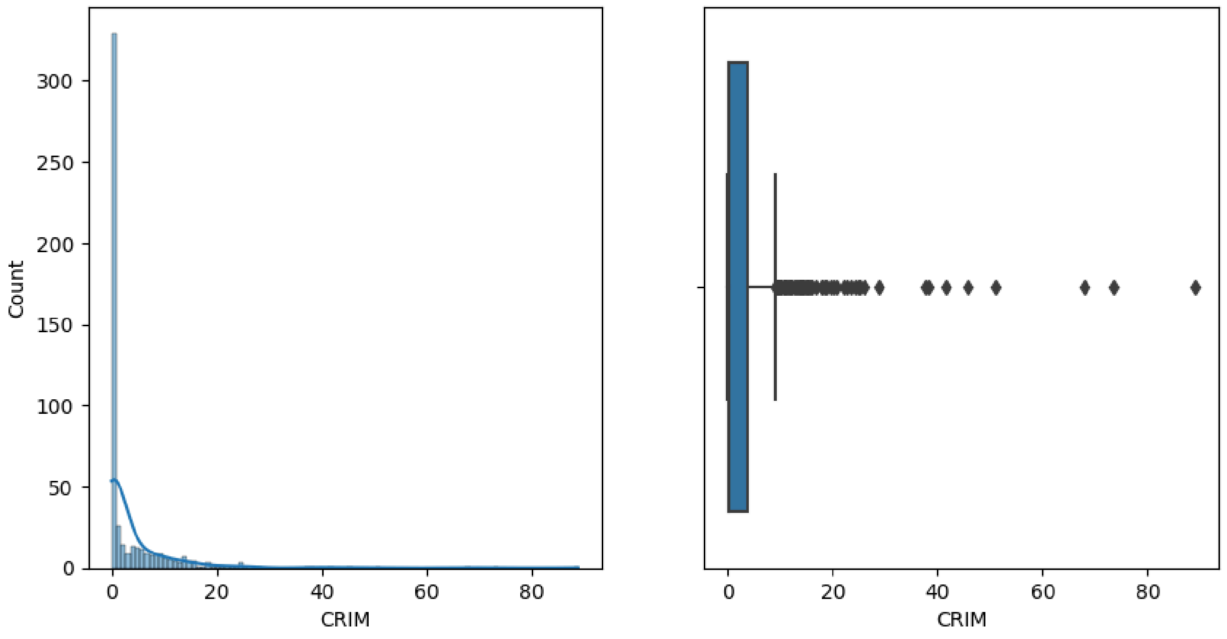
1. What does the distribution of 'MEDV' look like?

2. What can we infer form the correlation heatmap? Is there correlation between the dependent and independent variables?
3. What are all the inferences that can be found by doing univariate analysis for different variables?
4. Do bivariate analysis to visualize the relationship between the features having significant correlations (>= 0.7 or <= -0.7)

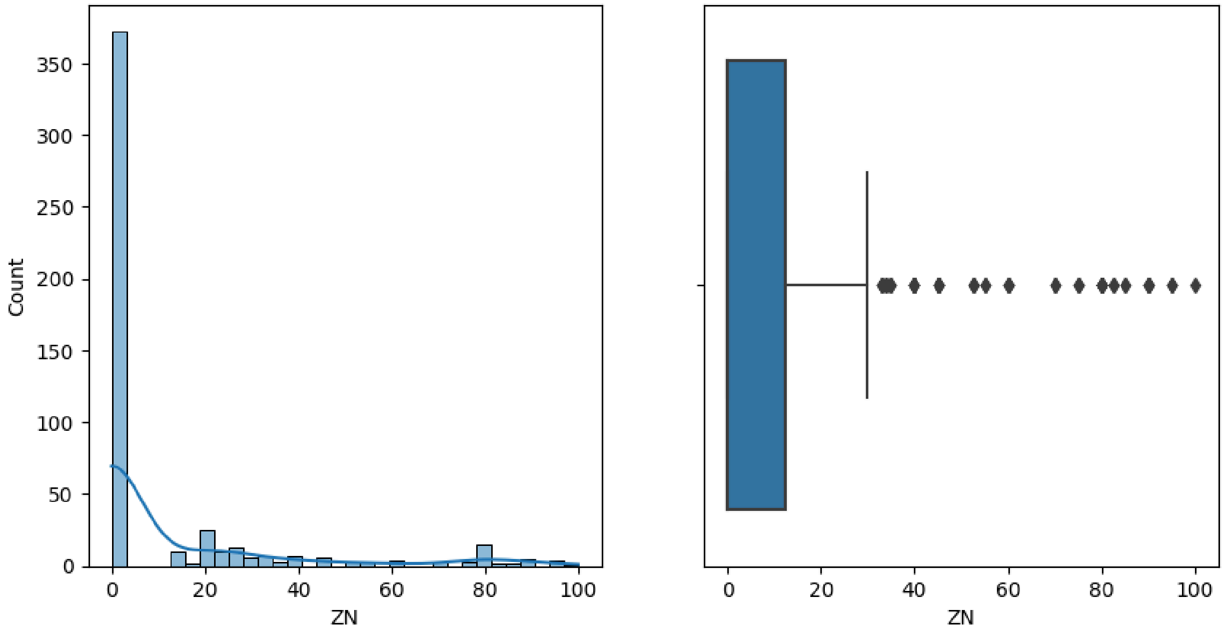
## Univariate Analysis

In [54]:
```
# Visualizing Distribution of all the columns using histogram and kde plot
for column in df.columns:
  plt.figure(figsize=(10,5))
  plt.subplot(1,2,1)
  sns.histplot(data=df, x=column, kde=True)
  print('Skew: ',round(df[column].skew(),2))
  plt.subplot(1,2,2)
  sns.boxplot(data=df, x=column)
  plt.show()
```

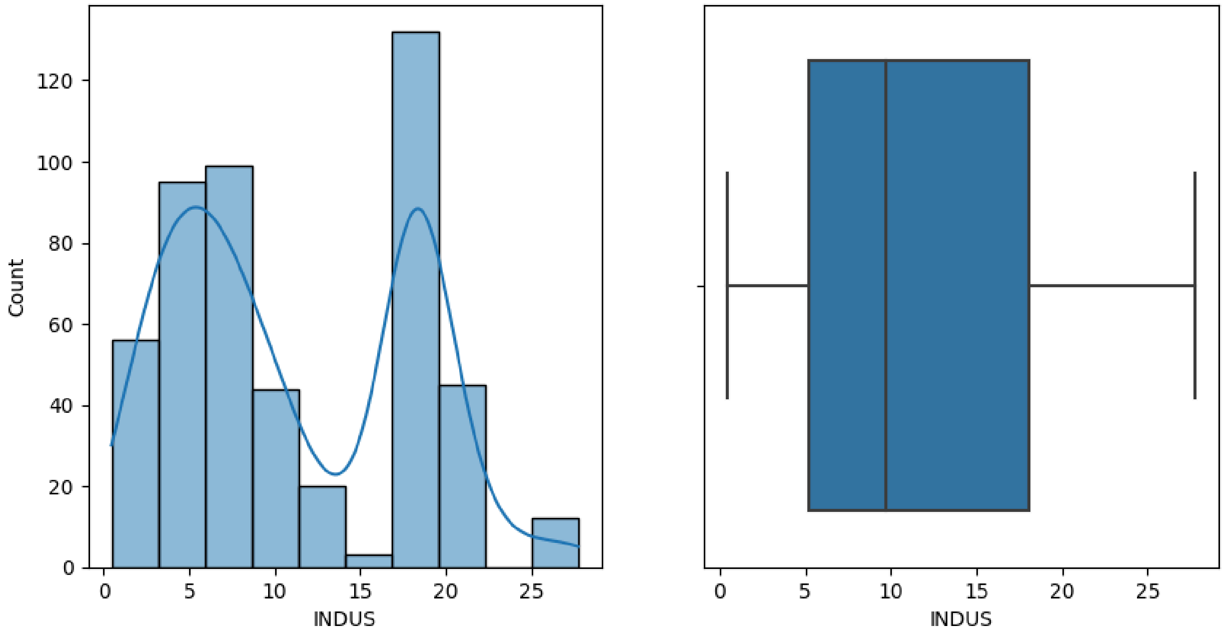
```
Skew:  5.22
```



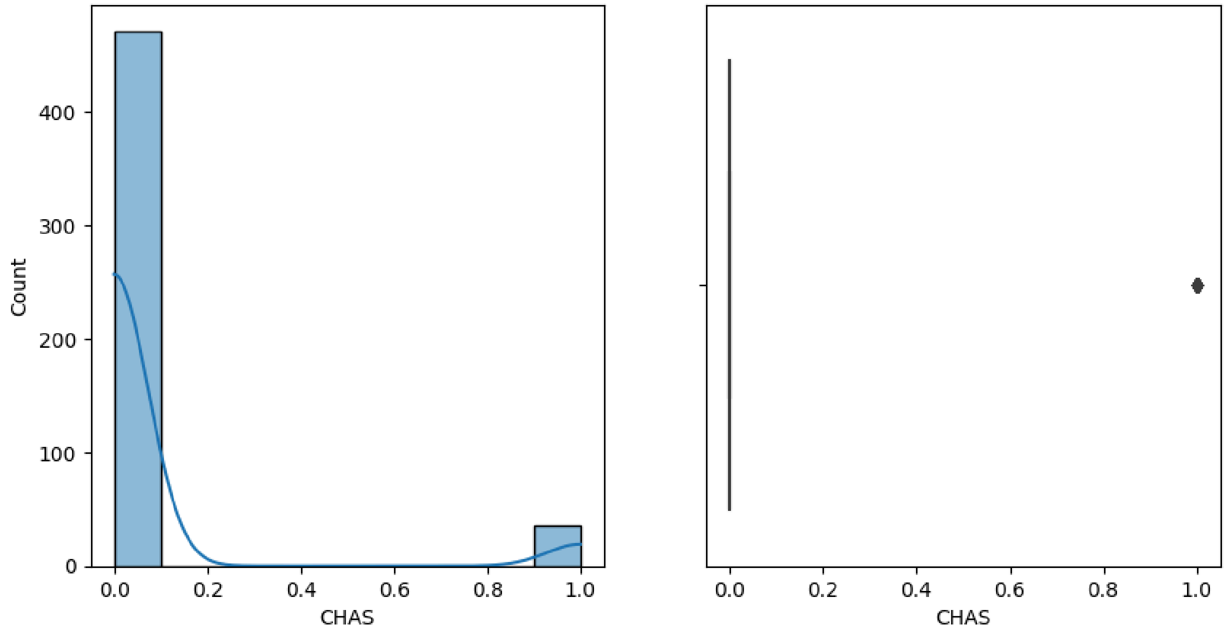
```
Skew:  2.23
```



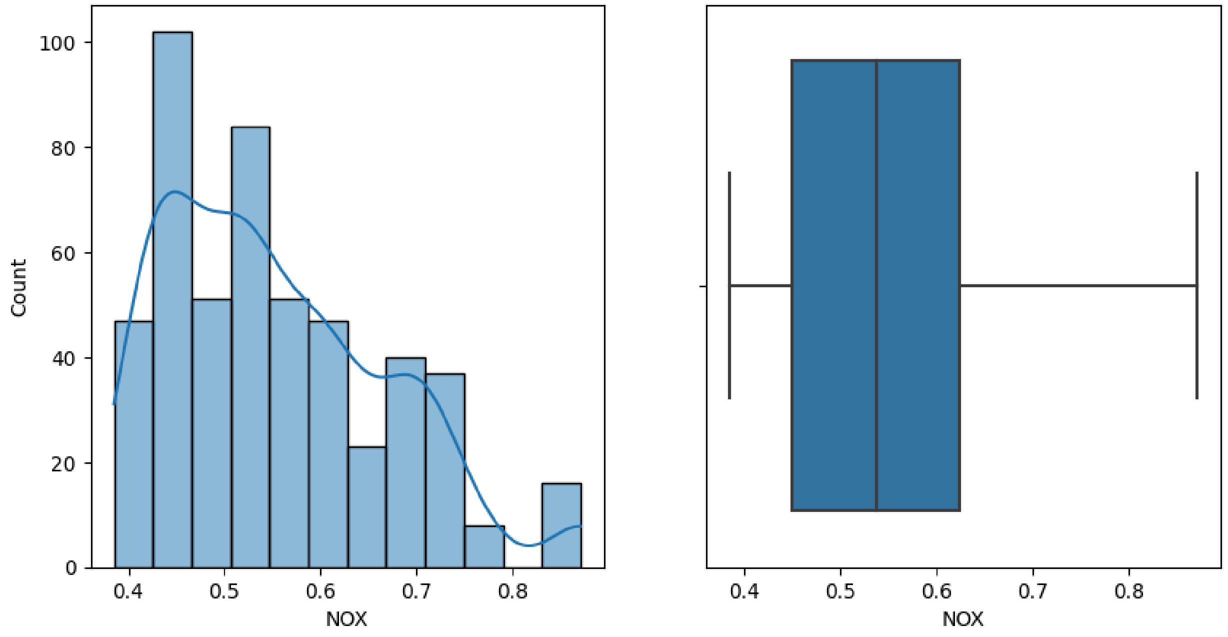
```
Skew:  0.3
```



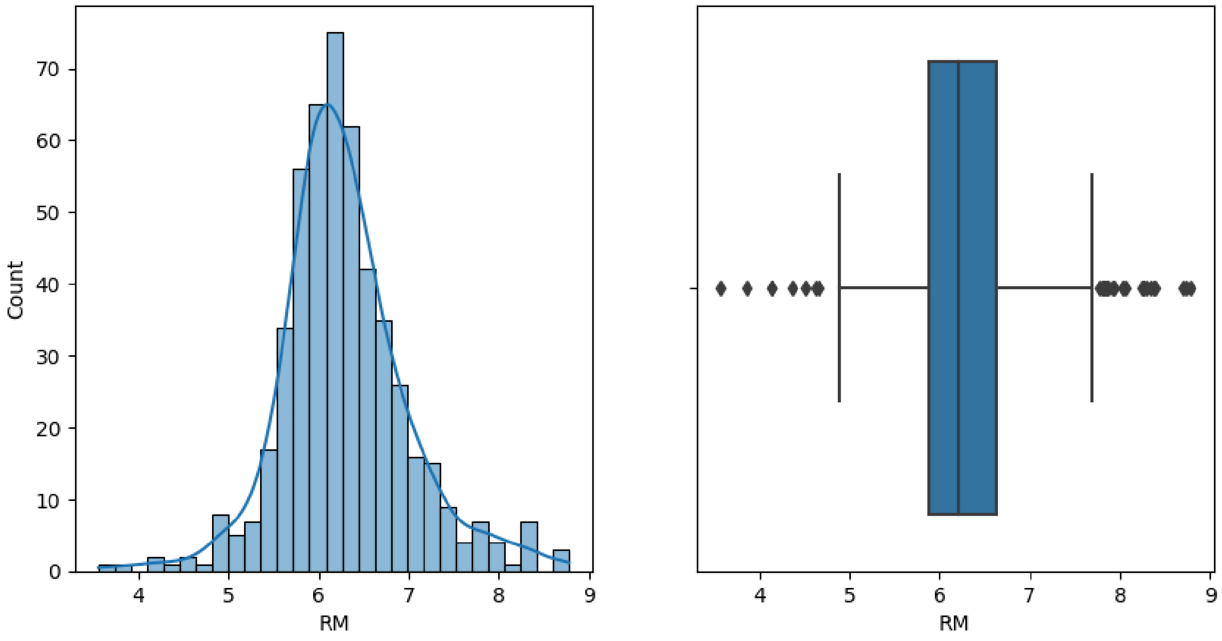
```
Skew:  3.41
```



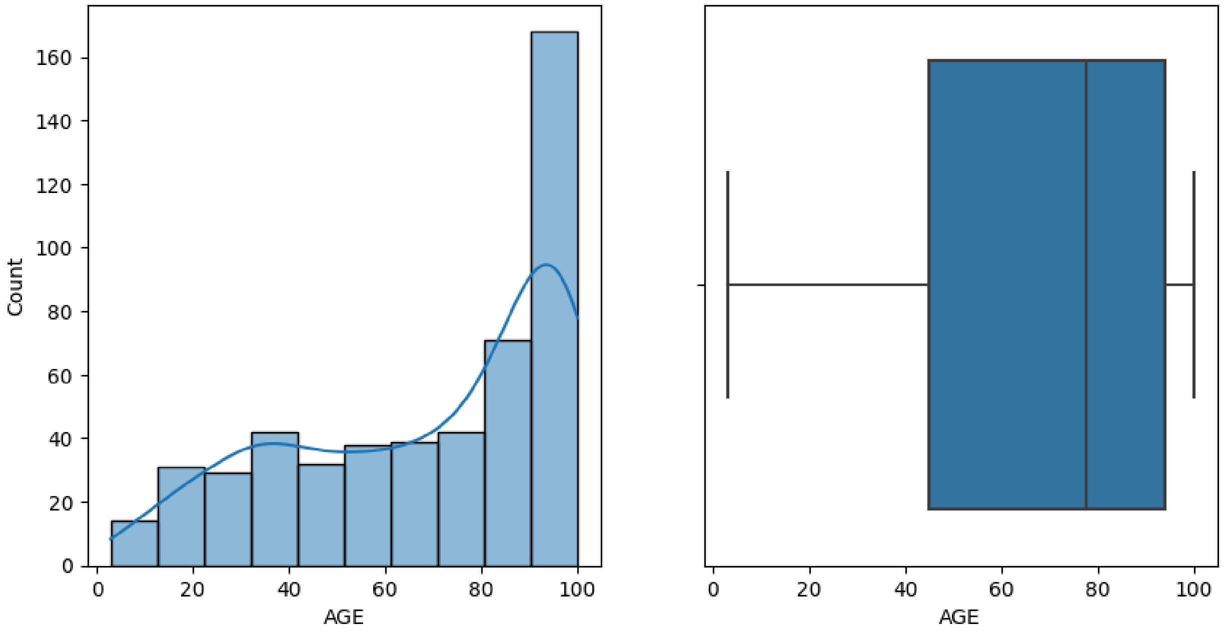
```
Skew:  0.73
```



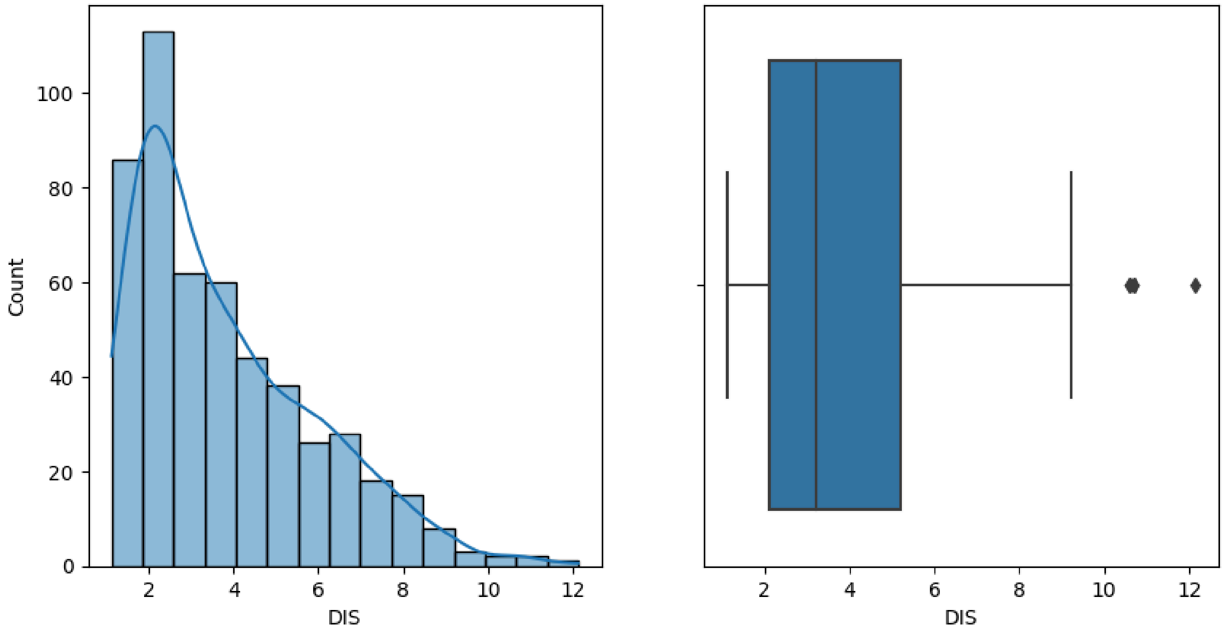
```
Skew:  0.4
```



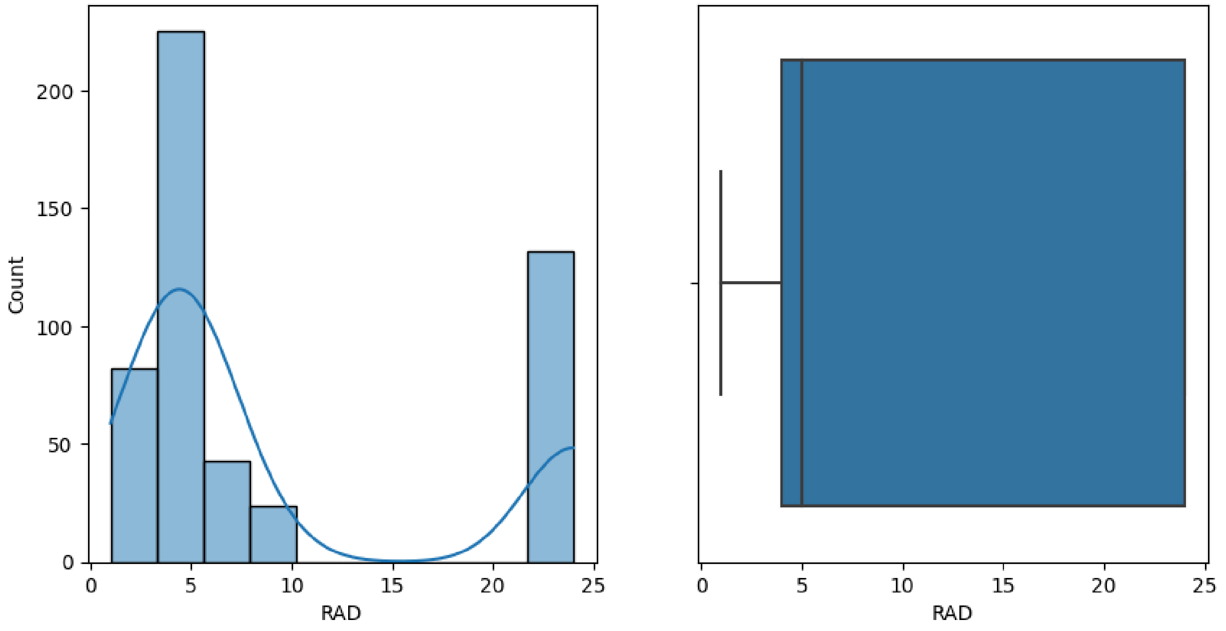
```
Skew:  -0.6
```



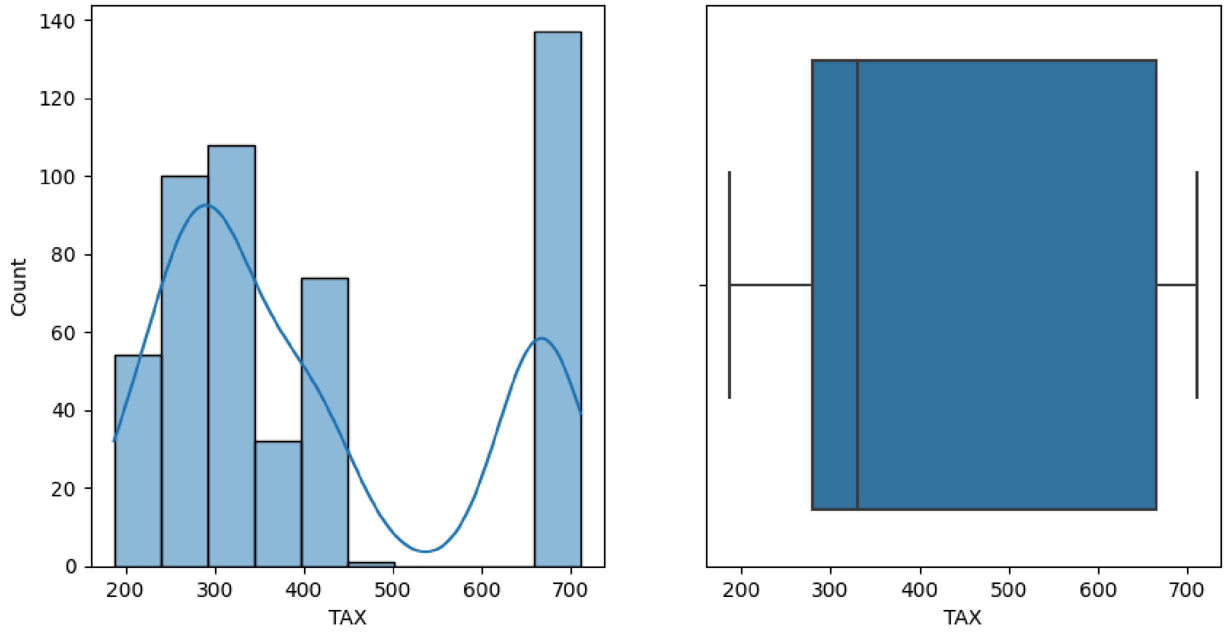
```
Skew:  1.01
```



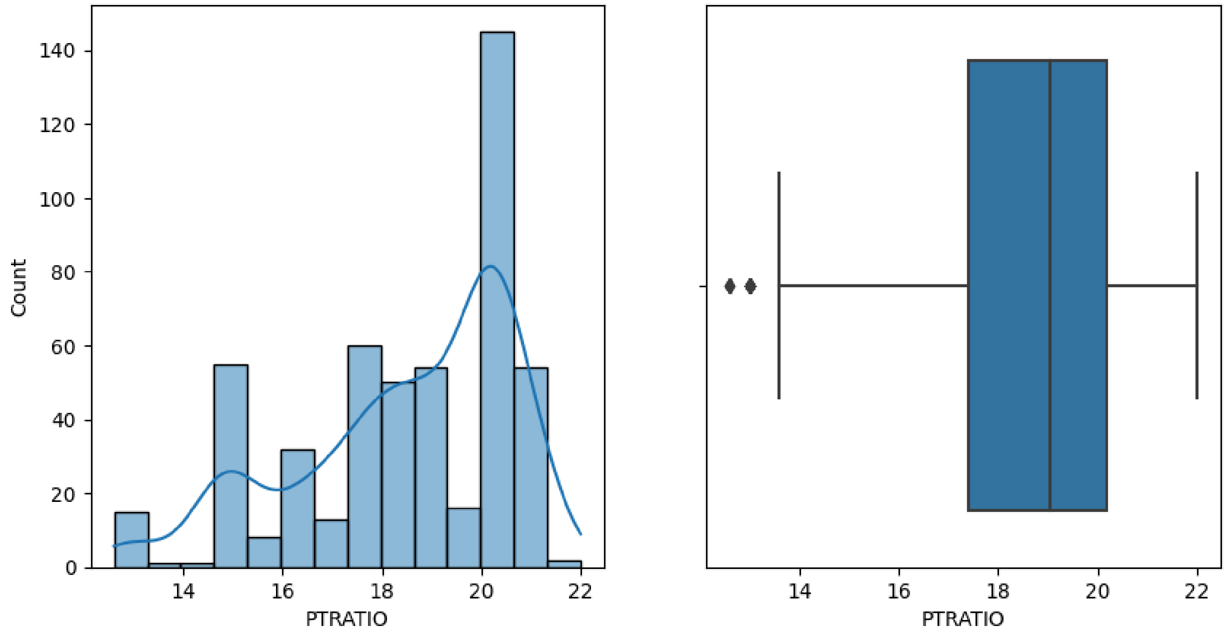
```
Skew:  1.0
```



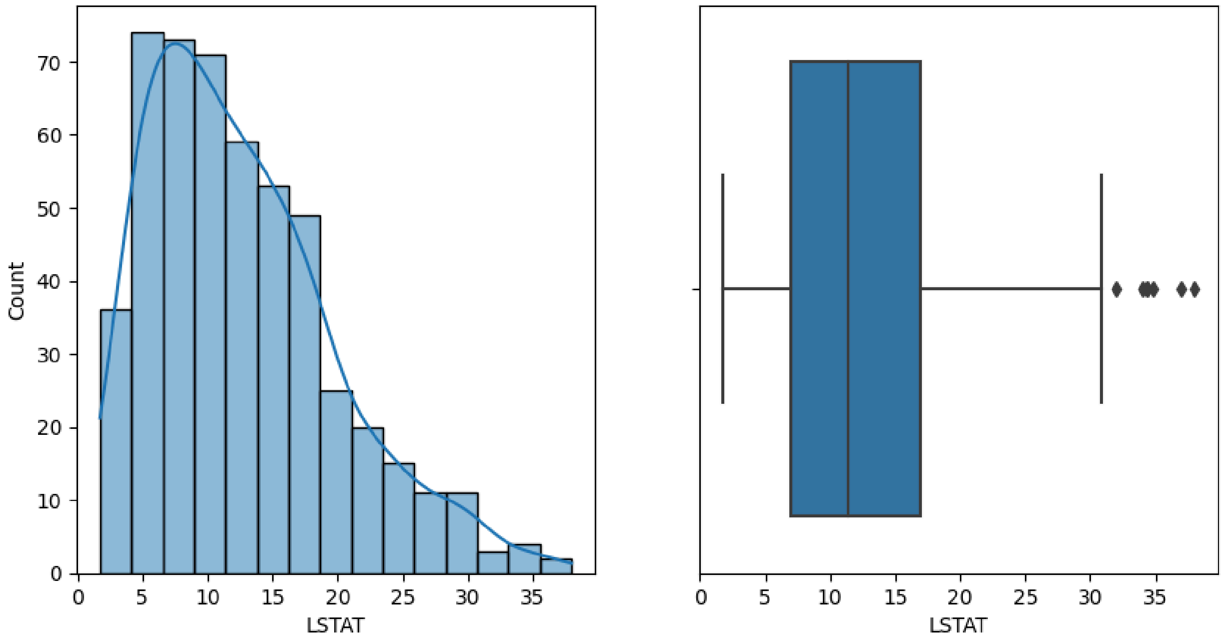
```
Skew:  0.67
```



