ASSIGNMENT - 4

In [1]:
```
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
```

In [2]:
```
df = pd.read_csv("housingdata.csv")
```

In [3]:
```
df
```

Out[3]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRA
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	
...	...	...	...	...	...	...	...	...	...	...	
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273.0	
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273.0	
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273.0	
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273.0	
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273.0	

506 rows × 14 columns

In [4]:
```
df.head(5)
```

Out[4]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATI
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.

In [5]:
```
df.tail(5)
```

Out[5]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRAT
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273.0	2
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273.0	2
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273.0	2
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273.0	2
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273.0	2

Data Preprocessing

In [6]:
```
# Handle missing values
data = df.dropna()
```

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
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
```

In [9]:
```
#check dataset summary
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     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
```

In [10]:
```
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
```

In [11]:
```
data.describe()
```

Out[11]:

	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

In [13]:
```
# Separate features and target
X = data.drop(columns=['MEDV']) # Features
y = data['MEDV'] # Target
```

In [19]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rar
```

In [21]:
```
# Initialize and train the model
model = LinearRegression()
model.fit(X_train, y_train)
```

Out[21]:

▾ LinearRegression

```
LinearRegression()
```

In [22]:
```
# Make predictions on the test set
y_pred = model.predict(X_test)
```

In [23]:
```
# Calculate evaluation metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 24.291119474973613
```

In [24]:
```
print("R-squared Score:", r2)
```

```
R-squared Score: 0.6687594935356307
```

In [ ]: