notebook_final

April 23, 2021

1 Final Project Submission

- Student name: Jonathan Lee
- Student pace: full time
- Scheduled project review date/time: April 27, 2pm
- Instructor name: James Irving

1.1 TABLE OF CONTENTS

Click to jump to matching Markdown Header.

- **Introduction**
- **OBTAIN**
- **SCRUB**
- **EXPLORE**
- **MODEL**
- **iNTERPRET**
- **Conclusions/Recommendations** ____

2 INTRODUCTION

This analysis focuses on creating a multiple regression model based on housing data from King County, Washington. We will work through an exploratory data analysis to clean the data that we have to prepare it for modeling, as well as working through an iterative approach to refining our model. The goal of this analysis is to create a model which explains how different attributes affect the value of a housing property in King County, and to extract specific variables which we can use to recommend to a homeowner in King County how to increase the value of his/her home.

3 OBTAIN

The data that we will use in this analysis has been provided as a .csv file. We will inspect the data types to determine how to approach the cleansing process.

[1]:
```
# Import packages to be used in notebook.
import pandas as pd
import numpy as np
import seaborn as sns
```

```
import matplotlib.pyplot as plt
from matplotlib.ticker import FuncFormatter
from matplotlib.gridspec import GridSpec

import statsmodels.api as sm
import statsmodels.stats.api as sms
import statsmodels.formula.api as smf

from scipy import stats

from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler

import warnings
warnings.simplefilter(action="ignore", category=FutureWarning)

%matplotlib inline
```

[2]:
```
# Load housing data
df = pd.read_csv('data/kc_house_data.csv')
display(df.head(5), df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   id             21597 non-null  int64
 1   date           21597 non-null  object
 2   price          21597 non-null  float64
 3   bedrooms       21597 non-null  int64
 4   bathrooms      21597 non-null  float64
 5   sqft_living    21597 non-null  int64
 6   sqft_lot       21597 non-null  int64
 7   floors         21597 non-null  float64
 8   waterfront     19221 non-null  float64
 9   view           21534 non-null  float64
 10  condition      21597 non-null  int64
 11  grade          21597 non-null  int64
 12  sqft_above     21597 non-null  int64
 13  sqft_basement  21597 non-null  object
 14  yr_built       21597 non-null  int64
 15  yr_renovated   17755 non-null  float64
 16  zipcode        21597 non-null  int64
 17  lat            21597 non-null  float64
 18  long           21597 non-null  float64
 19  sqft_living15  21597 non-null  int64
```

```
 20  sqft_lot15     21597 non-null  int64
dtypes: float64(8), int64(11), object(2)
memory usage: 3.5+ MB
```

```
           id        date     price  bedrooms  bathrooms  sqft_living  \
0  7129300520  10/13/2014  221900.0         3       1.00         1180
1  6414100192   12/9/2014  538000.0         3       2.25         2570
2  5631500400   2/25/2015  180000.0         2       1.00          770
3  2487200875   12/9/2014  604000.0         4       3.00         1960
4  1954400510   2/18/2015  510000.0         3       2.00         1680

   sqft_lot  floors  waterfront  view  …  grade  sqft_above  sqft_basement  \
0      5650     1.0         NaN   0.0  …      7        1180            0.0
1      7242     2.0         0.0   0.0  …      7        2170          400.0
2     10000     1.0         0.0   0.0  …      6         770            0.0
3      5000     1.0         0.0   0.0  …      7        1050          910.0
4      8080     1.0         0.0   0.0  …      8        1680            0.0

  yr_built  yr_renovated  zipcode      lat     long  sqft_living15  sqft_lot15
0     1955           0.0    98178  47.5112 -122.257           1340        5650
1     1951        1991.0    98125  47.7210 -122.319           1690        7639
2     1933           NaN    98028  47.7379 -122.233           2720        8062
3     1965           0.0    98136  47.5208 -122.393           1360        5000
4     1987           0.0    98074  47.6168 -122.045           1800        7503

[5 rows x 21 columns]
```

```
None
```

4 SCRUB

The data looks clean for the most part, but there are null values in the columns labeled 'waterfront', 'view' and 'yr_renovated' which will be addressed in this section. We also need to make sure to address the two columns that have been stored as object data types labeled 'date' and 'sqft_basement' in addition to checking for duplicated entries.

4.1 Checking for duplicates

[3]:
```
df[df['id'].duplicated(keep=False)]

# Duplicates in id have different dates, and can be considered as resold␣
↪properties.
```

[3]:
```
             id        date      price  bedrooms  bathrooms  sqft_living  \
93   6021501535   7/25/2014   430000.0         3       1.50         1580
94   6021501535  12/23/2014   700000.0         3       1.50         1580
313  4139480200   6/18/2014  1380000.0         4       3.25         4290
314  4139480200   12/9/2014  1400000.0         4       3.25         4290
```

```
324    7520000520   9/5/2014  232000.0         2       1.00         1240
…          …          …         …        …        …          …
20654  8564860270  3/30/2015  502000.0         4       2.50         2680
20763  6300000226  6/26/2014  240000.0         4       1.00         1200
20764  6300000226   5/4/2015  380000.0         4       1.00         1200
21564  7853420110  10/3/2014  594866.0         3       3.00         2780
21565  7853420110   5/4/2015  625000.0         3       3.00         2780

       sqft_lot  floors  waterfront  view  …  grade  sqft_above  \
93         5000     1.0         0.0   0.0  …      8        1290
94         5000     1.0         0.0   0.0  …      8        1290
313       12103     1.0         0.0   3.0  …     11        2690
314       12103     1.0         0.0   3.0  …     11        2690
324       12092     1.0         NaN   0.0  …      6         960
…           …     …           …   …  …    …          …
20654      5539     2.0         NaN   0.0  …      8        2680
20763      2171     1.5         0.0   0.0  …      7        1200
20764      2171     1.5         0.0   0.0  …      7        1200
21564      6000     2.0         0.0   0.0  …      9        2780
21565      6000     2.0         0.0   0.0  …      9        2780

       sqft_basement yr_built  yr_renovated  zipcode      lat     long  \
93             290.0     1939           0.0    98117  47.6870 -122.386
94             290.0     1939           0.0    98117  47.6870 -122.386
313           1600.0     1997           0.0    98006  47.5503 -122.102
314           1600.0     1997           0.0    98006  47.5503 -122.102
324            280.0     1922        1984.0    98146  47.4957 -122.352
…                …      …            …      …      …       …
20654            0.0     2013           0.0    98045  47.4759 -121.734
20763            0.0     1933           0.0    98133  47.7076 -122.342
20764            0.0     1933           0.0    98133  47.7076 -122.342
21564            0.0     2013           0.0    98065  47.5184 -121.886
21565            0.0     2013           NaN    98065  47.5184 -121.886

       sqft_living15  sqft_lot15
93              1570        4500
94              1570        4500
313             3860       11244
314             3860       11244
324             1820        7460
…                …          …
20654           2680        5992
20763           1130        1598
20764           1130        1598
21564           2850        6000
21565           2850        6000
```

```
[353 rows x 21 columns]
```

[4]:
```python
df[df.duplicated(keep=False)]

# There are no duplicated entries
```

[4]:
```
Empty DataFrame
Columns: [id, date, price, bedrooms, bathrooms, sqft_living, sqft_lot, floors,
waterfront, view, condition, grade, sqft_above, sqft_basement, yr_built,
yr_renovated, zipcode, lat, long, sqft_living15, sqft_lot15]
Index: []

[0 rows x 21 columns]
```

[5]:
```python
# Drop id and date columns since they are not controllable attributes
# that would affect the value of the property
df.drop(['id','date'], axis=1, inplace=True)
```

4.2 Checking null value counts

[6]:
```python
# Check number of NaN cells in dataframe
df.isna().sum()
```

[6]:
```
price               0
bedrooms            0
bathrooms           0
sqft_living         0
sqft_lot            0
floors              0
waterfront       2376
view               63
condition           0
grade               0
sqft_above          0
sqft_basement       0
yr_built            0
yr_renovated     3842
zipcode             0
lat                 0
long                0
sqft_living15       0
sqft_lot15          0
dtype: int64
```

[7]:
```python
# Check waterfront column's value counts
df['waterfront'].value_counts(dropna=False)
```

[7]:
```
0.0    19075
NaN     2376
1.0      146
Name: waterfront, dtype: int64
```

We will assume that homes with a missing value for 'waterfront' are not located on a waterfront

[8]:
```
# Replace Nan cells with 0.0
df['waterfront'].fillna(0.0, inplace=True)
```

[9]:
```
# Confirm that fillna method worked properly
df['waterfront'].value_counts(dropna=False)
```

[9]:
```
0.0    21451
1.0      146
Name: waterfront, dtype: int64
```

[10]:
```
#Check yr_renovated value counts
df['yr_renovated'].value_counts(dropna=False)
```

[10]:
```
0.0       17011
NaN        3842
2014.0       73
2003.0       31
2013.0       31
          …
1944.0        1
1948.0        1
1976.0        1
1934.0        1
1953.0        1
Name: yr_renovated, Length: 71, dtype: int64
```

Similar to the 'waterfront', we will assume that homes with a missing value for 'yr_renovated' have not undergone renovation.

[11]:
```
# Replace NaN cells with 0.0
df['yr_renovated'].fillna(0.0, inplace=True)
```

[12]:
```
# Confirm that fillna method worked properly
df['yr_renovated'].value_counts(dropna=False)
```

[12]:
```
0.0       20853
2014.0       73
2003.0       31
2013.0       31
2007.0       30
          …
```

```
1946.0        1
1959.0        1
1971.0        1
1951.0        1
1954.0        1
Name: yr_renovated, Length: 70, dtype: int64
```

Due to the ambiguous definition of the 'view' column, we will drop it to avoid including any variables in our regression model that we cannot explain.

[13]:
```
# Remove view due to ambiguous definition
df.drop('view', axis=1, inplace=True)
```

[14]:
```
#Confirm that all NaN cells have been addressed
df.isna().sum()
```

[14]:
```
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
condition        0
grade            0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
zipcode          0
lat              0
long             0
sqft_living15    0
sqft_lot15       0
dtype: int64
```

4.3 Converting Data Types

Great, no more null values to address. Now we need to check why 'sqft_basement' is being stored as an object data type. We will go ahead and remove the missing entries since the count is not large and convert the data type to float or int.

[15]:
```
# Check for non-numberical entries in sqft_basement
df['sqft_basement'].value_counts().sort_index()
```

[15]:
```
0.0       12826
10.0          2
100.0        42
```

```
1000.0      148
1008.0        1
           …
960.0        65
970.0        44
980.0        57
990.0        52
?           454
Name: sqft_basement, Length: 304, dtype: int64
```

[16]:
```
# Remove entries where sqft_basement is '?'
df = df[df['sqft_basement'] != '?']
```

[17]:
```
# Convert sqft_basement from object to float
df['sqft_basement'] = df['sqft_basement'].astype(float)
```

4.4 Feature Engineering

Because those properties that have not been renovated contain a value of 0.0 under their 'yr_renovated' column, this will skew the rest of the data where the other entries that have been renovated will contain a year number. We will engineer a binary feature that indicates whether or not the property has undergone any renovation in order to avoid this skew issue.

[18]:
```
# Define function to create column with value 1 if renovated, 0 if not␣
 ↪renovated.
def renov_bool(row):
    if row['yr_renovated'] > 0:
        val = 1
    else:
        val = 0
    return val
```

[19]:
```
# Apply function to create renovated column
df['renovated'] = df.apply(renov_bool, axis=1)
df.drop('yr_renovated', axis=1, inplace=True)
```

[20]:
```
# Verify that we have successfully removed null values and fixed data types
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21143 entries, 0 to 21596
Data columns (total 18 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   price          21143 non-null  float64
 1   bedrooms       21143 non-null  int64
 2   bathrooms      21143 non-null  float64
```

```
 3   sqft_living    21143 non-null  int64
 4   sqft_lot       21143 non-null  int64
 5   floors         21143 non-null  float64
 6   waterfront     21143 non-null  float64
 7   condition      21143 non-null  int64
 8   grade          21143 non-null  int64
 9   sqft_above     21143 non-null  int64
 10  sqft_basement  21143 non-null  float64
 11  yr_built       21143 non-null  int64
 12  zipcode        21143 non-null  int64
 13  lat            21143 non-null  float64
 14  long           21143 non-null  float64
 15  sqft_living15  21143 non-null  int64
 16  sqft_lot15     21143 non-null  int64
 17  renovated      21143 non-null  int64
dtypes: float64(7), int64(11)
memory usage: 3.1 MB
```

4.5 Checking for Correlation and Multicollinearity

We will move on to check for how correlated each column is with our target variable 'price' as well as check for multicollinearity

[21]:

```
# Create correlation matrix from dataframe
price_corr = df.corr()
price_corr.round(2)
```

[21]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	\
price	1.00	0.31	0.53	0.70	0.09	0.26	
bedrooms	0.31	1.00	0.51	0.58	0.03	0.18	
bathrooms	0.53	0.51	1.00	0.76	0.09	0.50	
sqft_living	0.70	0.58	0.76	1.00	0.17	0.35	
sqft_lot	0.09	0.03	0.09	0.17	1.00	-0.01	
floors	0.26	0.18	0.50	0.35	-0.01	1.00	
waterfront	0.27	0.00	0.06	0.11	0.02	0.02	
condition	0.04	0.03	-0.13	-0.06	-0.01	-0.26	
grade	0.67	0.36	0.67	0.76	0.11	0.46	
sqft_above	0.61	0.48	0.69	0.88	0.18	0.52	
sqft_basement	0.33	0.30	0.28	0.43	0.02	-0.25	
yr_built	0.05	0.16	0.51	0.32	0.05	0.49	
zipcode	-0.05	-0.15	-0.20	-0.20	-0.13	-0.06	
lat	0.31	-0.01	0.02	0.05	-0.09	0.05	
long	0.02	0.13	0.22	0.24	0.23	0.13	
sqft_living15	0.59	0.39	0.57	0.76	0.14	0.28	
sqft_lot15	0.08	0.03	0.09	0.18	0.72	-0.01	
renovated	0.12	0.02	0.05	0.05	0.00	0.00	

	waterfront	condition	grade	sqft_above	sqft_basement	\

price	0.27	0.04	0.67	0.61	0.33
bedrooms	0.00	0.03	0.36	0.48	0.30
bathrooms	0.06	-0.13	0.67	0.69	0.28
sqft_living	0.11	-0.06	0.76	0.88	0.43
sqft_lot	0.02	-0.01	0.11	0.18	0.02
floors	0.02	-0.26	0.46	0.52	-0.25
waterfront	1.00	0.02	0.08	0.07	0.08
condition	0.02	1.00	-0.15	-0.16	0.17
grade	0.08	-0.15	1.00	0.76	0.17
sqft_above	0.07	-0.16	0.76	1.00	-0.05
sqft_basement	0.08	0.17	0.17	-0.05	1.00
yr_built	-0.02	-0.36	0.45	0.43	-0.13
zipcode	0.03	0.00	-0.19	-0.26	0.08
lat	-0.01	-0.02	0.11	-0.00	0.11
long	-0.04	-0.11	0.20	0.34	-0.15
sqft_living15	0.09	-0.09	0.71	0.73	0.20
sqft_lot15	0.03	-0.00	0.12	0.20	0.02
renovated	0.07	-0.06	0.02	0.02	0.07

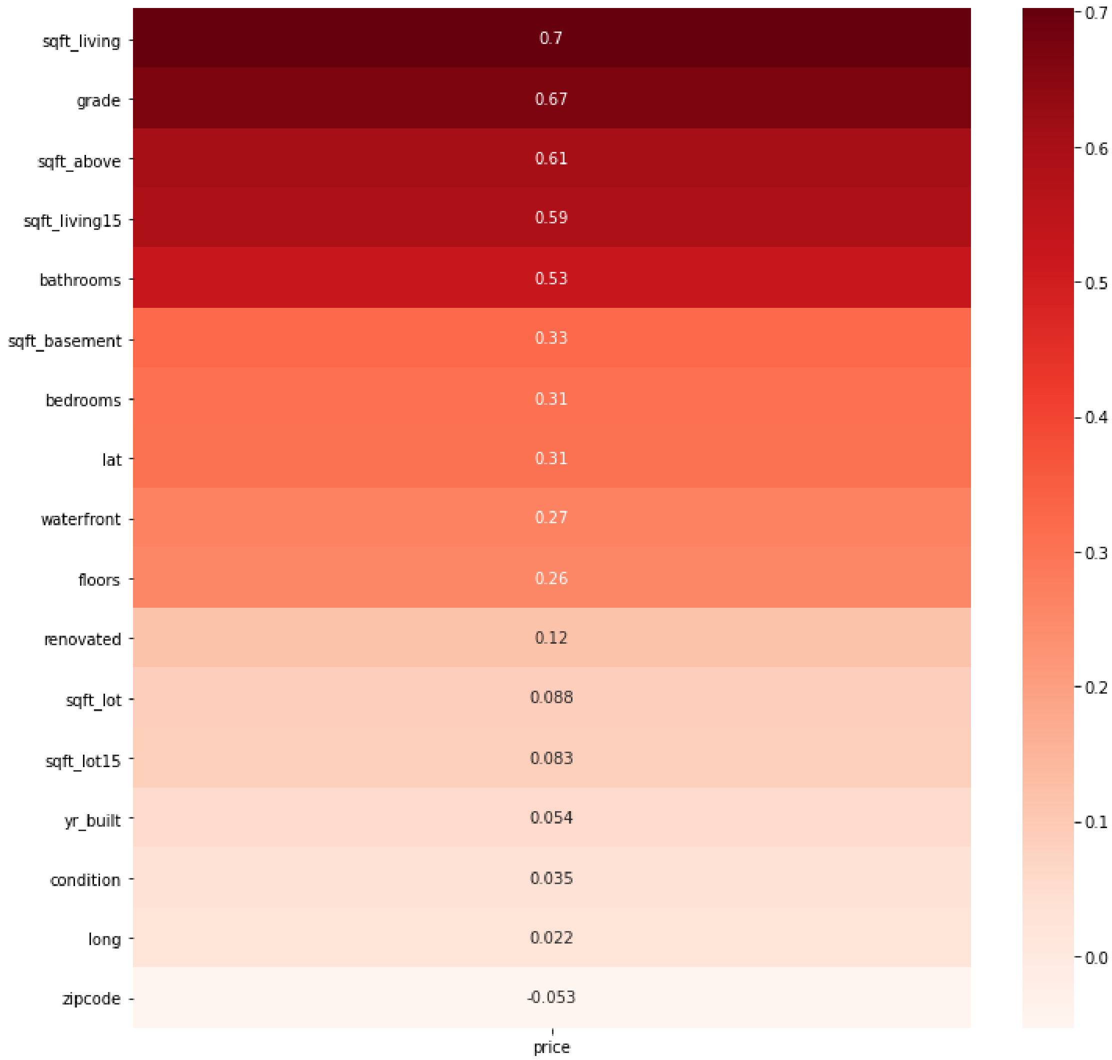
	yr_built	zipcode	lat	long	sqft_living15	sqft_lot15 \
price	0.05	-0.05	0.31	0.02	0.59	0.08
bedrooms	0.16	-0.15	-0.01	0.13	0.39	0.03
bathrooms	0.51	-0.20	0.02	0.22	0.57	0.09
sqft_living	0.32	-0.20	0.05	0.24	0.76	0.18
sqft_lot	0.05	-0.13	-0.09	0.23	0.14	0.72
floors	0.49	-0.06	0.05	0.13	0.28	-0.01
waterfront	-0.02	0.03	-0.01	-0.04	0.09	0.03
condition	-0.36	0.00	-0.02	-0.11	-0.09	-0.00
grade	0.45	-0.19	0.11	0.20	0.71	0.12
sqft_above	0.43	-0.26	-0.00	0.34	0.73	0.20
sqft_basement	-0.13	0.08	0.11	-0.15	0.20	0.02
yr_built	1.00	-0.35	-0.15	0.41	0.33	0.07
zipcode	-0.35	1.00	0.27	-0.56	-0.28	-0.15
lat	-0.15	0.27	1.00	-0.14	0.05	-0.08
long	0.41	-0.56	-0.14	1.00	0.33	0.26
sqft_living15	0.33	-0.28	0.05	0.33	1.00	0.18
sqft_lot15	0.07	-0.15	-0.08	0.26	0.18	1.00
renovated	-0.20	0.06	0.03	-0.06	0.00	0.00

	renovated
price	0.12
bedrooms	0.02
bathrooms	0.05
sqft_living	0.05
sqft_lot	0.00
floors	0.00
waterfront	0.07

```
condition          -0.06
grade               0.02
sqft_above          0.02
sqft_basement       0.07
yr_built           -0.20
zipcode             0.06
lat                 0.03
long               -0.06
sqft_living15       0.00
sqft_lot15          0.00
renovated           1.00
```

[22]:

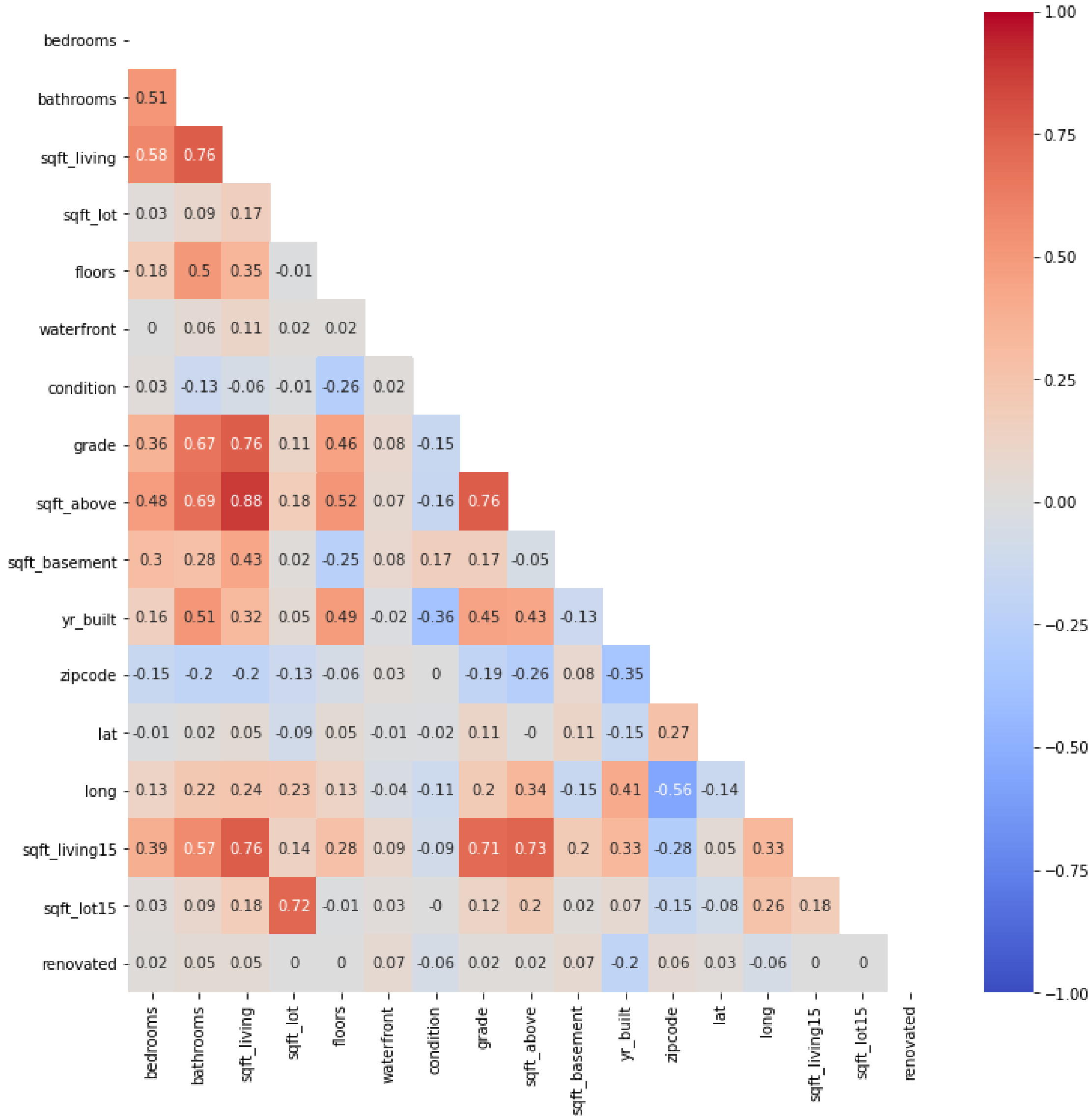
```python
# Correlation heatmap customization guide was utilized to create the following␣
↪visualizations:
# https://medium.com/@chrisshaw982/
↪seaborn-correlation-heatmaps-customized-10246f4f7f4b
fig, ax = plt.subplots(figsize=(12,12))
sns.heatmap(price_corr[['price']].drop('price').sort_values(by='price',␣
↪ascending=False), annot=True,
            ax=ax, cmap='Reds');
```



[23]:
```
# Drop price to only show correlation between independent variables
corr = df.drop('price', axis=1).corr().round(2)
```

[24]:
```
# Create mask for upper triangle of matrix
mask = np.zeros_like(corr)

mask[np.triu_indices_from(mask)] = True
```

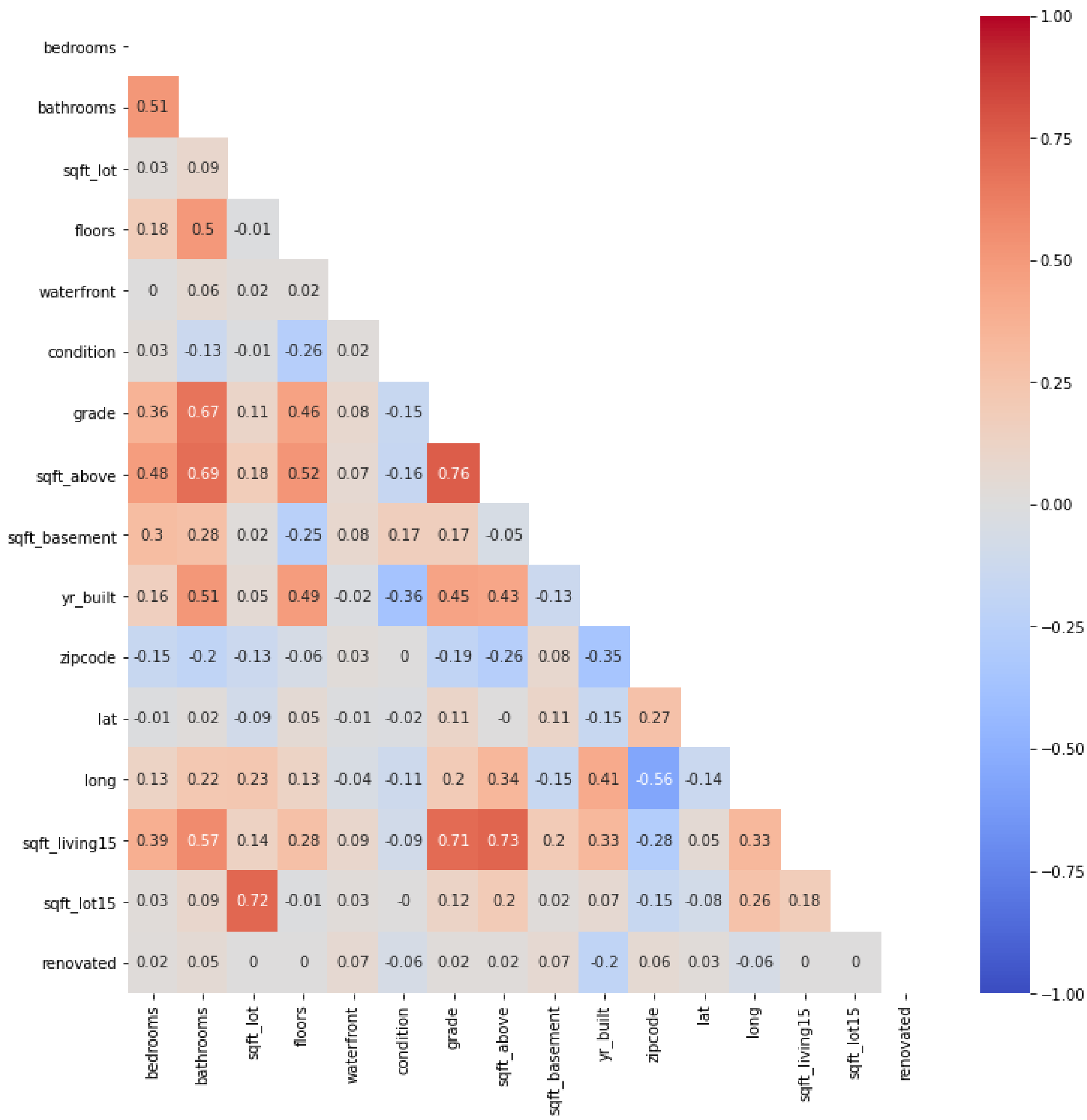
[25]:
```
#Create heatmap correlation matrix
fig, ax = plt.subplots(figsize=(12,12))
sns.heatmap(corr, annot=True, ax=ax, cmap='coolwarm', vmin=-1, vmax=1,\
            mask=mask);
```



From the correlation heatmap, we can see that other than ‘sqft_living’, we do not have any variables that are high enough to remove prior to running our baseline model. We will go ahead and remove ‘sqft_living’ to address the issue of multicollinearity in our dataset.

[26]:
```
# Remove sqft_living to get address multicollinearity
df.drop('sqft_living', axis=1, inplace=True)

corr = df.drop('price', axis=1).corr().round(2)

mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask)] = True
```

```
[27]: # Check heatmap correlation matrix after removing column
fig, ax = plt.subplots(figsize=(12,12))
sns.heatmap(corr, annot=True, ax=ax, cmap='coolwarm', vmin=-1, vmax=1,␣
 ↪mask=mask);
```



5 EXPLORE

In this section, we will explore the distributions as well as addressing the issue of outliers in each column. We will also be checking to see how much of a linear relationship each variable has with our target variable 'price'.

[28]:
```python
# Set theme and style for plots.
sns.set_theme('talk')
sns.set_style('darkgrid')
```

[29]:
```python
df.describe()
```

[29]:

	price	bedrooms	bathrooms	sqft_lot	floors
count	2.114300e+04	21143.000000	21143.000000	2.114300e+04	21143.000000
mean	5.405107e+05	3.372558	2.116079	1.508714e+04	1.493591
std	3.680751e+05	0.924917	0.768531	4.120920e+04	0.539249
min	7.800000e+04	1.000000	0.500000	5.200000e+02	1.000000
25%	3.220000e+05	3.000000	1.750000	5.043000e+03	1.000000
50%	4.500000e+05	3.000000	2.250000	7.620000e+03	1.500000
75%	6.450000e+05	4.000000	2.500000	1.069550e+04	2.000000
max	7.700000e+06	33.000000	8.000000	1.651359e+06	3.500000

	waterfront	condition	grade	sqft_above	sqft_basement
count	21143.000000	21143.000000	21143.000000	21143.000000	21143.000000
mean	0.006716	3.409923	7.658279	1789.069006	291.851724
std	0.081679	0.650498	1.174253	828.409769	442.498337
min	0.000000	1.000000	3.000000	370.000000	0.000000
25%	0.000000	3.000000	7.000000	1200.000000	0.000000
50%	0.000000	3.000000	7.000000	1560.000000	0.000000
75%	0.000000	4.000000	8.000000	2210.000000	560.000000
max	1.000000	5.000000	13.000000	9410.000000	4820.000000

	yr_built	zipcode	lat	long	sqft_living15
count	21143.000000	21143.000000	21143.000000	21143.000000	21143.00000
mean	1971.023223	98077.868893	47.560274	-122.213876	1987.27139
std	29.321938	53.535756	0.138591	0.140597	685.67034
min	1900.000000	98001.000000	47.155900	-122.519000	399.00000
25%	1952.000000	98033.000000	47.471250	-122.328000	1490.00000
50%	1975.000000	98065.000000	47.572000	-122.230000	1840.00000
75%	1997.000000	98117.000000	47.678200	-122.125000	2360.00000
max	2015.000000	98199.000000	47.777600	-121.315000	6210.00000

	sqft_lot15	renovated
count	21143.000000	21143.000000
mean	12738.941967	0.034196
std	27169.273663	0.181736
min	651.000000	0.000000
25%	5100.000000	0.000000
50%	7626.000000	0.000000
75%	10087.000000	0.000000
max	871200.000000	1.000000

5.1 Checking for Normality, Outliers, and Linearity

There appear to be some outliers, as in the case of bedrooms where the max number is 33. Although this might be an error in data collection, we will leave the outliers be for now to see how they affect the skew of our data and how our baseline model turns out with what has been provided.

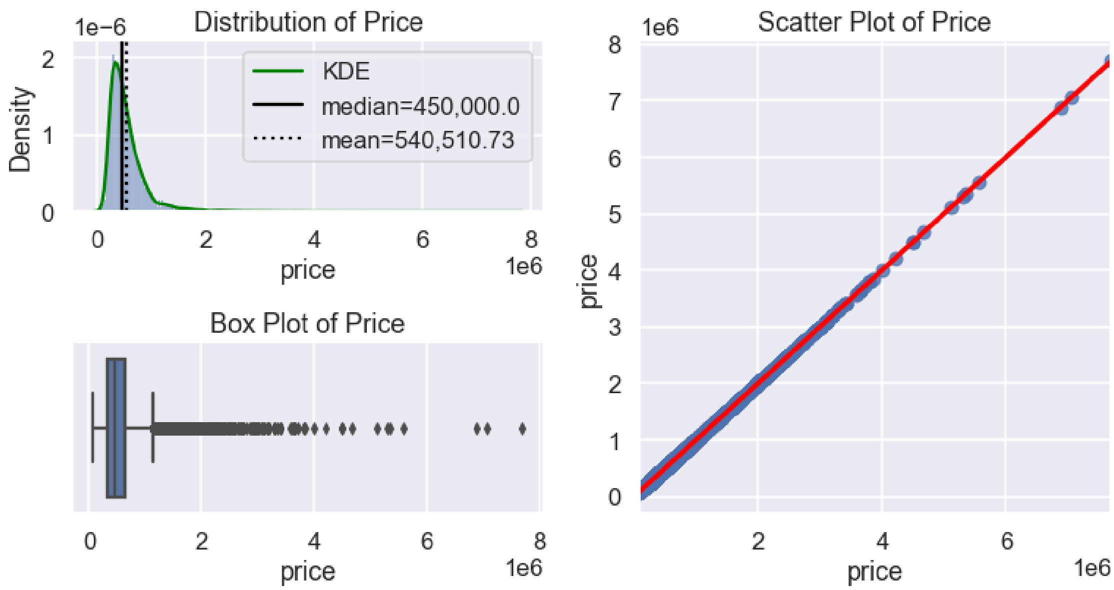
We will proceed to visualize how our data is distributed as well as the linearity of each variable against the price variable.

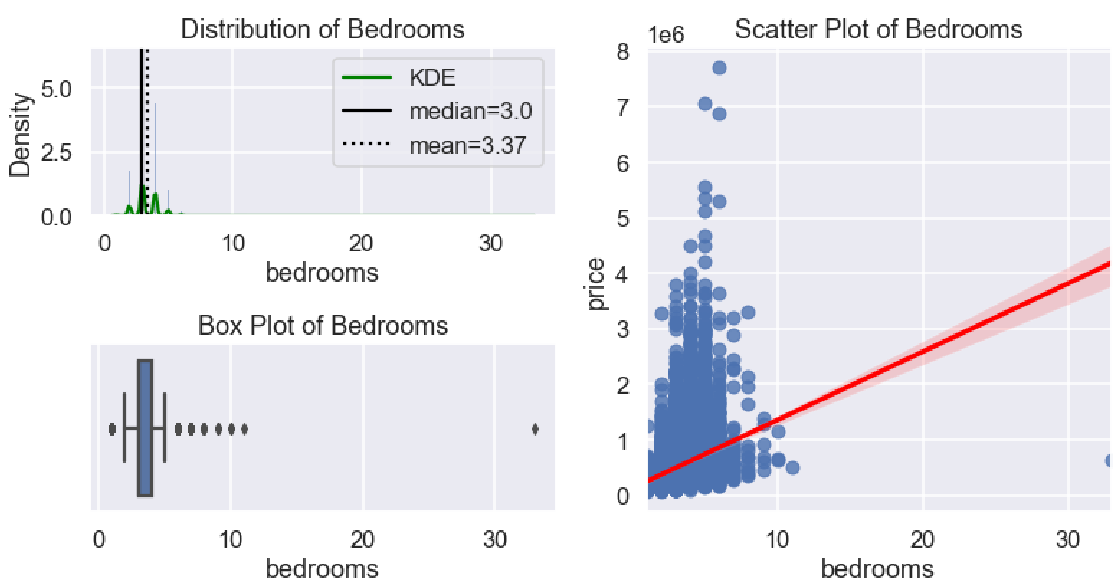
[30]:
```
# Create function to plot histogram and boxplot to indicate normality and␣
↪outliers
# and scatterplot to show linearity with the target variable
def plot_distribution_linearity(df, col=None, verbose=False,boxplot=True):
    """This function was written by James Irving during study group.
    Original function has been modified to include regression plot to
    illustrate linear relationship with 'price' column.

    Plots a histogram + KDE and a boxplot of the column.
    Also prints statistics for skew, kurtosis, and normaltest.

    Args:
        df_ (DataFrame): DataFrame containing column to plot
        col (str): Name of the column to plot.
        verbose (bool, optional): If true show figure and print stats. Defaults␣
↪to True.
        boxplot (bool, optional): If true, return subplots with boxplot.␣
↪Defaults to True.

    Returns:
        fig : Matplotlib Figure
        ax : Matplotlib Axis
    """

    # df = df_.copy()

    if col is None:
        data = df.copy()
        name = data.name
    else:
        data = df[col].copy()
        name = col

    ## Calc mean and mean skew and curtosis
    median = data.median().round(2)
    mean = data.mean().round(2)
    skew_val = round(stats.skew(data, bias=False),2)
    kurt_val = round(stats.kurtosis(data,bias=False),2)
```

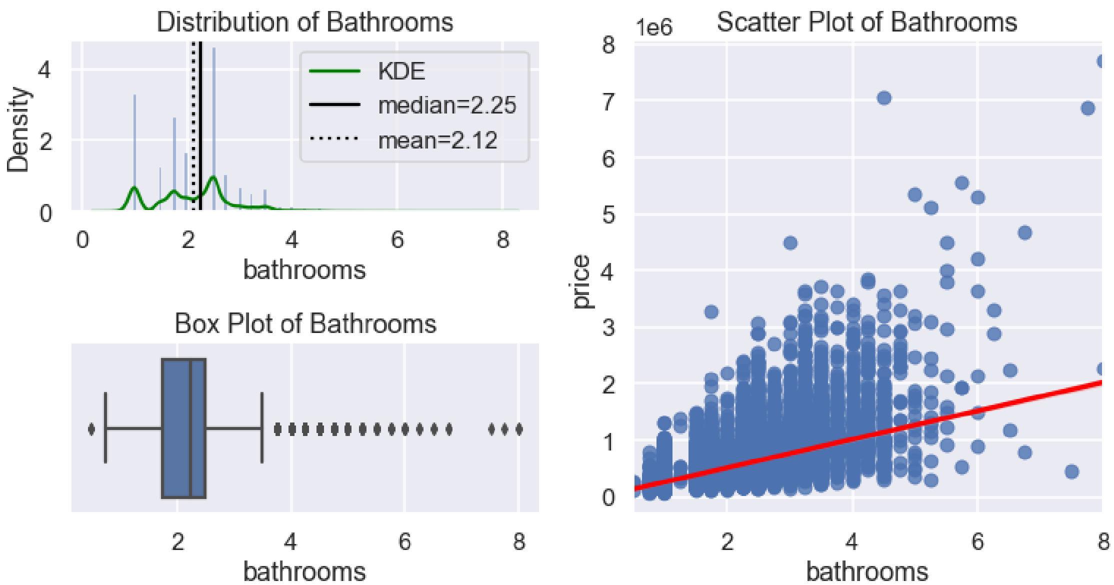
```python
## Plot distribution
fig = plt.figure(figsize=(11, 6))
gs = GridSpec(nrows=2, ncols=2)

ax0 = fig.add_subplot(gs[0, 0])
ax1 = fig.add_subplot(gs[1, 0])
ax2 = fig.add_subplot(gs[:, 1])

sns.histplot(data,alpha=0.5,stat='density',ax=ax0)
sns.kdeplot(data,color='green',label='KDE',ax=ax0)
ax0.set(ylabel='Density',title=name.title())
ax0.set_title(F"Distribution of {name.title()}")
ax0.axvline(median,label=f'median={median:,}',color='black')
ax0.axvline(mean,label=f'mean={mean:,}',color='black',ls=':')
ax0.legend()

## Plot Boxplot
sns.boxplot(data,x=col,ax=ax1)
ax1.set_title(F"Box Plot of {name.title()}")

# Plot Scatterplot to illustrate linearity
sns.regplot(data=df, x=col, y='price', line_kws={"color": "red"}, ax=ax2)
ax2.set_title(F"Scatter Plot of {name.title()}")

## Tweak Layout & Display
fig.tight_layout()

## Delete boxplot if unwanted
if boxplot == False:
    fig.delaxes(ax[1])

if verbose:
    plt.show()

    print('[i] Distribution Stats:')
    print(f"\tSkew = {skew_val}")
    print(f"\tKurtosis = {kurt_val}")
    print(f"\tN = {len(data):,}")


    ## Test for normality
    result = stats.normaltest(data)
    print('\n',result)
    if result[1]<.05:
        print('\t- p<.05: The distribution is NOT normally distributed.')
    elif result[1] >=.05:
```

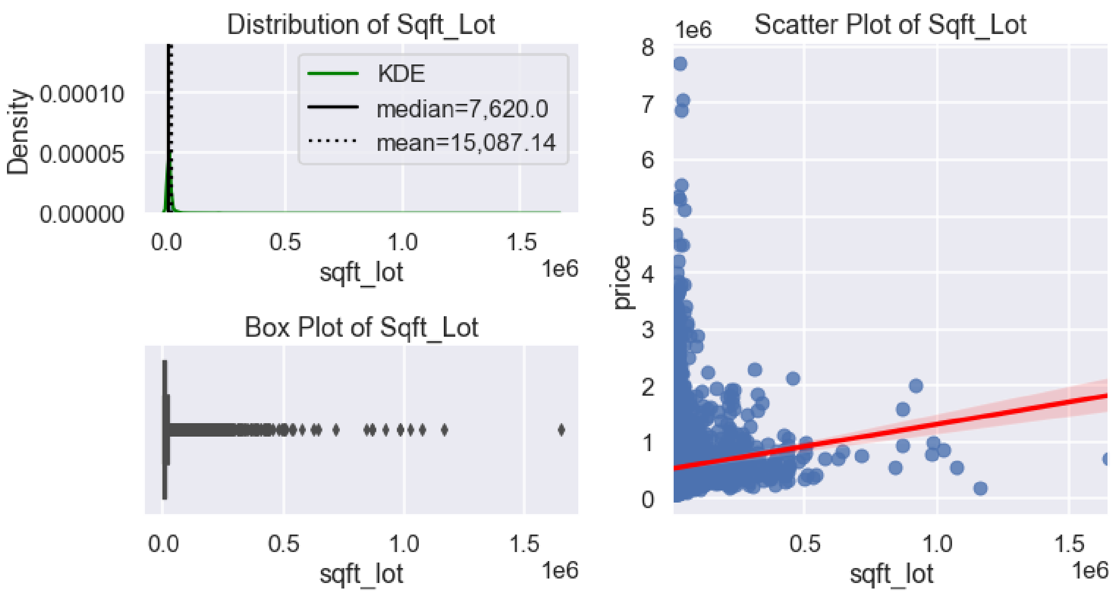
```
            print('\t- p>=.05: The distribution IS normally distributed')

    return fig, ax
```

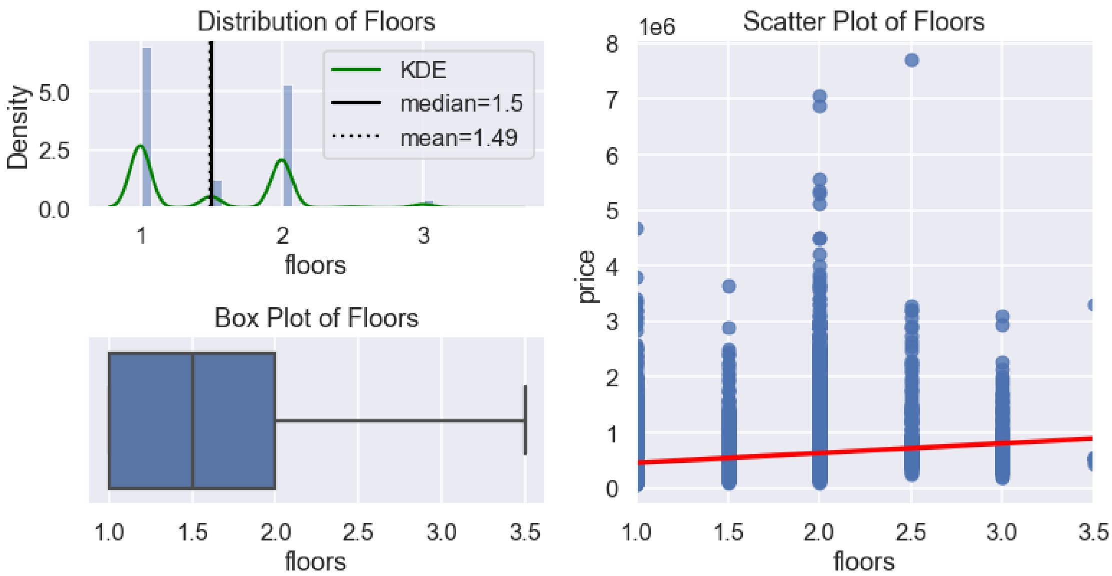
[31]:
```
# Create plot for all columns
for col in df:
    plot_distribution_linearity(df=df, col=col);
```

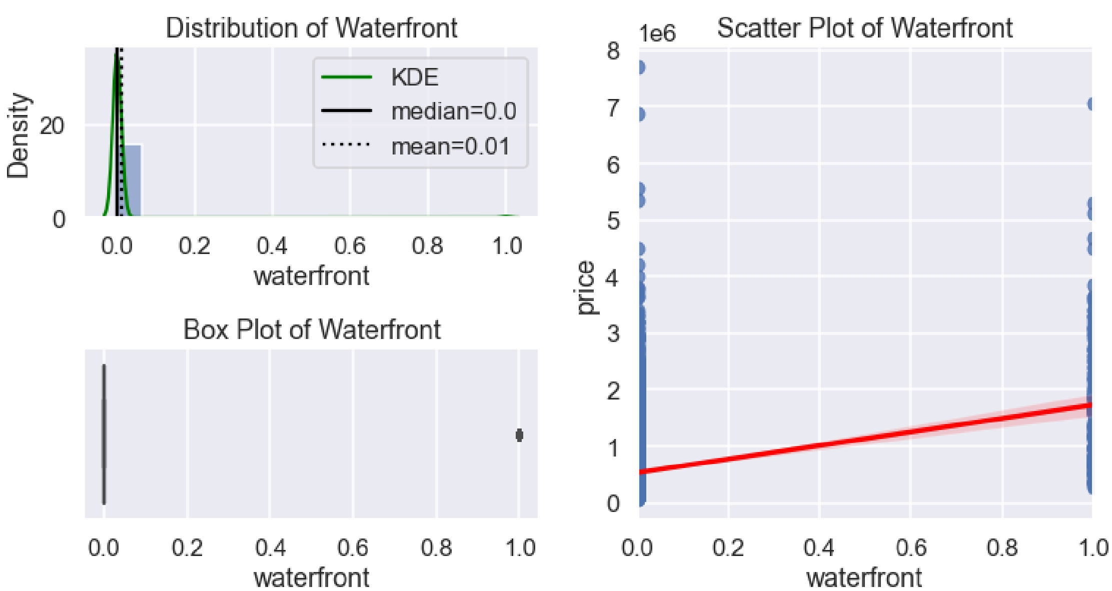


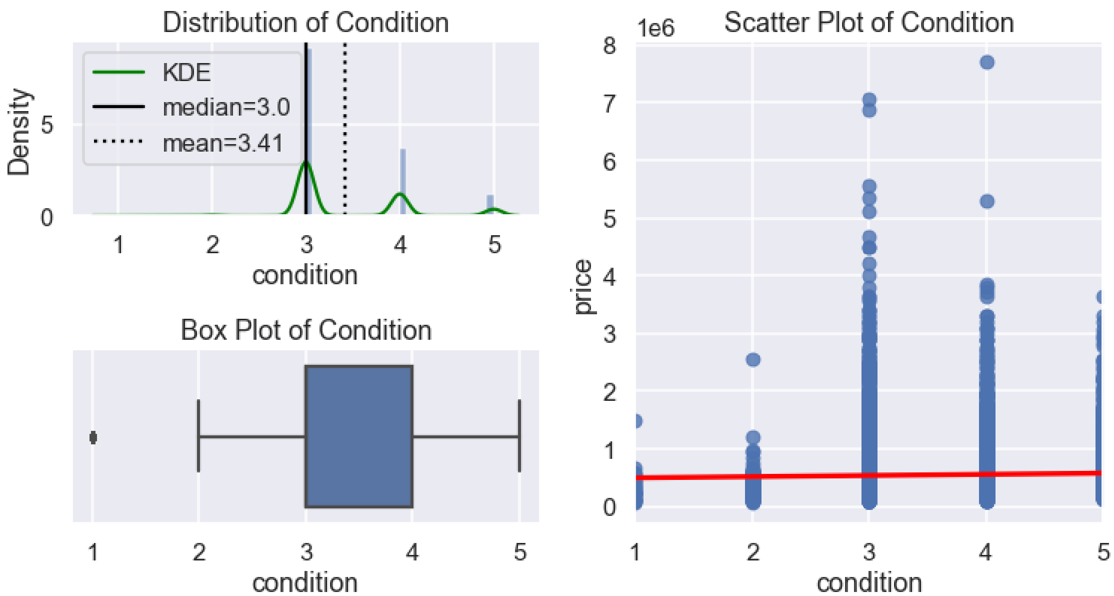


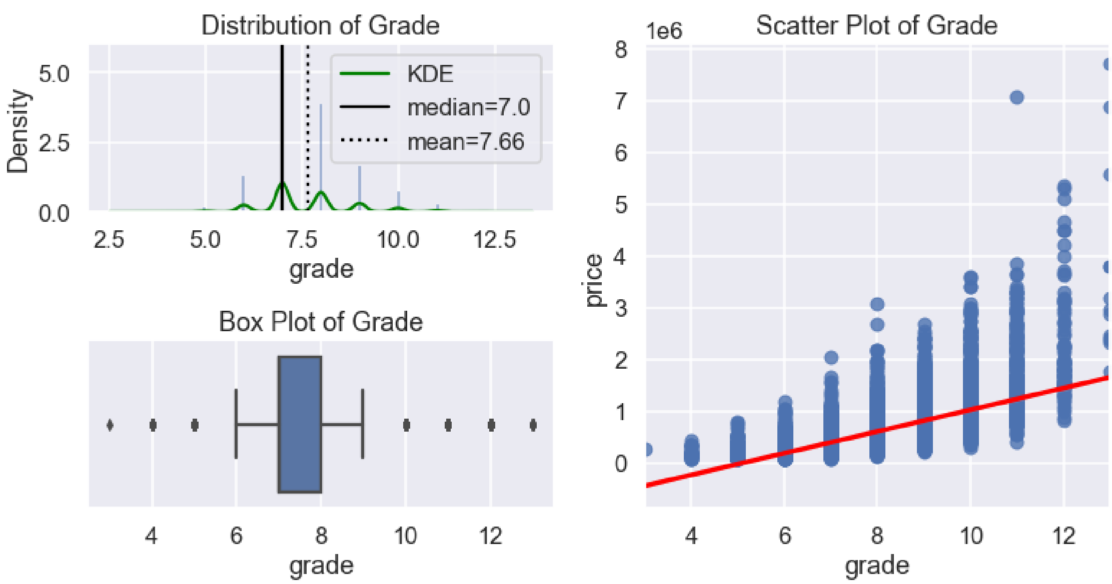

Distribution of Bathrooms
Density
KDE
median=2.25
mean=2.12
bathrooms
Box Plot of Bathrooms
bathrooms
Scatter Plot of Bathrooms
1e6
price
bathrooms

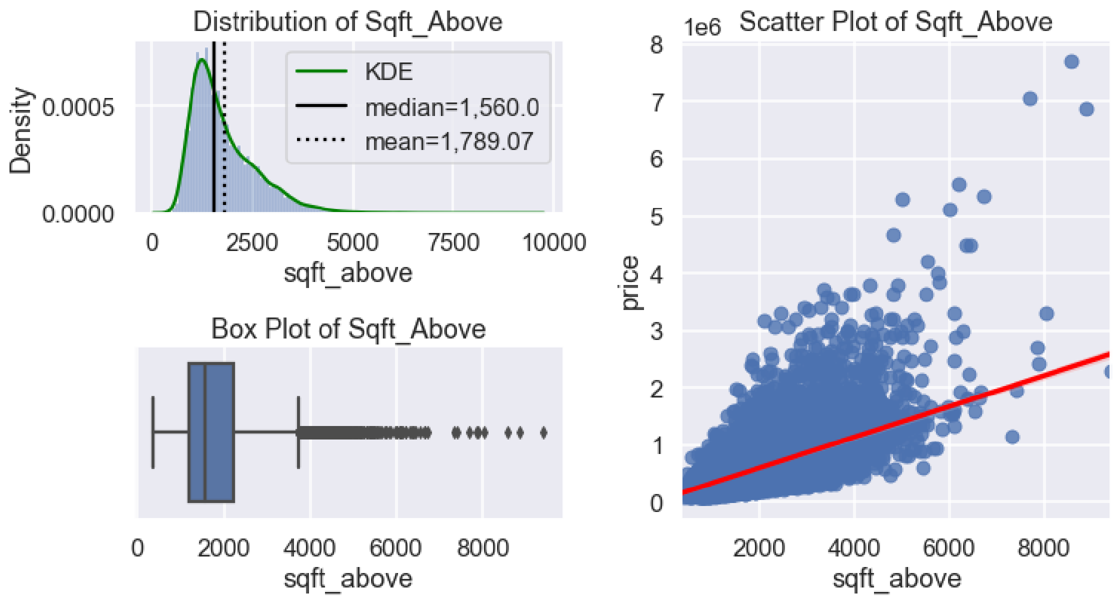

Distribution of Sqft_Lot
Density
KDE
median=7,620.0
mean=15,087.14
sqft_lot
1e6
Box Plot of Sqft_Lot
sqft_lot
1e6
Scatter Plot of Sqft_Lot
1e6
price
sqft_lot
1e6

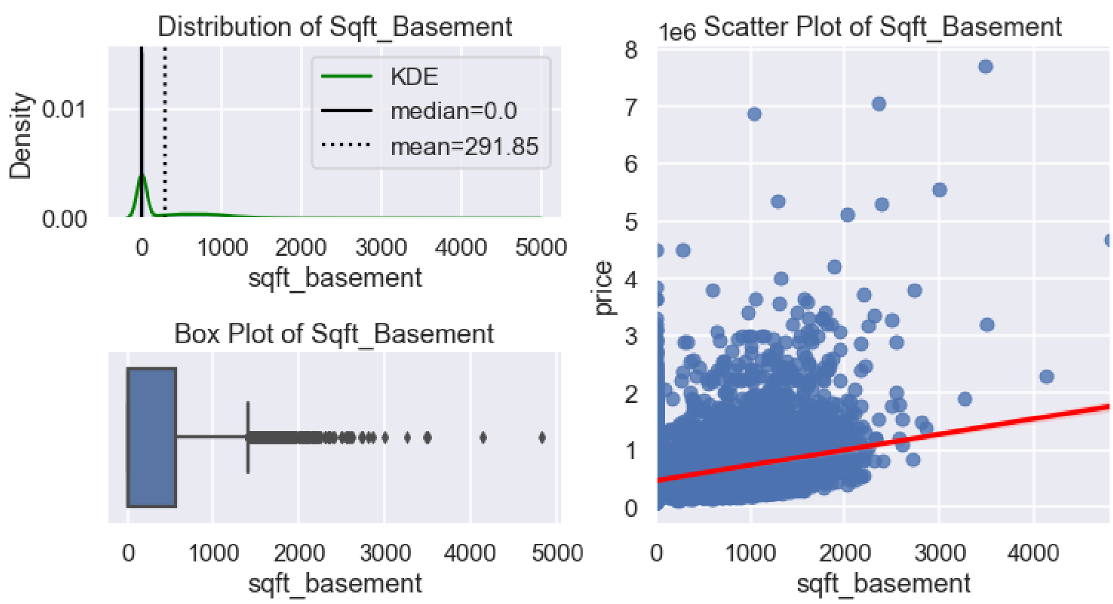

Distribution of Floors
Density
KDE
median=1.5
mean=1.49
floors
Box Plot of Floors
floors
Scatter Plot of Floors
1e6
price
floors

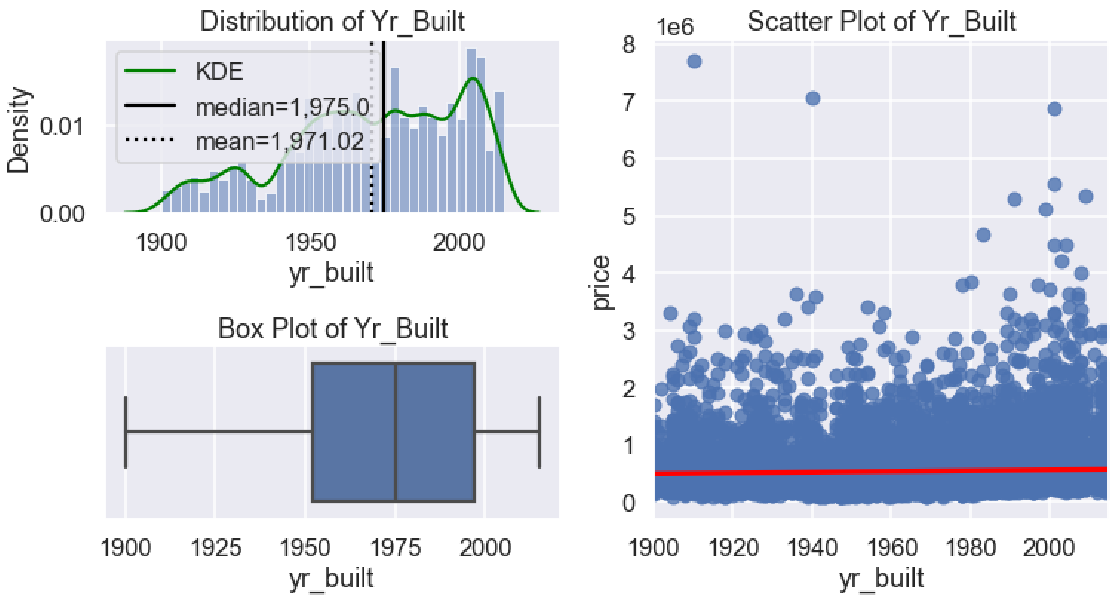

Distribution of Waterfront
Density
KDE
median=0.0
mean=0.01
waterfront
Box Plot of Waterfront
waterfront
Scatter Plot of Waterfront
1e6
price
waterfront

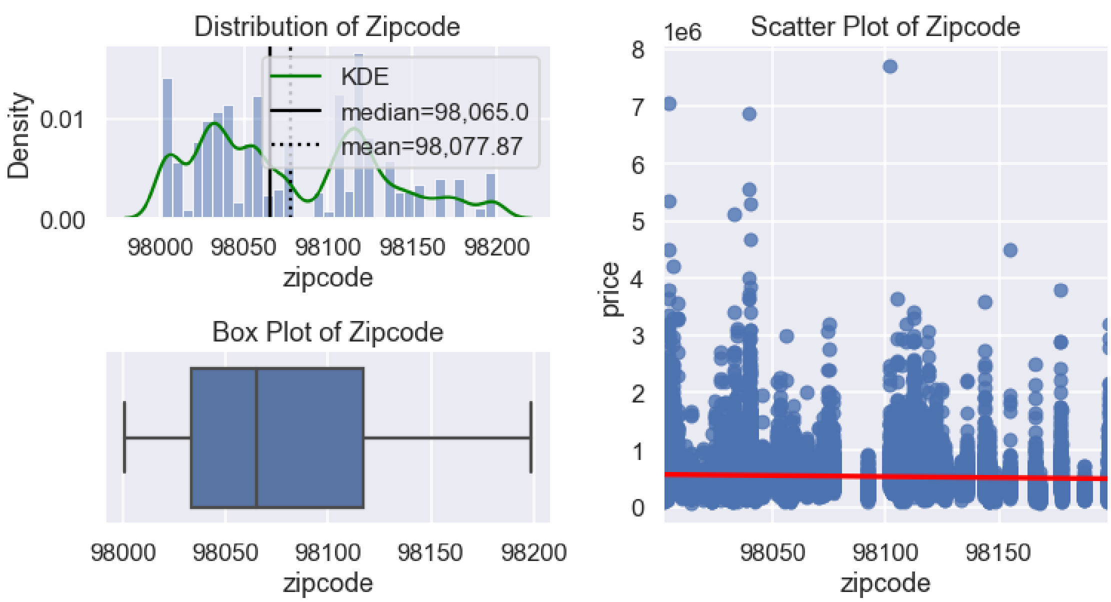

Distribution of Condition
KDE
median=3.0
mean=3.41
Density
condition
Box Plot of Condition
condition
Scatter Plot of Condition
1e6
price
condition

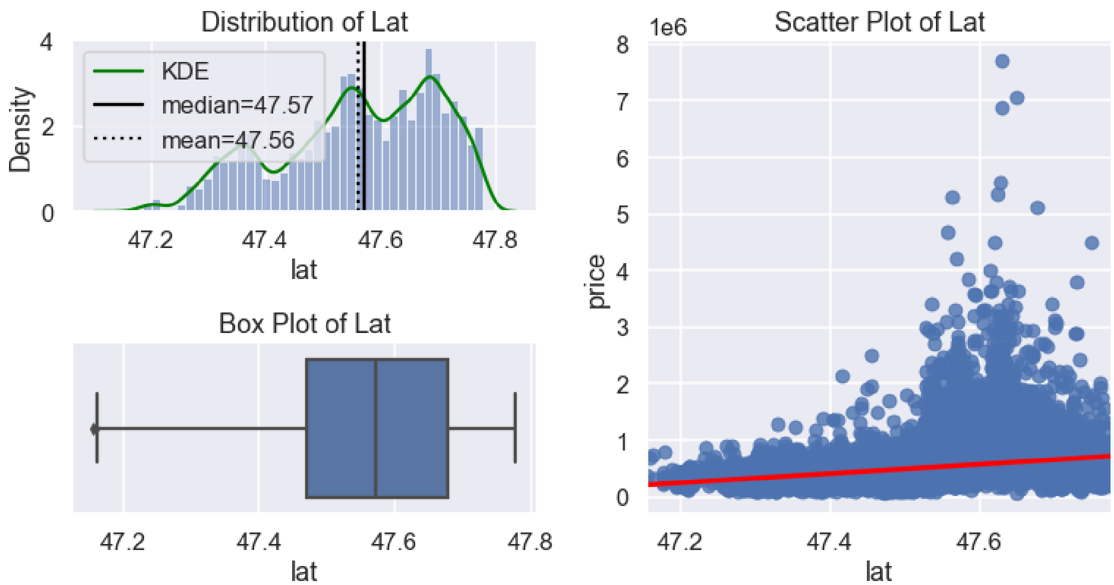

Distribution of Grade
KDE
median=7.0
mean=7.66
Density
grade
Box Plot of Grade
grade
Scatter Plot of Grade
1e6
price
grade

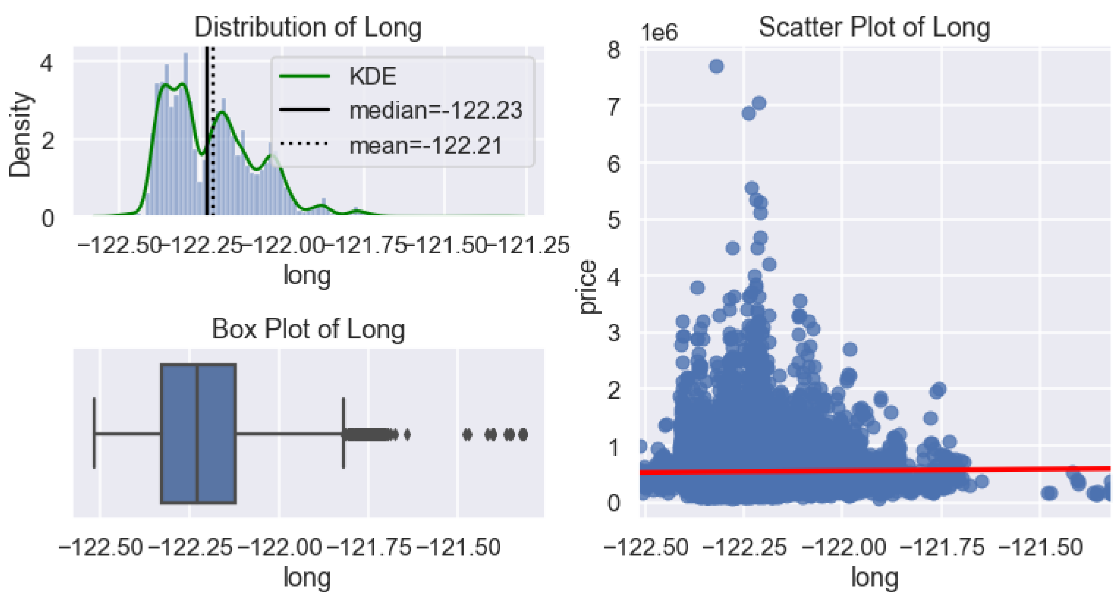

Distribution of Sqft_Above
Density
KDE
median=1,560.0
mean=1,789.07
sqft_above
Box Plot of Sqft_Above
sqft_above
1e6 Scatter Plot of Sqft_Above
price
sqft_above

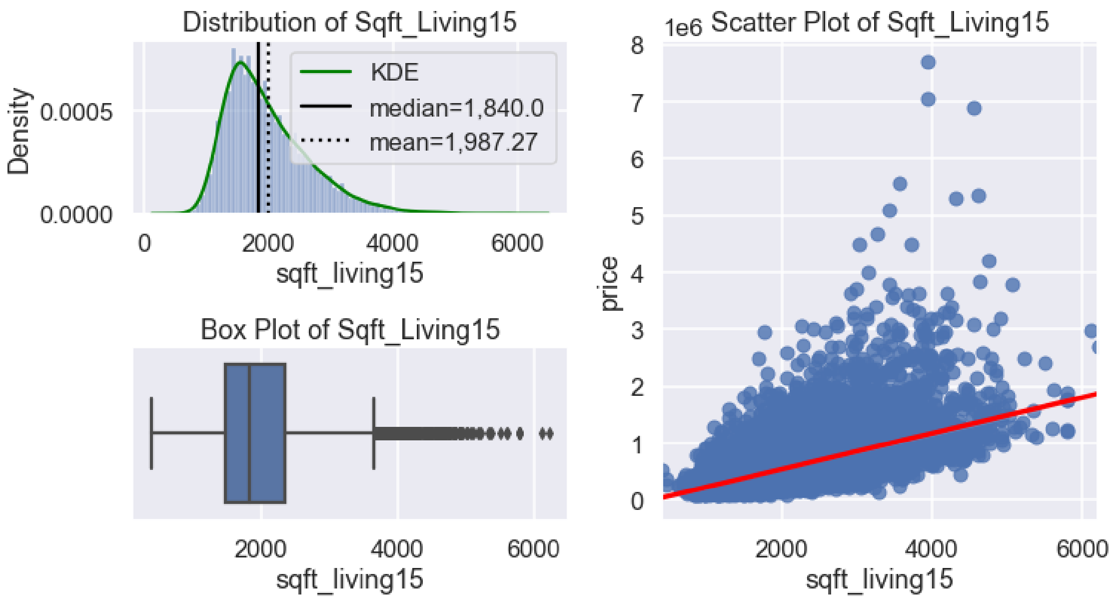

Distribution of Sqft_Basement
Density
KDE
median=0.0
mean=291.85
sqft_basement
Box Plot of Sqft_Basement
sqft_basement
1e6 Scatter Plot of Sqft_Basement
price
sqft_basement

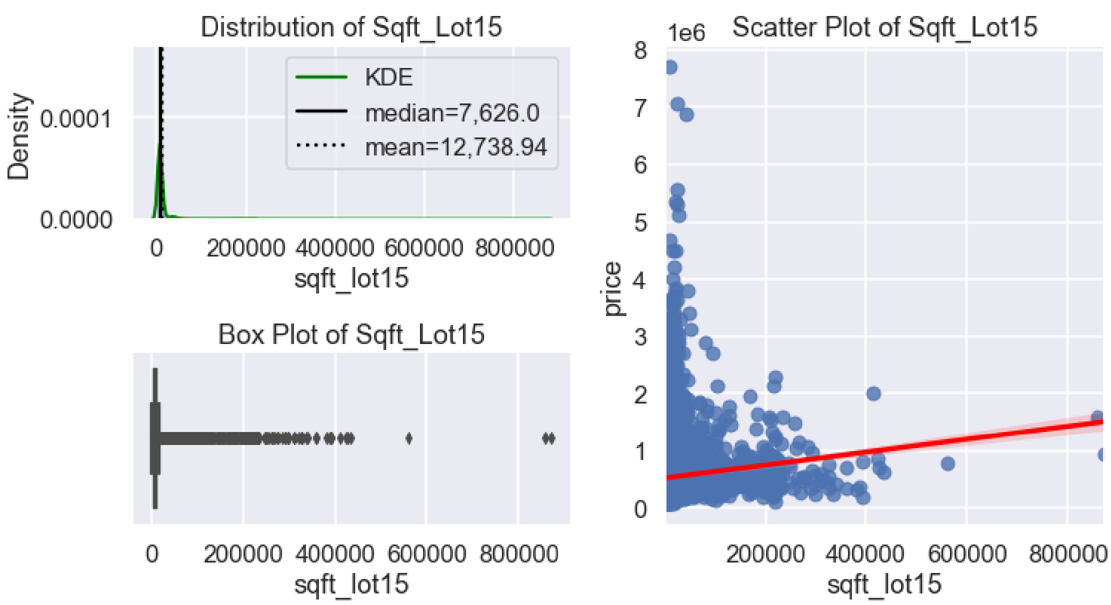

Distribution of Yr_Built
KDE
median=1,975.0
mean=1,971.02
Density
0.01
0.00
1900
1950
2000
yr_built
Box Plot of Yr_Built
1900
1925
1950
1975
2000
yr_built
Scatter Plot of Yr_Built
1e6
price
8
7
6
5
4
3
2
1
0
1900
1920
1940
1960
1980
2000
yr_built

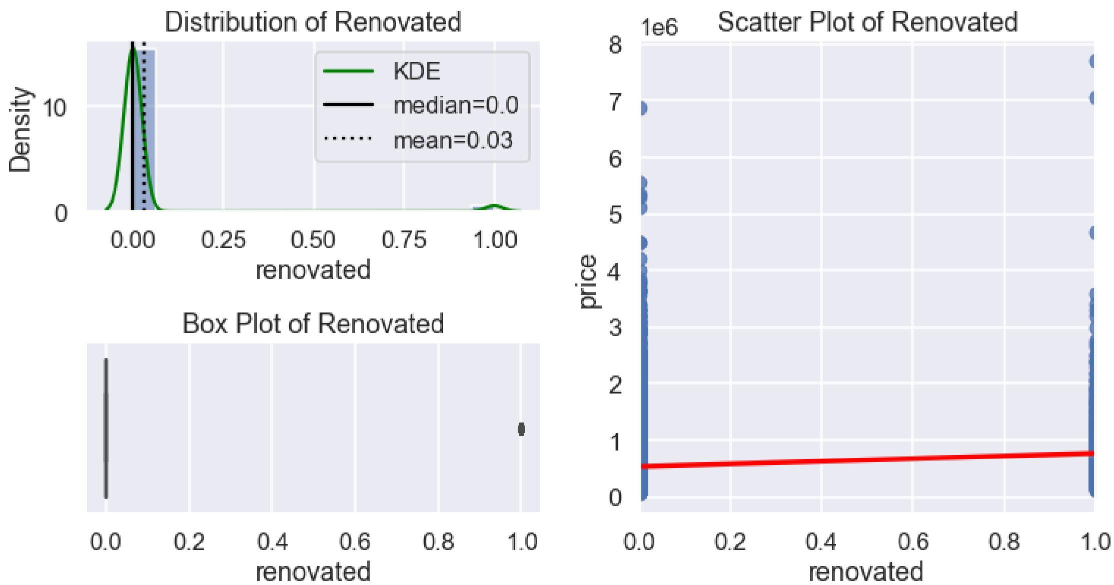

Distribution of Zipcode
KDE
median=98,065.0
mean=98,077.87
Density
0.01
0.00
98000
98050
98100
98150
98200
zipcode
Box Plot of Zipcode
98000
98050
98100
98150
98200
zipcode
Scatter Plot of Zipcode
1e6
price
8
7
6
5
4
3
2
1
0
98050
98100
98150
zipcode


Distribution of Lat
KDE
median=47.57
mean=47.56
Density
lat
Box Plot of Lat
lat
Scatter Plot of Lat
1e6
price
lat


Distribution of Long
KDE
median=-122.23
mean=-122.21
Density
long
Box Plot of Long
long
Scatter Plot of Long
1e6
price
long


Distribution of Sqft_Living15
Density
KDE
median=1,840.0
mean=1,987.27
sqft_living15
Box Plot of Sqft_Living15
sqft_living15
1e6
Scatter Plot of Sqft_Living15
price
sqft_living15


Distribution of Sqft_Lot15
Density
KDE
median=7,626.0
mean=12,738.94
sqft_lot15
Box Plot of Sqft_Lot15
sqft_lot15
1e6
Scatter Plot of Sqft_Lot15
price
sqft_lot15



[32]:
```
# Remove columns where there is weak linear relationship with price
df.drop(['condition', 'yr_built', 'renovated', 'sqft_lot15'], axis=1,␣
↪inplace=True)
```

5.2 One Hot Encoding

We can see that there are some categorical variables in our dataset, but other than the 'zipcode' column, the other variables are ordinal.

We will proceed to use One Hot Encoding prior to running our multiple regression model including the zipcode data.

[33]:
```
# One Hot Encode zipcodes column
encoder = OneHotEncoder(drop='first',sparse=False)
encoder.fit(df[['zipcode']])

ohe_vars = encoder.transform(df[['zipcode']])
ohe_vars

encoder.get_feature_names(['zipcode'])

df_ohe = pd.DataFrame(ohe_vars,columns=encoder.get_feature_names(['zipcode']),\
                      index=df.index)
df_ohe
```

[33]:
```
      zipcode_98002  zipcode_98003  zipcode_98004  zipcode_98005  \
0               0.0            0.0            0.0            0.0
1               0.0            0.0            0.0            0.0
```

```
2             0.0           0.0           0.0           0.0
3             0.0           0.0           0.0           0.0
4             0.0           0.0           0.0           0.0
…             …             …             …             …
21592         0.0           0.0           0.0           0.0
21593         0.0           0.0           0.0           0.0
21594         0.0           0.0           0.0           0.0
21595         0.0           0.0           0.0           0.0
21596         0.0           0.0           0.0           0.0

       zipcode_98006  zipcode_98007  zipcode_98008  zipcode_98010  \
0                0.0            0.0            0.0            0.0
1                0.0            0.0            0.0            0.0
2                0.0            0.0            0.0            0.0
3                0.0            0.0            0.0            0.0
4                0.0            0.0            0.0            0.0
…                …              …              …              …
21592            0.0            0.0            0.0            0.0
21593            0.0            0.0            0.0            0.0
21594            0.0            0.0            0.0            0.0
21595            0.0            0.0            0.0            0.0
21596            0.0            0.0            0.0            0.0

       zipcode_98011  zipcode_98014  …  zipcode_98146  zipcode_98148  \
0                0.0            0.0  …            0.0            0.0
1                0.0            0.0  …            0.0            0.0
2                0.0            0.0  …            0.0            0.0
3                0.0            0.0  …            0.0            0.0
4                0.0            0.0  …            0.0            0.0
…                …              …  …              …              …
21592            0.0            0.0  …            0.0            0.0
21593            0.0            0.0  …            1.0            0.0
21594            0.0            0.0  …            0.0            0.0
21595            0.0            0.0  …            0.0            0.0
21596            0.0            0.0  …            0.0            0.0

       zipcode_98155  zipcode_98166  zipcode_98168  zipcode_98177  \
0                0.0            0.0            0.0            0.0
1                0.0            0.0            0.0            0.0
2                0.0            0.0            0.0            0.0
3                0.0            0.0            0.0            0.0
4                0.0            0.0            0.0            0.0
…                …              …              …              …
21592            0.0            0.0            0.0            0.0
21593            0.0            0.0            0.0            0.0
21594            0.0            0.0            0.0            0.0
21595            0.0            0.0            0.0            0.0
```

```
21596            0.0            0.0            0.0            0.0

       zipcode_98178  zipcode_98188  zipcode_98198  zipcode_98199
0                1.0            0.0            0.0            0.0
1                0.0            0.0            0.0            0.0
2                0.0            0.0            0.0            0.0
3                0.0            0.0            0.0            0.0
4                0.0            0.0            0.0            0.0
…                …              …              …              …
21592            0.0            0.0            0.0            0.0
21593            0.0            0.0            0.0            0.0
21594            0.0            0.0            0.0            0.0
21595            0.0            0.0            0.0            0.0
21596            0.0            0.0            0.0            0.0

[21143 rows x 69 columns]
```

[34]:

```python
# Join One Hot Encoded dataframe with original dataframe and drop
# original zipcodes column
df_model = pd.concat([df.drop('zipcode',axis=1),df_ohe],axis=1)
df_model
```

[34]:

```
          price  bedrooms  bathrooms  sqft_lot  floors  waterfront  grade  \
0      221900.0         3       1.00      5650     1.0         0.0      7
1      538000.0         3       2.25      7242     2.0         0.0      7
2      180000.0         2       1.00     10000     1.0         0.0      6
3      604000.0         4       3.00      5000     1.0         0.0      7
4      510000.0         3       2.00      8080     1.0         0.0      8
…           …       …         …         …       …          …      …
21592  360000.0         3       2.50      1131     3.0         0.0      8
21593  400000.0         4       2.50      5813     2.0         0.0      8
21594  402101.0         2       0.75      1350     2.0         0.0      7
21595  400000.0         3       2.50      2388     2.0         0.0      8
21596  325000.0         2       0.75      1076     2.0         0.0      7

       sqft_above  sqft_basement      lat  …  zipcode_98146  zipcode_98148  \
0            1180            0.0  47.5112  …            0.0            0.0
1            2170          400.0  47.7210  …            0.0            0.0
2             770            0.0  47.7379  …            0.0            0.0
3            1050          910.0  47.5208  …            0.0            0.0
4            1680            0.0  47.6168  …            0.0            0.0
…             …             …      …     …          …              …
21592        1530            0.0  47.6993  …            0.0            0.0
21593        2310            0.0  47.5107  …            1.0            0.0
21594        1020            0.0  47.5944  …            0.0            0.0
21595        1600            0.0  47.5345  …            0.0            0.0
21596        1020            0.0  47.5941  …            0.0            0.0
```

```
       zipcode_98155  zipcode_98166  zipcode_98168  zipcode_98177  \
0                0.0            0.0            0.0            0.0
1                0.0            0.0            0.0            0.0
2                0.0            0.0            0.0            0.0
3                0.0            0.0            0.0            0.0
4                0.0            0.0            0.0            0.0
…                …             …             …             …
21592            0.0            0.0            0.0            0.0
21593            0.0            0.0            0.0            0.0
21594            0.0            0.0            0.0            0.0
21595            0.0            0.0            0.0            0.0
21596            0.0            0.0            0.0            0.0

       zipcode_98178  zipcode_98188  zipcode_98198  zipcode_98199
0                1.0            0.0            0.0            0.0
1                0.0            0.0            0.0            0.0
2                0.0            0.0            0.0            0.0
3                0.0            0.0            0.0            0.0
4                0.0            0.0            0.0            0.0
…                …             …             …             …
21592            0.0            0.0            0.0            0.0
21593            0.0            0.0            0.0            0.0
21594            0.0            0.0            0.0            0.0
21595            0.0            0.0            0.0            0.0
21596            0.0            0.0            0.0            0.0

[21143 rows x 81 columns]
```

6 MODEL

Finally, we have prepared our data enough to be able to run an initial iteration of our multiple regression model! As we create each model, we will include a QQ plot to address the normality of residuals as well as plotting price vs residuals in order to check for homoscedasticity of residuals.

6.1 Creating a Baseline Model

[35]:
```
# Create function to simultaneously run model and plot for normality
# and homoscedasticity of residuals.
def model_combined(df):
    ## Create a string representing the right side of the ~ in our formula
    features = ' + '.join(df.drop('price',axis=1).columns)

    ## Create the final formula and create the model
    f  = "price~"+features
```

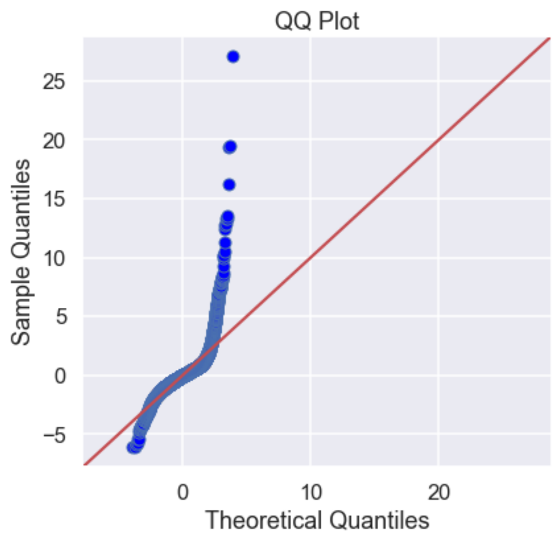
```
    # Model regression
    model = smf.ols(f, df).fit()
    display(model.summary())

    # Create QQ plot
    fig, ax = plt.subplots(ncols=2,figsize=(14,6))
    sm.graphics.qqplot(model.resid,dist=stats.norm,fit=True,line='45',\
                       ax=ax[0])
    ax[0].set_title('QQ Plot')

    # Create homoscedasticity plot
    resids = model.resid
    sns.scatterplot(x=model.predict(df.drop('price',axis=1), transform=True),\
                    y=model.resid, ax=ax[1])
    ax[1].axhline(0, color='r')
    ax[1].set_title('Homoscedasticity of Residuals')
    ax[1].set_xlabel('Predicted Price')
    ax[1].set_ylabel('Residuals')


    return model, fig, ax
```

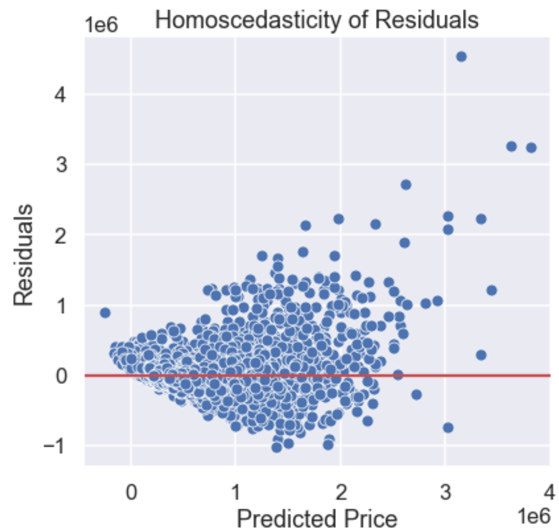
[36]:
```
# Run regression on cleaned dataframe
model_combined(df_model);
```

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.793
Model:                            OLS   Adj. R-squared:                  0.792
Method:                 Least Squares   F-statistic:                     1006.
Date:                Thu, 22 Apr 2021   Prob (F-statistic):               0.00
Time:                        22:29:30   Log-Likelihood:            -2.8434e+05
No. Observations:               21143   AIC:                         5.689e+05
Df Residuals:                   21062   BIC:                         5.695e+05
Df Model:                          80
Covariance Type:            nonrobust
=================================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
Intercept     -2.693e+07   6.51e+06     -4.140      0.000   -3.97e+07   -1.42e+07
bedrooms      -2.781e+04   1614.774    -17.224      0.000    -3.1e+04   -2.46e+04
bathrooms        1.29e+04  2621.916      4.920      0.000    7760.985     1.8e+04
sqft_lot          0.2365      0.031      7.642      0.000       0.176       0.297
floors        -6.476e+04   3127.358    -20.708      0.000   -7.09e+04   -5.86e+04
waterfront     8.813e+05   1.46e+04     60.424      0.000    8.53e+05     9.1e+05
```

grade	5.264e+04	1832.891	28.717	0.000	4.9e+04	5.62e+04
sqft_above	219.3557	3.115	70.429	0.000	213.251	225.460
sqft_basement	157.4838	3.686	42.720	0.000	150.258	164.709
lat	1.207e+05	6.69e+04	1.804	0.071	-1.04e+04	2.52e+05
long	-1.703e+05	4.83e+04	-3.523	0.000	-2.65e+05	-7.55e+04
sqft_living15	27.4453	2.985	9.193	0.000	21.594	33.297
zipcode_98002	5.715e+04	1.52e+04	3.760	0.000	2.74e+04	8.69e+04
zipcode_98003	-1.635e+04	1.37e+04	-1.195	0.232	-4.32e+04	1.05e+04
zipcode_98004	7.574e+05	2.47e+04	30.600	0.000	7.09e+05	8.06e+05
zipcode_98005	2.806e+05	2.64e+04	10.614	0.000	2.29e+05	3.32e+05
zipcode_98006	2.751e+05	2.16e+04	12.709	0.000	2.33e+05	3.18e+05
zipcode_98007	2.345e+05	2.73e+04	8.586	0.000	1.81e+05	2.88e+05
zipcode_98008	2.612e+05	2.6e+04	10.058	0.000	2.1e+05	3.12e+05
zipcode_98010	1.146e+05	2.33e+04	4.917	0.000	6.89e+04	1.6e+05
zipcode_98011	6.726e+04	3.38e+04	1.991	0.047	1034.942	1.33e+05
zipcode_98014	1.215e+05	3.71e+04	3.277	0.001	4.88e+04	1.94e+05
zipcode_98019	7.459e+04	3.67e+04	2.034	0.042	2728.027	1.46e+05
zipcode_98022	8.621e+04	2.03e+04	4.253	0.000	4.65e+04	1.26e+05
zipcode_98023	-5.151e+04	1.26e+04	-4.093	0.000	-7.62e+04	-2.68e+04
zipcode_98024	1.806e+05	3.27e+04	5.524	0.000	1.17e+05	2.45e+05
zipcode_98027	1.718e+05	2.23e+04	7.706	0.000	1.28e+05	2.16e+05
zipcode_98028	6.885e+04	3.28e+04	2.097	0.036	4507.689	1.33e+05
zipcode_98029	2.212e+05	2.55e+04	8.683	0.000	1.71e+05	2.71e+05
zipcode_98030	6440.1953	1.5e+04	0.428	0.669	-2.31e+04	3.59e+04
zipcode_98031	1.687e+04	1.57e+04	1.076	0.282	-1.39e+04	4.76e+04
zipcode_98032	9181.2935	1.81e+04	0.507	0.612	-2.63e+04	4.47e+04
zipcode_98033	3.43e+05	2.81e+04	12.190	0.000	2.88e+05	3.98e+05
zipcode_98034	1.685e+05	3.02e+04	5.583	0.000	1.09e+05	2.28e+05
zipcode_98038	5.275e+04	1.69e+04	3.115	0.002	1.96e+04	8.59e+04
zipcode_98039	1.275e+06	3.35e+04	38.079	0.000	1.21e+06	1.34e+06
zipcode_98040	5.198e+05	2.19e+04	23.764	0.000	4.77e+05	5.63e+05
zipcode_98042	2.321e+04	1.44e+04	1.615	0.106	-4962.935	5.14e+04
zipcode_98045	1.575e+05	3.13e+04	5.039	0.000	9.62e+04	2.19e+05
zipcode_98052	1.962e+05	2.88e+04	6.825	0.000	1.4e+05	2.53e+05
zipcode_98053	1.611e+05	3.08e+04	5.224	0.000	1.01e+05	2.21e+05
zipcode_98055	4.754e+04	1.74e+04	2.729	0.006	1.34e+04	8.17e+04
zipcode_98056	9.953e+04	1.89e+04	5.274	0.000	6.25e+04	1.37e+05
zipcode_98058	3.033e+04	1.65e+04	1.841	0.066	-1957.843	6.26e+04
zipcode_98059	7.367e+04	1.86e+04	3.969	0.000	3.73e+04	1.1e+05
zipcode_98065	1.18e+05	2.88e+04	4.098	0.000	6.15e+04	1.74e+05
zipcode_98070	-1.88e+04	2.17e+04	-0.867	0.386	-6.13e+04	2.37e+04
zipcode_98072	1.063e+05	3.36e+04	3.160	0.002	4.03e+04	1.72e+05
zipcode_98074	1.576e+05	2.72e+04	5.785	0.000	1.04e+05	2.11e+05
zipcode_98075	1.604e+05	2.62e+04	6.116	0.000	1.09e+05	2.12e+05
zipcode_98077	7.644e+04	3.5e+04	2.185	0.029	7873.688	1.45e+05
zipcode_98092	-2.541e+04	1.37e+04	-1.855	0.064	-5.23e+04	1439.737
zipcode_98102	5.076e+05	2.9e+04	17.532	0.000	4.51e+05	5.64e+05
zipcode_98103	3.306e+05	2.71e+04	12.201	0.000	2.78e+05	3.84e+05

```
zipcode_98105   4.71e+05   2.78e+04     16.967      0.000    4.17e+05    5.25e+05
zipcode_98106  1.245e+05   2.02e+04      6.177      0.000     8.5e+04    1.64e+05
zipcode_98107  3.323e+05    2.8e+04     11.882      0.000    2.77e+05    3.87e+05
zipcode_98108  1.132e+05   2.22e+04      5.099      0.000    6.97e+04    1.57e+05
zipcode_98109   4.99e+05   2.88e+04     17.319      0.000    4.43e+05    5.55e+05
zipcode_98112  6.152e+05   2.55e+04     24.168      0.000    5.65e+05    6.65e+05
zipcode_98115  3.155e+05   2.76e+04     11.436      0.000    2.61e+05     3.7e+05
zipcode_98116  3.002e+05   2.24e+04     13.379      0.000    2.56e+05    3.44e+05
zipcode_98117  2.948e+05   2.79e+04     10.552      0.000     2.4e+05     3.5e+05
zipcode_98118  1.769e+05   1.96e+04      9.036      0.000    1.39e+05    2.15e+05
zipcode_98119  4.967e+05   2.72e+04     18.259      0.000    4.43e+05     5.5e+05
zipcode_98122  3.457e+05   2.42e+04     14.279      0.000    2.98e+05    3.93e+05
zipcode_98125  1.726e+05   2.99e+04      5.780      0.000    1.14e+05    2.31e+05
zipcode_98126  1.959e+05   2.06e+04      9.494      0.000    1.55e+05    2.36e+05
zipcode_98133  1.233e+05   3.09e+04      3.996      0.000    6.28e+04    1.84e+05
zipcode_98136  2.492e+05   2.12e+04     11.770      0.000    2.08e+05    2.91e+05
zipcode_98144  2.904e+05   2.25e+04     12.881      0.000    2.46e+05    3.35e+05
zipcode_98146  1.078e+05   1.89e+04      5.692      0.000    7.07e+04    1.45e+05
zipcode_98148  4.939e+04   2.59e+04      1.907      0.057   -1381.603       1e+05
zipcode_98155  1.051e+05   3.21e+04      3.275      0.001    4.22e+04    1.68e+05
zipcode_98166  6.379e+04   1.73e+04      3.687      0.000    2.99e+04    9.77e+04
zipcode_98168  6.116e+04   1.83e+04      3.341      0.001    2.53e+04     9.7e+04
zipcode_98177  1.959e+05   3.22e+04      6.090      0.000    1.33e+05    2.59e+05
zipcode_98178  4.907e+04   1.89e+04      2.602      0.009    1.21e+04     8.6e+04
zipcode_98188  3.065e+04   1.95e+04      1.571      0.116   -7588.055    6.89e+04
zipcode_98198  1.591e+04   1.47e+04      1.079      0.281    -1.3e+04    4.48e+04
zipcode_98199  3.709e+05   2.65e+04     13.987      0.000    3.19e+05    4.23e+05
==============================================================================
Omnibus:                    20092.654   Durbin-Watson:                   1.985
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          3585946.415
Skew:                           4.107   Prob(JB):                         0.00
Kurtosis:                      66.270   Cond. No.                     2.47e+08
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly␣
 ↪specified.
[2] The condition number is large, 2.47e+08. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```





6.2 Removing Outliers to Fulfill Assumptions of Multiple Regressions

We have successfully run our baseline model, and our R2 value isn't too bad! However, we can see from the QQ plot and homoscedasticity plot that we are not fulfilling the assumptions of multiple regression.

We will try to address this issue by removing outliers that lie 1.5 times the IQR below the first quartile and 1.5 times the IQR above the third quartile.

[37]:
```
# Create function to remove outliers.
def find_outliers_IQR(data):
    """This function was written by James Irving during study group.

    Detects outliers using the 1.5*IQR thresholds.
    Returns a boolean Series where True=outlier"""
    res = data.describe()
    q1 = res['25%']
    q3 = res['75%']
    thresh = 1.5*(q3-q1)
    idx_outliers =(data < (q1-thresh)) | (data > (q3+thresh))
    return idx_outliers
```

In the 'Explore' section, we saw that we have many outliers several columns. We will proceed to remove outliers from those columns that have extreme outliers, based on our boxplot visualizations.

[38]:
```
# Create list of columns to remove outliers from
cols_outlier = ['price', 'bedrooms', 'bathrooms', 'sqft_lot', 'sqft_above', \
                'sqft_basement', 'sqft_living15']
df_outliers = df_model.copy()
```

```python
# Remove outliers for specified columns
for col in cols_outlier:
    df_outliers = df_outliers[~find_outliers_IQR(df_outliers[col])]
```

[39]:
```python
df_outliers
```

[39]:
```
          price  bedrooms  bathrooms  sqft_lot  floors  waterfront  grade  \
0      221900.0         3       1.00      5650     1.0         0.0      7
1      538000.0         3       2.25      7242     2.0         0.0      7
2      180000.0         2       1.00     10000     1.0         0.0      6
3      604000.0         4       3.00      5000     1.0         0.0      7
4      510000.0         3       2.00      8080     1.0         0.0      8
…           …       …        …         …       …       …
21592  360000.0         3       2.50      1131     3.0         0.0      8
21593  400000.0         4       2.50      5813     2.0         0.0      8
21594  402101.0         2       0.75      1350     2.0         0.0      7
21595  400000.0         3       2.50      2388     2.0         0.0      8
21596  325000.0         2       0.75      1076     2.0         0.0      7

       sqft_above  sqft_basement      lat  …  zipcode_98146  zipcode_98148  \
0            1180            0.0  47.5112  …            0.0            0.0
1            2170          400.0  47.7210  …            0.0            0.0
2             770            0.0  47.7379  …            0.0            0.0
3            1050          910.0  47.5208  …            0.0            0.0
4            1680            0.0  47.6168  …            0.0            0.0
…             …             …      …  …           …            …
21592        1530            0.0  47.6993  …            0.0            0.0
21593        2310            0.0  47.5107  …            1.0            0.0
21594        1020            0.0  47.5944  …            0.0            0.0
21595        1600            0.0  47.5345  …            0.0            0.0
21596        1020            0.0  47.5941  …            0.0            0.0

       zipcode_98155  zipcode_98166  zipcode_98168  zipcode_98177  \
0                0.0            0.0            0.0            0.0
1                0.0            0.0            0.0            0.0
2                0.0            0.0            0.0            0.0
3                0.0            0.0            0.0            0.0
4                0.0            0.0            0.0            0.0
…                …             …             …             …
21592            0.0            0.0            0.0            0.0
21593            0.0            0.0            0.0            0.0
21594            0.0            0.0            0.0            0.0
21595            0.0            0.0            0.0            0.0
21596            0.0            0.0            0.0            0.0

       zipcode_98178  zipcode_98188  zipcode_98198  zipcode_98199
0                1.0            0.0            0.0            0.0
```

```
1                0.0          0.0          0.0          0.0
2                0.0          0.0          0.0          0.0
3                0.0          0.0          0.0          0.0
4                0.0          0.0          0.0          0.0
…               …           …           …           …
21592            0.0          0.0          0.0          0.0
21593            0.0          0.0          0.0          0.0
21594            0.0          0.0          0.0          0.0
21595            0.0          0.0          0.0          0.0
21596            0.0          0.0          0.0          0.0

[16358 rows x 81 columns]
```

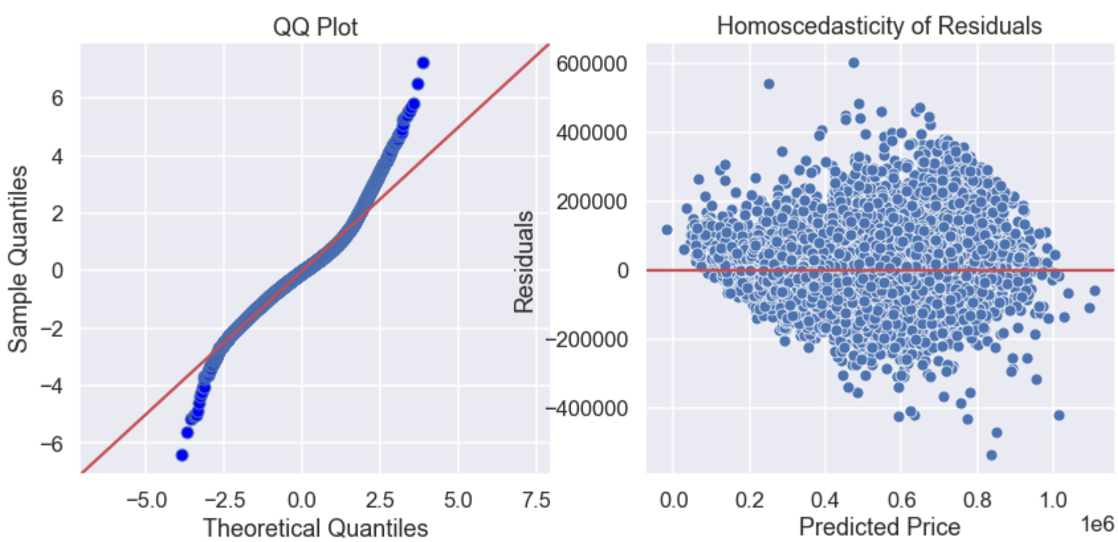
[40]:

```
# Run regression model on our dataset where outliers are removed.
model_combined(df_outliers);
```

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.807
Model:                            OLS   Adj. R-squared:                  0.806
Method:                 Least Squares   F-statistic:                     848.1
Date:                Thu, 22 Apr 2021   Prob (F-statistic):               0.00
Time:                        22:29:31   Log-Likelihood:            -2.0856e+05
No. Observations:               16358   AIC:                         4.173e+05
Df Residuals:                   16277   BIC:                         4.179e+05
Df Model:                          80
Covariance Type:            nonrobust
=================================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
Intercept     -2.098e+06   4.45e+06     -0.471      0.638   -1.08e+07    6.63e+06
bedrooms      -2850.0552   1115.288     -2.555      0.011   -5036.141    -663.969
bathrooms      6959.9481   1611.666      4.318      0.000    3800.906    1.01e+04
sqft_lot          2.8752      0.276     10.421      0.000       2.334       3.416
floors         -2.75e+04   1951.375    -14.092      0.000   -3.13e+04   -2.37e+04
waterfront     3.382e+05    1.87e+04     18.087      0.000    3.02e+05    3.75e+05
grade          3.475e+04   1149.769     30.221      0.000    3.25e+04     3.7e+04
sqft_above      130.3372      2.348     55.502      0.000     125.734     134.940
sqft_basement    90.9682      2.638     34.480      0.000      85.797      96.140
lat           -4.173e+04    4.19e+04     -0.995      0.320   -1.24e+05    4.05e+04
long          -3.107e+04    3.37e+04     -0.922      0.356   -9.71e+04     3.5e+04
sqft_living15    34.8750      2.108     16.545      0.000      30.743      39.007
zipcode_98002  3.279e+04   8193.908      4.001      0.000    1.67e+04    4.88e+04
zipcode_98003  5708.5590   7433.852      0.768      0.443   -8862.607    2.03e+04
zipcode_98004  5.463e+05    1.58e+04     34.605      0.000    5.15e+05    5.77e+05
```

zipcode_98005	3.521e+05	1.62e+04	21.728	0.000	3.2e+05	3.84e+05
zipcode_98006	2.942e+05	1.34e+04	21.879	0.000	2.68e+05	3.21e+05
zipcode_98007	2.793e+05	1.64e+04	17.022	0.000	2.47e+05	3.11e+05
zipcode_98008	2.716e+05	1.59e+04	17.079	0.000	2.4e+05	3.03e+05
zipcode_98010	1.052e+05	1.61e+04	6.535	0.000	7.37e+04	1.37e+05
zipcode_98011	1.656e+05	2.06e+04	8.024	0.000	1.25e+05	2.06e+05
zipcode_98014	1.37e+05	2.61e+04	5.255	0.000	8.59e+04	1.88e+05
zipcode_98019	1.15e+05	2.29e+04	5.023	0.000	7.01e+04	1.6e+05
zipcode_98022	3.168e+04	1.33e+04	2.382	0.017	5610.093	5.78e+04
zipcode_98023	-1.431e+04	7174.382	-1.995	0.046	-2.84e+04	-247.789
zipcode_98024	1.662e+05	2.45e+04	6.796	0.000	1.18e+05	2.14e+05
zipcode_98027	2.528e+05	1.46e+04	17.348	0.000	2.24e+05	2.81e+05
zipcode_98028	1.532e+05	2.01e+04	7.620	0.000	1.14e+05	1.93e+05
zipcode_98029	2.584e+05	1.6e+04	16.180	0.000	2.27e+05	2.9e+05
zipcode_98030	1.107e+04	8322.238	1.330	0.184	-5244.964	2.74e+04
zipcode_98031	2.545e+04	8842.824	2.878	0.004	8117.202	4.28e+04
zipcode_98032	1.577e+04	9774.747	1.614	0.107	-3384.816	3.49e+04
zipcode_98033	3.44e+05	1.74e+04	19.733	0.000	3.1e+05	3.78e+05
zipcode_98034	2.116e+05	1.86e+04	11.375	0.000	1.75e+05	2.48e+05
zipcode_98038	4.761e+04	1.05e+04	4.525	0.000	2.7e+04	6.82e+04
zipcode_98039	6.678e+05	3.71e+04	18.020	0.000	5.95e+05	7.4e+05
zipcode_98040	4.52e+05	1.42e+04	31.871	0.000	4.24e+05	4.8e+05
zipcode_98042	2.367e+04	8705.330	2.719	0.007	6610.572	4.07e+04
zipcode_98045	1.206e+05	2.06e+04	5.851	0.000	8.02e+04	1.61e+05
zipcode_98052	2.762e+05	1.77e+04	15.598	0.000	2.41e+05	3.11e+05
zipcode_98053	2.734e+05	2.02e+04	13.520	0.000	2.34e+05	3.13e+05
zipcode_98055	6.081e+04	1e+04	6.051	0.000	4.11e+04	8.05e+04
zipcode_98056	1.314e+05	1.12e+04	11.688	0.000	1.09e+05	1.53e+05
zipcode_98058	5.115e+04	9796.055	5.221	0.000	3.19e+04	7.03e+04
zipcode_98059	1.02e+05	1.12e+04	9.127	0.000	8.01e+04	1.24e+05
zipcode_98065	1.585e+05	1.86e+04	8.521	0.000	1.22e+05	1.95e+05
zipcode_98070	8.554e+04	1.89e+04	4.521	0.000	4.85e+04	1.23e+05
zipcode_98072	1.76e+05	2.12e+04	8.312	0.000	1.35e+05	2.18e+05
zipcode_98074	2.267e+05	1.72e+04	13.149	0.000	1.93e+05	2.6e+05
zipcode_98075	2.514e+05	1.72e+04	14.627	0.000	2.18e+05	2.85e+05
zipcode_98077	1.773e+05	2.61e+04	6.798	0.000	1.26e+05	2.28e+05
zipcode_98092	-1.667e+04	7912.858	-2.107	0.035	-3.22e+04	-1159.382
zipcode_98102	4.591e+05	1.73e+04	26.564	0.000	4.25e+05	4.93e+05
zipcode_98103	3.806e+05	1.66e+04	22.896	0.000	3.48e+05	4.13e+05
zipcode_98105	4.348e+05	1.7e+04	25.507	0.000	4.01e+05	4.68e+05
zipcode_98106	1.511e+05	1.2e+04	12.608	0.000	1.28e+05	1.75e+05
zipcode_98107	3.775e+05	1.7e+04	22.229	0.000	3.44e+05	4.11e+05
zipcode_98108	1.532e+05	1.3e+04	11.825	0.000	1.28e+05	1.79e+05
zipcode_98109	4.707e+05	1.74e+04	27.075	0.000	4.37e+05	5.05e+05
zipcode_98112	4.866e+05	1.58e+04	30.810	0.000	4.56e+05	5.18e+05
zipcode_98115	3.67e+05	1.69e+04	21.701	0.000	3.34e+05	4e+05
zipcode_98116	3.521e+05	1.35e+04	25.985	0.000	3.25e+05	3.79e+05
zipcode_98117	3.64e+05	1.72e+04	21.192	0.000	3.3e+05	3.98e+05

```
zipcode_98118  2.024e+05   1.17e+04     17.309      0.000     1.8e+05    2.25e+05
zipcode_98119  4.688e+05   1.65e+04     28.358      0.000    4.36e+05    5.01e+05
zipcode_98122  3.629e+05   1.46e+04     24.918      0.000    3.34e+05    3.91e+05
zipcode_98125  2.325e+05   1.83e+04     12.722      0.000    1.97e+05    2.68e+05
zipcode_98126  2.406e+05   1.23e+04     19.517      0.000    2.16e+05    2.65e+05
zipcode_98133  1.894e+05   1.89e+04      9.994      0.000    1.52e+05    2.27e+05
zipcode_98136  3.027e+05   1.26e+04     24.041      0.000    2.78e+05    3.27e+05
zipcode_98144  2.933e+05   1.36e+04     21.592      0.000    2.67e+05     3.2e+05
zipcode_98146  1.346e+05   1.11e+04     12.104      0.000    1.13e+05    1.56e+05
zipcode_98148  6.492e+04   1.37e+04      4.744      0.000    3.81e+04    9.17e+04
zipcode_98155  1.728e+05   1.97e+04      8.771      0.000    1.34e+05    2.11e+05
zipcode_98166  1.211e+05   1.03e+04     11.763      0.000    1.01e+05    1.41e+05
zipcode_98168   6.63e+04   1.07e+04      6.195      0.000    4.53e+04    8.73e+04
zipcode_98177  2.393e+05   1.98e+04     12.090      0.000    2.01e+05    2.78e+05
zipcode_98178  8.289e+04   1.09e+04      7.593      0.000    6.15e+04    1.04e+05
zipcode_98188  5.228e+04   1.09e+04      4.790      0.000    3.09e+04    7.37e+04
zipcode_98198  4.982e+04   8274.755      6.020      0.000    3.36e+04     6.6e+04
zipcode_98199  4.024e+05   1.63e+04     24.658      0.000     3.7e+05    4.34e+05
==============================================================================
Omnibus:                     1839.987   Durbin-Watson:                   2.005
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             6354.147
Skew:                           0.558   Prob(JB):                         0.00
Kurtosis:                       5.842   Cond. No.                     5.56e+07
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
 ↪specified.
[2] The condition number is large, 5.56e+07. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```



Great! We can see that although they are not quite perfect, our QQ plot and homoscedasticity plot look much better. We can see that our R2 value has gone up a bit as well.

Now we want to move on to addressing the nonsignificant P-values in our model. Since a nonsignificant P-value is indicates that our model would be no different than when the respective coefficient is 0, we will go ahead and remove those variables from our model.

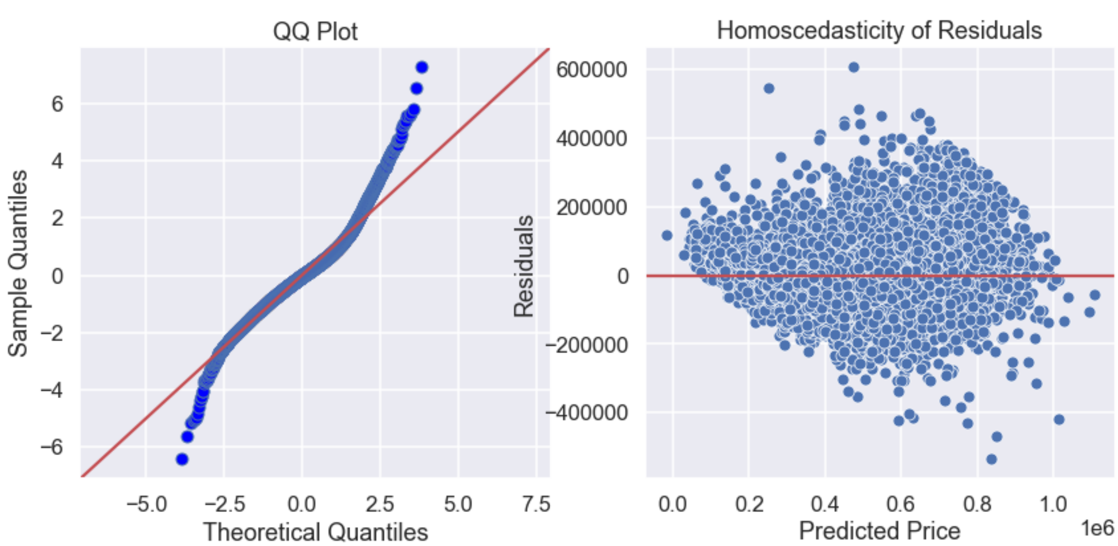
[41]:
```
# Create new dataframe after removing outliers
df_pvalues = df_outliers.drop(['lat', 'long'], axis=1)
```

[42]:
```
# Run regression on dataframe after removing non-significant variables
model_unscaled, fig_unscaled, ax_unscaled = model_combined(df_pvalues)
```

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.806
Model:                            OLS   Adj. R-squared:                  0.806
Method:                 Least Squares   F-statistic:                     869.8
Date:                Thu, 22 Apr 2021   Prob (F-statistic):               0.00
Time:                        22:29:31   Log-Likelihood:            -2.0856e+05
No. Observations:               16358   AIC:                         4.173e+05
Df Residuals:                   16279   BIC:                         4.179e+05
Df Model:                          78
Covariance Type:            nonrobust
=================================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
Intercept      -2.73e+05   8978.380    -30.405      0.000   -2.91e+05   -2.55e+05
bedrooms      -2849.0091   1115.274     -2.555      0.011   -5035.068    -662.950
bathrooms      6955.4124   1611.577      4.316      0.000    3796.545    1.01e+04
sqft_lot          2.8718      0.276     10.409      0.000       2.331       3.413
floors        -2.748e+04   1951.119    -14.082      0.000   -3.13e+04   -2.37e+04
waterfront     3.385e+05    1.87e+04     18.106      0.000    3.02e+05    3.75e+05
grade          3.476e+04   1149.100     30.250      0.000    3.25e+04     3.7e+04
sqft_above      130.3278      2.348     55.500      0.000     125.725     134.931
sqft_basement    90.9779      2.638     34.486      0.000      85.807      96.149
sqft_living15    34.8325      2.107     16.528      0.000      30.702      38.963
zipcode_98002  3.111e+04   7975.317      3.901      0.000    1.55e+04    4.67e+04
zipcode_98003  6760.4627   7310.504      0.925      0.355   -7568.927    2.11e+04
zipcode_98004  5.316e+05   9106.136     58.382      0.000    5.14e+05    5.49e+05
zipcode_98005  3.368e+05   9855.624     34.175      0.000    3.17e+05    3.56e+05
zipcode_98006  2.802e+05   7147.222     39.199      0.000    2.66e+05    2.94e+05
zipcode_98007  2.628e+05   9275.845     28.334      0.000    2.45e+05    2.81e+05
zipcode_98008  2.541e+05   7420.469     34.249      0.000     2.4e+05    2.69e+05
```

zipcode_98010	9.676e+04	1.36e+04	7.131	0.000	7.02e+04	1.23e+05
zipcode_98011	1.45e+05	8316.130	17.434	0.000	1.29e+05	1.61e+05
zipcode_98014	1.08e+05	1.39e+04	7.790	0.000	8.08e+04	1.35e+05
zipcode_98019	8.778e+04	8780.990	9.997	0.000	7.06e+04	1.05e+05
zipcode_98022	2.768e+04	8721.026	3.174	0.002	1.06e+04	4.48e+04
zipcode_98023	-1.125e+04	6426.804	-1.751	0.080	-2.39e+04	1342.806
zipcode_98024	1.443e+05	1.78e+04	8.107	0.000	1.09e+05	1.79e+05
zipcode_98027	2.366e+05	7721.041	30.640	0.000	2.21e+05	2.52e+05
zipcode_98028	1.339e+05	7433.627	18.012	0.000	1.19e+05	1.48e+05
zipcode_98029	2.397e+05	7261.295	33.009	0.000	2.25e+05	2.54e+05
zipcode_98030	6138.2120	7420.942	0.827	0.408	-8407.648	2.07e+04
zipcode_98031	1.904e+04	7352.505	2.590	0.010	4630.856	3.35e+04
zipcode_98032	1.343e+04	9420.742	1.426	0.154	-5032.423	3.19e+04
zipcode_98033	3.259e+05	6946.511	46.914	0.000	3.12e+05	3.4e+05
zipcode_98034	1.924e+05	6329.075	30.403	0.000	1.8e+05	2.05e+05
zipcode_98038	3.808e+04	6374.554	5.974	0.000	2.56e+04	5.06e+04
zipcode_98039	6.533e+05	3.45e+04	18.938	0.000	5.86e+05	7.21e+05
zipcode_98040	4.401e+05	9275.468	47.452	0.000	4.22e+05	4.58e+05
zipcode_98042	1.654e+04	6449.312	2.565	0.010	3897.958	2.92e+04
zipcode_98045	9.826e+04	8771.005	11.203	0.000	8.11e+04	1.15e+05
zipcode_98052	2.562e+05	6441.635	39.777	0.000	2.44e+05	2.69e+05
zipcode_98053	2.497e+05	7924.596	31.508	0.000	2.34e+05	2.65e+05
zipcode_98055	5.255e+04	7470.196	7.035	0.000	3.79e+04	6.72e+04
zipcode_98056	1.204e+05	6723.038	17.906	0.000	1.07e+05	1.34e+05
zipcode_98058	4.178e+04	6661.999	6.272	0.000	2.87e+04	5.48e+04
zipcode_98059	9.057e+04	6761.772	13.394	0.000	7.73e+04	1.04e+05
zipcode_98065	1.368e+05	7640.870	17.898	0.000	1.22e+05	1.52e+05
zipcode_98070	8.72e+04	1.74e+04	5.009	0.000	5.31e+04	1.21e+05
zipcode_98072	1.537e+05	8855.588	17.355	0.000	1.36e+05	1.71e+05
zipcode_98074	2.065e+05	7135.626	28.933	0.000	1.92e+05	2.2e+05
zipcode_98075	2.322e+05	8918.569	26.034	0.000	2.15e+05	2.5e+05
zipcode_98077	1.529e+05	1.69e+04	9.066	0.000	1.2e+05	1.86e+05
zipcode_98092	-1.899e+04	7377.054	-2.575	0.010	-3.35e+04	-4532.805
zipcode_98102	4.472e+05	1.07e+04	41.757	0.000	4.26e+05	4.68e+05
zipcode_98103	3.677e+05	6345.370	57.943	0.000	3.55e+05	3.8e+05
zipcode_98105	4.209e+05	8354.953	50.372	0.000	4.04e+05	4.37e+05
zipcode_98106	1.443e+05	7015.786	20.573	0.000	1.31e+05	1.58e+05
zipcode_98107	3.657e+05	7473.143	48.939	0.000	3.51e+05	3.8e+05
zipcode_98108	1.443e+05	8224.265	17.550	0.000	1.28e+05	1.6e+05
zipcode_98109	4.596e+05	1.07e+04	42.874	0.000	4.39e+05	4.81e+05
zipcode_98112	4.742e+05	8524.249	55.627	0.000	4.57e+05	4.91e+05
zipcode_98115	3.523e+05	6321.102	55.740	0.000	3.4e+05	3.65e+05
zipcode_98116	3.449e+05	7158.996	48.178	0.000	3.31e+05	3.59e+05
zipcode_98117	3.517e+05	6393.125	55.010	0.000	3.39e+05	3.64e+05
zipcode_98118	1.929e+05	6445.325	29.926	0.000	1.8e+05	2.06e+05
zipcode_98119	4.58e+05	8775.961	52.187	0.000	4.41e+05	4.75e+05
zipcode_98122	3.514e+05	7434.646	47.260	0.000	3.37e+05	3.66e+05
zipcode_98125	2.165e+05	6684.149	32.394	0.000	2.03e+05	2.3e+05

```
zipcode_98126  2.341e+05   6939.693     33.740      0.000     2.21e+05    2.48e+05
zipcode_98133  1.739e+05   6376.183     27.274      0.000     1.61e+05    1.86e+05
zipcode_98136  2.968e+05   7511.155     39.511      0.000     2.82e+05    3.11e+05
zipcode_98144  2.827e+05   7165.040     39.455      0.000     2.69e+05    2.97e+05
zipcode_98146  1.294e+05   7342.058     17.621      0.000     1.15e+05    1.44e+05
zipcode_98148  6.152e+04   1.25e+04      4.912      0.000      3.7e+04    8.61e+04
zipcode_98155  1.554e+05   6586.017     23.588      0.000     1.42e+05    1.68e+05
zipcode_98166  1.174e+05   7950.541     14.771      0.000     1.02e+05    1.33e+05
zipcode_98168  6.001e+04   7585.424      7.912      0.000     4.51e+04    7.49e+04
zipcode_98177  2.244e+05   7994.410     28.071      0.000     2.09e+05     2.4e+05
zipcode_98178  7.431e+04   7442.250      9.985      0.000     5.97e+04    8.89e+04
zipcode_98188  4.692e+04   9296.686      5.047      0.000     2.87e+04    6.51e+04
zipcode_98198  4.784e+04   7434.966      6.435      0.000     3.33e+04    6.24e+04
zipcode_98199  3.922e+05   7476.812     52.459      0.000     3.78e+05    4.07e+05
==============================================================================
Omnibus:                     1844.788   Durbin-Watson:                   2.004
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             6380.863
Skew:                           0.559   Prob(JB):                         0.00
Kurtosis:                       5.848   Cond. No.                     5.37e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly␣
 ↪specified.
[2] The condition number is large, 5.37e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```



7 iNTERPRET

Now that we have our final model with outliers removed and only significant P-values included, all that's left in our analysis is to scale our model coefficients to determine which coefficients have the largest effect on the variability of housing price. Since there are multiple coefficients for zipcode, we will examine which of the other variables have high coefficients.

We should also note that zipcode, as well as some other variables are ones that we cannot control, and therefore will not be appropriate variables to provide recommendations for changing. However, we will still include those variables as part of our model, as long as they have a high enough coefficient to indicate that they are valid predictors for the value of a house.

7.1 Scaling the Dataset

[43]:
```
# Create copy of final dataset to scale
df_unscaled = df_pvalues.copy()
```

[44]:
```
# Create list of columns except for zipcode
numeric_cols = [col for col in df_unscaled.columns if \
                col.startswith('zipcode')==False]
numeric_cols
```

[44]:
```
['price',
 'bedrooms',
 'bathrooms',
 'sqft_lot',
 'floors',
 'waterfront',
 'grade',
 'sqft_above',
 'sqft_basement',
 'sqft_living15']
```

[45]:
```
# Create scaler object
scaler = StandardScaler()
scaler
```

[45]:
```
StandardScaler()
```

[46]:
```
# Scale our dataset used to form our final model
df_scaled = df_unscaled.copy()
df_scaled[numeric_cols] = scaler.fit_transform(df_scaled[numeric_cols])
df_scaled.describe().round(2)
```

[46]:
```
          price  bedrooms  bathrooms  sqft_lot    floors  waterfront  \
count  16358.00  16358.00   16358.00  16358.00  16358.00    16358.00
mean      -0.00     -0.00       0.00      0.00      0.00       -0.00
std        1.00      1.00       1.00      1.00      1.00        1.00
```

```
min       -1.95     -1.62      -2.24     -1.93     -0.85        -0.04
25%       -0.77     -0.32      -0.72     -0.69     -0.85        -0.04
50%       -0.18     -0.32       0.04      0.00     -0.85        -0.04
75%        0.59      0.98       0.80      0.58      0.98        -0.04
max        3.55      2.28       3.08      3.28      3.74        27.89

          grade  sqft_above  sqft_basement  sqft_living15  …  zipcode_98146  \
count  16358.00    16358.00       16358.00       16358.00  …       16358.00
mean      -0.00        0.00          -0.00          -0.00  …           0.01
std        1.00        1.00           1.00           1.00  …           0.12
min       -3.83       -1.96          -0.67          -2.71  …           0.00
25%       -0.44       -0.75          -0.67          -0.74  …           0.00
50%       -0.44       -0.24          -0.67          -0.17  …           0.00
75%        0.69        0.59           0.68           0.63  …           0.00
max        4.08        2.91           2.91           2.80  …           1.00

       zipcode_98148  zipcode_98155  zipcode_98166  zipcode_98168  \
count       16358.00       16358.00       16358.00       16358.00
mean            0.00           0.02           0.01           0.01
std             0.06           0.15           0.11           0.11
min             0.00           0.00           0.00           0.00
25%             0.00           0.00           0.00           0.00
50%             0.00           0.00           0.00           0.00
75%             0.00           0.00           0.00           0.00
max             1.00           1.00           1.00           1.00

       zipcode_98177  zipcode_98178  zipcode_98188  zipcode_98198  \
count       16358.00       16358.00       16358.00       16358.00
mean            0.01           0.01           0.01           0.01
std             0.10           0.12           0.08           0.12
min             0.00           0.00           0.00           0.00
25%             0.00           0.00           0.00           0.00
50%             0.00           0.00           0.00           0.00
75%             0.00           0.00           0.00           0.00
max             1.00           1.00           1.00           1.00

       zipcode_98199
count       16358.00
mean            0.01
std             0.12
min             0.00
25%             0.00
50%             0.00
75%             0.00
max             1.00

[8 rows x 79 columns]
```

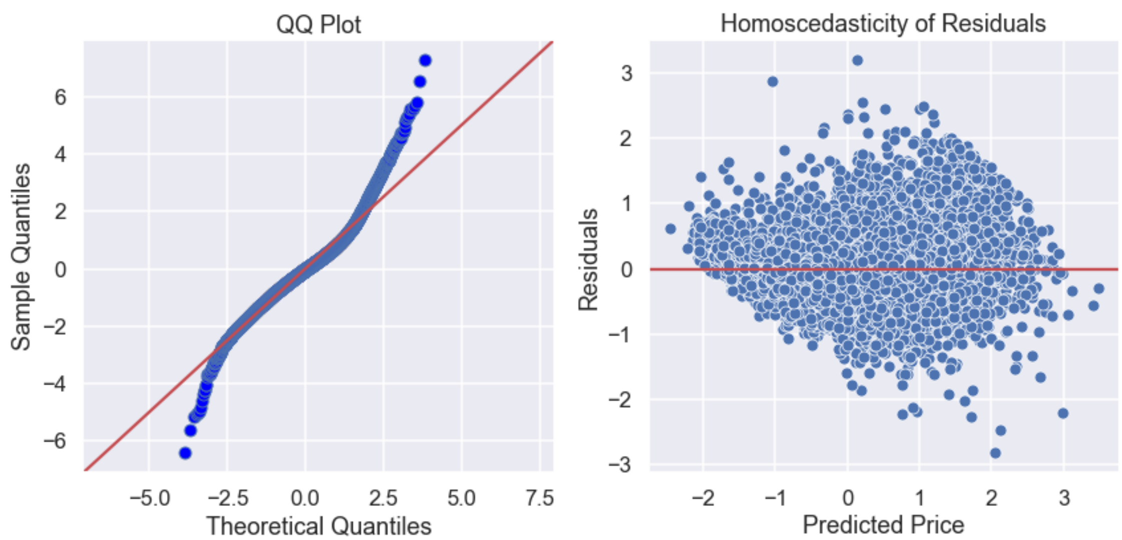
7.2 Creating a Scaled Model

[47]:
```
# Run regression model on scaled data
model_scaled, fig_scaled, ax_scaled = model_combined(df_scaled)
```

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.806
Model:                            OLS   Adj. R-squared:                  0.806
Method:                 Least Squares   F-statistic:                     869.8
Date:                Thu, 22 Apr 2021   Prob (F-statistic):               0.00
Time:                        22:29:33   Log-Likelihood:                -9777.5
No. Observations:               16358   AIC:                         1.971e+04
Df Residuals:                   16279   BIC:                         2.032e+04
Df Model:                          78
Covariance Type:            nonrobust
=================================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
Intercept        -1.0046      0.027    -37.858      0.000      -1.057      -0.953
bedrooms         -0.0116      0.005     -2.555      0.011      -0.020      -0.003
bathrooms         0.0242      0.006      4.316      0.000       0.013       0.035
sqft_lot          0.0514      0.005     10.409      0.000       0.042       0.061
floors           -0.0790      0.006    -14.082      0.000      -0.090      -0.068
waterfront        0.0640      0.004     18.106      0.000       0.057       0.071
grade             0.1623      0.005     30.250      0.000       0.152       0.173
sqft_above        0.3970      0.007     55.500      0.000       0.383       0.411
sqft_basement     0.1700      0.005     34.486      0.000       0.160       0.180
sqft_living15     0.0908      0.005     16.528      0.000       0.080       0.102
zipcode_98002     0.1642      0.042      3.901      0.000       0.082       0.247
zipcode_98003     0.0357      0.039      0.925      0.355      -0.040       0.111
zipcode_98004     2.8061      0.048     58.382      0.000       2.712       2.900
zipcode_98005     1.7778      0.052     34.175      0.000       1.676       1.880
zipcode_98006     1.4788      0.038     39.199      0.000       1.405       1.553
zipcode_98007     1.3872      0.049     28.334      0.000       1.291       1.483
zipcode_98008     1.3414      0.039     34.249      0.000       1.265       1.418
zipcode_98010     0.5107      0.072      7.131      0.000       0.370       0.651
zipcode_98011     0.7653      0.044     17.434      0.000       0.679       0.851
zipcode_98014     0.5701      0.073      7.790      0.000       0.427       0.714
zipcode_98019     0.4633      0.046      9.997      0.000       0.372       0.554
zipcode_98022     0.1461      0.046      3.174      0.002       0.056       0.236
zipcode_98023    -0.0594      0.034     -1.751      0.080      -0.126       0.007
zipcode_98024     0.7616      0.094      8.107      0.000       0.577       0.946
zipcode_98027     1.2487      0.041     30.640      0.000       1.169       1.329
zipcode_98028     0.7067      0.039     18.012      0.000       0.630       0.784
zipcode_98029     1.2652      0.038     33.009      0.000       1.190       1.340
```

zipcode_98030	0.0324	0.039	0.827	0.408	-0.044	0.109
zipcode_98031	0.1005	0.039	2.590	0.010	0.024	0.177
zipcode_98032	0.0709	0.050	1.426	0.154	-0.027	0.168
zipcode_98033	1.7202	0.037	46.914	0.000	1.648	1.792
zipcode_98034	1.0157	0.033	30.403	0.000	0.950	1.081
zipcode_98038	0.2010	0.034	5.974	0.000	0.135	0.267
zipcode_98039	3.4485	0.182	18.938	0.000	3.092	3.805
zipcode_98040	2.3232	0.049	47.452	0.000	2.227	2.419
zipcode_98042	0.0873	0.034	2.565	0.010	0.021	0.154
zipcode_98045	0.5187	0.046	11.203	0.000	0.428	0.609
zipcode_98052	1.3525	0.034	39.777	0.000	1.286	1.419
zipcode_98053	1.3179	0.042	31.508	0.000	1.236	1.400
zipcode_98055	0.2774	0.039	7.035	0.000	0.200	0.355
zipcode_98056	0.6354	0.035	17.906	0.000	0.566	0.705
zipcode_98058	0.2205	0.035	6.272	0.000	0.152	0.289
zipcode_98059	0.4780	0.036	13.394	0.000	0.408	0.548
zipcode_98065	0.7218	0.040	17.898	0.000	0.643	0.801
zipcode_98070	0.4602	0.092	5.009	0.000	0.280	0.640
zipcode_98072	0.8112	0.047	17.355	0.000	0.720	0.903
zipcode_98074	1.0897	0.038	28.933	0.000	1.016	1.164
zipcode_98075	1.2255	0.047	26.034	0.000	1.133	1.318
zipcode_98077	0.8072	0.089	9.066	0.000	0.633	0.982
zipcode_98092	-0.1002	0.039	-2.575	0.010	-0.177	-0.024
zipcode_98102	2.3603	0.057	41.757	0.000	2.250	2.471
zipcode_98103	1.9407	0.033	57.943	0.000	1.875	2.006
zipcode_98105	2.2214	0.044	50.372	0.000	2.135	2.308
zipcode_98106	0.7618	0.037	20.573	0.000	0.689	0.834
zipcode_98107	1.9304	0.039	48.939	0.000	1.853	2.008
zipcode_98108	0.7618	0.043	17.550	0.000	0.677	0.847
zipcode_98109	2.4259	0.057	42.874	0.000	2.315	2.537
zipcode_98112	2.5028	0.045	55.627	0.000	2.415	2.591
zipcode_98115	1.8598	0.033	55.740	0.000	1.794	1.925
zipcode_98116	1.8205	0.038	48.178	0.000	1.746	1.895
zipcode_98117	1.8563	0.034	55.010	0.000	1.790	1.922
zipcode_98118	1.0181	0.034	29.926	0.000	0.951	1.085
zipcode_98119	2.4174	0.046	52.187	0.000	2.327	2.508
zipcode_98122	1.8546	0.039	47.260	0.000	1.778	1.932
zipcode_98125	1.1429	0.035	32.394	0.000	1.074	1.212
zipcode_98126	1.2359	0.037	33.740	0.000	1.164	1.308
zipcode_98133	0.9179	0.034	27.274	0.000	0.852	0.984
zipcode_98136	1.5665	0.040	39.511	0.000	1.489	1.644
zipcode_98144	1.4922	0.038	39.455	0.000	1.418	1.566
zipcode_98146	0.6829	0.039	17.621	0.000	0.607	0.759
zipcode_98148	0.3247	0.066	4.912	0.000	0.195	0.454
zipcode_98155	0.8200	0.035	23.588	0.000	0.752	0.888
zipcode_98166	0.6199	0.042	14.771	0.000	0.538	0.702
zipcode_98168	0.3168	0.040	7.912	0.000	0.238	0.395
zipcode_98177	1.1845	0.042	28.071	0.000	1.102	1.267

```
zipcode_98178     0.3922      0.039      9.985      0.000      0.315      0.469
zipcode_98188     0.2477      0.049      5.047      0.000      0.151      0.344
zipcode_98198     0.2525      0.039      6.435      0.000      0.176      0.329
zipcode_98199     2.0703      0.039     52.459      0.000      1.993      2.148
==============================================================================
Omnibus:                     1844.788   Durbin-Watson:                   2.004
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             6380.863
Skew:                           0.559   Prob(JB):                         0.00
Kurtosis:                       5.848   Cond. No.                         122.
==============================================================================

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



7.3 Selecting Variables to Recommend

Now that we have a scaled model, we can pick out the variables with the highest coefficients. This means that we are selecting variables which have the largest impact on the variability of the value of a house.

[48]:
```
# Create dataframe of coefficients sorted by highest absolute value
coeffs = model_scaled.params.sort_values().to_frame('coeffs')
coeffs['abs'] = coeffs['coeffs'].abs()
coeffs.sort_values('abs', ascending=False, inplace=True)
coeffs.reset_index(inplace=True)
coeffs[~coeffs['index'].str.startswith('zipcode')]
```

[48]:
```
            index    coeffs       abs
33      Intercept -1.004551  1.004551
53     sqft_above  0.397017  0.397017
62  sqft_basement  0.170033  0.170033
64          grade  0.162327  0.162327
68  sqft_living15  0.090830  0.090830
70         floors -0.078977  0.078977
72     waterfront  0.063982  0.063982
74       sqft_lot  0.051372  0.051372
77      bathrooms  0.024169  0.024169
78       bedrooms -0.011559  0.011559
```

We can see that aside from the intercept, our coefficients for 'sqft_above', 'sqft_basement', and 'grade' have the most impact on price. Therefore, we will select those variables to interpret and make recommendations to our stakeholder on.

[49]:
```python
model_unscaled.summary()
```

[49]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.806
Model:                            OLS   Adj. R-squared:                  0.806
Method:                 Least Squares   F-statistic:                     869.8
Date:                Thu, 22 Apr 2021   Prob (F-statistic):               0.00
Time:                        22:29:33   Log-Likelihood:            -2.0856e+05
No. Observations:               16358   AIC:                         4.173e+05
Df Residuals:                   16279   BIC:                         4.179e+05
Df Model:                          78
Covariance Type:            nonrobust
================================================================================
=
                    coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-
Intercept      -2.73e+05   8978.380    -30.405      0.000   -2.91e+05
-2.55e+05
bedrooms      -2849.0091   1115.274     -2.555      0.011   -5035.068
-662.950
bathrooms      6955.4124   1611.577      4.316      0.000    3796.545
1.01e+04
sqft_lot          2.8718      0.276     10.409      0.000       2.331
3.413
floors        -2.748e+04   1951.119    -14.082      0.000   -3.13e+04
-2.37e+04
waterfront     3.385e+05    1.87e+04     18.106      0.000    3.02e+05
```

```
3.75e+05
grade           3.476e+04   1149.100     30.250      0.000    3.25e+04
3.7e+04
sqft_above       130.3278      2.348     55.500      0.000     125.725
134.931
sqft_basement     90.9779      2.638     34.486      0.000      85.807
96.149
sqft_living15     34.8325      2.107     16.528      0.000      30.702
38.963
zipcode_98002  3.111e+04   7975.317      3.901      0.000    1.55e+04
4.67e+04
zipcode_98003  6760.4627   7310.504      0.925      0.355   -7568.927
2.11e+04
zipcode_98004  5.316e+05   9106.136     58.382      0.000    5.14e+05
5.49e+05
zipcode_98005  3.368e+05   9855.624     34.175      0.000    3.17e+05
3.56e+05
zipcode_98006  2.802e+05   7147.222     39.199      0.000    2.66e+05
2.94e+05
zipcode_98007  2.628e+05   9275.845     28.334      0.000    2.45e+05
2.81e+05
zipcode_98008  2.541e+05   7420.469     34.249      0.000     2.4e+05
2.69e+05
zipcode_98010  9.676e+04    1.36e+04     7.131      0.000    7.02e+04
1.23e+05
zipcode_98011   1.45e+05   8316.130     17.434      0.000    1.29e+05
1.61e+05
zipcode_98014   1.08e+05    1.39e+04     7.790      0.000    8.08e+04
1.35e+05
zipcode_98019  8.778e+04   8780.990      9.997      0.000    7.06e+04
1.05e+05
zipcode_98022  2.768e+04   8721.026      3.174      0.002    1.06e+04
4.48e+04
zipcode_98023 -1.125e+04   6426.804     -1.751      0.080   -2.39e+04
1342.806
zipcode_98024  1.443e+05    1.78e+04     8.107      0.000    1.09e+05
1.79e+05
zipcode_98027  2.366e+05   7721.041     30.640      0.000    2.21e+05
2.52e+05
zipcode_98028  1.339e+05   7433.627     18.012      0.000    1.19e+05
1.48e+05
zipcode_98029  2.397e+05   7261.295     33.009      0.000    2.25e+05
2.54e+05
zipcode_98030  6138.2120   7420.942      0.827      0.408   -8407.648
2.07e+04
zipcode_98031  1.904e+04   7352.505      2.590      0.010    4630.856
3.35e+04
```

zipcode_98032	1.343e+04	9420.742	1.426	0.154	-5032.423	3.19e+04
zipcode_98033	3.259e+05	6946.511	46.914	0.000	3.12e+05	3.4e+05
zipcode_98034	1.924e+05	6329.075	30.403	0.000	1.8e+05	2.05e+05
zipcode_98038	3.808e+04	6374.554	5.974	0.000	2.56e+04	5.06e+04
zipcode_98039	6.533e+05	3.45e+04	18.938	0.000	5.86e+05	7.21e+05
zipcode_98040	4.401e+05	9275.468	47.452	0.000	4.22e+05	4.58e+05
zipcode_98042	1.654e+04	6449.312	2.565	0.010	3897.958	2.92e+04
zipcode_98045	9.826e+04	8771.005	11.203	0.000	8.11e+04	1.15e+05
zipcode_98052	2.562e+05	6441.635	39.777	0.000	2.44e+05	2.69e+05
zipcode_98053	2.497e+05	7924.596	31.508	0.000	2.34e+05	2.65e+05
zipcode_98055	5.255e+04	7470.196	7.035	0.000	3.79e+04	6.72e+04
zipcode_98056	1.204e+05	6723.038	17.906	0.000	1.07e+05	1.34e+05
zipcode_98058	4.178e+04	6661.999	6.272	0.000	2.87e+04	5.48e+04
zipcode_98059	9.057e+04	6761.772	13.394	0.000	7.73e+04	1.04e+05
zipcode_98065	1.368e+05	7640.870	17.898	0.000	1.22e+05	1.52e+05
zipcode_98070	8.72e+04	1.74e+04	5.009	0.000	5.31e+04	1.21e+05
zipcode_98072	1.537e+05	8855.588	17.355	0.000	1.36e+05	1.71e+05
zipcode_98074	2.065e+05	7135.626	28.933	0.000	1.92e+05	2.2e+05
zipcode_98075	2.322e+05	8918.569	26.034	0.000	2.15e+05	2.5e+05
zipcode_98077	1.529e+05	1.69e+04	9.066	0.000	1.2e+05	1.86e+05
zipcode_98092	-1.899e+04	7377.054	-2.575	0.010	-3.35e+04	-4532.805
zipcode_98102	4.472e+05	1.07e+04	41.757	0.000	4.26e+05	4.68e+05
zipcode_98103	3.677e+05	6345.370	57.943	0.000	3.55e+05	3.8e+05
zipcode_98105	4.209e+05	8354.953	50.372	0.000	4.04e+05	

```
4.37e+05
zipcode_98106  1.443e+05   7015.786     20.573      0.000    1.31e+05
1.58e+05
zipcode_98107  3.657e+05   7473.143     48.939      0.000    3.51e+05
3.8e+05
zipcode_98108  1.443e+05   8224.265     17.550      0.000    1.28e+05
1.6e+05
zipcode_98109  4.596e+05   1.07e+04     42.874      0.000    4.39e+05
4.81e+05
zipcode_98112  4.742e+05   8524.249     55.627      0.000    4.57e+05
4.91e+05
zipcode_98115  3.523e+05   6321.102     55.740      0.000     3.4e+05
3.65e+05
zipcode_98116  3.449e+05   7158.996     48.178      0.000    3.31e+05
3.59e+05
zipcode_98117  3.517e+05   6393.125     55.010      0.000    3.39e+05
3.64e+05
zipcode_98118  1.929e+05   6445.325     29.926      0.000     1.8e+05
2.06e+05
zipcode_98119   4.58e+05   8775.961     52.187      0.000    4.41e+05
4.75e+05
zipcode_98122  3.514e+05   7434.646     47.260      0.000    3.37e+05
3.66e+05
zipcode_98125  2.165e+05   6684.149     32.394      0.000    2.03e+05
2.3e+05
zipcode_98126  2.341e+05   6939.693     33.740      0.000    2.21e+05
2.48e+05
zipcode_98133  1.739e+05   6376.183     27.274      0.000    1.61e+05
1.86e+05
zipcode_98136  2.968e+05   7511.155     39.511      0.000    2.82e+05
3.11e+05
zipcode_98144  2.827e+05   7165.040     39.455      0.000    2.69e+05
2.97e+05
zipcode_98146  1.294e+05   7342.058     17.621      0.000    1.15e+05
1.44e+05
zipcode_98148  6.152e+04   1.25e+04      4.912      0.000     3.7e+04
8.61e+04
zipcode_98155  1.554e+05   6586.017     23.588      0.000    1.42e+05
1.68e+05
zipcode_98166  1.174e+05   7950.541     14.771      0.000    1.02e+05
1.33e+05
zipcode_98168  6.001e+04   7585.424      7.912      0.000    4.51e+04
7.49e+04
zipcode_98177  2.244e+05   7994.410     28.071      0.000    2.09e+05
2.4e+05
zipcode_98178  7.431e+04   7442.250      9.985      0.000    5.97e+04
8.89e+04
```

```
zipcode_98188   4.692e+04   9296.686      5.047      0.000     2.87e+04
6.51e+04
zipcode_98198   4.784e+04   7434.966      6.435      0.000     3.33e+04
6.24e+04
zipcode_98199   3.922e+05   7476.812     52.459      0.000     3.78e+05
4.07e+05
==============================================================================
Omnibus:                     1844.788   Durbin-Watson:                   2.004
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             6380.863
Skew:                           0.559   Prob(JB):                         0.00
Kurtosis:                       5.848   Cond. No.                     5.37e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 5.37e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

8 CONCLUSIONS & RECOMMENDATIONS

8.0.1 Key Takeaways

Our final model has an R2 value of 0.806, indicating that with the included variables, the model is capable of explaining 80.6% of the variability in a property's price.

As we can see in our three plots below, there does seem to be a strong linear relationship between price and our three selected variables: living space above ground, living space below ground and grade.

According to our model, for each foot of living space above ground that is increased, we see an increase in property value of approximately \$130.33. For each foot of living space below ground that is increased, we see an increase in property value of approximately \$90.98. Lastly, when the property grade is increased by 1 point, we see an increase in property value of approximately \$34,760.

An idea for future analysis would be to explore what costs would be involved in making these renovations, and to determine whether these recommendations would be cost-effective.

[50]:
```
# Define function to notate dollar amounts in thousands
def thousands(x, pos):
    """Source: https://stackoverflow.com/questions/61330427/
 ↪set-y-axis-in-millions"""
    'The two args are the value and tick position'
    return '%1.0fK' % (x * 1e-3)

formatter = FuncFormatter(thousands)
```

[51]:
```python
# Group data by sqft_above and find aggregate mean
df_sqftabove = df_unscaled.groupby('sqft_above').mean()
```

[52]:
```python
# Reset index before plotting
df_sqftabove.reset_index(inplace=True)
```

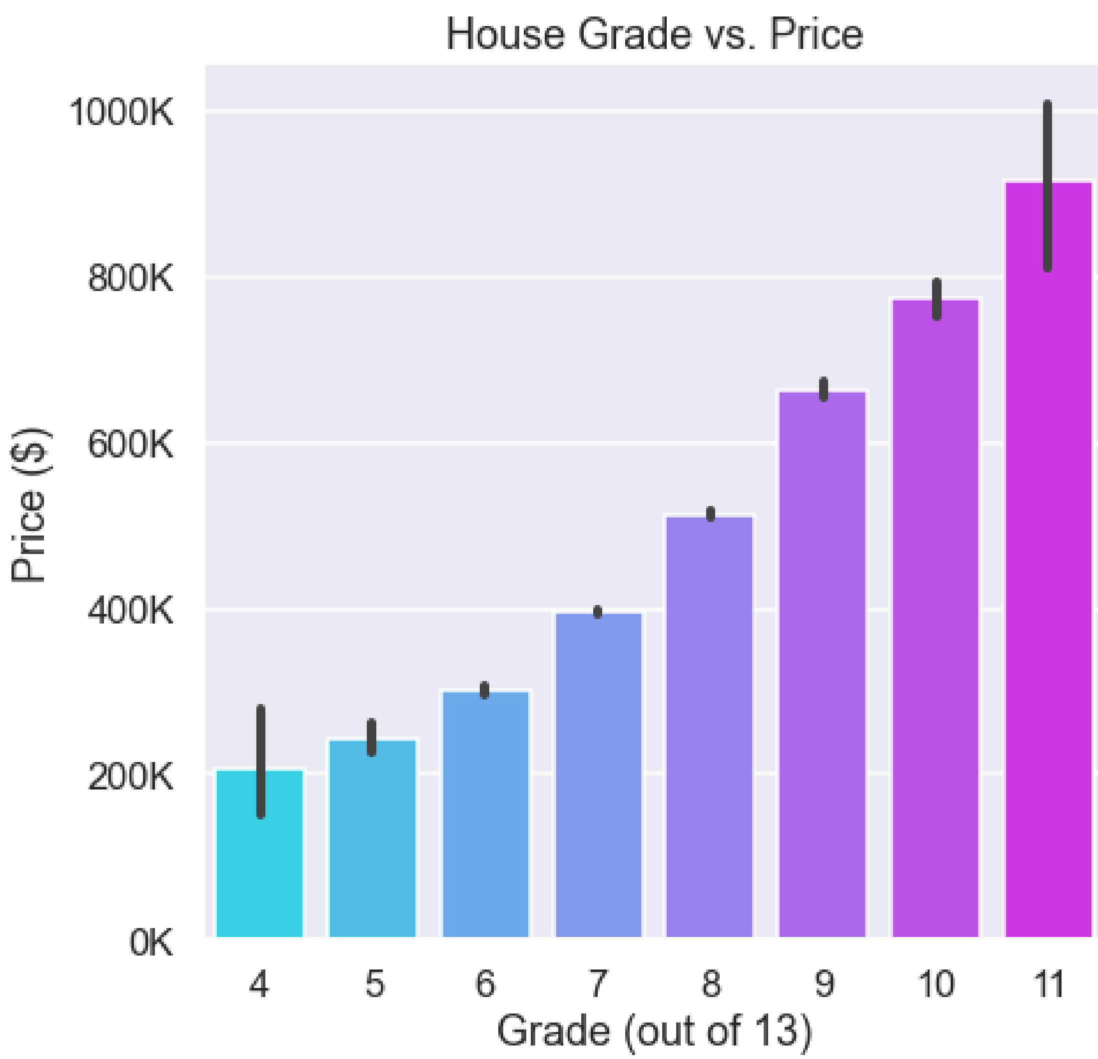
[53]:
```python
# Plot scatter graph for sqft_above vs price
fig, ax = plt.subplots(figsize=(7,7))

sns.regplot(data=df_sqftabove, x='sqft_above', y='price', marker='x', \
            line_kws={"color": "deepskyblue"}, \
            scatter_kws={"color": "dodgerblue"}, ax=ax)

ax.set_title('Living Space Above Ground vs. Price')
ax.set_xlabel('Square Footage Above Ground (ft)')
ax.set_ylabel('Price ($)')
ax.yaxis.set_major_formatter(formatter);
ax.set_ylim([150000, 800000]);
```

[54]:
```
# Group data points by sqft_basement and calculate aggregate mean
df_sqftbasement = df_unscaled.groupby('sqft_basement').mean()

df_sqftbasement.reset_index(inplace=True)
```

[55]:
```
# Plot scatter graph for sqft_below vs price
fig, ax = plt.subplots(figsize=(7,7))

sns.regplot(data=df_sqftbasement, x='sqft_basement', y='price', marker='x', \
            line_kws={"color": "deepskyblue"}, \
            scatter_kws={"color": "dodgerblue"}, ax=ax)

ax.set_title('Living Space Below Ground vs. Price')
ax.set_xlabel('Square Footage Below Ground (ft)')
```

```python
ax.set_ylabel('Price ($)')
ax.yaxis.set_major_formatter(formatter);
ax.set_ylim([300000, 700000]);
```

[56]:
```python
# Plot bar graph for grade vs price
fig, ax = plt.subplots(figsize=(7,7))

sns.barplot(data=df_unscaled, x='grade', y='price', palette='cool', ax=ax)

ax.set_title('House Grade vs. Price')
ax.set_xlabel('Grade (out of 13)')
ax.set_ylabel('Price ($)')
ax.yaxis.set_major_formatter(formatter);
```


House Grade vs. Price
Price ($)
1000K
800K
600K
400K
200K
0K
4
5
6
7
8
9
10
11
Grade (out of 13)

```
[ ]:
```