

Final Project Submission



Boma Yangu Housing Agency House Price Prediction Model

Author: Nyaoke Brian Antony

- Student pace: Full time
- Scheduled project review date/time: 09/30/2022
- Instructor name: Antony Muiko
- Blog post URL: [My blog \(https://www.linkedin.com/in/africandataScientist/\)](https://www.linkedin.com/in/africandataScientist/)

Overview

This project employs the use of various Linear Regression models to try and predict house prices. Linear Regression is used here due to its ability to perform inference statistic giving us more insight and relationship about the data we are working on. The data used to develop this model is the King County House Sales Dataset which is a CSV file that contains 21597 rows and 21 columns of data. Each column in the data set represents an attribute about a house and the target variable Price which we will develop a model to try to predict.

The other 20 columns of this dataset will therefore be used to try and develop an accurate model to help the managers of Boma Yangu Housing Agency predict the price of a house given certain parameters.

We will start developing the model by using a simple linear regressor and continue to explore by incorporating multiple linear regressors to increase accuracy of the base model while also adding and removing other features from the model.

Business Problem

The recent burst in real estate bubble in the country has made many real estate agencies run into multiple losses. This has been particularly worsened by the inability of some real estate agencies to provide a reasonable price of a given housing unit. These problems have led to the recent undervaluation and sometimes over valuation of houses more than the current market prices.

Boma Yangu Housing Agency therefor needs a model that will help them predict the true value of a house not by guessing but by making data driven decisions from already available data.

This model should be accurate enough to give a price prediction when certain parameters are fed into the model. This will really help the agency avoid losses by investing in overpriced houses due to the current bubble and make profit by buying and investing in underpriced houses in the current bubble which have a potential of selling at high prices after the bubble burst.

Data Understanding

The data used to develop this regression model contains house sale prices for King County, which includes Seattle. It includes homes sold between September 9 2014 and January 1 2015

The data set contains 21597 records and 21 attributes.

We will get a better understanding on the data by investigating the column_names.md file found in the data directory which will give us more idea about the data and the attributes in each column as shown below.

Column Names and Descriptions for King County Data Set

- id - Unique identifier for a house
- date - Date house was sold
- price - Sale price (prediction target)
- bedrooms - Number of bedrooms
- bathrooms - Number of bathrooms
- sqft_living - Square footage of living space in the home
- sqft_lot - Square footage of the lot
- floors - Number of floors (levels) in house
- waterfront - Whether the house is on a waterfront
 - Includes Duwamish, Elliott Bay, Puget Sound, Lake Union, Ship Canal, Lake Washington, Lake Sammamish, other lake, and river/slough waterfronts
- view - Quality of view from house
 - Includes views of Mt. Rainier, Olympics, Cascades, Territorial, Seattle Skyline, Puget Sound, Lake Washington, Lake Sammamish, small lake / river / creek, and other
- condition - How good the overall condition of the house is. Related to maintenance of house.
 - See the [King County Assessor Website](https://info.kingcounty.gov/assessor/esales/Glossary.aspx?type=r) (<https://info.kingcounty.gov/assessor/esales/Glossary.aspx?type=r>) for further explanation of each condition code
- grade - Overall grade of the house. Related to the construction and design of the house.
 - See the [King County Assessor Website](https://info.kingcounty.gov/assessor/esales/Glossary.aspx?type=r) (<https://info.kingcounty.gov/assessor/esales/Glossary.aspx?type=r>) for further explanation of each building grade code
- sqft_above - Square footage of house apart from basement
- sqft_basement - Square footage of the basement
- yr_built - Year when house was built
- yr_renovated - Year when house was renovated

- zipcode - ZIP Code used by the United States Postal Service
- lat - Latitude coordinate
- long - Longitude coordinate
- sqft_living15 - The square footage of interior housing living space for the nearest 15 neighbors
- sqft_lot15 - The square footage of the land lots of the nearest 15 neighbors

```
In [2]: #Import the relevant packages for this module
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
from scipy.stats import norm
import statsmodels.api as sm
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_absolute_error
%matplotlib inline
```

```
In [3]: #Read the metadata contained in the column_names.md
with open('data/column_names.md') as f:
    print(f.read())

# Column Names and Descriptions for King County Data Set
* `id` - Unique identifier for a house
* `date` - Date house was sold
* `price` - Sale price (prediction target)
* `bedrooms` - Number of bedrooms
* `bathrooms` - Number of bathrooms
* `sqft_living` - Square footage of living space in the home
* `sqft_lot` - Square footage of the lot
* `floors` - Number of floors (levels) in house
* `waterfront` - Whether the house is on a waterfront
    * Includes Duwamish, Elliott Bay, Puget Sound, Lake Union, Ship Canal, Lake Washington, Lake Sammamish, other lake, and river/slough waterfronts
* `view` - Quality of view from house
    * Includes views of Mt. Rainier, Olympics, Cascades, Territorial, Seattle Skyline, Puget Sound, Lake Washington, Lake Sammamish, small lake / river / creek, and other
* `condition` - How good the overall condition of the house is. Related to main
tenance of house.
    * See the [King County Assessor Website](https://info.kingcounty.gov/assessor/esales/Glossary.aspx?type=r) for further explanation of each condition code
* `grade` - Overall grade of the house. Related to the construction and design
of the house.
    * See the [King County Assessor Website](https://info.kingcounty.gov/assessor/esales/Glossary.aspx?type=r) for further explanation of each building grade c
ode
* `sqft_above` - Square footage of house apart from basement
* `sqft_basement` - Square footage of the basement
* `yr_built` - Year when house was built
* `yr_renovated` - Year when house was renovated
* `zipcode` - ZIP Code used by the United States Postal Service
* `lat` - Latitude coordinate
* `long` - Longitude coordinate
* `sqft_living15` - The square footage of interior housing living space for the
nearest 15 neighbors
* `sqft_lot15` - The square footage of the land lots of the nearest 15 neighbor
s
```

The data in column_name.md file explains the attribute found in each of the 21 columns found on the dataset

Lets load the data set to confirm

```
In [4]: #Loading the data
data = pd.read_csv('data/kc_house_data.csv', index_col=0)

#Get the shape of the data and display the first 3 rows
print(data.shape)
data.head(3)

(21597, 20)
```

Out[4]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront		
	id									
1	7129300520	10/13/2014	221900.0	3	1.00	1180	5650	1.0	NaN	N
2	6414100192	12/9/2014	538000.0	3	2.25	2570	7242	2.0	NO	N
3	5631500400	2/25/2015	180000.0	2	1.00	770	10000	1.0	NO	N

```
In [5]: #check the data type of every column
data.dtypes
```

Out[5]:

date	object
price	float64
bedrooms	int64
bathrooms	float64
sqft_living	int64
sqft_lot	int64
floors	float64
waterfront	object
view	object
condition	object
grade	object
sqft_above	int64
sqft_basement	object
yr_built	int64
yr_renovated	float64
zipcode	int64
lat	float64
long	float64
sqft_living15	int64
sqft_lot15	int64
dtype:	object

Some of the data in the data set contain object which cannot be fed directly to a linear model since they may be categorical values such as the grade column.

The sqft_basement column seem to have integer values but loaded as an object.

In [6]: #Check the distribution of the data set that contains integer values
data.describe()

Out[6]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	sqf
count	2.159700e+04	21597.000000	21597.000000	21597.000000	2.159700e+04	21597.000000	21597
mean	5.402966e+05	3.373200	2.115826	2080.321850	1.509941e+04	1.494096	1788
std	3.673681e+05	0.926299	0.768984	918.106125	4.141264e+04	0.539683	827
min	7.800000e+04	1.000000	0.500000	370.000000	5.200000e+02	1.000000	370
25%	3.220000e+05	3.000000	1.750000	1430.000000	5.040000e+03	1.000000	1190
50%	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	1.500000	1560
75%	6.450000e+05	4.000000	2.500000	2550.000000	1.068500e+04	2.000000	2210
max	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3.500000	9410

Data Preparation

DATA CLEANING

Lets start by cleaning the data set.

Check and fix columns with missing values in the dataset as shown bellow.

In [7]: #Check fo null values in the dataset
data.isna().sum()

Out[7]:

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	

The waterfron and yr_renovated variables have alot of missing data and therefore require cleaning

The view column too has null values and requires further action

```
In [8]: #check the number of variables lost when removing then null values
d = data.dropna()
data.shape[0] - d.shape[0] #thats a lot of data signals lost lets try and reduce
```

Out[8]: 5835

Deleting all columns with empty entries will result into the loss of 5835 records which is quite a large number and might cause loss of signals in our dataset.

```
In [9]: #We will add the empty records to the NONE Value since its the majority category
print(data['view'].value_counts())

#replacing null view values with NONE
data["view"].fillna("NONE", inplace = True)
```

NONE	19422
AVERAGE	957
GOOD	508
FAIR	330
EXCELLENT	317

Name: view, dtype: int64

```
In [10]: #replace the null values with the majority category since it makes sense to have
data['waterfront'].value_counts()
data["waterfront"].fillna("NO", inplace = True)
```

```
In [11]: #Fill in the missing yr_renovated column by the mean
yr_mean = data['yr_renovated'].mean()
data["yr_renovated"].fillna(yr_mean, inplace = True)
```

```
In [12]: #our data now has no null values
print(data.shape)
data.isna().sum()

(21597, 20)
```

```
Out[12]: date          0
price         0
bedrooms      0
bathrooms     0
sqft_living   0
sqft_lot       0
floors        0
waterfront     0
view          0
condition      0
grade          0
sqft_above     0
sqft_basement  0
yr_built       0
yr_renovated   0
zipcode        0
lat            0
long           0
sqft_living15  0
sqft_lot15     0
dtype: int64
```

Now we have a clean dataset without missing values

```
In [17]: #shape of the new dataset
data.shape
```

```
Out[17]: (21597, 20)
```

```
In [49]: #Checking on the correlation to know which features are important for developing
plt.show()
#for i in data.columns:
#    plt.figure()
#    plt.scatter(data[i],data['price'])
```

In [50]: `#check the correlation to choose the best features`

```
data.corr()['price']
```

C:\Users\admin\AppData\Local\Temp\ipykernel_12732\1079947814.py:2: FutureWarning:
g: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.

```
data.corr()['price']
```

Out[50]:

price	1.000000
bedrooms	0.308787
bathrooms	0.525906
sqft_living	0.701917
sqft_lot	0.089876
floors	0.256804
sqft_above	0.605368
yr_built	0.053953
yr_renovated	0.118125
zipcode	-0.053402
lat	0.306692
long	0.022036
sqft_living15	0.585241
sqft_lot15	0.082845
Name: price, dtype:	float64

lets drop some columns we are not interested using for the model in like:

date, sqft_above, sqft_basement, zipcode, long

In [51]: `#drop unused row for this model`

```
data.drop(['date', 'sqft_above', 'sqft_basement', 'zipcode', 'long', 'sqft_lot15'])
```

In [53]: `#Total number of non categorical variables we will use`

```
print(len(data.columns))
data.columns
```

15

Out[53]:

Index(['price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'grade', 'yr_built', 'yr_renovated', 'lat', 'sqft_living15', 'sqft_lot15'],	dtype='object')

sqft_living seem to have the highest correlation with the price column

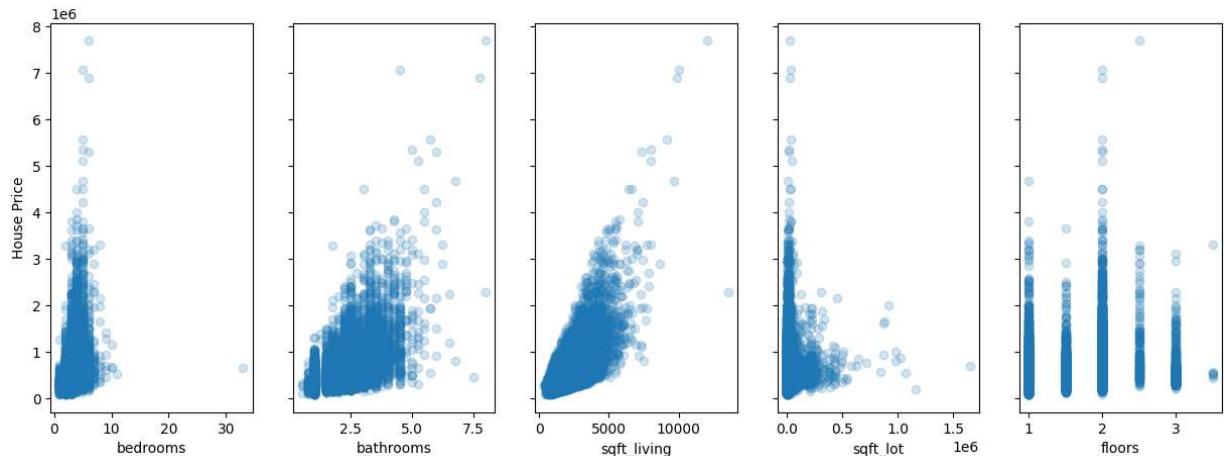
There seem to be a relationship between price and size of the house or rooms

Virtualize the relationship between features and target(house price)

For each numerical feature in the cleand data subset create a scatter plot to investigate the leanear relationship with the price variable for best 5 correlations

```
In [72]: numeric_cols = data[['price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot'],
fig, axes = plt.subplots(ncols=5, figsize=(15,5), sharey=True)
axes[0].set_ylabel('House Price')

for i, col in enumerate(numeric_cols.drop("price", axis=1).columns):
    ax = axes[i]
    ax.scatter(numeric_cols[col], numeric_cols['price'], alpha = 0.2)
    ax.set_xlabel(col)
```



There seem to be some linear relationship with price in bathrooms and sqft_living variables. The floor variable seem to contain discrete variables and the yr_built column doesn't seem to produce a good linear relationship with the target variable price.

DATA MODELING

Base Model

We will now develop a simple linear regression model from which to improve our model on. The sqft_living column shows a strong linear correlation with the house price column so we will develop a linear regression model from these two columns to see how accurate the model will be.

```
In [62]: #set the X variable to be equal to the independent variable sqft_living and the y
X = data[['sqft_living']]
y = data['price']

#develop the simple Linear regression model

base_model = sm.OLS(y, sm.add_constant(X)).fit()
print(base_model.summary())
```

OLS Regression Results

Dep. Variable:	price	R-squared:	0.493			
Model:	OLS	Adj. R-squared:	0.493			
Method:	Least Squares	F-statistic:	2.097e+04			
Date:	Fri, 30 Sep 2022	Prob (F-statistic):	0.00			
Time:	00:16:47	Log-Likelihood:	-3.0006e+05			
No. Observations:	21597	AIC:	6.001e+05			
Df Residuals:	21595	BIC:	6.001e+05			
Df Model:	1					
Covariance Type:	nonrobust					
coef	std err	t	P> t	[0.025	0.975]	
const	-4.399e+04	4410.023	-9.975	0.000	-5.26e+04	-3.53e+04
sqft_living	280.8630	1.939	144.819	0.000	277.062	284.664
Omnibus:	14801.942	Durbin-Watson:	1.982			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	542662.604			
Skew:	2.820	Prob(JB):	0.00			
Kurtosis:	26.901	Cond. No.	5.63e+03			

Notes:

- [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
- [2] The condition number is large, 5.63e+03. This might indicate that there are strong multicollinearity or other numerical problems.

This model has a p value less than 0.05 hence its statistically significant but it only explains about 50% of the variance in the target data by a r_squared value of 0.49 hence its not really a good model.

The intercept is 4.4 meaning if we had a room with 0 Square footage of living space the price of the house we would expect is

$4.4 + 280.8630 \times \text{sqft_living}$

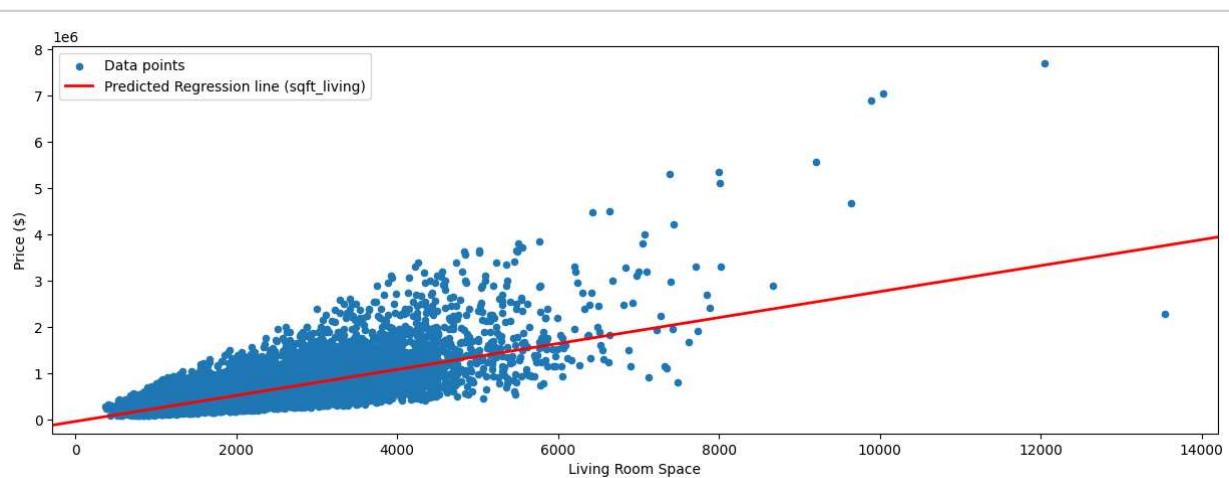
The sqft_living coefficient is about 281 meaning for each additional 1 foot of space in the house price. This suggests the bigger the living space the more expensive the house price.

Let's visualize the model with Square footage of living space as the predictor

In [63]: `from cProfile import label`

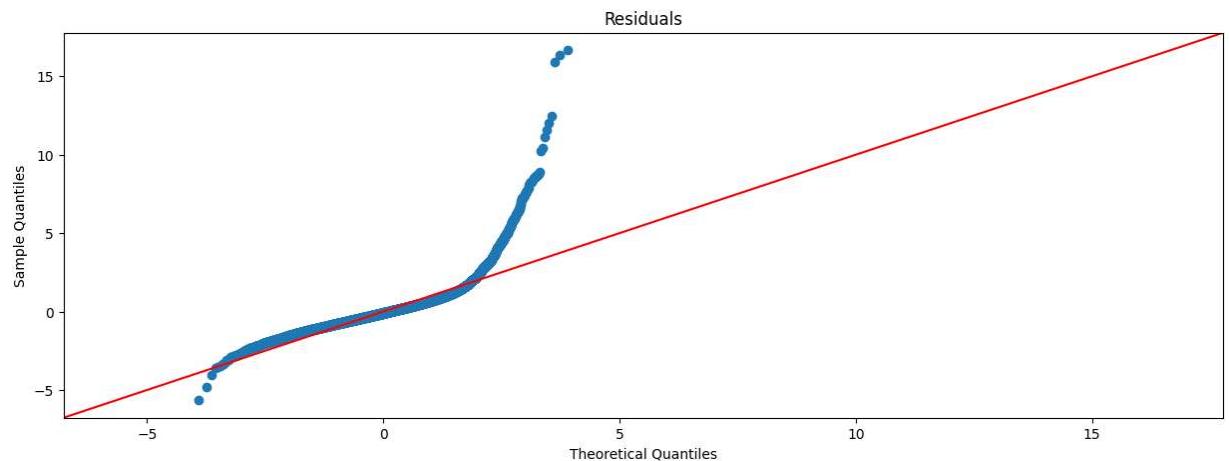
`from matplotlib.pyplot import scatter, ylabel`

```
fig, ax = plt.subplots(figsize=(15,5))
data.plot(y='price', x='sqft_living', kind='scatter', label='Data points', ax=ax)
sm.graphics.abline_plot(model_results=base_model, label='Predicted Regression line', ax=ax)
ax.legend()
plt.show()
```



In [64]: `#Plot the residuals to see how well the model performs`

```
fig, ax = plt.subplots(figsize=(15,5))
sm.graphics.qqplot(base_model.resid, dist= norm, line='45', fit=True, ax=ax)
ax.set_title('Residuals')
plt.show()
```



In [66]: *#calculate the mean absolute error of this model*

```
pre_1 = base_model.predict(sm.add_constant(X))
mean_absolute_error(y, pre_1)
```

Out[66]: 173824.8874961748

This show how bad our model is since its affected by the outliers in the sqft_living coloum. This can be attributed to the fact that some large houses are vary expensive relative to the average price of housing.

Multi Regression

Lets try and add more features into the module to try and see how goo our model performance will increase

Looking at the correlation mattrix in our previous code we find that the next value with a higher correlation matrix is bathrooms(0.525906), bedrooms(0.308787), floors(0.256804), sqft_lot (0.089876) and finally yr_built(0.053953)

lets try fitting all these features into our model and see how well it performs relative to the base model above.

This increases our Adj. R-squared value by about 6% indicating an improvement on our model to explain the varience of 12% more of out target variable. The model is also still statistically significant wit a p value less than 0.05

```
In [73]: #multi regression with all the data points
X_all = data[['bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors', 'yr_built', 'lat', 'long', 'yr_renovated']]
multi_model_all = sm.OLS(y, sm.add_constant(X_all)).fit()
print(multi_model_all.summary())
```

OLS Regression Results

Dep. Variable:	price	R-squared:	0.616			
Model:	OLS	Adj. R-squared:	0.616			
Method:	Least Squares	F-statistic:	3850.			
Date:	Fri, 30 Sep 2022	Prob (F-statistic):	0.00			
Time:	00:24:39	Log-Likelihood:	-2.9705e+05			
No. Observations:	21597	AIC:	5.941e+05			
Df Residuals:	21587	BIC:	5.942e+05			
Df Model:	9					
Covariance Type:	nonrobust					
const	-2.314e+07	5.92e+05	-39.053	0.000	-2.43e+07	-2.2e+07
bedrooms	-5.755e+04	2107.467	-27.306	0.000	-6.17e+04	-5.34e+04
bathrooms	6.153e+04	3613.566	17.029	0.000	5.45e+04	6.86e+04
sqft_living	244.9043	3.490	70.178	0.000	238.064	251.745
sqft_lot	-0.1256	0.038	-3.266	0.001	-0.201	-0.050
floors	3.698e+04	3541.206	10.443	0.000	3e+04	4.39e+04
yr_built	-2743.5614	69.938	-39.228	0.000	-2880.646	-2606.477
yr_renovated	32.7660	4.451	7.361	0.000	24.041	41.491
lat	5.975e+05	1.15e+04	51.748	0.000	5.75e+05	6.2e+05
sqft_living15	82.2603	3.512	23.420	0.000	75.376	89.145

