Briana Churchill

Student ID: 011009463

Dr. William Sewell

2024 April 25

Data Analytics Graduate Capstone (NKM2)

Research Question

A. Summarize the original real-data research question

The research question for this capstone project is: Using the USA real estate dataset provided via Kaggle, can a multiple linear regression model be created to predict housing prices in Salt Lake County, Utah?

This study is to build a predictive model that will estimate house prices in Salt Lake County. There are many factors that contribute to the price of a house such as square footage, lot size, location, and more. With so many variables, this analysis requires a method that can utilize multiple independent variables to make predictions. A simple linear regression would model the relationship between a single dependent variable and a single independent variable. Since there are multiple independent variables, multiple linear regression (MLR) will be used to create the predictive model. MLR is a valuable tool for predicting housing prices because it allows modeling of the relationship between the dependent variable and each independent variable (Geeksforgeeks, 2023).

Having the data needed to make housing price predictions can help many businesses and professions. However, this study was specifically created to make predictions to optimize marketing and pricing strategies for real estate investors (LinkedIn, 2024), as Utah has a growing population and is a hot market for real estate investment over the past couple of years (GuidingOurGrowth, n.d.) Being able to predict housing prices would allow real estate investors to maximize their investment profits to further grow their investment portfolio(s). To be able to utilize the model for these predictions, there would need to be sufficient evidence to accept the hypothesis and reject the null hypothesis. These hypotheses are explained below:

Hypothesis:

H0: A predictive MLR model regarding Salt Lake County’s housing market can be made from the research dataset with a model accuracy > 70%.

Null Hypothesis:

H1: A reliable predictive model pertaining to Salt Lake County cannot be constructed from the USA real estate dataset.

Data Collection

B. Data-collection process

The data utilized to complete this study was collected from a publicly available dataset from Kaggle.com, which is an online community platform for data scientists. The dataset consists of 2,226,382 rows of collected data from realtor.com, “the second most visited real estate listing website in the United States as of 2024” (Sakib, 2024). The data covers real estate listings in the United States from March 2022 through March 2024. The 12 available variables provided in the data set are:

- Listing Broker
- Status of listing
- Price
- Number of bedrooms
- Number of bathrooms
- Property size in acres
- Street
- City name
- State name
- Zip code
- House area in square feet
- Previously sold date

Broker information and street addresses were categorically encoded by the uploader to maintain privacy. The dataset was uploaded to Kaggle.com as a downloadable csv file. An advantage of using Kaggle.com is that the website is data-centric, powered by individuals participating in data science practices. Additionally, the website is easy to navigate and has a search function with many filtering options. The usage of filtering is essential when searching for a reliable dataset in any subject because the disadvantage of Kaggle is that that there are no templates or specific parameters for uploaders, and they choose what kind of data to provide. Fortunately, the real estate data was uploaded large and reliable. Though a challenge was presented upon the realization that there is no variable for county information provided. This issue was addressed and corrected in the data cleaning and transformation steps, which are described below.

Data Extraction and Preparation

C. Describe the data-extraction and -preparation process

As mentioned above, the initial data set contained over 2 million rows of data regarding listings of houses throughout the United States. Many steps were taken to slim down the data to focus specifically on Salt Lake County, Utah. Listed in chronological order, here were the steps taken:

- Necessary libraries were imported
    - Numpy, used to transform the data mathematically
    - Pandas, provides the logical structure
- CSV file was read into Data Frame "df" using pandas
- The index column created by Pandas name updated to "Listing"
- A new data frame structure was created containing only listings in the state of Utah
- A new data frame structure was created from the Utah data frame with listings in cities of Salt Lake County
- Any instances of null values were printed for all variables
- Null values were dropped from rows that would later be used in the regression model
    - 1 null value from Price
    - 127 null values from Bedrooms column
    - 141 null values from Bathrooms column
    - 148 null values from Lot Size column
    - 118 null values from House Size column
- Null values were printed again to ensure the above instances were dropped
- Data frame was checked for duplicate rows
- Columns were renamed for consistency and comprehension
    - "acre_lot" was changed to "Lot_Size"
    - "house_size" was changed to "House_Size"
    - "price" was changed to "Price"
    - "city" was changed to "City"
- Cities were grouped into two categories, "North" and "South" based on location of city within the county
- Data type for "bed" and "bath" columns was changed from float64 to integer
- A categorical variable named "Bathrooms" was created based on ranges within the "bath" column
    - 1-2 bathrooms were categorized as "Lower"
    - 3-4 bathrooms were categorized as "Higher"
    - All other values for bathroom were categorized as "other"
- A categorical variable named "Bedrooms" was created based on ranges within the "bed" column
    - 1-3 bedrooms were categorized as "Lower"

    - 4-6 bedrooms were categorized as "Higher"
    - All other values for bedroom were categorized as "other"
- Any instances with a value of "other" in the "Bathrooms" and "Bedrooms" columns were dropped
- Values found in the "City", "Bathrooms" and "Bedrooms" columns were mapped to numeric values
    - Values of "Lower" were changed to 0
    - Values of "Higher" were changed to 1
- Remaining outliers were dropped
    - Any values within the Price column >= 999999
    - Any values within the Lot Size column >= 1.25
    - Any values within the House Size column >= 15000
- Remaining numeric variables with datatype float64 were changed to datatype integer
    - House Size
    - Price
    - City
- Final data frame was created with transformed variables needed for the regression model
    - Lot Size
    - House Size
    - Price
    - Bedrooms
    - Bathrooms
    - City
- Final model was printed for final visual inspection

As mentioned above regarding data collection, a disadvantage to using Kaggle (and many other platforms) is that individuals utilizing the data are at the mercy of the uploader. For this dataset specifically, no column was available for county information. Therefore, it was necessary to group all the cities within the Salt Lake County into their own data frame. This could only be done after variables with state value "Utah" were extracted, otherwise there could have been a possibility of having instances from other states with a common city name. The original and subsequent data frames were created using the Pandas tool for Python via Anaconda-Navigator. Pandas was imported as "pd" so using the library was as simple as including two letters in the code. An example can be seen in the screenshot below for input 2, with the use of "pd.read_csv" to read the original CSV file to organize the data into a data frame. Output 2 shows this new data frame, which includes many null values at first glance. Removing or filling in the null values was the next necessary step because missing data may lead to bias and loss of precision (Hughes, R. et al., 2019) which could impact the efficiency of the model. Duplicate data could also affect the analysis, so the data frame was checked and none were found.

Once variables were renamed for consistency, and easier comprehension, the data was ready for specified transformation. Preparation included steps to avoid multicollinearity, which happens when there is high correlation between two or more independent variables in a multiple linear regression. This high correlation can lead to skewed or misleading results (Hayes, 2024) which would negatively affect the model. The first step to reduce any multicollinearity was to group bedrooms and bathrooms into a range of numbers within two new categorical variables created "Bathrooms" and "Bedrooms." Values in the "bath" column of 1-2 were grouped as "Lower" and values of 3-4 grouped as "Higher." For bedrooms, values in the "bed" column of 1-3 were grouped as "Lower" and values of 4-6 were grouped as "Higher." These transformations were executed utilizing the Numpy library for the mathematical formulas seen in input 10 of the images included below.

A similar process was completed with the values in the "City" column, which were categorized as "Higher" or "Lower" depending on their physical location within the county (Northern cities = Higher, Southern cities = Lower). A disadvantage to using this technique was that it was time consuming, and the code needed to be updated many times because the parameters used resulted in various outliers among the bed and bath columns, such as the very few listings with > 5 bathrooms. However, the parameters used (Ie: 1-2 = lower, etc.) were the most appropriate given the data provided so this disadvantage was easily taken care of by grouping outliers as "other" and dropping all "other" values from the data frame. An advantage to this technique was that I was able to streamline the process of changing the values to 0 or 1 within all three applicable columns by using "Higher" and "Lower" in all categorized variables. Being able to speed up the process of transforming these values made up for time lost dealing with the disadvantage mentioned above. Final transformation steps included changing float64 datatypes to integer for consistency, and dropping any remaining outliers in the remaining columns. Once these steps were completed, all the necessary variables were moved into a final data frame, which was ready for modeling and analysis. All scripts used in the cleaning and transformation steps detailed above can be seen in the images here:

In [1]:
```
# Import necessary packages
import numpy as np
import pandas as pd
```

In [2]:
```
# Remove pandas duplication of the first column within the file
df = pd.read_csv('./realtor-data.zip.csv')
# Show an example of a transaction in the dataset
df.index.name = 'Listing'
df
```

```
Out[2]:
         brokered_by    status     price  bed  bath  acre_lot     street        city        state  zip_code  house_size  prev_sold_date
Listing
      0     103378.0  for_sale  105000.0  3.0   2.0      0.12  1962661.0    Adjuntas  Puerto Rico     601.0       920.0             NaN
      1      52707.0  for_sale   80000.0  4.0   2.0      0.08  1902874.0    Adjuntas  Puerto Rico     601.0      1527.0             NaN
      2     103379.0  for_sale   67000.0  2.0   1.0      0.15  1404990.0  Juana Diaz  Puerto Rico     795.0       748.0             NaN
      3      31239.0  for_sale  145000.0  4.0   2.0      0.10  1947675.0       Ponce  Puerto Rico     731.0      1800.0             NaN
      4      34632.0  for_sale   65000.0  6.0   2.0      0.05   331151.0    Mayaguez  Puerto Rico     680.0         NaN             NaN
    ...          ...       ...       ...  ...   ...       ...        ...         ...          ...       ...         ...             ...
2226377      23009.0      sold  359900.0  4.0   2.0      0.33   353094.0    Richland   Washington   99354.0      3600.0      2022-03-25
2226378      18208.0      sold  350000.0  3.0   2.0      0.10  1062149.0    Richland   Washington   99354.0      1616.0      2022-03-25
2226379      76856.0      sold  440000.0  6.0   3.0      0.50   405677.0    Richland   Washington   99354.0      3200.0      2022-03-24
2226380      53618.0      sold  179900.0  2.0   1.0      0.09   761379.0    Richland   Washington   99354.0       933.0      2022-03-24
2226381     108243.0      sold  580000.0  5.0   3.0      0.31   307704.0    Richland   Washington   99354.0      3615.0      2022-03-23

2226382 rows × 12 columns
```

```
In [3]: # Create df containing only listings in Utah state
        utah_df = df[df['state'] == 'Utah']
```

```
In [4]: # Create df containing only listings in Salt Lake County
        sl_county = utah_df[utah_df['city'].isin(['Millcreek', 'Bluffdale', 'Cottonwood Heights', 'Draper', 'Herriman',
                       'Holladay', 'Midvale', 'Murray', 'Riverton', 'Salt Lake City', 'Sandy', 'South Jordan', 'South Salt Lak
                      'Taylorsville', 'West Jordan', 'West Valley City', 'Kearns', 'Magna', 'Copperton', 'Emigration Canyon',
        sl_county
```

```
Out[4]:
```