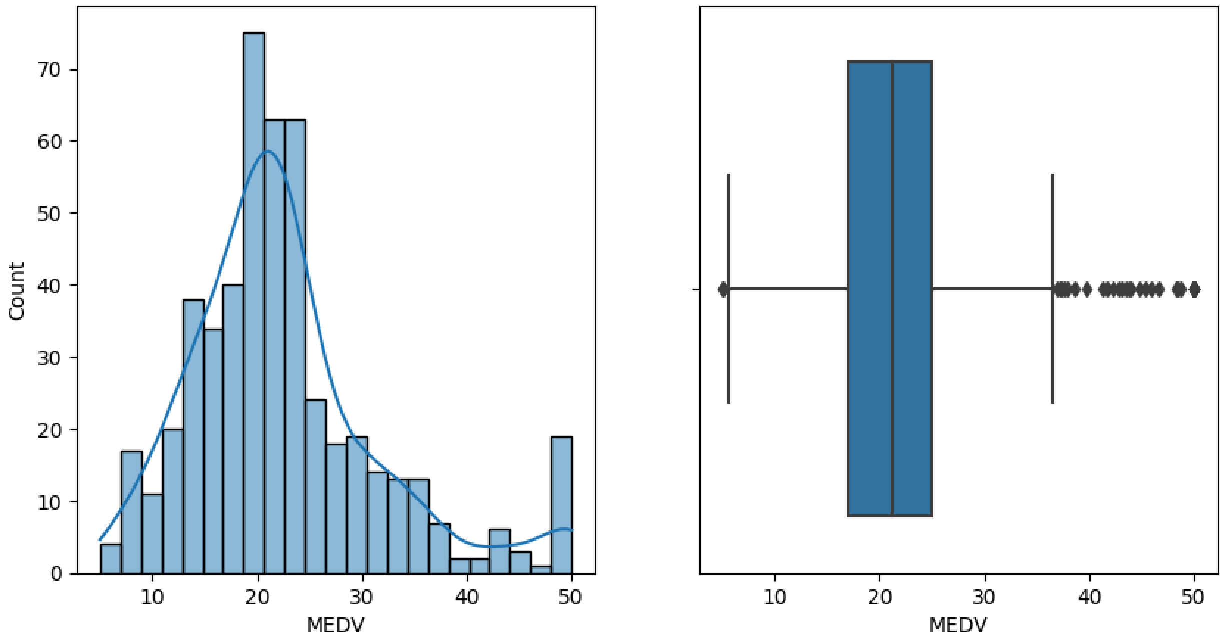
```
Skew:  -0.8
```



```
Skew:  0.91
```



```
Skew:  1.11
```



In [55]:
```python
for column in df.columns:
    print(f'{column} Skew: {round(df[column].skew(), 2)}')
```

```
CRIM Skew: 5.22
ZN Skew: 2.23
INDUS Skew: 0.3
CHAS Skew: 3.41
NOX Skew: 0.73
RM Skew: 0.4
AGE Skew: -0.6
DIS Skew: 1.01
RAD Skew: 1.0
TAX Skew: 0.67
PTRATIO Skew: -0.8
LSTAT Skew: 0.91
MEDV Skew: 1.11
```

1. CRIM Skew: 5.22 (Highly positively skewed)

   > Insight: The distribution of crime rates is highly skewed to the right, indicating that most areas have low crime rates, but there are a few areas with extremely high crime rates.

2. ZN Skew: 2.23 (Positively skewed)

   > Insight: The distribution of the proportion of residential land zoned for large lots is positively skewed, suggesting that most areas have a low proportion of such land, but there are some areas with higher proportions.

3. INDUS Skew: 0.3 (Approximately symmetric)

   > Insight: The distribution of the proportion of non-retail business acres per town is close to symmetric, with a slight right skew. Most areas have a moderate proportion of non-retail business acres.

4. CHAS Skew: 3.41 (Highly positively skewed)

   > Insight: The distribution of the Charles River dummy variable is highly skewed to the right, indicating that a majority of areas do not border the river, but there are a few areas that do.

5. NOX Skew: 0.73 (Positively skewed)

   > Insight: The distribution of nitric oxides concentration is positively skewed, indicating that most areas have lower concentrations, but there are some areas with higher concentrations.

6. RM Skew: 0.4 (Positively skewed)

   > Insight: The distribution of average rooms per dwelling is slightly positively skewed, suggesting that most areas have a moderate number of rooms, but there are some areas with more rooms.

7. AGE Skew: -0.6 (Negatively skewed)

   > Insight: The distribution of the proportion of owner-occupied units built before 1940 is negatively skewed, indicating that most areas have newer buildings, but there are some areas with older buildings.

8. DIS Skew: 1.01 (Positively skewed)

   > Insight: The distribution of weighted distances to employment centers is positively skewed, suggesting that most areas are closer to employment centers, but there are some areas that are farther away.

9. RAD Skew: 1.0 (Positively skewed)

> Insight: The distribution of the index of accessibility to radial highways is positively skewed, indicating that most areas have lower accessibility, but there are some areas with higher accessibility.

10. TAX Skew: 0.67 (Positively skewed)

> Insight: The distribution of the full-value property tax rate is positively skewed, suggesting that most areas have lower tax rates, but there are some areas with higher rates.

11. PTRATIO Skew: -0.8 (Negatively skewed)

> Insight: The distribution of pupil-teacher ratio is negatively skewed, indicating that most areas have a higher ratio, but there are some areas with lower ratios.

12. LSTAT Skew: 0.91 (Positively skewed)

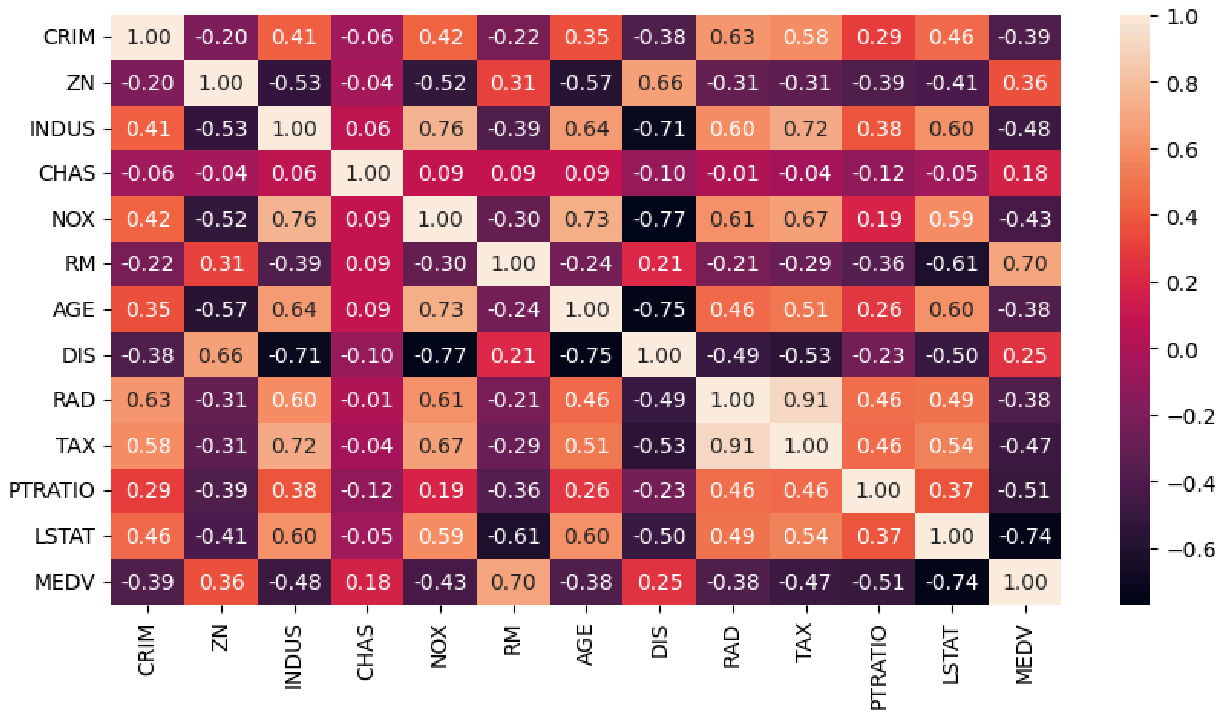
> Insight: The distribution of the percentage of lower status of the population is positively skewed, suggesting that most areas have a lower percentage of lower-status population, but there are some areas with higher percentages.

13. MEDV Skew: 1.11 (Positively skewed)

> Insight: The distribution of median value of owner-occupied homes is positively skewed, indicating that most areas have lower median home values, but there are some areas with higher values.

## Bivariate Analysis

In [56]:
```
# Heat map for correlation analysis

plt.figure(figsize = (10, 5))

sns.heatmap(df.corr(), annot = True, fmt = '0.2f')

plt.show()
```



**Observation:**

- We can see a high positive correlation among the following variables:

1. NOX and INDUS
2. TAX and INDUS
3. AGE and NOX
4. RM and MEDV
5. RAD and TAX

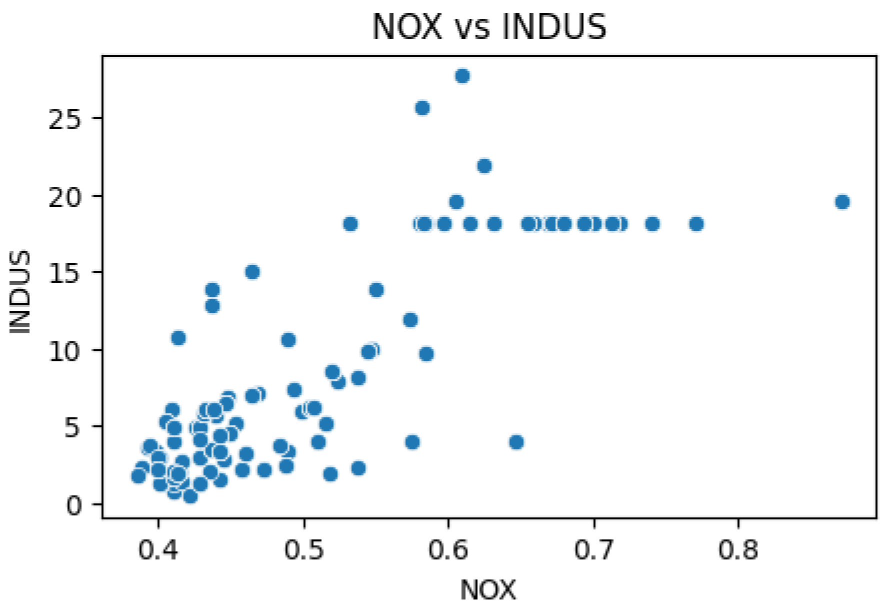
- We can see a high negative correlation among the following variables:

1. DIS and INDUS
2. DIS and NOX
3. DIS and AGE
4. LSTAT and MEDV

## Bivariate Analysis

In [57]:
```
plt.figure(figsize = (5,3))
plt.title('NOX vs INDUS')
print(df[['NOX','INDUS']].corr())
sns.scatterplot(data=df, x='NOX', y='INDUS')
plt.show()
```

```
            NOX     INDUS
NOX    1.000000  0.763651
INDUS  0.763651  1.000000
```

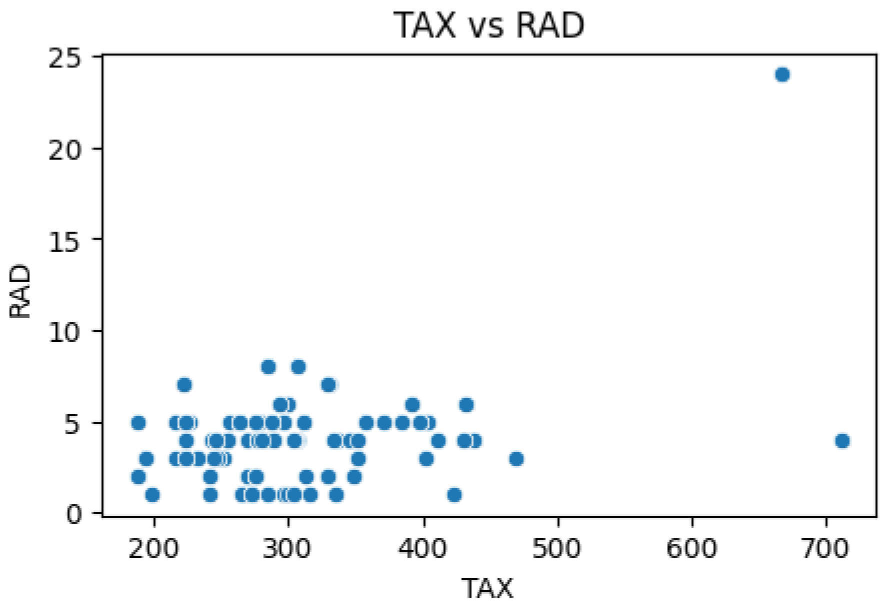


**Observation:**

- INDUS represents the proportion of land used for non-retail businesses, which could include industrial facilities. The positive correlation between INDUS and NOX implies that areas with more industrial land use are associated with higher levels of nitric oxide emissions. Nitric oxides are common pollutants associated with industrial activities.

In [58]:
```
plt.figure(figsize = (5,3))
print(df[['TAX','RAD']].corr())
plt.title('TAX vs RAD')
sns.scatterplot(data=df, x='TAX', y='RAD')
plt.show()
```

```
          TAX       RAD
TAX  1.000000  0.910228
RAD  0.910228  1.000000
```

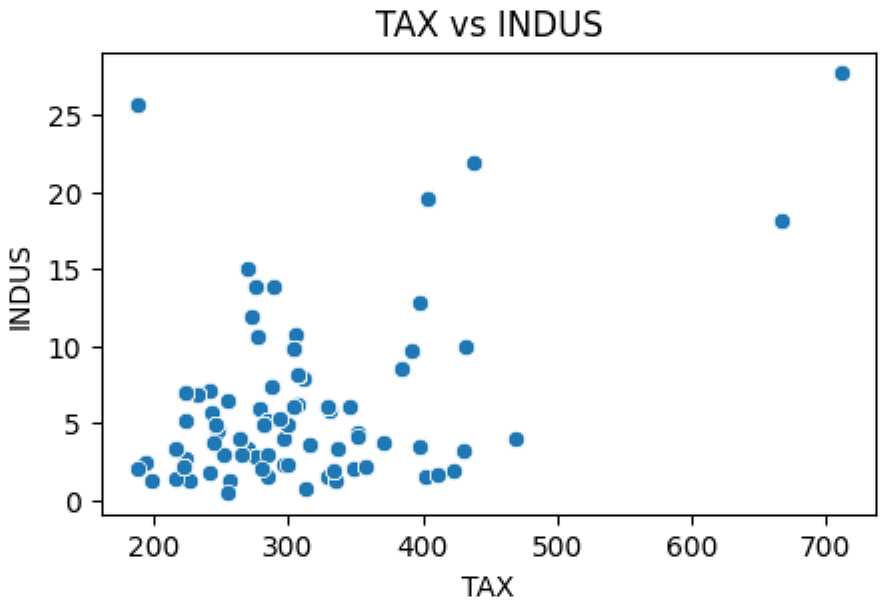


**Observation:**

- A positive correlation of 0.91 between the property tax rate (TAX) and index of accessibility to radial highways (RAD) implies that areas with higher accessibility to radial highways tend to have higher property tax rates

In [59]:
```
plt.figure(figsize = (5,3))
print(df[['TAX','INDUS']].corr())
plt.title('TAX vs INDUS')
sns.scatterplot(data=df, x='TAX', y='INDUS')
plt.show()
```

```
             TAX    INDUS
TAX      1.00000  0.72076
INDUS    0.72076  1.00000
```

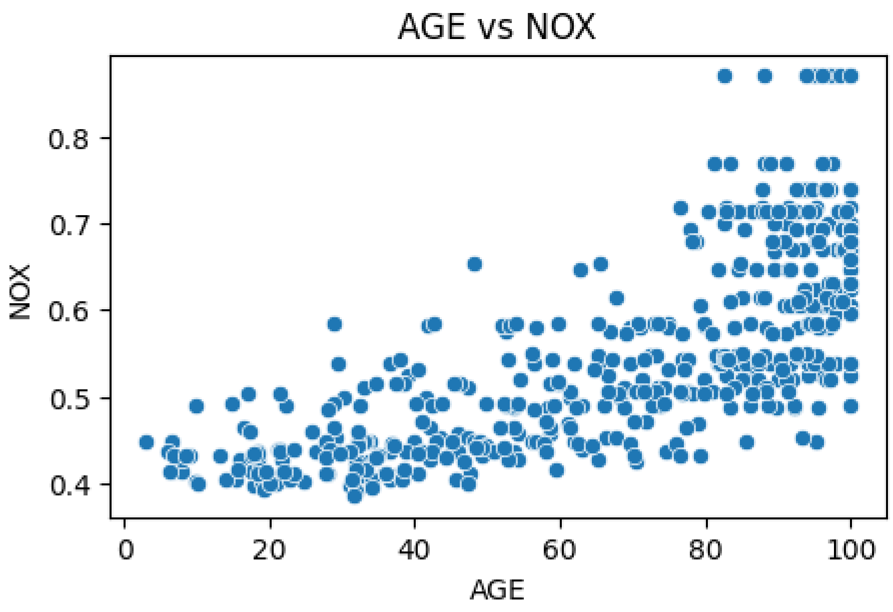


**Observation:**

- The positive correlation of 0.72 between the property tax rate (TAX) and the proportion of non-retail business acres per town (INDUS) implies that areas with a higher proportion of non-retail business acres tend to have higher property tax rates.

In [60]:
```
plt.figure(figsize = (5,3))
print(df[['AGE','NOX']].corr())
plt.title('AGE vs NOX')
sns.scatterplot(data=df, x='AGE', y='NOX')
plt.show()
```

```
           AGE      NOX
AGE    1.00000  0.73147
NOX    0.73147  1.00000
```

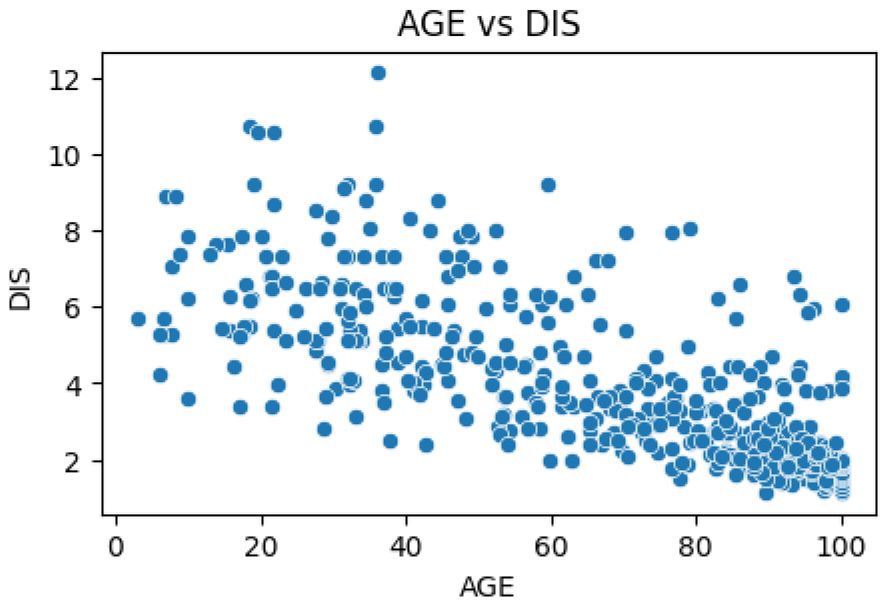


**Observation:**

- A positive correlation of 0.73 between the proportion of owner-occupied units built prior to 1940 (AGE) and nitric oxides concentration (NOX) indicates that areas with a higher proportion of older housing units tend to have higher concentrations of nitric oxides.
- Older houses may be constructed with materials that contribute to higher emissions. For example, buildings with older heating systems or less energy-efficient.
- There could be other reasons too like this houses may be located near major highways, industrial zones, Major employment centers.

In [61]:
```
plt.figure(figsize = (5,3))
print(df[['AGE','DIS']].corr())
plt.title('AGE vs DIS')
sns.scatterplot(data=df, x='AGE', y='DIS')
plt.show()
```

```
          AGE       DIS
AGE  1.000000 -0.747881
DIS -0.747881  1.000000
```

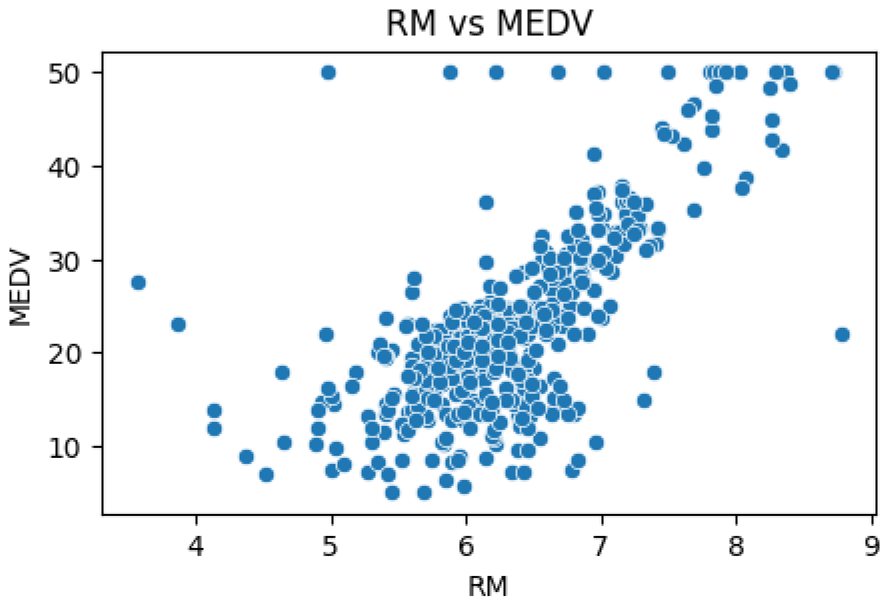


**Observation:**

- A negative correlation of -0.74 between the proportion of owner-occupied units built prior to 1940 (AGE) and weighted distances to employment centers (DIS) indicates that areas with a higher proportion of older housing units tend to be located farther away from employment centers.

In [62]:
```
plt.figure(figsize = (5,3))
print(df[['RM','MEDV']].corr())
plt.title('RM vs MEDV')
sns.scatterplot(data=df, x='RM', y='MEDV')
plt.show()
```

```
           RM      MEDV
RM    1.00000  0.69536
MEDV  0.69536  1.00000
```

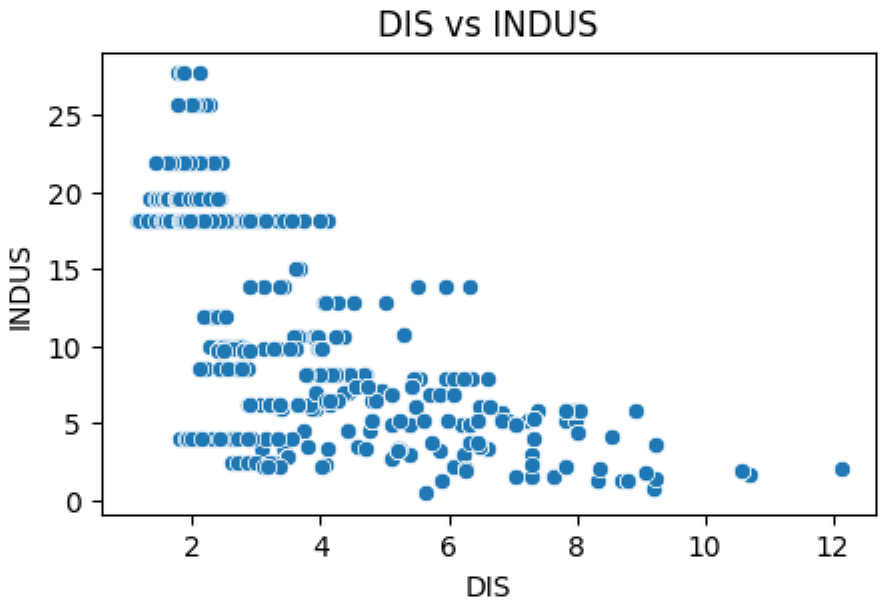


**Observation:**

- A positive correlation of 0.69 between the average number of rooms per dwelling (RM) and the median value of owner-occupied homes (MEDV) indicates that, on average, homes with a higher number of rooms tend to have higher median values.

In [63]:
```
plt.figure(figsize = (5,3))
print(df[['DIS','INDUS']].corr())
plt.title('DIS vs INDUS')
sns.scatterplot(data=df, x='DIS', y='INDUS')
plt.show()
```

```
            DIS     INDUS
DIS    1.000000 -0.708027
INDUS -0.708027  1.000000
```

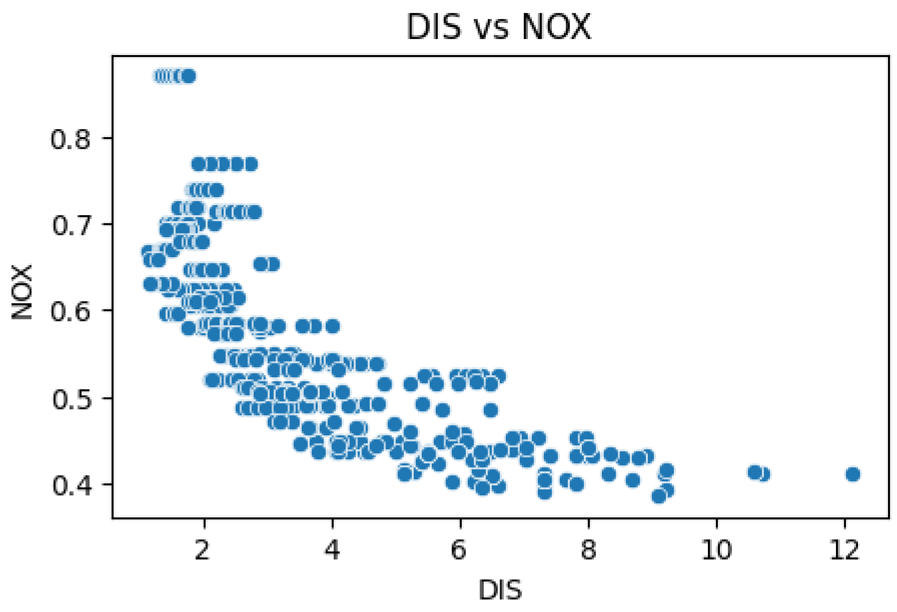


**Observation:**

- A negative correlation of -0.7 between the proportion of non-retail business acres per town (INDUS) and weighted distances to employment centers (DIS) indicates that areas with a higher proportion of non-retail business acres (potentially industrial zones) tend to be located farther away from employment centers.

In [64]:
```
plt.figure(figsize = (5,3))
print(df[['DIS','NOX']].corr())
plt.title('DIS vs NOX')
sns.scatterplot(data=df, x='DIS', y='NOX')
plt.show()
```

```
         DIS      NOX
DIS  1.00000 -0.76923
NOX -0.76923  1.00000
```

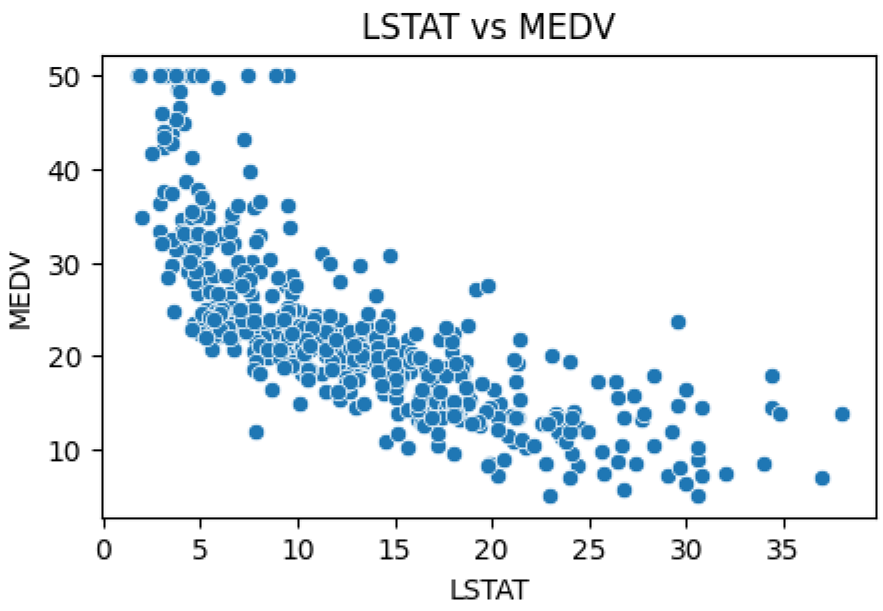


**Observation:**

- A negative correlation of -0.76 between weighted distances to employment centers (DIS) and nitric oxides concentration (NOX) indicates that areas closer to employment centers (lower DIS values) tend to have higher concentrations of nitric oxides
- Areas closer to employment centers may experience higher levels of traffic, leading to increased emissions of pollutants like nitric oxides.

In [65]:
```
plt.figure(figsize = (5,3))
print(df[['LSTAT','MEDV']].corr())
plt.title('LSTAT vs MEDV')
sns.scatterplot(data=df, x='LSTAT', y='MEDV')
plt.show()
```

```
          LSTAT      MEDV
LSTAT  1.000000 -0.737663
MEDV  -0.737663  1.000000
```



**Observation:**

- A negative correlation of -0.73 between the percentage of lower status of the population (LSTAT) and the median value of owner-occupied homes (MDEV) indicates that areas with a higher percentage of lower-status population tend to have lower median home values.

**Summary:**

- Areas with a higher proportion of non-retail business and Areas closer to employment centers have higher concentrations of nitric oxides.
- Areas with a higher proportion of older housing and Areas with a higher proportion of non-retail business tend to be located farther away from employment centers.
- Higher property tax rates are positively correlated with greater accessibility to radial highways
- Areas with a higher percentage of lower-status population tend to have lower median home values
- Homes with a higher average number of rooms tend to have higher median values.

**Important Note:**

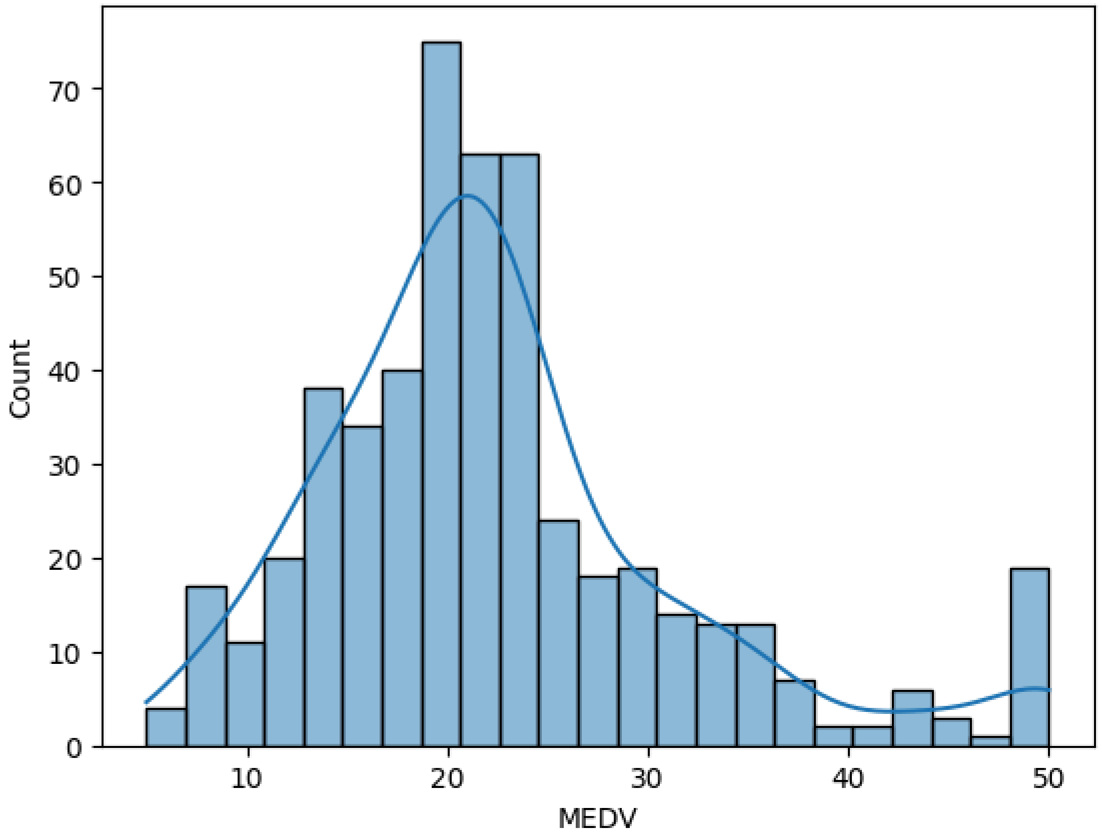
- High correaltion between certain inpendent variable indicates Multicolinearity.
- Target variable MEDV is right-skewed. So we need to do logarithmic transformation of it. Which can make distribution more symmetric.

## Data Preprocessing

- Missing value treatment
- Log transformation of dependent variable if skewed
- Feature engineering (if needed)
- Outlier detection and treatment (if needed)
- Preparing data for modeling
- Any other preprocessing steps (if needed)

In [66]:
```
# See the distribution and skewness of dependent variable MEDV
print('Skew: ', round(df['MEDV'].skew(),2))
sns.histplot(data=df, x='MEDV', kde=True)
plt.show()
```

```
Skew:  1.11
```

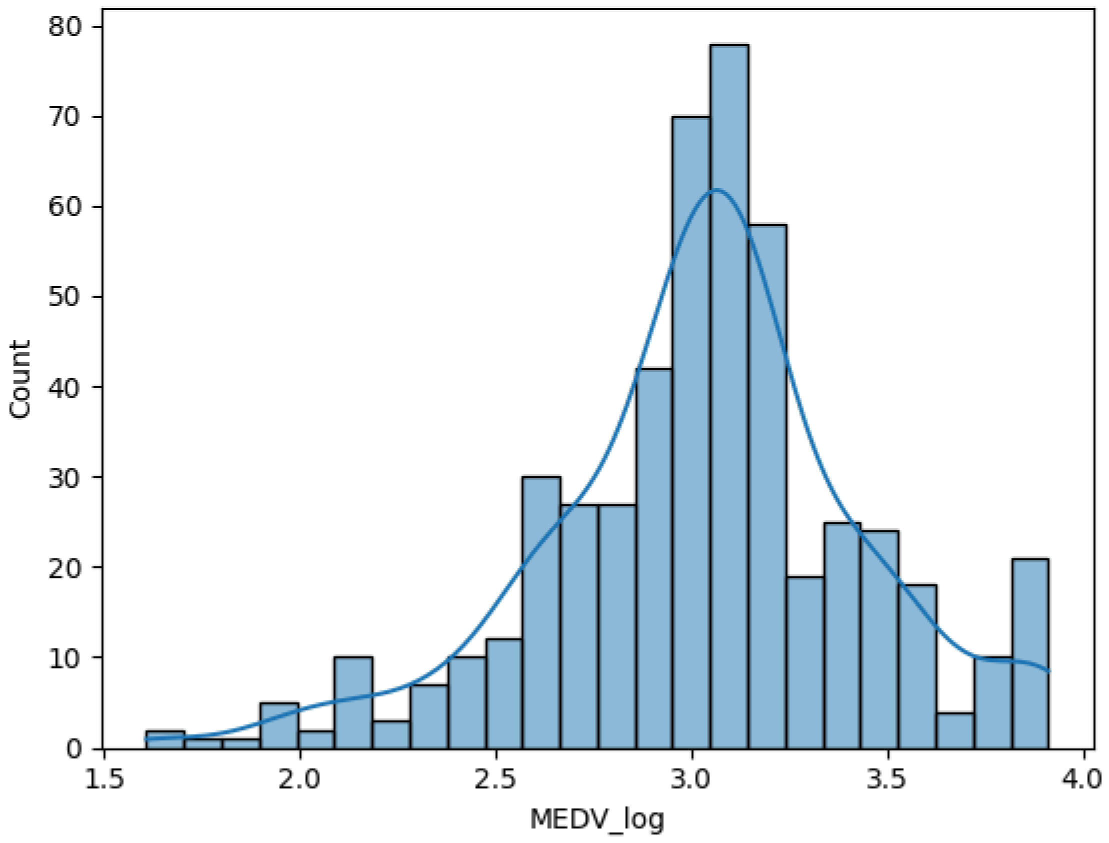


As the dependent variable is sightly skewed, we will apply a log transformation on the 'MEDV' column and check the distribution of the transformed column.

