D600 – Statistical Data Mining

Performance Assessment #1 – Linear Regression Analysis

Bernard Connelly

Master of Science, Data Analytics, Western Governors University

Dr. Keiona Middleton

January 24, 2025

D600 – Statistical Data Mining: Linear Regression Analysis

Purpose of Analysis

B1. Purpose of Research Question

The dataset presented provides a variety of metrics to review about specific houses in a housing market. Working under the assumption that this dataset comes from a realtor company or another interested party whose motivation is profit, a research question related to affecting the price of homes is most relevant. For this project, the research question is “How do the variables square_footage, num_bedrooms, crime_rate, fireplace, house_color, and garage impact the price of a home?”

B2. Goal of Analysis

To maximize the value of the dependent variable, price, a set of variables that represent a direct relationship to improving this value should be selected. This analysis aims to identify variables that have an apparent effect on increasing the price and profitability of homes. In doing so, suggestions can be made based on the company on which specific metrics are most important when acquiring new houses in the future to turn a maximum profit.

Summarization of Data Preparation

C1. Identification of Variables

The dependent variable in this scenario is ‘price,’ which is the monetary value at which the home was most recently sold. All other variables are independent variables, which can be used to explain the changes in price. It was recommended in the lectures that we only select six total variables and utilize them for the project. As there was no data dictionary, the variables could only be selected based on their titles, which I used to try and achieve a variety of options. For this project, I selected square_footage, num_bedrooms, crime_rate, fireplace, house_color, and garage as my independent variables. These were selected based upon a few assumptions – some were quantitative continuous (square_footage, crime_rate), others were quantitative discrete (num_bathrooms), some represented Boolean responses (garage and fireplace), and some were qualitative (house_color).

C2. Statistical Description of Variables

Below is a screenshot of the code and output descriptive statistics of each of the variables utilized for this analysis

```
#Variable Descriptions
print('Descriptive Statistics\n')
print('\nQuantitative Variables\n')#Showing Mean, Standard Deviation, and 5-number summaries
print('Price')
print(df['price'].describe().apply(
    lambda x: np.format_float_positional(x, precision=2, trim='-')))#Coverting to actual values as scientific notation was more difficult to read
print('\nSquare Footage')
print(df['square_footage'].describe())
print('\nNumber of Bedrooms')
print(df['num_bedrooms'].describe())
print('\nCrime Rate')
print(df['crime_rate'].describe())
print('\n\nQualitative Variables\n')#Showing frequencies in descending order
print(df['fireplace'].value_counts())
print('\n')
print(df['house_color'].value_counts())
print('\n')
print(df['garage'].value_counts())
```

```
Descriptive Statistics


Quantitative Variables

Price
count          7000
mean      307281.52
std       150173.44
min           85000
25%        192107.5
50%        279322.5
75%       391877.75
max         1046675
Name: price, dtype: object

Square Footage
count    7000.000000
mean     1048.947459
std       426.010482
min       550.000000
25%       660.815000
50%       996.320000
75%      1342.292500
max      2874.700000
Name: square_footage, dtype: float64

Number of Bedrooms
count    7000.000000
mean        3.008571
std         1.021940
min         1.000000
25%         2.000000
50%         3.000000
75%         4.000000
max         7.000000
Name: num_bedrooms, dtype: float64

Crime Rate
count    7000.000000
mean       31.226194
std        18.025327
min         0.030000
25%        17.390000
50%        30.385000
75%        43.670000
max        99.730000
Name: crime_rate, dtype: float64
```

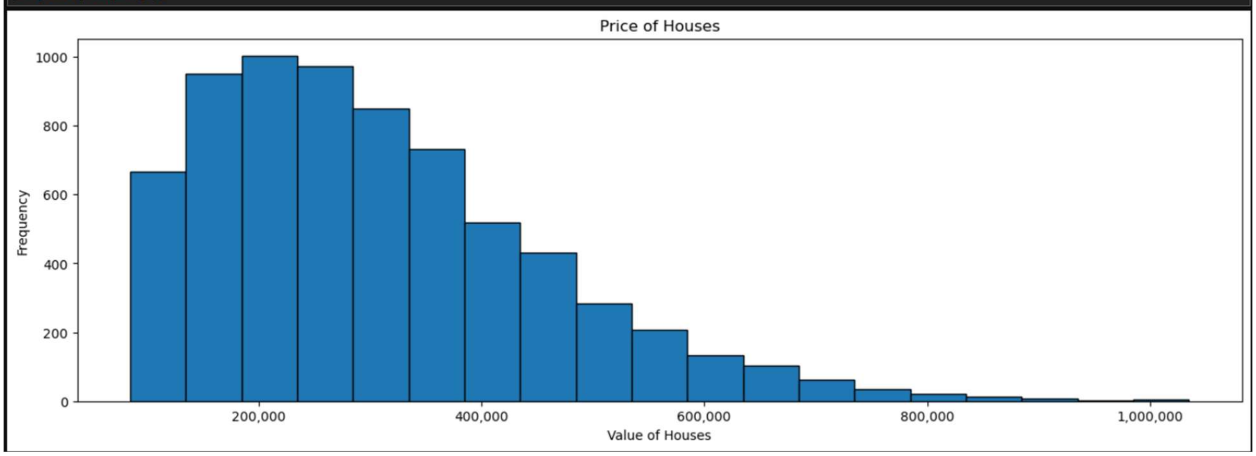
```
Qualitative Variables

fireplace
0    5172
1    1828
Name: count, dtype: int64


house_color
White     1446
Yellow    1423
Blue      1409
Green     1375
Red       1347
Name: count, dtype: int64


garage
0    4488
1    2512
Name: count, dtype: int64
```

The above descriptive statistics outline the counts, means, standard deviations, min/max and quartiles for the quantitative data visible above, along with raw counts and distributions of the categorical data. Notably, the price column was formatted to be more readable using the lambda function (ChatGPT, OpenAI, 2025). This permits the values to be more readable overall to the user.

C3. Statistical Visualizations

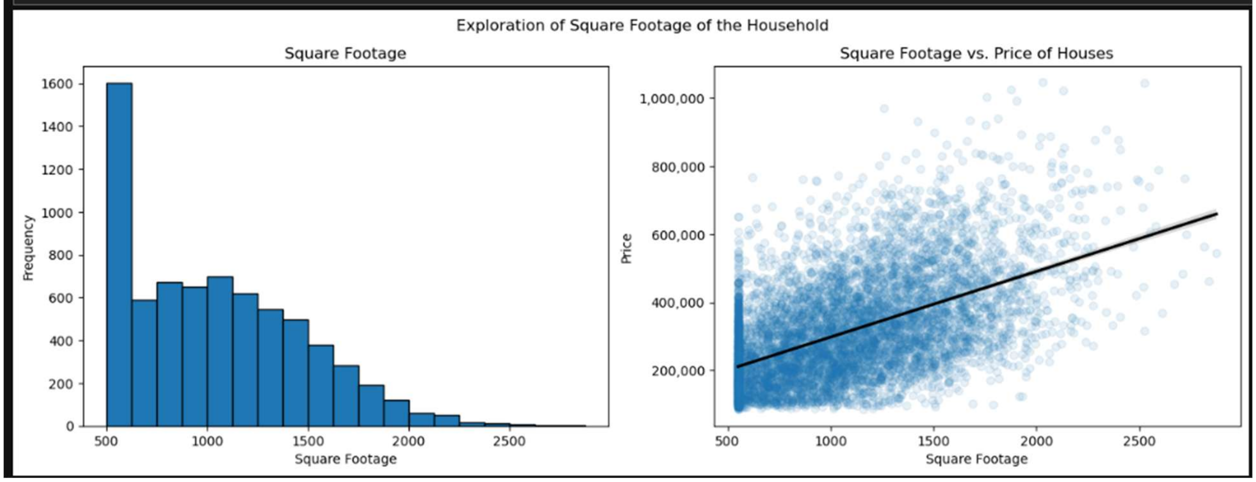
Below are the univariate and bivariate visualizations for the variables selected in the analysis. Matplotlib's tick formatters were utilized to properly format the y-axes for the charts below by following guidance from Tutorialspoint (n.d.).

```
#Price - Dependant Variable
##Univariate Analysis
plt.figure(figsize = [16,5])
plt.title("Price of Houses")
bins = np.arange(85000, 1050000,50000)
plt.hist(data=df, x='price', bins=bins, edgecolor='black', density=False)
plt.gca().xaxis.set_major_formatter(mticker.StrMethodFormatter('{x:,.0f}'))
plt.xlabel("Value of Houses")
plt.ylabel("Frequency");
```



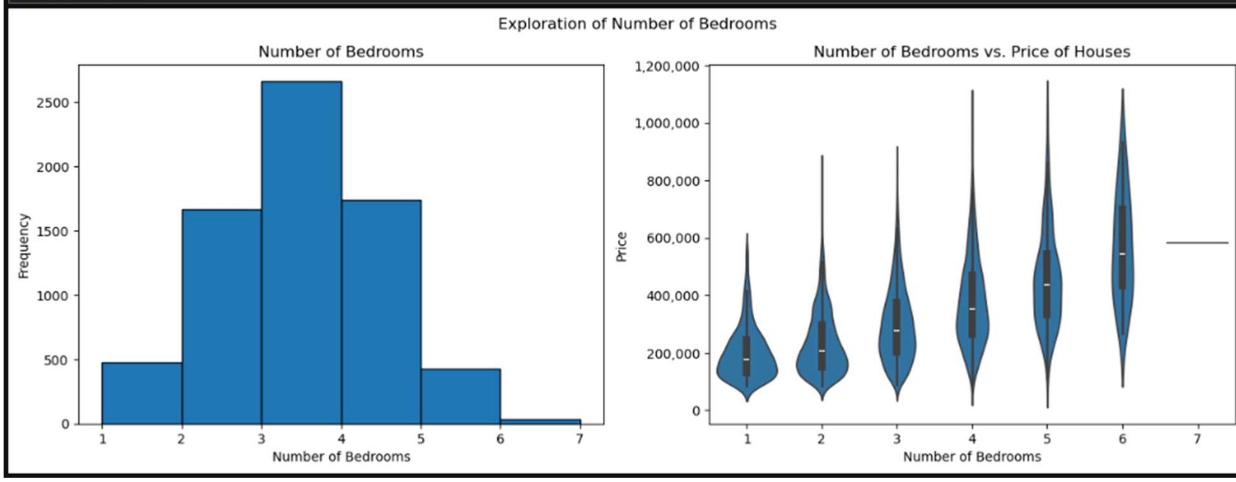
The price variable demonstrates a right-skewed distribution. This can be sub-optimal for a multiple regression analysis, so the residuals will be checked after the regression model is completed to ensure viability.

```python
#Square Footage - Independent Variable
##Univariate and Bivariate Analysis
plt.figure(figsize = [16,5])
plt.suptitle("Exploration of Square Footage of the Household")

#Left Plot: Univariate exploration of square_footage
plt.subplot (1, 2, 1)
plt.title("Square Footage")
bins = np.arange(500, 2900, 125)
plt.hist(data=df, x='square_footage', bins=bins, edgecolor='black', density=False)
plt.xlabel("Square Footage")
plt.ylabel("Frequency")

#Right Plot: Bivariate exploration of square_footage vs price
plt.subplot(1, 2, 2)
plt.title("Square Footage vs. Price of Houses")
sns.regplot(data=df, x="square_footage", y="price",scatter_kws={'alpha' :1/10}, line_kws={'color': 'black'})
plt.gca().yaxis.set_major_formatter(mticker.StrMethodFormatter('{x:,.0f}'))
plt.xlabel("Square Footage")
plt.ylabel("Price");
```



Square footage also appears to be an abnormal distribution, but the expectations of a housing dataset can explain this away. The square_footage variable will not scale to 0, as a minimum square footage value must be maintained for a viable living area in a house. As a result, there is a cluster of values visible around 500 in both the univariate histogram and bivariate scatterplot. This is expected and will not affect the multiple regression analysis.

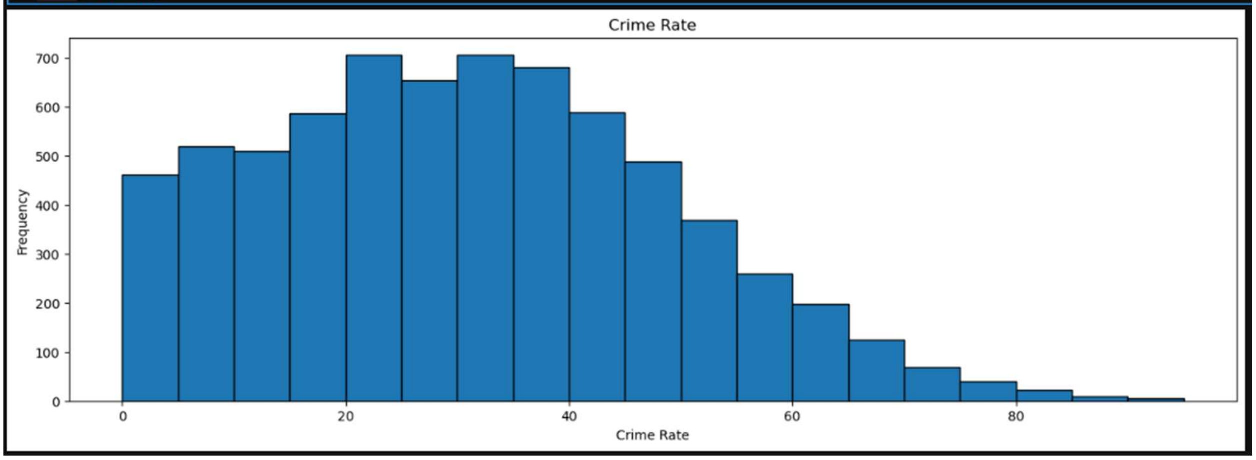
```
#Number of Bedrooms - Independent Variable
##Univariate and Bivariate Analysis
plt.figure(figsize = [16,5])
plt.suptitle("Exploration of Number of Bedrooms")

#Left Plot: Univariate exploration of num_bedrooms
plt.subplot(1, 2, 1)
plt.title("Number of Bedrooms")
bins = np.arange(1, 8, 1)
plt.hist(data=df, x='num_bedrooms', bins=bins, edgecolor='black', density=False)
plt.xlabel("Number of Bedrooms")
plt.ylabel("Frequency")

#Right Plot: Bivariate exploration of num_bedrooms vs price
plt.subplot(1, 2, 2)
plt.title("Number of Bedrooms vs. Price of Houses")
sns.violinplot(data=df, x="num_bedrooms", y="price")
plt.gca().yaxis.set_major_formatter(mticker.StrMethodFormatter('{x:,.0f}'))
plt.xlabel("Number of Bedrooms")
plt.ylabel("Price");
```



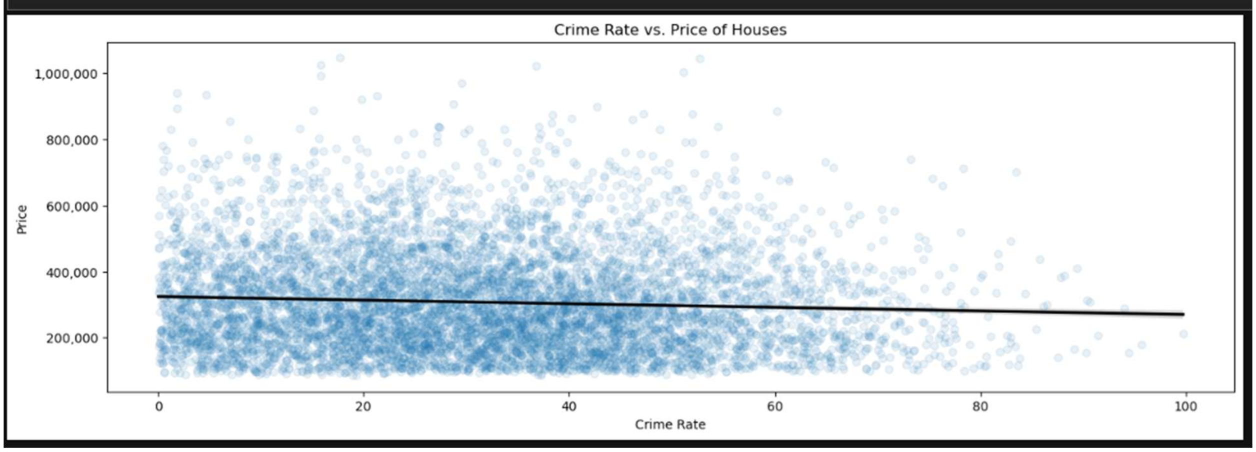
The num_bedrooms variable demonstrated a normal distribution in the histogram and violin plot. Notably, since seven bedrooms represented an unusual violin plot as a horizontal line only, I confirmed this was due to a low response rate, which was confirmed as only one house returned a value of 7 for num_bedrooms, noted in the code below.

```
#Demonstrating that the horizontal line for 7 bedrooms is due to low response value
df['num_bedrooms'].value_counts()
##

num_bedrooms
3    2662
4    1738
2    1665
1     479
5     424
6      31
7       1
Name: count, dtype: int64
```

```
#Crime Rate - Independent Variable
##Univariate exploration of crime_rate
plt.figure(figsize = [16,5])
plt.title("Crime Rate")
bins = np.arange(0, 100, 5)
plt.hist(data=df, x='crime_rate', bins=bins, edgecolor='black', density=False)
plt.xlabel("Crime Rate")
plt.ylabel("Frequency");
```

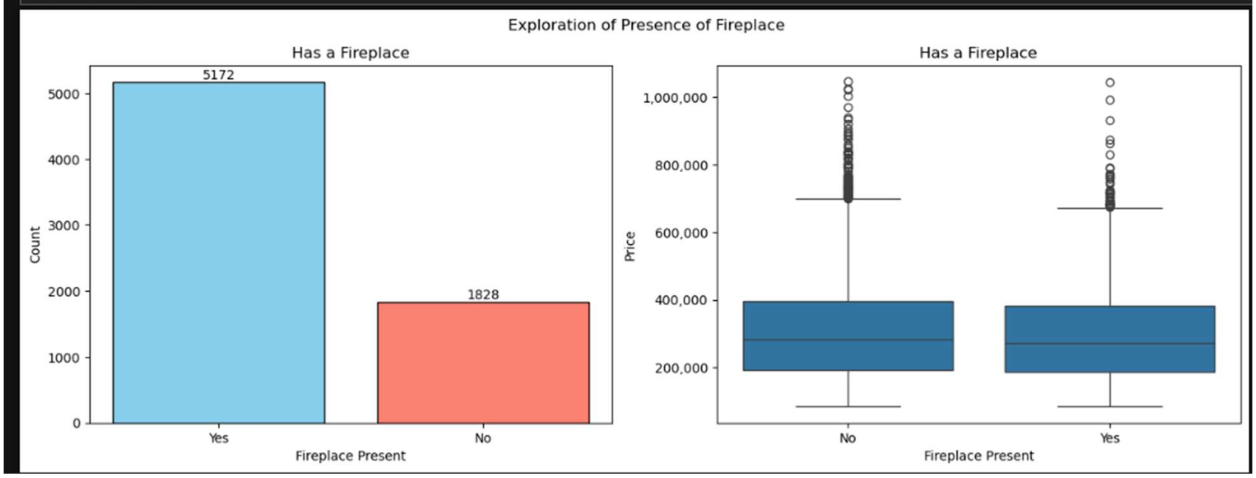



```
#Crime Rate - Independent Variable
##Bivariate exploration of crime_rate vs price
plt.figure(figsize = [16,5])
plt.title("Crime Rate vs. Price of Houses")
sns.regplot(data=df, x="crime_rate", y="price",scatter_kws={'alpha' :1/10}, line_kws={'color': 'black'})
plt.gca().yaxis.set_major_formatter(mticker.StrMethodFormatter('{x:,.0f}'))
plt.xlabel("Crime Rate")
plt.ylabel("Price");
```

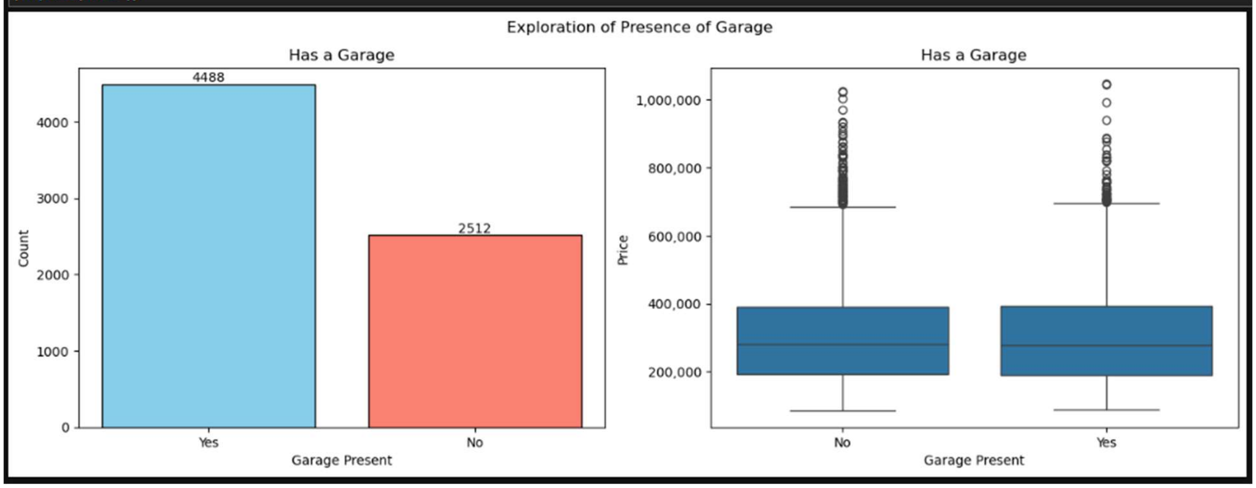


The crime_rate variable demonstrated a close-to-normal distribution in both visualizations. However, clustering values on the lower end slightly skew the data. This will be reviewed as necessary as a part of the statistical analysis for this variable.

```python
#Has a Fireplace - Independent Variable
##Univariate and Bivariate Analysis
plt.figure(figsize = [16,5])
plt.suptitle("Exploration of Presence of Fireplace")

#Left Plot: Univariate Analysis of fireplace
plt.subplot(1, 2, 1)
plt.title("Has a Fireplace")
counts = df['fireplace'].value_counts()
categories = ['Yes', 'No']
counts = counts.sort_index()
bars = plt.bar(categories, counts, color=['skyblue', 'salmon'], edgecolor='black')
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width() / 2, yval + 0.5,
             f'{int(yval)}', ha='center', va='bottom')
plt.xlabel("Fireplace Present")
plt.ylabel("Count")

#Right Plot: Bivariate Analysis of fireplace vs price
plt.subplot(1, 2, 2)
plt.title("Has a Fireplace")
sns.boxplot(data=df, x="fireplace", y="price")
plt.xticks([0, 1], ['No', 'Yes'])
plt.gca().yaxis.set_major_formatter(mticker.StrMethodFormatter('{x:,.0f}'))
plt.xlabel("Fireplace Present")
plt.ylabel("Price");
```

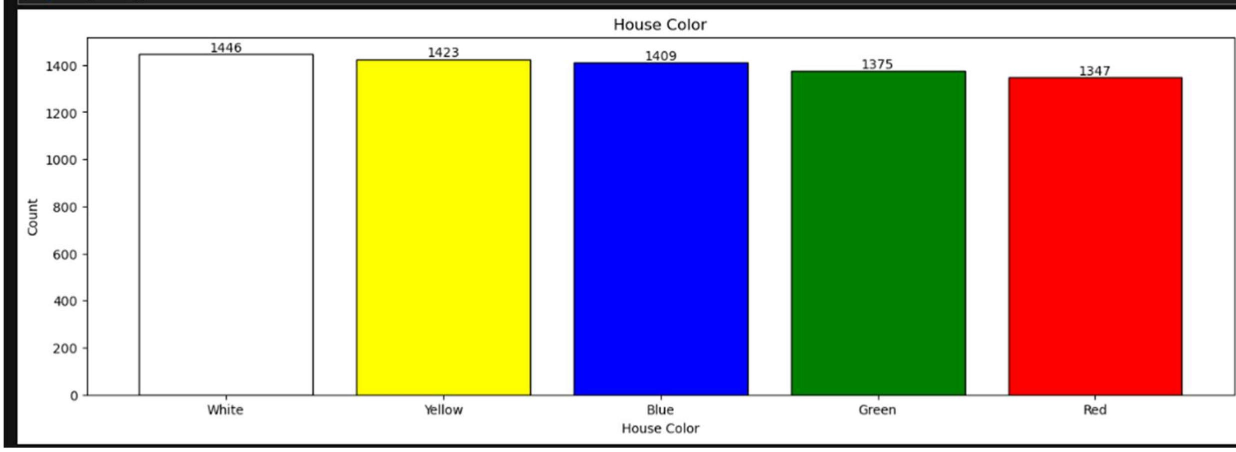


```python
#Has a Garage - Independent Variable
##Univariate and Bivariate Analysis
plt.figure(figsize = [16,5])
plt.suptitle("Exploration of Presence of Garage")

#Left Plot: Univariate Analysis of fireplace
plt.subplot(1, 2, 1)
plt.title("Has a Garage")
counts = df['garage'].value_counts()
categories = ['Yes', 'No']
counts = counts.sort_index()
bars = plt.bar(categories, counts, color=['skyblue', 'salmon'], edgecolor='black')
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width() / 2, yval + 0.5,
             f'{int(yval)}', ha='center', va='bottom')
plt.xlabel("Garage Present")
plt.ylabel("Count")

#Right Plot: Bivariate Analysis of garage vs price
plt.subplot(1, 2, 2)
plt.title("Has a Garage")
sns.boxplot(data=df, x="garage", y="price")
plt.xticks([0, 1], ['No', 'Yes'])
plt.gca().yaxis.set_major_formatter(mticker.StrMethodFormatter('{x:,.0f}'))
plt.xlabel("Garage Present")
plt.ylabel("Price");
```

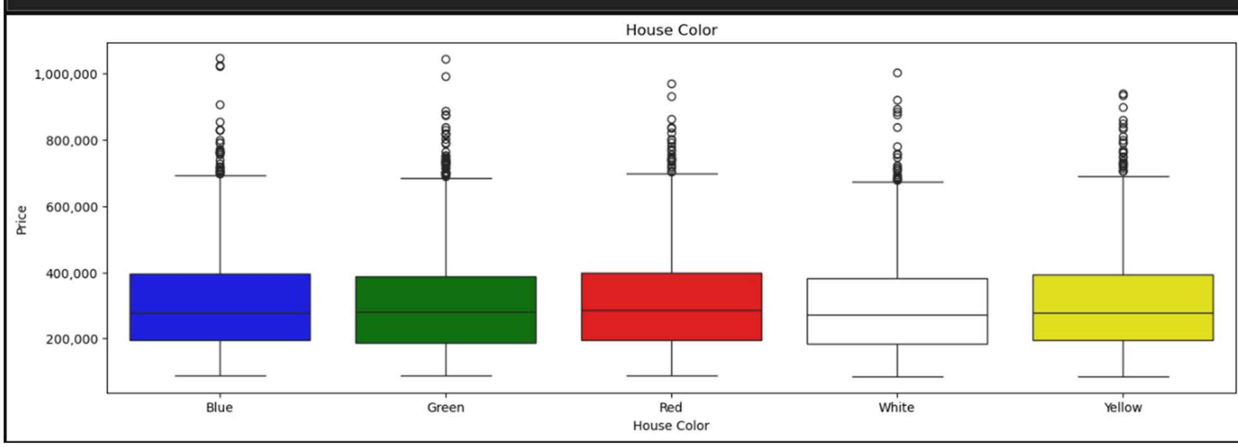


There is nothing notable about the distribution of has_fireplace or has_garage categorical variables. The boxplots represent an average spread of values with an acceptable amount of outliers.

```
#House Color - Independent Variable
##Univariate Analysis of house_color
plt.figure(figsize=[16,5])
plt.title("House Color")
counts = df['house_color'].value_counts()
categories = ["White", "Yellow", "Blue", "Green", "Red"]
bars = plt.bar(categories, counts, color=["white", "yellow", "blue", "green", "red"], edgecolor='black')
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width() / 2, yval + 0.5,
             f'{int(yval)}', ha='center', va='bottom')
plt.xlabel("House Color")
plt.ylabel("Count");
```




```
#House Color - Independent Variable
##Bivariate Analysis of house_color vs price

plt.figure(figsize=[16,5])
plt.title("House Color")
sns.boxplot(data=df, x="house_color", y="price", hue="house_color", palette=['blue', 'green', 'red', 'white', 'yellow'],legend=False)
plt.gca().yaxis.set_major_formatter(mticker.StrMethodFormatter('{x:,.0f}'))
plt.xlabel("House Color")
plt.ylabel("Price");
```



The uniform distribution of house color may have a negative effect on the multiple regression model, as this distribution of data may have lower predictive power, along with minimal contribution to optimizing the R^2 value.

Linear Regression Modelling

D1. Creation of Training & Test Data Sets

Before splitting the datasets, one-hot encoding was performed on the categorical variable "house_color," and it was converted from boolean values to binary integers. Hence, it is usable in multiple linear regression.

```
#Performing one-hot encoding
df_encoded = pd.get_dummies(df, columns=['house_color'], drop_first=True)

print(df_encoded)
```

```
#Converting one-hot-encoding columns from boolean values to binary integers
one_hot_columns = ['house_color_Green', 'house_color_Red', 'house_color_White', 'house_color_Yellow']
df_encoded[one_hot_columns] = df_encoded[one_hot_columns].astype(int)
print(df_encoded[one_hot_columns].head()) #Confirming the change worked
```

```
   house_color_Green  house_color_Red  house_color_White  house_color_Yellow
0                  0                0                  0                   0
1                  1                0                  0                   0
2                  1                0                  0                   0
3                  0                1                  0                   0
4                  0                0                  1                   0
```

The next step was to split the data into two separate models, training and test, then exporting them to share as a part of the assessment. An 80/20 training to test split was utilized using the code below.

```
#Splitting the Dataset into Test and Training
y = df_encoded.price
X = df_encoded[['square_footage', 'num_bedrooms', 'crime_rate', 'fireplace', 'house_color_Green', 'house_color_Red', 'house_color_White', 'house_color_Yellow', 'garage']].assign(const=1)

#Splitting the Dataset into a Test and Training dataset with an 80/20 split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
y_train = y_train.to_frame() #Converting to dataframe for ease in exporting
y_test = y_test.to_frame()
print(f"Training data: {X_train.shape}, Testing data: {X_test.shape}")
print(f"Training labels: {y_train.shape}, Testing labels: {y_test.shape}")
```

The following was used to export the data:

```python
#Combining the datasets for exporting
train_data = pd.concat([X_train, y_train], axis=1)
test_data = pd.concat([X_test, y_test], axis=1)
```

```python
#Exporting the Test & Train Datasets to share
train_data.to_csv("training_data", index=False)
test_data.to_csv("test_data", index=False)
```

D2. Optimization of the Model

To optimize the model, backward stepwise elimination was used to remove the statistically insignificant variables to the research question. This method allows values to be removed one at a time and re-tests the model after removal. The initial model returned the details below as a result.

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.475
Model:                            OLS   Adj. R-squared:                  0.475
Method:                 Least Squares   F-statistic:                     563.0
Date:                Mon, 20 Jan 2025   Prob (F-statistic):               0.00
Time:                        21:25:11   Log-Likelihood:                -72931.
No. Observations:                5600   AIC:                         1.459e+05
Df Residuals:                    5590   BIC:                         1.459e+05
Df Model:                           9
Covariance Type:            nonrobust
======================================================================================
                         coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------
square_footage       178.3021      3.464     51.477      0.000     171.512     185.092
num_bedrooms        6.239e+04   1432.519     43.556      0.000     5.96e+04    6.52e+04
crime_rate          -207.9232     81.337     -2.556      0.011    -367.375     -48.471
fireplace          -3362.1861   3360.903     -1.000      0.317   -9950.861    3226.489
house_color_Green   2415.6035   4644.512      0.520      0.603   -6689.444    1.15e+04
house_color_Red    -3103.3786   4693.769     -0.661      0.509   -1.23e+04    6098.233
house_color_White  -8700.1073   4622.537     -1.882      0.060   -1.78e+04     361.860
house_color_Yellow -1008.5859   4631.374     -0.218      0.828   -1.01e+04    8070.706
garage              -739.9534   3053.995     -0.242      0.809   -6726.970    5247.064
const              -5.761e+04   7061.773     -8.158      0.000   -7.15e+04   -4.38e+04
==============================================================================
Omnibus:                      472.472   Durbin-Watson:                   1.986
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              621.867
Skew:                           0.726   Prob(JB):                    9.19e-136
Kurtosis:                       3.745   Cond. No.                     6.67e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 6.67e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

As one note indicated, there was a potential problem with multicollinearity; the Variance Inflation Factor was run on the dataset to identify any potential issues. The below results were returned.

	feature	VIF
0	square_footage	5.523828
1	num_bedrooms	6.458648
2	crime_rate	3.400465
3	fireplace	1.321931
4	house_color_Green	1.809261
5	house_color_Red	1.776863
6	house_color_White	1.803543
7	house_color_Yellow	1.810742
8	garage	1.520540

Both square_footage and num_bedrooms have VIF over 5, which presents a mild concern for multicollinearity. Neither has a value so high that it presents a significant concern, but to be sure, I wanted to rule this out. To do so, I reviewed the relationships between these variables using a correlation matrix and ultimately determined that the correlations were weak, so I kept all variables in the dataset. The results of the correlation matrix are below.

	square_footage	num_bedrooms	crime_rate	house_color_White	const
square_footage	1.000000	0.094199	-0.038409	-0.001391	NaN
num_bedrooms	0.094199	1.000000	-0.039409	-0.011361	NaN
crime_rate	-0.038409	-0.039409	1.000000	-0.008506	NaN
house_color_White	-0.001391	-0.011361	-0.008506	1.000000	NaN
const	NaN	NaN	NaN	NaN	NaN

To further ensure multicollinearity was not a concern, the dataset was scaled to provide a better view of the continuous variables and assess the concerns more directly. After doing so, the cond No. dropped down to 7.37, and the note indicating multicollinearity was a concern. The scaled dataset was not utilized for the remainder of the analysis as it was not advised as a part of the instructions.

```
#Scaling the training dataset separately to further demonstrate multicollinearity is not a concern.
#Note - the scaled values will not be used for the final model
scaler = MinMaxScaler()
reg_df_minmax = pd.DataFrame(scaler.fit_transform(X_train), columns=X_train.columns, index=X_train.index)

X = reg_df_minmax[['square_footage', 'num_bedrooms', 'crime_rate', 'house_color_White']].assign(const=1)
model = sm.OLS(y_train, X)
results = model.fit()
print(results.summary())

#Cond. No. has dropped down to 7.37
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.475
Model:                            OLS   Adj. R-squared:                  0.475
Method:                 Least Squares   F-statistic:                     1267.
Date:                Mon, 20 Jan 2025   Prob (F-statistic):               0.00
Time:                        21:59:58   Log-Likelihood:                -72932.
No. Observations:                5600   AIC:                         1.459e+05
Df Residuals:                    5595   BIC:                         1.459e+05
Df Model:                           4
Covariance Type:            nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
square_footage     4.082e+05   7916.130     51.561      0.000    3.93e+05    4.24e+05
num_bedrooms       3.744e+05   8588.245     43.597      0.000    3.58e+05    3.91e+05
crime_rate        -2.066e+04   8103.124     -2.550      0.011   -3.65e+04   -4777.303
house_color_White -8240.2692   3631.218     -2.269      0.023   -1.54e+04   -1121.673
const              1.012e+05   4518.148     22.409      0.000    9.24e+04     1.1e+05
==============================================================================
Omnibus:                      473.581   Durbin-Watson:                   1.987
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              623.595
Skew:                           0.727   Prob(JB):                    3.87e-136
Kurtosis:                       3.746   Cond. No.                         7.37
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

After ruling out multicollinearity, I initiated backward stepwise elimination. Using the p-value as the primary metric for whether a variable was statistically significant, I determined whether a variable should stay or be removed. I re-ran the model after each variable was removed in the following order: garage (.809 p-value), house_color_Yellow (.829 p-value), house_color_Red (.522 p-value), house_color_Green (.319 p-value), and fireplace (.326 p-value). This left the results below when the model was run, with no p-values outside an acceptable range.

```
]: #Final data model
X_train = X_train[['square_footage', 'num_bedrooms', 'crime_rate', 'house_color_White']].assign(const=1)
model = sm.OLS(y_train, X_train)
results = model.fit()
print(results.summary())

                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.475
Model:                            OLS   Adj. R-squared:                  0.475
Method:                 Least Squares   F-statistic:                     1267.
Date:                Mon, 20 Jan 2025   Prob (F-statistic):               0.00
Time:                        21:59:58   Log-Likelihood:                -72932.
No. Observations:                5600   AIC:                         1.459e+05
Df Residuals:                    5595   BIC:                         1.459e+05
Df Model:                           4
Covariance Type:            nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
square_footage      178.3441      3.459     51.561      0.000     171.563     185.125
num_bedrooms        6.24e+04   1431.374     43.597      0.000    5.96e+04    6.52e+04
crime_rate         -207.2474     81.275     -2.550      0.011    -366.578     -47.917
house_color_White -8240.2692   3631.218     -2.269      0.023   -1.54e+04   -1121.673
const             -5.924e+04   6290.899     -9.417      0.000   -7.16e+04   -4.69e+04
==============================================================================
Omnibus:                      473.581   Durbin-Watson:                   1.987
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              623.595
Skew:                           0.727   Prob(JB):                    3.87e-136
Kurtosis:                       3.746   Cond. No.                     4.95e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.95e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

The final data model included square_footage, num_bedrooms, crime_rate, and house_color_White. Notably, the above output contains an R2 and adjusted R2 at .475, indicating that the variables in the model predict around 48% of the dependent variable of the price change. Also included are the F statistic, probability F Statistic, and coefficient estimates.

D3. Mean Squared Error (MSE) Analysis

Below is the code and results of the Mean Squared Error after optimization

```
]: #Calculating Mean Squared Error for the Training Dataset
model = sm.OLS(y_train, X_train).fit()
y_pred = model.predict(X_train)
mse_train = mean_squared_error(y_train, y_pred)
print ("Mean Squared Error (MSE) - Training Dataset:", mse_train)

Mean Squared Error (MSE) - Training Dataset: 12013060202.521719
```

Taking the square root of the MSE above, the result is $109,604.11 – indicating the model's predictions will differ from the actual house prices by around that value.

D4. Prediction

Using the model developed using the training dataset, the test dataset was analyzed using only the variables deemed relevant to the analysis. The optimized model produced the following results in its output.

```
#Running the optimized model on the test data set
X_test = X_test[['square_footage', 'num_bedrooms', 'crime_rate', 'house_color_White']].assign(const=1)
model = sm.OLS(y_test, X_test)
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.448
Model:                            OLS   Adj. R-squared:                  0.446
Method:                 Least Squares   F-statistic:                     283.1
Date:                Mon, 20 Jan 2025   Prob (F-statistic):          2.97e-178
Time:                        21:59:59   Log-Likelihood:                -18213.
No. Observations:                1400   AIC:                         3.644e+04
Df Residuals:                    1395   BIC:                         3.646e+04
Df Model:                           4
Covariance Type:            nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
square_footage      183.9118      6.828     26.935      0.000     170.517     197.306
num_bedrooms       5.296e+04   2931.830     18.063      0.000     4.72e+04    5.87e+04
crime_rate         -145.8722    162.601     -0.897      0.370    -464.841     173.096
house_color_White -1375.5093   7084.563     -0.194      0.846   -1.53e+04    1.25e+04
const             -4.057e+04     1.3e+04    -3.122      0.002   -6.61e+04   -1.51e+04
==============================================================================
Omnibus:                       81.323   Durbin-Watson:                   1.974
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               94.509
Skew:                           0.610   Prob(JB):                     3.00e-21
Kurtosis:                       3.361   Cond. No.                     5.11e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.11e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Additionally, the MSE was calculated for the test dataset and produced results in line with what was expected compared to the training dataset.

```
#Calculating Mean Squared Error for the Test Dataset
model = sm.OLS(y_test, x_test).fit()
y_pred = model.predict(x_test)
mse_test = mean_squared_error(y_test, y_pred)
print ("Mean Squared Error (MSE) - Test Dataset:", mse_test)

Mean Squared Error (MSE) - Test Dataset: 11681955037.764385
```

Comparing the MSE and RMSE of the two datasets demonstrated that the model was consistent across both datasets and generalized well to the new data. There is no indication of overfitting or underfitting, and the data was correctly split, resulting in little to no data leakage.

Summary and Results

E1. Libraries Utilized

- pandas & numpy – For general usefulness with data frames and Python coding
- matplotlib.pyplot - Used for data visualizations
- seaborn - Used for data visualizations
- statsmodels.api - Used to create Ordinary Least Squares (OLS) Regression Model
- matplotlib.ticker - Used to scale the axes in data plots properly
- sklearn.model_selection: train_test_split - Used to split the datasets
- statsmodels.stats.outliers_influence: variance_inflation_factor - Used to check for multicollinearity in the data models
- sklearn.preprocessing: MinMaxScaler - Used for a diagnostic test of VIF, scaling the data frame to solve for Multicollinearity

E2. Method of Optimization

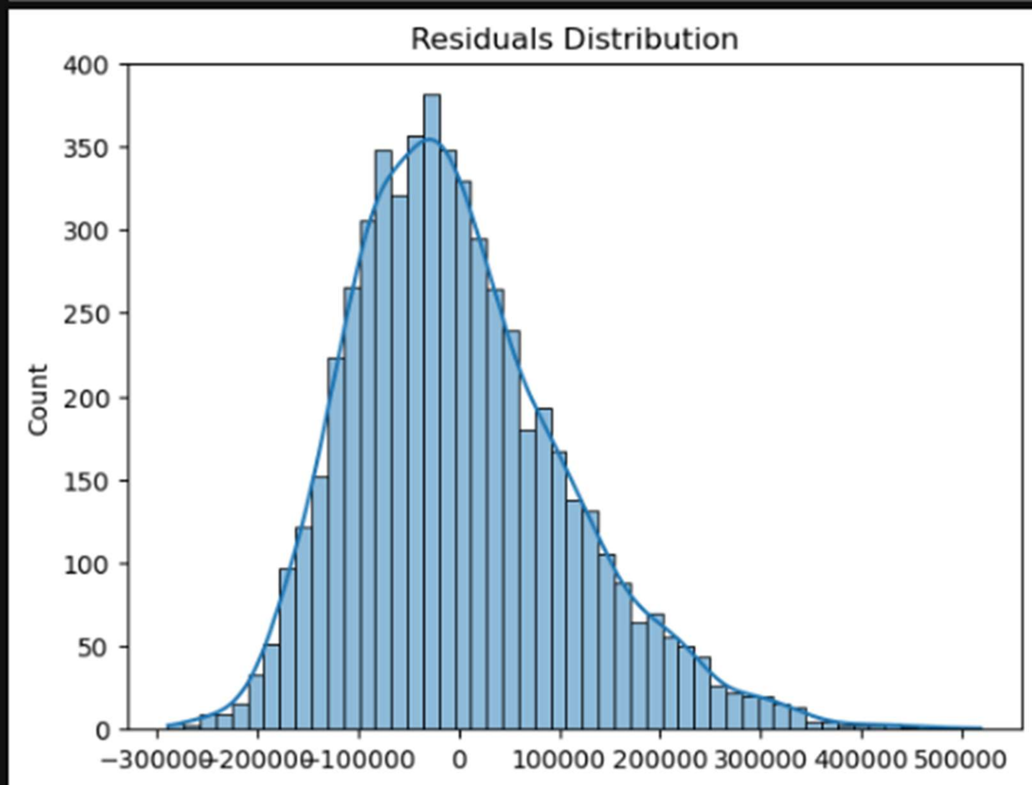
Backward Stepwise Elimination was utilized to optimize the model. Instructions were to begin with six independent variables and one dependent variable in the analysis, so Forward Stepwise Selection was not optimal as it would add additional variables to the data model. Backward Stepwise Elimination allowed me to eliminate variables one at a time when they were statistically insignificant to ensure only the most relevant variables remained. This was validated by checking for the mean squared error on both the training and testing datasets post-optimization to confirm that the error was reduced in the model.

E3. Verification of Assumptions

The major assumptions of backward stepwise elimination include linear relationships between predictor variables and their target, low multicollinearity, and a normal distribution of residuals. Linear relationships were confirmed using the visualizations listed above in section C3 – the scatterplots represented straight trend lines, and the correlation matrix presented demonstrated linear associations. For multicollinearity, VIF was calculated for the predictor variables, and the highest value was 6.4, which presented a moderate concern for multicollinearity. To rule this out, the dataset was temporarily scaled, and the model was re-run, demonstrating a significantly reduced condition number. To assess the normal distribution of residuals, histograms of the residuals were plotted after model optimization for both the training and test datasets; both demonstrated slightly skewed residuals but still a normal distribution.

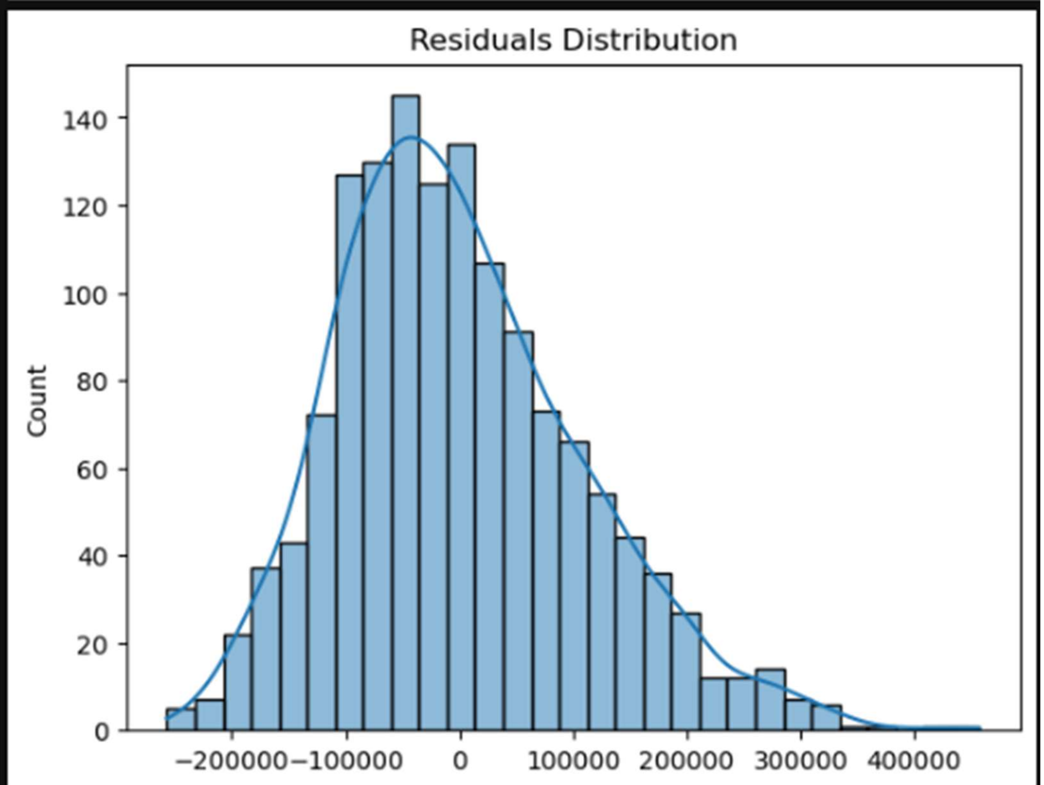
Training Residuals:

```
#Checking Residuals to ensure close to normal distribution
residuals = results.resid
sns.histplot(residuals, kde=True)
plt.title("Residuals Distribution")
plt.show()
##Skewed Right slightly - close to normal distribution
```



Test Residuals:

```
#Checking Residuals to ensure close to normal distribution
residuals = results.resid
sns.histplot(residuals, kde=True)
plt.title("Residuals Distribution")
plt.show()
##Skewed Right slightly - close to normal distribution
```



E4. Regression Equation and Coefficient Estimates

Using the training data model, the final regression output is as follows:

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.475
Model:                            OLS   Adj. R-squared:                  0.475
Method:                 Least Squares   F-statistic:                     1267.
Date:                Thu, 23 Jan 2025   Prob (F-statistic):               0.00
Time:                        21:12:15   Log-Likelihood:                -72932.
No. Observations:                5600   AIC:                         1.459e+05
Df Residuals:                    5595   BIC:                         1.459e+05
Df Model:                           4
Covariance Type:            nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
square_footage      178.3441      3.459     51.561      0.000     171.563     185.125
num_bedrooms        6.24e+04   1431.374     43.597      0.000    5.96e+04    6.52e+04
crime_rate         -207.2474     81.275     -2.550      0.011    -366.578     -47.917
house_color_White -8240.2692   3631.218     -2.269      0.023   -1.54e+04   -1121.673
const             -5.924e+04   6290.899     -9.417      0.000   -7.16e+04   -4.69e+04
==============================================================================
Omnibus:                      473.581   Durbin-Watson:                   1.987
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              623.595
Skew:                           0.727   Prob(JB):                    3.87e-136
Kurtosis:                       3.746   Cond. No.                     4.95e+03
==============================================================================
```

From the above, the regression equation is:

price^=−59,240+178.34(square footage)+62,400(num bedrooms)−207.25(crime rate)−8,240.27(house color White)

To explain, the general estimates are as follows:

- Assuming all other variables remain constant for the below:
    - Every additional square foot increases the price of the house by $178.34
    - Every additional bedroom increases the price of the house by $62,400
    - A one-unit increase in crime rate decreases the price of the house by $207.25
    - The house being white decreases the price of the house by $8,240.27 compared to another color.

The intercept is represented at -59,240, which is not meaningful.

E5. Model Metrics

Per the above output, the R2 and adjusted R2 are the same at 0.475. This means the model explains 47.5% of the variation of the pricing in houses. More broadly speaking, the variables selected have a definitive impact on the price of a house, but more factors were not included, which make up more than half the explanatory power. The p-values for all variables are less than 0.05, meaning they are likely to impact housing prices statistically. As stated above, the output indicates there would be multicollinearity concerns, but those have been ruled out separately.

When assessing the MSE for both datasets, it is most helpful to look at them alongside one another, and the mean price of houses was also included as a relevant comparison.

```
]: #Comparing both MSE
   print ("Mean Squared Error (MSE) - Training Dataset:", mse_train)
   print ("Mean Squared Error (MSE) - Test Dataset:", mse_test)

   Mean Squared Error (MSE) - Training Dataset: 12013060202.521719
   Mean Squared Error (MSE) - Test Dataset: 11681955037.764385

]: #Validating results by calculating RMSE
   training_rmse = np.sqrt(12013060202.521719)
   test_rmse = np.sqrt(11681955037.764385)

   print("Training RMSE:", training_rmse)
   print("Test RMSE:", test_rmse)
   print("Mean housing Price:", df["price"].mean())
   ##The model's predictions are off by a significant amount, so it wo

   Training RMSE: 109604.10668639072
   Test RMSE: 108083.09320964303
   Mean housing Price: 307281.5217142857
```

The two RMSE values for both datasets were within $1600 of one another, indicating the model was not overfit or underfit and generalized to data effectively. The relative size of the RMSE, however, was nearly 33% of the mean price of the house. This means the model has predictive power, but the prediction error is significantly higher than what would be considered an excellently fitted model.

E6. Results and Implications

The results of the analysis indicate that square footage and the number of bathrooms increase the value of a home. Additionally, higher crime rates or white houses will reduce the value of the homes. All of these variables were statistically significant in terms of the price of the houses. The overall predictive power of the model was off by an average of around $108,000, which makes sense when considering the R2 value indicated that the model only accounted for around 47.5% of the variance. The model was not incorrect, but it was incomplete.

E7. Recommendations

Based upon the analysis, the business should ensure they invest in houses that are in lower crime areas and whose houses are not white in color. Additionally, focusing on the number of rooms and square footage counts will yield direct increases in profit. The best way to optimize their earnings would be to run another regression model encapsulating additional variables. Using the model prepared and implementing forward stepwise selection to add these variables would increase the efficacy of the prediction and provide further relevant variables to calculate the overall changes in price.

References

OpenAI. (2025, January 16). Explanation of lambda functions in Python. *ChatGPT*. Retrieved from https://chat.openai.com

Tutorialspoint. (n.d.). *Matplotlib tick formatters*. Tutorialspoint. https://www.tutorialspoint.com/matplotlib/matplotlib_tick_formatters.htm