Listing	brokered_by	status	price	bed	bath	acre_lot	street	city	state	zip_code	house_size	prev_sold_date
1180574	83262.0	for_sale	389900.0	4.0	2.0	0.14	1751463.0	Copperton	Utah	84006.0	2382.0	2014-11-13
1180575	84692.0	for_sale	389000.0	2.0	2.0	0.15	1749134.0	Bingham Canyon	Utah	84006.0	1641.0	2021-04-16
1180576	76462.0	for_sale	2300000.0	22.0	16.0	1.22	1723290.0	Copperton	Utah	84006.0	13312.0	NaN
1180577	82742.0	for_sale	459000.0	5.0	3.0	0.16	1475633.0	West Valley City	Utah	84128.0	1878.0	2019-01-23
1180578	76168.0	for_sale	425000.0	4.0	2.0	0.13	1809050.0	Sandy	Utah	84070.0	1970.0	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...
2006786	55199.0	sold	1399000.0	6.0	3.0	0.22	472477.0	Salt Lake City	Utah	84108.0	3373.0	2022-04-15
2006787	108354.0	sold	525000.0	2.0	1.0	0.18	457391.0	Salt Lake City	Utah	84115.0	1683.0	2022-04-20
2006789	32394.0	sold	839000.0	3.0	3.0	0.19	724315.0	Salt Lake City	Utah	84109.0	2276.0	2022-03-09
2006790	11512.0	sold	1195000.0	4.0	2.0	0.09	520020.0	Salt Lake City	Utah	84103.0	2328.0	2022-03-16
2006792	22738.0	sold	1350000.0	5.0	4.0	0.24	669512.0	Holladay	Utah	84124.0	3600.0	2022-04-15

2598 rows × 12 columns

```
In [5]: # Drop null values in variables used in later analysis
        Saltlake = sl_county.dropna(subset=['price', 'bed', 'bath','acre_lot','house_size'])
        Saltlake.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2323 entries, 1180574 to 2006792
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   brokered_by     2318 non-null   float64
 1   status          2323 non-null   object 
 2   price           2323 non-null   float64
 3   bed             2323 non-null   float64
 4   bath            2323 non-null   float64
 5   acre_lot        2323 non-null   float64
 6   street          2323 non-null   float64
 7   city            2323 non-null   object 
 8   state           2323 non-null   object 
 9   zip_code        2323 non-null   float64
 10  house_size      2323 non-null   float64
 11  prev_sold_date  1241 non-null   object 
dtypes: float64(8), object(4)
memory usage: 235.9+ KB
```

In [6]:
```
# Check for null values again
print(Saltlake.isnull().sum())
```

```
brokered_by          5
status               0
price                0
bed                  0
bath                 0
acre_lot             0
street               0
city                 0
state                0
zip_code             0
house_size           0
prev_sold_date    1082
dtype: int64
```

In [7]:
```
# Check for duplicates
duplicates = Saltlake.duplicated(keep=False)
duplicate_rows = Saltlake[duplicates]
print(Saltlake.duplicated().value_counts())
```

```
False    2323
dtype: int64
```

In [8]:
```
# Clean up and prepare the data
# Rename columns for consistency and comprehension
Saltlake = Saltlake.rename(columns={'acre_lot': 'Lot_Size'})
Saltlake = Saltlake.rename(columns={'house_size': 'House_Size'})
Saltlake = Saltlake.rename(columns={'price': 'Price'})
Saltlake = Saltlake.rename(columns={'city': 'City'})
```

In [9]:
```
# Group cities in categories "Higher" and "Lower" by splitting based on location of city within the county
Saltlake['City'] = Saltlake['City'].replace({
'Millcreek': 'Higher',
'Holladay': 'Higher',
'Murray': 'Higher',
'Salt Lake City': 'Higher',
'South Salt Lake': 'Higher',
'Taylorsville': 'Higher',
'West Valley City': 'Higher',
'Magna': 'Higher',
'Kearns': 'Higher',
'Emigration Canyon': 'Higher',
'Bluffdale': 'Lower',
'Cottonwood Heights': 'Lower',
'Draper': 'Lower',
'Herriman': 'Lower',
'Midvale': 'Lower',
'Riverton': 'Lower',
'Sandy': 'Lower',
'South Jordan': 'Lower',
'West Jordan': 'Lower',
'Copperton': 'Lower',
'Bingham Canyon': 'Lower'});
```

In [10]:
```
# Change data type for bed and bath
Saltlake['bed'] = Saltlake['bed'].astype(int);
Saltlake['bath'] = Saltlake['bath'].astype(int);

# Create categorical variable based on the ranges in bath, 1-2 = Lower, 3-4 = Higher. All else = other
Saltlake['Bathrooms'] = np.where(Saltlake['bath'].isin([1, 2]), 'Lower',
                                 np.where(Saltlake['bath'].isin([3, 4]), 'Higher', 'Other'))

# Create categorical variable based on the ranges in bed, 1-3 = Lower, 4-6 = Higher. All else = other
Saltlake['Bedrooms'] = np.where(Saltlake['bed'].isin([1, 2,3]), 'Lower',
                                 np.where(Saltlake['bed'].isin([4, 5, 6]), 'Higher', 'Other'))

# Drop remaining rows in bedroom and bathroom that = "Other"
Saltlake.drop(Saltlake[Saltlake['Bedrooms'] == 'Other'].index, inplace=True)
Saltlake.drop(Saltlake[Saltlake['Bathrooms'] == 'Other'].index, inplace=True)
```

In [11]:
```
# Change all Lower/Higher values to 1 or 0 by mapping
sl_map = {'Lower': 0, 'Higher': 1}
```

```python
# Apply the mapping to applicable columns
convert = ["City", "Bedrooms","Bathrooms"]
Saltlake[convert] = Saltlake[convert].replace(sl_map)
```

In [12]:
```python
# Drop remaining outliers
Saltlake.drop(Saltlake[Saltlake['Price'] >= 999999].index, inplace=True)
Saltlake.drop(Saltlake[Saltlake['Lot_Size'] >= 1.25].index, inplace=True)
Saltlake.drop(Saltlake[Saltlake['House_Size'] >= 15000].index, inplace=True)

# Change remaining numeric variables to integer
Saltlake['House_Size'] = Saltlake['House_Size'].astype(int)
Saltlake['Price'] = Saltlake['Price'].astype(int);
Saltlake['City'] = Saltlake['City'].astype(int);

# Create final df for model
model_df = Saltlake[["Lot_Size", "House_Size","Price", "Bedrooms","Bathrooms","City"]]

# Visually inspect the new dataframe
pd.set_option("display.max_columns", None)
print(model_df.head(5))
```

```
         Lot_Size  House_Size   Price  Bedrooms  Bathrooms  City
Listing
1180574      0.14        2382  389900         1          0     0
1180575      0.15        1641  389000         0          0     0
1180577      0.16        1878  459000         1          1     1
1180578      0.13        1970  425000         1          0     0
1180579      0.11        1464  449000         1          0     1
```

In [13]:
```python
# Review final model information
model_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1995 entries, 1180574 to 2006789
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Lot_Size    1995 non-null   float64
 1   House_Size  1995 non-null   int64
 2   Price       1995 non-null   int64
 3   Bedrooms    1995 non-null   int64
 4   Bathrooms   1995 non-null   int64
 5   City        1995 non-null   int64
dtypes: float64(1), int64(5)
memory usage: 109.1 KB
```

Analysis

D. Report on data-analysis process

Initial data exploration was completed by creating a univariate visualization to provide insights regarding the dependent variable "Price." Then univariate visualizations were created for each independent variable, along with bivariate visuals to demonstrate the relationships between the independent variables and "Price." An advantage of being able to view the plotted data is that it allows viewers to better understand the data through visuals, rather than through only numbers printed on a screen. A disadvantage of utilizing visualizations is that unless specified, the graphs will automatically be created with colors that do not suit the data. A fix to this was to add colors for each viz that appropriately fits the data. Based off the visuals created, the following interpretations were made:

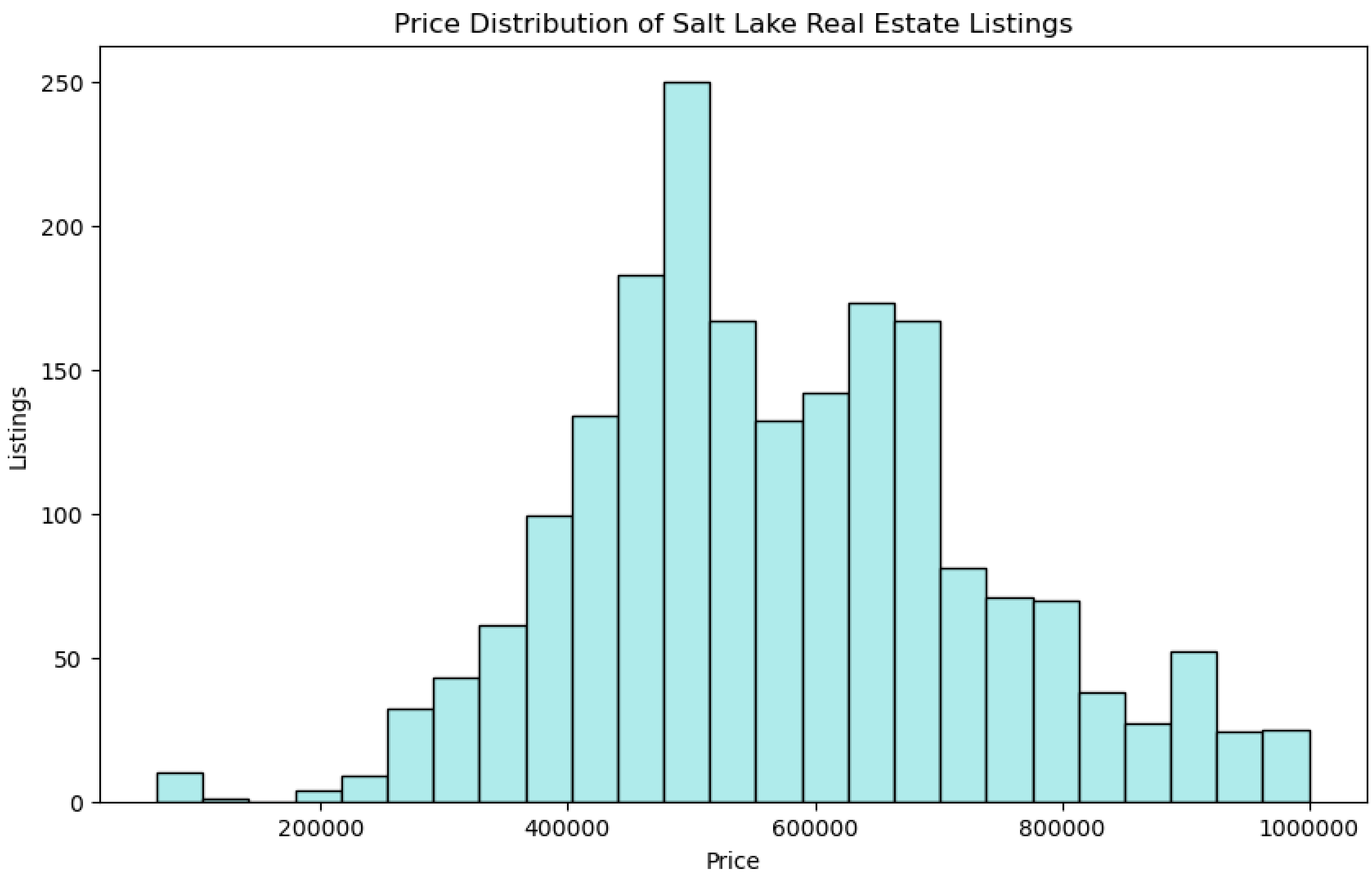
- Majority of listings fall between price range 375,000 to 700,000
- Distribution of bedrooms is almost even
    - 4-6 bedrooms make up 48.4% of listings
    - 1-3 bedrooms make up 51.6% of listings
- Average price for a home with 1-3 bedrooms is about 500,000
- Average price for a home with 4-6 bedrooms is > 600,000
- Listings with 1-2 bathrooms make up most of the listings at 60.4%
- Listings with 3-4 bathrooms make up 39.6% of listings
- Average price for a home with 1-2 bathrooms is 500,000
- Average price for a home with 3-4 bathrooms is > 600,000
- Cities located in the southern half of Salt Lake County make up most of the listings at 58.2%
- Northern located cities make up 41.8% of listings
- Average price of a southern city is 600,000
- Average price of a northern city is 500,000
- Most listings fall between ~1,000 to 4,000 square feet
- In relation to house size, the higher the square footage, the higher the price
- Most listings have a lot size of about 0.3 acres or less
- Greater distribution of price is seen in listings with 0.0 to 0.2 acres as compared to higher acre values
- Higher acreage is associated with higher price

The scripts used and visualizations created can be seen in the images below:

```
In [14]: # Import remaining libraries/packages needed for model
         import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from statsmodels.formula.api import ols
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn import preprocessing
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression
import warnings
# Suppress all warnings
warnings.filterwarnings("ignore")
```

In [15]:

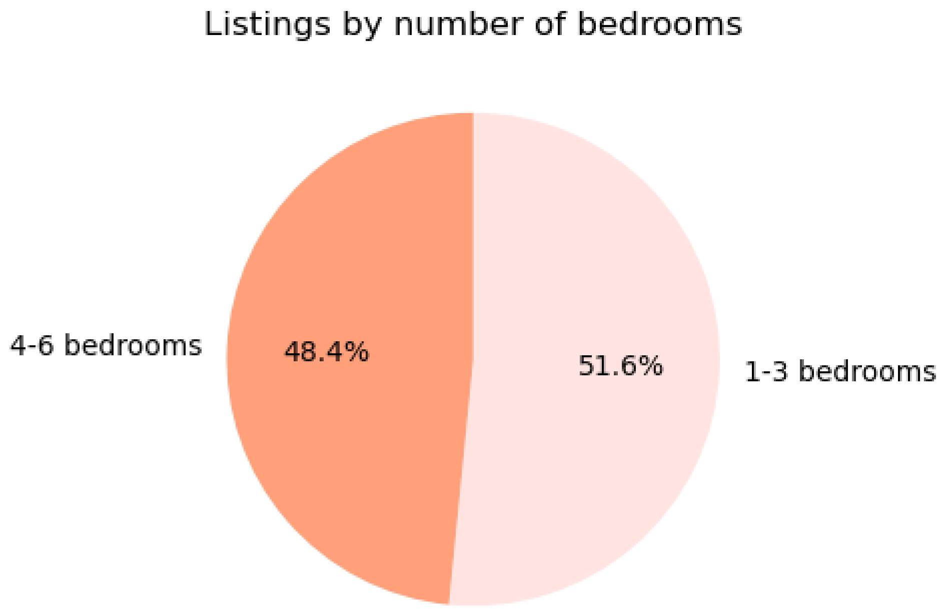
```
# Create univariate visualization of dependent variable "Price"
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
plt.hist(model_df['Price'], bins=25, color='paleturquoise', edgecolor='black')
plt.xlabel('Price')
plt.ylabel('Listings')
plt.title('Price Distribution of Salt Lake Real Estate Listings')
# Keep full price rather than scientific notion
plt.ticklabel_format(style='plain', axis='x')
plt.show()
```

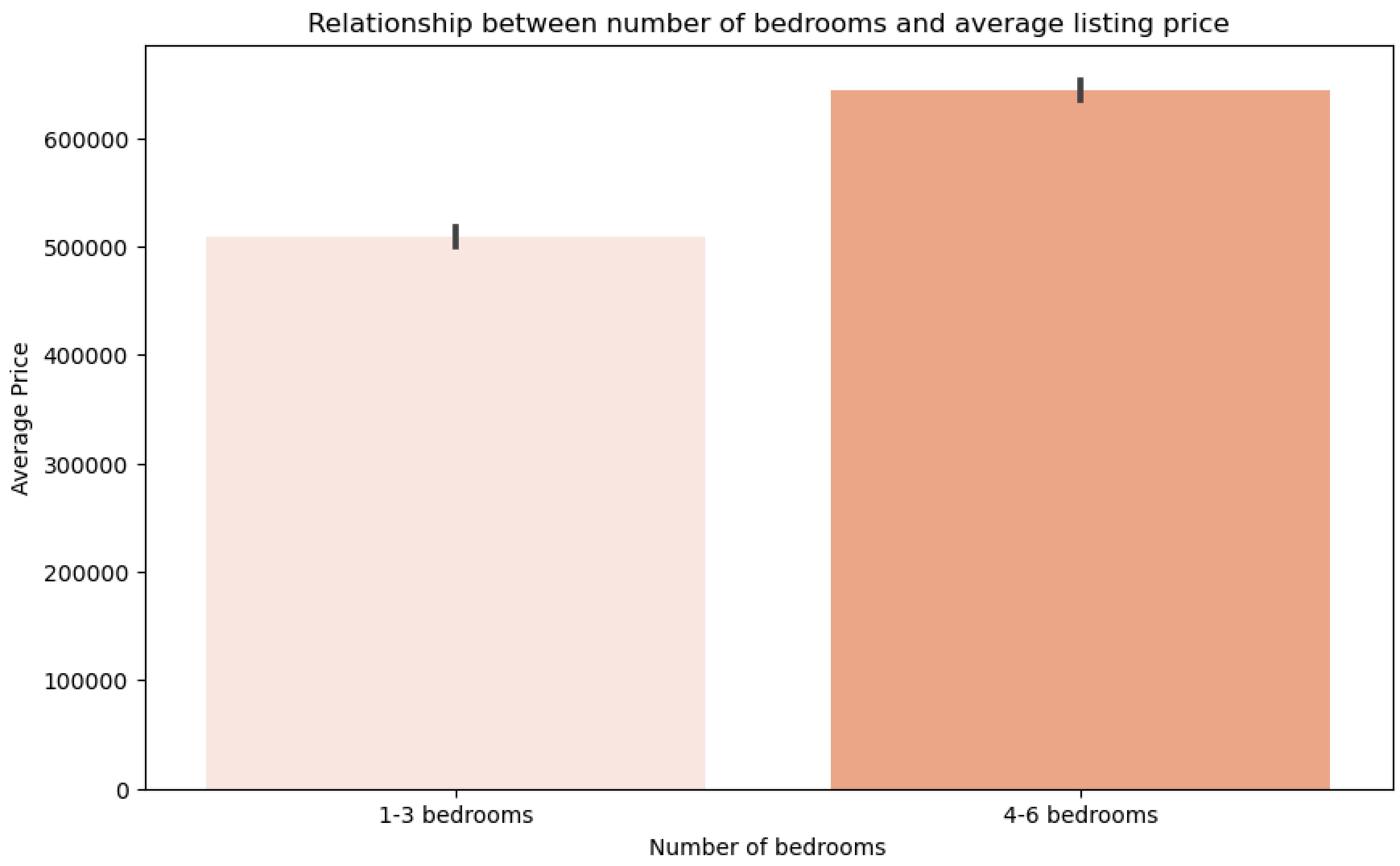


```
In [16]: # Create univariate visualization of variable "Bedroom"
plt.figure(figsize=[14, 4])
# Create subplot for bedrooms
plt.subplot(1, 2, 1)
plt.title("Listings by number of bedrooms")
Bedrooms_counts = model_df["Bedrooms"].value_counts()
Bedrooms_labels = ["1-3 bedrooms", "4-6 bedrooms"]
colors = ['mistyrose', 'lightsalmon']
plt.pie(Bedrooms_counts, labels = Bedrooms_labels, colors=colors, autopct='%1.1f%%', startangle=90, counterclock=False
plt.xticks([])
plt.yticks([])

# Create Bivariate visualization of Bedrooms vs Price
```

```
colors = ['mistyrose', 'lightsalmon']
plt.figure(figsize=(10, 6))
sns.barplot(data=model_df, x='Bedrooms', y='Price', estimator=np.mean, palette=colors)
plt.xlabel('Number of bedrooms')
plt.colors=colors
plt.xticks(ticks=[0, 1], labels= Bedrooms_labels)
plt.ylabel('Average Price')
plt.title('Relationship between number of bedrooms and average listing price')
plt.show()
```

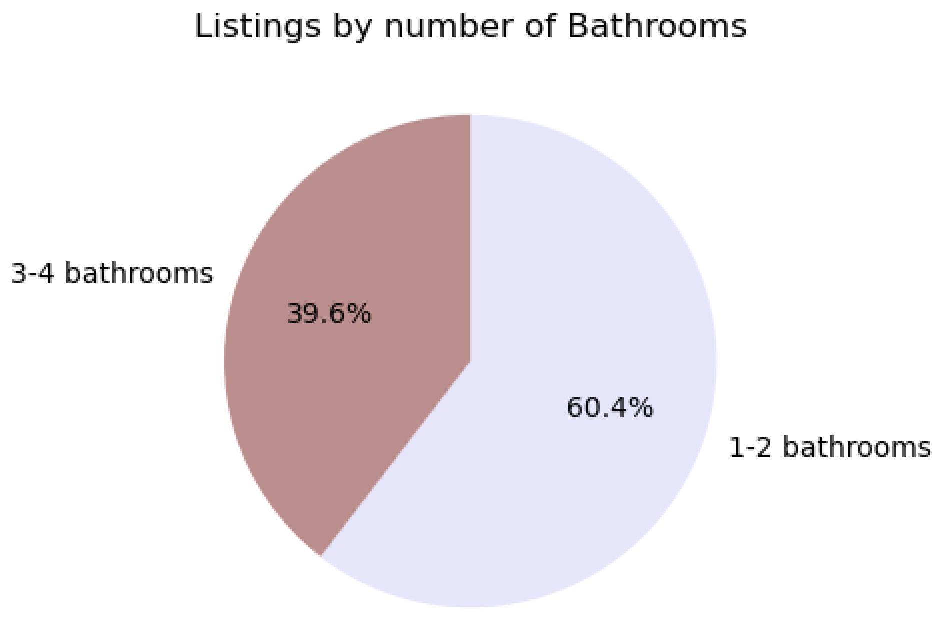


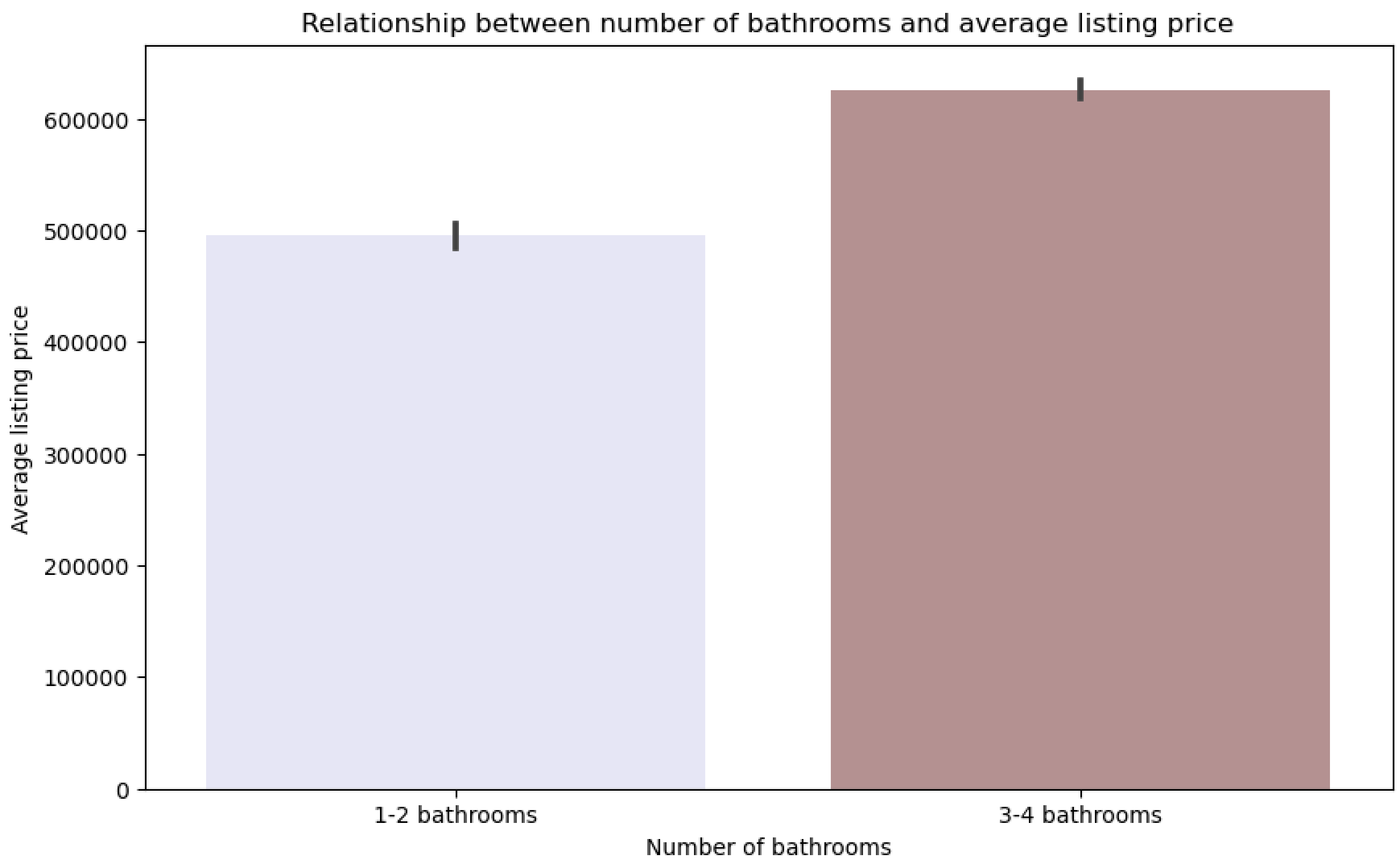


In [17]:
```
# Create univariate visualization of variable "Bathrooms"
plt.figure(figsize=[14, 4])
plt.subplot(1, 2, 1)
plt.title("Listings by number of Bathrooms")
Bathrooms_counts = model_df["Bathrooms"].value_counts()
Bathrooms_labels = ["1-2 bathrooms", "3-4 bathrooms"]
colors = ['lavender', 'rosybrown']
plt.pie(Bathrooms_counts, labels=Bathrooms_labels, colors=colors, autopct='%1.1f%%', startangle=90, counterclock=False
plt.xticks([])
plt.yticks([])

# Create Bivariate visualization of Bathrooms vs Price
colors = ['lavender', 'rosybrown']
```

```
plt.figure(figsize=(10, 6))
sns.barplot(data=model_df, x='Bathrooms', y='Price', estimator=np.mean, palette=colors)
plt.xlabel('Number of bathrooms')
plt.colors=colors
plt.xticks(ticks=[0, 1], labels=Bathrooms_labels)
plt.ylabel('Average listing price')
plt.title('Relationship between number of bathrooms and average listing price')
plt.show()
```

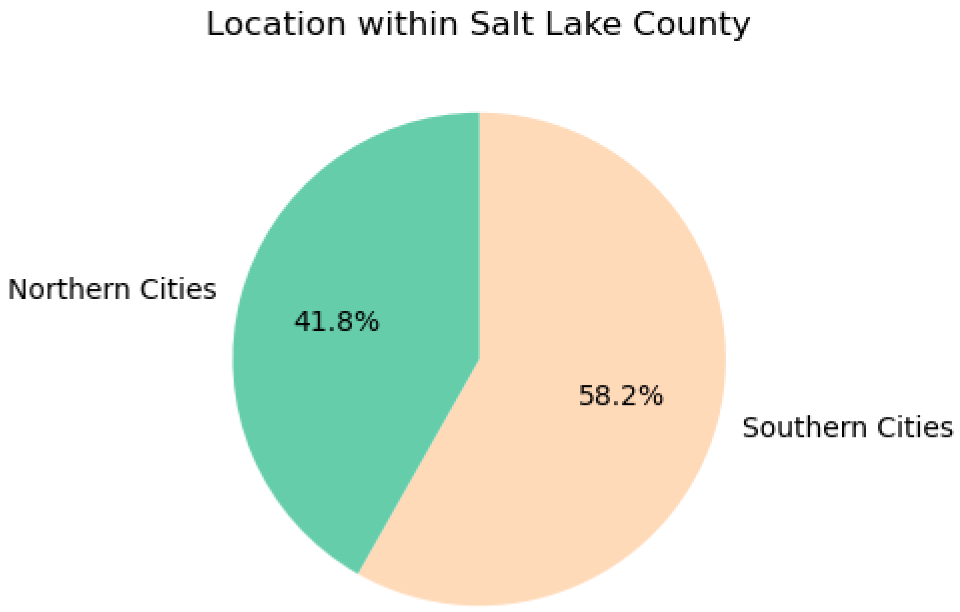


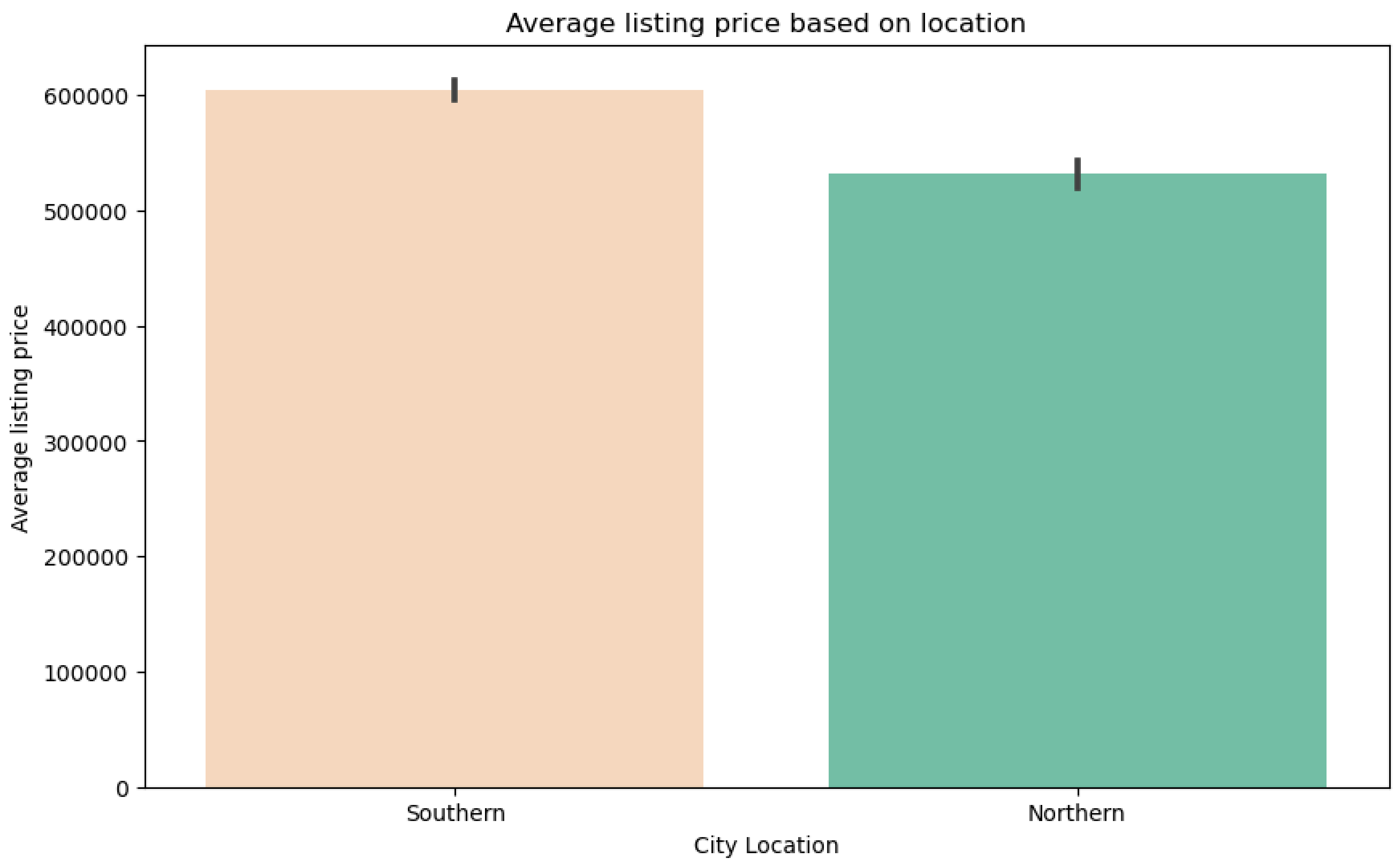


In [18]:
```
# Create univariate visualization of variable "City"
plt.figure(figsize=[14, 4])
colors = ['peachpuff', 'mediumaquamarine']
plt.subplot(1, 2, 1)
plt.title("Location within Salt Lake County")
City_counts = model_df["City"].value_counts()
City_labels = ["Southern Cities", "Northern Cities"]
plt.pie(City_counts, labels=City_labels, colors=colors, autopct='%1.1f%%', startangle=90, counterclock=False)
plt.xticks([])
plt.yticks([])

# Create Bivariate visualization of City vs Price
colors = ['peachpuff', 'mediumaquamarine']
```

```
plt.figure(figsize=(10, 6))
sns.barplot(data=model_df, x='City', y='Price', estimator=np.mean, palette=colors)
plt.xlabel('City Location')
Loc_labels = ["Southern", "Northern"]
plt.colors=colors
plt.xticks(ticks=[0, 1], labels= Loc_labels)
plt.ylabel('Average listing price')
plt.title('Average listing price based on location')
plt.show()
```

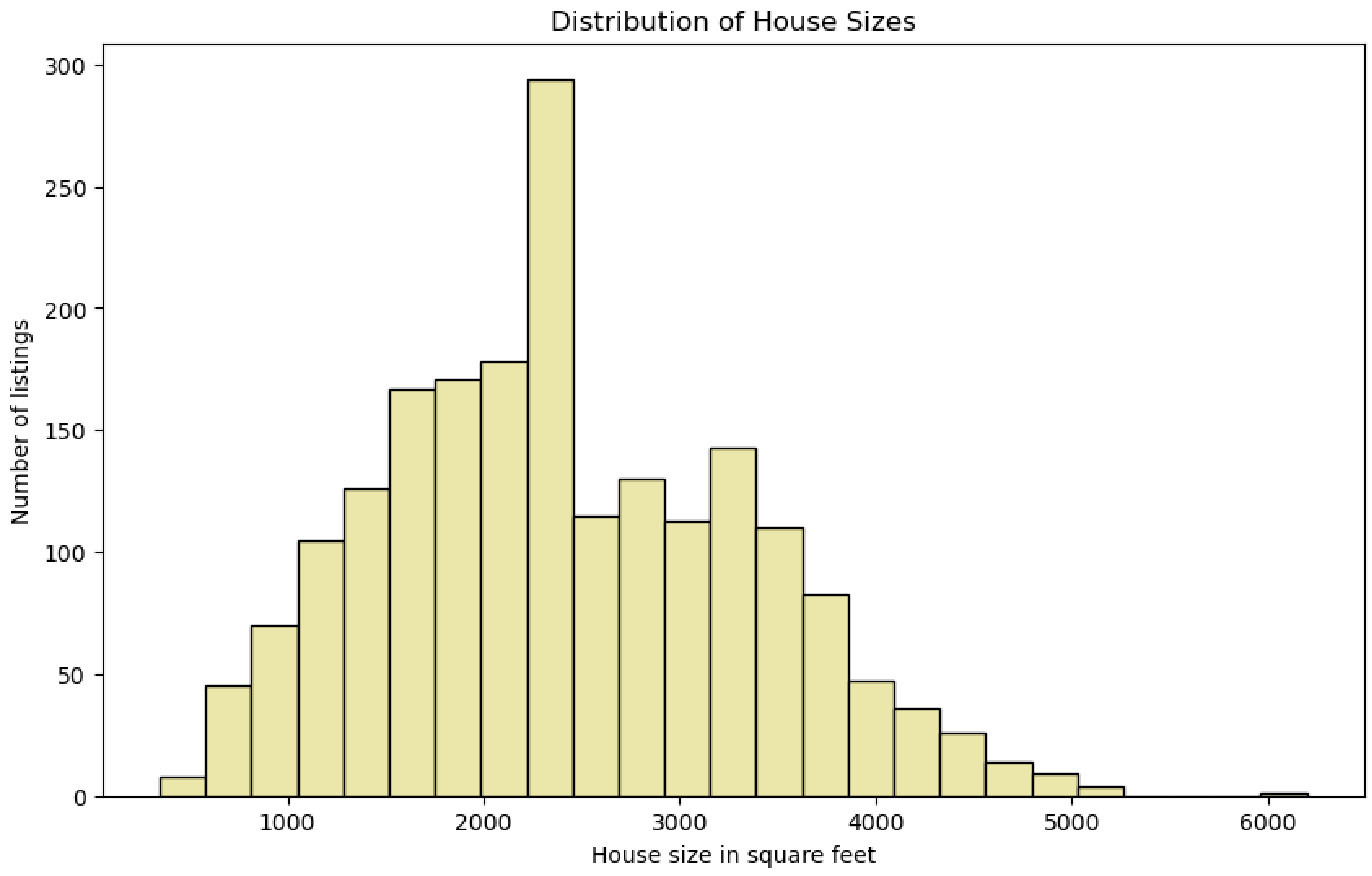


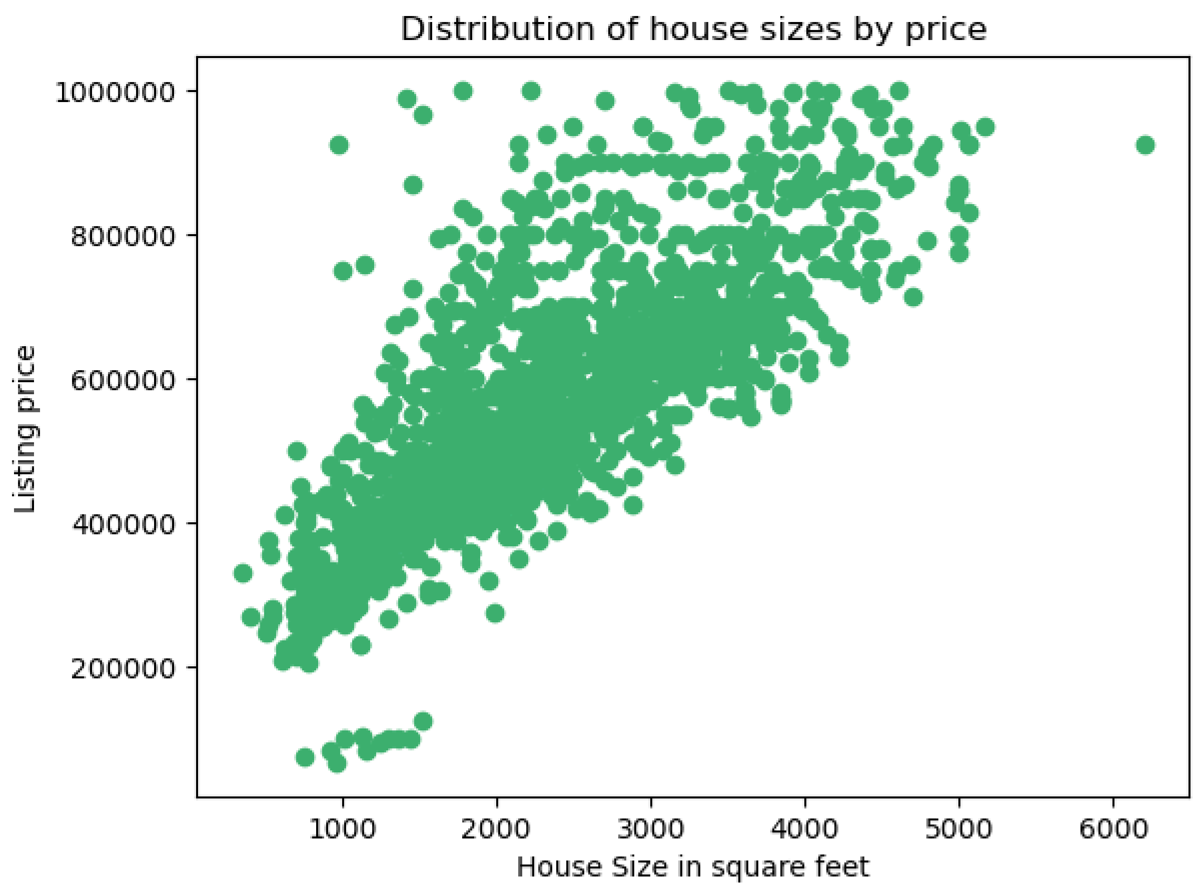


In [19]:
```
# Create univariate visualization of variable "House Size"
plt.figure(figsize=(10, 6))
plt.hist(model_df['House_Size'], bins=25, color='palegoldenrod', edgecolor='black')
plt.xlabel('House size in square feet')
plt.ylabel('Number of listings')
plt.title('Distribution of House Sizes')
plt.ticklabel_format(style='plain', axis='y')
plt.show()

# Create Bivariate visualization of House Size vs Price
plt.scatter(model_df['House_Size'], model_df['Price'], color = 'mediumseagreen')
plt.xlabel('House Size in square feet')
plt.ylabel('Listing price')
```

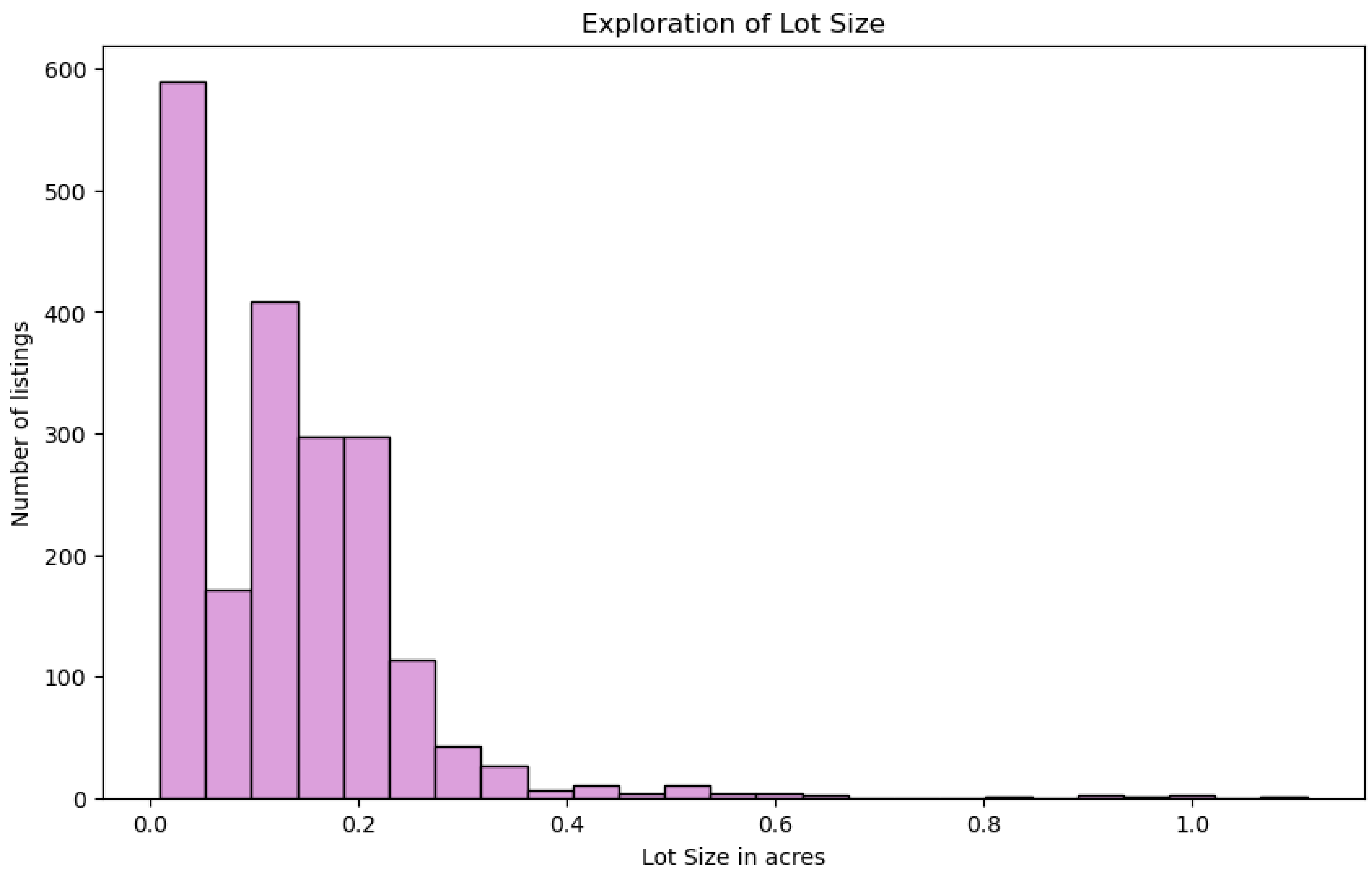
```python
plt.title('Distribution of house sizes by price')
# Keep full price rather than scientific notion
plt.ticklabel_format(style='plain', axis='y')
plt.show()
```

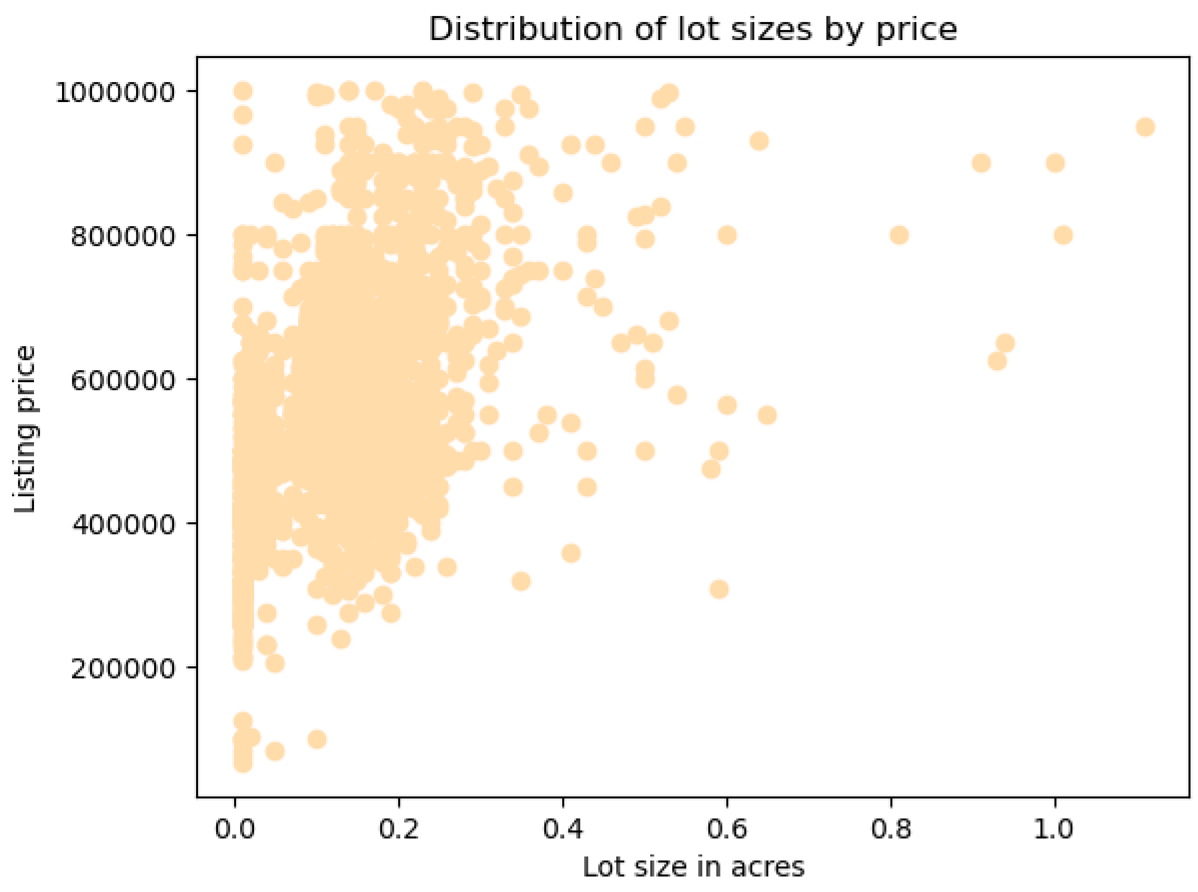




In [20]:
```
# Create univariate visualization of variable "Lot Size"
plt.figure(figsize=(10, 6))
plt.hist(model_df['Lot_Size'], bins=25, color='plum', edgecolor='black')
plt.xlabel('Lot Size in acres')
plt.ylabel('Number of listings')
plt.title('Exploration of Lot Size')
plt.ticklabel_format(style='plain', axis='y')
plt.show()

# Create Bivariate visualization of Lot Size vs Price
plt.scatter(model_df['Lot_Size'], model_df['Price'], color = 'navajowhite')
plt.xlabel('Lot size in acres')
plt.ylabel('Listing price')
plt.title('Distribution of lot sizes by price')
# Keep full price rather than scientific notion
plt.ticklabel_format(style='plain', axis='y')
plt.show()
```


Exploration of Lot Size
Number of listings
0
100
200
300
400
500
600
0.0
0.2
0.4
0.6
0.8
1.0
Lot Size in acres



In [21]:
```python
# Save transformed dataframe
model_df.to_csv('model_df.csv', index=False)
```

D. Report on data-analysis process continued

Following the exploratory steps above, the data was normalized, and feature selection was performed. Although measures were taken in the data preparation stages to avoid multicollinearity, there is still a possibility of high correlation among independent variables present in the model. Taking an extra measure of using the VIF value to eliminate features was necessary to ensure a reliable model. Variables with a VIF value of 5 or greater were removed one by one starting with the highest value. The advantage of completing this step is that it will check for and correct any multicollinearity present in the data (Statology, 2020). Fortunately for this model, only one variable needed to be removed due to the VIF value being 7.39 for "House Size." Once any possibility of multicollinearity was removed, an additional feature selection method was executed. Backward stepwise elimination was chosen to assess the variables based on the statistical significance of the p-value. Any variables found to have a p-value of less than or equal to 0.05 would indicate no statistical significance and would be removed.

There was no necessity to remove any additional variables however due to all p-values in the model having a value of 0.00. No disadvantages were discovered when using these reduction methods, as they were both convenient and reliable.

Given that all the p-values were 0.00 and no more variables needed to be removed, the final model was already created and ready for review. Values related to the residual standard error, R-squared, and Durbin-Watson were assessed to ensure model efficacy. An added measure to plotting the residuals was completed too, though this is discussed in more detail in the data summary section below.

Based off the regression results of the model pertaining to RSE, R^2, and Durbin-Watson, the following observations were made:

- The value for residual standard error (RSE) is 0.13 which indicates:
  - The model fits the data well
  - There is less variation between the model and the true data points
- The value for R-squared is 0.406
  - This means that the independent variables explain around 41% of variability in the dependent variable
- The value for Durbin-Watson is 1.346
  - A value of 2 indicates no autocorrelation amongst residuals, so this model's value is satisfactory when taking p-values and RSE into consideration.

See the images below for the steps explained above.

In [22]:
```
# Check for high multicollinearity (VIF > 5) among variables
X = model_df[["Bedrooms", "Bathrooms", "Lot_Size", "City","House_Size"]]
vif_df = pd.DataFrame()
vif_df["Variable"] = X.columns

vif_df["VIF"] = [variance_inflation_factor(X.values, i)
for i in range(len(X.columns))]

print(vif_df)
```

```
    Variable       VIF
0   Bedrooms  2.562805
1  Bathrooms  3.976381
2   Lot_Size  3.542308
3       City  1.487503
4 House_Size  7.392724
```

In [23]:
```
# Remove "House_Size" and recheck for multicollinearity
X = model_df[["Bedrooms", "Bathrooms", "Lot_Size", "City"]]
vif_df = pd.DataFrame()
vif_df["Variable"] = X.columns
```

```
vif_df["VIF"] = [variance_inflation_factor(X.values, i)
for i in range(len(X.columns))]

print(vif_df)
```

```
    Variable       VIF
0   Bedrooms  2.500293
1  Bathrooms  1.761538
2   Lot_Size  2.568113
3       City  1.421663
```

In [24]:

```
# Normalize the data to allow better interpretation of the results
scaler = MinMaxScaler()
norm_df = scaler.fit_transform(model_df)
norm_df = pd.DataFrame(norm_df, columns=model_df.columns)
norm_df
```

Out[24]:

	Lot_Size	House_Size	Price	Bedrooms	Bathrooms	City
0	0.118182	0.347573	0.345774	1.0	0.0	0.0
1	0.127273	0.220950	0.344809	0.0	0.0	0.0
2	0.136364	0.261449	0.419884	1.0	1.0	1.0
3	0.109091	0.277170	0.383419	1.0	0.0	0.0
4	0.090909	0.190704	0.409159	1.0	0.0	1.0
...	...	...	...	...	...	...
1990	0.200000	0.365003	0.651544	1.0	0.0	1.0
1991	0.163636	0.152256	0.494959	0.0	0.0	0.0
1992	0.027273	0.269310	0.474582	0.0	1.0	0.0
1993	0.154545	0.228127	0.490669	0.0	0.0	1.0
1994	0.163636	0.329460	0.827435	0.0	1.0	1.0

1995 rows × 6 columns

In [25]:

```
# Create final model for Multiple Linear Regression
# Set dependent variable
y = norm_df.Price
# Set independent variables
X = norm_df[["Bedrooms", "Bathrooms", "Lot_Size", "City"]].assign(const=1)
model = sm.OLS(y, X)
```

```
reg_results = model.fit()
print(reg_results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Price   R-squared:                       0.406
Model:                            OLS   Adj. R-squared:                  0.404
Method:                 Least Squares   F-statistic:                     339.5
Date:                Thu, 25 Apr 2024   Prob (F-statistic):          6.24e-223
Time:                        05:13:34   Log-Likelihood:                 1172.6
No. Observations:                1995   AIC:                            -2335.
Df Residuals:                    1990   BIC:                            -2307.
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Bedrooms       0.0619      0.007      9.034      0.000       0.048       0.075
Bathrooms      0.1188      0.007     16.743      0.000       0.105       0.133
Lot_Size       0.6975      0.034     20.748      0.000       0.632       0.763
City          -0.0362      0.007     -5.287      0.000      -0.050      -0.023
const          0.3786      0.007     50.539      0.000       0.364       0.393
==============================================================================
Omnibus:                      184.746   Durbin-Watson:                   1.346
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              272.373
Skew:                           0.707   Prob(JB):                     7.16e-60
Kurtosis:                       4.131   Cond. No.                         15.4
==============================================================================

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

In [26]:
```
# Retrieve residual standard error
print("The residual standard error of the model is")
reg_results.resid.std(ddof=X.shape[1])
```

```
The residual standard error of the model is
```

Out[26]:
```
0.1345981533659512
```

Data Summary and Implications

E. Summarize the implications of the data analysis

The regression equation for the reduced model:

Y^ = 0.3786 + 0.0619 (Bedrooms) + 0.1188 (Bathrooms) + 0.6975 (Lot Size) - 0.0362 (City)

For the final model the coefficients for the remaining columns and their significance are as follows:

- With a coefficient value of 0.0619 and all other factors held constant, listings with a higher number of bedrooms are, on average, associated with a higher price as compared to houses with a lower number of bedrooms.
- With a coefficient value of 0.1188 and all other factors held constant, listings with a higher number of bathrooms are, on average, associated with a higher price as compared to houses with a lower number of bathrooms.
- With a coefficient value of 0.6975 and all other factors held constant, listings with more acres on the lot are, on average, associated with a higher price as compared to houses with lower acreage.
- With a coefficient value of -0.0362 and all other factors held constant, listings within cities in the northern half of Salt Lake County, on average, are associated with a lower price in comparison to houses listed in the southern half of the county.

Referring to the implications found after viewing the univariate and bivariate visualizations, the model's predictions come as no surprise. The bivariate visualizations demonstrated higher prices for listings with more bedrooms, more bathrooms, and for those with higher acreage in comparison to lower values in these variables. Also, they showed that listings in the southern part of the county have a higher price than listings in the northern part of the county. The p-values, R-squared value, the residual standard error value, and the Durbin-Watson value indicate that the model does have statistical significance. Unfortunately for the hypothesis of this research question, there is evidence stacking up to accept the null hypothesis. Specifically the model's accuracy, which is being based off the R-squared value, or otherwise the statistical measure of how well the model is at making predictions on a scale of 0 to 1. Preferably, the final model would have a R-squared value of > 0.70, which would correlate to the independent variables explaining at least 70% of variability in the dependent variable price.

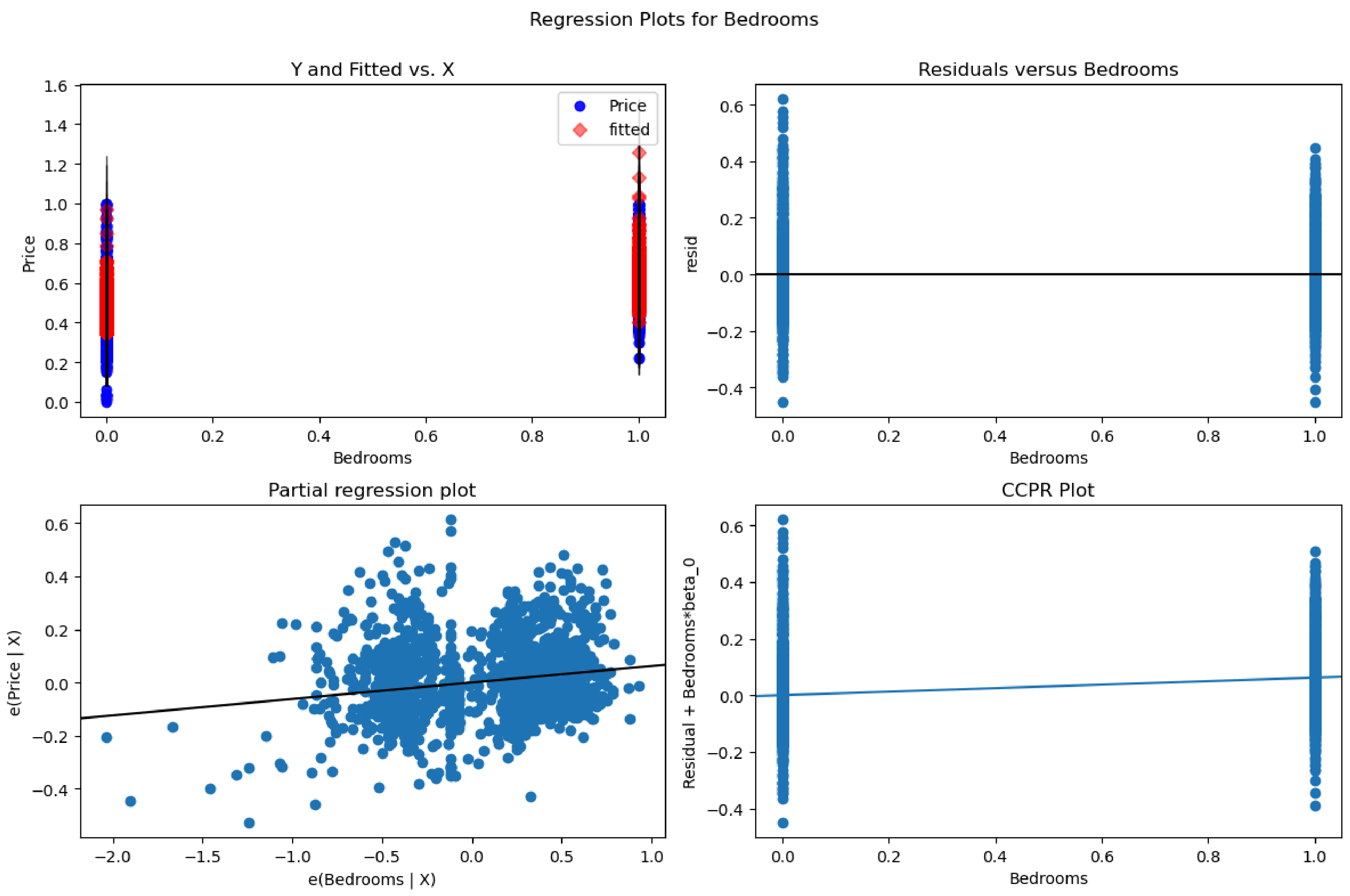
To further analyze the model to provide additional evidence either in support or against the hypothesis, it was necessary to also review the residuals, or the difference between actual values and predicted values. To review the residuals, regression plots were created and can be seen above in the images included below. In the residuals plots (top right visualization) for variables Bedrooms, Bathrooms, and City the data points are plotted along the y axis at various points rather than being centered around the line of best fit, which is 0. Only the residuals plot for variable Lot Size shows a differing distribution of data points; with many closer to the line of best fit in comparison to the other variables mentioned previously. However, the visual does not show a normal distribution along the line of best fit, but rather a potential pattern which can skew the analysis. Therefore, this model cannot satisfy the assumptions for multiple linear regression and is not a reliable model. (Zach, 2020)

Having unsatisfied assumptions for multiple linear regression, irregular residuals plots, and statistical values mentioned above, there was sufficient evidence to accept the null hypothesis. It was not possible to create a multiple linear regression model to predict housing prices in Salt Lake County using the USA real estate dataset provided via Kaggle with an accuracy of > 70%. A major limitation of the model is the lack of additional listing data, such as the duration of listings, type of home (Ie: condominium, townhome, etc.) and other potential factors that could provide more insight to the Salt Lake real estate market. Additionally, although the dataset contained over 2 million instances initially, the cleaning and transformation processes reduced the data frame to only 1,995 rows. Overall, there is not enough data to move forward with utilizing this reduced model to make predictions regarding price.

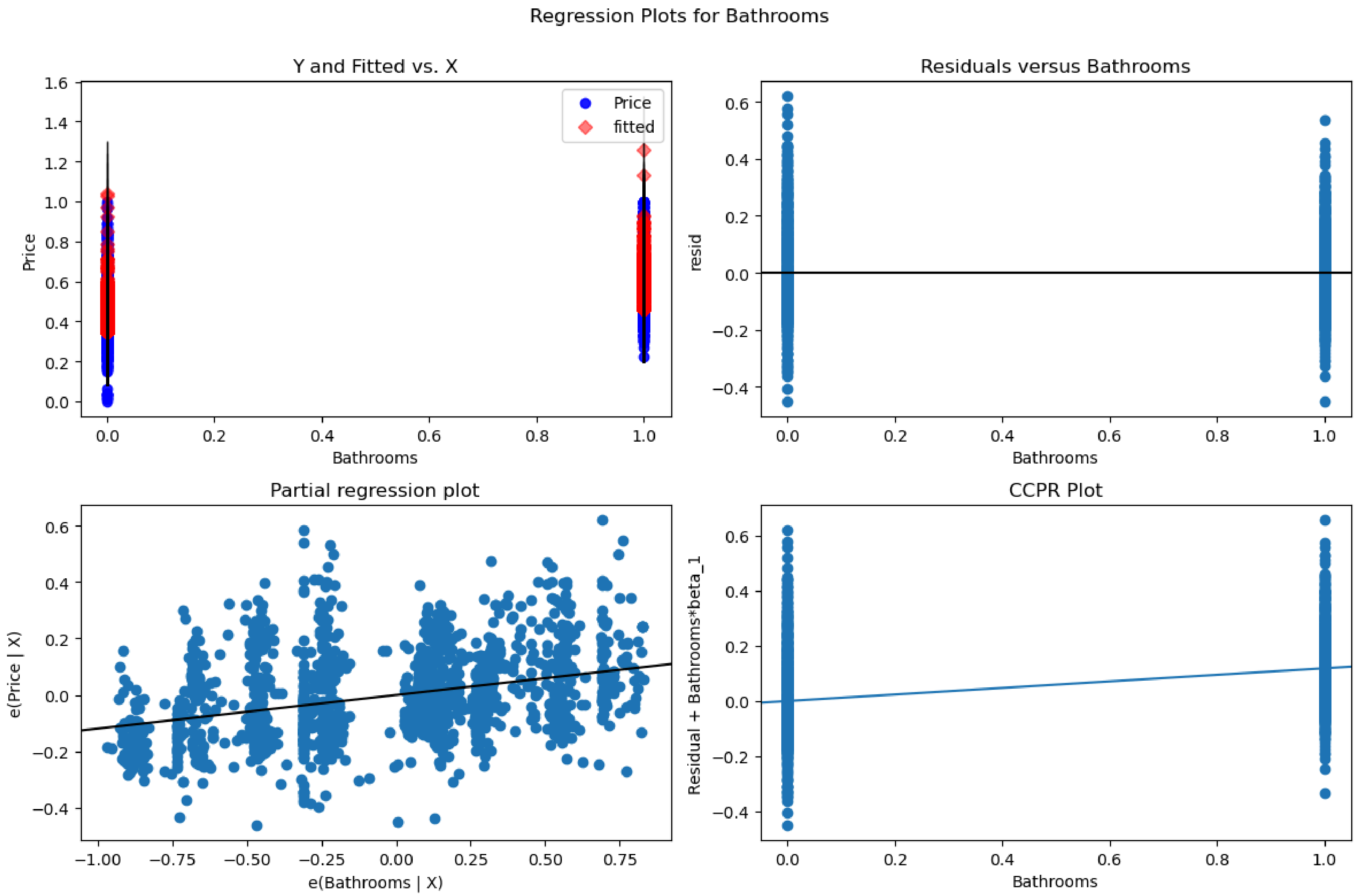
There are still benefits to reviewing the results of this analysis for business purposes. Rather than using the model for predictions, an alternative course of action for real estate investors would be to utilize the information provided by the initial exploratory visualizations. Although the regression model may not be reliable enough to make accurate predictions, the bivariate explorations still could prove useful. There are relationships between the independent variables and dependent variable that could still help investors improve their portfolio(s). Perhaps growing their investments to include more properties in southern Salt Lake County cities, with more bedrooms, more bathrooms, and larger lot sizes. Or in the context of preparing to sell properties that may be undervalued (Ie: located in the northern county area, sparce number of bedrooms, etc.) investors could consider implementing changes to appreciate the value through renovations or small upgrades.

Although the hypothesis has been rejected and a highly accurate multiple linear regression model cannot be created in regarding Salt Lake County, Utah, it may still be possible to create predictive models related to other geographical areas and in other contexts. For instance, businesses looking to expand into new locations could use the USA real estate data to make predictions as to which state or city to move their business franchise into if there is predicted housing growth in that area given past trends. Alternatively, a clustering technique could be executed on the data to provide mortgage lenders greater understanding of listings in a particular geographical area. Truthfully with 2 million rows to work with, the possibilities are endless.

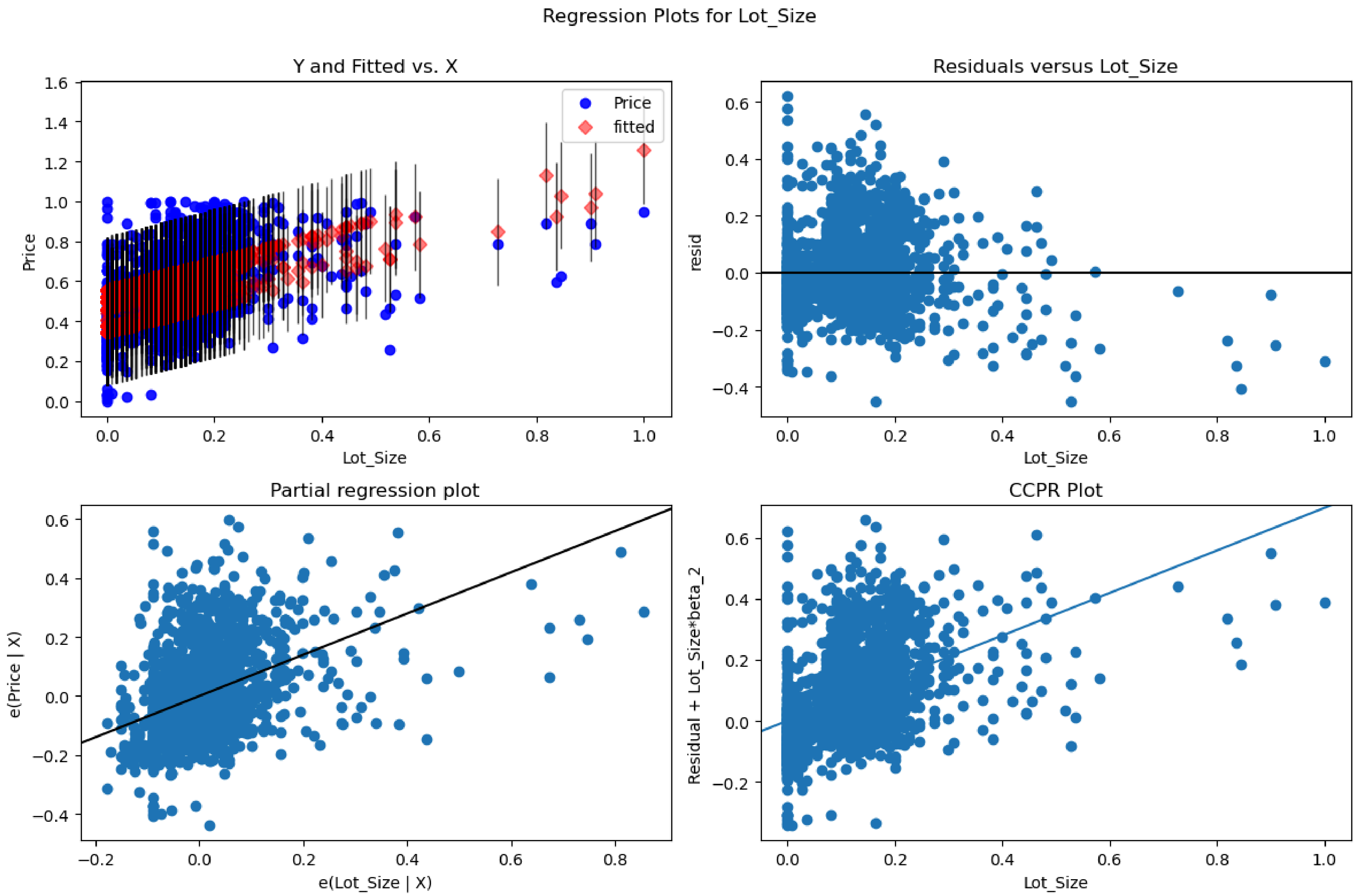
In [27]:
```python
# Create Regression Plot for Bedrooms variable
fig = plt.figure(figsize = [12,8])
sm.graphics.plot_regress_exog(reg_results, 'Bedrooms', fig=fig);
```



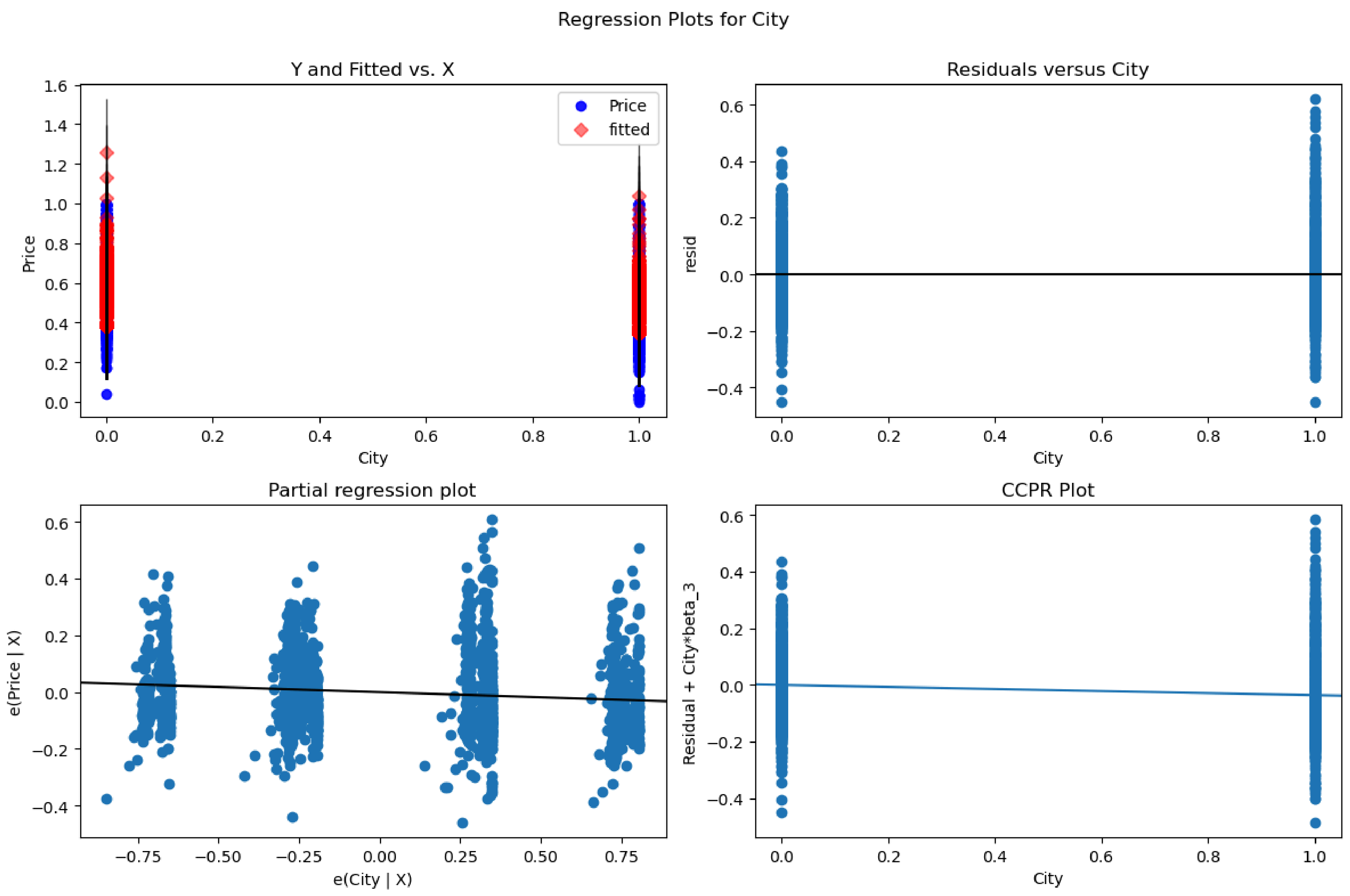
```
In [28]: # Create Regression Plot for Bathrooms variable
         fig = plt.figure(figsize = [12,8])
         sm.graphics.plot_regress_exog(reg_results, 'Bathrooms', fig=fig);
```



```
In [29]: # Create Regression Plot for Lot Size variable
         fig = plt.figure(figsize = [12,8])
         sm.graphics.plot_regress_exog(reg_results, 'Lot_Size', fig=fig);
```



In [30]:
```
# Create Regression Plot for City variable
fig = plt.figure(figsize = [12,8])
sm.graphics.plot_regress_exog(reg_results, 'City', fig=fig);
```



```
In [31]: # Save final dataframe
         norm_df.to_csv('norm_df.csv', index=False)
```

F. Acknowledge sources, using in-text citations and references, for content that is quoted.

Churchill, Briana. (2023, Aug 22). Performance Assessment: Predictive Modeling Task 1. Assignment for MS Data Analytics Course D208. Western Governors University.

Guiding Our Growth: The Future of Housing in Utah. Retrieved April 14, 2024, from https://guidingourgrowth.utah.gov/the-future-of-housing-in-utah/

Hayes, A. (2024, Mar 29). Multicollinearity: Meaning, Examples, and FAQs. Retrieved April 16, 2024, from https://www.investopedia.com/terms/m/multicollinearity.asp

How can predictive analytics help you make profitable real estate investments? (2024, Mar 20). Retrieved April 14, 2024, from https://www.linkedin.com/advice/3/how-can-predictive-analytics-help-you-make-profitable-4wbec#:~:text=Improved%20Decision%20Making%3A%20Predictive%20analytics,investment%20decisions%20in%20real%20estate.&text=Ris

How to Calculate VIF in Python. (2022, July 20) Retrieved April 14, 2024, from https://www.statology.org/how-to-calculate-vif-in-python/

Hughes, R., et. Al., (2019, Mar 16). Accounting for missing data in statistical analyses: multiple imputation is not always the answer. Retrieved April 14, 2024, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6693809/

ML | Multiple Linear Regression using Python. (2023, Jan 25). Retrieved April 15, 2024, from https://www.geeksforgeeks.org/ml-multiple-linear-regression-using-python/

Multiple Linear Regression using R to predict housing prices. (2023, Nov 6). Retrieved April 16, 2024, from https://www.geeksforgeeks.org/multiple-linear-regression-using-r-to-predict-housing-prices/

Ogunbiyi, I., (2022, Aug 1). Top Evaluation Metrics for Regression Problems in Machine Learning. Retrieved April 21, 2024, from https://www.freecodecamp.org/news/evaluation-metrics-for-regression-problems-machine-learning/#:~:text=A%20regression%20model%20can%20only,residuals%20as%20being%20a%20distance.

SAS vs R vs Python. (2023, June 12). Retrieved April 14, 2024, from https://www.geeksforgeeks.org/sas-vs-r-vs-python/

Sakib, A. (2024, Mar 30). USA Real Estate Dataset. Retrieved April 13, 2024, from https://www.kaggle.com/datasets/ahmedshahriarsakib/usa-real-estate-dataset

What are the key features and benefits of using Tableau for data visualization? (2024, Mar 20). Retrieved April 16, 2024, from https://www.linkedin.com/advice/0/what-key-features-benefits-using-tableau-data-visualization