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

In [3]:
```
from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_s
```

In [5]:
```
sns.set(style="whitegrid")
```

In [122…
```
boston = fetch_openml(name='boston', version=1, as_frame=True)
#df1= pd.read_csv("Desktop/datasets/boston/train.csv")

df = boston.frame

df.head()
```

Out[122…

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	39
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	39
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	39
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	39
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	39

In [86]:
```
df.shape
df.info()
df.describe()
```

```
<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    category
 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    category
 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: category(2), float64(12)
memory usage: 49.0 KB
```

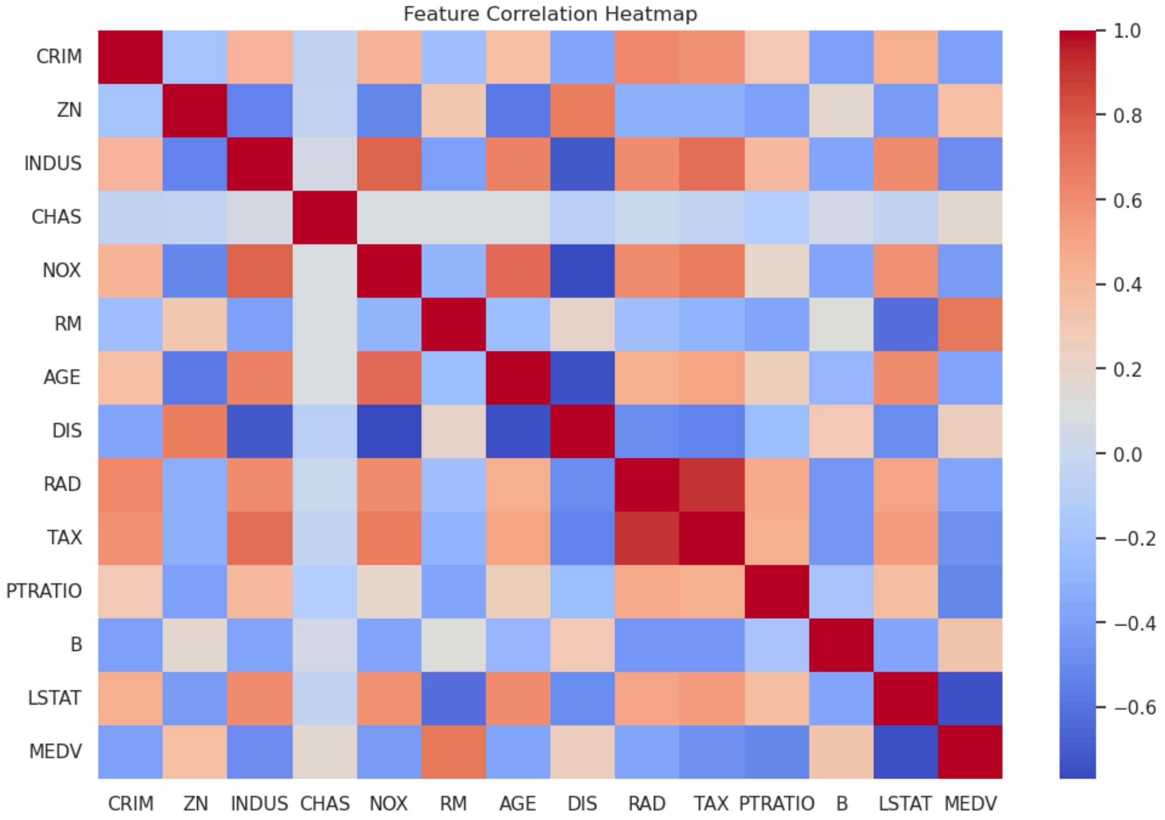
Out[86]:

	CRIM	ZN	INDUS	NOX	RM	AGE	
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.(
mean	3.613524	11.363636	11.136779	0.554695	6.284634	68.574901	3.7
std	8.601545	23.322453	6.860353	0.115878	0.702617	28.148861	2.1
min	0.006320	0.000000	0.460000	0.385000	3.561000	2.900000	1.1
25%	0.082045	0.000000	5.190000	0.449000	5.885500	45.025000	2.1
50%	0.256510	0.000000	9.690000	0.538000	6.208500	77.500000	3.2
75%	3.677083	12.500000	18.100000	0.624000	6.623500	94.075000	5.1
max	88.976200	100.000000	27.740000	0.871000	8.780000	100.000000	12.1

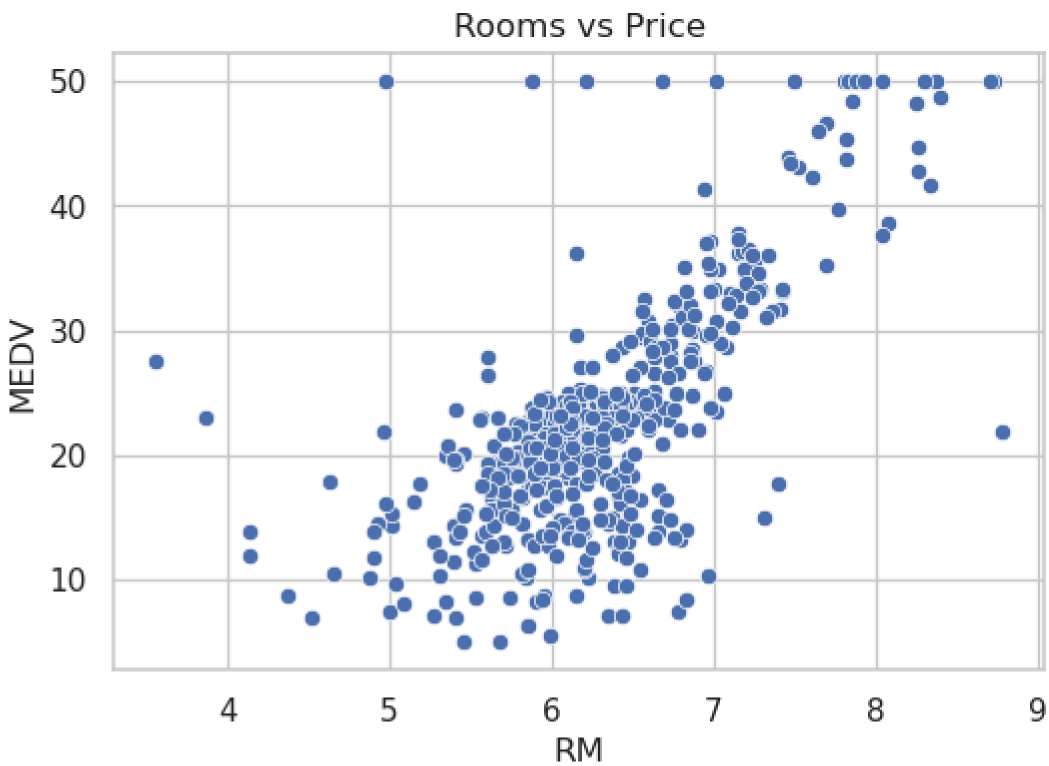
In [88]:
```
df.isnull().sum()
```

Out[88]:
```
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 [90]:
```
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(), annot=False, cmap='coolwarm')
plt.title("Feature Correlation Heatmap")
plt.show()
```



In [92]:
```
plt.figure(figsize=(6,4))
sns.scatterplot(x=df['RM'], y=df['MEDV'])
plt.title("Rooms vs Price")
plt.show()
```



In [94]:
```
X = df.drop('MEDV', axis=1)  # independent variables
y = df['MEDV']               # target variable
```

In [96]:
```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
```

```
)

print("Training samples:", X_train.shape)
print("Testing samples:", X_test.shape)
```

```
Training samples: (404, 13)
Testing samples: (102, 13)
```

In [98]:
```
model = LinearRegression()
model.fit(X_train, y_train)
```

Out[98]:

```
▾ LinearRegression
LinearRegression()
```

In [100…
```
coeff_df = pd.DataFrame(model.coef_, X.columns, columns=['Coefficient'])
coeff_df
```

Out[100…

	Coefficient
CRIM	-0.113056
ZN	0.030110
INDUS	0.040381
CHAS	2.784438
NOX	-17.202633
RM	4.438835
AGE	-0.006296
DIS	-1.447865
RAD	0.262430
TAX	-0.010647
PTRATIO	-0.915456
B	0.012351
LSTAT	-0.508571

In [102…
```
X.dtypes
```

Out[102…
```
CRIM          float64
ZN            float64
INDUS         float64
CHAS         category
NOX           float64
RM            float64
AGE           float64
DIS           float64
RAD          category
TAX           float64
PTRATIO       float64
B             float64
LSTAT         float64
dtype: object
```

In [104…
```
X = X.apply(pd.to_numeric)
y = pd.to_numeric(y)
```

In [106…
```
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_state=42
)
```

In [108…
```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train, y_train)
```

Out[108…
```
▾ LinearRegression
LinearRegression()
```

In [110…
```
y_pred = model.predict(X_test)
```

In [112…
```
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred)))
print("R2 Score:", r2_score(y_test, y_pred))
```

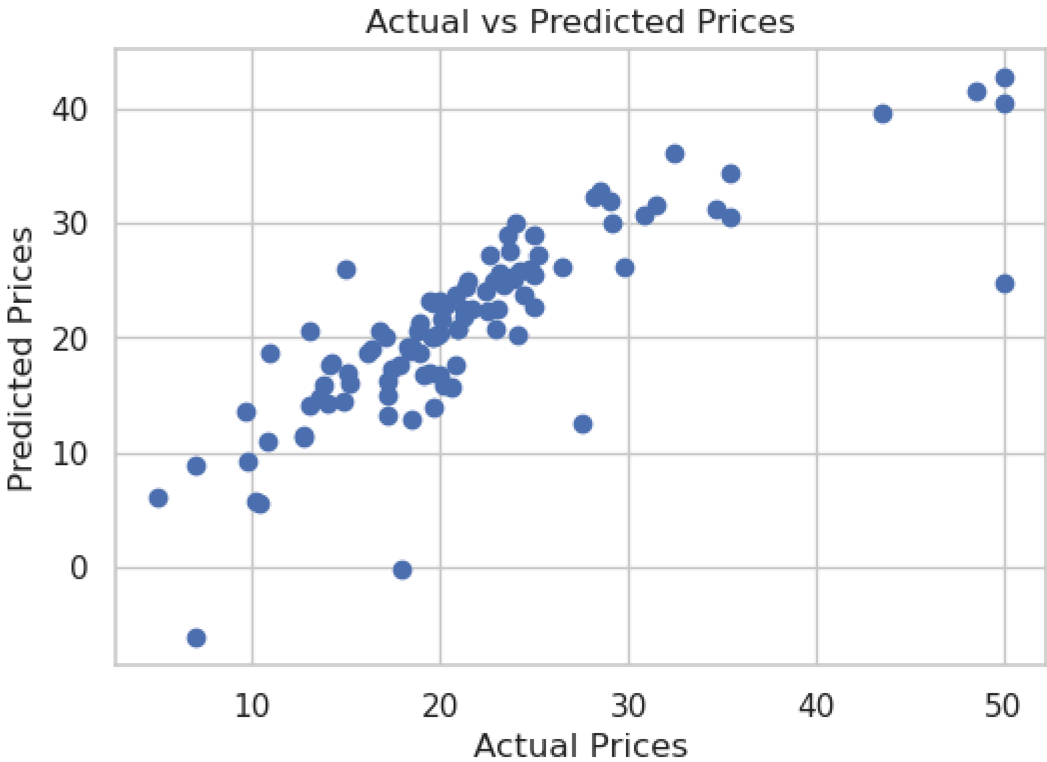
```
RMSE: 4.928602182665346
R2 Score: 0.6687594935356307
```

In [114…
```
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: 3.189091965887852
Mean Squared Error: 24.291119474973613
Root Mean Squared Error: 4.928602182665346
R² Score: 0.6687594935356307
```

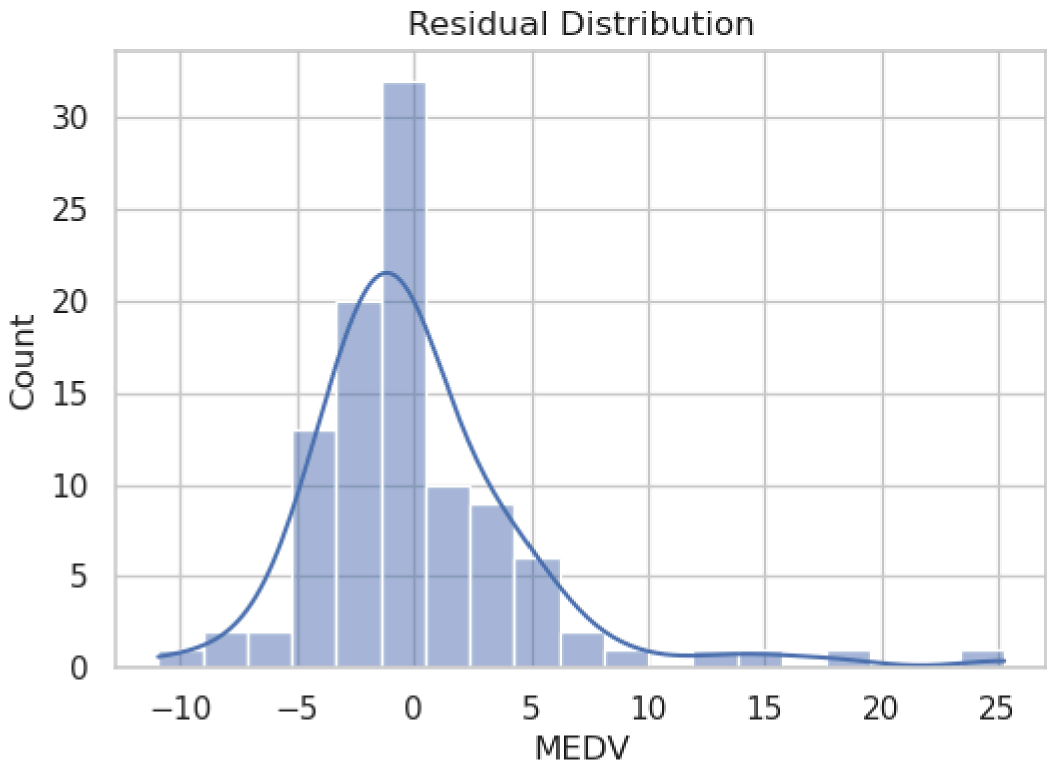
In [116…
```
plt.figure(figsize=(6,4))
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual vs Predicted Prices")
plt.show()
```



In [118…

```
residuals = y_test - y_pred

plt.figure(figsize=(6,4))
sns.histplot(residuals, kde=True)
plt.title("Residual Distribution")
plt.show()
```



In [ ]:

```
##multiple
```

In [124…

```
import pandas as pd
import numpy as np
```

```python
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_s
```

In [126…

```python
boston = fetch_openml(name='boston', version=1, as_frame=True)
df = boston.frame
df = df.apply(pd.to_numeric)  # ensure all values are numeric
```

In [128…

```python
print(df.shape)
df.head()
```

(506, 14)

Out[128…

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	39
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	39
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	39
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	39
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	39

In [130…

```python
X = df.drop("MEDV", axis=1)   # multiple input variables
y = df["MEDV"]                # house price
```

In [132…

```python
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

In [134…

```python
model = LinearRegression()
model.fit(X_train, y_train)
```

Out[134…

LinearRegression

LinearRegression()

In [136…

```python
coeff_df = pd.DataFrame(model.coef_, X.columns, columns=['Coefficient'])
coeff_df
```

Out[136...

	Coefficient
CRIM	-0.113056
ZN	0.030110
INDUS	0.040381
CHAS	2.784438
NOX	-17.202633
RM	4.438835
AGE	-0.006296
DIS	-1.447865
RAD	0.262430
TAX	-0.010647
PTRATIO	-0.915456
B	0.012351
LSTAT	-0.508571

In [138...

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

In [140...

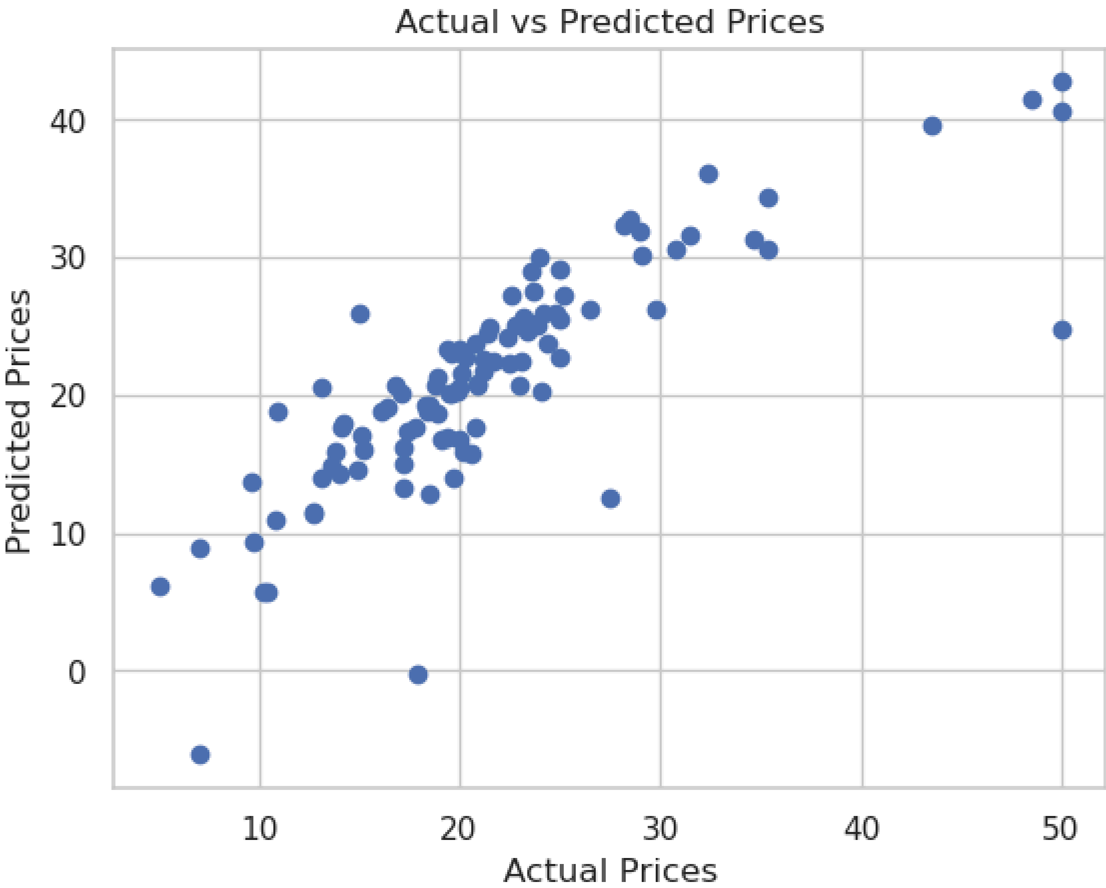
```
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print("MAE:", mae)
print("MSE:", mse)
print("RMSE:", rmse)
print("R² Score:", r2)
```

```
MAE: 3.189091965887852
MSE: 24.291119474973613
RMSE: 4.928602182665346
R² Score: 0.6687594935356307
```

In [142...

```
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual vs Predicted Prices")
plt.show()
```



In [144…
```
#
```

In [146…
```
#singular
```

In [148…
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

In [150…
```
boston = fetch_openml(name='boston', version=1, as_frame=True)
df = boston.frame
df = df.apply(pd.to_numeric)
```

In [152…
```
X = df[["RM"]]      # singular feature
y = df["MEDV"]      # target
```

In [154…
```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

In [156…
```
model = LinearRegression()
model.fit(X_train, y_train)
```

Out[156…

