In [1]:
```python
#Importing Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [29]:
```python
#Exercise 1
#Loading the dataset
data_cols = ['CRIM','ZN','INDUS','CHAS','NOX','RM','AGE','DIS','RAD','TAX','PTRATIO
df = pd.read_csv("housing.csv", header=None, delimiter=r"\s+", names=data_cols)
df
```

Out[29]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	I
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.9
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.9
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.8
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.6
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.9
...	...	...	...	...	...	...	...	...	...	...	...	.
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273.0	21.0	391.9
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273.0	21.0	396.9
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273.0	21.0	396.9
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273.0	21.0	393.4
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273.0	21.0	396.9

506 rows × 14 columns

In [30]:
```python
print("Features of the dataset")
for i in df.columns[:-1]:
  print(i)
print("\nTarget of the dataset")
print(df.columns[-1])
```

```
Features of the dataset
CRIM
ZN
INDUS
CHAS
NOX
RM
AGE
DIS
RAD
TAX
PTRATIO
B
LSTAT

Target of the dataset
PRICE
```

In [31]:

```python
#Shape of the dataset
print("Shape = ",df.shape)
```

```
Shape =  (506, 14)
```

In [32]:

```python
#Exercise 2

#Checking for the null values
df.isnull().sum()
```

Out[32]:

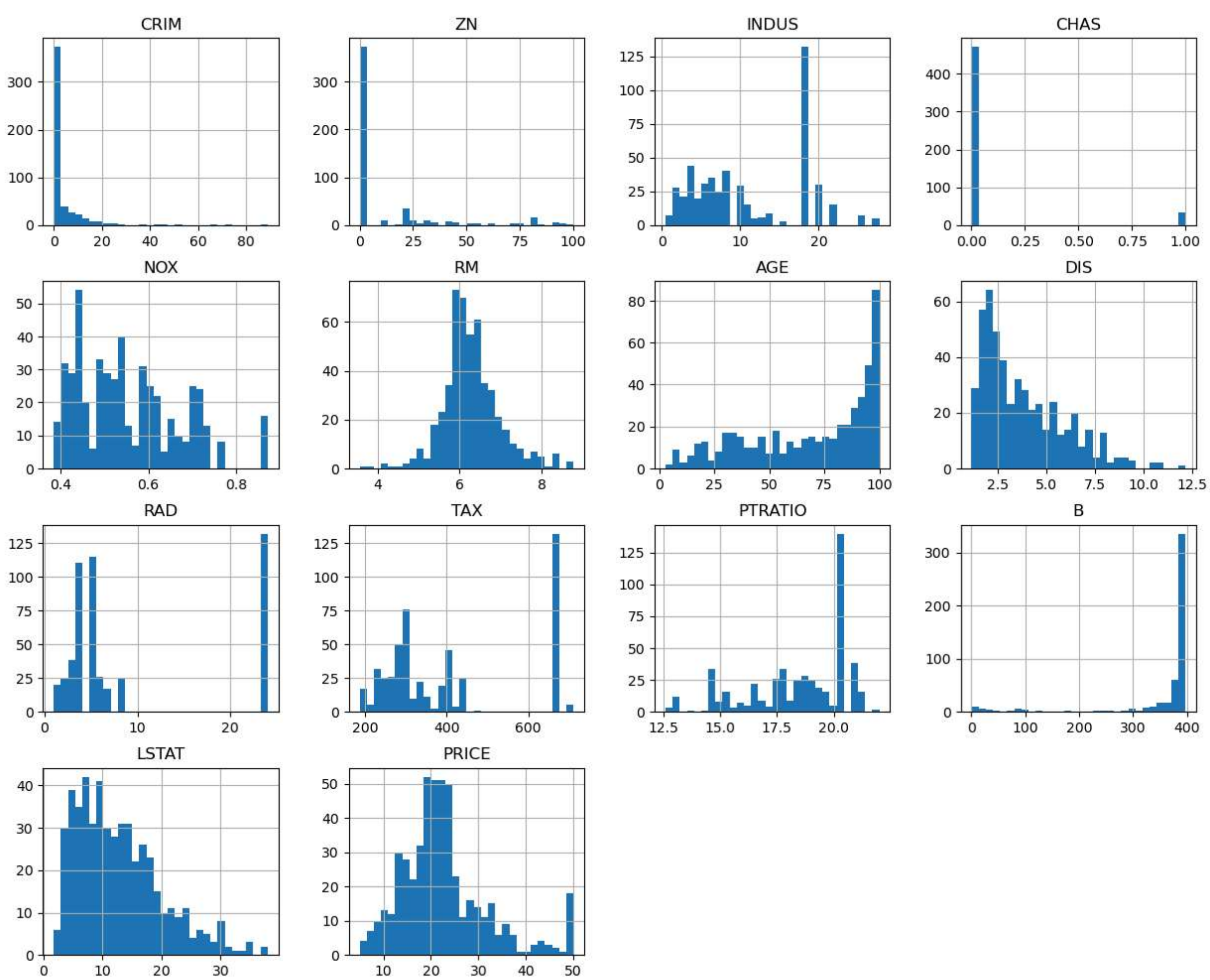
```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
PRICE      0
dtype: int64
```

In [33]:

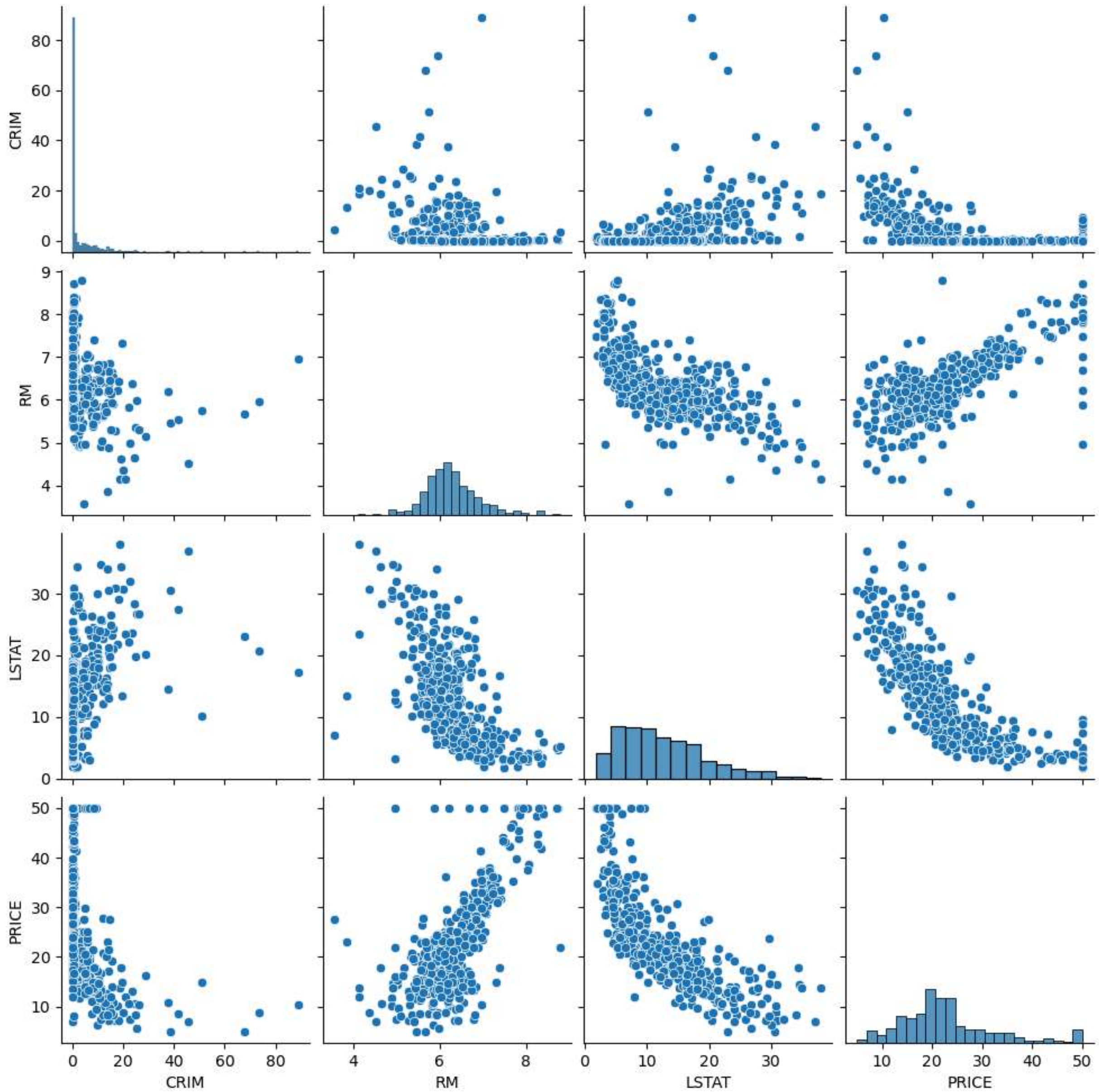
```python
#Information about the dataset
df.info()
```

```
<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  PRICE    506 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 55.5 KB
```

In [34]:
```
#Plotting histogram
df.hist(bins=30, figsize=(15,12))
plt.show()
```



In [35]:
```
#Plotting pairplots
sns.pairplot(df[['CRIM','RM','LSTAT','PRICE']])
plt.show()
```



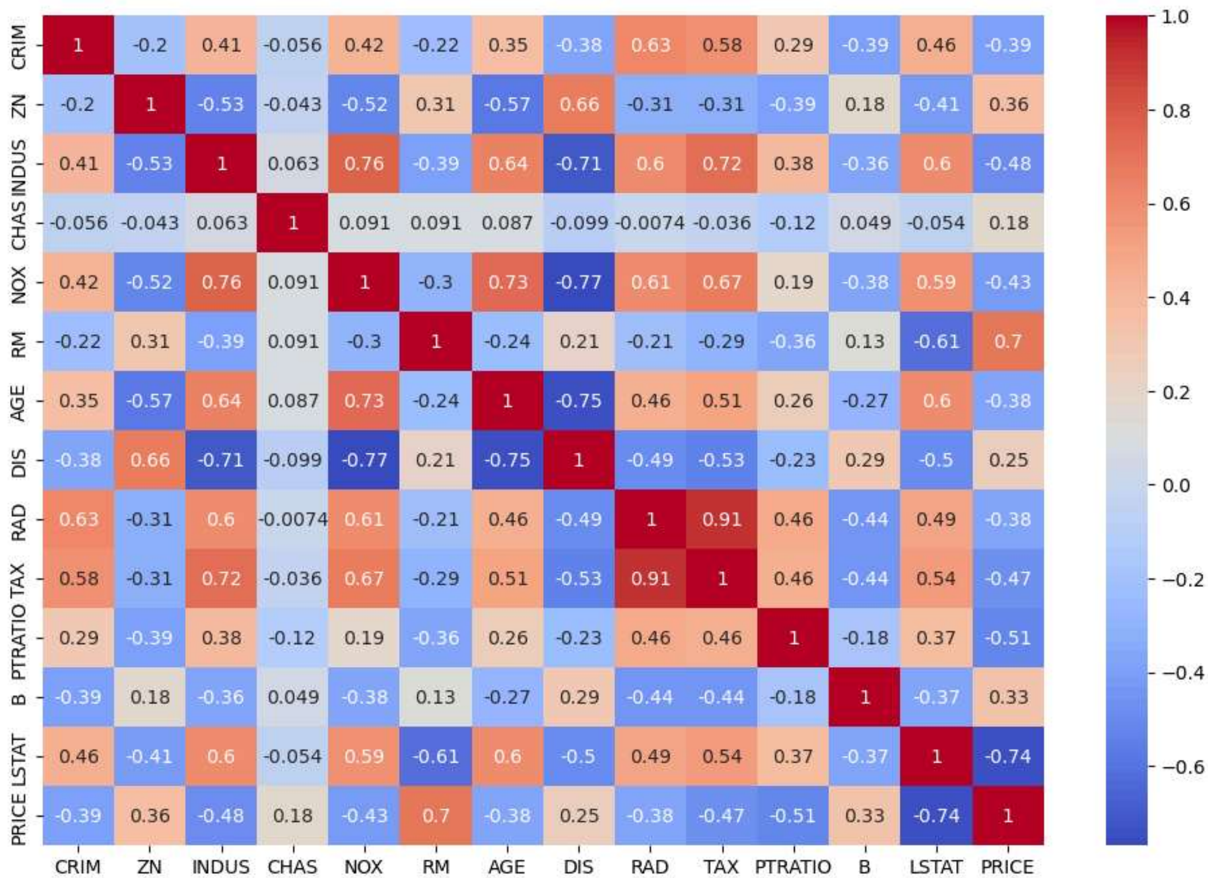
In [36]:
```
#Correlation matrix
corr = df.corr()
corr
```

Out[36]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	
CRIM	1.000000	-0.200469	0.406583	-0.055892	0.420972	-0.219247	0.352734	-0.379
ZN	-0.200469	1.000000	-0.533828	-0.042697	-0.516604	0.311991	-0.569537	0.664
INDUS	0.406583	-0.533828	1.000000	0.062938	0.763651	-0.391676	0.644779	-0.708
CHAS	-0.055892	-0.042697	0.062938	1.000000	0.091203	0.091251	0.086518	-0.099
NOX	0.420972	-0.516604	0.763651	0.091203	1.000000	-0.302188	0.731470	-0.769
RM	-0.219247	0.311991	-0.391676	0.091251	-0.302188	1.000000	-0.240265	0.205
AGE	0.352734	-0.569537	0.644779	0.086518	0.731470	-0.240265	1.000000	-0.747
DIS	-0.379670	0.664408	-0.708027	-0.099176	-0.769230	0.205246	-0.747881	1.000
RAD	0.625505	-0.311948	0.595129	-0.007368	0.611441	-0.209847	0.456022	-0.494
TAX	0.582764	-0.314563	0.720760	-0.035587	0.668023	-0.292048	0.506456	-0.534
PTRATIO	0.289946	-0.391679	0.383248	-0.121515	0.188933	-0.355501	0.261515	-0.232
B	-0.385064	0.175520	-0.356977	0.048788	-0.380051	0.128069	-0.273534	0.291
LSTAT	0.455621	-0.412995	0.603800	-0.053929	0.590879	-0.613808	0.602339	-0.496
PRICE	-0.388305	0.360445	-0.483725	0.175260	-0.427321	0.695360	-0.376955	0.249

In [37]:

```
#Heatmap visualization
plt.figure(figsize=(12,8))
sns.heatmap(corr,annot=True,cmap='coolwarm')
plt.show()
```



In [38]:
```
#Exercise 3

#RM is highly correlated with PRICE
X = df['RM']
y = df['PRICE']
```

In [39]:
```
#Train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [40]:
```
#Scalarization of the data
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train.values.reshape(-1,1))
X_test = scaler.transform(X_test.values.reshape(-1,1))
```

In [41]:
```
#Fit the model
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train,y_train)
```

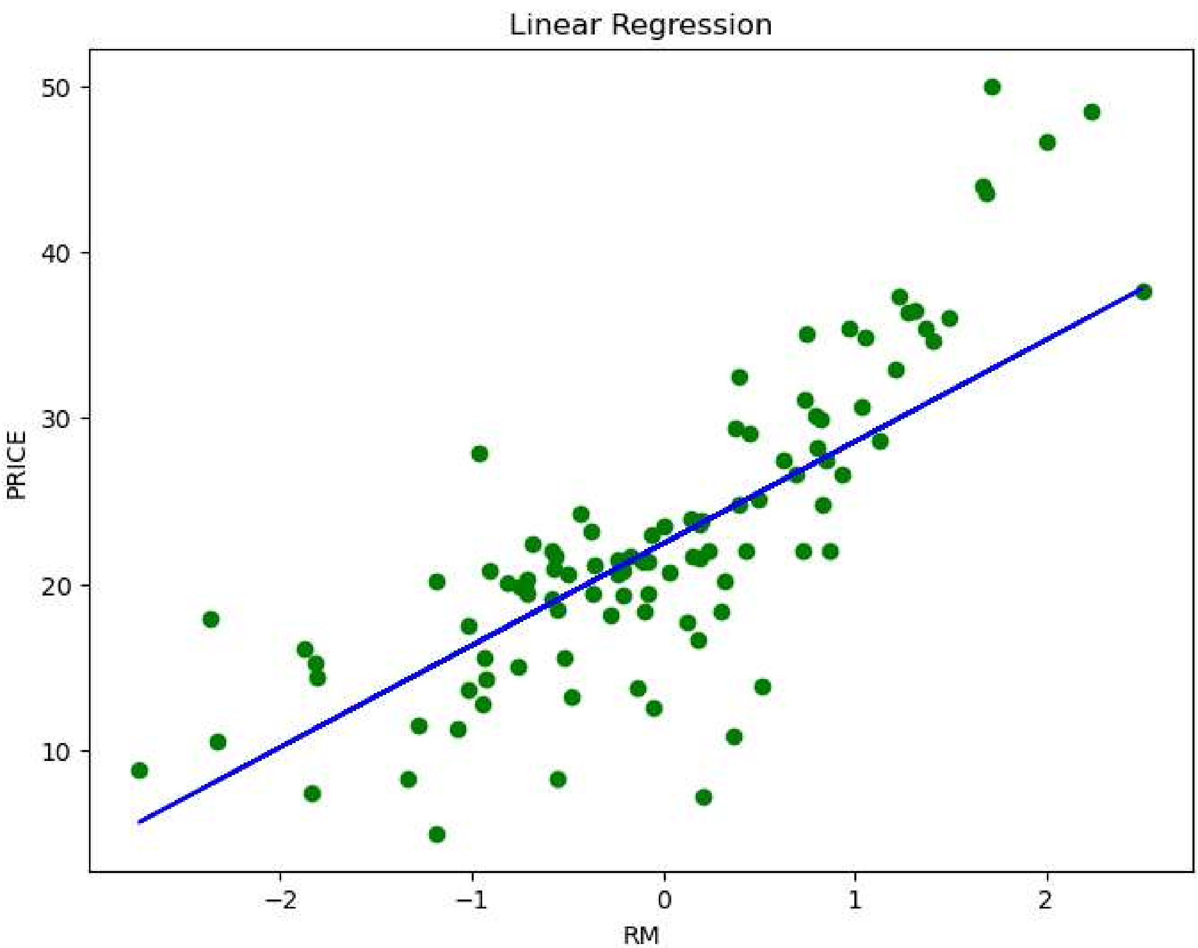
Out[41]:
```
LinearRegression

LinearRegression()
```

In [42]:
```python
#Intercept
print("Intercept = ",model.intercept_)
print("Slope = ",model.coef_)
```

```
Intercept =  22.441336633663354
Slope =  [6.13292429]
```

In [43]:
```python
#Prediction
y_pred = model.predict(X_test)
```

In [44]:
```python
#Plot
plt.figure(figsize=(8,6))
plt.scatter(X_test,y_test,color='green')
plt.plot(X_test,y_pred,color='blue')
plt.xlabel('RM')
plt.ylabel('PRICE')
plt.title('Linear Regression')
plt.show()
```



In [45]:
```python
#Exercise 5
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score, clas
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
print("Mean Absolute Error:", mae)
```

```
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R2 Score:", r2)
```

```
Mean Absolute Error: 4.0900649551844195
