House Price Prediction using Linear Regression Model

By Meenakshi Sharad Sethi

Disclaimer : This project is done on an individual capacity as part of academic learning and is not meant for commercial or professional use without further validation and refinement.

Introduction:

This project aims to build a linear regression model to predict house prices using various features such as average area income, age of the houses, number of rooms, number of bedrooms, and population. The dataset used for this project is the USA_housing.csv, which contains information about different areas or ZIP codes in the United States.

The project explores two main approaches:

1. Building a linear regression model without feature selection.
2. Building a linear regression model with feature selection techniques, including Recursive Feature Elimination (RFE) and correlation-based feature selection.

The code provided covers data preprocessing, splitting the data into training and testing sets, model fitting and model evaluation using metrics such as R-squared and Root Mean Squared Error (RMSE).

Potential Uses:

The analysis and models developed in this project can be useful in various scenarios related to the real estate industry, such as:

- ***Property valuation***: The linear regression model can be used to estimate the market value of residential properties based on the available features, assisting real estate agents, appraisers, and homeowners in pricing decisions.
- ***Investment analysis***: Investors and developers can leverage the model to identify potentially undervalued or overvalued properties in specific areas, aiding in investment decisions and portfolio management.
- ***Demographic analysis***: By analyzing the relationships between house prices and demographic features like population and income, urban planners and policymakers can gain insights into housing affordability and develop strategies to address housing market concerns.

Importing necessary libraries

In [28]:
```
import os
os.environ['OPENBLAS_NUM_THREADS'] = '5'
```

In [29]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [30]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from math import sqrt
import matplotlib.pyplot as plt
import seaborn as sns
```

In [31]:

```
# Load the dataset

df = pd.read_csv('USA_housing.csv')
df
```

Out[31]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Addres
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Ap 674\nLaurabury, N 3701.
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson View Suite 079\nLak Kathleen, CA.
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabet Stravenue\nDanieltowr WI 06482.
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO A 4482
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFP(AE 0938
...	...	...	...	...	...	...	.
4995	60567.944140	7.830362	6.137356	3.46	22837.361035	1.060194e+06	USNS Williams\nFP(AP 30153-765
4996	78491.275435	6.999135	6.576763	4.02	25616.115489	1.482618e+06	PSC 9258, Bo 8489\nAPO AA 42991 335
4997	63390.686886	7.250591	4.805081	2.13	33266.145490	1.030730e+06	4215 Tracy Garde Suite 076\nJoshualanc VA 01.
4998	68001.331235	5.534388	7.130144	5.44	42625.620156	1.198657e+06	USS Wallace\nFPO A 7331
4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1.298950e+06	37778 George Ridge Apt. 509\nEast Holl NV 2.

5000 rows × 7 columns

In [32]:

```
df.head()
```

Out[32]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Views Suite 079\nLake Kathleen, CA...
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabeth Stravenue\nDanieltown, WI 06482...
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO AP 44820
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFPO AE 09386

In [33]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   Avg. Area Income             5000 non-null   float64
 1   Avg. Area House Age          5000 non-null   float64
 2   Avg. Area Number of Rooms    5000 non-null   float64
 3   Avg. Area Number of Bedrooms 5000 non-null   float64
 4   Area Population              5000 non-null   float64
 5   Price                        5000 non-null   float64
 6   Address                      5000 non-null   object 
dtypes: float64(6), object(1)
memory usage: 273.6+ KB
```

In [34]:

```
# Rename columns
df = df.rename(columns={
    'Avg. Area Income': 'Income',
    'Avg. Area House Age': 'Age',
    'Avg. Area Number of Rooms': 'Rooms',
    'Avg. Area Number of Bedrooms': 'Bedrooms',
    'Area Population': 'Population'
})

# Display the dataframe with renamed column names
df.head()
```

Out[34]:

	Income	Age	Rooms	Bedrooms	Population	Price	Address
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Views Suite 079\nLake Kathleen, CA...

	Income	Age	Rooms	Bedrooms	Population	Price	Address
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabeth Stravenue\nDanieltown, WI 06482...
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO AP 44820
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFPO AE 09386

In [35]:
```
# Extract features (x) and target variable (y)

x = df[['Income', 'Age', 'Bedrooms', 'Rooms', 'Population']]
y = df['Price']
```

In [36]:
```
# Split the data into training and testing sets

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_sta
```

In [37]:
```
"""Simple linear regression"""

# Initializing linear regression model
lm = LinearRegression()

# Fitting the model on the training data
lm.fit(x_train, y_train)

# Finding and printing intercept and coefficients
print ("Intercept: ", lm.intercept_)
print ("Coefficient: ", lm.coef_)
print ("Coefficient:", lm.coef_.tolist())

# Making predictions on the test data
y_prediction = lm.predict(x_test)
```

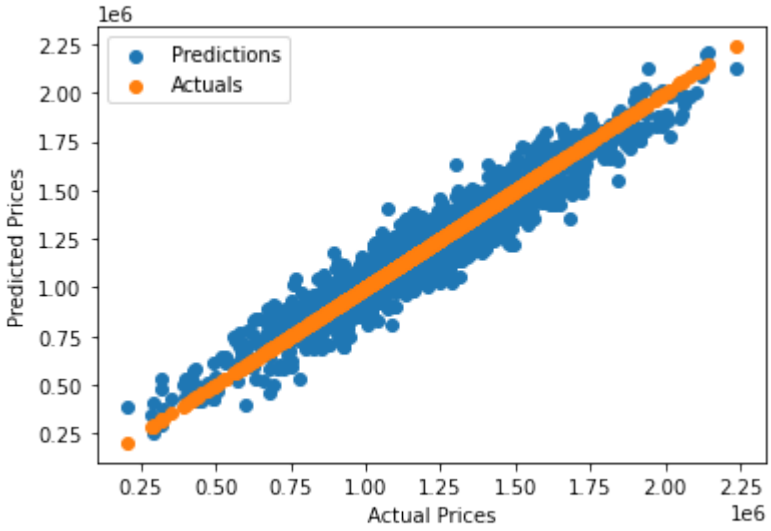
```
Intercept:  -2637802.975948643
Coefficient:  [2.15878084e+01 1.65754717e+05 2.03541647e+03 1.19343499e+05
 1.53756226e+01]
Coefficient: [21.587808369668696, 165754.71703855688, 2035.4164670963692, 119343.4994
8550854, 15.37562263465772]
```

In [43]:
```
# Evaluating the model

# Finding the R-Square Score
r2 = r2_score(y_test, y_prediction)

# Finding the RMSE Score
rmse = sqrt(mean_squared_error(y_test, y_prediction))

print("R-Square:", r2, "RMSE:", rmse)

# Plot predictions vs. actuals
plt.scatter(y_test, y_prediction, label="Predictions")
plt.scatter(y_test, y_test, label="Actuals")
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
plt.legend()
```

```
R-Square: 0.9181304271845878 RMSE: 100579.39402034934
```

Out[43]:

```
<matplotlib.legend.Legend at 0x1dab72a4e80>
```



In [45]:

```
# Example: Predicting the price for a new data point
new_data_point = [[240000, 25, 3, 5, 130000]]

# Predict the price using the trained model
predicted_price = lm.predict(new_data_point)

# Display the predicted price
print("Predicted Price for the new data point:", predicted_price)
```

```
Predicted Price for the new data point: [9288793.6480701]
```

Trying to see if the model performance can be improved with feature selection.

In [50]:

```
# Importing necessary library

from sklearn.feature_selection import RFE
```

In [52]:

```
# Use Recursive Feature Elimination (RFE)

select = RFE(lm, n_features_to_select=3) # Select the top 3 features
x_train_selected = select.fit_transform(x_train, y_train)
x_test_selected = select.transform(x_test)
```

In [53]:

```
'''Simple Regression Model with feature selection'''

# Fitting the model on the selected features
lm.fit(x_train_selected, y_train)

# Making predictions on the test data
y_pred_selected = lm.predict(x_test_selected)
```

In [56]:

```
# Evaluating model

# Finding the R-Square Score
r2_selected = r2_score(y_test, y_pred_selected)

# Finding the RMSE Score
```

```
rmse = sqrt(mean_squared_error(y_test, y_pred_selected))

print("R-Square with feature selection:", r2_selected, "RMSE with feature selection:
```

```
R-Square with feature selection: 0.33681061416044866 RMSE with feature selection: 286
263.8685336438
```

Insights

The goal of feature selection is to identify and retain the most relevant features for improved model performance. In a comparison between scenarios with and without feature selection:

Without Feature Selection: R-Square: 0.9181 RMSE: 100579.39

With Feature Selection: R-Square: 0.3368 RMSE: 286263.87

The decrease in R-squared and the increase in RMSE after feature selection suggest a suboptimal set of features, potentially failing to capture underlying data patterns effectively. Possible reasons include

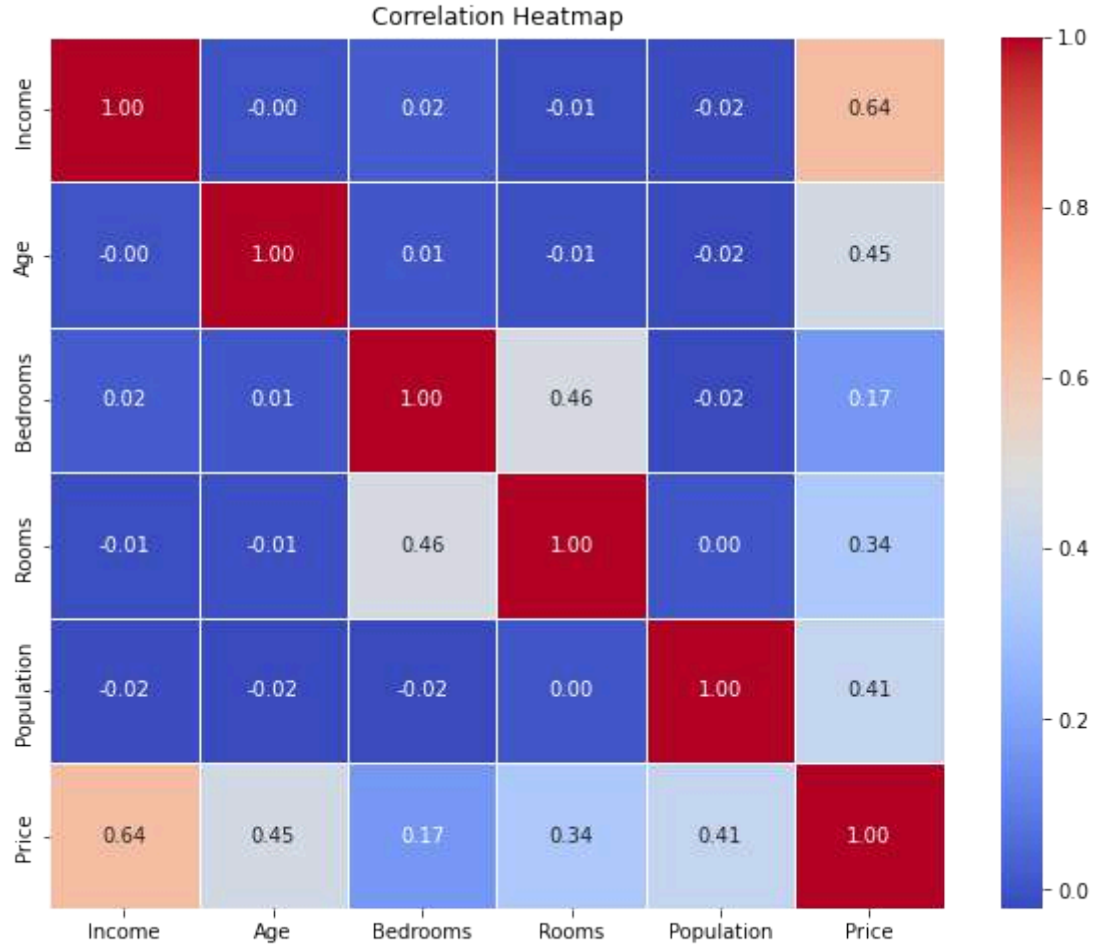
- the discarding of important features,
- a weak linear relationship, or
- a model oversimplified by fewer features.

Correlation-based Feature Selection

In [63]:

```
correlation_matrix = df[['Income', 'Age', 'Bedrooms', 'Rooms', 'Population', 'Price'

# Drop one of the highly correlated features
df_selected = df[['Income', 'Age', 'Rooms', 'Population', 'Price']]

print(correlation_matrix)

# Plot the correlation heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=.
plt.title('Correlation Heatmap')
plt.show()
```

```
                Income       Age  Bedrooms     Rooms  Population     Price
Income        1.000000 -0.002007  0.019788 -0.011032   -0.016234  0.639734
Age          -0.002007  1.000000  0.006149 -0.009428   -0.018743  0.452543
Bedrooms      0.019788  0.006149  1.000000  0.462695   -0.022168  0.171071
Rooms        -0.011032 -0.009428  0.462695  1.000000    0.002040  0.335664
Population   -0.016234 -0.018743 -0.022168  0.002040    1.000000  0.408556
Price         0.639734  0.452543  0.171071  0.335664    0.408556  1.000000
```



Insights from correlation matrix

Strong Positive Correlation: Income and Price have a strong positive correlation of approximately 0.64. This suggests that as the average area income increases, the average house price also tends to increase.

Moderate Positive Correlations: Age and Price have a moderate positive correlation of approximately 0.45. This indicates that, on average, older houses tend to have higher prices. Rooms and Price also have a moderate positive correlation of approximately 0.34. This suggests that houses with more rooms tend to have higher prices.

Weak Correlations: Bedrooms has a weak positive correlation (0.17) with Price. Population has a weak positive correlation (0.41) with Price.

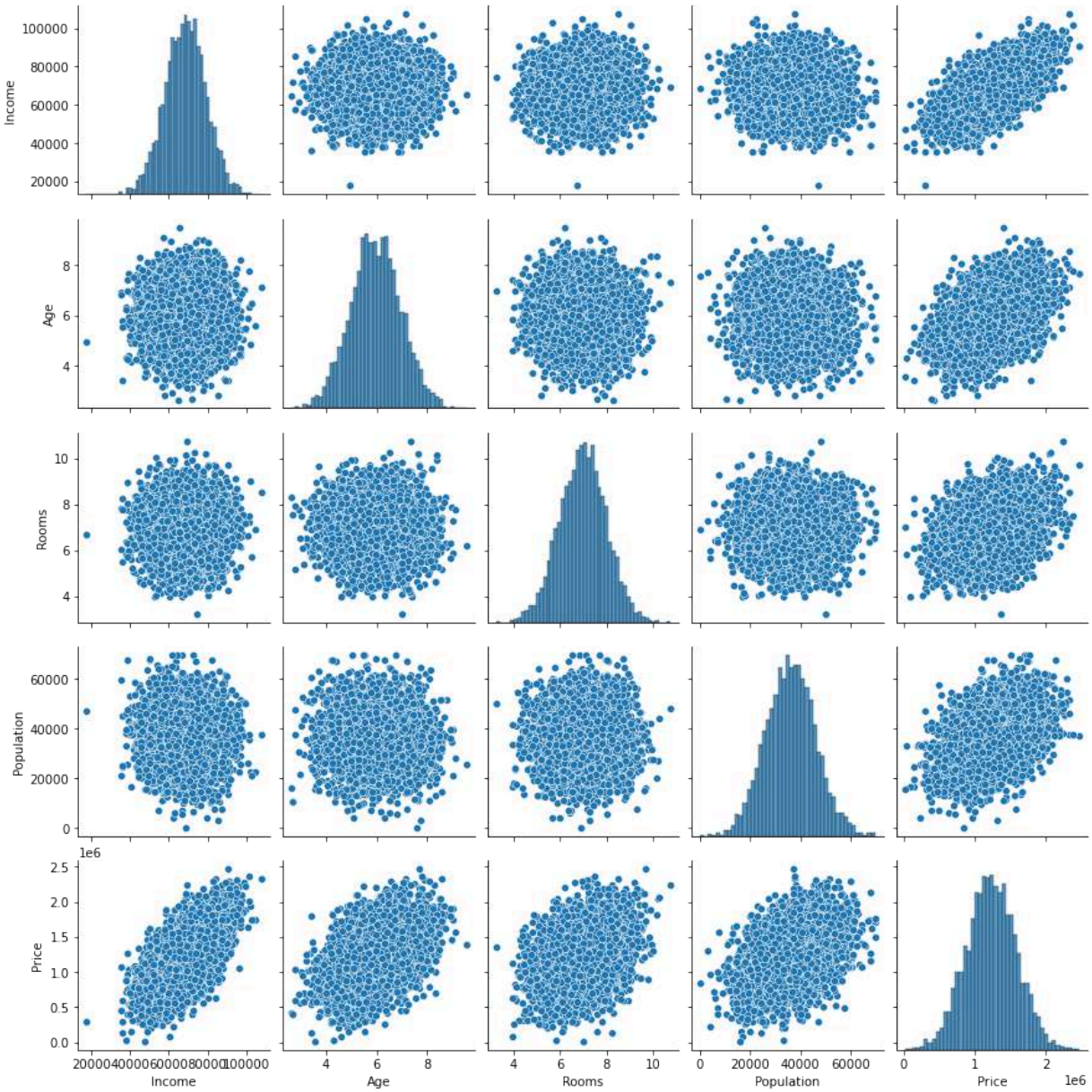
Weak Negative Correlations: There are weak negative correlations between Income and Age, Age and Rooms, Income and Population. However, these correlations are close to zero.

- Feature Selection: Considering the correlations, we decided to drop one of the highly correlated features to avoid multicollinearity. For example, you've chosen to drop Bedrooms from your selected features. Now, we can proceed to train linear regression model using the selected features (Income, Age, Rooms, Population) and evaluate its performance.

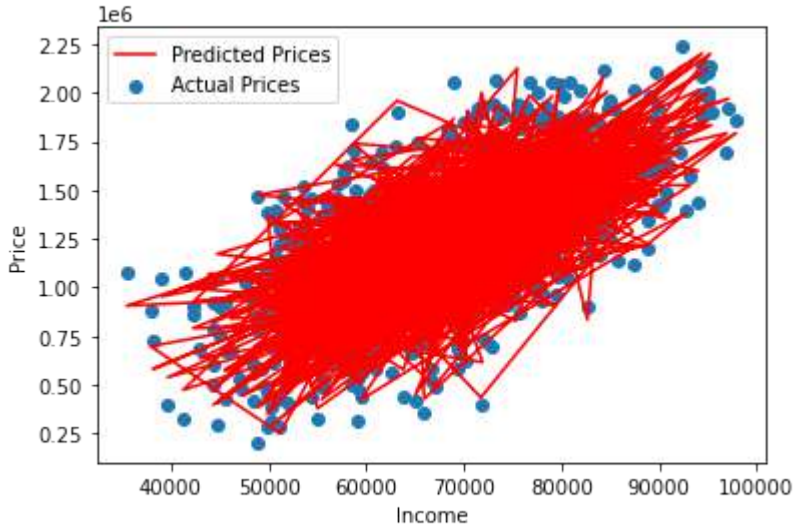
In [69]:

```
# Selected features
selected_features = ['Income', 'Age', 'Rooms', 'Population']

# Creating a pair plot
sns.pairplot(df[selected_features + ['Price']])
plt.show()
```

```
# Split the data into training and testing sets
x = df[selected_features]
y = df['Price']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_sta

# Initialize linear regression model
lm = LinearRegression()

# Fitting the model on the training data
lm.fit(x_train, y_train)

# Making predictions on the test data
y_pred = lm.predict(x_test)

# Evaluating new model
r2 = r2_score(y_test, y_pred)
rmse = sqrt(mean_squared_error(y_test, y_pred))

print("R-Square:", r2, "RMSE:", rmse)

# Plot the regression line for one feature (e.g., Income)
plt.scatter(x_test['Income'], y_test, label="Actual Prices")
plt.plot(x_test['Income'], y_pred, color='red', label="Predicted Prices")
plt.xlabel('Income')
plt.ylabel('Price')
plt.legend()
plt.show()
```




```
R-Square: 0.9181522262660747 RMSE: 100566.0026937565
```



Analysis

- The minor differences in R-squared and RMSE between the models with and without bedroom (using correlation-based feature selection) suggest that the inclusion or exclusion of the bedroom feature has a minimal impact on the overall predictive performance of the linear regression model.

- **Comparative Analysis:** ***With Bedroom (No Feature Selection):*** R-Square: 0.9181304271845878 RMSE: 100579.39402034934

Without Bedroom (Correlation-based Feature Selection): R-Square: 0.9181522262660747 RMSE: 100566.0026937565

- **R-Square:** Both models have very similar R-squared values, indicating that they explain a similar proportion of the variance in the target variable (house prices).
- **RMSE:** The Root Mean Squared Error is also very close in both cases, suggesting that the models make similar errors in predicting house prices.

Summary

The removal of the Bedrooms feature through correlation-based feature selection seems to have a minimal impact on model performance, and both models are comparable in their predictive ability.

Conclusion:

From the above following conclusions can be drawn:

1. **Without Feature Selection:** The linear regression model without feature selection performs well, with an R-squared of 0.9181 and an RMSE of 100579.39. This model captures the underlying patterns in the data effectively, indicating a strong linear relationship between the features and the target variable (house prices).

2. **Feature Selection with Recursive Feature Elimination (RFE):** The linear regression model with RFE-based feature selection (selecting the top 3 features) performs poorly, with an R-squared of 0.3368 and an RMSE of 286263.87. This suggests that the selected features may not be the most relevant or that the linear relationship is weakened by the reduced number of features.

3. **Correlation-based Feature Selection:** The correlation analysis revealed a strong positive correlation between Income and Price, as well as moderate positive correlations between Age, Rooms, and Price. After dropping the Bedrooms feature due to its relatively weak correlation with Price, the linear regression model without Bedrooms performed similarly to the model without feature selection.The R-squared was 0.9181, and the RMSE was 100566.00, indicating a minimal impact on the model's predictive performance.

In summary, the linear regression model without feature selection or with correlation-based feature selection (excluding Bedrooms) performed well in predicting house prices. However, the RFE-based feature selection approach did not improve the model's performance, potentially due to the removal of important features or the oversimplification of the linear model.

-- End --