```
▾ LinearRegression
LinearRegression()
```

In [162…

```
print("Slope (Coefficient):", model.coef_[0])
print("Intercept:", model.intercept_)
#Price=(Slope×RM)+Intercept
```

```
Slope (Coefficient): 9.348301406497727
Intercept: -36.24631889813795
```

In [164…

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

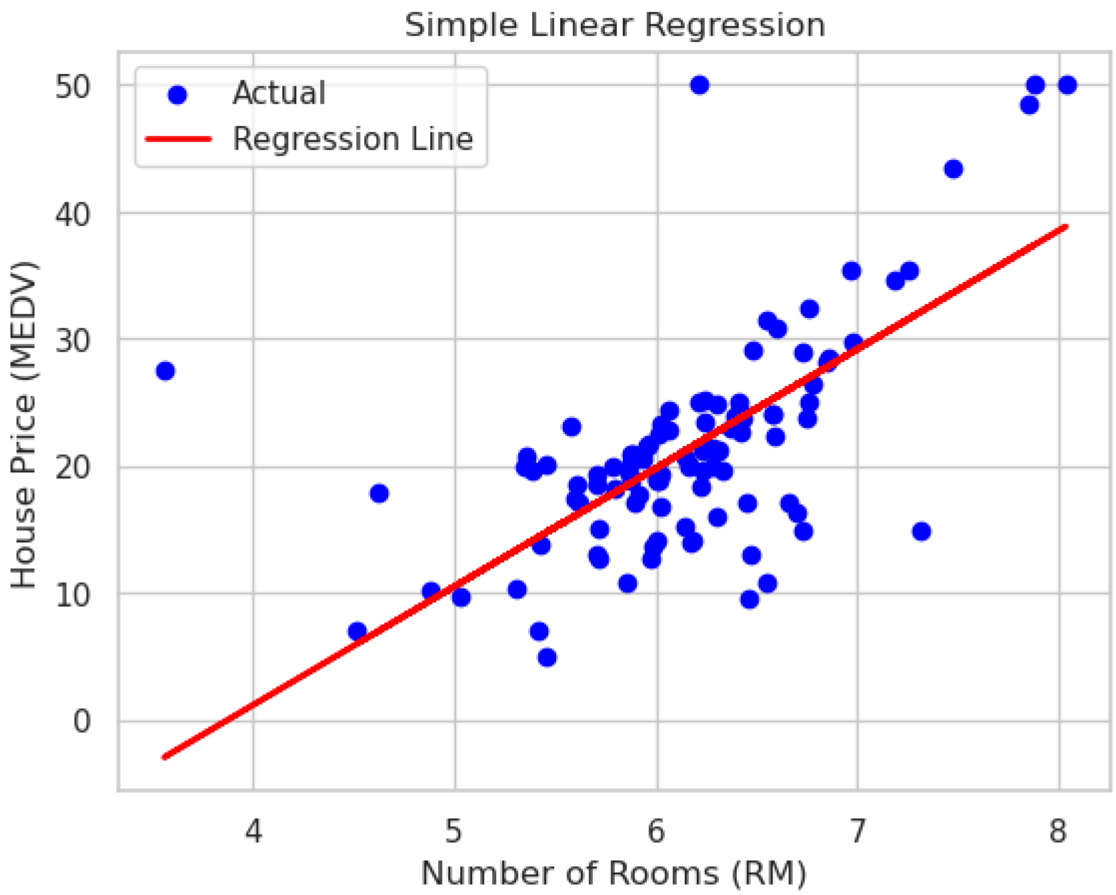
In [166…

```
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred)))
print("R2 Score:", r2_score(y_test, y_pred))
```

```
RMSE: 6.792994578778734
R2 Score: 0.3707569232254778
```

In [168…

```
plt.scatter(X_test, y_test, color='blue', label="Actual")
plt.plot(X_test, y_pred, color='red', linewidth=2, label="Regression Line
plt.xlabel("Number of Rooms (RM)")
plt.ylabel("House Price (MEDV)")
plt.title("Simple Linear Regression")
plt.legend()
plt.show()
```



In [ ]: