assgnment_4_Nishant

February 3, 2026

[7]:
```
import numpy as np
import pandas as pd
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 r2_score, mean_squared_error
```

[11]:
```
df = pd.read_csv("HousingData.csv")
df.head()
```

[11]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	\
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1	296	15.3	
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2	242	17.8	
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2	242	17.8	
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3	222	18.7	
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3	222	18.7	

	B	LSTAT	MEDV
0	396.90	4.98	24.0
1	396.90	9.14	21.6
2	392.83	4.03	34.7
3	394.63	2.94	33.4
4	396.90	NaN	36.2

[13]:
```
features = df.columns.drop("MEDV")
target = "MEDV"
```

[15]:
```
df.isnull().sum()
```

[15]:
```
CRIM       20
ZN         20
INDUS      20
CHAS       20
NOX         0
RM          0
```

```
AGE        20
DIS         0
RAD         0
TAX         0
PTRATIO     0
B           0
LSTAT      20
MEDV        0
dtype: int64
```

[6]:
```
df = df.dropna()
```

[7]:
```
df.isnull().sum()
```

[7]:
```
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
```

[8]:
```
features = df.columns.drop("MEDV")
target = "MEDV"

results = []

for feature in features:
    X = df[[feature]]
    y = df[target]

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

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

    y_pred = model.predict(X_test)
```

```
    r2 = r2_score(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)

    results.append([feature, r2, mse])
```

[9]:
```
results_df = pd.DataFrame(
    results, columns=["Feature", "R2 Score", "Mean Squared Error"]
)

results_df.sort_values(by="R2 Score", ascending=False)
```

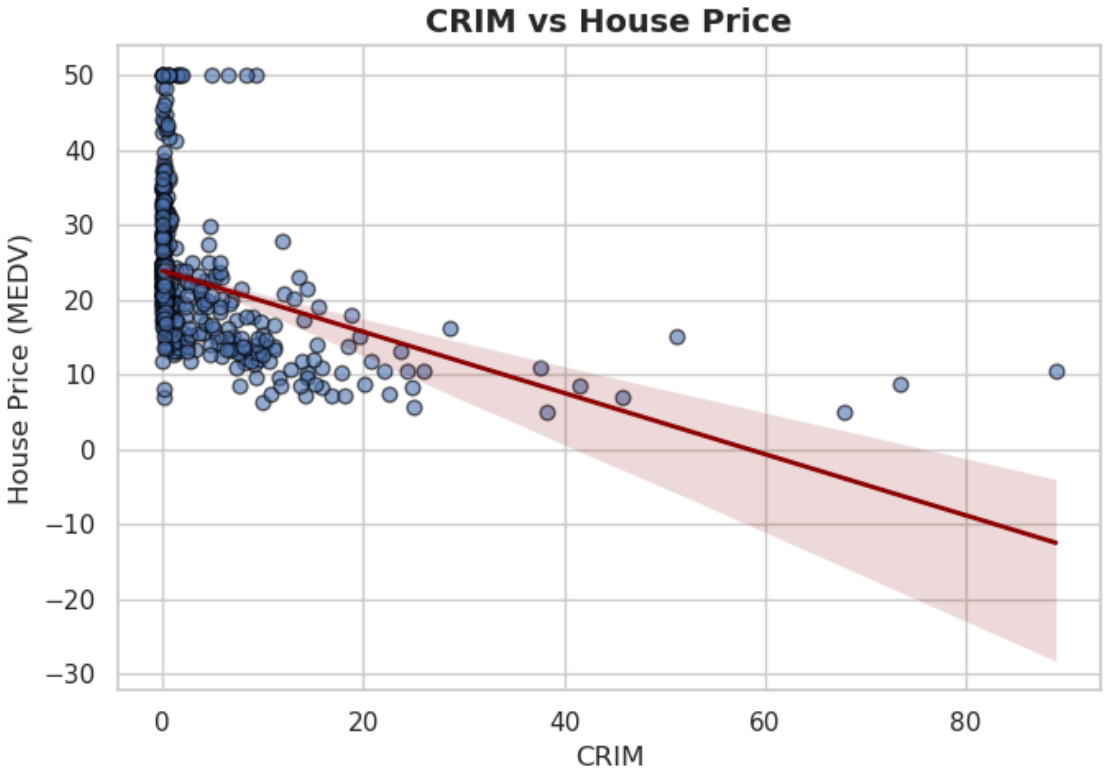
[9]:

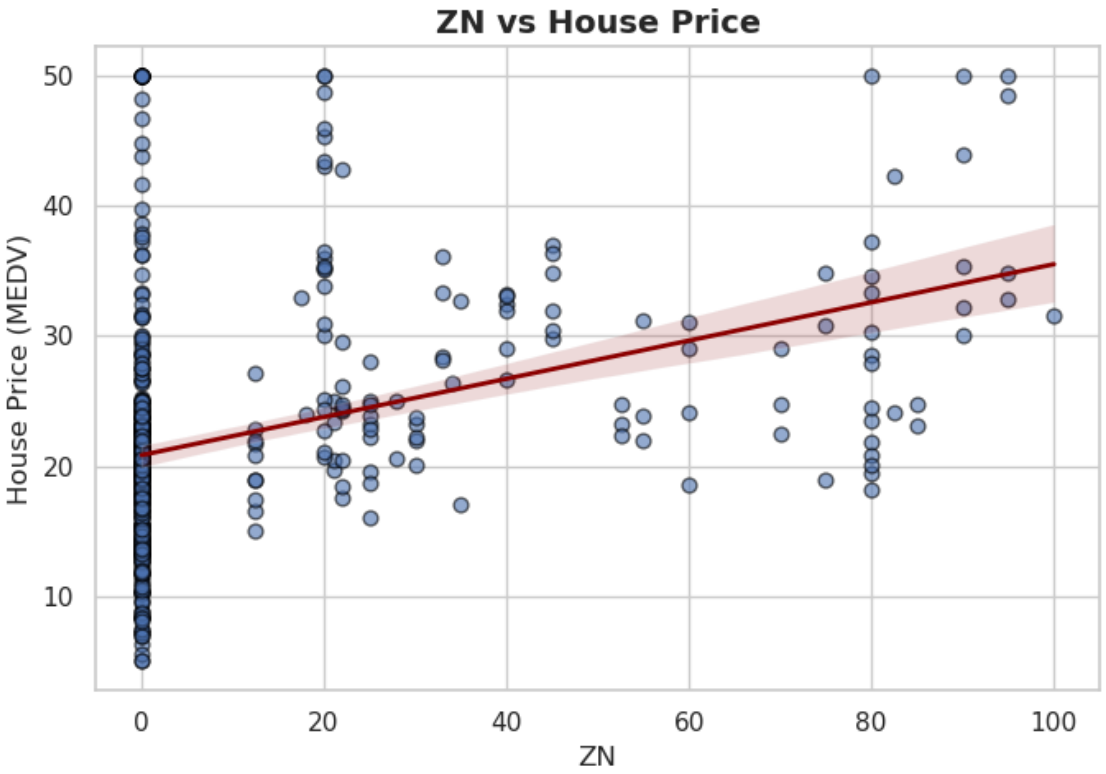
	Feature	R2 Score	Mean Squared Error
5	RM	0.478680	43.971498
12	LSTAT	0.416253	49.236978
10	PTRATIO	0.172987	69.755573
2	INDUS	0.119382	74.276944
1	ZN	0.103697	75.599982
4	NOX	0.075738	77.958155
11	B	0.075087	78.013084
9	TAX	0.046167	80.452434
0	CRIM	0.041634	80.834726
6	AGE	0.003567	84.045536
3	CHAS	-0.023619	86.338587
8	RAD	-0.065704	89.888294
7	DIS	-0.081723	91.239499

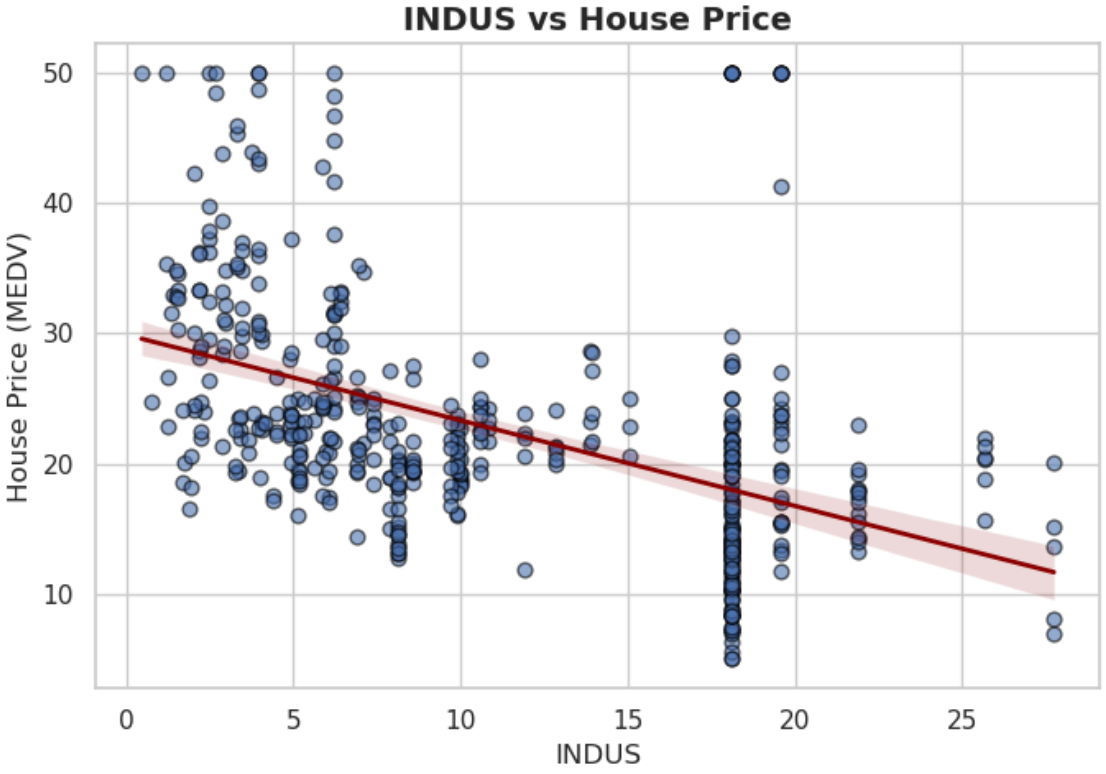
[17]:
```
import matplotlib.pyplot as plt
import seaborn as sns

target = "MEDV"
features = df.drop(columns=[target]).columns
```

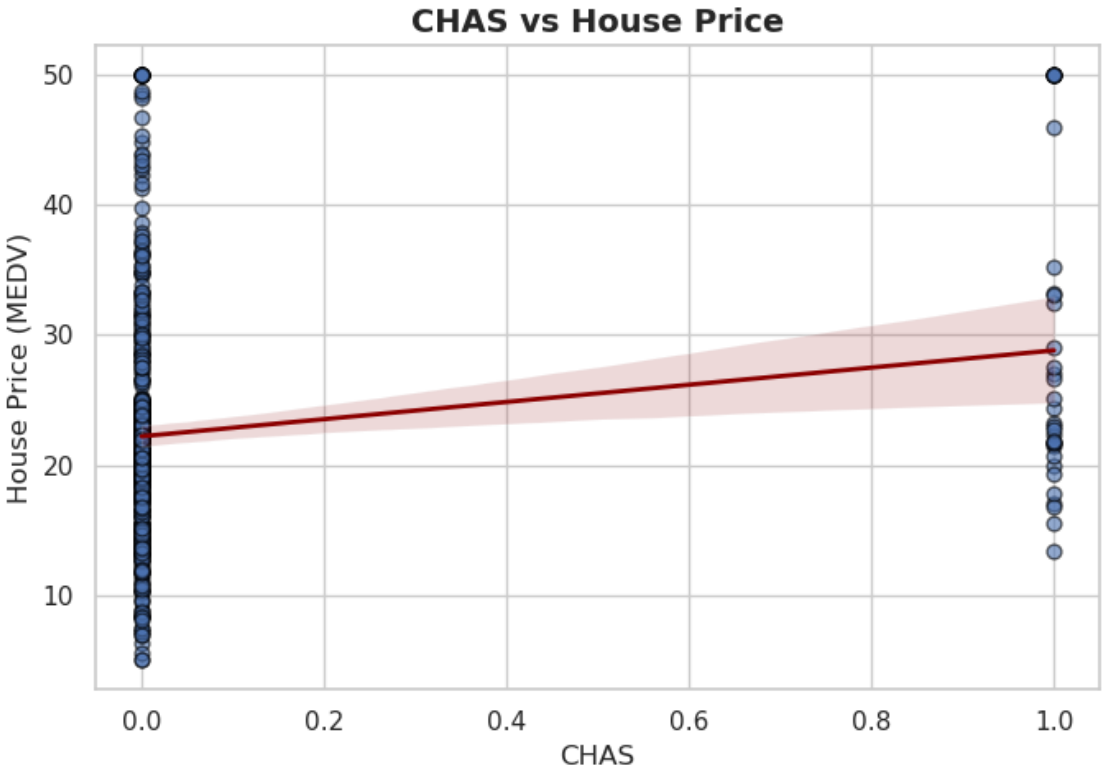
[19]:
```
sns.set_theme(style="whitegrid", context="notebook")

for feature in features:
    plt.figure(figsize=(7, 5))

    sns.regplot(
        x=df[feature],
        y=df[target],
        scatter_kws={
            "alpha": 0.6,
            "s": 40,
            "edgecolor": "black"
        },
        line_kws={
```

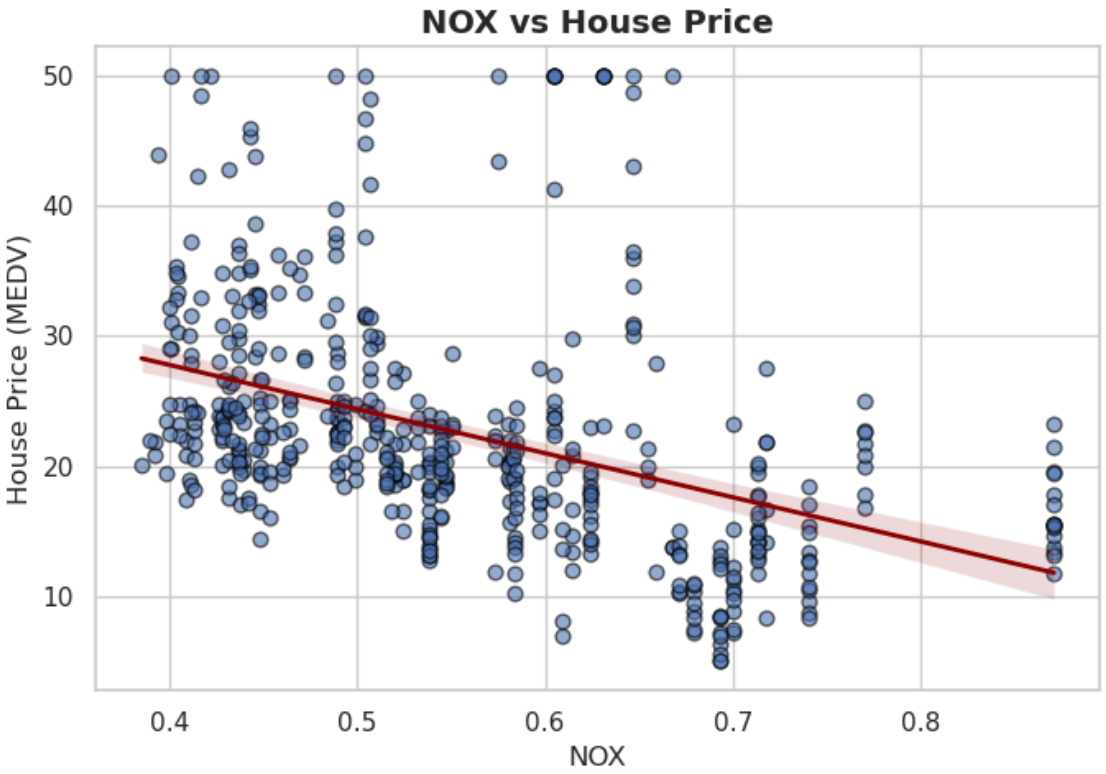
```
            "color": "darkred",
            "linewidth": 2
        }
    )

    plt.xlabel(feature, fontsize=12)
    plt.ylabel("House Price (MEDV)", fontsize=12)
    plt.title(f"{feature} vs House Price", fontsize=14, fontweight="bold")

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

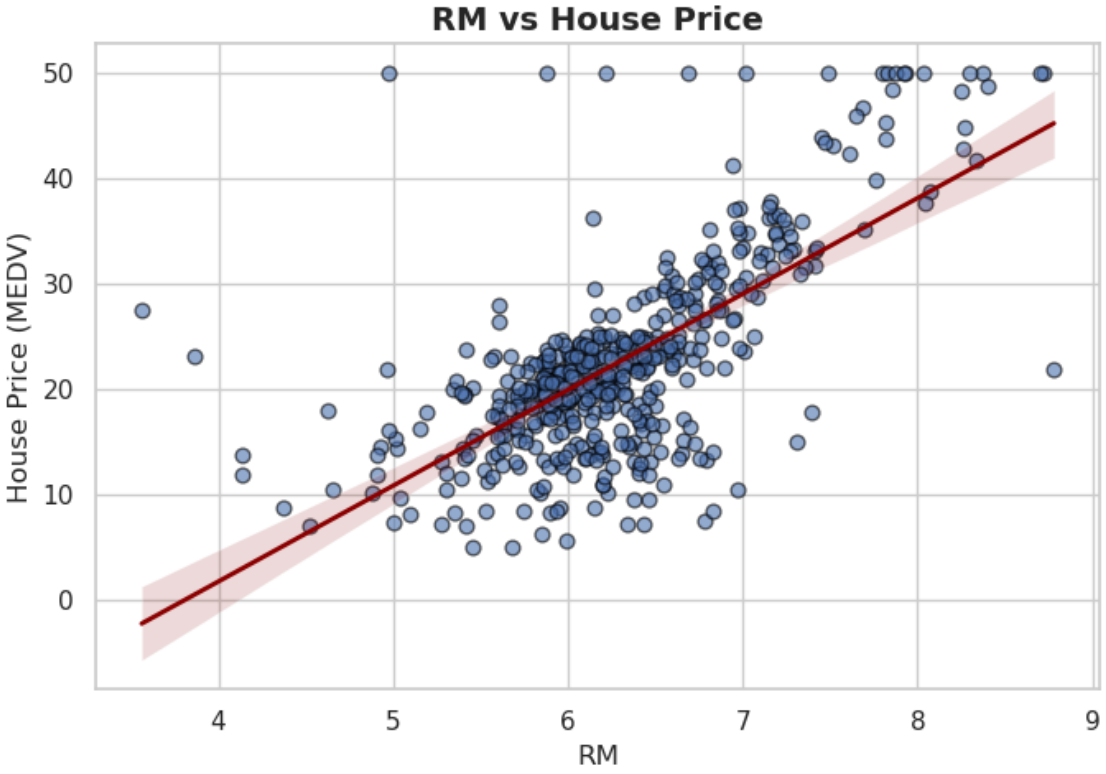


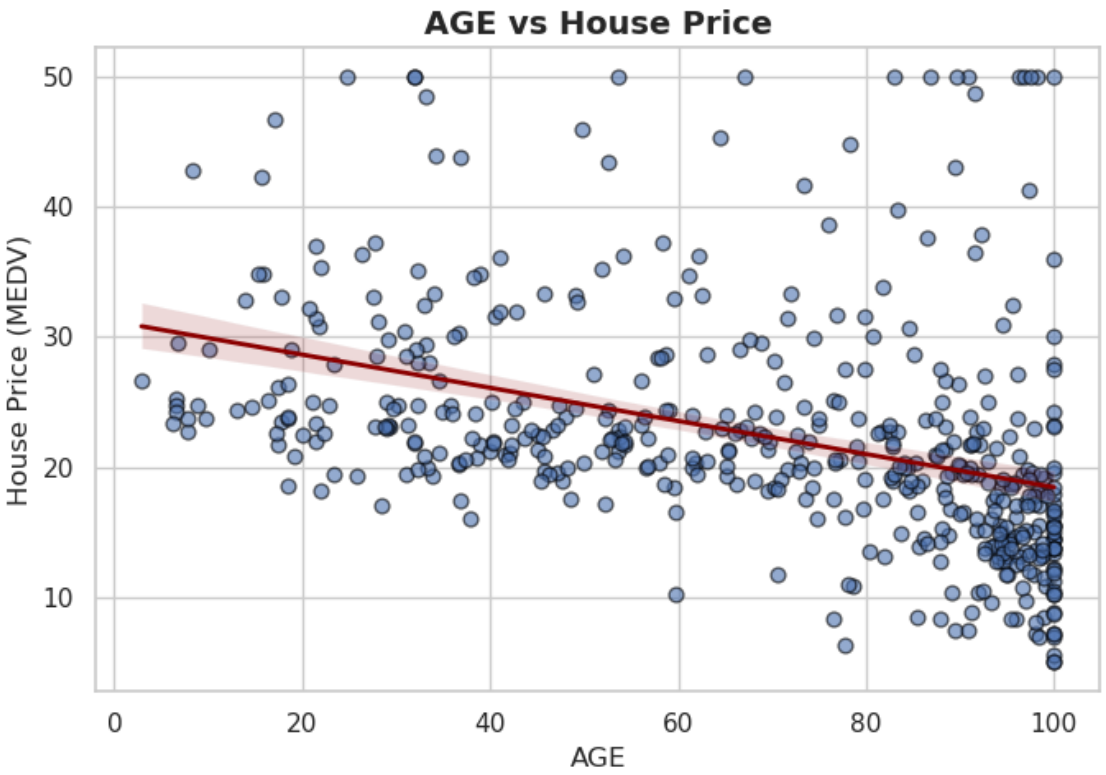

ZN vs House Price
House Price (MEDV)
ZN
0
20
40
60
80
100
10
20
30
40
50

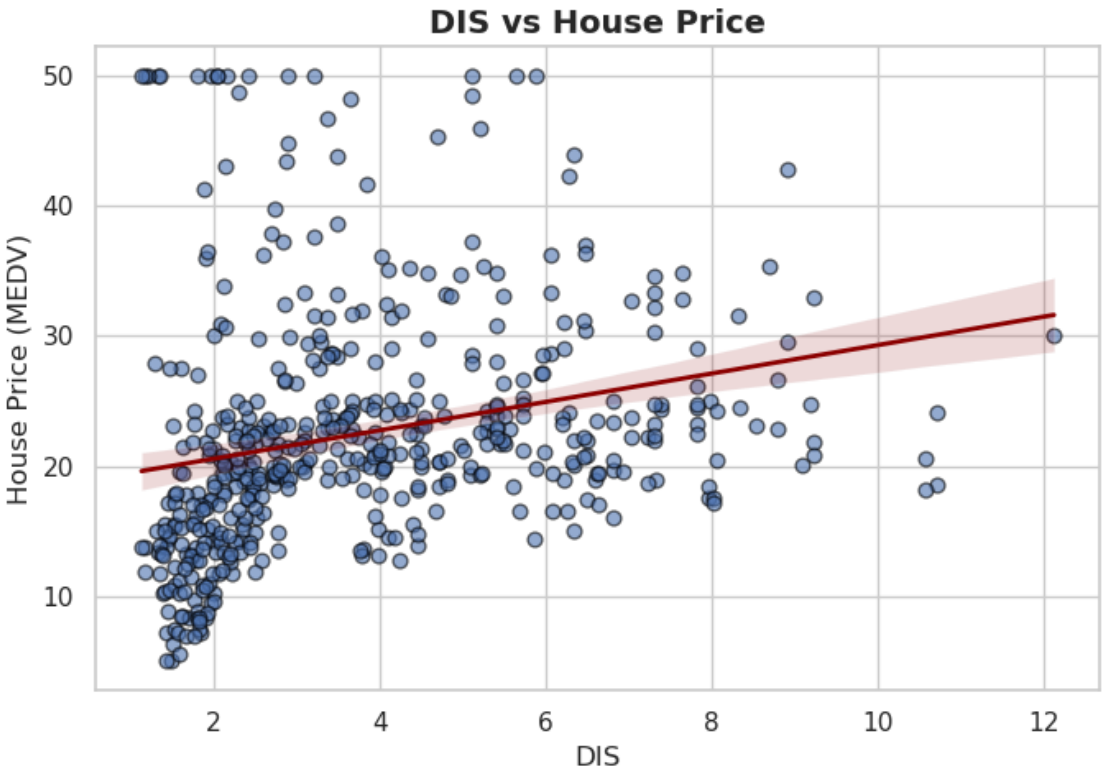

INDUS vs House Price
House Price (MEDV)
INDUS
0
5
10
15
20
25
10
20
30
40
50

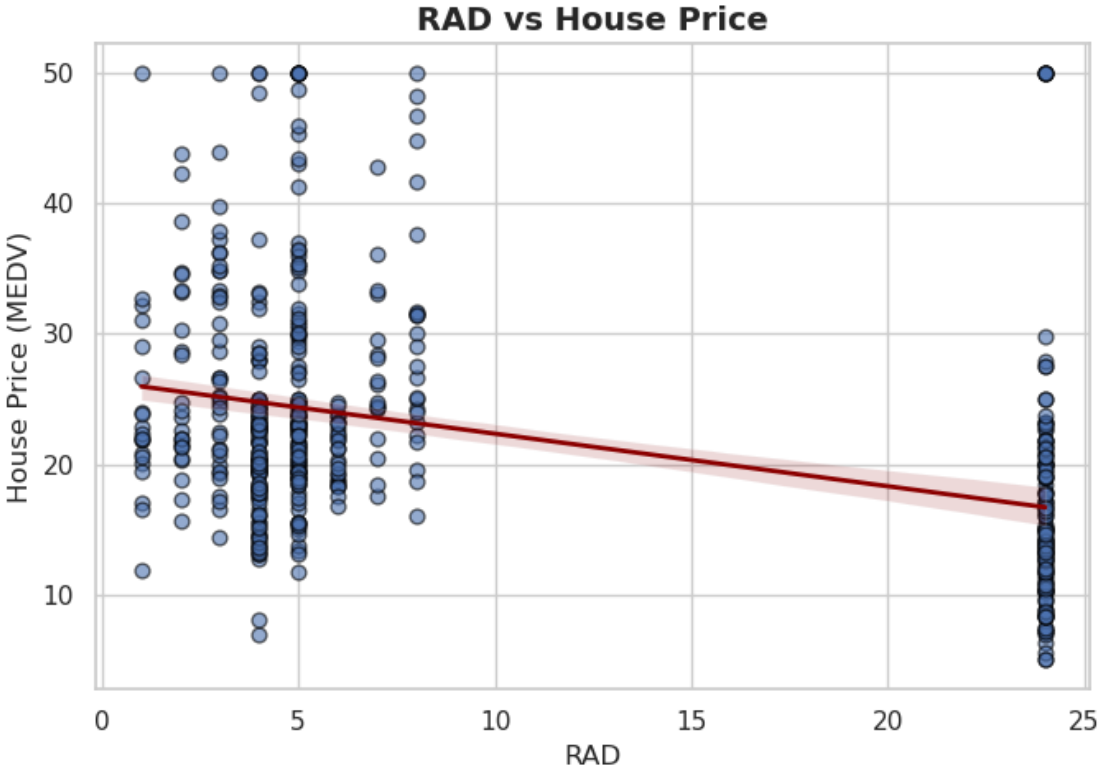

CHAS vs House Price
House Price (MEDV)
50
40
30
20
10
0.0
0.2
0.4
0.6
0.8
1.0
CHAS

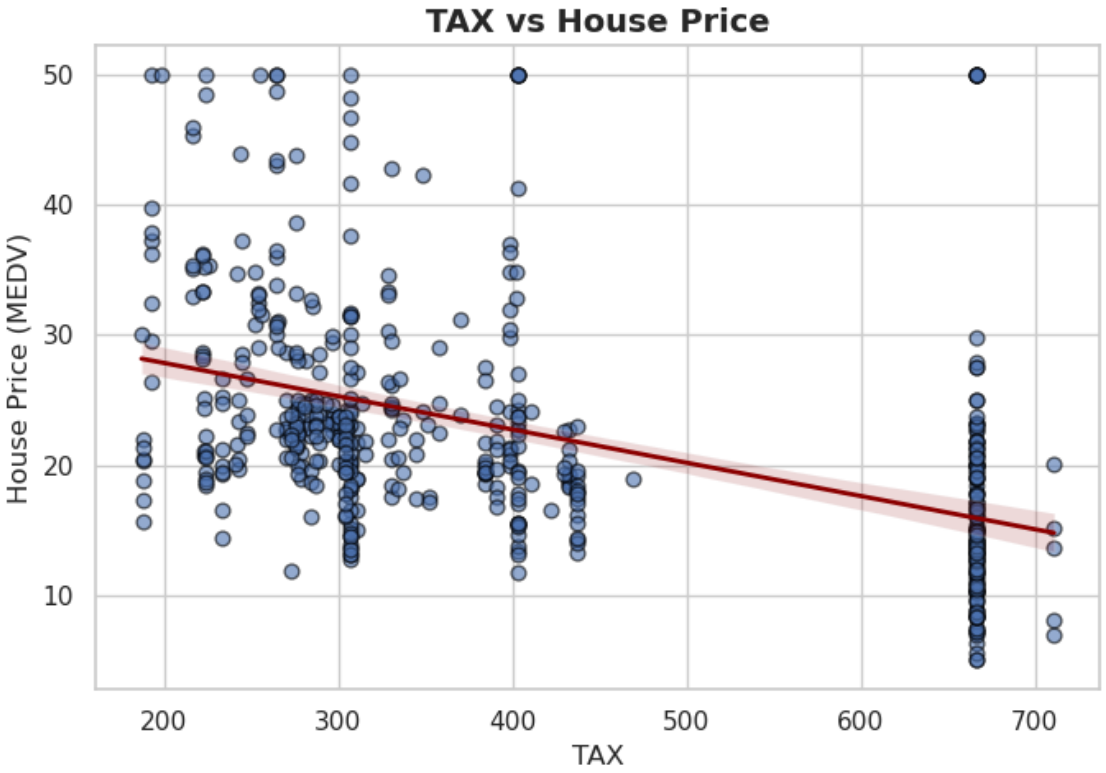

NOX vs House Price
House Price (MEDV)
50
40
30
20
10
0.4
0.5
0.6
0.7
0.8
NOX

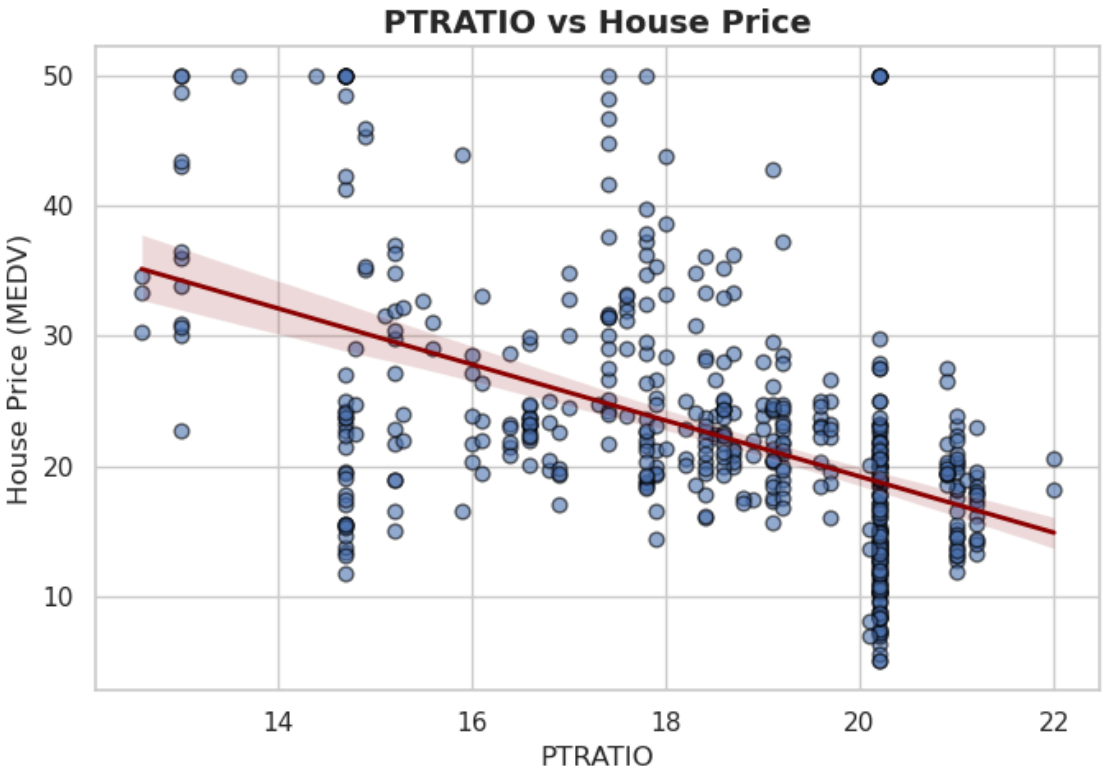

RM vs House Price
House Price (MEDV)
50
40
30
20
10
0
4
5
6
7
8
9
RM

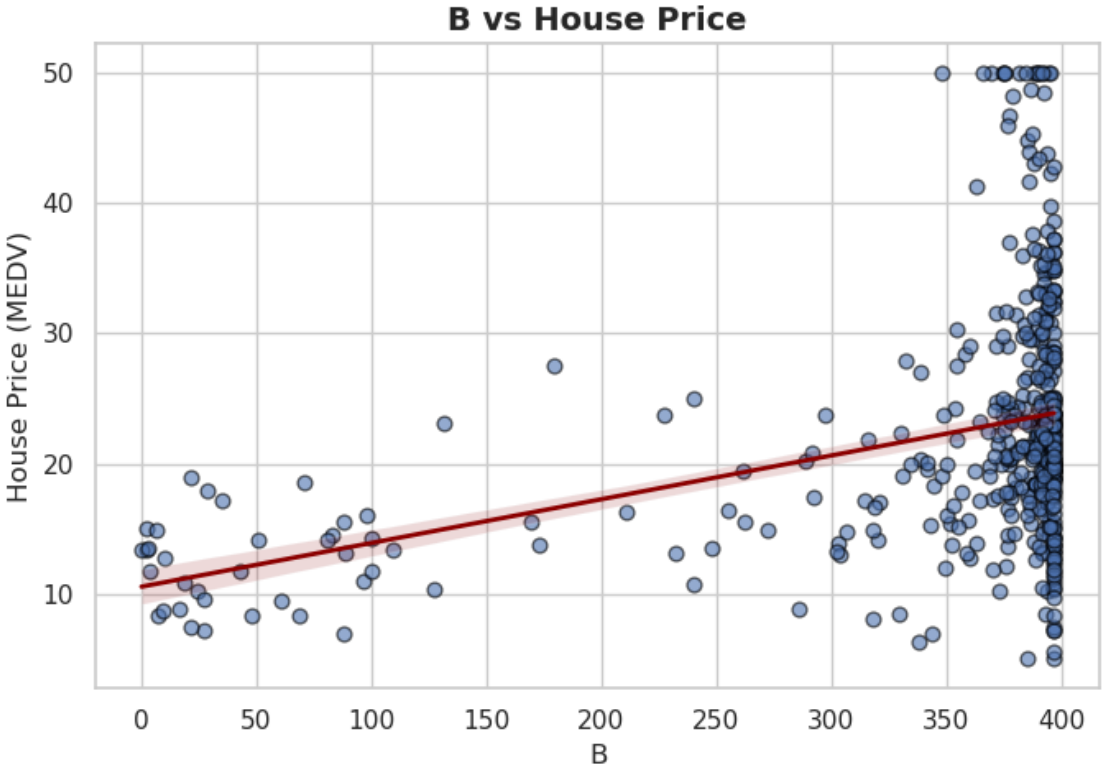

AGE vs House Price
House Price (MEDV)
AGE
50
40
30
20
10
0
20
40
60
80
100

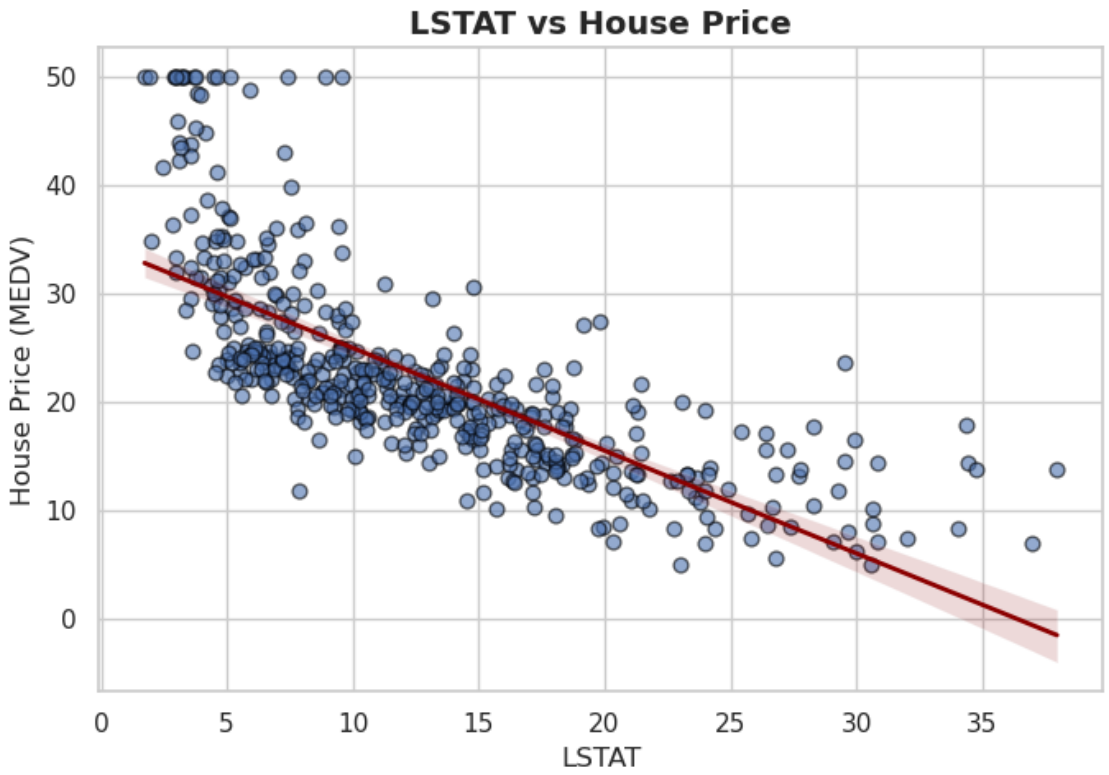

DIS vs House Price
House Price (MEDV)
DIS
50
40
30
20
10
2
4
6
8
10
12


RAD vs House Price
House Price (MEDV)
RAD
50
40
30
20
10
0
5
10
15
20
25


TAX vs House Price
House Price (MEDV)
50
40
30
20
10
200
300
400
500
600
700
TAX


PTRATIO vs House Price
House Price (MEDV)
PTRATIO
50
40
30
20
10
14
16
18
20
22


B vs House Price
House Price (MEDV)
B
50
40
30
20
10
0
50
100
150
200
250
300
350
400



[31]:
```
X = df.drop("MEDV", axis=1)
y = df["MEDV"]
```

[45]:
```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

[47]:
```
from sklearn.impute import SimpleImputer

imputer = SimpleImputer(strategy="mean")

X_train = imputer.fit_transform(X_train)
X_test = imputer.transform(X_test)
```

[49]:
```
from sklearn.linear_model import LinearRegression

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

[49]:
```
LinearRegression()
```

[51]:
```
y_pred = multi_model.predict(X_test)

r2_multi = r2_score(y_test, y_pred)
mse_multi = mean_squared_error(y_test, y_pred)

print("R2 Score (Multiple Regression):", r2_multi)
print("Mean Squared Error:", mse_multi)
```

```
R2 Score (Multiple Regression): 0.659060424186024
Mean Squared Error: 25.00238892351461
```

[15]:
```
importance = pd.DataFrame({
    "Feature": X.columns,
    "Coefficient": multi_model.coef_
})

importance.sort_values(by="Coefficient", ascending=False)
```

[15]:

	Feature	Coefficient
5	RM	4.258091
3	CHAS	1.983837
8	RAD	0.235588
1	ZN	0.042440
2	INDUS	0.025673
11	B	0.009594
9	TAX	-0.011997
6	AGE	-0.021741
0	CRIM	-0.112187
12	LSTAT	-0.388620
10	PTRATIO	-0.975835
7	DIS	-1.424189
4	NOX	-17.079257

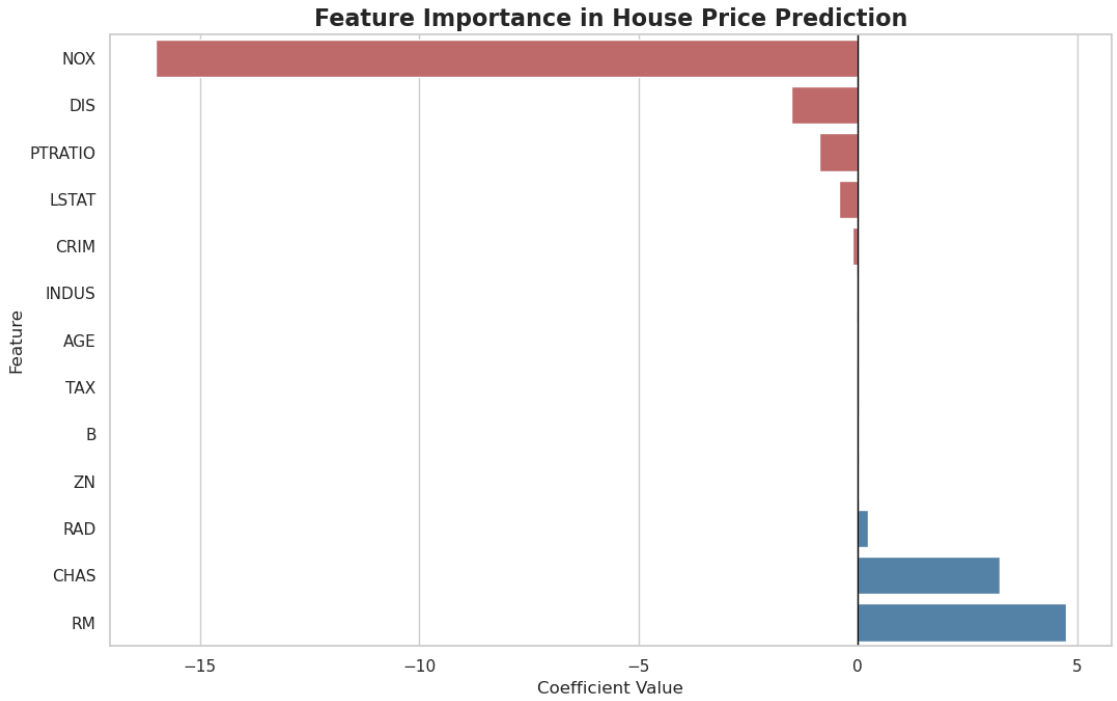
[55]:
```
from sklearn.linear_model import LinearRegression

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

[55]:
```
LinearRegression()
```

[57]:
```
import pandas as pd

importance = pd.DataFrame({
    "Feature": X.columns,
    "Coefficient": multi_model.coef_.flatten()
```

```
})
```

[61]:
```
plt.figure(figsize=(11, 7))

importance_sorted = importance.sort_values(by="Coefficient")

# Create a hue column
importance_sorted["Sign"] = importance_sorted["Coefficient"].apply(
    lambda x: "Positive" if x > 0 else "Negative"
)

sns.barplot(
    data=importance_sorted,
    x="Coefficient",
    y="Feature",
    hue="Sign",
    palette={"Positive": "steelblue", "Negative": "indianred"},
    legend=False
)

plt.axvline(0, color="black", linewidth=1)

plt.title("Feature Importance in House Price Prediction", fontsize=16,␣
 ↪fontweight="bold")
plt.xlabel("Coefficient Value")
plt.ylabel("Feature")

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



[17]:
```
best_part_a = results_df.sort_values(by="R2 Score", ascending=False).iloc[0]
best_part_a
```

[17]:
```
Feature                     RM
R2 Score                0.47868
Mean Squared Error    43.971498
Name: 5, dtype: object
```

[18]:
```
comparison = pd.DataFrame({
    "Model": ["Best Single Feature (Part A)", "Multiple Features (Part B)"],
    "R2 Score": [best_part_a["R2 Score"], r2_multi],
    "Mean Squared Error": [best_part_a["Mean Squared Error"], mse_multi]
})

comparison
```

[18]:

	Model	R2 Score	Mean Squared Error
0	Best Single Feature (Part A)	0.478680	43.971498
1	Multiple Features (Part B)	0.627085	31.454048

[69]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
import pandas as pd
```

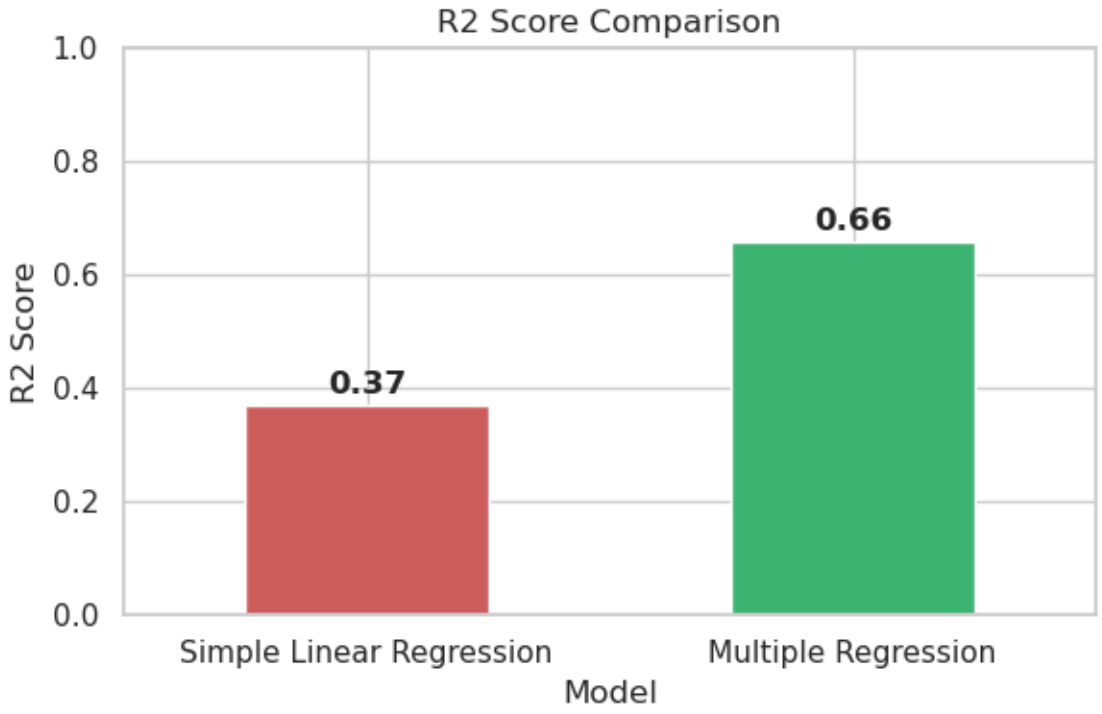
```
# Step 1: Convert X_train/X_test back to DataFrame if they were NumPy arrays
X_train = pd.DataFrame(X_train, columns=X.columns)
X_test = pd.DataFrame(X_test, columns=X.columns)

# Step 2: Simple linear regression with one feature, e.g., "RM"
X_simple_train = X_train[["RM"]]
X_simple_test = X_test[["RM"]]

simple_model = LinearRegression()
simple_model.fit(X_simple_train, y_train)

# Step 3: Predict
y_pred_simple = simple_model.predict(X_simple_test)

# Step 4: Compute R2
r2_simple = r2_score(y_test, y_pred_simple)
print("R2 Score (Simple Linear Regression):", r2_simple)
```

```
R2 Score (Simple Linear Regression): 0.3707569232254778
```

[71]:
```
comparison = pd.DataFrame({
    "Model": ["Simple Linear Regression", "Multiple Regression"],
    "R2 Score": [r2_simple, r2_multi]
})
```

[73]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Step 1: Create comparison DataFrame
comparison = pd.DataFrame({
    "Model": ["Simple Linear Regression", "Multiple Regression"],
    "R2 Score": [r2_simple, r2_multi]
})

# Step 2: Assign colors: green for the higher R2, red for lower
colors = ["mediumseagreen" if x == max(comparison["R2 Score"]) else "indianred"
          for x in comparison["R2 Score"]]

# Step 3: Plot
plt.figure(figsize=(6,4))
comparison.set_index("Model")["R2 Score"].plot(
    kind="bar",
    color=colors,
    title="R2 Score Comparison"
)

plt.ylabel("R2 Score")
```

```
plt.xticks(rotation=0)
plt.ylim(0,1)  # makes it easier to compare
plt.tight_layout()

# Step 4: Optional: annotate R2 values on top of bars
for i, v in enumerate(comparison["R2 Score"]):
    plt.text(i, v + 0.02, f"{v:.2f}", ha='center', fontweight='bold')

plt.show()
```



[21]:
```
best_feature = results_df.sort_values(
    by="R2 Score", ascending=False
).iloc[0]["Feature"]

best_feature
```

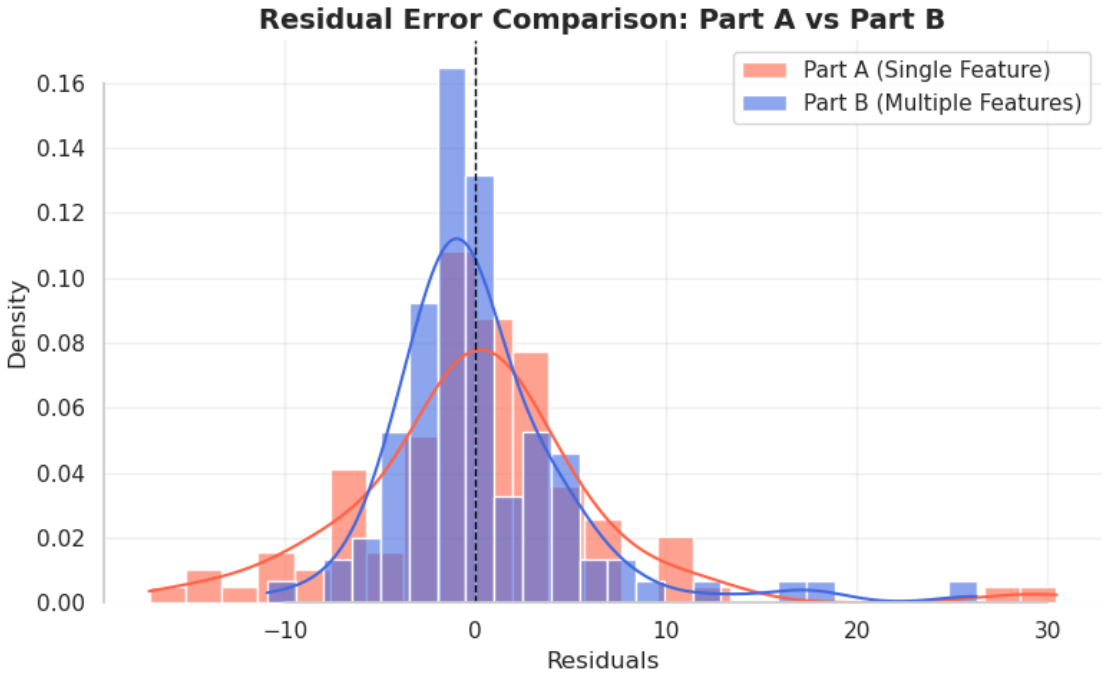
[21]: 'RM'

[22]:
```
X_best = df[[best_feature]]
y = df["MEDV"]

X_train_a, X_test_a, y_train_a, y_test_a = train_test_split(
    X_best, y, test_size=0.2, random_state=42
```

```
)

model_a = LinearRegression()
model_a.fit(X_train_a, y_train_a)

y_pred_a = model_a.predict(X_test_a)
```

[83]:
```
# Step 1: Compute residuals (Series)
residuals_a = y_test - y_pred_simple      # Single feature model
residuals_b = y_test - y_pred             # Multiple features model

# Step 2: Plot improved histogram
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(8,5))

sns.histplot(
    residuals_a,
    bins=25,
    color="tomato",
    label="Part A (Single Feature)",
    kde=True,
    stat="density",
    alpha=0.6
)

sns.histplot(
    residuals_b,
    bins=25,
    color="royalblue",
    label="Part B (Multiple Features)",
    kde=True,
    stat="density",
    alpha=0.6
)

# Add vertical line at 0
plt.axvline(0, color="black", linestyle="--", linewidth=1)

# Labels and title
plt.xlabel("Residuals")
plt.ylabel("Density")
plt.title("Residual Error Comparison: Part A vs Part B", fontsize=14,␣
 ↪fontweight="bold")
plt.legend()
sns.despine(trim=True)
```

```
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
```



[ ]: