Table of Contents

1 Business Understanding
2 Data Understanding
3 Data Preparation and Preprocessing
  3.1 Null Values and Duplicates
  3.2 Data Types
  3.3 Outliers
    3.3.1 Price
    3.3.2 Bedrooms
  3.4 Relevancy
    3.4.1 Checking For Linearity
    3.4.2 Heatmap
    3.4.3 VIF
4 Train-Test Split, Encoding, and Scaling
    4.0.1 tt_split
    4.0.2 scaling
    4.0.3 encoding
5 Modeling
  5.1 Baseline
  5.2 Iterations
    5.2.1 1
    5.2.2 2
    5.2.3 3
    5.2.4 4
    5.2.5 5
    5.2.6 6
    5.2.7 7
    5.2.8 8
    5.2.9 9
    5.2.10 10
    5.2.11 11
    5.2.12 12
    5.2.13 13
6 Conclusions

1 Business Understanding

I have been hired by a real estate development company in King County, Washington, to model real estate prices based on the King County Housing dataset. My intention is to build a model that will accurately describe housing prices based on the features contained in this data set, and offer advice on the best improvements that can be made to increase house value. Additionally, given the robust location data in these records, I will make suggestions about the best areas to look for investment houses. This model should ideally be able to assist the real estate company in making informed decisions about what type and where to build new houses.

2 Data Understanding

The dataset contains records of all recorded house sales in Kings County, WA from May 2014 to May 2015. The data comes directly from the King County assessors office. It is a robust set with 21597 entries to work with. There are 21 columns in this dataset:

- 'price' is the sale price of the house and will be our model's target feature.
- The majority of the columns are already formatted in a way that is easy to work with, however, categorical values will need to be encoded for purposes of this regression model.
- The continuous values may need to be scaled. I will make this determination during the exploratory stage of this process.

Information about the terms and data contained in this set can be examined at https://info.kingcounty.gov/assessor/esales/Glossary.aspx?type=r (https://info.kingcounty.gov/assessor/esales/Glossary.aspx?type=r).

3 Data Preparation and Preprocessing

I will start here by looking at null values, data types, duplicates, outliers, and relevancy. Then decide which data is continuous or categorical, and make the appropriate transformations. Finally I will examine the relationship between variables and check for multicollinearity.

In [50]:

```
# importing relevent libraries for scrubbing and EDA
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
%matplotlib inline
sns.set_context('talk')

from scipy import stats

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2
from sklearn.feature_selection import RFE

from statsmodels.regression import linear_model
from statsmodels.formula.api import ols
from statsmodels.stats.outliers_influence import variance_inflation_fac

# suppressing scientific notation
# adapted from https://re-thought.com/how-to-suppress-scientific-notati
pd.options.display.float_format = '{:.2f}'.format

df = pd.read_csv('data/kc_house_data.csv')
```

In [51]:
```
def prelim(df):
    """displays core information
    on a dataframe at the beginning of EDA

    accepts a DataFrame as input, and displays
    the head, info, and sum of all null values
    for each column in that DataFrame
    """
    return (display(df.head()),
            display(df.info()),
            display(df.isna().sum()))

prelim(df)
```

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront
0	7129300520	10/13/2014	221900.00	3	1.00	1180	5650	1.00	nan
1	6414100192	12/9/2014	538000.00	3	2.25	2570	7242	2.00	0.00
2	5631500400	2/25/2015	180000.00	2	1.00	770	10000	1.00	0.00
3	2487200875	12/9/2014	604000.00	4	3.00	1960	5000	1.00	0.00
4	1954400510	2/18/2015	510000.00	3	2.00	1680	8080	1.00	0.00

5 rows × 21 columns

```
<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
```

```
None

id                 0
date               0
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
```

Out[51]:
```
(None, None, None)
```

3.1 Null Values and Duplicates

In [53]:
```
print('Waterfront value counts:')
print(df['waterfront'].value_counts())
print('--------------')
print('View value counts:')
print(df['view'].value_counts())
print('--------------')
print('yr_renovated value counts:')
print(df['yr_renovated'].value_counts())
print('--------------')
```

```
Waterfront value counts:
0.00    19075
1.00      146
Name: waterfront, dtype: int64
--------------
View value counts:
0.00    19422
2.00      957
3.00      508
1.00      330
4.00      317
Name: view, dtype: int64
--------------
yr_renovated value counts:
0.00       17011
2014.00       73
2003.00       31
2013.00       31
2007.00       30
           ...
1946.00        1
1959.00        1
1971.00        1
1951.00        1
1954.00        1
Name: yr_renovated, Length: 70, dtype: int64
--------------
```

'waterfront' is a binary, categorical variable that is missing 2376 entries. A value of 1 for 'waterfront' means the house has a waterfront view, whereas a value of 0 means the house has no waterfront view. With more time I would try and impute a value to each null entry based on location, either using the **'zipcode'** column or plotting the houses using the **'lat'** and **'long'** variables. However, for the sake of simplicity and clarity, I will assign the null entries a 1 or 0 based on the proportion of waterfront views in the data set.

In [54]:
```
# calculating expected number of additional waterfront view houses
waterfront_ratio = 146/19075

# imputing null values according to ratio of expected waterfront view h
# note the minimal impact on model output despite not being able to set
# random seed for the .choice method
df['waterfront'] = df['waterfront'].fillna(
    pd.Series(np.random.choice([0, 1], p=[1-0.008, .008], size=len(df))
```

'view', also a categorical, missing 63 entries, is a count of the number of times that a house has been viewed. The vast majority of the entries for this feature are 0, which I will use to impute as a value to the NaNs.

In [55]:
```
# imputing NaNs with value of 0
df['view'] = df['view'].fillna(0)
```

'yr_renovated' is a (semi-)continuous variable. The overwhelming majority of entries for this variable are 0.00, suggesting that most houses have never been renovated, thereby obviating any usefulness this column may have as a predictor. An alternative option would be to turn this variable into a binary, with a value of 0 for never renovated, and a value of 1 for 'has been renovated' independent of time.

In [56]:
```
# creating binary from categorical
def is_renovated(value):
    if value == 0:
        return 0
    else:
        return 1

df['is_renovated'] = df['yr_renovated'].apply(is_renovated)
df = df.drop('yr_renovated', axis=1)
```

In [57]:
```
# sanity check that NaNs have been removed
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     21597 non-null  float64
 9   view           21597 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  zipcode        21597 non-null  int64  
 16  lat            21597 non-null  float64
 17  long           21597 non-null  float64
 18  sqft_living15  21597 non-null  int64  
 19  sqft_lot15     21597 non-null  int64  
 20  is_renovated   21597 non-null  int64  
dtypes: float64(7), int64(12), object(2)
memory usage: 3.5+ MB
```

In [58]:
```
# investigating duplicate entries
dupes = df.loc[df['id'].duplicated(keep=False)]
dupes
```

Out[58]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	wate
93	6021501535	7/25/2014	430000.00	3	1.50	1580	5000	1.00	
94	6021501535	12/23/2014	700000.00	3	1.50	1580	5000	1.00	
313	4139480200	6/18/2014	1380000.00	4	3.25	4290	12103	1.00	
314	4139480200	12/9/2014	1400000.00	4	3.25	4290	12103	1.00	
324	7520000520	9/5/2014	232000.00	2	1.00	1240	12092	1.00	
...	...	...	...	...	...	...	...	...	
20654	8564860270	3/30/2015	502000.00	4	2.50	2680	5539	2.00	
20763	6300000226	6/26/2014	240000.00	4	1.00	1200	2171	1.50	
20764	6300000226	5/4/2015	380000.00	4	1.00	1200	2171	1.50	
21564	7853420110	10/3/2014	594866.00	3	3.00	2780	6000	2.00	
21565	7853420110	5/4/2015	625000.00	3	3.00	2780	6000	2.00	

353 rows × 21 columns

There are a number of duplicate entries in the **'id'** column, which would suggest some cleaning is in order, but after looking at the corresponding price for each entry, it is clear that there were some houses that were sold multiple times in the year that the data was collected.

A side quest analyzing the houses that were flipped multiple times could be interesting, but is beyond the scope of business problem. The absence of data in the **'yr_renovated'** column would frustrate this analysis, as I would want to know what sort of improvements were made to the properties between the sales.

3.2 Data Types

The majority of the columns are of the expected data type. We could recast **'date'** as a pandas date-time type, but it is not immediately clear that doing so would benefit our analysis. Alternatively I can use **'date'** and **'yr_built'** to engineer an **'age'** feature.

In [59]:
```
# isolating the year and recasting as int.
df['date'] = [x.strip()[-4:] for x in df['date']]
df['date'] = df['date'].astype(int)

# engineering 'age' feature as difference between 'date' and 'yr_built'
df['age'] = df['date'] - df['yr_built']
df = df.drop(['date', 'yr_built'], axis=1)
df['age']
```

Out[59]:
```
0        59
1        63
2        82
3        49
4        28
         ..
21592     5
21593     1
21594     5
21595    11
21596     6
Name: age, Length: 21597, dtype: int64
```

'sqft_basement' is another object data type that should be recast as an int, however, it appears that the variable **'sqft_living'** is the sum of **'sqft_basement'** and **'sqft_above'**. If this is the case, we will most likely want to eliminate one or two of those columns to reduce the impact of multicollinearity.

The rest of the columns are in a data type suitable to perform a regression model.

In [60]:
```
# recast 'sqft_basement' as int
df['sqft_basement'] = df['sqft_basement'].replace('?', 0.0)
df['sqft_basement'] = df['sqft_basement'].astype(str).astype(float).ast
df['sqft_basement'].head()
```

Out[60]:
```
0      0
1    400
2      0
3    910
4      0
Name: sqft_basement, dtype: int64
```

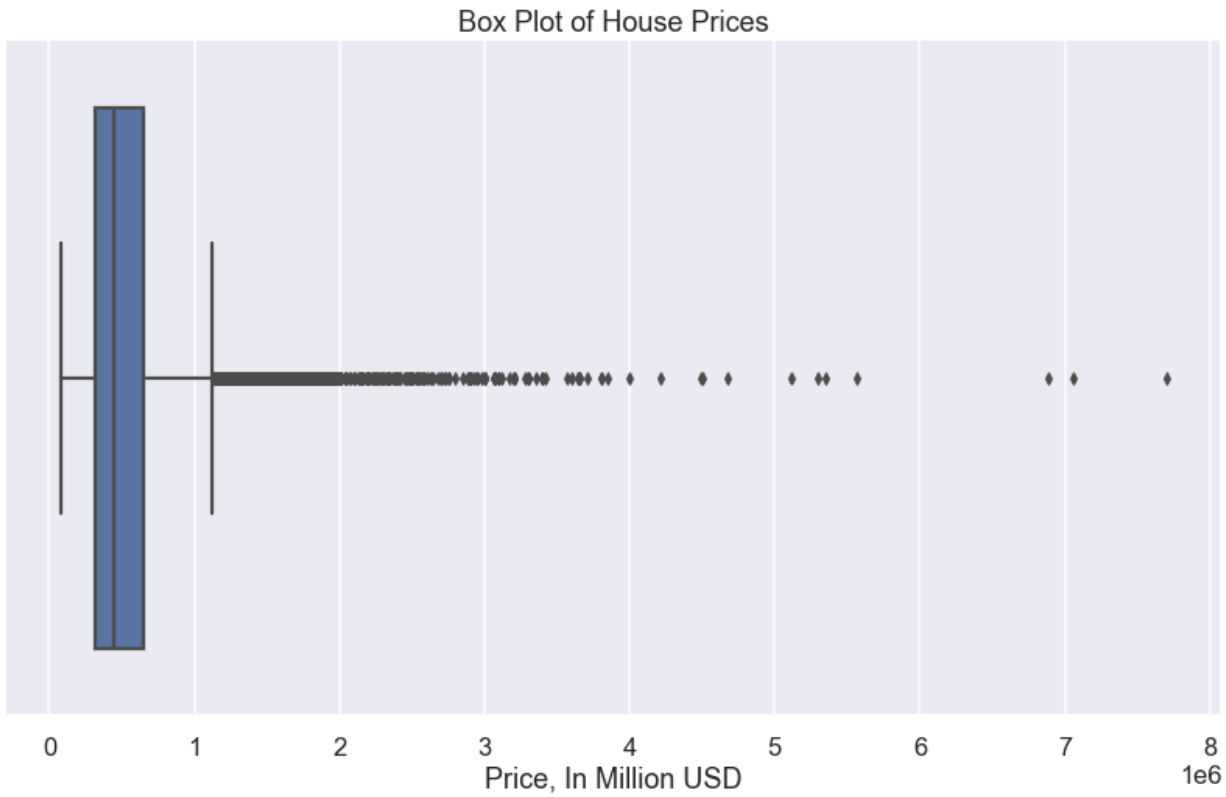
3.3 Outliers

3.3.1 Price

In [61]:
```
df['price'].describe()
```

Out[61]:
```
count     21597.00
mean     540296.57
std      367368.14
min       78000.00
25%      322000.00
50%      450000.00
75%      645000.00
max     7700000.00
Name: price, dtype: float64
```

In [62]:
```
fig, ax = plt.subplots(figsize=(14,8))
ax = sns.boxplot(x=df['price'])
ax.set_title('Box Plot of House Prices')
ax.set_xlabel('Price, In Million USD');
```



As we can see, the **'price'** column has a lot of outliers, defined above using the IQR method. In order to preserve more data, I will remove the values that lie more than 3 standard deviations away from the mean, using z-score to identify them.

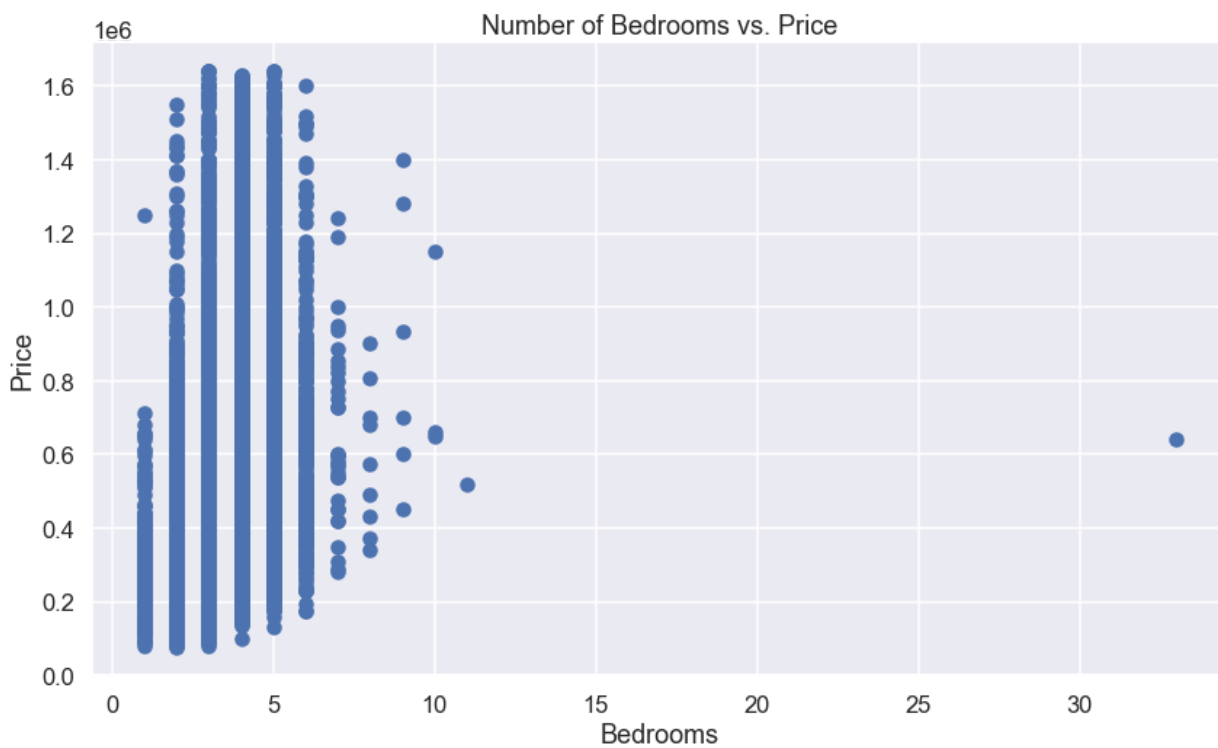
In [63]:
```
# identifying outliers based on z-score and removing.
df['z_score'] = np.abs(stats.zscore(df['price']))
df = (df.loc[df['z_score'] < 3])
df = df.drop('z_score', axis=1)

# checking new aggregate values
df['price'].describe()
```

Out[63]:
```
count     21191.00
mean     507010.29
std      259462.21
min       78000.00
25%      320000.00
50%      447000.00
75%      627650.00
max     1640000.00
Name: price, dtype: float64
```

3.3.2 Bedrooms

'bedrooms' also has a pretty obvious outlier we should examine.

In [64]:
```
plt.figure(figsize=(14,8))
plt.scatter(df['bedrooms'], df['price'])
plt.xlabel('Bedrooms')
plt.ylabel('Price')
plt.title('Number of Bedrooms vs. Price');
```



In [65]:
```
# investigating outlier
df.loc[df['bedrooms'] > 15]
```

Out[65]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view
15856	2402100895	640000.00	33	1.75	1620	6000	1.00	0.00	0.00

Given the sale price this house, and other variables (only 1.75 bathrooms for 33 bedrooms?) it is very likely that this is an entry error. Let's change this entry to the more likely 3 bedrooms the data reflects.

In [66]:
```
# replacing entry error
df['bedrooms'].replace({33 : 3}, inplace=True)
```

In [67]:
```
df.shape
```

Out[67]: (21191, 20)

After removing outliers from our data set, we have culled 406 entries.

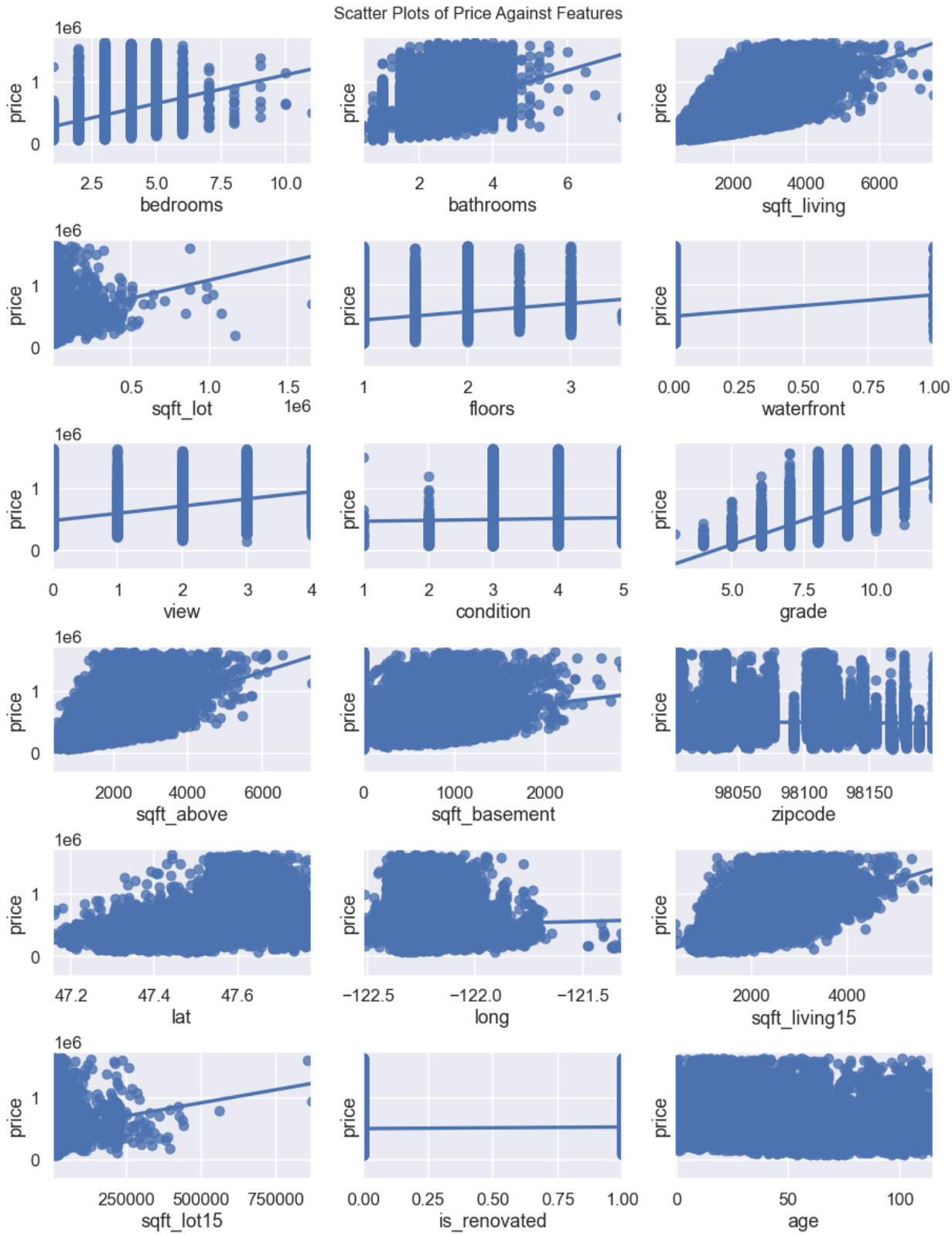
3.4 Relevancy

Deciding which columns are relevant to a regression model is subjective. Intuitively, a column like **'id'**, which is a unique identifier of a particular house will not have any sort of linear relationship to house price, and can be eliminated.

3.4.1 Checking For Linearity

In [68]:

```
fig,ax = plt.subplots(figsize=(15,20), ncols=3, nrows=6, sharey=True)
fig.suptitle('Scatter Plots of Price Against Features', fontsize=16, y=
cols = list(df.drop(['price','id'],axis=1).columns)
plt.subplots_adjust(hspace = 0.6)
for i, col in enumerate(cols):
    sns.regplot(x=col, y='price', ci=None, data=df, ax=ax[i//3][i%3])
```



A cursory examination of scatter plots of the independent variables illustrates which variables we will want to treat as **categorical** from their distinctive distributions. We can also deduce which features have an apparent linear relationship with price, and which features have a more 'blobish', less defined relationship.

Looking at the scatter plot of **'sqft_lot'** it is hard to imagine a linear relationship between the variables. We will keep it in our regression for the time being, but this is a strong candidate for removal.

'sqft_basement' is complicated by the presence of so many entries with 0. Whether or not so many houses really have no basement, or if 0 is being used as a placeholder value is unknown. Coupled with the correlation between this and **'sqft_living'** and **'sqft_above'**, this variable is ripe for dropping.

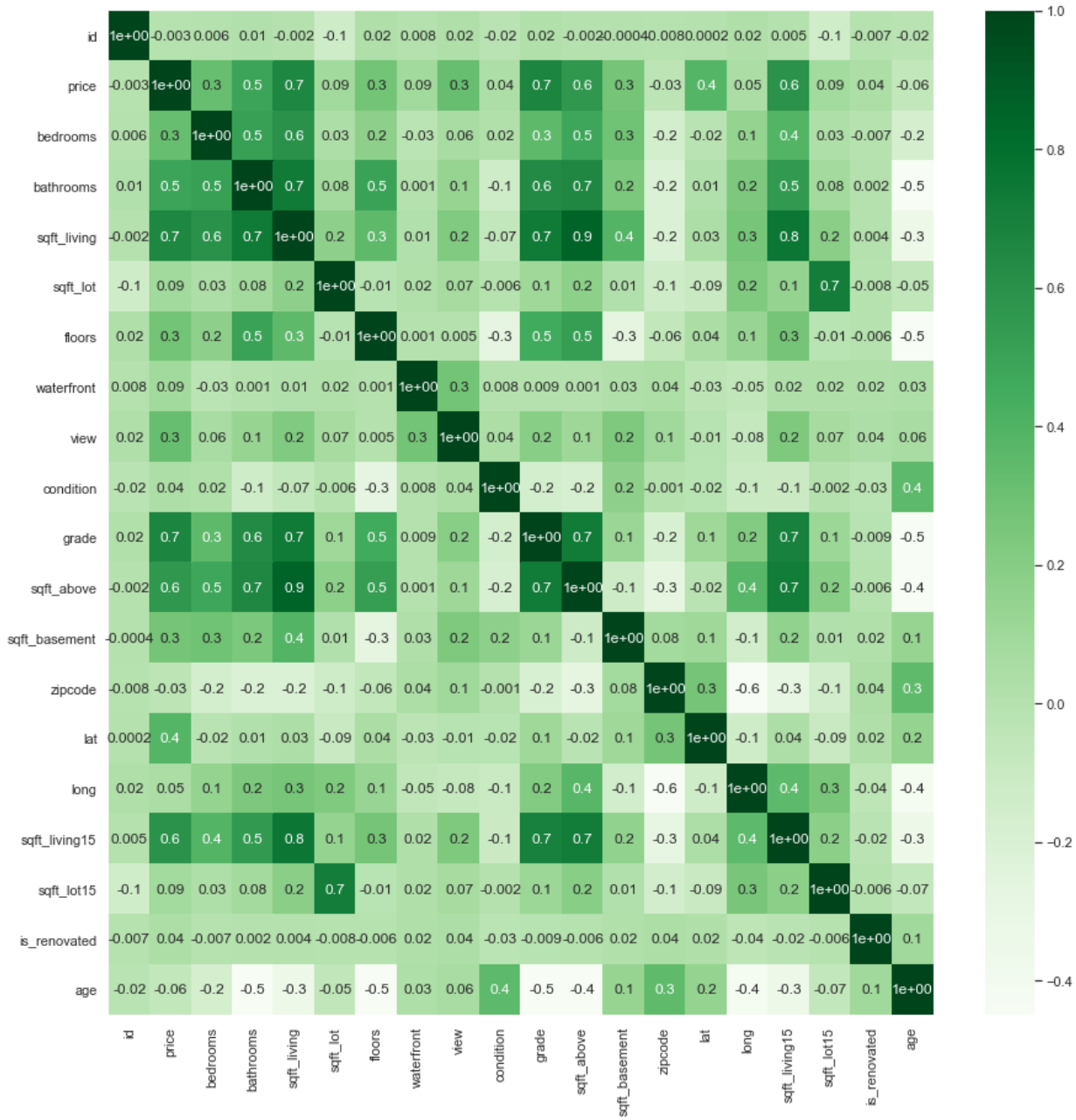
Unsurprisingly, given its similarity to **'sqft_lot'**, **'sqft_lot15'** lacks a clear linear relationship with **'price'**. It's on the chopping block.

3.4.2 Heatmap

Creating a heat map of the variables will help us decide which features can be discarded, and give us an inking of which features may interfere with our modeling due to multicollinearity. I will also run a **variance inflation factor** to test for multicollinearity.

In [69]:

```
# plot a heatmap using seaborn
sns.set(font_scale=1)
fig, ax = plt.subplots(figsize=(15,15))
ax = sns.heatmap(data=df.corr(), annot=True, robust=True, fmt='.1', cma
plt.tight_layout;
```



Right off the bat, we can see that **'price'** most strongly correlates, intuitively and perhaps

unsurprisingly with **'sqft_living'**, and **'grade'**.

Other significant correlations include:

'bathrooms' and **'sqft_living'**

'sqft_living' and **'sqft_above'**

'sqft_living' and **'grade'**

'grade' and **'sqft_above'**

The relationships between **'sqft_above'**, **'sqft_living'**, and **'sqft_basement'** suggest some strong collinearity and require further investigation. The relationship between **'sqft_living15'** and **'sqft_lot15'** also will be looked at closely.

3.4.3 VIF

In [70]:

```
vif_dct = {}

for i in range(len(df.columns)):
    vif = variance_inflation_factor(df.values, i).round(2)
    v = df.columns[i]
    vif_dct[v] = vif

vif_dct
```

Out[70]:

```
{'id': 3.65,
 'price': 3.48,
 'bedrooms': 1.73,
 'bathrooms': 3.16,
 'sqft_living': 127.6,
 'sqft_lot': 2.11,
 'floors': 2.02,
 'waterfront': 1.1,
 'view': 1.31,
 'condition': 1.21,
 'grade': 3.62,
 'sqft_above': 108.15,
 'sqft_basement': 31.86,
 'zipcode': 0.49,
 'lat': 1.51,
 'long': 1.84,
 'sqft_living15': 2.9,
 'sqft_lot15': 2.15,
 'is_renovated': 1.02,
 'age': 2.29}
```

Generally we are looking really good here. With very few VIF values over 10, we can eliminate the multicollinearity and move forward with our processing.

In [71]:
```
df = df.drop(['sqft_basement'], axis=1)

vif_dct = {}

for i in range(len(df.columns)):
    vif = variance_inflation_factor(df.values, i).round(3)
    v = df.columns[i]
    vif_dct[v] = vif

vif_dct
```

Out[71]:
```
{'id': 3.645,
 'price': 3.475,
 'bedrooms': 1.73,
 'bathrooms': 3.164,
 'sqft_living': 8.088,
 'sqft_lot': 2.105,
 'floors': 2.015,
 'waterfront': 1.099,
 'view': 1.305,
 'condition': 1.207,
 'grade': 3.621,
 'sqft_above': 6.314,
 'zipcode': 0.489,
 'lat': 1.505,
 'long': 1.842,
 'sqft_living15': 2.902,
 'sqft_lot15': 2.146,
 'is_renovated': 1.021,
 'age': 2.286}
```

By dropping the **'sqft_basement'** column we have eliminated any multicollinearity flags. Important to note the marginally higher values of **'sqft_living'** and **'sqft_above'**. If we need to minimize multicollinearity later, these would be two candidates for dropping.

Let's go ahead and eliminate irrelevant variables (**'id', 'lat', 'long'**) and prepare our data for modeling via a trait-test split and encoding and scaling our training data.

4 Train-Test Split, Encoding, and Scaling

In [72]:
```
continuous = ['bedrooms', 'bathrooms', 'floors',
              'condition', 'grade', 'sqft_living',
              'sqft_lot', 'sqft_above',
              'sqft_living15', 'sqft_lot15', 'age']
categorical = [ 'waterfront', 'view', 'zipcode', 'is_renovated']
```

A note on categorical vs. continuous:

Variables that have no relation to a linear value and are non-ordinal (ie. **'zipcode'**) and those that are binary values, I have treated as categorical.

Variables that might be considered categorical, but that I have chosen to keep as continuous are generally ordinal and whose order has an implicit value associated with it (ie. **'grade.'**)

In [73]:
```
# eliminating variables with no linearity
preprocessed = df.drop(['id', 'lat', 'long'], axis=1)

# setting X and y for train_test_split and regression modeling
y = preprocessed['price']
X = preprocessed.drop('price', axis=1)
```

4.0.1 tt_split

In [74]:
```
# creating the tt split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=

# checking shapes of split variables
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

Out[74]:
```
((15893, 15), (15893,), (5298, 15), (5298,))
```

4.0.2 scaling

As there is no uniform scale between the variables, it makes sense, for interpretability to scale our variables. This will add a layer of complexity to the model, but is unavoidable.

In [75]:
```
# scaling the X_train continuous data
scaler = StandardScaler()
X_train_scaled = pd.DataFrame(scaler.fit_transform(X_train[continuous])
                              columns=X_train[continuous].columns)

# only transform the test data
X_test_scaled = pd.DataFrame(scaler.transform(X_test[continuous]),
                              columns=X_test[continuous].columns)
```

4.0.3 encoding

In [76]:

```
# instantiating ohe
ohe = OneHotEncoder(categories='auto', drop='first', sparse=False)

# fit_transforming categorical columns of X_train
feature_arr_train = ohe.fit_transform(X_train[categorical])

# get feature names for assigning to columns
feature_labels = ohe.get_feature_names(categorical)
feature_labels = np.array(feature_labels).ravel()
features_train = pd.DataFrame(feature_arr_train, columns=feature_labels

# transforming categorical columns of X_test
feature_arr_test = ohe.transform(X_test[categorical])

# get feature names for assigning to columns
feature_labels = ohe.get_feature_names(categorical)
feature_labels = np.array(feature_labels).ravel()
features_test = pd.DataFrame(feature_arr_test, columns=feature_labels)
```

5 Modeling

5.1 Baseline

In [77]:

```
# merging processed continuous values with encoded categories
X_train = X_train_scaled.merge(features_train, left_index=True, right_i
X_test = X_test_scaled.merge(features_test, left_index=True, right_inde

# adding constant
X_train_int = sm.add_constant(X_train)
X_test_int = sm.add_constant(X_test)

# resetting indicies to ensure alignment
X_train_int.reset_index(drop=True, inplace=True)
y_train.reset_index(drop=True, inplace=True)

# fitting model
model = sm.OLS(y_train ,X_train_int).fit()
model.summary2() #.summary2() supresses scientific notation and increas
```

Out[77]:

Model:	OLS	Adj. R-squared:	0.833
Dependent Variable:	price	AIC:	412937.7822
Date:	2021-09-29 09:07	BIC:	413605.3883
No. Observations:	15893	Log-Likelihood:	-2.0638e+05
Df Model:	86	F-statistic:	920.8
Df Residuals:	15806	Prob (F-statistic):	0.00
R-squared:	0.834	Scale:	1.1198e+10

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
const	319455.0005	6511.8920	49.0572	0.0000	306690.9494	332219.0517
bedrooms	-8249.8061	1120.1671	-7.3648	0.0000	-10445.4613	-6054.1508
bathrooms	12812.3838	1495.0439	8.5699	0.0000	9881.9273	15742.8403
floors	-11248.3649	1320.2299	-8.5200	0.0000	-13836.1660	-8660.5637
condition	13723.9051	950.3962	14.4402	0.0000	11861.0201	15586.7902
grade	60485.0029	1570.3064	38.5180	0.0000	57407.0233	63562.9825
sqft_living	64722.5732	2369.3876	27.3162	0.0000	60078.3031	69366.8432
sqft_lot	9615.2342	1248.3643	7.7023	0.0000	7168.2978	12062.1706
sqft_above	42838.0057	2217.1787	19.3210	0.0000	38492.0824	47183.9290
sqft_living15	21271.2920	1522.3551	13.9726	0.0000	18287.3024	24255.2817
sqft_lot15	-105.4847	1295.3305	-0.0814	0.9351	-2644.4803	2433.5109
age	20362.5129	1409.1733	14.4500	0.0000	17600.3724	23124.6533
waterfront_1.0	205790.8084	14141.5085	14.5523	0.0000	178071.8384	233509.7784
view_1.0	63323.0947	7050.1521	8.9818	0.0000	49503.9924	77142.1970
view_2.0	62404.2141	4388.4474	14.2201	0.0000	53802.3564	71006.0717
view_3.0	127128.3083	6219.0020	20.4419	0.0000	114938.3549	139318.2617
view_4.0	230691.6236	9573.2627	24.0975	0.0000	211926.9365	249456.3106

zipcode_98002	12444.3226	10905.8190	1.1411	0.2539	-8932.3267	33820.9719
zipcode_98003	-10001.3927	9739.1702	-1.0269	0.3045	-29091.2774	9088.4920
zipcode_98004	603682.2977	10504.9365	57.4665	0.0000	583091.4237	624273.1716
zipcode_98005	318908.2372	11568.4949	27.5670	0.0000	296232.6673	341583.8070
zipcode_98006	272810.7218	8709.5494	31.3232	0.0000	255739.0114	289882.4322
zipcode_98007	249390.8169	12184.7599	20.4674	0.0000	225507.2974	273274.3364
zipcode_98008	244964.8030	9720.9596	25.1997	0.0000	225910.6131	264018.9929
zipcode_98010	87443.4413	13544.6401	6.4559	0.0000	60894.4015	113992.4811
zipcode_98011	132941.1050	10851.6925	12.2507	0.0000	111670.5497	154211.6603
zipcode_98014	113213.9988	13104.2564	8.6395	0.0000	87528.1613	138899.8362
zipcode_98019	100947.5812	11056.8026	9.1299	0.0000	79274.9866	122620.1758
zipcode_98022	-4695.7084	10414.1093	-0.4509	0.6521	-25108.5506	15717.1338
zipcode_98023	-27017.7730	8521.6824	-3.1705	0.0015	-43721.2426	-10314.3034
zipcode_98024	160445.3575	15423.1699	10.4029	0.0000	130214.1850	190676.5300
zipcode_98027	178955.9933	8938.2847	20.0213	0.0000	161435.9355	196476.0510
zipcode_98028	129664.4187	9570.4655	13.5484	0.0000	110905.2146	148423.6229
zipcode_98029	215675.7186	9363.9903	23.0325	0.0000	197321.2294	234030.2078
zipcode_98030	310.8996	10027.8952	0.0310	0.9753	-19344.9190	19966.7183
zipcode_98031	5204.4947	9894.5332	0.5260	0.5989	-14189.9192	24598.9086
zipcode_98032	570.9227	12697.7009	0.0450	0.9641	-24318.0196	25459.8650
zipcode_98033	353961.0226	8895.6078	39.7905	0.0000	336524.6164	371397.4288
zipcode_98034	187730.7050	8337.8827	22.5154	0.0000	171387.5036	204073.9063
zipcode_98038	32201.3861	8193.3106	3.9302	0.0001	16141.5625	48261.2096
zipcode_98039	766051.7424	32634.0873	23.4740	0.0000	702085.2083	830018.2765
zipcode_98040	478923.9152	10313.1278	46.4383	0.0000	458709.0082	499138.8223
zipcode_98042	4511.2292	8347.2404	0.5404	0.5889	-11850.3142	20872.7726
zipcode_98045	94981.0748	10288.8303	9.2315	0.0000	74813.7937	115148.3559
zipcode_98052	239599.7283	8270.5569	28.9702	0.0000	223388.4933	255810.9633
zipcode_98053	223060.2236	9055.7774	24.6318	0.0000	205309.8669	240810.5803
zipcode_98055	43517.6414	9921.4316	4.3862	0.0000	24070.5035	62964.7792
zipcode_98056	101500.7067	8904.3903	11.3990	0.0000	84047.0860	118954.3275
zipcode_98058	25403.4974	8636.1815	2.9415	0.0033	8475.5965	42331.3983
zipcode_98059	94853.8989	8627.5651	10.9943	0.0000	77942.8871	111764.9108
zipcode_98065	106887.3468	9392.5924	11.3800	0.0000	88476.7942	125297.8995
zipcode_98070	96294.2755	13904.5065	6.9254	0.0000	69039.8564	123548.6946
zipcode_98072	171719.5871	9830.6045	17.4679	0.0000	152450.4807	190988.6935

zipcode_98074	191675.0644	8916.7653	21.4960	0.0000	174197.1873	209152.9416
zipcode_98075	204027.1191	9513.8181	21.4453	0.0000	185378.9503	222675.2878
zipcode_98077	154458.0592	11245.6033	13.7350	0.0000	132415.3938	176500.7246
zipcode_98092	-35213.5198	9372.8669	-3.7570	0.0002	-53585.4082	-16841.6314
zipcode_98102	429196.1907	14453.7883	29.6944	0.0000	400865.1167	457527.2646
zipcode_98103	322763.2467	8393.6959	38.4531	0.0000	306310.6452	339215.8483
zipcode_98105	438444.9005	10743.5748	40.8100	0.0000	417386.2682	459503.5328
zipcode_98106	129651.2022	9322.6762	13.9071	0.0000	111377.6933	147924.7110
zipcode_98107	319875.2983	10091.4641	31.6976	0.0000	300094.8773	339655.7193
zipcode_98108	117002.6698	11186.6458	10.4591	0.0000	95075.5677	138929.7718
zipcode_98109	456577.1508	14203.9216	32.1444	0.0000	428735.8441	484418.4576
zipcode_98112	500736.9509	10762.9959	46.5239	0.0000	479640.2511	521833.6508
zipcode_98115	327575.7968	8372.7057	39.1242	0.0000	311164.3383	343987.2552
zipcode_98116	291387.6959	9533.0582	30.5660	0.0000	272701.8142	310073.5775
zipcode_98117	308748.7625	8498.1730	36.3312	0.0000	292091.3740	325406.1511
zipcode_98118	168111.4241	8587.8601	19.5755	0.0000	151278.2385	184944.6097
zipcode_98119	441108.1108	11643.3616	37.8849	0.0000	418285.7939	463930.4278
zipcode_98122	316834.7993	9978.9916	31.7502	0.0000	297274.8373	336394.7614
zipcode_98125	199843.2295	8952.7055	22.3221	0.0000	182294.9053	217391.5537
zipcode_98126	189810.7087	9222.2897	20.5817	0.0000	171733.9688	207887.4486
zipcode_98133	155106.0136	8503.3388	18.2406	0.0000	138438.4994	171773.5278
zipcode_98136	247015.8227	9983.4988	24.7424	0.0000	227447.0261	266584.6193
zipcode_98144	245505.6993	9463.6331	25.9420	0.0000	226955.8988	264055.4998
zipcode_98146	113396.1932	9695.0866	11.6963	0.0000	94392.7175	132399.6689
zipcode_98148	67415.1800	19279.5343	3.4967	0.0005	29625.0933	105205.2666
zipcode_98155	137552.9566	8802.4451	15.6267	0.0000	120299.1600	154806.7532
zipcode_98166	88158.1018	10110.5045	8.7195	0.0000	68340.3595	107975.8441
zipcode_98168	58838.7974	9924.5824	5.9286	0.0000	39385.4836	78292.1112
zipcode_98177	216685.7159	10153.6522	21.3407	0.0000	196783.3992	236588.0326
zipcode_98178	53945.2561	9970.6194	5.4104	0.0000	34401.7046	73488.8076
zipcode_98188	36605.2158	12413.1670	2.9489	0.0032	12273.9924	60936.4391
zipcode_98198	15586.8004	9769.5289	1.5955	0.1106	-3562.5908	34736.1915
zipcode_98199	376370.6965	9701.9625	38.7933	0.0000	357353.7432	395387.6498
is_renovated_1	8138.3204	2091.5129	3.8911	0.0001	4038.7165	12237.9243

Omnibus: 3369.470 Durbin-Watson: 1.994

Prob(Omnibus):	0.000	Jarque-Bera (JB):	19097.634
Skew:	0.902	Prob(JB):	0.000
Kurtosis:	8.058	Condition No.:	145

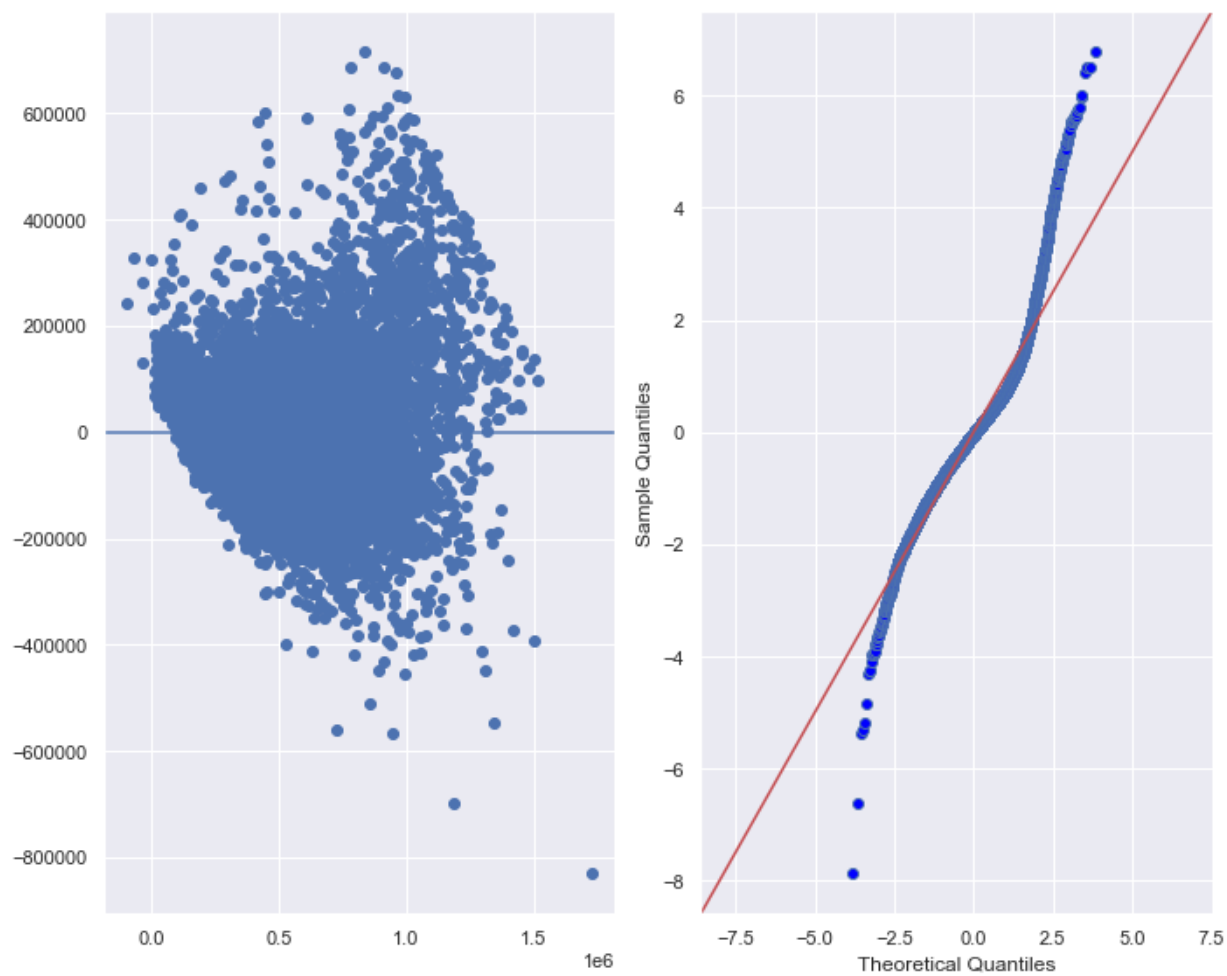
Type *Markdown* and LaTeX: α^2

In [78]:

```
def model_diagnostics(model, X_train, y_train):
    '''
    Expects as inputs OLS results, and the X and y data fed to the mode
    Returns a scatter plot and QQ plot of residuals'''

    fig, axes = plt.subplots(ncols=2, figsize=(10,8))
    # predicting our 'y hat' values
    model_preds = model.predict(X_train)
    # obtaining residuals
    residuals = y_train - model_preds
    # plotting
    axes[0].scatter(model_preds, residuals)
    axes[0].axhline()
    sm.graphics.qqplot(residuals, dist=stats.norm, line='45', fit=True,
    plt.tight_layout();
```

In [79]:

```
model_diagnostics(model, X_train_int, y_train)
```



The homoscedasticity of the residuals are acceptable, with no obvious red flags. The QQ-plot, plotting residuals against a normal distribution, is a little problematic. If we are unable to verify the assumptions necessary to legitimize our regression model, we will have to approach the results with skepticism.

In [86]:
```
def train_test_validation(X_train, y_train, X_test):
    linreg = LinearRegression()
    linreg.fit(X_train, y_train)

    # checking predictions from test against train
    y_train_pred = linreg.predict(X_train)
    y_test_pred = linreg.predict(X_test)

    metrics_dct = {}

    for split, y_pred in [('train', y_train_pred), ('test', y_test_pred
        y_true = y_train.copy() if split == 'train' else y_test.copy()

        r2 = r2_score(y_true, y_pred).round(3)
        mse = mean_squared_error(y_true, y_pred).round(3)
        mae = mean_absolute_error(y_true, y_pred).round(3)
        split_dct = {
            'r2': r2,
            'mse': mse,
            'mae': mae,
            'rmse': np.sqrt(mse).round(3)
        }
        metrics_dct[split] = split_dct
    return metrics_dct
```

In [87]:
```
train_test_validation(X_train_int, y_train, X_test_int)
```

Out[87]:
```
{'train': {'r2': 0.834,
  'mse': 11137049637.959,
  'mae': 73662.564,
  'rmse': 105532.221},
 'test': {'r2': 0.842,
  'mse': 10829518354.041,
  'mae': 73977.151,
  'rmse': 104064.972}}
```

Our test model has performed well against the training model, suggesting the training is neither over nor under fit. While the values themselves are large, the difference between the Train and Test data is not that great.

Let's continue to refine our model and see if we can improve the normality of our residuals without sacrificing R2.

5.2 Iterations

In this section we will attempt to brute force a better QQ plot, while preserving as high a R2 value as possible. We will programmatically eliminate variables, comparing the JB Score, until we happen

upon an imporoved version.

5.2.1 1

In [90]:

```
# merging processed continuous values with encoded categories
# dropping sqft_lot due to high p-value
X_train_2 = X_train_scaled.drop(['sqft_lot15'],
                                axis=1).merge(features_train,
                                              left_index=True, right_in
X_test_2 = X_test_scaled.drop(['sqft_lot15'],
                                axis=1).merge(features_test,
                                              left_index=True, right_in
# adding constant
X_train_2_int = sm.add_constant(X_train_2)
X_test_2_int = sm.add_constant(X_test_2)
# resetting indicies to ensure alignment
X_train_2_int.reset_index(drop=True, inplace=True)
y_train.reset_index(drop=True, inplace=True)

# fitting model
model_2 = sm.OLS(y_train ,X_train_2_int).fit()
model_2.summary2()
```

Out[90]:

Model:	OLS	Adj. R-squared:	0.833
Dependent Variable:	price	AIC:	412935.7888
Date:	2021-09-29 09:12	BIC:	413595.7214
No. Observations:	15893	Log-Likelihood:	-2.0638e+05
Df Model:	85	F-statistic:	931.7
Df Residuals:	15807	Prob (F-statistic):	0.00
R-squared:	0.834	Scale:	1.1198e+10

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
const	319459.9585	6511.4028	49.0616	0.0000	306696.8663	332223.0507
bedrooms	-8245.7956	1119.0488	-7.3686	0.0000	-10439.2589	-6052.3323
bathrooms	12814.1792	1494.8343	8.5723	0.0000	9884.1335	15744.2250
floors	-11244.5935	1319.3759	-8.5227	0.0000	-13830.7208	-8658.4663
condition	13723.5911	950.3585	14.4404	0.0000	11860.7799	15586.4022
grade	60486.9097	1570.0825	38.5247	0.0000	57409.3689	63564.4504
sqft_living	64715.0475	2367.5103	27.3346	0.0000	60074.4572	69355.6377
sqft_lot	9547.5809	931.7636	10.2468	0.0000	7721.2180	11373.9439
sqft_above	42836.3114	2217.0115	19.3216	0.0000	38490.7161	47181.9068
sqft_living15	21265.2063	1520.4720	13.9859	0.0000	18284.9078	24245.5048
age	20362.9376	1409.1194	14.4508	0.0000	17600.9029	23124.9723
waterfront_1.0	205790.2200	14141.0623	14.5527	0.0000	178072.1247	233508.3152
view_1.0	63330.2819	7049.3781	8.9838	0.0000	49512.6967	77147.8671
view_2.0	62405.1978	4388.2929	14.2208	0.0000	53803.6431	71006.7525
view_3.0	127127.4217	6218.7971	20.4424	0.0000	114937.8701	139316.9733

view_4.0	230694.3632	9572.9028	24.0987	0.0000	211930.3817	249458.3446
zipcode_98002	12441.2597	10905.4114	1.1408	0.2540	-8934.5907	33817.1101
zipcode_98003	-10002.9010	9738.8466	-1.0271	0.3044	-29092.1512	9086.3493
zipcode_98004	603682.6532	10504.6055	57.4684	0.0000	583092.4282	624272.8783
zipcode_98005	318901.8093	11567.8622	27.5679	0.0000	296227.4799	341576.1387
zipcode_98006	272814.7676	8709.1340	31.3251	0.0000	255743.8714	289885.6638
zipcode_98007	249393.3396	12184.3377	20.4684	0.0000	225510.6479	273276.0314
zipcode_98008	244965.7395	9720.6474	25.2006	0.0000	225912.1618	264019.3173
zipcode_98010	87379.0533	13521.1177	6.4624	0.0000	60876.1202	113881.9864
zipcode_98011	132943.5515	10851.3100	12.2514	0.0000	111673.7461	154213.3568
zipcode_98014	113074.9384	12992.1099	8.7034	0.0000	87608.9210	138540.9558
zipcode_98019	100893.0357	11036.1491	9.1421	0.0000	79260.9244	122525.1469
zipcode_98022	-4745.3044	10395.9590	-0.4565	0.6481	-25122.5700	15631.9612
zipcode_98023	-27018.1763	8521.4132	-3.1706	0.0015	-43721.1181	-10315.2344
zipcode_98024	160314.8812	15339.2395	10.4513	0.0000	130248.2219	190381.5405
zipcode_98027	178920.0822	8927.1195	20.0423	0.0000	161421.9098	196418.2547
zipcode_98028	129666.2508	9570.1383	13.5490	0.0000	110907.6880	148424.8135
zipcode_98029	215678.3647	9363.6396	23.0336	0.0000	197324.5629	234032.1665
zipcode_98030	314.2769	10027.4944	0.0313	0.9750	-19340.7560	19969.3097
zipcode_98031	5210.4851	9893.9489	0.5266	0.5985	-14182.7834	24603.7535
zipcode_98032	568.2630	12697.2599	0.0448	0.9643	-24319.8148	25456.3408
zipcode_98033	353962.0316	8895.3197	39.7919	0.0000	336526.1903	371397.8730
zipcode_98034	187732.2372	8337.5995	22.5163	0.0000	171389.5911	204074.8833
zipcode_98038	32186.1733	8190.9233	3.9295	0.0001	16131.0292	48241.3174
zipcode_98039	766045.1678	32632.9620	23.4746	0.0000	702080.8398	830009.4959
zipcode_98040	478927.0218	10312.7332	46.4404	0.0000	458712.8884	499141.1553
zipcode_98042	4502.7509	8346.3288	0.5395	0.5896	-11857.0057	20862.5075
zipcode_98045	94953.6901	10283.0102	9.2340	0.0000	74797.8170	115109.5631
zipcode_98052	239602.0221	8270.2490	28.9716	0.0000	223391.3906	255812.6537
zipcode_98053	223030.5603	9048.1641	24.6493	0.0000	205295.1265	240765.9942
zipcode_98055	43523.2285	9920.8827	4.3870	0.0000	24077.1668	62969.2902
zipcode_98056	101505.1241	8903.9452	11.4000	0.0000	84052.3758	118957.8725
zipcode_98058	25404.8975	8635.8930	2.9418	0.0033	8477.5621	42332.2329
zipcode_98059	94853.0309	8627.2874	10.9945	0.0000	77942.5634	111763.4984
zipcode_98065	106899.0716	9391.1937	11.3829	0.0000	88491.2606	125306.8826
zipcode_98070	96164.4383	13812.3597	6.9622	0.0000	69090.6376	123238.2390

zipcode_98072	171703.4473	9828.2976	17.4703	0.0000	152438.8628	190968.0318
zipcode_98074	191675.3130	8916.4846	21.4967	0.0000	174197.9862	209152.6399
zipcode_98075	204033.9057	9513.1541	21.4476	0.0000	185387.0385	222680.7730
zipcode_98077	154404.4351	11225.9548	13.7542	0.0000	132400.2831	176408.5872
zipcode_98092	-35244.4702	9364.8638	-3.7635	0.0002	-53600.6714	-16888.2689
zipcode_98102	429202.4138	14453.1321	29.6962	0.0000	400872.6262	457532.2015
zipcode_98103	322765.9534	8393.3664	38.4549	0.0000	306313.9979	339217.9089
zipcode_98105	438451.0353	10742.9731	40.8128	0.0000	417393.5826	459508.4881
zipcode_98106	129651.7917	9322.3804	13.9076	0.0000	111378.8626	147924.7208
zipcode_98107	319876.5469	10091.1354	31.6988	0.0000	300096.7704	339656.3234
zipcode_98108	117006.0458	11186.2175	10.4598	0.0000	95079.7834	138932.3082
zipcode_98109	456584.4742	14203.1906	32.1466	0.0000	428744.6004	484424.3480
zipcode_98112	500744.3130	10762.2780	46.5277	0.0000	479649.0204	521839.6056
zipcode_98115	327579.9283	8372.2890	39.1267	0.0000	311169.2869	343990.5697
zipcode_98116	291391.6723	9532.6336	30.5678	0.0000	272706.6230	310076.7216
zipcode_98117	308752.2721	8497.7967	36.3332	0.0000	292095.6212	325408.9230
zipcode_98118	168114.3644	8587.5144	19.5766	0.0000	151281.8565	184946.8722
zipcode_98119	441113.0683	11642.8366	37.8871	0.0000	418291.7805	463934.3560
zipcode_98122	316838.9221	9978.5497	31.7520	0.0000	297279.8265	336398.0177
zipcode_98125	199842.8146	8952.4228	22.3228	0.0000	182295.0448	217390.5845
zipcode_98126	189813.3884	9221.9412	20.5828	0.0000	171737.3317	207889.4451
zipcode_98133	155105.8003	8503.0712	18.2412	0.0000	138438.8107	171772.7898
zipcode_98136	247017.4315	9983.1656	24.7434	0.0000	227449.2882	266585.5748
zipcode_98144	245511.4235	9463.0747	25.9441	0.0000	226962.7176	264060.1293
zipcode_98146	113394.6943	9694.7644	11.6965	0.0000	94391.8501	132397.5385
zipcode_98148	67411.2327	19278.8675	3.4966	0.0005	29622.4532	105200.0123
zipcode_98155	137548.6848	8802.0122	15.6270	0.0000	120295.7368	154801.6328
zipcode_98166	88153.0620	10109.9974	8.7194	0.0000	68336.3138	107969.8102
zipcode_98168	58835.5143	9924.1887	5.9285	0.0000	39382.9724	78288.0563
zipcode_98177	216686.6017	10153.3273	21.3414	0.0000	196784.9219	236588.2816
zipcode_98178	53943.9589	9970.2934	5.4105	0.0000	34401.0465	73486.8713
zipcode_98188	36601.1014	12412.6741	2.9487	0.0032	12270.8443	60931.3586
zipcode_98198	15585.8850	9769.2154	1.5954	0.1106	-3562.8916	34734.6617
zipcode_98199	376378.1601	9701.2247	38.7970	0.0000	357362.6530	395393.6672
is_renovated_1	8138.0515	2091.4446	3.8911	0.0001	4038.5816	12237.5215

Omnibus:	3369.620	Durbin-Watson:	1.994
Prob(Omnibus):	0.000	Jarque-Bera (JB):	19098.153
Skew:	0.902	Prob(JB):	0.000
Kurtosis:	8.058	Condition No.:	144

This iteration of our model shows a very slight improvement of the normality of residuals, and a very small reduction in R2. Since there are no more p-values that we should use as a basis for exclusion, we will attempt to engineer better normality by trial and error.

5.2.2 2

In [91]:

```
# merging processed continuous values with encoded categories
X_train_3 = X_train_scaled.drop(['sqft_lot',
                                 'sqft_lot15'],
                                axis=1).merge(features_train,
                                              left_index=True, right_inde
X_test_3 = X_test_scaled.drop(['sqft_lot',
                                'sqft_lot15'],
                               axis=1).merge(features_test,
                                             left_index=True, right_inde
# adding constant
X_train_3_int = sm.add_constant(X_train_3)
X_test_3_int = sm.add_constant(X_test_3)

# resetting indicies to ensure alignment
X_train_3_int.reset_index(drop=True, inplace=True)
y_train.reset_index(drop=True, inplace=True)

# fitting model
model_3 = sm.OLS(y_train ,X_train_3_int).fit()
model_3.summary2() #.summary2() supresses scientific notation and incre
```

Out[91]:

Model:	OLS	Adj. R-squared:	0.832
Dependent Variable:	price	AIC:	413039.0076
Date:	2021-09-29 09:12	BIC:	413691.2665
No. Observations:	15893	Log-Likelihood:	-2.0643e+05
Df Model:	84	F-statistic:	935.4
Df Residuals:	15808	Prob (F-statistic):	0.00
R-squared:	0.833	Scale:	1.1271e+10

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
const	320146.6405	6532.4400	49.0087	0.0000	307342.3129	332950.9681
bedrooms	-8862.8207	1121.0971	-7.9055	0.0000	-11060.2989	-6665.3425
bathrooms	12658.2349	1499.6656	8.4407	0.0000	9718.7193	15597.7504
floors	-11775.9314	1322.6860	-8.9030	0.0000	-14368.5468	-9183.3161
condition	13592.1520	953.3926	14.2566	0.0000	11723.3937	15460.9103
grade	60643.1910	1575.1642	38.4996	0.0000	57555.6894	63730.6926
sqft_living	66237.9424	2370.6003	27.9414	0.0000	61591.2954	70884.5894
sqft_above	43548.8794	2223.1976	19.5884	0.0000	39191.1586	47906.6002
sqft_living15	21207.2636	1525.4546	13.9023	0.0000	18217.1986	24197.3286
age	21164.6127	1411.5661	14.9937	0.0000	18397.7820	23931.4433
waterfront_1.0	201975.4825	14182.5837	14.2411	0.0000	174176.0007	229774.9643
view_1.0	62822.5798	7072.3534	8.8828	0.0000	48959.9606	76685.1991
view_2.0	63506.3537	4401.3836	14.4287	0.0000	54879.1397	72133.5676
view_3.0	130144.4309	6232.2225	20.8825	0.0000	117928.5640	142360.2978

view_4.0	231888.0271	9603.6288	24.1459	0.0000	213063.8192	250712.2350
zipcode_98002	10899.3246	10940.1828	0.9963	0.3191	-10544.6816	32343.3309
zipcode_98003	-11398.5145	9769.8731	-1.1667	0.2433	-30548.5802	7751.5512
zipcode_98004	601470.5803	10536.8765	57.0824	0.0000	580817.1004	622124.0603
zipcode_98005	318000.3831	11605.5151	27.4008	0.0000	295252.2498	340748.5165
zipcode_98006	270190.3284	8733.9553	30.9356	0.0000	253070.7798	287309.8770
zipcode_98007	247584.1102	12223.0671	20.2555	0.0000	223625.5044	271542.7159
zipcode_98008	242880.3649	9750.4321	24.9097	0.0000	223768.4059	261992.3240
zipcode_98010	96147.3167	13538.3270	7.1019	0.0000	69610.6516	122683.9818
zipcode_98011	130970.7121	10885.2316	12.0320	0.0000	109634.4166	152307.0076
zipcode_98014	132202.1672	12899.5366	10.2486	0.0000	106917.6040	157486.7304
zipcode_98019	106299.1287	11059.7321	9.6114	0.0000	84620.7924	127977.4651
zipcode_98022	9101.9778	10341.6019	0.8801	0.3788	-11168.7415	29372.6971
zipcode_98023	-28455.1748	8548.2394	-3.3288	0.0009	-45210.6991	-11699.6506
zipcode_98024	178072.0161	15291.0867	11.6455	0.0000	148099.7420	208044.2902
zipcode_98027	182111.0829	8950.9847	20.3454	0.0000	164566.1319	199656.0338
zipcode_98028	128314.0231	9600.6535	13.3651	0.0000	109495.6472	147132.3991
zipcode_98029	213702.8516	9392.3981	22.7527	0.0000	195292.6801	232113.0232
zipcode_98030	-796.6731	10059.8363	-0.0792	0.9369	-20515.0998	18921.7536
zipcode_98031	4592.4273	9926.2559	0.4627	0.6436	-14864.1665	24049.0210
zipcode_98032	-568.4029	12738.4712	-0.0446	0.9644	-25537.2594	24400.4536
zipcode_98033	351760.6865	8921.9286	39.4265	0.0000	334272.6889	369248.6842
zipcode_98034	186026.0506	8363.3117	22.2431	0.0000	169633.0058	202419.0955
zipcode_98038	34318.6662	8215.1691	4.1775	0.0000	18215.9976	50421.3347
zipcode_98039	764211.1355	32739.6353	23.3421	0.0000	700037.7158	828384.5551
zipcode_98040	476245.0802	10343.2671	46.0440	0.0000	455971.0969	496519.0635
zipcode_98042	5299.9113	8373.3742	0.6329	0.5268	-11112.8571	21712.6798
zipcode_98045	101183.7330	10298.7301	9.8249	0.0000	80997.0474	121370.4186
zipcode_98052	237840.5679	8295.6157	28.6706	0.0000	221580.2149	254100.9209
zipcode_98053	225964.0189	9073.3331	24.9042	0.0000	208179.2510	243748.7867
zipcode_98055	42240.9653	9952.6707	4.2442	0.0000	22732.5954	61749.3352
zipcode_98056	99818.0220	8931.6581	11.1758	0.0000	82310.9532	117325.0907
zipcode_98058	25835.7360	8664.1503	2.9819	0.0029	8853.0131	42818.4590
zipcode_98059	93816.6182	8655.0243	10.8396	0.0000	76851.7833	110781.4532
zipcode_98065	105470.0847	9420.9953	11.1952	0.0000	87003.8594	123936.3100
zipcode_98070	116831.3296	13709.1821	8.5221	0.0000	89959.7690	143702.8902

zipcode_98072	174362.2828	9857.1364	17.6889	0.0000	155041.1712	193683.3944
zipcode_98074	190044.0151	8944.3400	21.2474	0.0000	172512.0884	207575.9418
zipcode_98075	203013.5005	9543.8721	21.2716	0.0000	184306.4226	221720.5785
zipcode_98077	162157.8155	11237.2069	14.4304	0.0000	140131.6082	184184.0228
zipcode_98092	-32100.4819	9390.5735	-3.4184	0.0006	-50507.0771	-13693.8867
zipcode_98102	426052.3032	14497.3152	29.3884	0.0000	397635.9118	454468.6945
zipcode_98103	320040.6382	8416.7011	38.0245	0.0000	303542.9441	336538.3324
zipcode_98105	434897.5626	10772.6356	40.3706	0.0000	413781.9680	456013.1572
zipcode_98106	127975.1545	9351.5541	13.6849	0.0000	109645.0418	146305.2672
zipcode_98107	317422.5806	10121.4229	31.3615	0.0000	297583.4373	337261.7240
zipcode_98108	114533.0169	11220.3403	10.2076	0.0000	92539.8700	136526.1638
zipcode_98109	452768.7112	14244.9349	31.7845	0.0000	424847.0139	480690.4085
zipcode_98112	496932.7225	10791.1696	46.0499	0.0000	475780.7992	518084.6459
zipcode_98115	324566.8614	8394.6010	38.6638	0.0000	308112.4860	341021.2369
zipcode_98116	288234.6526	9558.9419	30.1534	0.0000	269498.0361	306971.2691
zipcode_98117	305758.4879	8520.6626	35.8844	0.0000	289057.0172	322459.9585
zipcode_98118	165380.0930	8611.5551	19.2044	0.0000	148500.4627	182259.7234
zipcode_98119	437560.1633	11675.8902	37.4755	0.0000	414674.0867	460446.2399
zipcode_98122	313790.7074	10006.8692	31.3575	0.0000	294176.1024	333405.3124
zipcode_98125	197816.1634	8979.6300	22.0294	0.0000	180215.0642	215417.2625
zipcode_98126	187000.5729	9248.1259	20.2204	0.0000	168873.1913	205127.9545
zipcode_98133	153339.2338	8529.2414	17.9781	0.0000	136620.9477	170057.5198
zipcode_98136	244339.4245	10012.5172	24.4034	0.0000	224713.7487	263965.1003
zipcode_98144	242416.9129	9489.3152	25.5463	0.0000	223816.7726	261017.0531
zipcode_98146	111636.6110	9725.0785	11.4793	0.0000	92574.3480	130698.8741
zipcode_98148	65844.1937	19341.5703	3.4043	0.0007	27932.5098	103755.8776
zipcode_98155	136033.8519	8829.6721	15.4064	0.0000	118726.6873	153341.0164
zipcode_98166	86574.9394	10142.0214	8.5363	0.0000	66695.4207	106454.4581
zipcode_98168	57679.3469	9956.1359	5.7933	0.0000	38164.1849	77194.5090
zipcode_98177	213911.9383	10183.0473	21.0067	0.0000	193952.0041	233871.8724
zipcode_98178	51919.5720	10001.0716	5.1914	0.0000	32316.3310	71522.8131
zipcode_98188	35028.7012	12452.4853	2.8130	0.0049	10620.4097	59436.9928
zipcode_98198	14030.4982	9800.1141	1.4317	0.1523	-5178.8432	33239.8396
zipcode_98199	372688.1912	9726.3754	38.3173	0.0000	353623.3860	391752.9964
is_renovated_1	7772.1793	2098.0070	3.7046	0.0002	3659.8463	11884.5124

Omnibus:	3380.169	Durbin-Watson:	1.996
Prob(Omnibus):	0.000	Jarque-Bera (JB):	18568.140
Skew:	0.914	Prob(JB):	0.000
Kurtosis:	7.970	Condition No.:	144

5.2.3 3

In [92]:

```
# merging processed continuous values with encoded categories
X_train_4 = X_train_scaled.drop(['sqft_lot15',
                                 'bedrooms'],
                                axis=1).merge(features_train,
                                              left_index=True, right_inde
X_test_4 = X_test_scaled.drop(['sqft_lot15',
                                 'bedrooms'],
                                axis=1).merge(features_test,
                                              left_index=True, right_inde
# adding constant
X_train_4_int = sm.add_constant(X_train_4)
X_test_4_int = sm.add_constant(X_test_4)

# resetting indicies to ensure alignment
X_train_4_int.reset_index(drop=True, inplace=True)
y_train.reset_index(drop=True, inplace=True)

# fitting model
model_4 = sm.OLS(y_train ,X_train_4_int).fit()
model_4.summary2() #.summary2() supresses scientific notation and incre
```

Out[92]:

Model:	OLS	Adj. R-squared:	0.832
Dependent Variable:	price	AIC:	412988.2866
Date:	2021-09-29 09:12	BIC:	413640.5455
No. Observations:	15893	Log-Likelihood:	-2.0641e+05
Df Model:	84	F-statistic:	939.0
Df Residuals:	15808	Prob (F-statistic):	0.00
R-squared:	0.833	Scale:	1.1235e+10

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
const	318078.5357	6519.6660	48.7876	0.0000	305299.2467	330857.8246
bathrooms	10609.6020	1467.0545	7.2319	0.0000	7734.0078	13485.1961
floors	-11267.5289	1321.5945	-8.5257	0.0000	-13858.0048	-8677.0530
condition	13608.9463	951.8317	14.2976	0.0000	11743.2476	15474.6449
grade	61886.8242	1561.1708	39.6413	0.0000	58826.7513	64946.8970
sqft_living	59425.3216	2259.8492	26.2961	0.0000	54995.7593	63854.8838
sqft_lot	9917.0308	931.9807	10.6408	0.0000	8090.2422	11743.8193
sqft_above	43018.3320	2220.6077	19.3723	0.0000	38665.6876	47370.9764
sqft_living15	21787.7512	1521.3755	14.3211	0.0000	18805.6818	24769.8206
age	19427.8234	1405.7569	13.8202	0.0000	16672.3795	22183.2673
waterfront_1.0	208354.2180	14160.5912	14.7137	0.0000	180597.8440	236110.5921
view_1.0	64842.8377	7058.2571	9.1868	0.0000	51007.8487	78677.8267
view_2.0	64055.3027	4389.9569	14.5913	0.0000	55450.4864	72660.1191
view_3.0	129356.7807	6221.8950	20.7906	0.0000	117161.1568	141552.4046

view_4.0	233115.4723	9583.3765	24.3250	0.0000	214330.9613	251899.9834
zipcode_98002	12302.5849	10923.7632	1.1262	0.2601	-9109.2370	33714.4068
zipcode_98003	-9497.7693	9755.0081	-0.9736	0.3303	-28618.6979	9623.1593
zipcode_98004	603758.8696	10522.2934	57.3790	0.0000	583133.9744	624383.7648
zipcode_98005	318623.7098	11587.2843	27.4977	0.0000	295911.3109	341336.1088
zipcode_98006	273062.7330	8723.7378	31.3011	0.0000	255963.2120	290162.2541
zipcode_98007	246801.4555	12199.7729	20.2300	0.0000	222888.5090	270714.4020
zipcode_98008	242937.1494	9733.1141	24.9599	0.0000	223859.1356	262015.1633
zipcode_98010	89748.1868	13540.0618	6.6283	0.0000	63208.1213	116288.2523
zipcode_98011	133683.0757	10869.1220	12.2993	0.0000	112378.3568	154987.7946
zipcode_98014	116799.5818	13004.1391	8.9817	0.0000	91309.9858	142289.1778
zipcode_98019	102451.4734	11052.7072	9.2694	0.0000	80786.9065	124116.0402
zipcode_98022	-3671.0800	10412.4451	-0.3526	0.7244	-24080.6600	16738.5000
zipcode_98023	-26978.0129	8535.7641	-3.1606	0.0016	-43709.0841	-10246.9416
zipcode_98024	162966.1567	15360.8479	10.6092	0.0000	132857.1428	193075.1707
zipcode_98027	181022.0885	8937.5888	20.2540	0.0000	163503.3950	198540.7820
zipcode_98028	130221.9795	9585.9597	13.5847	0.0000	111432.4051	149011.5539
zipcode_98029	217096.2991	9377.4301	23.1509	0.0000	198715.4665	235477.1316
zipcode_98030	-70.0939	10044.2478	-0.0070	0.9944	-19757.9654	19617.7776
zipcode_98031	4855.6994	9910.4960	0.4900	0.6242	-14570.0031	24281.4020
zipcode_98032	-814.1333	12717.2576	-0.0640	0.9490	-25741.4088	24113.1421
zipcode_98033	354508.1992	8909.9928	39.7877	0.0000	337043.5970	371972.8014
zipcode_98034	187314.0170	8351.4491	22.4289	0.0000	170944.2242	203683.8098
zipcode_98038	33172.1135	8203.6246	4.0436	0.0001	17092.0736	49252.1534
zipcode_98039	765973.0281	32687.9245	23.4329	0.0000	701900.9676	830045.0885
zipcode_98040	477736.4837	10328.8351	46.2527	0.0000	457490.7887	497982.1787
zipcode_98042	4794.2300	8360.2927	0.5735	0.5663	-11592.8973	21181.3573
zipcode_98045	96387.3970	10298.4859	9.3594	0.0000	76201.1899	116573.6041
zipcode_98052	239834.1249	8284.1186	28.9511	0.0000	223596.3075	256071.9422
zipcode_98053	227415.2035	9043.7834	25.1460	0.0000	209688.3564	245142.0506
zipcode_98055	44770.4281	9936.1461	4.5058	0.0000	25294.4484	64246.4078
zipcode_98056	101670.4266	8918.9139	11.3994	0.0000	84188.3380	119152.5152
zipcode_98058	24855.8094	8650.1164	2.8735	0.0041	7900.5945	41811.0243
zipcode_98059	94420.6777	8641.6186	10.9263	0.0000	77482.1196	111359.2358
zipcode_98065	109350.4521	9401.1071	11.6317	0.0000	90923.2100	127777.6943
zipcode_98070	100227.2673	13824.5960	7.2499	0.0000	73129.4822	127325.0523

zipcode_98072	173567.6553	9841.5893	17.6361	0.0000	154277.0177	192858.2928
zipcode_98074	192720.8479	8930.3717	21.5804	0.0000	175216.3007	210225.3950
zipcode_98075	204673.2680	9528.7808	21.4795	0.0000	185995.7708	223350.7653
zipcode_98077	156435.9341	11241.4712	13.9160	0.0000	134401.3684	178470.4998
zipcode_98092	-35220.6897	9380.6365	-3.7546	0.0002	-53607.8072	-16833.5721
zipcode_98102	431781.1509	14473.2309	29.8331	0.0000	403411.9675	460150.3343
zipcode_98103	325581.6226	8398.7860	38.7653	0.0000	309119.0441	342044.2011
zipcode_98105	439113.6470	10760.6906	40.8072	0.0000	418021.4660	460205.8279
zipcode_98106	130344.8265	9337.6069	13.9591	0.0000	112042.0518	148647.6011
zipcode_98107	322540.2949	10101.6438	31.9295	0.0000	302739.9207	342340.6690
zipcode_98108	118838.5728	11202.2890	10.6084	0.0000	96880.8085	140796.3370
zipcode_98109	459643.7495	14221.0335	32.3214	0.0000	431768.9017	487518.5973
zipcode_98112	502930.5896	10776.3075	46.6700	0.0000	481807.7977	524053.3814
zipcode_98115	329386.6982	8382.7931	39.2932	0.0000	312955.4676	345817.9288
zipcode_98116	294486.4673	9539.4170	30.8705	0.0000	275788.1218	313184.8128
zipcode_98117	311494.0872	8503.9454	36.6294	0.0000	294825.3842	328162.7902
zipcode_98118	170118.6199	8597.6624	19.7866	0.0000	153266.2208	186971.0189
zipcode_98119	443640.3860	11657.3851	38.0566	0.0000	420790.5816	466490.1904
zipcode_98122	318677.8424	9992.2301	31.8926	0.0000	299091.9316	338263.7531
zipcode_98125	200944.0677	8966.2517	22.4112	0.0000	183369.1916	218518.9438
zipcode_98126	193418.5471	9224.4641	20.9680	0.0000	175337.5453	211499.5489
zipcode_98133	155952.8931	8516.6145	18.3116	0.0000	139259.3572	172646.4291
zipcode_98136	249880.7466	9992.4020	25.0071	0.0000	230294.4988	269466.9943
zipcode_98144	247848.8573	9473.6862	26.1618	0.0000	229279.3518	266418.3628
zipcode_98146	114306.4416	9710.3024	11.7717	0.0000	95273.1412	133339.7420
zipcode_98148	68540.7353	19310.7286	3.5494	0.0004	30689.5045	106391.9661
zipcode_98155	137672.5372	8816.8214	15.6148	0.0000	120390.5615	154954.5129
zipcode_98166	88869.8023	10126.5570	8.7759	0.0000	69020.5955	108719.0092
zipcode_98168	60529.8125	9938.2355	6.0906	0.0000	41049.7372	80009.8877
zipcode_98177	218026.0865	10168.7985	21.4407	0.0000	198094.0815	237958.0914
zipcode_98178	53855.8753	9987.0792	5.3926	0.0000	34280.0608	73431.6897
zipcode_98188	35894.8486	12433.2102	2.8870	0.0039	11524.3385	60265.3587
zipcode_98198	16496.3537	9784.8870	1.6859	0.0918	-2683.1410	35675.8484
zipcode_98199	379816.3596	9706.3180	39.1308	0.0000	360790.8692	398841.8500
is_renovated_1	8622.6395	2093.9312	4.1179	0.0000	4518.2954	12726.9836

Omnibus:	3419.341	Durbin-Watson:	1.996
Prob(Omnibus):	0.000	Jarque-Bera (JB):	19235.688
Skew:	0.919	Prob(JB):	0.000
Kurtosis:	8.067	Condition No.:	140

5.2.4 4

In [93]:

```
# merging processed continuous values with encoded categories
X_train_5 = X_train_scaled.drop(['sqft_lot15',
                                 'bathrooms'],
                                axis=1).merge(features_train,
                                              left_index=True, right_inde
X_test_5 = X_test_scaled.drop(['sqft_lot15',
                                 'bathrooms'],
                                axis=1).merge(features_test,
                                              left_index=True, right_inde
# adding constant
X_train_5_int = sm.add_constant(X_train_5)
X_test_5_int = sm.add_constant(X_test_5)

# resetting indicies to ensure alignment
X_train_5_int.reset_index(drop=True, inplace=True)
y_train.reset_index(drop=True, inplace=True)

# fitting model
model_5 = sm.OLS(y_train ,X_train_5_int).fit()
model_5.summary2() #.summary2() supresses scientific notation and incre
```

Out[93]:

Model:	OLS	Adj. R-squared:	0.832
Dependent Variable:	price	AIC:	413007.5019
Date:	2021-09-29 09:12	BIC:	413659.7608
No. Observations:	15893	Log-Likelihood:	-2.0642e+05
Df Model:	84	F-statistic:	937.6
Df Residuals:	15808	Prob (F-statistic):	0.00
R-squared:	0.833	Scale:	1.1249e+10

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
const	318120.1655	6524.4337	48.7583	0.0000	305331.5312	330908.7998
bedrooms	-6325.8209	1098.9167	-5.7564	0.0000	-8479.8229	-4171.8188
floors	-8800.3711	1291.1478	-6.8159	0.0000	-11331.1681	-6269.5741
condition	14260.2070	950.4663	15.0034	0.0000	12397.1847	16123.2294
grade	61182.5453	1571.5747	38.9307	0.0000	58102.0796	64263.0110
sqft_living	72664.3114	2183.3329	33.2814	0.0000	68384.7299	76943.8930
sqft_lot	9466.2618	933.8490	10.1368	0.0000	7635.8114	11296.7123
sqft_above	40363.6885	2203.2014	18.3205	0.0000	36045.1625	44682.2145
sqft_living15	21056.3149	1523.7582	13.8187	0.0000	18069.5751	24043.0546
age	16501.4477	1338.2364	12.3307	0.0000	13878.3518	19124.5436
waterfront_1.0	207303.6465	14172.3411	14.6273	0.0000	179524.2414	235083.0516
view_1.0	64194.8227	7064.7983	9.0866	0.0000	50347.0123	78042.6332
view_2.0	63503.0095	4396.4689	14.4441	0.0000	54885.4290	72120.5900
view_3.0	128391.2276	6231.2865	20.6043	0.0000	116177.1953	140605.2598

view_4.0	231447.9816	9594.4204	24.1232	0.0000	212641.8233	250254.1400
zipcode_98002	13832.6154	10929.1745	1.2657	0.2057	-7589.8132	35255.0440
zipcode_98003	-8701.3059	9759.9624	-0.8915	0.3727	-27831.9454	10429.3337
zipcode_98004	604334.6466	10528.3854	57.4005	0.0000	583697.8104	624971.4828
zipcode_98005	320800.7553	11592.2268	27.6738	0.0000	298078.6684	343522.8421
zipcode_98006	272881.3002	8729.0747	31.2612	0.0000	255771.3180	289991.2823
zipcode_98007	249808.2153	12212.1438	20.4557	0.0000	225871.0205	273745.4102
zipcode_98008	246096.1378	9742.0114	25.2613	0.0000	227000.6842	265191.5914
zipcode_98010	86091.6347	13551.2455	6.3530	0.0000	59529.6479	112653.6215
zipcode_98011	134248.7846	10875.0890	12.3446	0.0000	112932.3697	155565.1995
zipcode_98014	112781.6628	13021.8170	8.6610	0.0000	87257.4161	138305.9095
zipcode_98019	101767.3674	11060.9498	9.2006	0.0000	80086.6441	123448.0907
zipcode_98022	-4199.2503	10419.5705	-0.4030	0.6869	-24622.7969	16224.2963
zipcode_98023	-26662.2051	8540.8260	-3.1217	0.0018	-43403.1983	-9921.2119
zipcode_98024	161514.5072	15373.7270	10.5059	0.0000	131380.2486	191648.7657
zipcode_98027	179716.6753	8947.0780	20.0866	0.0000	162179.3818	197253.9688
zipcode_98028	130277.6869	9591.7878	13.5822	0.0000	111476.6888	149078.6851
zipcode_98029	218503.4669	9379.2680	23.2964	0.0000	200119.0319	236887.9020
zipcode_98030	1550.0807	10049.4189	0.1542	0.8774	-18147.9267	21248.0880
zipcode_98031	6684.2421	9915.1091	0.6741	0.5002	-12750.5027	26118.9869
zipcode_98032	-850.8439	12725.2553	-0.0669	0.9467	-25793.7958	24092.1079
zipcode_98033	354992.2006	8914.8766	39.8202	0.0000	337518.0257	372466.3756
zipcode_98034	189441.7340	8354.3021	22.6759	0.0000	173066.3489	205817.1191
zipcode_98038	33800.5510	8207.5106	4.1182	0.0000	17712.8941	49888.2079
zipcode_98039	769421.1101	32705.3106	23.5259	0.0000	705314.9709	833527.2492
zipcode_98040	480087.1975	10335.4595	46.4505	0.0000	459828.5179	500345.8770
zipcode_98042	5105.7683	8365.1450	0.6104	0.5416	-11290.8701	21502.4068
zipcode_98045	95377.4788	10306.4395	9.2542	0.0000	75175.6817	115579.2758
zipcode_98052	240327.6193	8288.7540	28.9944	0.0000	224080.7160	256574.5225
zipcode_98053	223684.5634	9068.5623	24.6659	0.0000	205909.1469	241459.9798
zipcode_98055	45445.1110	9941.0624	4.5715	0.0000	25959.4948	64930.7272
zipcode_98056	101065.1683	8924.1873	11.3249	0.0000	83572.7433	118557.5933
zipcode_98058	25547.4854	8655.6534	2.9515	0.0032	8581.4174	42513.5533
zipcode_98059	94909.0633	8647.0417	10.9759	0.0000	77959.8753	111858.2513
zipcode_98065	108006.6611	9411.8089	11.4757	0.0000	89558.4420	126454.8802
zipcode_98070	96463.8779	13843.9462	6.9679	0.0000	69328.1643	123599.5915

zipcode_98072	171968.6727	9850.7559	17.4574	0.0000	152660.0675	191277.2779
zipcode_98074	191908.7178	8936.8619	21.4738	0.0000	174391.4491	209425.9865
zipcode_98075	204563.1029	9534.7387	21.4545	0.0000	185873.9274	223252.2784
zipcode_98077	153764.4559	11251.4138	13.6662	0.0000	131710.4015	175818.5103
zipcode_98092	-35437.3275	9386.2825	-3.7754	0.0002	-53835.5118	-17039.1432
zipcode_98102	432788.9885	14480.1592	29.8884	0.0000	404406.2247	461171.7523
zipcode_98103	325114.5476	8408.1045	38.6668	0.0000	308633.7038	341595.3914
zipcode_98105	442904.8869	10754.9749	41.1814	0.0000	421823.9094	463985.8643
zipcode_98106	131607.9888	9340.9291	14.0894	0.0000	113298.7023	149917.2753
zipcode_98107	323692.1032	10104.4008	32.0348	0.0000	303886.3251	343497.8812
zipcode_98108	120249.4689	11205.4190	10.7314	0.0000	98285.5696	142213.3683
zipcode_98109	461114.2072	14225.8609	32.4138	0.0000	433229.8973	488998.5171
zipcode_98112	504254.8786	10779.1117	46.7807	0.0000	483126.5901	525383.1671
zipcode_98115	328868.4759	8390.1092	39.1972	0.0000	312422.9049	345314.0469
zipcode_98116	293704.4705	9550.6361	30.7523	0.0000	274984.1344	312424.8067
zipcode_98117	310649.4762	8514.3678	36.4853	0.0000	293960.3441	327338.6082
zipcode_98118	169529.4689	8605.5896	19.6999	0.0000	152661.5317	186397.4062
zipcode_98119	445474.8363	11658.3498	38.2108	0.0000	422623.1408	468326.5318
zipcode_98122	320122.4845	9994.0294	32.0314	0.0000	300533.0469	339711.9221
zipcode_98125	200616.8560	8972.4677	22.3592	0.0000	183029.7960	218203.9160
zipcode_98126	190952.1869	9242.1006	20.6611	0.0000	172836.6156	209067.7583
zipcode_98133	154627.0271	8522.3596	18.1437	0.0000	137922.2301	171331.8241
zipcode_98136	248779.2117	10003.9067	24.8682	0.0000	229170.4134	268388.0100
zipcode_98144	247580.0447	9481.6610	26.1115	0.0000	228994.9076	266165.1818
zipcode_98146	113121.6910	9716.9133	11.6417	0.0000	94075.4326	132167.9494
zipcode_98148	69344.9142	19321.6940	3.5890	0.0003	31472.1902	107217.6383
zipcode_98155	137782.3657	8822.1268	15.6178	0.0000	120489.9910	155074.7405
zipcode_98166	88469.7948	10133.0819	8.7308	0.0000	68607.7985	108331.7912
zipcode_98168	57837.6324	9946.2311	5.8150	0.0000	38341.8850	77333.3799
zipcode_98177	217406.7783	10176.2304	21.3642	0.0000	197460.2060	237353.3506
zipcode_98178	53963.0802	9993.1254	5.4000	0.0000	34375.4146	73550.7458
zipcode_98188	37185.6246	12440.9118	2.9890	0.0028	12800.0185	61571.2307
zipcode_98198	15561.5606	9791.5868	1.5893	0.1120	-3631.0664	34754.1876
zipcode_98199	379054.3975	9718.4046	39.0038	0.0000	360005.2160	398103.5790
is_renovated_1	8933.9735	2094.1673	4.2661	0.0000	4829.1667	13038.7804

Omnibus:	3348.871	Durbin-Watson:	1.993
Prob(Omnibus):	0.000	Jarque-Bera (JB):	18641.962
Skew:	0.901	Prob(JB):	0.000
Kurtosis:	7.990	Condition No.:	134

5.2.5 5

In [94]:

```
# merging processed continuous values with encoded categories
X_train_6 = X_train_scaled.drop(['sqft_lot15',
                                 'floors'],
                                axis=1).merge(features_train,
                                              left_index=True, right_inde
X_test_6 = X_test_scaled.drop(['sqft_lot15',
                                 'floors'],
                                axis=1).merge(features_test,
                                              left_index=True, right_inde
# adding constant
X_train_6_int = sm.add_constant(X_train_6)
X_test_6_int = sm.add_constant(X_test_6)

# resetting indicies to ensure alignment
X_train_6_int.reset_index(drop=True, inplace=True)
y_train.reset_index(drop=True, inplace=True)

# fitting model
model_6 = sm.OLS(y_train ,X_train_6_int).fit()
model_6.summary2() #.summary2() supresses scientific notation and incre
```

Out[94]:

Model:	OLS	Adj. R-squared:	0.832
Dependent Variable:	price	AIC:	413006.6525
Date:	2021-09-29 09:12	BIC:	413658.9114
No. Observations:	15893	Log-Likelihood:	-2.0642e+05
Df Model:	84	F-statistic:	937.7
Df Residuals:	15808	Prob (F-statistic):	0.00
R-squared:	0.833	Scale:	1.1248e+10

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
const	322776.8994	6514.4718	49.5477	0.0000	310007.7915	335546.0072
bedrooms	-8268.2953	1121.5784	-7.3720	0.0000	-10466.7169	-6069.8738
bathrooms	10060.9478	1462.8132	6.8778	0.0000	7193.6671	12928.2286
condition	14089.3109	951.5380	14.8069	0.0000	12224.1880	15954.4339
grade	60252.8031	1573.3951	38.2948	0.0000	57168.7693	63336.8370
sqft_living	70711.2463	2265.6712	31.2098	0.0000	66270.2722	75152.2204
sqft_lot	9859.6822	933.1509	10.5660	0.0000	8030.6001	11688.7644
sqft_above	33549.6996	1935.1286	17.3372	0.0000	29756.6268	37342.7725
sqft_living15	22337.4688	1518.6871	14.7084	0.0000	19360.6688	25314.2688
age	24218.4914	1337.5473	18.1066	0.0000	21596.7462	26840.2366
waterfront_1.0	203646.5308	14170.8248	14.3708	0.0000	175870.0978	231422.9638
view_1.0	63338.4920	7065.3325	8.9647	0.0000	49489.6344	77187.3496
view_2.0	61770.5931	4397.5915	14.0465	0.0000	53150.8121	70390.3740
view_3.0	125770.7125	6230.8293	20.1852	0.0000	113557.5763	137983.8487

view_4.0	228404.8173	9590.7899	23.8150	0.0000	209605.7751	247203.8595
zipcode_98002	12395.1302	10930.0917	1.1340	0.2568	-9029.0963	33819.3566
zipcode_98003	-9017.5394	9760.2001	-0.9239	0.3555	-28148.6449	10113.5661
zipcode_98004	602352.2781	10527.2175	57.2186	0.0000	581717.7310	622986.8251
zipcode_98005	320492.1408	11592.5345	27.6464	0.0000	297769.4508	343214.8308
zipcode_98006	271993.3186	8728.3104	31.1622	0.0000	254884.8347	289101.8025
zipcode_98007	248399.9609	12211.3550	20.3417	0.0000	224464.3122	272335.6097
zipcode_98008	246522.9906	9740.9263	25.3080	0.0000	227429.6640	265616.3171
zipcode_98010	86348.7470	13551.1777	6.3720	0.0000	59786.8931	112910.6009
zipcode_98011	132578.2104	10875.7843	12.1902	0.0000	111260.4327	153895.9881
zipcode_98014	112008.4375	13020.9102	8.6022	0.0000	86485.9683	137530.9067
zipcode_98019	98047.5386	11056.0637	8.8682	0.0000	76376.3926	119718.6847
zipcode_98022	-4284.7885	10419.3469	-0.4112	0.6809	-24707.8968	16138.3199
zipcode_98023	-25954.9115	8539.7838	-3.0393	0.0024	-42693.8619	-9215.9611
zipcode_98024	160860.2497	15373.8222	10.4633	0.0000	130725.8045	190994.6949
zipcode_98027	175651.3018	8939.0622	19.6499	0.0000	158129.7203	193172.8832
zipcode_98028	128805.6237	9591.2639	13.4295	0.0000	110005.6524	147605.5950
zipcode_98029	211315.5881	9370.7970	22.5504	0.0000	192947.7571	229683.4191
zipcode_98030	49.1015	10050.1407	0.0049	0.9961	-19650.3206	19748.5236
zipcode_98031	4848.6994	9916.2501	0.4890	0.6249	-14588.2817	24285.6806
zipcode_98032	-539.7963	12725.3298	-0.0424	0.9662	-25482.8941	24403.3015
zipcode_98033	353317.6184	8915.1299	39.6312	0.0000	335842.9469	370792.2899
zipcode_98034	188392.1045	8356.1092	22.5454	0.0000	172013.1773	204771.0317
zipcode_98038	30517.0054	8207.1143	3.7184	0.0002	14430.1252	46603.8855
zipcode_98039	763565.5097	32705.5184	23.3467	0.0000	699458.9631	827672.0564
zipcode_98040	478165.1453	10335.6851	46.2635	0.0000	457906.0236	498424.2670
zipcode_98042	4382.2200	8365.2066	0.5239	0.6004	-12014.5391	20778.9791
zipcode_98045	95471.4514	10306.1034	9.2636	0.0000	75270.3133	115672.5895
zipcode_98052	238729.5467	8288.3316	28.8031	0.0000	222483.4714	254975.6220
zipcode_98053	225981.4227	9062.0004	24.9373	0.0000	208218.8684	243743.9771
zipcode_98055	41798.3287	9941.2668	4.2045	0.0000	22312.3119	61284.3455
zipcode_98056	100636.8659	8923.5129	11.2777	0.0000	83145.7628	118127.9691
zipcode_98058	25760.3814	8655.3372	2.9762	0.0029	8794.9333	42725.8296
zipcode_98059	94795.5692	8646.8105	10.9631	0.0000	77846.8345	111744.3040
zipcode_98065	104769.8355	9409.1171	11.1349	0.0000	86326.8928	123212.7782
zipcode_98070	93479.4181	13840.0190	6.7543	0.0000	66351.4023	120607.4340

zipcode_98072	171568.6301	9850.5287	17.4172	0.0000	152260.4703	190876.7900
zipcode_98074	190884.9798	8936.1814	21.3609	0.0000	173369.0450	208400.9147
zipcode_98075	204633.3703	9534.4241	21.4626	0.0000	185944.8116	223321.9291
zipcode_98077	154807.6546	11251.2620	13.7591	0.0000	132753.8977	176861.4115
zipcode_98092	-35868.3614	9385.7720	-3.8216	0.0001	-54265.5451	-17471.1777
zipcode_98102	413463.8846	14367.1157	28.7785	0.0000	385302.6992	441625.0701
zipcode_98103	308998.4999	8255.0907	37.4313	0.0000	292817.5806	325179.4193
zipcode_98105	427792.2385	10694.0768	40.0027	0.0000	406830.6282	448753.8488
zipcode_98106	124759.0225	9325.7457	13.3779	0.0000	106479.4972	143038.5477
zipcode_98107	306520.8892	9991.2770	30.6788	0.0000	286936.8466	326104.9318
zipcode_98108	111614.0268	11193.5886	9.9712	0.0000	89673.3163	133554.7373
zipcode_98109	444682.1669	14166.3546	31.3900	0.0000	416914.4960	472449.8379
zipcode_98112	487366.7006	10671.2950	45.6708	0.0000	466449.7452	508283.6561
zipcode_98115	319801.5099	8341.2299	38.3399	0.0000	303451.7479	336151.2720
zipcode_98116	282857.0460	9501.3442	29.7702	0.0000	264233.3276	301480.7645
zipcode_98117	300286.0608	8458.6358	35.5005	0.0000	283706.1699	316865.9517
zipcode_98118	162032.2658	8577.1791	18.8911	0.0000	145220.0164	178844.5152
zipcode_98119	427681.5083	11561.7885	36.9909	0.0000	405019.0840	450343.9326
zipcode_98122	304534.4392	9895.9007	30.7738	0.0000	285137.3452	323931.5333
zipcode_98125	195354.3668	8957.1451	21.8099	0.0000	177797.3407	212911.3928
zipcode_98126	183977.7287	9217.2999	19.9600	0.0000	165910.7696	202044.6878
zipcode_98133	150033.6714	8501.4161	17.6481	0.0000	133369.9262	166697.4167
zipcode_98136	239475.2796	9966.3705	24.0283	0.0000	219940.0566	259010.5026
zipcode_98144	234771.0795	9400.0139	24.9756	0.0000	216345.9800	253196.1789
zipcode_98146	111321.9312	9713.6481	11.4604	0.0000	92282.0729	130361.7895
zipcode_98148	68203.7218	19322.2756	3.5298	0.0004	30329.8577	106077.5860
zipcode_98155	136458.9690	8821.0025	15.4698	0.0000	119168.7980	153749.1401
zipcode_98166	87331.3300	10132.4180	8.6190	0.0000	67470.6350	107192.0250
zipcode_98168	56109.7229	9941.4830	5.6440	0.0000	36623.2823	75596.1635
zipcode_98177	215537.7856	10175.4100	21.1822	0.0000	195592.8214	235482.7498
zipcode_98178	51547.9511	9988.8853	5.1605	0.0000	31968.5966	71127.3056
zipcode_98188	36229.8343	12440.6904	2.9122	0.0036	11844.6620	60615.0065
zipcode_98198	15483.3087	9791.3181	1.5813	0.1138	-3708.7917	34675.4091
zipcode_98199	369332.1039	9687.8091	38.1234	0.0000	350342.8929	388321.3148
is_renovated_1	7750.6022	2095.6828	3.6984	0.0002	3642.8250	11858.3795

Omnibus:	3372.529	Durbin-Watson:	1.992
Prob(Omnibus):	0.000	Jarque-Bera (JB):	18977.927
Skew:	0.905	Prob(JB):	0.000
Kurtosis:	8.038	Condition No.:	139

5.2.6 6

In [95]:

```
# merging processed continuous values with encoded categories
X_train_7 = X_train_scaled.drop(['sqft_lot15',
                                 'condition'],
                                axis=1).merge(features_train,
                                              left_index=True, right_inde
X_test_7 = X_test_scaled.drop(['sqft_lot15',
                                 'condition'],
                                axis=1).merge(features_test,
                                              left_index=True, right_inde
# adding constant
X_train_7_int = sm.add_constant(X_train_7)
X_test_7_int = sm.add_constant(X_test_7)

# resetting indicies to ensure alignment
X_train_7_int.reset_index(drop=True, inplace=True)
y_train.reset_index(drop=True, inplace=True)

# fitting model
model_7 = sm.OLS(y_train ,X_train_7_int).fit()
model_7.summary2() #.summary2() supresses scientific notation and incre
```

Out[95]:

Model:	OLS	Adj. R-squared:	0.831
Dependent Variable:	price	AIC:	413142.0786
Date:	2021-09-29 09:12	BIC:	413794.3375
No. Observations:	15893	Log-Likelihood:	-2.0649e+05
Df Model:	84	F-statistic:	928.1
Df Residuals:	15808	Prob (F-statistic):	0.00
R-squared:	0.831	Scale:	1.1345e+10

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
const	320860.0465	6553.2774	48.9618	0.0000	308014.8754	333705.2177
bedrooms	-7981.2426	1126.2193	-7.0868	0.0000	-10188.7609	-5773.7243
bathrooms	14236.0224	1501.3468	9.4822	0.0000	11293.2115	17178.8334
floors	-12104.8633	1326.6535	-9.1244	0.0000	-14705.2556	-9504.4711
grade	60846.5179	1580.1560	38.5067	0.0000	57749.2318	63943.8039
sqft_living	67250.2568	2376.4388	28.2988	0.0000	62592.1656	71908.3479
sqft_lot	9365.9728	937.7743	9.9874	0.0000	7527.8282	11204.1173
sqft_above	39911.5148	2222.1844	17.9605	0.0000	35555.7800	44267.2497
sqft_living15	19925.5923	1527.5685	13.0440	0.0000	16931.3837	22919.8009
age	26046.8364	1361.8817	19.1256	0.0000	23377.3930	28716.2799
waterfront_1.0	202084.6090	14231.2373	14.2001	0.0000	174189.7607	229979.4573
view_1.0	63943.5463	7095.3703	9.0120	0.0000	50035.8112	77851.2814
view_2.0	62066.6453	4416.9405	14.0520	0.0000	53408.9380	70724.3526
view_3.0	126578.8679	6259.3671	20.2223	0.0000	114309.7944	138847.9414

view_4.0	232743.6470	9634.4752	24.1574	0.0000	213858.9766	251628.3174
zipcode_98002	18083.3900	10969.7134	1.6485	0.0993	-3418.4994	39585.2795
zipcode_98003	-10732.9458	9802.4315	-1.0949	0.2736	-29946.8295	8480.9380
zipcode_98004	604490.1179	10573.1827	57.1720	0.0000	583765.4739	625214.7620
zipcode_98005	324157.7242	11637.7807	27.8539	0.0000	301346.3467	346969.1017
zipcode_98006	278391.0557	8757.4928	31.7889	0.0000	261225.3710	295556.7405
zipcode_98007	252489.1211	12262.1559	20.5909	0.0000	228453.8969	276524.3453
zipcode_98008	246468.4813	9783.6846	25.1918	0.0000	227291.3435	265645.6190
zipcode_98010	92662.0778	13604.5976	6.8111	0.0000	65995.5147	119328.6409
zipcode_98011	130293.5051	10920.7432	11.9308	0.0000	108887.6027	151699.4075
zipcode_98014	109097.3924	13074.1722	8.3445	0.0000	83470.5236	134724.2613
zipcode_98019	97321.6691	11105.5641	8.7633	0.0000	75553.4966	119089.8415
zipcode_98022	2108.9947	10453.0625	0.2018	0.8401	-18380.2002	22598.1897
zipcode_98023	-27471.2224	8577.1069	-3.2029	0.0014	-44283.3302	-10659.1146
zipcode_98024	157925.3074	15438.6990	10.2292	0.0000	127663.6965	188186.9184
zipcode_98027	181070.4280	8984.2755	20.1541	0.0000	163460.2233	198680.6327
zipcode_98028	127914.0655	9631.9772	13.2801	0.0000	109034.2915	146793.8394
zipcode_98029	215295.7065	9424.8641	22.8434	0.0000	196821.8979	233769.5151
zipcode_98030	-229.8451	10093.0286	-0.0228	0.9818	-20013.3324	19553.6421
zipcode_98031	7773.6319	9957.0779	0.7807	0.4350	-11743.3765	27290.6404
zipcode_98032	2233.4944	12779.8054	0.1748	0.8613	-22816.3818	27283.3706
zipcode_98033	356306.0732	8952.0269	39.8017	0.0000	338759.0792	373853.0671
zipcode_98034	187312.1526	8392.0976	22.3201	0.0000	170862.6840	203761.6212
zipcode_98038	32437.2908	8244.4943	3.9344	0.0001	16277.1415	48597.4401
zipcode_98039	769263.3550	32845.6991	23.4205	0.0000	704882.0383	833644.6716
zipcode_98040	486518.9896	10366.7072	46.9309	0.0000	466199.0611	506838.9181
zipcode_98042	9277.9104	8394.3390	1.1053	0.2691	-7175.9515	25731.7723
zipcode_98045	91941.6536	10348.1579	8.8848	0.0000	71658.0839	112225.2234
zipcode_98052	239839.7751	8324.3411	28.8119	0.0000	223523.1170	256156.4332
zipcode_98053	221495.3075	9106.7335	24.3221	0.0000	203645.0710	239345.5440
zipcode_98055	40472.8370	9983.5267	4.0540	0.0001	20903.9858	60041.6882
zipcode_98056	107812.2603	8951.4107	12.0442	0.0000	90266.4742	125358.0463
zipcode_98058	27254.0802	8691.4381	3.1357	0.0017	10217.8701	44290.2904
zipcode_98059	98389.4726	8680.2326	11.3349	0.0000	81375.2266	115403.7186
zipcode_98065	105042.9946	9451.7508	11.1136	0.0000	86516.4850	123569.5042
zipcode_98070	101436.3874	13897.8706	7.2987	0.0000	74194.9757	128677.7991

zipcode_98072	171148.0184	9892.5241	17.3007	0.0000	151757.5428	190538.4939
zipcode_98074	190020.6787	8974.0801	21.1744	0.0000	172430.4581	207610.8992
zipcode_98075	205335.9257	9574.9644	21.4451	0.0000	186567.9034	224103.9481
zipcode_98077	157171.0170	11297.7556	13.9117	0.0000	135026.1273	179315.9068
zipcode_98092	-33360.3314	9425.2190	-3.5395	0.0004	-51834.8357	-14885.8270
zipcode_98102	423003.3542	14541.2742	29.0898	0.0000	394500.7982	451505.9101
zipcode_98103	319954.6182	8446.0075	37.8824	0.0000	303399.4801	336509.7562
zipcode_98105	435171.3154	10810.8428	40.2532	0.0000	413980.8304	456361.8005
zipcode_98106	123697.8068	9374.1905	13.1956	0.0000	105323.3242	142072.2895
zipcode_98107	316279.2056	10154.0617	31.1480	0.0000	296376.0863	336182.3248
zipcode_98108	112322.3049	11254.6699	9.9801	0.0000	90261.8681	134382.7416
zipcode_98109	448170.6164	14284.0813	31.3755	0.0000	420172.1878	476169.0450
zipcode_98112	495226.9137	10825.8621	45.7448	0.0000	474006.9892	516446.8382
zipcode_98115	323690.3075	8422.7027	38.4307	0.0000	307180.8494	340199.7656
zipcode_98116	288454.1668	9592.8165	30.0698	0.0000	269651.1523	307257.1813
zipcode_98117	306404.3153	8551.8281	35.8291	0.0000	289641.7567	323166.8739
zipcode_98118	161677.3203	8632.0457	18.7299	0.0000	144757.5263	178597.1144
zipcode_98119	433982.6952	11708.4665	37.0657	0.0000	411032.7652	456932.6251
zipcode_98122	309444.2409	10030.6007	30.8500	0.0000	289783.1195	329105.3623
zipcode_98125	194880.8540	9004.3545	21.6430	0.0000	177231.2920	212530.4159
zipcode_98126	186027.4394	9278.5243	20.0492	0.0000	167840.4734	204214.4055
zipcode_98133	155401.1995	8558.6783	18.1571	0.0000	138625.2138	172177.1851
zipcode_98136	242083.4090	10042.5939	24.1057	0.0000	222398.7795	261768.0385
zipcode_98144	241729.3931	9521.3387	25.3882	0.0000	223066.4832	260392.3029
zipcode_98146	106602.2643	9746.7006	10.9373	0.0000	87497.6194	125706.9092
zipcode_98148	59024.6897	19396.1937	3.0431	0.0023	21005.9376	97043.4417
zipcode_98155	136196.6564	8859.0986	15.3736	0.0000	118831.8125	153561.5002
zipcode_98166	87796.3425	10176.1123	8.6277	0.0000	67850.0017	107742.6832
zipcode_98168	50720.2114	9973.0893	5.0857	0.0000	31171.8187	70268.6041
zipcode_98177	213850.4913	10217.8439	20.9291	0.0000	193822.3517	233878.6309
zipcode_98178	47251.3190	10024.6763	4.7135	0.0000	27601.8100	66900.8279
zipcode_98188	31747.7845	12489.3040	2.5420	0.0110	7267.3240	56228.2449
zipcode_98198	14103.4792	9832.5882	1.4344	0.1515	-5169.5152	33376.4735
zipcode_98199	373131.8861	9762.0735	38.2226	0.0000	353997.1086	392266.6636
is_renovated_1	6030.0189	2099.9937	2.8714	0.0041	1913.7917	10146.2460

Omnibus:	3198.211	Durbin-Watson:	1.992
Prob(Omnibus):	0.000	Jarque-Bera (JB):	18128.931
Skew:	0.850	Prob(JB):	0.000
Kurtosis:	7.948	Condition No.:	143

5.2.7 7

In [96]:

```
# merging processed continuous values with encoded categories
X_train_8 = X_train_scaled.drop(['sqft_lot15',
                                 'grade'],
                                axis=1).merge(features_train,
                                              left_index=True, right_inde
X_test_8 = X_test_scaled.drop(['sqft_lot15',
                                 'grade'],
                                axis=1).merge(features_test,
                                              left_index=True, right_inde
# adding constant
X_train_8_int = sm.add_constant(X_train_8)
X_test_8_int = sm.add_constant(X_test_8)

# resetting indicies to ensure alignment
X_train_8_int.reset_index(drop=True, inplace=True)
y_train.reset_index(drop=True, inplace=True)

# fitting model
model_8 = sm.OLS(y_train ,X_train_8_int).fit()
model_8.summary2() #.summary2() supresses scientific notation and incre
```

Out[96]:

Model:	OLS	Adj. R-squared:	0.817
Dependent Variable:	price	AIC:	414360.0576
Date:	2021-09-29 09:13	BIC:	415012.3165
No. Observations:	15893	Log-Likelihood:	-2.0710e+05
Df Model:	84	F-statistic:	845.8
Df Residuals:	15808	Prob (F-statistic):	0.00
R-squared:	0.818	Scale:	1.2248e+10

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
const	305139.3207	6798.9082	44.8806	0.0000	291812.6851	318465.9563
bedrooms	-13462.3469	1161.7685	-11.5878	0.0000	-15739.5457	-11185.1481
bathrooms	15790.5951	1561.2976	10.1138	0.0000	12730.2737	18850.9164
floors	-10355.3447	1379.6710	-7.5057	0.0000	-13059.6573	-7651.0321
condition	14304.2930	993.8168	14.3933	0.0000	12356.2987	16252.2872
sqft_living	80074.3792	2440.7224	32.8077	0.0000	75290.2848	84858.4735
sqft_lot	9896.2727	974.4481	10.1558	0.0000	7986.2432	11806.3022
sqft_above	58391.8470	2279.9026	25.6116	0.0000	53922.9779	62860.7161
sqft_living15	35489.8060	1542.5995	23.0065	0.0000	32466.1351	38513.4769
age	4511.8858	1409.5167	3.2010	0.0014	1749.0723	7274.6992
waterfront_1.0	204629.7886	14789.5353	13.8361	0.0000	175640.6124	233618.9648
view_1.0	63232.1844	7372.6608	8.5766	0.0000	48780.9282	77683.4406
view_2.0	70082.0849	4584.8048	15.2857	0.0000	61095.3445	79068.8254
view_3.0	141643.8855	6492.0396	21.8181	0.0000	128918.7474	154369.0237

view_4.0	246688.3212	10002.4947	24.6627	0.0000	227082.2907	266294.3516
zipcode_98002	1190.5264	11401.4413	0.1044	0.9168	-21157.5990	23538.6517
zipcode_98003	3444.1524	10178.9242	0.3384	0.7351	-16507.7001	23396.0049
zipcode_98004	629493.1379	10963.9771	57.4147	0.0000	608002.4921	650983.7837
zipcode_98005	347612.3717	12073.2291	28.7920	0.0000	323947.4656	371277.2778
zipcode_98006	301905.7315	9074.2309	33.2707	0.0000	284119.2040	319692.2591
zipcode_98007	276887.7849	12721.2303	21.7658	0.0000	251952.7226	301822.8472
zipcode_98008	261312.9076	10156.7437	25.7280	0.0000	241404.5315	281221.2837
zipcode_98010	73860.2448	14136.4305	5.2248	0.0000	46151.2286	101569.2610
zipcode_98011	137709.6165	11348.2115	12.1349	0.0000	115465.8276	159953.4055
zipcode_98014	100908.7171	13583.9109	7.4285	0.0000	74282.7023	127534.7320
zipcode_98019	88314.3282	11537.2123	7.6547	0.0000	65700.0761	110928.5804
zipcode_98022	-12336.2176	10870.7627	-1.1348	0.2565	-33644.1525	8971.7172
zipcode_98023	-17272.2511	8908.2754	-1.9389	0.0525	-34733.4870	188.9848
zipcode_98024	146273.8370	16038.1651	9.1204	0.0000	114837.2040	177710.4700
zipcode_98027	189923.0678	9331.7360	20.3524	0.0000	171631.8008	208214.3348
zipcode_98028	135403.4167	10007.8112	13.5298	0.0000	115786.9652	155019.8683
zipcode_98029	238469.3945	9773.4893	24.3996	0.0000	219312.2407	257626.5484
zipcode_98030	-3854.3523	10486.7428	-0.3675	0.7132	-24409.5644	16700.8598
zipcode_98031	6183.9308	10347.6499	0.5976	0.5501	-14098.6432	26466.5048
zipcode_98032	12603.4082	13275.5340	0.9494	0.3424	-13418.1526	38624.9690
zipcode_98033	369893.2822	9293.1988	39.8026	0.0000	351677.5525	388109.0119
zipcode_98034	197195.8195	8716.1748	22.6241	0.0000	180111.1228	214280.5162
zipcode_98038	19544.9087	8559.6808	2.2834	0.0224	2766.9581	36322.8593
zipcode_98039	790478.8004	34123.0572	23.1655	0.0000	723593.7160	857363.8848
zipcode_98040	512024.5156	10748.1814	47.6382	0.0000	490956.8542	533092.1771
zipcode_98042	-848.5439	8727.8810	-0.0972	0.9226	-17956.1862	16259.0983
zipcode_98045	95622.4548	10754.5719	8.8913	0.0000	74542.2671	116702.6425
zipcode_98052	257386.3299	8636.0359	29.8038	0.0000	240458.7145	274313.9454
zipcode_98053	218581.7021	9462.3405	23.1002	0.0000	200034.4353	237128.9689
zipcode_98055	43848.6259	10375.8488	4.2260	0.0000	23510.7788	64186.4731
zipcode_98056	97719.9414	9311.7116	10.4943	0.0000	79467.9245	115971.9582
zipcode_98058	32762.3229	9029.7247	3.6283	0.0003	15063.0326	50461.6132
zipcode_98059	89839.1040	9021.9065	9.9579	0.0000	72155.1382	107523.0698
zipcode_98065	82128.6790	9798.8249	8.3815	0.0000	62921.8645	101335.4934
zipcode_98070	93330.1242	14445.5871	6.4608	0.0000	65015.1257	121645.1228

zipcode_98072	182829.8448	10274.5825	17.7944	0.0000	162690.4911	202969.1986
zipcode_98074	216397.7312	9301.2095	23.2655	0.0000	198166.2996	234629.1628
zipcode_98075	218835.7156	9941.3077	22.0128	0.0000	199349.6186	238321.8125
zipcode_98077	174442.7804	11728.1656	14.8738	0.0000	151454.2381	197431.3226
zipcode_98092	-32835.2976	9794.1164	-3.3526	0.0008	-52032.8829	-13637.7124
zipcode_98102	480292.1621	15052.1843	31.9085	0.0000	450788.1640	509796.1602
zipcode_98103	355998.5226	8731.8001	40.7703	0.0000	338883.1985	373113.8467
zipcode_98105	481597.9764	11174.4193	43.0983	0.0000	459694.8400	503501.1128
zipcode_98106	139956.5494	9745.8889	14.3606	0.0000	120853.4956	159059.6032
zipcode_98107	352651.2970	10516.3399	33.5337	0.0000	332038.0714	373264.5227
zipcode_98108	129164.5622	11694.5582	11.0448	0.0000	106241.8942	152087.2302
zipcode_98109	506024.4595	14793.7853	34.2052	0.0000	477026.9529	535021.9662
zipcode_98112	553573.0472	11164.0907	49.5851	0.0000	531690.1561	575455.9383
zipcode_98115	355809.4512	8722.6388	40.7915	0.0000	338712.0842	372906.8182
zipcode_98116	322878.0779	9933.0872	32.5053	0.0000	303408.0939	342348.0619
zipcode_98117	339476.7616	8848.2768	38.3664	0.0000	322133.1297	356820.3935
zipcode_98118	180608.8066	8974.9288	20.1237	0.0000	163016.9225	198200.6907
zipcode_98119	492109.4748	12097.8169	40.6775	0.0000	468396.3737	515822.5758
zipcode_98122	361657.7573	10364.9932	34.8922	0.0000	341341.1884	381974.3263
zipcode_98125	216106.4341	9352.5633	23.1067	0.0000	197774.3433	234438.5249
zipcode_98126	211900.1965	9626.2013	22.0129	0.0000	193031.7440	230768.6490
zipcode_98133	171772.4247	8881.5028	19.3405	0.0000	154363.6662	189181.1833
zipcode_98136	274744.1785	10413.8235	26.3826	0.0000	254331.8965	295156.4605
zipcode_98144	268157.5299	9877.9361	27.1471	0.0000	248795.6484	287519.4114
zipcode_98146	116850.8117	10138.9305	11.5250	0.0000	96977.3515	136724.2719
zipcode_98148	70318.8006	20162.8384	3.4875	0.0005	30797.3374	109840.2637
zipcode_98155	149297.3619	9200.1439	16.2277	0.0000	131264.0305	167330.6933
zipcode_98166	98546.9349	10569.8744	9.3234	0.0000	77828.7755	119265.0943
zipcode_98168	54865.8741	10378.7507	5.2864	0.0000	34522.3388	75209.4094
zipcode_98177	241067.7528	10598.3086	22.7459	0.0000	220293.8590	261841.6466
zipcode_98178	52645.5111	10427.4697	5.0487	0.0000	32206.4811	73084.5411
zipcode_98188	40858.7650	12981.4025	3.1475	0.0016	15413.7354	66303.7947
zipcode_98198	20838.0323	10216.2350	2.0397	0.0414	813.0465	40863.0181
zipcode_98199	416143.4856	10088.5264	41.2492	0.0000	396368.8231	435918.1482
is_renovated_1	9259.7315	2187.1459	4.2337	0.0000	4972.6762	13546.7869

Omnibus:	3247.326	Durbin-Watson:	2.011
Prob(Omnibus):	0.000	Jarque-Bera (JB):	18244.169
Skew:	0.868	Prob(JB):	0.000
Kurtosis:	7.954	Condition No.:	134

5.2.8 8

In [97]:

```
# merging processed continuous values with encoded categories
X_train_9 = X_train_scaled.drop(['sqft_lot15',
                                 'sqft_living'],
                                axis=1).merge(features_train,
                                              left_index=True, right_inde
X_test_9 = X_test_scaled.drop(['sqft_lot15',
                                 'sqft_living'],
                                axis=1).merge(features_test,
                                              left_index=True, right_inde
# adding constant
X_train_9_int = sm.add_constant(X_train_9)
X_test_9_int = sm.add_constant(X_test_9)

# resetting indicies to ensure alignment
X_train_9_int.reset_index(drop=True, inplace=True)
y_train.reset_index(drop=True, inplace=True)

# fitting mode
model_9 = sm.OLS(y_train ,X_train_9_int).fit()
model_9.summary2() #.summary2() supresses scientific notation and incre
```

Out[97]:

Model:	OLS	Adj. R-squared:	0.825
Dependent Variable:	price	AIC:	413667.8217
Date:	2021-09-29 09:13	BIC:	414320.0805
No. Observations:	15893	Log-Likelihood:	-2.0675e+05
Df Model:	84	F-statistic:	891.8
Df Residuals:	15808	Prob (F-statistic):	0.00
R-squared:	0.826	Scale:	1.1726e+10

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
const	314822.7222	6661.0473	47.2632	0.0000	301766.3097	327879.1346
bedrooms	1029.3382	1091.2422	0.9433	0.3456	-1109.6210	3168.2974
bathrooms	28818.7395	1407.4829	20.4754	0.0000	26059.9125	31577.5666
floors	-21962.0289	1289.1610	-17.0359	0.0000	-24488.9315	-19435.1263
condition	15649.9736	969.8521	16.1365	0.0000	13748.9529	17550.9943
grade	67714.2310	1583.7656	42.7552	0.0000	64609.8698	70818.5922
sqft_lot	11146.4378	951.6204	11.7131	0.0000	9281.1533	13011.7222
sqft_above	80377.6167	1780.9815	45.1311	0.0000	76886.6897	83868.5436
sqft_living15	30216.9105	1519.4245	19.8871	0.0000	27238.6650	33195.1559
age	22581.2741	1439.5997	15.6858	0.0000	19759.4945	25403.0536
waterfront_1.0	201471.1103	14470.0600	13.9233	0.0000	173108.1423	229834.0784
view_1.0	72294.5145	7206.0248	10.0325	0.0000	58169.8840	86419.1451
view_2.0	71500.6325	4477.7406	15.9680	0.0000	62723.7502	80277.5148
view_3.0	139908.4596	6345.8637	22.0472	0.0000	127469.8430	152347.0762

view_4.0	248700.7771	9773.0107	25.4477	0.0000	229544.5614	267856.9928
zipcode_98002	9697.6293	11159.3543	0.8690	0.3849	-12175.9781	31571.2366
zipcode_98003	-12550.4359	9965.5907	-1.2594	0.2079	-32084.1304	6983.2585
zipcode_98004	604401.6086	10749.6369	56.2253	0.0000	583331.0942	625472.1229
zipcode_98005	320319.6623	11837.6132	27.0595	0.0000	297116.5901	343522.7345
zipcode_98006	277969.9419	8910.2221	31.1967	0.0000	260504.8902	295434.9936
zipcode_98007	246403.9885	12468.0875	19.7628	0.0000	221965.1149	270842.8620
zipcode_98008	244043.2328	9947.3633	24.5335	0.0000	224545.2660	263541.1995
zipcode_98010	89673.1088	13836.2893	6.4810	0.0000	62552.4035	116793.8142
zipcode_98011	135982.7520	11103.8805	12.2464	0.0000	114217.8798	157747.6243
zipcode_98014	117114.9001	13294.3464	8.8094	0.0000	91056.4648	143173.3354
zipcode_98019	102486.9168	11293.4571	9.0749	0.0000	80350.4528	124623.3809
zipcode_98022	-11910.8142	10635.1068	-1.1200	0.2628	-32756.8365	8935.2082
zipcode_98023	-26519.0455	8720.1917	-3.0411	0.0024	-43611.6159	-9426.4751
zipcode_98024	160305.2977	15697.0932	10.2124	0.0000	129537.2046	191073.3909
zipcode_98027	186072.9334	9131.4574	20.3771	0.0000	168174.2354	203971.6313
zipcode_98028	133940.1844	9792.0958	13.6784	0.0000	114746.5598	153133.8090
zipcode_98029	210167.9127	9579.8658	21.9385	0.0000	191390.2830	228945.5425
zipcode_98030	-1328.3072	10261.2445	-0.1294	0.8970	-21441.5169	18784.9025
zipcode_98031	3979.1723	10124.6628	0.3930	0.6943	-15866.3217	23824.6663
zipcode_98032	4877.2009	12992.4766	0.3754	0.7074	-20589.5353	30343.9371
zipcode_98033	354607.6164	9102.8092	38.9558	0.0000	336765.0722	372450.1607
zipcode_98034	187904.8149	8532.1074	22.0233	0.0000	171180.9112	204628.7186
zipcode_98038	28816.2017	8381.0623	3.4383	0.0006	12388.3636	45244.0398
zipcode_98039	749964.3423	33388.8392	22.4615	0.0000	684518.4091	815410.2755
zipcode_98040	477199.9486	10553.1239	45.2188	0.0000	456514.6220	497885.2752
zipcode_98042	1235.7640	8540.1671	0.1447	0.8849	-15503.9376	17975.4655
zipcode_98045	94182.6900	10522.8661	8.9503	0.0000	73556.6723	114808.7077
zipcode_98052	238885.9978	8463.1457	28.2266	0.0000	222297.2669	255474.7287
zipcode_98053	220551.7627	9258.7864	23.8208	0.0000	202403.4853	238700.0401
zipcode_98055	44165.2210	10152.3014	4.3503	0.0000	24265.5521	64064.8898
zipcode_98056	103079.4759	9111.4774	11.3131	0.0000	85219.9409	120939.0108
zipcode_98058	24577.7342	8837.3081	2.7811	0.0054	7255.6023	41899.8660
zipcode_98059	90983.3298	8827.3673	10.3070	0.0000	73680.6831	108285.9766
zipcode_98065	107094.9920	9610.2809	11.1438	0.0000	88257.7452	125932.2387
zipcode_98070	95932.9717	14134.5897	6.7871	0.0000	68227.5636	123638.3799

zipcode_98072	174504.1451	10057.0384	17.3514	0.0000	154791.2028	194217.0875
zipcode_98074	189163.8257	9124.0154	20.7325	0.0000	171279.7147	207047.9366
zipcode_98075	195115.1571	9729.3617	20.0543	0.0000	176044.4984	214185.8159
zipcode_98077	149434.0164	11486.3413	13.0097	0.0000	126919.4771	171948.5556
zipcode_98092	-38489.2106	9582.5696	-4.0166	0.0001	-57272.1399	-19706.2812
zipcode_98102	438961.6361	14785.8002	29.6881	0.0000	409979.7811	467943.4911
zipcode_98103	334208.4643	8578.4887	38.9589	0.0000	317393.6479	351023.2807
zipcode_98105	443449.6711	10992.0063	40.3429	0.0000	421904.0850	464995.2572
zipcode_98106	138489.0978	9534.1265	14.5256	0.0000	119801.1225	157177.0732
zipcode_98107	326762.5274	10323.3363	31.6528	0.0000	306527.6109	346997.4440
zipcode_98108	128544.2838	11439.0313	11.2373	0.0000	106122.4777	150966.0899
zipcode_98109	465427.7443	14530.7701	32.0305	0.0000	436945.7775	493909.7111
zipcode_98112	510706.0614	11007.0383	46.3981	0.0000	489131.0108	532281.1120
zipcode_98115	341173.7798	8552.4805	39.8918	0.0000	324409.9425	357937.6172
zipcode_98116	304560.8707	9742.5568	31.2609	0.0000	285464.3480	323657.3933
zipcode_98117	321036.7446	8683.8749	36.9693	0.0000	304015.3592	338058.1299
zipcode_98118	179013.6229	8778.3770	20.3926	0.0000	161807.0026	196220.2431
zipcode_98119	449813.7286	11910.0024	37.7677	0.0000	426468.7654	473158.6918
zipcode_98122	323007.9139	10208.7302	31.6404	0.0000	302997.6383	343018.1895
zipcode_98125	208032.9525	9156.1441	22.7206	0.0000	190085.8657	225980.0393
zipcode_98126	201447.8130	9427.0258	21.3692	0.0000	182969.7671	219925.8590
zipcode_98133	162618.2513	8696.8956	18.6984	0.0000	145571.3440	179665.1586
zipcode_98136	256859.8337	10209.4187	25.1591	0.0000	236848.2085	276871.4588
zipcode_98144	259965.6960	9668.7109	26.8873	0.0000	241013.9198	278917.4722
zipcode_98146	119420.9549	9918.3710	12.0404	0.0000	99979.8165	138862.0933
zipcode_98148	63712.7868	19728.1441	3.2295	0.0012	25043.3741	102382.1995
zipcode_98155	142032.3256	9005.7928	15.7712	0.0000	124379.9444	159684.7067
zipcode_98166	94054.4650	10343.4973	9.0931	0.0000	73780.0305	114328.8996
zipcode_98168	68805.0177	10148.8525	6.7796	0.0000	48912.1092	88697.9263
zipcode_98177	223031.6373	10387.4817	21.4712	0.0000	202670.9883	243392.2862
zipcode_98178	64360.8071	10195.4377	6.3127	0.0000	44376.5863	84345.0280
zipcode_98188	42255.8444	12700.4889	3.3271	0.0009	17361.4376	67150.2512
zipcode_98198	17737.0149	9996.8000	1.7743	0.0760	-1857.8534	37331.8832
zipcode_98199	391114.3660	9912.2070	39.4578	0.0000	371685.3096	410543.4223
is_renovated_1	8949.3158	2140.0210	4.1819	0.0000	4754.6306	13144.0011

Omnibus:	3522.093	Durbin-Watson:	1.997
Prob(Omnibus):	0.000	Jarque-Bera (JB):	18683.544
Skew:	0.967	Prob(JB):	0.000
Kurtosis:	7.947	Condition No.:	132

5.2.9 9

In [98]:

```
# merging processed continuous values with encoded categories
X_train_10 = X_train_scaled.drop(['sqft_lot15',
                                  'sqft_above'],
                                 axis=1).merge(features_train,
                                               left_index=True, right_inde
X_test_10 = X_test_scaled.drop(['sqft_lot15',
                                'sqft_above'],
                               axis=1).merge(features_test,
                                             left_index=True, right_inde
# adding constant
X_train_10_int = sm.add_constant(X_train_10)
X_test_10_int = sm.add_constant(X_test_10)

# resetting indicies to ensure alignment
X_train_10_int.reset_index(drop=True, inplace=True)
y_train.reset_index(drop=True, inplace=True)

# fitting mode
model_10 = sm.OLS(y_train ,X_train_10_int).fit()
model_10.summary2() #.summary2() supresses scientific notation and incr
```

Out[98]:

Model:	OLS	Adj. R-squared:	0.829
Dependent Variable:	price	AIC:	413304.7820
Date:	2021-09-29 09:13	BIC:	413957.0409
No. Observations:	15893	Log-Likelihood:	-2.0657e+05
Df Model:	84	F-statistic:	916.8
Df Residuals:	15808	Prob (F-statistic):	0.00
R-squared:	0.830	Scale:	1.1461e+10

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
const	326092.9367	6578.4762	49.5697	0.0000	313198.3729	338987.5005
bedrooms	-8486.7097	1132.0803	-7.4966	0.0000	-10705.7162	-6267.7031
bathrooms	9056.4161	1499.4814	6.0397	0.0000	6117.2616	11995.5706
floors	1284.7067	1162.4755	1.1051	0.2691	-993.8779	3563.2913
condition	12046.0272	957.4645	12.5812	0.0000	10169.2876	13922.7668
grade	66012.0675	1561.8974	42.2640	0.0000	62950.5704	69073.5646
sqft_living	93052.7115	1880.2826	49.4887	0.0000	89367.1432	96738.2798
sqft_lot	10112.2846	942.2088	10.7325	0.0000	8265.4479	11959.1213
sqft_living15	25143.3360	1524.8117	16.4895	0.0000	22154.5313	28132.1408
age	23729.1486	1414.6792	16.7735	0.0000	20956.2160	26502.0812
waterfront_1.0	207313.8154	14306.4030	14.4910	0.0000	179271.6337	235355.9971
view_1.0	56115.1398	7121.8986	7.8792	0.0000	42155.4062	70074.8734
view_2.0	55070.5912	4423.0288	12.4509	0.0000	46400.9504	63740.2321
view_3.0	115671.9854	6262.9487	18.4693	0.0000	103395.8917	127948.0792

view_4.0	215176.5005	9650.8373	22.2961	0.0000	196259.7586	234093.2425
zipcode_98002	18125.4584	11029.0764	1.6434	0.1003	-3492.7893	39743.7061
zipcode_98003	-10713.6791	9852.7984	-1.0874	0.2769	-30026.2878	8598.9296
zipcode_98004	596048.7618	10620.0733	56.1247	0.0000	575232.2067	616865.3169
zipcode_98005	311773.6951	11697.3452	26.6534	0.0000	288845.5642	334701.8259
zipcode_98006	262985.7212	8796.0566	29.8981	0.0000	245744.4471	280226.9954
zipcode_98007	245075.2121	12324.9177	19.8845	0.0000	220916.9676	269233.4567
zipcode_98008	241582.7220	9832.8610	24.5689	0.0000	222309.1930	260856.2511
zipcode_98010	92600.2376	13676.6904	6.7707	0.0000	65792.3645	119408.1107
zipcode_98011	128710.9120	10976.1195	11.7264	0.0000	107196.4659	150225.3581
zipcode_98014	116934.6261	13142.6673	8.8973	0.0000	91173.4991	142695.7532
zipcode_98019	100374.2898	11165.3270	8.9898	0.0000	78488.9754	122259.6042
zipcode_98022	2162.0412	10511.4541	0.2057	0.8370	-18441.6078	22765.6903
zipcode_98023	-28988.1602	8620.5645	-3.3627	0.0008	-45885.4498	-12090.8705
zipcode_98024	166777.7912	15515.1407	10.7494	0.0000	136366.3456	197189.2368
zipcode_98027	170524.6229	9020.9333	18.9032	0.0000	152842.5648	188206.6811
zipcode_98028	124603.4031	9678.5553	12.8742	0.0000	105632.3307	143574.4755
zipcode_98029	212501.9725	9471.8087	22.4352	0.0000	193936.1471	231067.7979
zipcode_98030	985.3736	10144.8351	0.0971	0.9226	-18899.6604	20870.4075
zipcode_98031	3663.3490	10009.4591	0.3660	0.7144	-15956.3326	23283.0305
zipcode_98032	-5378.9885	12842.1439	-0.4189	0.6753	-30551.0555	19793.0784
zipcode_98033	351640.5787	8998.6449	39.0771	0.0000	334002.2083	369278.9490
zipcode_98034	183999.1399	8432.9507	21.8191	0.0000	167469.5945	200528.6852
zipcode_98038	35244.9489	8285.2745	4.2539	0.0000	19004.8658	51485.0319
zipcode_98039	763680.5064	33014.7955	23.1315	0.0000	698967.7414	828393.2714
zipcode_98040	471395.2502	10426.0182	45.2134	0.0000	450959.0653	491831.4351
zipcode_98042	7132.0375	8442.9250	0.8447	0.3983	-9417.0585	23681.1334
zipcode_98045	98697.8919	10401.5559	9.4888	0.0000	78309.6558	119086.1280
zipcode_98052	236575.0431	8365.5755	28.2796	0.0000	220177.5609	252972.5253
zipcode_98053	233603.1033	9137.3452	25.5658	0.0000	215692.8645	251513.3421
zipcode_98055	40972.0656	10036.1471	4.0824	0.0000	21300.0726	60644.0587
zipcode_98056	102937.1839	9007.8802	11.4275	0.0000	85280.7111	120593.6566
zipcode_98058	25514.2821	8736.9999	2.9203	0.0035	8388.7658	42639.7985
zipcode_98059	100165.5526	8723.8616	11.4818	0.0000	83065.7887	117265.3164
zipcode_98065	113146.5738	9495.5123	11.9158	0.0000	94534.2866	131758.8610
zipcode_98070	97639.0878	13973.8611	6.9873	0.0000	70248.7262	125029.4494

zipcode_98072	168928.9586	9942.3057	16.9909	0.0000	149440.9053	188417.0118
zipcode_98074	192603.2844	9020.7476	21.3511	0.0000	174921.5902	210284.9787
zipcode_98075	211342.4934	9616.9226	21.9761	0.0000	192492.2281	230192.7587
zipcode_98077	159154.0884	11354.6647	14.0166	0.0000	136897.6504	181410.5263
zipcode_98092	-31775.9157	9472.7664	-3.3544	0.0008	-50343.6184	-13208.2131
zipcode_98102	403628.9459	14560.9066	27.7200	0.0000	375087.9081	432169.9837
zipcode_98103	303511.0222	8431.5704	35.9970	0.0000	286984.1825	320037.8618
zipcode_98105	420063.5599	10826.0236	38.8013	0.0000	398843.3187	441283.8010
zipcode_98106	117698.3669	9410.7356	12.5068	0.0000	99252.2516	136144.4822
zipcode_98107	299790.0235	10154.9616	29.5215	0.0000	279885.1404	319694.9067
zipcode_98108	102330.0964	11291.0657	9.0629	0.0000	80198.3196	124461.8731
zipcode_98109	435667.6791	14327.6814	30.4074	0.0000	407583.7893	463751.5689
zipcode_98112	477527.2205	10820.2042	44.1329	0.0000	456318.3860	498736.0549
zipcode_98115	310474.2149	8422.8223	36.8611	0.0000	293964.5224	326983.9074
zipcode_98116	273709.2115	9599.6949	28.5123	0.0000	254892.7146	292525.7085
zipcode_98117	292910.3581	8557.1779	34.2298	0.0000	276137.3133	309683.4029
zipcode_98118	155579.2936	8663.2286	17.9586	0.0000	138598.3774	172560.2099
zipcode_98119	417597.6496	11714.6186	35.6476	0.0000	394635.6610	440559.6382
zipcode_98122	297614.2612	10045.0663	29.6279	0.0000	277924.7854	317303.7370
zipcode_98125	189243.3105	9040.2168	20.9335	0.0000	171523.4543	206963.1666
zipcode_98126	178401.2278	9310.7557	19.1608	0.0000	160151.0846	196651.3711
zipcode_98133	145423.2072	8587.6721	16.9339	0.0000	128590.3903	162256.0241
zipcode_98136	231512.4630	10067.3658	22.9963	0.0000	211779.2778	251245.6482
zipcode_98144	225952.8268	9518.9358	23.7372	0.0000	207294.6270	244611.0267
zipcode_98146	110355.0376	9806.9790	11.2527	0.0000	91132.2401	129577.8350
zipcode_98148	70116.3069	19504.0696	3.5950	0.0003	31886.1058	108346.5080
zipcode_98155	132082.1364	8900.4645	14.8399	0.0000	114636.2108	149528.0619
zipcode_98166	85027.7376	10227.0557	8.3140	0.0000	64981.5419	105073.9334
zipcode_98168	53575.9645	10036.6031	5.3381	0.0000	33903.0775	73248.8515
zipcode_98177	208231.8438	10262.6581	20.2902	0.0000	188115.8633	228347.8244
zipcode_98178	44485.1146	10074.8587	4.4155	0.0000	24737.2424	64232.9869
zipcode_98188	31539.2436	12555.2040	2.5120	0.0120	6929.6117	56148.8755
zipcode_98198	13757.9528	9883.1296	1.3921	0.1639	-5614.1086	33130.0142
zipcode_98199	356212.8031	9757.8445	36.5053	0.0000	337086.3149	375339.2913
is_renovated_1	8018.4024	2115.9219	3.7896	0.0002	3870.9542	12165.8506

Omnibus:	3312.093	Durbin-Watson:	1.991
Prob(Omnibus):	0.000	Jarque-Bera (JB):	18295.676
Skew:	0.892	Prob(JB):	0.000
Kurtosis:	7.944	Condition No.:	132

5.2.10 10

In [99]:

```
# merging processed continuous values with encoded categories
X_train_11 = X_train_scaled.drop(['sqft_lot15',
                                  'sqft_living15'],
                                axis=1).merge(features_train,
                                              left_index=True, right_inde
X_test_11 = X_test_scaled.drop(['sqft_lot15',
                                'sqft_living15'],
                                axis=1).merge(features_test,
                                              left_index=True, right_inde
# adding constant
X_train_11_int = sm.add_constant(X_train_11)
X_test_11_int = sm.add_constant(X_test_11)

# resetting indicies to ensure alignment
X_train_11_int.reset_index(drop=True, inplace=True)
y_train.reset_index(drop=True, inplace=True)

# fitting mode
model_11 = sm.OLS(y_train ,X_train_11_int).fit()
model_11.summary2() #.summary2() supresses scientific notation and incr
```

Out[99]:

Model:	OLS	Adj. R-squared:	0.831
Dependent Variable:	price	AIC:	413129.2522
Date:	2021-09-29 09:13	BIC:	413781.5111
No. Observations:	15893	Log-Likelihood:	-2.0648e+05
Df Model:	84	F-statistic:	929.0
Df Residuals:	15808	Prob (F-statistic):	0.00
R-squared:	0.832	Scale:	1.1335e+10

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
const	318254.7724	6550.7861	48.5827	0.0000	305414.4845	331095.0604
bedrooms	-8975.7603	1124.6905	-7.9806	0.0000	-11180.2821	-6771.2386
bathrooms	12479.1145	1503.8141	8.2983	0.0000	9531.4672	15426.7617
floors	-12771.4817	1322.9199	-9.6540	0.0000	-15364.5555	-10178.4078
condition	12912.6307	954.4090	13.5295	0.0000	11041.8802	14783.3812
grade	65819.4770	1532.4300	42.9510	0.0000	62815.7395	68823.2146
sqft_living	71846.8117	2326.1306	30.8868	0.0000	67287.3304	76406.2929
sqft_lot	9499.1158	937.4749	10.1327	0.0000	7661.5581	11336.6734
sqft_above	46929.4732	2211.0952	21.2245	0.0000	42595.4743	51263.4720
age	21120.1900	1416.7194	14.9078	0.0000	18343.2583	23897.1216
waterfront_1.0	203980.2093	14227.2428	14.3373	0.0000	176093.1907	231867.2280
view_1.0	72042.4758	7064.8905	10.1973	0.0000	58194.4845	85890.4670
view_2.0	69219.1763	4387.9258	15.7749	0.0000	60618.3412	77820.0115
view_3.0	136329.0673	6221.8438	21.9114	0.0000	124133.5438	148524.5909

view_4.0	243048.6551	9590.5540	25.3425	0.0000	224250.0753	261847.2349
zipcode_98002	7556.7526	10966.7042	0.6891	0.4908	-13939.2386	29052.7437
zipcode_98003	-10892.7486	9798.3997	-1.1117	0.2663	-30098.7296	8313.2324
zipcode_98004	613913.3916	10543.4097	58.2272	0.0000	593247.1059	634579.6773
zipcode_98005	328751.0070	11617.2609	28.2985	0.0000	305979.8506	351522.1634
zipcode_98006	283750.5753	8727.1900	32.5134	0.0000	266644.2875	300856.8631
zipcode_98007	253443.6946	12255.6433	20.6798	0.0000	229421.2358	277466.1533
zipcode_98008	245787.1160	9780.1194	25.1313	0.0000	226616.9664	264957.2656
zipcode_98010	90081.0974	13602.7009	6.6223	0.0000	63418.2521	116743.9428
zipcode_98011	139393.3596	10908.0348	12.7790	0.0000	118012.3672	160774.3520
zipcode_98014	111879.9385	13071.5530	8.5590	0.0000	86258.2037	137501.6733
zipcode_98019	105244.3259	11099.4590	9.4819	0.0000	83488.1202	127000.5317
zipcode_98022	-6570.9046	10458.9290	-0.6283	0.5298	-27071.5985	13929.7893
zipcode_98023	-25698.9696	8573.1793	-2.9976	0.0027	-42503.3790	-8894.5602
zipcode_98024	163603.0826	15431.5555	10.6019	0.0000	133355.4737	193850.6915
zipcode_98027	186590.1158	8964.9361	20.8133	0.0000	169017.8184	204162.4132
zipcode_98028	133415.8948	9625.0861	13.8613	0.0000	114549.6281	152282.1614
zipcode_98029	218894.0839	9418.2587	23.2415	0.0000	200433.2226	237354.9453
zipcode_98030	2320.4203	10087.9956	0.2300	0.8181	-17453.2017	22094.0423
zipcode_98031	7246.3625	9953.5854	0.7280	0.4666	-12263.8003	26756.5252
zipcode_98032	-2202.0469	12773.6217	-0.1724	0.8631	-27239.8025	22835.7087
zipcode_98033	356629.8711	8947.8476	39.8565	0.0000	339091.0692	374168.6729
zipcode_98034	188539.3560	8388.5620	22.4758	0.0000	172096.8176	204981.8944
zipcode_98038	37710.5912	8231.5985	4.5812	0.0000	21575.7192	53845.4633
zipcode_98039	786566.3636	32800.0077	23.9807	0.0000	722274.6073	850858.1199
zipcode_98040	491565.1761	10336.1077	47.5581	0.0000	471305.2260	511825.1262
zipcode_98042	7518.8584	8394.7423	0.8957	0.3704	-8935.7940	23973.5108
zipcode_98045	92780.7428	10344.9307	8.9687	0.0000	72503.4987	113057.9870
zipcode_98052	246116.1513	8307.7928	29.6247	0.0000	229831.9297	262400.3729
zipcode_98053	230195.0427	9089.0863	25.3265	0.0000	212379.3968	248010.6886
zipcode_98055	44155.5387	9981.6583	4.4237	0.0000	24590.3499	63720.7274
zipcode_98056	102090.2510	8958.4852	11.3959	0.0000	84530.5982	119649.9038
zipcode_98058	26695.1633	8688.3911	3.0725	0.0021	9664.9256	43725.4010
zipcode_98059	102257.2621	8663.8702	11.8027	0.0000	85275.0882	119239.4359
zipcode_98065	117941.8321	9415.3685	12.5265	0.0000	99486.6358	136397.0284
zipcode_98070	92332.4419	13894.3845	6.6453	0.0000	65097.8635	119567.0204

zipcode_98072	178917.9069	9874.9795	18.1183	0.0000	159561.8208	198273.9930
zipcode_98074	199919.7068	8951.5738	22.3335	0.0000	182373.6011	217465.8125
zipcode_98075	219699.5799	9504.9533	23.1142	0.0000	201068.7874	238330.3725
zipcode_98077	167112.5606	11257.7833	14.8442	0.0000	145046.0211	189179.1000
zipcode_98092	-32023.0717	9419.4807	-3.3997	0.0007	-50486.3283	-13559.8152
zipcode_98102	427960.2562	14541.5488	29.4302	0.0000	399457.1618	456463.3506
zipcode_98103	315471.4018	8428.5519	37.4289	0.0000	298950.4788	331992.3248
zipcode_98105	438733.7464	10808.8780	40.5901	0.0000	417547.1127	459920.3801
zipcode_98106	121687.0714	9362.0698	12.9979	0.0000	103336.3467	140037.7960
zipcode_98107	314598.3461	10145.9566	31.0073	0.0000	294711.1138	334485.5784
zipcode_98108	112900.6939	11250.9859	10.0347	0.0000	90847.4782	134953.9096
zipcode_98109	452277.9015	14286.9894	31.6566	0.0000	424273.7726	480282.0304
zipcode_98112	501841.4674	10828.0328	46.3465	0.0000	480617.2880	523065.6468
zipcode_98115	323379.2631	8418.2431	38.4141	0.0000	306878.5464	339879.9797
zipcode_98116	284631.2648	9578.7921	29.7147	0.0000	265855.7398	303406.7898
zipcode_98117	301606.5306	8534.4755	35.3398	0.0000	284877.9851	318335.0760
zipcode_98118	164507.8060	8636.3151	19.0484	0.0000	147579.6433	181435.9687
zipcode_98119	437135.9657	11710.7880	37.3276	0.0000	414181.4854	460090.4460
zipcode_98122	312162.6778	10034.1452	31.1100	0.0000	292494.6087	331830.7470
zipcode_98125	195602.0212	9002.1911	21.7283	0.0000	177956.6998	213247.3425
zipcode_98126	180012.1081	9251.7017	19.4572	0.0000	161877.7175	198146.4987
zipcode_98133	150429.3561	8548.6330	17.5969	0.0000	133673.0604	167185.6519
zipcode_98136	241794.4745	10037.3969	24.0894	0.0000	222120.0316	261468.9174
zipcode_98144	241508.7473	9516.7894	25.3771	0.0000	222854.7546	260162.7400
zipcode_98146	107078.6961	9743.6678	10.9896	0.0000	87979.9958	126177.3964
zipcode_98148	64280.3207	19395.8641	3.3141	0.0009	26262.2147	102298.4267
zipcode_98155	134855.4266	8853.9059	15.2312	0.0000	117500.7612	152210.0920
zipcode_98166	85170.4740	10169.7739	8.3749	0.0000	65236.5571	105104.3909
zipcode_98168	54689.3674	9980.6323	5.4795	0.0000	35126.1896	74252.5452
zipcode_98177	217488.0870	10215.4703	21.2901	0.0000	197464.6000	237511.5740
zipcode_98178	51352.0452	10029.7428	5.1200	0.0000	31692.6053	71011.4851
zipcode_98188	33146.6568	12486.3714	2.6546	0.0079	8671.9445	57621.3690
zipcode_98198	11880.6772	9825.5491	1.2092	0.2266	-7378.5199	31139.8743
zipcode_98199	373875.6957	9759.0957	38.3105	0.0000	354746.7549	393004.6365
is_renovated_1	7093.8445	2102.9374	3.3733	0.0007	2971.8473	11215.8417

Omnibus:	3234.960	Durbin-Watson:	1.993
Prob(Omnibus):	0.000	Jarque-Bera (JB):	18717.041
Skew:	0.855	Prob(JB):	0.000
Kurtosis:	8.034	Condition No.:	135

5.2.11 11

In [100]:

```
# merging processed continuous values with encoded categories
X_train_12 = X_train_scaled.drop(['sqft_lot15',
                                  'age'],
                                 axis=1).merge(features_train,
                                               left_index=True, right_inde
X_test_12 = X_test_scaled.drop(['sqft_lot15',
                                 'age'],
                                 axis=1).merge(features_test,
                                               left_index=True, right_inde
# adding constant
X_train_12_int = sm.add_constant(X_train_12)
X_test_12_int = sm.add_constant(X_test_12)

# resetting indicies to ensure alignment
X_train_12_int.reset_index(drop=True, inplace=True)
y_train.reset_index(drop=True, inplace=True)

# fitting mode
model_12 = sm.OLS(y_train ,X_train_12_int).fit()
model_12.summary2() #.summary2() supresses scientific notation and incr
```

Out[100]:

Model:	OLS	Adj. R-squared:	0.831
Dependent Variable:	price	AIC:	413142.3765
Date:	2021-09-29 09:13	BIC:	413794.6354
No. Observations:	15893	Log-Likelihood:	-2.0649e+05
Df Model:	84	F-statistic:	928.1
Df Residuals:	15808	Prob (F-statistic):	0.00
R-squared:	0.831	Scale:	1.1345e+10

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
const	309416.1699	6516.6213	47.4811	0.0000	296642.8489	322189.4909
bedrooms	-6789.4145	1121.8036	-6.0522	0.0000	-8988.2774	-4590.5516
bathrooms	5908.6799	1425.6762	4.1445	0.0000	3114.1920	8703.1679
floors	-17365.6298	1257.7210	-13.8072	0.0000	-19830.9064	-14900.3532

5.2.12 12

In [101]:

```
# merging processed continuous values with encoded categories
X_train_13 = X_train_scaled.drop(['sqft_lot15',
                                  'condition',
                                  'sqft_above'],
                                axis=1).merge(features_train,
                                              left_index=True, right_inde
X_test_13 = X_test_scaled.drop(['sqft_lot15',
                                'condition',
                                'sqft_above'],
                                axis=1).merge(features_test,
                                              left_index=True, right_inde
# adding constant
X_train_13_int = sm.add_constant(X_train_13)
X_test_13_int = sm.add_constant(X_test_13)

# resetting indicies to ensure alignment
X_train_13_int.reset_index(drop=True, inplace=True)
y_train.reset_index(drop=True, inplace=True)

# fitting mode
model_13 = sm.OLS(y_train ,X_train_13_int).fit()
model_13.summary2() #.summary2() supresses scientific notation and incr
```

Out[101]:

Model:	OLS	Adj. R-squared:	0.827
Dependent Variable:	price	AIC:	413461.1276
Date:	2021-09-29 09:13	BIC:	414105.7129
No. Observations:	15893	Log-Likelihood:	-2.0665e+05
Df Model:	83	F-statistic:	916.8
Df Residuals:	15809	Prob (F-statistic):	0.00
R-squared:	0.828	Scale:	1.1575e+10

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
const	326931.3479	6610.7811	49.4543	0.0000	313973.4630	339889.2329
bedrooms	-8237.9810	1137.5245	-7.2420	0.0000	-10467.6588	-6008.3032
bathrooms	10542.0645	1502.2422	7.0176	0.0000	7597.4984	13486.6305
floors	-233.9873	1161.9285	-0.2014	0.8404	-2511.4996	2043.5251

5.2.13 13

In [103]:

```
# merging processed continuous values with encoded categories
X_train_14 = X_train_scaled.drop(['sqft_lot',
                                  'sqft_lot15',
                                 'condition',
                                 'sqft_above',
                                 'floors'],
                              axis=1).merge(features_train,
                                            left_index=True, right_inde
X_test_14 = X_test_scaled.drop(['sqft_lot',
                                'sqft_lot15',
                                'condition',
                                'sqft_above',
                                'floors'],
                              axis=1).merge(features_test,
                                            left_index=True, right_inde
# adding constant
X_train_14_int = sm.add_constant(X_train_14)
X_test_14_int = sm.add_constant(X_test_14)

# resetting indicies to ensure alignment
X_train_14_int.reset_index(drop=True, inplace=True)
y_train.reset_index(drop=True, inplace=True)

# fitting mode
model_14 = sm.OLS(y_train ,X_train_14_int).fit()
model_14.summary2() #.summary2() supresses scientific notation and incr
```

Out[103]:

Model:	OLS	Adj. R-squared:	0.826
Dependent Variable:	price	AIC:	413567.2812
Date:	2021-09-29 09:14	BIC:	414196.5191
No. Observations:	15893	Log-Likelihood:	-2.0670e+05
Df Model:	81	F-statistic:	931.7
Df Residuals:	15811	Prob (F-statistic):	0.00
R-squared:	0.827	Scale:	1.1654e+10

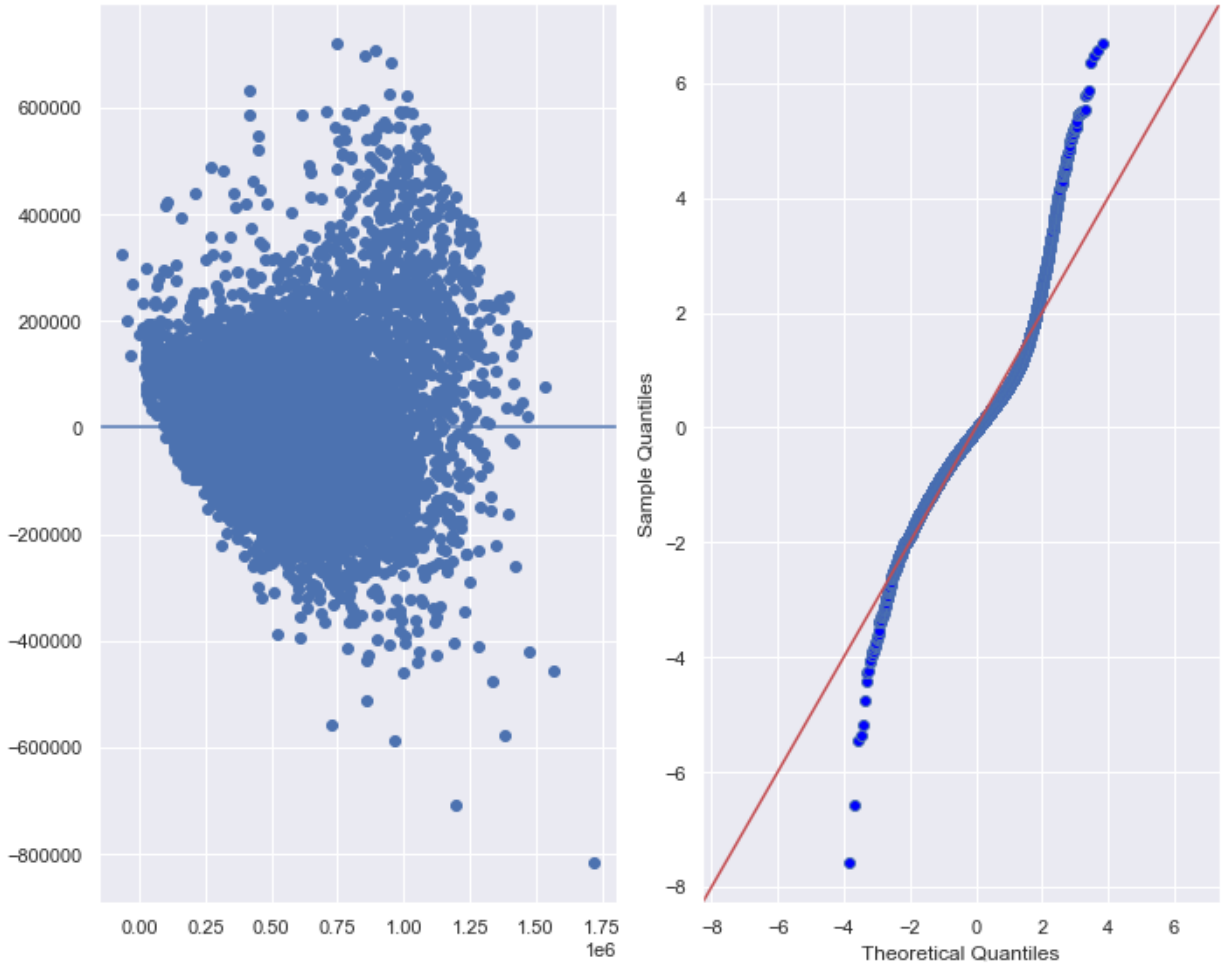
	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
const	327875.3472	6627.3998	49.4727	0.0000	314884.8879	340865.8065
bedrooms	-8885.1430	1139.6961	-7.7961	0.0000	-11119.0773	-6651.2088
bathrooms	10173.7692	1485.2447	6.8499	0.0000	7262.5203	13085.0181
grade	66161.8003	1562.3758	42.3469	0.0000	63099.3656	69224.2350
sqft_living	95645.8619	1885.1764	50.7358	0.0000	91950.7011	99341.0227
sqft_living15	23759.0735	1533.2468	15.4959	0.0000	20753.7350	26764.4120
age	29605.9172	1286.0121	23.0215	0.0000	27085.1868	32126.6475
waterfront_1.0	199868.0652	14414.0304	13.8662	0.0000	171614.9220	228121.2084
view_1.0	56533.1671	7177.4271	7.8765	0.0000	42464.5915	70601.7426
view_2.0	56294.5424	4456.9792	12.6306	0.0000	47558.3549	65030.7298

view_3.0	118893.1459	6307.0348	18.8509	0.0000	106530.6384	131255.6534
view_4.0	218945.3076	9727.3142	22.5083	0.0000	199878.6625	238011.9526
zipcode_98002	21139.9964	11113.4113	1.9022	0.0572	-643.5570	42923.5499
zipcode_98003	-12702.2966	9933.2243	-1.2788	0.2010	-32172.5489	6767.9558
zipcode_98004	594798.8720	10706.0509	55.5573	0.0000	573813.7914	615783.9526
zipcode_98005	315948.8757	11778.4145	26.8244	0.0000	292861.8401	339035.9112
zipcode_98006	265643.5361	8851.4313	30.0114	0.0000	248293.7214	282993.3508
zipcode_98007	246084.5180	12424.4815	19.8064	0.0000	221731.1175	270437.9185
zipcode_98008	241021.3911	9906.9335	24.3286	0.0000	221602.6718	260440.1105
zipcode_98010	105997.7121	13756.9952	7.7050	0.0000	79032.4328	132962.9914
zipcode_98011	124549.9996	11064.9586	11.2563	0.0000	102861.4189	146238.5803
zipcode_98014	133076.7722	13107.7956	10.1525	0.0000	107383.9981	158769.5464
zipcode_98019	102717.3972	11236.3623	9.1415	0.0000	80692.8457	124741.9487
zipcode_98022	22226.7709	10499.0004	2.1170	0.0343	1647.5328	42806.0089
zipcode_98023	-30700.7050	8689.3115	-3.5332	0.0004	-47732.7464	-13668.6637
zipcode_98024	182842.6069	15541.7542	11.7646	0.0000	152378.9963	213306.2175
zipcode_98027	175994.5860	9087.9315	19.3657	0.0000	158181.2039	193807.9682
zipcode_98028	121890.4037	9758.1778	12.4911	0.0000	102763.2624	141017.5450
zipcode_98029	210005.4167	9535.2452	22.0241	0.0000	191315.2488	228695.5846
zipcode_98030	-703.5760	10229.0567	-0.0688	0.9452	-20753.6936	19346.5416
zipcode_98031	5330.3238	10091.4000	0.5282	0.5974	-14449.9709	25110.6185
zipcode_98032	-4838.3718	12948.3580	-0.3737	0.7087	-30218.6300	20541.8865
zipcode_98033	351492.3686	9069.5613	38.7552	0.0000	333714.9942	369269.7430
zipcode_98034	182108.7831	8499.5602	21.4257	0.0000	165448.6759	198768.8902
zipcode_98038	37403.5824	8345.9349	4.4817	0.0000	21044.5983	53762.5665
zipcode_98039	764563.8548	33289.3353	22.9672	0.0000	699312.9613	829814.7482
zipcode_98040	475633.5930	10492.3141	45.3316	0.0000	455067.4609	496199.7251
zipcode_98042	11990.4911	8506.9235	1.4095	0.1587	-4684.0491	28665.0314
zipcode_98045	102366.7093	10467.1379	9.7798	0.0000	81849.9253	122883.4932
zipcode_98052	235063.9316	8433.6797	27.8720	0.0000	218532.9576	251594.9056
zipcode_98053	234905.3470	9206.9229	25.5140	0.0000	216858.7282	252951.9657
zipcode_98055	36995.5795	10115.7257	3.6572	0.0003	17167.6036	56823.5553
zipcode_98056	106570.7072	9070.2814	11.7494	0.0000	88791.9213	124349.4931
zipcode_98058	27600.5547	8809.0063	3.1332	0.0017	10333.8978	44867.2117
zipcode_98059	101881.2574	8792.3854	11.5874	0.0000	84647.1794	119115.3354
zipcode_98065	109539.4042	9559.7457	11.4584	0.0000	90801.2124	128277.5960

zipcode_98070	123520.7964	13931.7661	8.8661	0.0000	96212.9461	150828.6468
zipcode_98072	171359.0382	10022.0351	17.0982	0.0000	151714.7065	191003.3699
zipcode_98074	189350.4736	9092.8977	20.8240	0.0000	171527.3572	207173.5900
zipcode_98075	211046.2373	9696.0860	21.7661	0.0000	192040.8031	230051.6715
zipcode_98077	169409.3311	11422.0567	14.8318	0.0000	147020.7974	191797.8648
zipcode_98092	-27087.9523	9544.7279	-2.8380	0.0045	-45796.7073	-8379.1972
zipcode_98102	395339.3852	14603.4879	27.0716	0.0000	366714.8835	423963.8868
zipcode_98103	298403.4157	8389.1205	35.5703	0.0000	281959.7829	314847.0485
zipcode_98105	413836.3973	10867.9788	38.0785	0.0000	392533.9196	435138.8751
zipcode_98106	111092.6462	9474.0661	11.7260	0.0000	92522.3962	129662.8962
zipcode_98107	294349.9798	10156.3753	28.9818	0.0000	274442.3259	314257.6337
zipcode_98108	96130.1199	11374.5789	8.4513	0.0000	73834.6481	118425.5917
zipcode_98109	424729.8961	14393.0955	29.5093	0.0000	396517.7876	452942.0047
zipcode_98112	469166.5953	10834.0137	43.3050	0.0000	447930.6930	490402.4976
zipcode_98115	304390.1578	8464.4114	35.9612	0.0000	287798.9462	320981.3694
zipcode_98116	268293.1683	9650.0537	27.8022	0.0000	249377.9626	287208.3741
zipcode_98117	288081.2033	8592.8720	33.5256	0.0000	271238.1943	304924.2124
zipcode_98118	147404.4356	8707.0484	16.9293	0.0000	130337.6277	164471.2434
zipcode_98119	408106.0804	11738.5560	34.7663	0.0000	385097.1719	431114.9888
zipcode_98122	288246.9200	10045.9103	28.6930	0.0000	268555.7901	307938.0498
zipcode_98125	183092.2361	9102.2127	20.1151	0.0000	165250.8612	200933.6110
zipcode_98126	172420.5755	9368.7173	18.4039	0.0000	154056.8212	190784.3299
zipcode_98133	144060.8049	8647.6155	16.6590	0.0000	127110.4924	161011.1174
zipcode_98136	224786.6483	10125.3846	22.2003	0.0000	204939.7398	244633.5568
zipcode_98144	219824.3316	9546.4992	23.0267	0.0000	201112.1045	238536.5588
zipcode_98146	102608.4230	9873.5532	10.3922	0.0000	83255.1327	121961.7133
zipcode_98148	61032.6594	19657.7028	3.1048	0.0019	22501.3202	99563.9985
zipcode_98155	129558.4878	8973.4399	14.4380	0.0000	111969.5223	147147.4533
zipcode_98166	83195.3716	10311.5200	8.0682	0.0000	62983.6165	103407.1267
zipcode_98168	45379.8787	10101.2896	4.4925	0.0000	25580.1992	65179.5582
zipcode_98177	203261.0697	10343.2494	19.6516	0.0000	182987.1214	223535.0180
zipcode_98178	36891.1894	10147.6451	3.6354	0.0003	17000.6479	56781.7310
zipcode_98188	25910.8203	12655.3025	2.0474	0.0406	1104.9842	50716.6563
zipcode_98198	10936.0017	9964.2933	1.0975	0.2724	-8595.1494	30467.1527
zipcode_98199	350208.7538	9822.8096	35.6526	0.0000	330954.9268	369462.5807
is_renovated_1	5771.4082	2127.1498	2.7132	0.0067	1601.9520	9940.8644

Omnibus:	3182.381	Durbin-Watson:	1.991
Prob(Omnibus):	0.000	Jarque-Bera (JB):	16988.032
Skew:	0.863	Prob(JB):	0.000
Kurtosis:	7.762	Condition No.:	126

In [104]:
```
model_diagnostics(model_14, X_train_14_int, y_train)
```



In [105]:
```
train_test_validation(X_train_14_int, y_train, X_test_14_int)
```

Out[105]:
```
{'train': {'r2': 0.827,
  'mse': 11594317781.624,
  'mae': 75511.853,
  'rmse': 107676.914},
 'test': {'r2': 0.835,
  'mse': 11322373305.419,
  'mae': 75705.557,
  'rmse': 106406.641}}
```

6 Conclusions

Our 14th model, dropping the **'sqft_lot', 'sqft_lot15','floors', 'condition', and 'sqft_above'** variables, has a robust R2 score of .827, and a moderately improved distribution of the residuals. The model is neither over nor underfit.

Below we will examine the coefficients to make actionable suggestions to best answer the business questions.

In [106]:

```
def feature_selection(X_train, y_train, n):
    '''
    Performs a linear regression model and selects the top n
    features based on positive coefficient values'''

    # create and fit linear regression model
    linreg = LinearRegression()
    linreg.fit(X_train, y_train)

    # extracting top n features by recursive feature elimination
    selector = RFE(linreg, n_features_to_select=n)
    selector = selector.fit(X_train ,y_train)
    estimators = selector.estimator_

    # creating df
    coefs = pd.DataFrame(selector.support_ , index=X_train.columns)
    coefs = coefs.loc[coefs[0] == True]
    coefs['coef_values'] = estimators.coef_

    return coefs.drop(0, axis=1).sort_values(['coef_values'], ascending
```

In [107]:
```
feature_selection(X_train_14_int, y_train, 60)
```

Out[107]:

	coef_values
zipcode_98148	56575.59
grade	58393.85
view_1.0	72831.13
view_2.0	76056.80
zipcode_98166	83778.62
zipcode_98045	85006.55
zipcode_98019	87201.66
zipcode_98056	93704.53
zipcode_98059	93716.20
zipcode_98106	95678.49
zipcode_98146	95846.61
zipcode_98108	96255.02
zipcode_98010	97837.15
zipcode_98065	100885.05
sqft_living	108370.83
zipcode_98028	116447.76
zipcode_98070	117487.72
zipcode_98014	121767.76
zipcode_98011	122508.93
zipcode_98155	127148.79
zipcode_98133	137733.21
view_3.0	144114.80
zipcode_98118	153608.14
zipcode_98126	172081.08
zipcode_98034	173347.52
zipcode_98072	174585.22
zipcode_98027	175665.55
zipcode_98125	181300.81
zipcode_98024	184590.02
zipcode_98077	184686.61
zipcode_98074	194171.50
zipcode_98029	199680.28
waterfront_1.0	206484.18

	coef_values
zipcode_98177	211174.64
zipcode_98075	223126.30
zipcode_98144	224202.33
zipcode_98053	228478.86
zipcode_98136	229836.19
zipcode_98052	234141.49
zipcode_98008	239410.47
zipcode_98007	249761.06
view_4.0	250734.22
zipcode_98116	272911.17
zipcode_98006	278526.44
zipcode_98117	296953.69
zipcode_98107	298480.84
zipcode_98122	303519.30
zipcode_98103	304815.65
zipcode_98115	315199.78
zipcode_98005	334764.25
zipcode_98033	350682.37
zipcode_98199	364845.91
zipcode_98102	417209.57
zipcode_98119	430384.37
zipcode_98105	444007.71
zipcode_98109	448623.91
zipcode_98040	498498.35
zipcode_98112	503829.51
zipcode_98004	614697.19
zipcode_98039	808612.18

The results are a classic good news/bad news situation. We can, with a high level of certainty, explain the change in our dependent variable (**'price'**), by the changes in our independent variables. The bad news is that the explanation that the model offers makes it hard to suggest actionable renovations to existing houses that will have a large impact on home price. Of the top 60 features that we identified as having a positive impact on house price, the overwhelming majority are features that are outside of the control of the homeowner -- ie. location.

Of the features that are in our control, for every square foot of living space we add to a house we can expect an increase in the price of the house of 129.50USD (since our features have been scaled, we need to divide the coefficient by the standard deviation to arrive at the unit price.)

We can also offer advice on what type of room to build to increase the square foot of living space in a house. For each additional bathroom that we add to the house, we can expect an increase in house price of $13,889.68$USD, where as an additional bedroom would actually decrease the cost of the house by $9,356.22$USD. A caveat is that we would have to investigate the difference if size between bathrooms and bedrooms -- data which is not contained in this data set.

Also worth noting is the relative importance of the **'grade'** variable, suggesting that during renovation we should endeavor to conform to the Kings County guidelines and best practices for building to ensure a higher grade, which correlates well with house price.

If we are looking to purchase and renovate existing houses, the data is clear. Buying a house in the following zip codes has a large impact on the price of the house:

98039

98004

98112

98040

Any exploration of potential home purchases may want to focus on these zip codes. An area for further study would be to look at existing home prices in zip codes adjacent to the above -- there may be under priced houses that will go up in value as housing becomes more scarce in the highest demand areas.