In [67]:
```
# Log transformation of dependent variable MEDV
df['MEDV_log'] = np.log(df['MEDV'])
```

In [68]:
```
# See the distribution and skewness of dependent variable MEDV
print('Skew: ', round(df['MEDV_log'].skew(),2))
sns.histplot(data=df, x='MEDV_log', kde=True)
plt.show()
```

```
Skew:  -0.33
```



**Observation:**

- After performing log transformation on MEDV column its skew has decreased from 1.11 to -0.33.
- Now MEDV_log have nearly normal distribution with very little skew of -0.33.

## Split the dataset

In [69]:
```
# Split the data into Independent and Dependent variables

X = df.drop(columns={'MEDV','MEDV_log'})
Y = df['MEDV_log']
```

In [70]:
```
# Adding a constant term to X
X_with_constant = sm.add_constant(X)
```

In [71]:
```
# Spliting dataset into 70:30 ratio of train test set
x_train,x_test,y_train,y_test = train_test_split(X_with_constant, Y, test_size=0.3, ra
```

## Check for Multicolinearity

In [72]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Function to check VIF
def checking_vif(train):
    vif_data = pd.DataFrame()
    vif_data["variables"] = train.columns

    # Calculating VIF for each feature
    vif_data["VIF"] = [
        variance_inflation_factor(train.values, i) for i in range(len(train.columns))
    ]
    return vif_data
```

In [73]:
```
# Using checking_vif function we created earlier to check vif on x_train data after nc
checking_vif(x_train).sort_values(by='VIF',ascending=False)
```

Out[73]:

| | variables | VIF |
|---|---|---|
| **0** | const | 535.372593 |
| **10** | TAX | 10.191941 |
| **9** | RAD | 8.345247 |
| **5** | NOX | 4.396157 |
| **8** | DIS | 4.355469 |
| **3** | INDUS | 3.999538 |
| **7** | AGE | 3.150170 |
| **12** | LSTAT | 2.861881 |
| **2** | ZN | 2.743574 |
| **11** | PTRATIO | 1.943409 |
| **1** | CRIM | 1.924114 |
| **6** | RM | 1.860950 |
| **4** | CHAS | 1.076564 |

### Observations:

There are two variables with a high VIF - RAD and TAX (greater than 5). Let's remove TAX as it has the highest VIF values and check the multicollinearity again.

In [74]:
```
# Create the model after dropping TAX
x_train = x_train.drop(columns =['TAX'])
x_test = x_test.drop(columns=['TAX'])

# Check for VIF
checking_vif(x_train).sort_values(by='VIF',ascending=False)
```

Out[74]:

| | variables | VIF |
|---|---|---|
| 0 | const | 532.025529 |
| 5 | NOX | 4.361847 |
| 8 | DIS | 4.333734 |
| 3 | INDUS | 3.270983 |
| 7 | AGE | 3.149005 |
| 9 | RAD | 2.942862 |
| 11 | LSTAT | 2.860251 |
| 2 | ZN | 2.483399 |
| 1 | CRIM | 1.923159 |
| 10 | PTRATIO | 1.909750 |
| 6 | RM | 1.857918 |
| 4 | CHAS | 1.050708 |

**Observation:**

- All the VIF are less than 5

## Model Building - Linear Regression

In [75]:
```
# Create the model using OLS from stats library

model1 = sm.OLS(y_train, x_train).fit()
y_train_pred = model1.predict(x_train)
y_test_pred = model1.predict(x_test)
```

In [76]:
```
# Get the model summary
print(model1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:               MEDV_log   R-squared:                       0.769
Model:                            OLS   Adj. R-squared:                  0.761
Method:                 Least Squares   F-statistic:                     103.3
Date:                Thu, 23 Nov 2023   Prob (F-statistic):          1.40e-101
Time:                        01:08:56   Log-Likelihood:                 76.596
No. Observations:                 354   AIC:                            -129.2
Df Residuals:                     342   BIC:                            -82.76
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          4.6324      0.243     19.057      0.000       4.154       5.111
CRIM          -0.0128      0.002     -7.445      0.000      -0.016      -0.009
ZN             0.0010      0.001      1.425      0.155      -0.000       0.002
INDUS         -0.0004      0.003     -0.148      0.883      -0.006       0.005
CHAS           0.1196      0.039      3.082      0.002       0.043       0.196
NOX           -1.0598      0.187     -5.675      0.000      -1.427      -0.692
RM             0.0532      0.021      2.560      0.011       0.012       0.094
AGE            0.0003      0.001      0.461      0.645      -0.001       0.002
DIS           -0.0503      0.010     -4.894      0.000      -0.071      -0.030
RAD            0.0076      0.002      3.699      0.000       0.004       0.012
PTRATIO       -0.0452      0.007     -6.659      0.000      -0.059      -0.032
LSTAT         -0.0298      0.002    -12.134      0.000      -0.035      -0.025
==============================================================================
Omnibus:                       30.699   Durbin-Watson:                   1.923
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               83.718
Skew:                           0.372   Prob(JB):                     6.62e-19
Kurtosis:                       5.263   Cond. No.                     2.09e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spec
ified.
[2] The condition number is large, 2.09e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

We can see that p-value for **ZN, INDUS and AGE** are higher than 0.05. So we drop this columns.

In [77]:
```
# Dropping columns with high p-values
x_train_new = x_train.drop(columns=['ZN','INDUS','AGE'])
x_test_new = x_test.drop(columns=['ZN','INDUS','AGE'])
```

In [78]:
```
# Checking VIF of new training set after dropping columns
checking_vif(x_train_new)
```

Out[78]:

| | variables | VIF |
|---|---|---|
| 0 | const | 526.961418 |
| 1 | CRIM | 1.892679 |
| 2 | CHAS | 1.049602 |
| 3 | NOX | 3.528194 |
| 4 | RM | 1.748438 |
| 5 | DIS | 2.582254 |
| 6 | RAD | 2.838523 |
| 7 | PTRATIO | 1.591527 |
| 8 | LSTAT | 2.437311 |

In [79]:

```
# Create second model after removing few columns

model2 = sm.OLS(y_train, x_train_new).fit()
y_train_pred = model2.predict(x_train_new)
y_test_pred = model2.predict(x_test_new)

# Get the model summary
print(model2.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:               MEDV_log   R-squared:                       0.767
Model:                            OLS   Adj. R-squared:                  0.762
Method:                 Least Squares   F-statistic:                     142.1
Date:                Thu, 23 Nov 2023   Prob (F-statistic):          2.61e-104
Time:                        01:08:56   Log-Likelihood:                 75.486
No. Observations:                 354   AIC:                            -133.0
Df Residuals:                     345   BIC:                            -98.15
Df Model:                           8                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          4.6494      0.242     19.242      0.000       4.174       5.125
CRIM          -0.0125      0.002     -7.349      0.000      -0.016      -0.009
CHAS           0.1198      0.039      3.093      0.002       0.044       0.196
NOX           -1.0562      0.168     -6.296      0.000      -1.386      -0.726
RM             0.0589      0.020      2.928      0.004       0.019       0.098
DIS           -0.0441      0.008     -5.561      0.000      -0.060      -0.028
RAD            0.0078      0.002      3.890      0.000       0.004       0.012
PTRATIO       -0.0485      0.006     -7.832      0.000      -0.061      -0.036
LSTAT         -0.0293      0.002    -12.949      0.000      -0.034      -0.025
==============================================================================
Omnibus:                       32.514   Durbin-Watson:                   1.925
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               87.354
Skew:                           0.408   Prob(JB):                     1.07e-19
Kurtosis:                       5.293   Cond. No.                         690.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spec
ified.
```

**Observations:**

- All the VIF Scores are now less than 5 indicating no multicollinearity.
- Now, all the p values are lesser than 0.05 implying all the current variables are significant for the model.
- The R-Squared value did not change by much. It is still coming out to be ~0.76.

## Checking linear regression assumptions

Assumptions of linear regression

1. Mean of residuals should be 0
2. No Heteroscedasticity
3. Linearity of variables
4. Normality of error terms

**1. Mean of residuals should be 0**

In [80]:
```
residual = model2.resid
```

In [81]:
```
residual.mean()
```

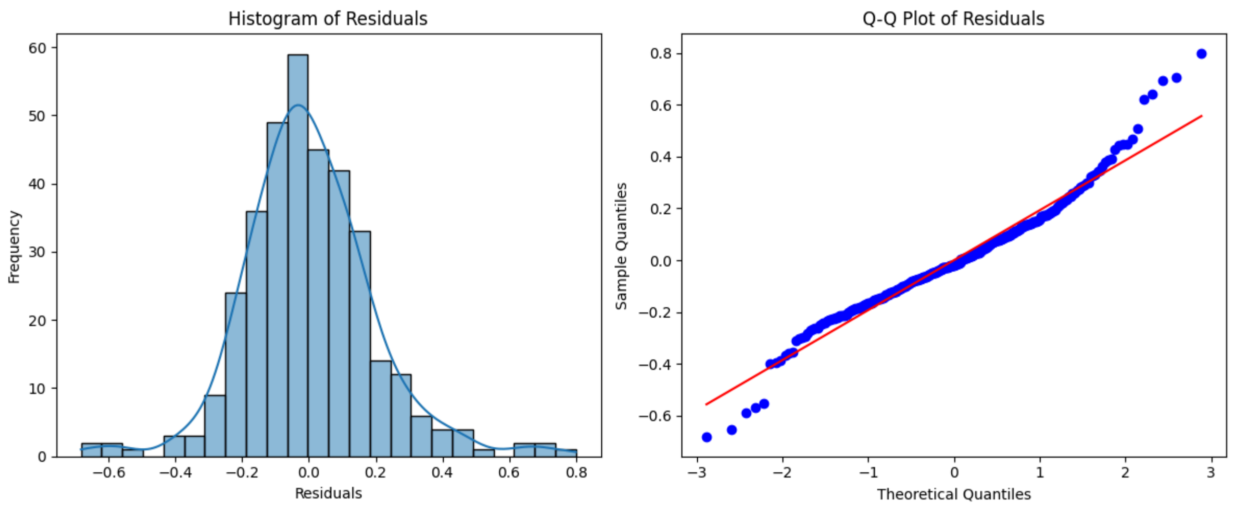
Out[81]:
```
-2.5303049047106675e-15
```

The mean of residuals is very close to 0. Hence, the corresponding assumption is satisfied.

### 2. Tests for Normality

In [82]:
```
# Plot histogram of residuals
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
sns.histplot(residual, kde = True)
plt.title('Histogram of Residuals')
plt.xlabel('Residuals')
plt.ylabel('Frequency')
print('Skew: ',round(residual.skew(),2))

# Plot q-q plot of residuals
import pylab
import scipy.stats as stats
plt.subplot(1,2,2)
stats.probplot(residual, dist = "norm", plot = pylab)
plt.title('Q-Q Plot of Residuals')
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Sample Quantiles')

plt.tight_layout()
plt.show()
```

```
Skew:  0.41
```

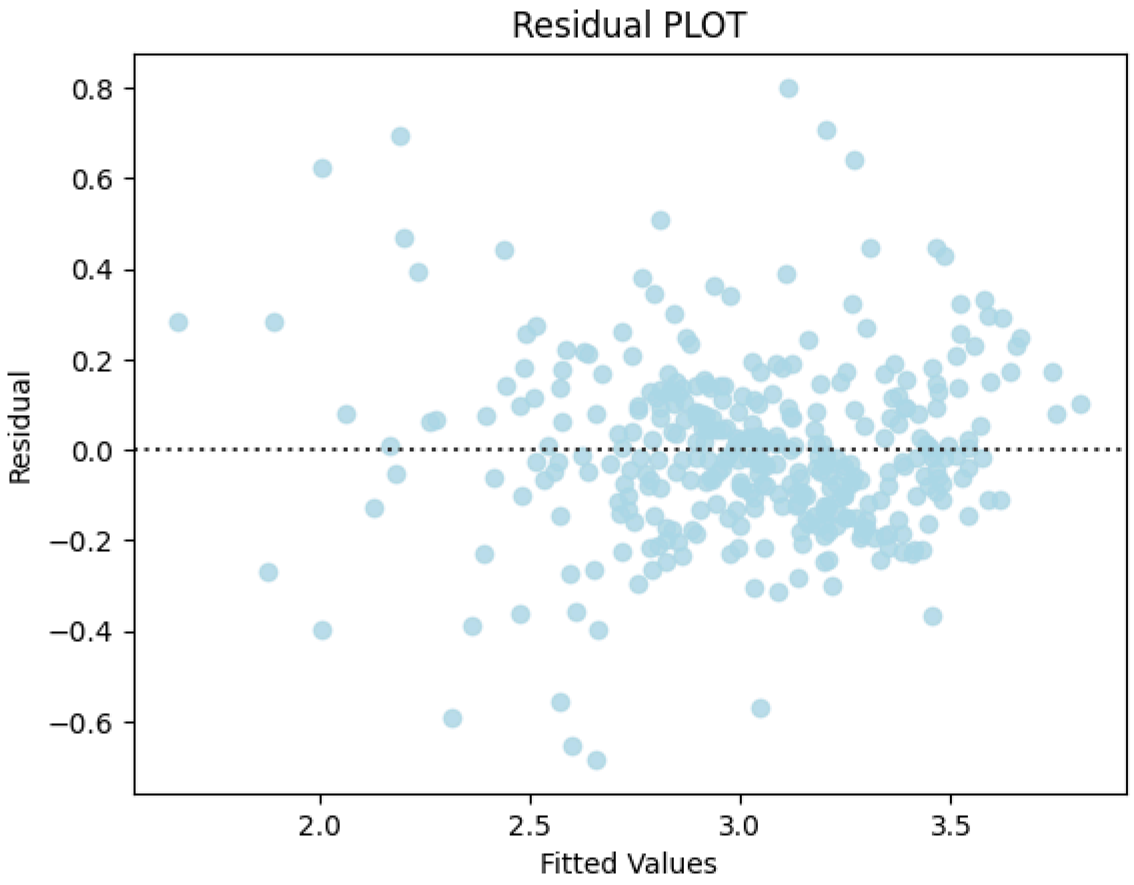


We can see that the error terms are normally distributed. The assumption of normality is satisfied.

### 3. Linearity of Variables

In [83]:
```
#  Residual Plot

fitted = model2.fittedvalues

sns.residplot(x = fitted, y = residual, color = "lightblue")
```

```
plt.xlabel("Fitted Values")

plt.ylabel("Residual")

plt.title("Residual PLOT")

plt.show()
```



We can see that there is no pattern in the residuals vs fitted values scatter plot now, i.e., the linearity assumption is satisfied

**4. Test for Homoscedasticity**

- We will use Goldfeld–Quandt test to check homoscedasticity.
- Null hypothesis : Residuals are homoscedastic
- Alternate hypothesis : Residuals are hetroscedastic

In [84]:
```
# Importing the required library

from statsmodels.stats.diagnostic import het_white

from statsmodels.compat import lzip

import statsmodels.stats.api as sms
```

In [85]:
```
name = ["F statistic", "p-value"]

test = sms.het_goldfeldquandt(y_train, x_train_new)

lzip(name, test)
```

Out[85]:
```
[('F statistic', 1.0835082923425292), ('p-value', 0.30190120067668275)]
```

As we observe from the above test, the p-value is greater than 0.05, so we fail to reject the null-hypothesis. That means the residuals are homoscedastic.

We have verified all the assumptions of the linear regression model.

## Model Performance Check

1. How does the model is performing? Check using Rsquared, RSME, MAE, MAPE
2. Is there multicollinearity? Check using VIF
3. How does the model is performing after cross validation?

### 1. Evaluation Metrics

In [86]:
```
from sklearn.metrics import mean_absolute_error

mse_train = mean_squared_error(y_train, y_train_pred)
mse_test = mean_squared_error(y_test, y_test_pred)

r2_train = r2_score(y_train, y_train_pred)
r2_test = r2_score(y_test, y_test_pred)

rmse_train = np.sqrt(mse_train)
rmse_test = np.sqrt(mse_test)

mae_train = mean_absolute_error(y_train, y_train_pred)
mae_test = mean_absolute_error(y_test, y_test_pred)

mape_train = np.mean(np.abs((y_train - y_train_pred) / y_train)) * 100
mape_test = np.mean(np.abs((y_test - y_test_pred) / y_test)) * 100

metrics_df = print(
                    pd.DataFrame(
                        {
                            'Data':['Train','Test'],
                            'R2':[r2_train,r2_test],
                            'MSE':[mse_train, mse_test],
                            'RMSE':[rmse_train, rmse_test],
                            'MAE':[mae_train, mae_test],
                            'MAPE':[mape_train, mape_test]
                            }
                        )
                    )
```

```
    Data        R2       MSE      RMSE       MAE      MAPE
0  Train  0.767174  0.038222  0.195504  0.143686  4.981813
1   Test  0.772486  0.039222  0.198045  0.151284  5.257965
```

**Observation:**

- An $R^2$ of 0.76 for the training data and 0.77 for the test data suggests that the model explains approximately 76.7% and 77.2% of the variance in the respective datasets.
- Lower MSE, RMSE, MAE, MAPE values indicate better model performance.
- The model appears to generalize reasonably well from the training to the test dataset, as the performance metrics on the test set are comparable to those on the training set.
- The differences between training and test performance metrics are relatively small, suggesting that the model is not significantly overfitting or underfitting.

## 2. Cross-validation

In [87]:
```
# Import the required function

from sklearn.model_selection import cross_val_score

# Build the regression model and cross-validate
lr = LinearRegression()

cv_Score11 = cross_val_score(lr, x_train, y_train, cv = 10)
cv_Score12 = cross_val_score(lr, x_train, y_train, cv = 10,
                             scoring = 'neg_mean_squared_error')


print("RSquared: %0.3f (+/- %0.3f)" % (cv_Score11.mean(), cv_Score11.std() * 2))
print("Mean Squared Error: %0.3f (+/- %0.3f)" % (-1*cv_Score12.mean(), cv_Score12.std(
```

```
RSquared: 0.727 (+/- 0.240)
Mean Squared Error: 0.041 (+/- 0.024)
```

**Observation:**

- RSqured value of 0.72 incidates the model explains approximately 72.7% of the variance in the target variable.
- The low MSE of 0.041 indicates that, on average, the model's predictions are close to the actual values.
- The standard deviations ("+/-") provide insights into the variability of performance across different folds

## Final Model

In [88]:
```
# Creat feature and its coefs dataframe.
```

```
coef = model2.params
pd.DataFrame({'Feature' : coef.index, 'Coefs' : coef.values})
```

Out[88]:

| | Feature | Coefs |
|---|---|---|
| **0** | const | 4.649386 |
| **1** | CRIM | -0.012500 |
| **2** | CHAS | 0.119773 |
| **3** | NOX | -1.056225 |
| **4** | RM | 0.058907 |
| **5** | DIS | -0.044069 |
| **6** | RAD | 0.007848 |
| **7** | PTRATIO | -0.048504 |
| **8** | LSTAT | -0.029277 |

### Final Model

MEDV_log = 4.6 - (0.012 *CRIM)* + *(0.119* CHAS) - (1.056 *NOX)* + *(0.058* RM) - (0.044 *DIS)* + *(0.007* RAD) - (0.048 *PTRATIO)* - *(0.029* LSTAT)

## Actionable Insights and Recommendations

**The implications of each coefficient based on the linear regression model**

**1. Intercept (const):**

> Coefficient: 4.649386
>
> **Insight:** It represents the baseline house price when all other variables are zero.

**2. CRIM (Crime Rate):**

> **Coefficient:** -0.012500
>
> **Insight:** This suggests that higher crime rates are associated with lower house prices.
>
> **Recommendation:** Implement community safety initiatives to address and reduce crime rates.Build Gated community with increased police surveillance. Provide credit to install security system.

**3. CHAS (Charles River Dummy Variable):**

> **Coefficient:** 0.119773
>
> **Insight:** This suggests that houses near the Charles River may have higher prices.

> **Recommendation:** Charles River provides a lot of recreational activities such as kayaking, boating and hiking which are associated with a better quality of life. These unique selling points should be promoted to potential buyers.

**4. NOX (Nitric Oxide Concentration):**

> **Coefficient:** -1.056225
>
> **Insight:** Higher nitric oxide levels are associated with lower house prices.
>
> **Recommendation:** Since nitric oxide presents a health hazard, the sources of high concentration should be investigated and remedied.

**5. RM (Average Number of Rooms):**

> **Coefficient:** 0.058907
>
> **Insight:** This implies that houses with more rooms tend to have higher prices.
>
> **Recommendation:** Encourage property development or renovations that result in an increase in the number of rooms in houses.

**6. DIS (Weighted Distances to Employment Centers):**

> **Coefficient:** -0.044069
>
> **Insight:** This suggests that, on average, price of houses decrease as it farther away from employement center.
>
> **Recommendation:** Promote the unique benefits of living farther from employment centers, such as tranquility or larger property sizes. Such houses tend to have private garages. Promote any transportation system like railway station or bus station if any near the area for ease of commute.

**7. RAD (Accessibility to Radial Highways):**

> **Coefficient:** 0.007848
>
> **Insight:** This implies a positive association between accessibility to highways and house prices.
>
> **Recommendation:** Easy highway access provides lots of benefits such as lesser commute time, quick weekend getaways and better connectivity all of which should be highlighted.

**8. PTRATIO (Pupil-Teacher Ratio):**

> **Coefficient:** -0.048504
>
> **Insight:** Higher pupil-teacher ratios may be associated with lower house prices.

> **Recommendation:** Advocate for improvements in education resources, potentially leading to lower pupil-teacher ratios.

**9. LSTAT (Percentage of Lower Status of the Population):**

> **Coefficient:** -0.029277
>
> **Insight:** Higher percentages of lower-status population may be associated with lower house prices.
>
> **Recommendation:** Focus on community development programs to address socioeconomic disparities. This might involve initiatives related to education, job opportunities, and affordable housing.