Omnibus:	17662.005	Durbin-Watson:	1.994
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1154418.694
Skew:	3.494	Prob(JB):	0.00
Kurtosis:	38.129	Cond. No.	1.69e+07

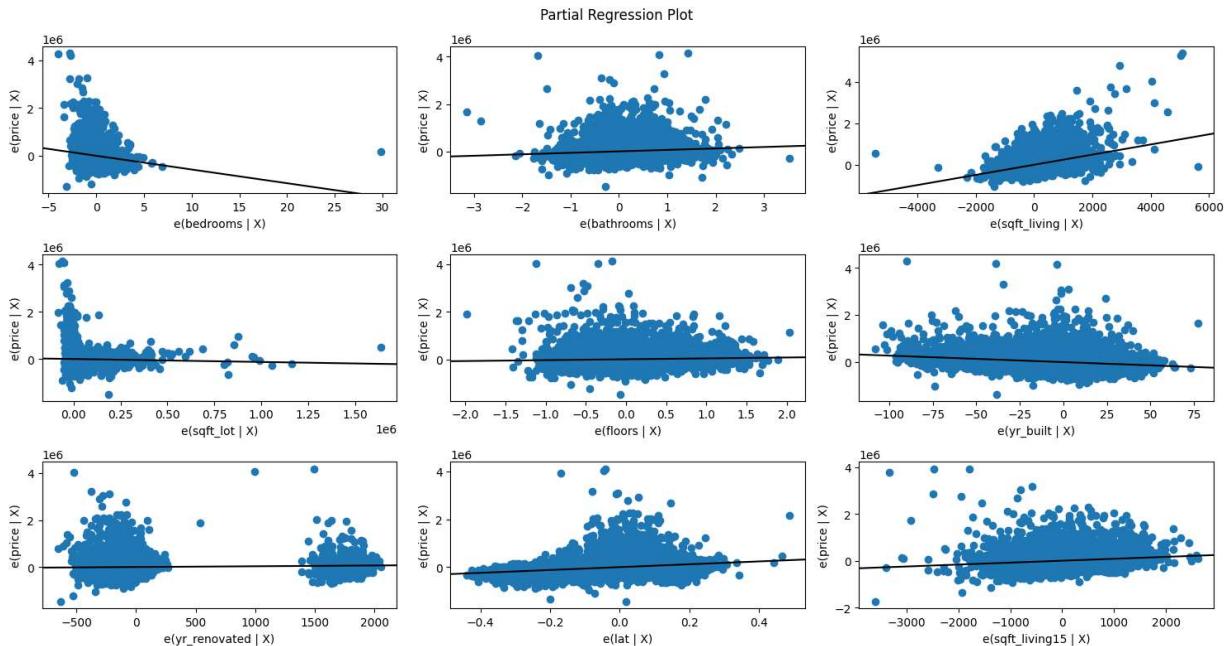
Notes:

- [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
- [2] The condition number is large, 1.69e+07. This might indicate that there are strong multicollinearity or other numerical problems.

Create a partial Regression Plots for all the features plotted to see how each perform

```
In [76]: fig = plt.figure(figsize=(15,8))
sm.graphics.plot_partregress_grid(
    multi_model_all,
    exog_idx=list(X_all.columns),
    grid=(3,3),
    fig=fig
)
plt.tight_layout()
plt.show()
```

```
eval_env: 1
```



The models tries to fit the data well though there exists some outliers that the linear model does not capture well. There are still some categorical variables not added in this model. Lets add that and see if the model accuracy will increase

```
In [77]: #calculate the mean absolute error of this model
pre_2 = multi_model_all.predict(sm.add_constant(X_all))
mean_absolute_error(y, pre_2)
```

Out[77]: 142490.4839760155

Adding the categorical variables to the model

The model above did not use any categorical variables in the data set. We will use one-hot-encoding to transform the categorical columns to feed them to the model to increase the signal in the data.

Our dataset include 3 categorical columns ['waterfront', 'condition', 'grade']

```
In [81]: #view the categorical values an their respective categories
cat_data = data[['waterfront','view', 'condition', 'grade']]
for col in cat_data.columns:
    print("the categorical values in " + col + " are:")
    print(cat_data[col].value_counts())
```

the categorical values in waterfront are:

```
NO      21451
YES     146
```

Name: waterfront, dtype: int64

the categorical values in view are:

```
NONE      19485
AVERAGE    957
GOOD       508
FAIR        330
EXCELLENT   317
```

Name: view, dtype: int64

the categorical values in condition are:

```
Average    14020
Good       5677
Very Good  1701
Fair        170
Poor        29
```

Name: condition, dtype: int64

the categorical values in grade are:

```
7 Average    8974
8 Good       6065
9 Better      2615
6 Low Average 2038
10 Very Good  1134
11 Excellent   399
5 Fair        242
12 Luxury      89
4 Low          27
13 Mansion     13
3 Poor         1
```

Name: grade, dtype: int64

```
In [83]: X_3 = data.drop('price', axis=1)
X_categorical = pd.get_dummies(X_3, columns=['waterfront','view', 'condition', '...
print(X_3.shape)
print(X_categorical.shape) #We now have more features in our dataset to bust the
X_categorical.head()
```

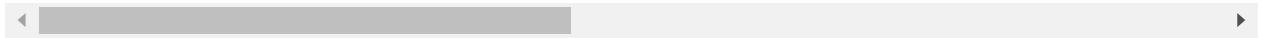
```
(21597, 14)
(21597, 29)
```

Out[83]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	yr_built	yr_renovated	lat	sq
	id								
1	7129300520	3	1.00	1180	5650	1.0	1955	0.000000	47.5112
2	6414100192	3	2.25	2570	7242	2.0	1951	1991.000000	47.7210
3	5631500400	2	1.00	770	10000	1.0	1933	83.636778	47.7379
4	2487200875	4	3.00	1960	5000	1.0	1965	0.000000	47.5208
5	1954400510	3	2.00	1680	8080	1.0	1987	0.000000	47.6168

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	yr_built	yr_renovated	lat	sq
	id								
1	7129300520	3	1.00	1180	5650	1.0	1955	0.000000	47.5112
2	6414100192	3	2.25	2570	7242	2.0	1951	1991.000000	47.7210
3	5631500400	2	1.00	770	10000	1.0	1933	83.636778	47.7379
4	2487200875	4	3.00	1960	5000	1.0	1965	0.000000	47.5208
5	1954400510	3	2.00	1680	8080	1.0	1987	0.000000	47.6168

5 rows × 29 columns



In [84]: #Run the full model containing the full data set to see how this will improve it
`full_model = sm.OLS(y, sm.add_constant(X_categorical)).fit()
print(full_model.summary())`

OLS Regression Results														
Dep. Variable:	price	R-squared:	0.725											
Model:	OLS	Adj. R-squared:	0.725											
Method:	Least Squares	F-statistic:	1965.											
Date:	Fri, 30 Sep 2022	Prob (F-statistic):	0.00											
Time:	00:33:54	Log-Likelihood:	-2.9343e+05											
No. Observations:	21597	AIC:	5.869e+05											
Df Residuals:	21567	BIC:	5.872e+05											
Df Model:	29													
Covariance Type:	nonrobust													
=====														
=====														
	coef	std err	t	P> t	[0.025	0.975]								

const	-2.256e+07	5.24e+05	-43.055	0.000	-2.36e+07	-								
2.15e+07														
bedrooms	-1.898e+04	1856.450	-10.224	0.000	-2.26e+04	-								
1.53e+04														
bathrooms	4.54e+04	3109.411	14.600	0.000	3.93e+04	-								
5.15e+04														
sqft_living	129.2218	3.298	39.177	0.000	122.757	-								
135.687														
sqft_lot	0.0955	0.046	2.087	0.037	0.006	-								
0.185														
floors	2.373e+04	3120.839	7.605	0.000	1.76e+04	-								
2.99e+04														
yr_built	-2227.0619	66.139	-33.672	0.000	-2356.700	-								
2097.424														
yr_renovated	34.1641	3.839	8.900	0.000	26.640	-								
41.688														
lat	5.765e+05	9979.433	57.769	0.000	5.57e+05	-								
5.96e+05														
sqft_living15	26.1074	3.192	8.178	0.000	19.850	-								
32.364														
sqft_lot15	-0.4502	0.070	-6.440	0.000	-0.587	-								
-0.313														
waterfront_YES	5.559e+05	1.95e+04	28.442	0.000	5.18e+05	-								
5.94e+05														
view_EXCELLENT	1.872e+05	1.46e+04	12.811	0.000	1.59e+05	-								
2.16e+05														
view_FAIR	5.342e+04	1.23e+04	4.338	0.000	2.93e+04	-								
7.76e+04														
view_GOOD	6.529e+04	1.06e+04	6.151	0.000	4.45e+04	-								
8.61e+04														
view_NONE	-6.784e+04	6541.366	-10.370	0.000	-8.07e+04	-								
-5.5e+04														
condition_Fair	-6257.8973	1.5e+04	-0.416	0.678	-3.58e+04	-								
2.32e+04														
condition_Good	3.738e+04	3303.996	11.313	0.000	3.09e+04	-								
4.39e+04														

condition_Poor	-5.676e+04	3.61e+04	-1.571	0.116	-1.28e+05
1.41e+04					
condition_Very Good	7.543e+04	5308.922	14.208	0.000	6.5e+04
8.58e+04					
grade_11 Excellent	2.526e+05	1.14e+04	22.081	0.000	2.3e+05
2.75e+05					
grade_12 Luxury	7.184e+05	2.18e+04	32.888	0.000	6.76e+05
7.61e+05					
grade_13 Mansion	1.917e+06	5.47e+04	35.029	0.000	1.81e+06
2.02e+06					
grade_3 Poor	-2.637e+05	1.93e+05	-1.366	0.172	-6.42e+05
1.15e+05					
grade_4 Low	-4.152e+05	3.83e+04	-10.826	0.000	-4.9e+05
-3.4e+05					
grade_5 Fair	-4.271e+05	1.55e+04	-27.548	0.000	-4.58e+05
3.97e+05					
grade_6 Low Average	-3.988e+05	9806.518	-40.667	0.000	-4.18e+05
-3.8e+05					
grade_7 Average	-3.51e+05	8052.586	-43.584	0.000	-3.67e+05
3.35e+05					
grade_8 Good	-2.864e+05	7228.850	-39.613	0.000	-3.01e+05
2.72e+05					
grade_9 Better	-1.651e+05	7074.169	-23.337	0.000	-1.79e+05
1.51e+05					
<hr/>					
Omnibus:	14774.043	Durbin-Watson:	1.984		
Prob(Omnibus):	0.000	Jarque-Bera (JB):	803522.467		
Skew:	2.675	Prob(JB):	0.00		
Kurtosis:	32.399	Cond. No.	2.02e+07		
<hr/>					

Notes:

- [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
- [2] The condition number is large, 2.02e+07. This might indicate that there are strong multicollinearity or other numerical problems.

In [87]: `#calculate the mean absolute error of this model
pre_3 = full_model.predict(sm.add_constant(X_categorical))
mean_absolute_error(y, pre_3)`

Out[87]: 119092.80264611663

The model has greatly improved by the inclusion of the categorical variables and the variance in the target variable can be explained up to about 73%

The model is also statistically significant with a p value of less than 0.05

Explaining the model

Model Summary

- Our model is statistically significant overall, and explains about 73% of the variance in House sales Price.
- On average this model may be off by about \$119k in its predictions on home price this is due to the outliers caused by high expensive houses that affect the model predictive ability.

All of our coefficients are statistically significant

- const: when all the features used in this prediction are zero, we would expect a home sale price of -\$22558k. This might not be realistic in real world since we cannot have a house with zero rooms for example
- bedrooms: For each increase 1 bedroom space we see an associated decrease in House price by about \$19k. This can be brought about by the fact that houses in towns tend to be small and expensive while those out of town are big and cheaper.
- bathrooms: For each increase 1 bathroom space we see an associated increase in House price by about \$45k.
- sqft_living: For each increase in 1 inch of squarefoot of living room space we see an associated increase in House price by about \$129
- sqft_lot: For each increase in 1 inch of Square footage of the lot space we see an associated increase in House price by a small amount of about \$0.1. This suggests that the Square footage of the lot is not a real concern for home owners
- floors : For each increase in 1 house floor, we see an associated increase in House price by about \$24k
- yr_built: For each increase in a house's year by 1 we see an associated decrease in House price by about \$2k. This can be brought about by the fact that houses built long time ago have a low price than recent houses.
- yr_renovated: For each increase in a house's year of renovation by 1 we see an associated increase in House price by about \$34. This can be brought about by the fact that houses that are recently renewed are much expensive due to the renovations done
- lat: For each increase 1 latitude, we see an associated increase in House price by about \$577k. This can be attributed to the fact that temperatures at high latitude areas are favorable for many people hence the higher price house price

Type *Markdown* and *LaTeX*: α^2