Summary:

The main motive of the research is to develop a regression model based on the kc_house.csv. Here we did linear regression, histogram, and plotting some diagram with the help of the dataset.

Business Problem:

From this product and linear regression, it is very easy to understand the house price with different needs of the real estate investor. In this project the customer is real estate investor and user is estimator or could be house buyer who can determine the price guide.

1) how house price is increased due to its different attributes. 2) is it good idea to buy an old house and doing renovation (adding different attributes) to increase house value! 3) what are main key attributes takes part of changing house price?

What is your role in this project?

Linear regression is mainly used for analyzing and predict the value of the variable which is based on the overall value of another variable. All the variable that have been predicted is called dependent variable.

Why linear regression is used?

Linear regression is used to predict modeling and analysis. The variable is used for predicting other variables value which is called independent variable.

Where linear is being used?

Such as it used for medical researchers which often use linear regression for understanding the relationship between drug and blood pressure of patients, estimate price of a house, real estate investor, or an estimator.

What should this project accomplish for the business?

From this product and linear regression, it is very easy to understand the house price with different needs of the real estate investor. In this project the customer is real estate investor and user is estimator.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.neighbors import KNeighborsRegressor
from sklearn.preprocessing import PolynomialFeatures
from sklearn import metrics
import matplotlib.pyplot as plt
import seaborn as sns
from mpl_toolkits.mplot3d import Axes3D
%matplotlib inline
```

Figure: Importing Libraries

Here in the above image all the necessary libraries have been imported whereas NumPy, panda, sklearn, and many more. These libraries make the coding very easy for the following regression method.

```
df = pd.read_csv("E:/Extra/python/dsc-phase-2-project-main/data/kc_house_data.csv")
df.head()
```

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basement	yr_built	yr_ren
0	7129300520	10/13/2014	221900	3	1.000	1180	5650	1.000	NaN	0.000	...	7	1180	0	1955	
1	6414100192	12-09-2014	538000	3	2.250	2570	7242	2.000	0.000	0.000	...	7	2170	400	1951	19
2	5631500400	2/25/2015	180000	2	1.000	770	10000	1.000	0.000	0.000	...	6	770	0	1933	
3	2487200875	12-09-2014	604000	4	3.000	1960	5000	1.000	0.000	0.000	...	7	1050	910	1965	
4	1954400510	2/18/2015	510000	3	2.000	1680	8080	1.000	0.000	0.000	...	8	1680	0	1987	

5 rows × 21 columns

Figure: Read the dataset

The dataset has been imported and the head of the dataset had been seen in the above image. There are many variables by which the regression model can establish.

```
df.describe()
```

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	yr_b
count	21597.000	21597.000	21597.000	21597.000	21597.000	21597.000	21597.000	19221.000	21534.000	21597.000	21597.000	21597.000	21597.
mean	4580474287.771	540296.574	3.373	2.116	2080.322	15099.409	1.494	0.008	0.234	3.410	7.658	1788.597	1971.
std	2876735715.748	367368.140	0.926	0.769	918.106	41412.637	0.540	0.087	0.766	0.651	1.173	827.760	29.
min	1000102.000	78000.000	1.000	0.500	370.000	520.000	1.000	0.000	0.000	1.000	3.000	370.000	1900.
25%	2123049175.000	322000.000	3.000	1.750	1430.000	5040.000	1.000	0.000	0.000	3.000	7.000	1190.000	1951.
50%	3904930410.000	450000.000	3.000	2.250	1910.000	7618.000	1.500	0.000	0.000	3.000	7.000	1560.000	1975.
75%	7308900490.000	645000.000	4.000	2.500	2550.000	10685.000	2.000	0.000	0.000	4.000	8.000	2210.000	1997.
max	9900000190.000	7700000.000	33.000	8.000	13540.000	1651359.000	3.500	1.000	4.000	5.000	13.000	9410.000	2015.

Figure: Describe the dataset

Here in the above image, the dataset has been described as the above measurement can be seen from it.

```
pd.set_option('display.float_format', lambda x: '%.3f' % x)
```

```
df.describe()
```

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	yr_b
count	21597.000	21597.000	21597.000	21597.000	21597.000	21597.000	21597.000	19221.000	21534.000	21597.000	21597.000	21597.000	21597.
mean	4580474287.771	540296.574	3.373	2.116	2080.322	15099.409	1.494	0.008	0.234	3.410	7.658	1788.597	1971.
std	2876735715.748	367368.140	0.926	0.769	918.106	41412.637	0.540	0.087	0.766	0.651	1.173	827.760	29.
min	1000102.000	78000.000	1.000	0.500	370.000	520.000	1.000	0.000	0.000	1.000	3.000	370.000	1900.
25%	2123049175.000	322000.000	3.000	1.750	1430.000	5040.000	1.000	0.000	0.000	3.000	7.000	1190.000	1951.
50%	3904930410.000	450000.000	3.000	2.250	1910.000	7618.000	1.500	0.000	0.000	3.000	7.000	1560.000	1975.
75%	7308900490.000	645000.000	4.000	2.500	2550.000	10685.000	2.000	0.000	0.000	4.000	8.000	2210.000	1997.
max	9900000190.000	7700000.000	33.000	8.000	13540.000	1651359.000	3.500	1.000	4.000	5.000	13.000	9410.000	2015.

Figure: Dropping scientific notation

Here the above image shows the function used to drop the scientific notation that exists in this dataset. The data description has been shown without any scientific notation.

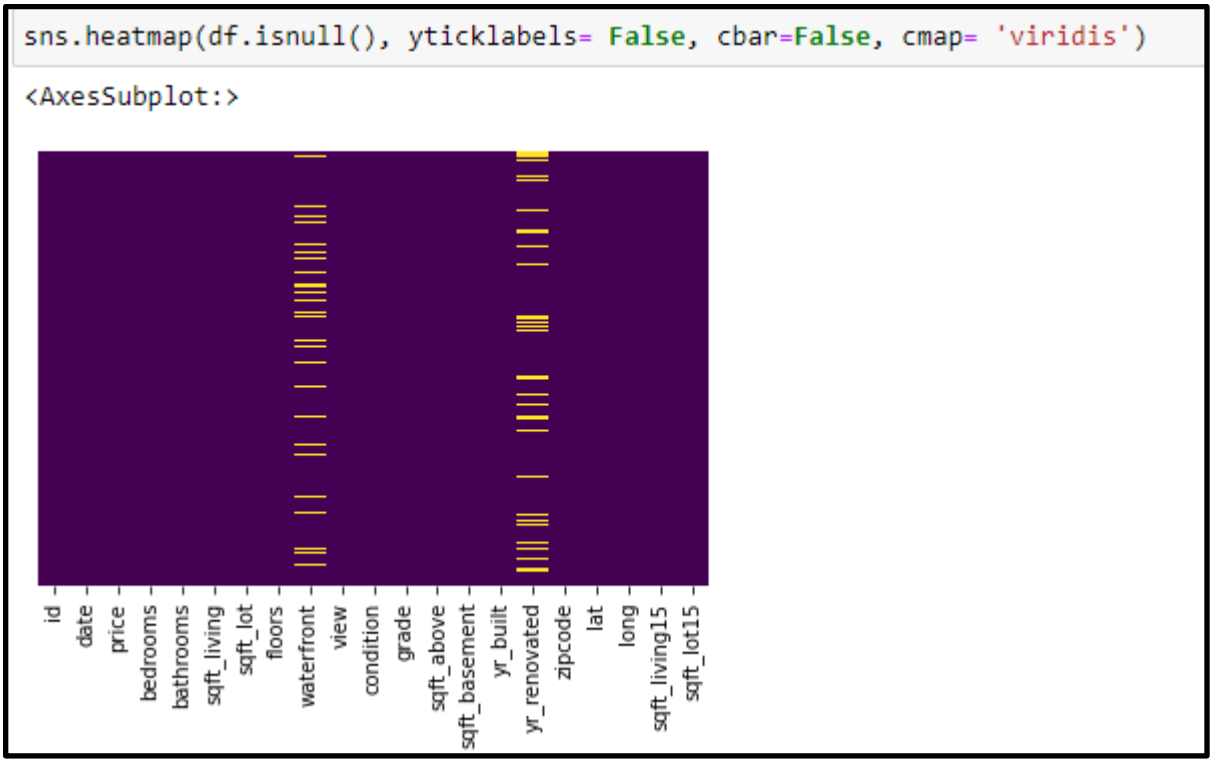

Figure: Checking null value in a plot

Here as it can be seen in the waterfront and yr. renovated there is some null value consistent in this dataset. The null value was shown in a plot for a clear view for the viewers.

```
sns.countplot(df.bedrooms, order = df['bedrooms'].value_counts().index)
```

```
<AxesSubplot:xlabel='bedrooms', ylabel='count'>
```

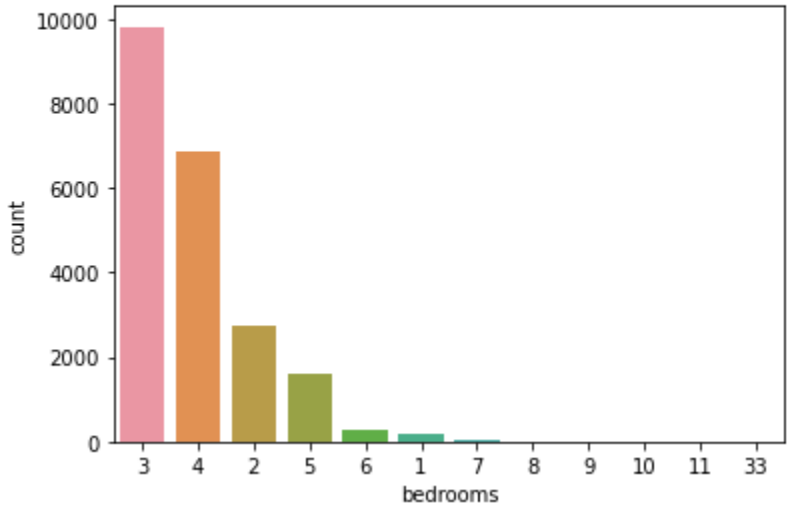

Figure: Count plot between count and bedrooms

The above image describes the count plot between count and bedrooms.

```
plt.hist('waterfront', data = df, bins = 5)
```

```
(array([19075.,     0.,     0.,     0.,   146.]),
 array([0. , 0.2, 0.4, 0.6, 0.8, 1. ]),
 <BarContainer object of 5 artists>)
```

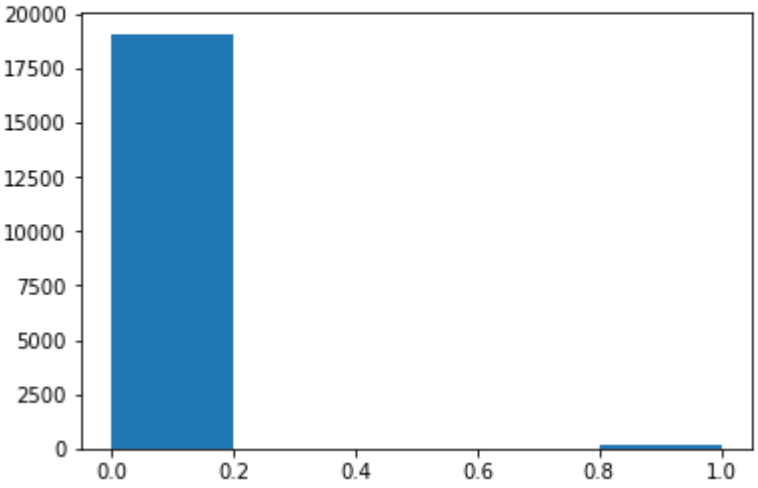

Figure: Waterfront Histogram

Here the above function is used to show the histogram of the waterfront.

```
fig,axes = plt.subplots(nrows=1,ncols=1, figsize=(10,10))
sns.distplot(df['waterfront'], hist=True, kde=True, rug=False, label='waterfront', norm_hist=True)
```

```
<AxesSubplot:xlabel='waterfront', ylabel='Density'>
```

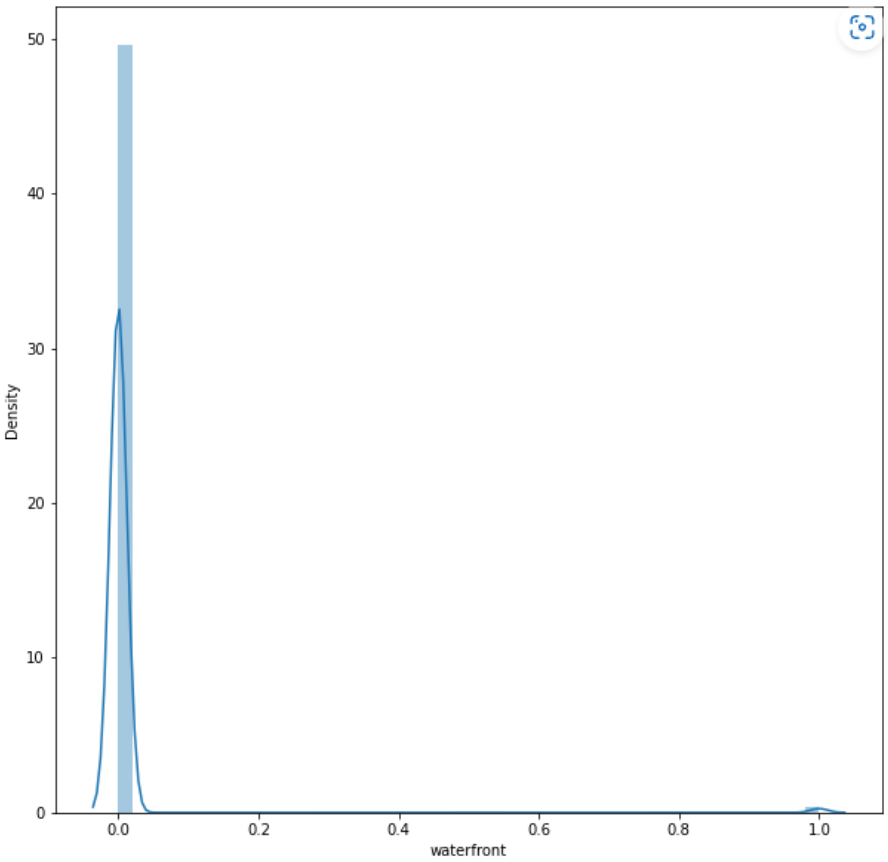

Figure: Subplot between density and waterfront

Here the above function is used to show the subplot between the density and waterfront. Here it shows the density is too high for no waterfront and low for a single waterfront.

```
fig,axes = plt.subplots(nrows=1,ncols=1, figsize=(10,10))
sns.distplot(df['price'], hist=True, kde=True, rug=False, label='price', norm_hist=True)
```

```
<AxesSubplot:xlabel='price', ylabel='Density'>
```

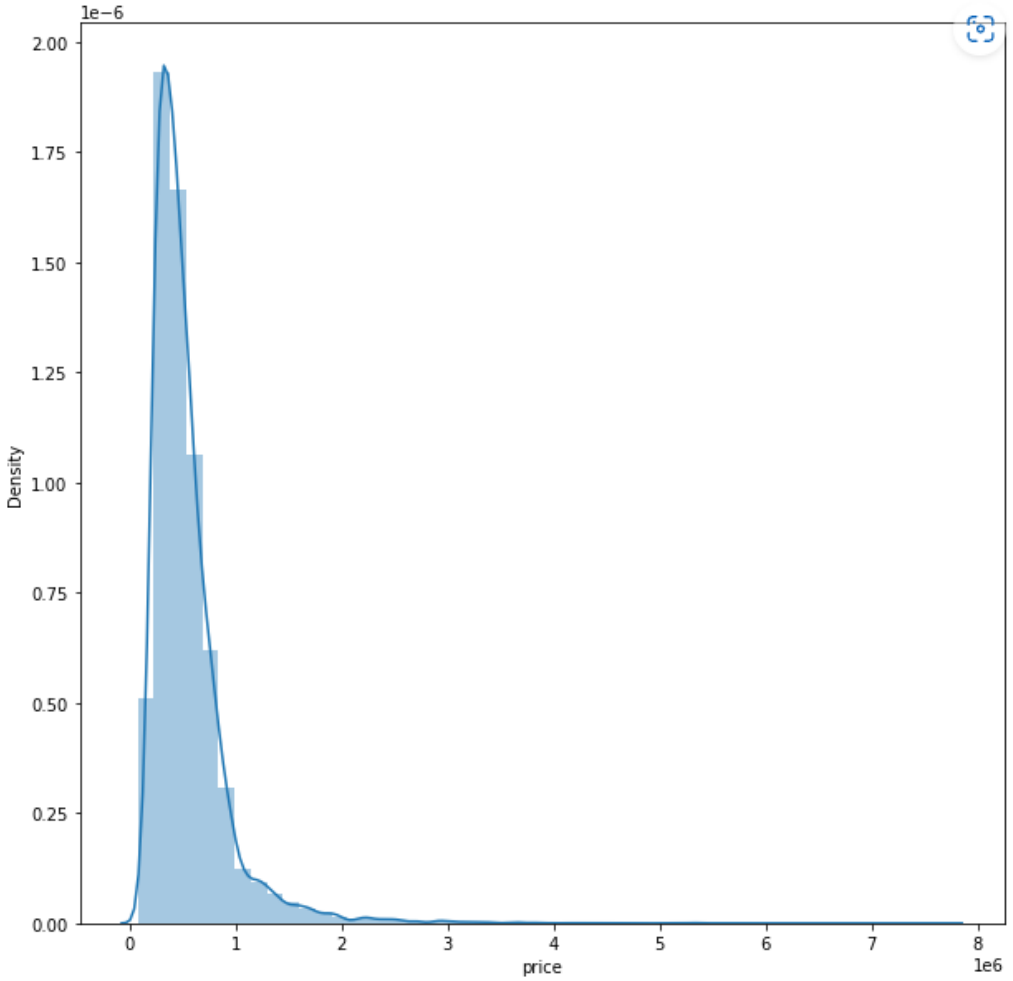

Figure: Subplot between density and price

The above function is used to show the subplot between density and price.

```
#mean median mode

print('Mean', round(df['price'].mean(), 2))
print('Median', df['price'].median())
print('Mode', df['price'].mode()[0])

Mean 540296.57
Median 450000.0
Mode 350000
```

Figure: Mean median mode

Mean median and mode have been shown with the above function. Here this value shows the mean is greater than the median and the median greater than the mode.

```
In [143]: len(df[df['sqft_living']==1300])

Out[143]: 138
```

Figure: Mean median mode

Here with the above function, it can be understood that 138 houses have 1300 sqft.

```
def correlation_heatmap(df1):
    _, ax = plt.subplots(figsize = (15, 12))
    colormap= sns.diverging_palette(320, 210, as_cmap = True)
    sns.heatmap(df.corr(), annot=True, cmap = colormap)

correlation_heatmap(df)
#with the correlation matrix it can be observed that the price has the better
#correlation coefficient with the living area(sqft)(0.7)
# we use the living area as feature when create
#regression after the model of the linear relationship among
#response and explanatory variable.
```

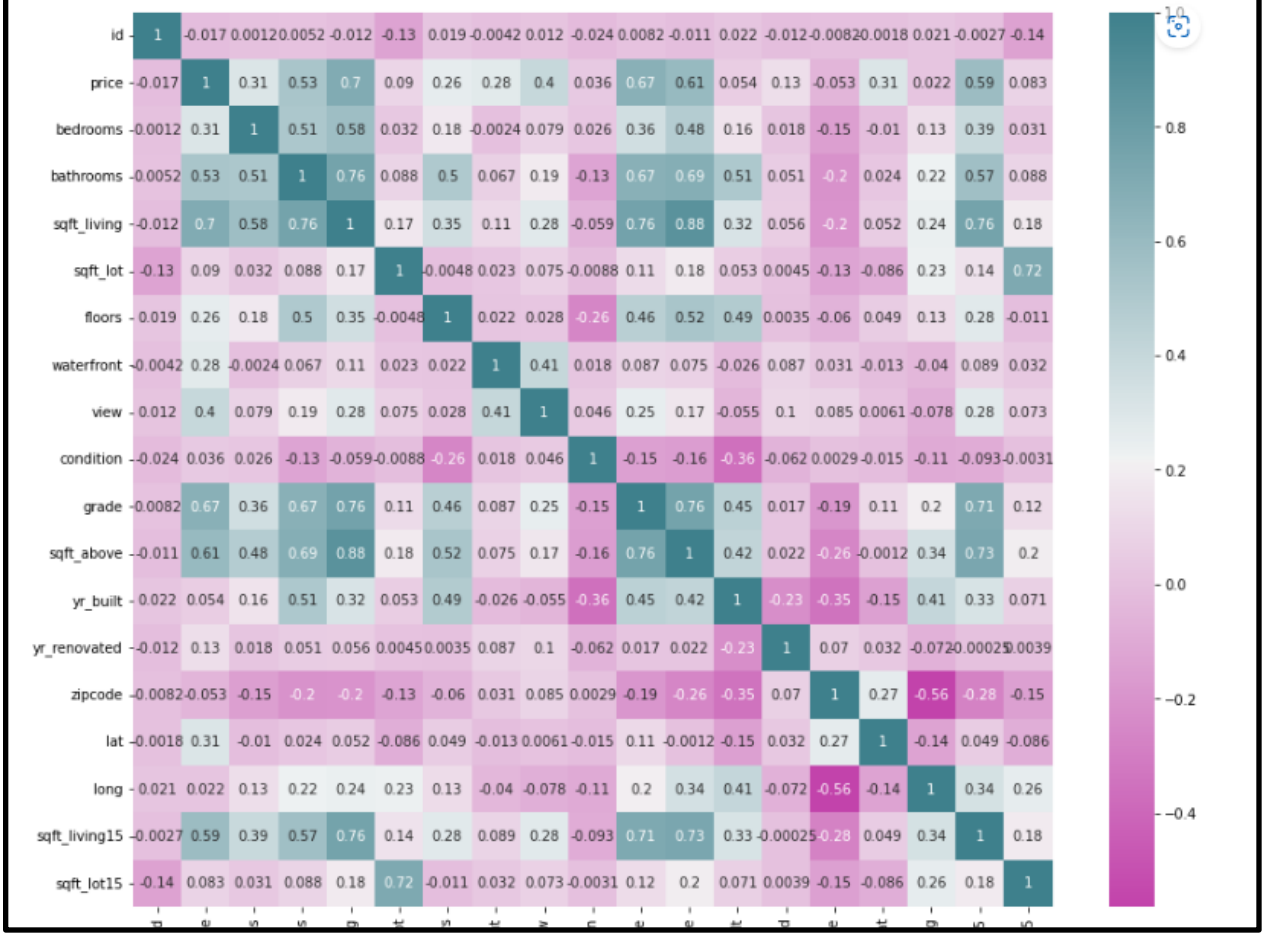

Figure: Correlation heatmap

A correlation heatmap is shown with all of the data.

```
train_data, test_data = train_test_split(df, train_size =0.8, random_state = 3)
reg = linear_model.LinearRegression()
x_train = np.array(train_data['sqft_living']).reshape(-1,1)
y_train = np.array(train_data['price']).reshape(-1, 1)
reg.fit(x_train, y_train)
#evaluate simple model
x_test = np.array(test_data['sqft_living']).reshape(-1, 1)
y_test = np.array(test_data['price']).reshape(-1, 1)
pred = reg.predict(x_test)
print('Simple Model')
mean_squared_error = metrics.mean_squared_error(y_test, pred)
print('Mean Squared Error (MSE) ', round(np.sqrt(mean_squared_error), 2))
print('R-squared (training) ', round(reg.score(x_train, y_train), 3))
print('R-squared (testing) ', round(reg.score(x_test, y_test), 3))
print('Intercept: ', reg.intercept_)
print('Coefficient:', reg.coef_)
```

```
Simple Model
Mean Squared Error (MSE)  271171.89
R-squared (training)  0.499
R-squared (testing)  0.467
Intercept:  [-43739.61859471]
Coefficient: [[281.48917629]]
```

Figure: Train test split

The dataset is split here between train and test data where the mean squared error, intercept and coefficient have been shown.

```
_, ax = plt.subplots(figsize= (10, 12))
plt.scatter(x_test, y_test, color= 'blue', label = 'data')
plt.plot(x_test, reg.predict(x_test), color='red', label= ' Predicted Regression line')
plt.xlabel('Living Space (sqft)')
plt.ylabel('price')
plt.legend()
plt.gca().spines['right'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
```

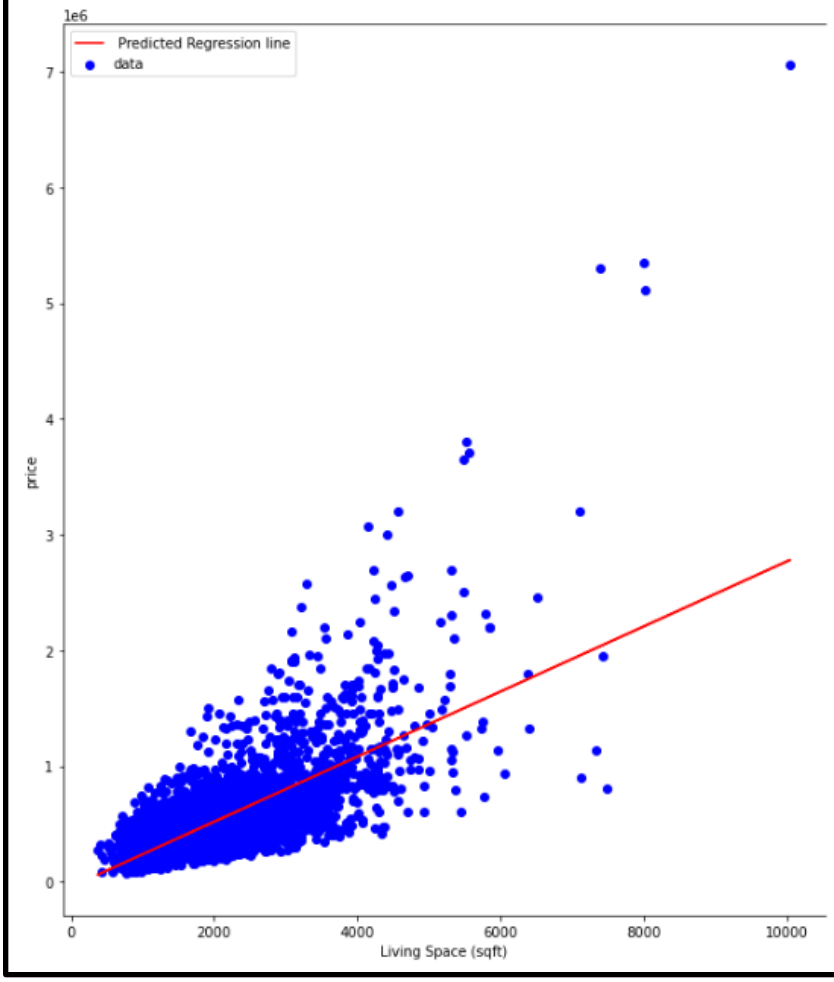

Figure: Regression prediction

Here a linear regression has been used with the above function and found to be a poor fit. For improving the model there are more features need to be added.

```
_ , axes = plt.subplots(2, 1, figsize=(15,10))
sns.boxplot(x= train_data['grade'], y=train_data['price'],ax = axes[0])
sns.boxplot(x=train_data['bedrooms'], y=train_data['price'], ax=axes[1])

_ , axes = plt.subplots(1, 1, figsize=(15,10))
sns.boxplot(x=train_data['bathrooms'], y=train_data['price'])
```

```
<AxesSubplot:xlabel='bathrooms', ylabel='price'>
```

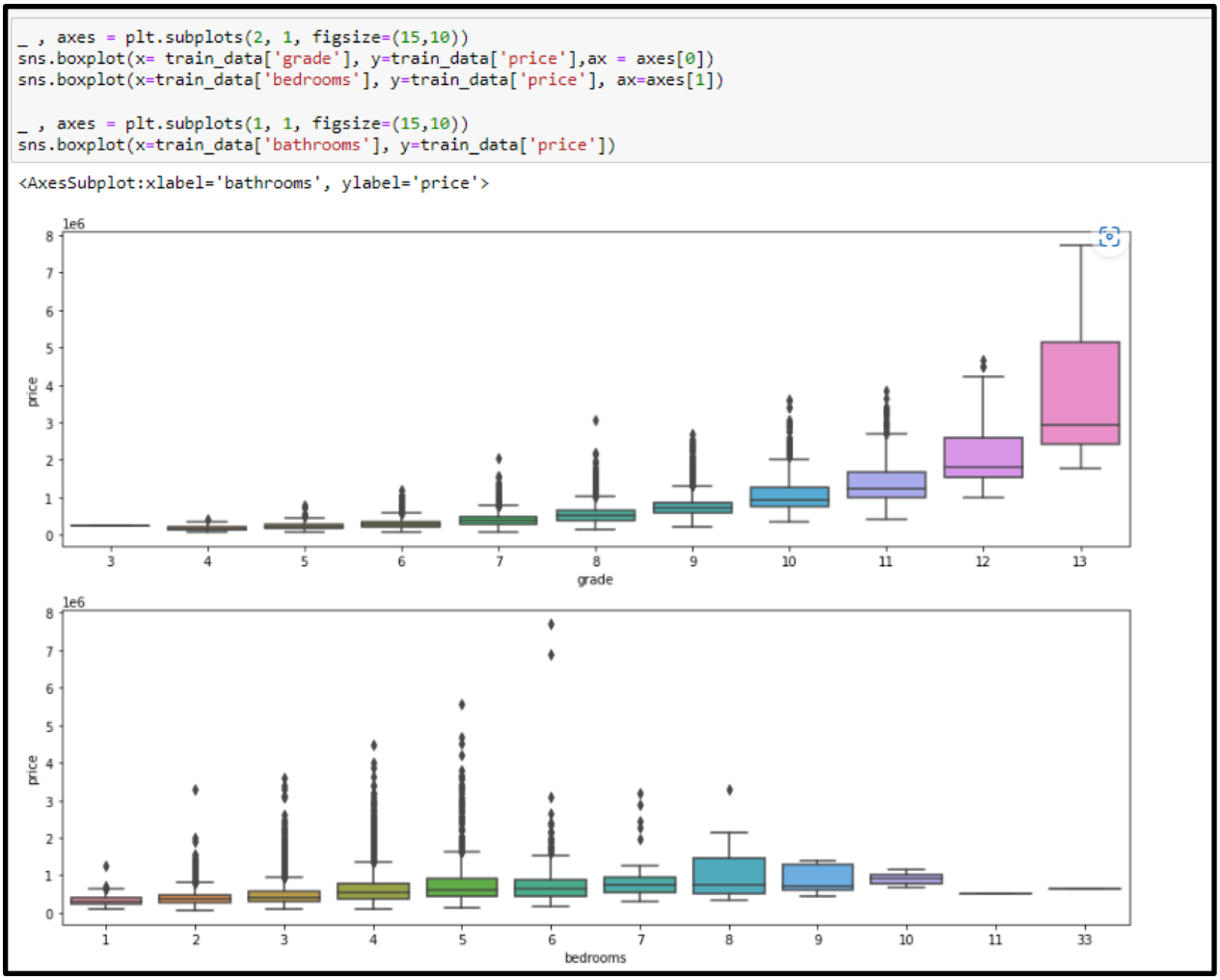

Figure: Subplot between price and grade; price and bedrooms

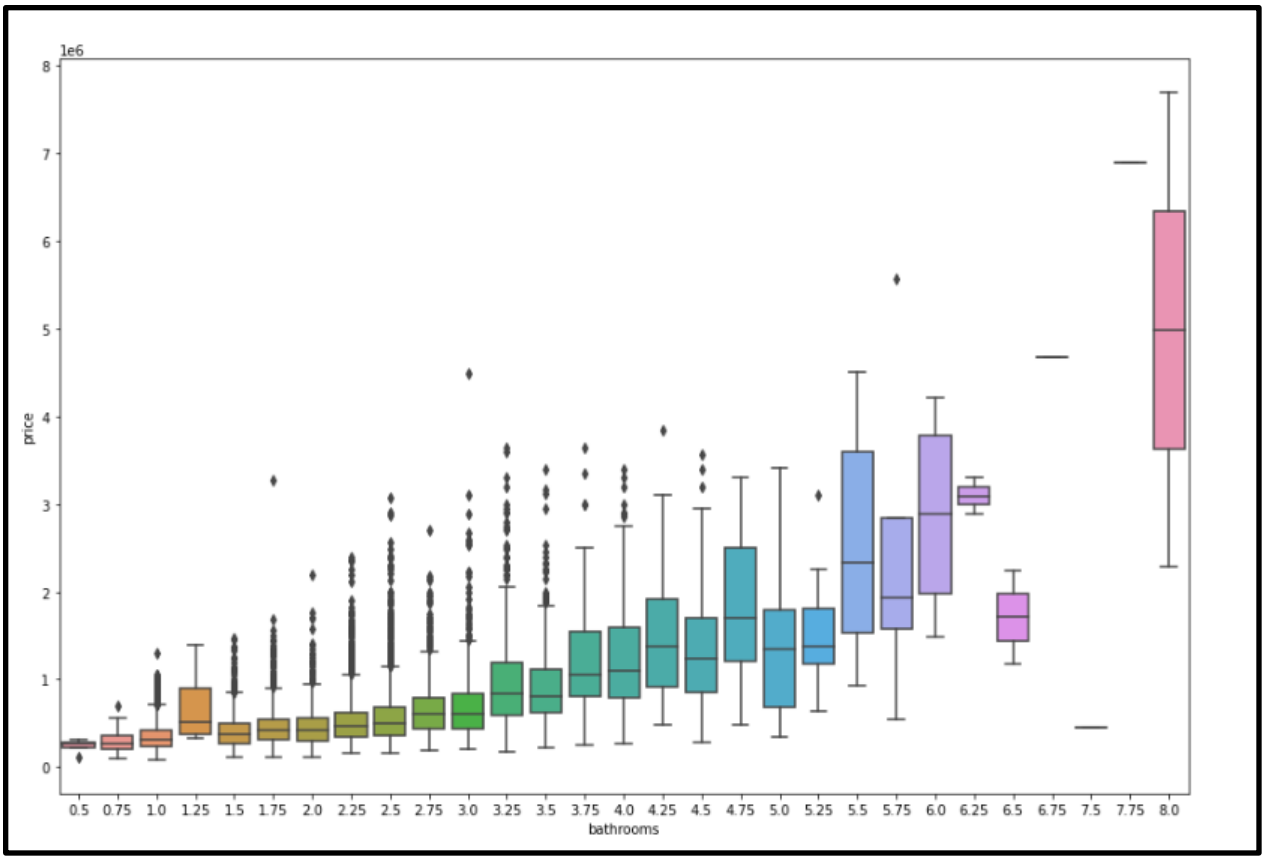

Figure: Subplot between bathrooms and price.

In these 2 diagrams, a subplot has been shown between these variables.

All the output that has been generated asper the deliverables, the linear regression has been done asper the model, and all the plot have been shown in the output by using above functions with help of the required library.

```
fig,axes = plt.subplots(nrows=1,ncols=1, figsize=(15,10))
plt.title('House prices by condition')
plt.xlabel('condition')
plt.ylabel('House Prices')
plt.legend()
sns.barplot(x='condition', y='price', data = df)
```

```
No handles with labels found to put in legend.
```

```
<AxesSubplot:title={'center':'House prices by condition'}, xlabel='condition', ylabel='price'>
```

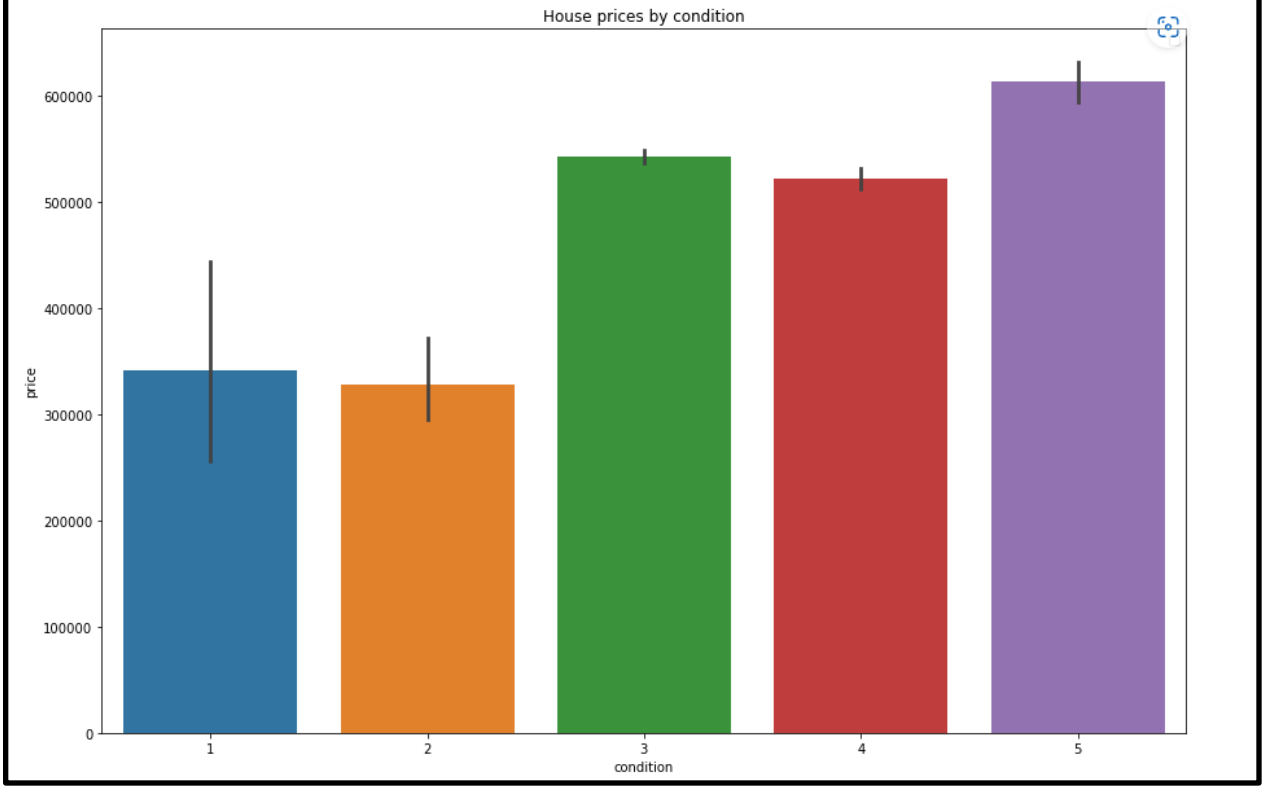

Figure: Prices of the house by condition

The above image shows the bar graph against the condition and price. It can be understood from the above image that with great condition the price of house also increased.

```
In [127]: fig,axes = plt.subplots(nrows=1,ncols=1, figsize=(15,10))
          plt.title('House prices by view')
          plt.xlabel('view')
          plt.ylabel('House Prices')
          plt.legend()
          sns.barplot(x='view', y='price', data = df)

          No handles with labels found to put in legend.

Out[127]: <AxesSubplot:title={'center':'House prices by view'}, xlabel='view', ylabel='price'>
```

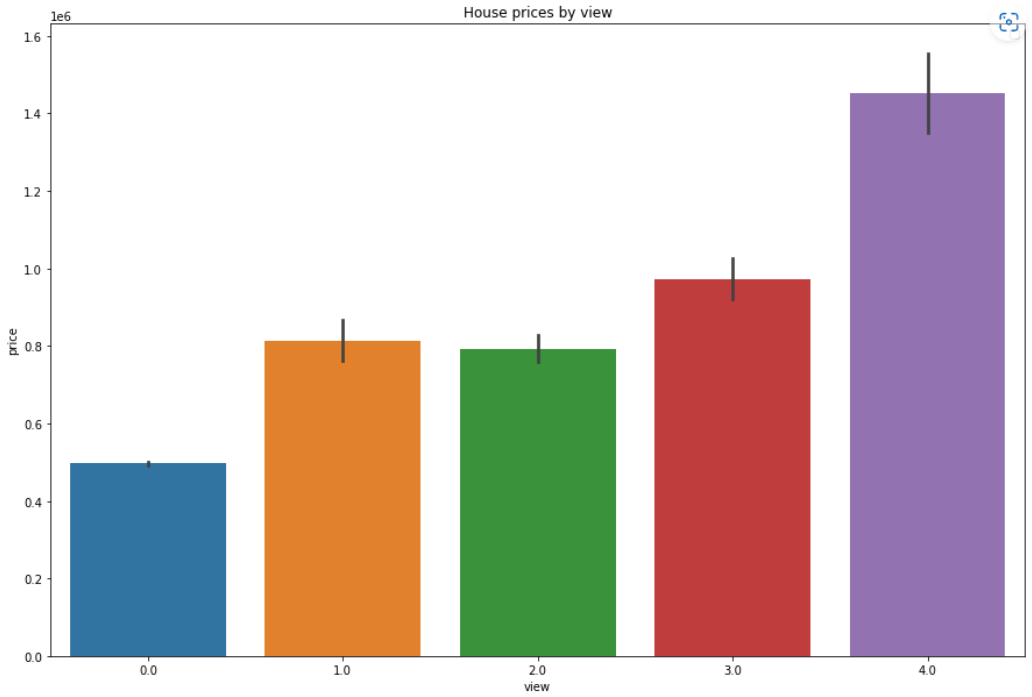

Figure: Prices of the house by views

The above image shows the bar graph against the views and price. In this above plot it can be understood that with the great view the price margin also increased.

```
from sklearn.metrics import mean_squared_error, r2_score
mse = mean_squared_error(y_pred , y_test)
score = r2_score(y_test, y_pred)

print("MSE : ",mse)
print("Score : ",score)
```

```
MSE :  40588007476.24703
Score :  0.6941570370782899
```

```
model.intercept_
```

```
6823320.88883746
```

```
model.coef_
```

```
array([-3.86979081e+04,  4.44105598e+04,  1.50252939e+02,  8.26054642e+03,
        6.07796250e+05,  5.28800990e+04,  2.71279724e+04,  9.53553206e+04,
        3.28877692e+01, -2.72368652e+03, -5.69886525e+02,  2.74696653e+01,
        5.89662072e+05, -2.10345081e+05,  2.05332836e+01, -2.30580542e-01])
```

Figure: MSE and accuracy score of R2

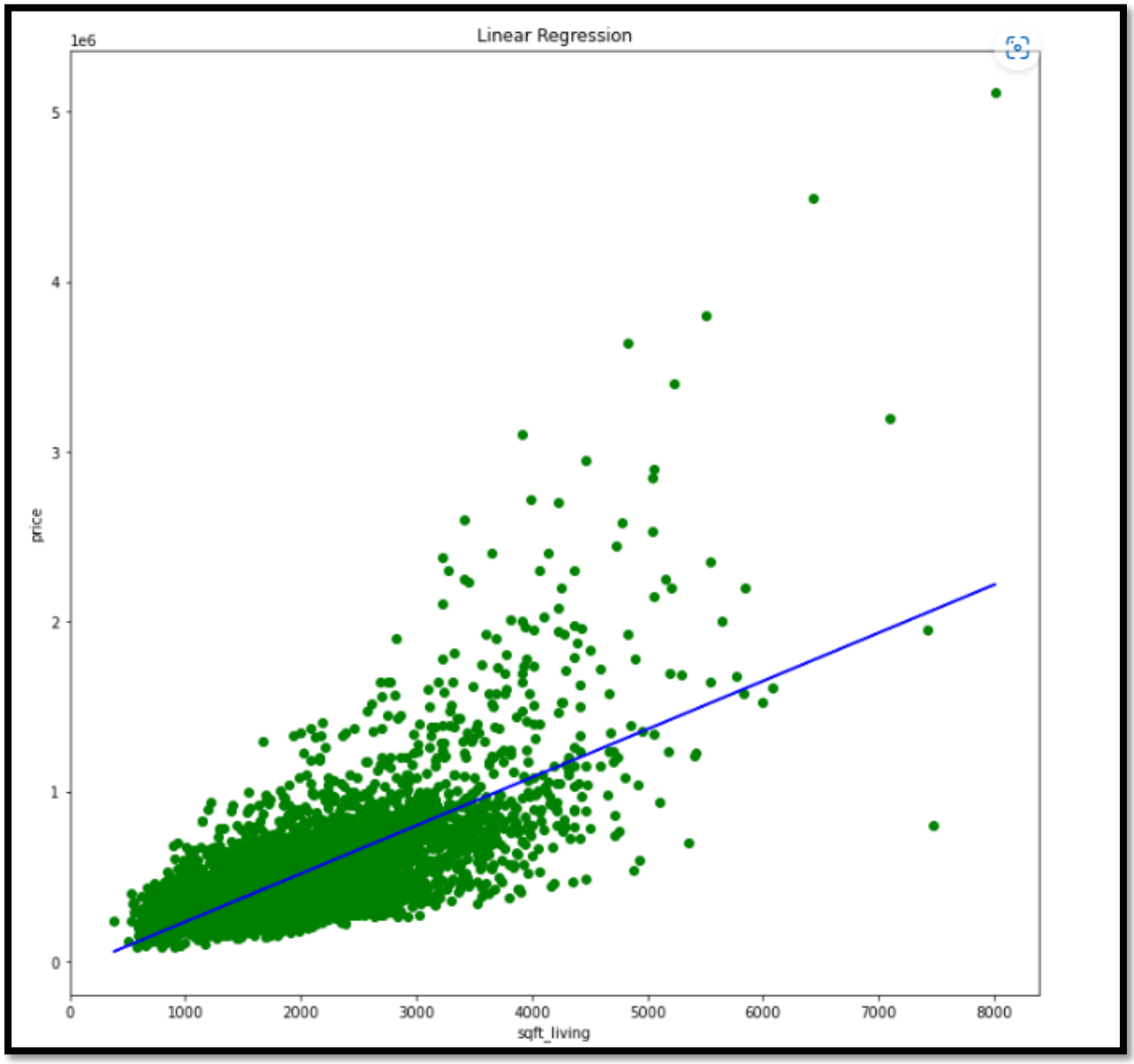

Figure: Linear regression after train test split

```
print("MSE : ",mse)
print("Score : ",score)
```

```
MSE :  63325700668.92157
Score :  0.4682107464832944
```

```
model = LinearRegression()
poly = PolynomialFeatures(degree=4)
poly.fit_transform(x_train)

model.fit(x_train , y_train)
poly.fit_transform(x_test)
y_pred = model.predict(x_test)
mse = mean_squared_error(y_pred , y_test)
score = r2_score(y_test, y_pred)
print("MSE : ",mse)
print("Score : ",score)
```

```
MSE :  63325700668.92157
Score :  0.4682107464832944
```

Figure: Polynomial Regression

Theory:

in our model 3 and 2a r-squared number slightly increased but they number i was looking around 70% above but some reason this model unable to reach that far! Even though we can see that price can increase based on those used variables, but we can only predict 53.3% correctly and rest of the 46.7% reason we don't know! Which means we might need more relevant data!

Brief findings:

From the given dataset we can see that there are multiple variables can influence the price! There are 18 different variables can influence the price with other different many variables which we don't have the data or questions yet! From the correlation heatmap we can see which the variables have more influences on the price! Below I have given a list of the variables have more influences on the price- -bedrooms, bathrooms, floors, sqft_living, sqft_lot', grade, condition, floors…. We want built few models which can tell us which are the variable influences most to predict price! The end user will buyers/ property value estimators but our real estate company (our client here can guide them! Some of the data from buyers might have helped to improve this model! few of the questions could be= what is the family size? What is the purpose of buying the property investment or live in? income of household? Those are answers could help to determine more affordability and that might help me to get the model improved a lot!

Conclusion:

In the conclusion by adding more variables shows our model improves! however it is very hard to determine very perfect price as price depends on other variables like economy, countries current GDP and cash rate etc! Currently happening in Australia for example! House price is going down due to bank interest rate is increasing!

Thank You