Mean Squared Error: 30.657592804650935
Root Mean Squared Error: 5.536929907868704
R2 Score: 0.633543994842449
```

In [46]:
```
#Exercise 4 (Multi Linear Regression)

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

In [47]:
```
#Train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [48]:
```
#Scalarization of the data
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_s = scaler.fit_transform(X_train)
X_test_s = scaler.transform(X_test)
```

In [49]:
```
#Fit the model
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train_s,y_train)
```

Out[49]:
```
LinearRegression

LinearRegression()
```

In [50]:
```
print("Intercept = ",model.intercept_)
print("Slope = ",model.coef_)
```

```
Intercept =  22.44133663366336
Slope =  [-0.93451207  0.85487686 -0.10446819  0.81541757 -1.90731862  2.54650028
  0.25941464 -2.92654009  2.80505451 -1.95699832 -2.15881929  1.09153332
 -3.91941941]
```

In [51]:
```
y_pred = model.predict(X_test_s)
```

In [52]:
```
#Exercise 5
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score, clas
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
print("Mean Absolute Error:", mae)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R2 Score:", r2)
```

```
Mean Absolute Error: 3.113043746893427
Mean Squared Error: 18.49542012244839
Root Mean Squared Error: 4.300630200615765
R2 Score: 0.7789207451814418
```

In [ ]:

```

```