Exploratory Data Analysis for Machine Learning
IBM Skills Network

Project Report

By:
Than Phuc Hau

TABLE OF CONTENTS

Brief Description

House price depends on various factors such as number of bedrooms, bathrooms, square footage of living area, of lot, above ground, of basement, number of floors, facing waterfront or not, view rating, condition rating, construction grade, latitude, longitude,.... In this dataset, we want to estimate House price using the above features.

```
import pandas as pd
data = pd.read_csv('lab1_kc_house_data.csv')
data.head()
```

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	...
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	...
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	...
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	...
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	...

5 rows × 21 columns

Initial plan

The plan would go as follows:

- Check for duplicates and deal with any
- Check for missing values and deal with any
- Calculate correlation values
- Check for skewness of data
- Visualize through boxplots to check for outliers
- Apply feature engineering to formulate possible useful features
- Use seaborn pair plots to see underlying patterns
- Construct hypothesis about data set

Data cleaning & Feature engineering

Check for any duplicates

```
data.id.is_unique
```

```
False
```

We will drop rows with duplicates ids.

```
data.drop_duplicates(subset='id',inplace=True)
data
```

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	...
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	...
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	...
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	...
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	...
...	...	...	...	...	...	...	...	...	...	...	...
21608	263000018	20140521T000000	360000.0	3	2.50	1530	1131	3.0	0	0	...
21609	6600060120	20150223T000000	400000.0	4	2.50	2310	5813	2.0	0	0	...
21610	1523300141	20140623T000000	402101.0	2	0.75	1020	1350	2.0	0	0	...
21611	291310100	20150116T000000	400000.0	3	2.50	1600	2388	2.0	0	0	...
21612	1523300157	20141015T000000	325000.0	2	0.75	1020	1076	2.0	0	0	...

21436 rows × 21 columns

```
data.id.is_unique
```

```
True
```

Check for missing value

```
data.isnull().sum()
```

	0
id	0
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

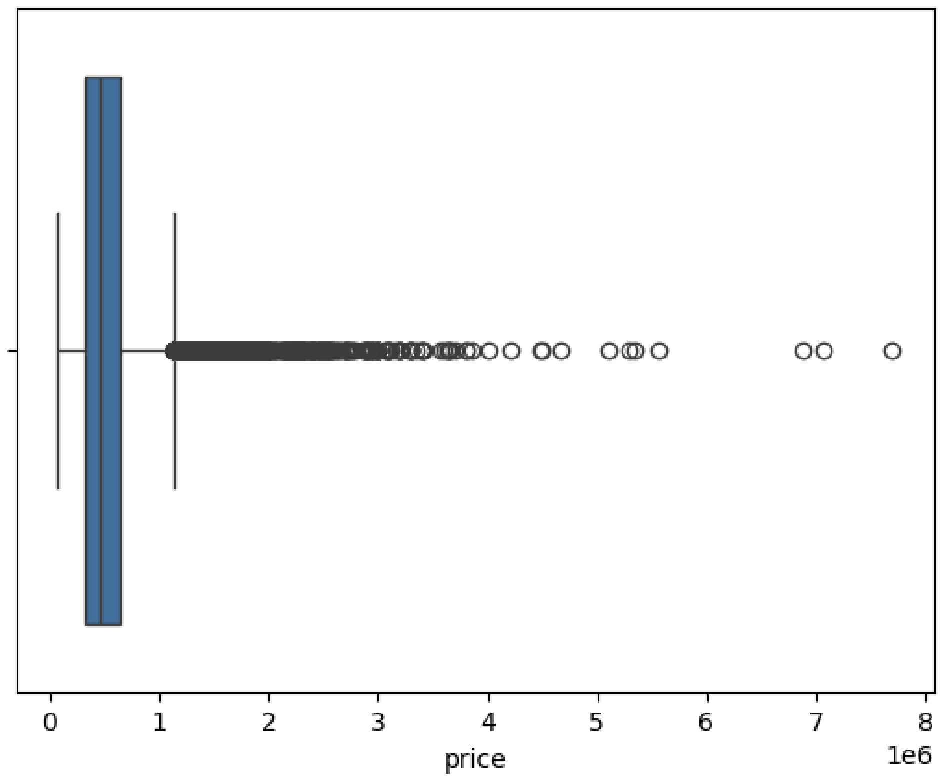
dtype: int64

There is no missing value in these columns, so we will bypass this step.

Check for outliners

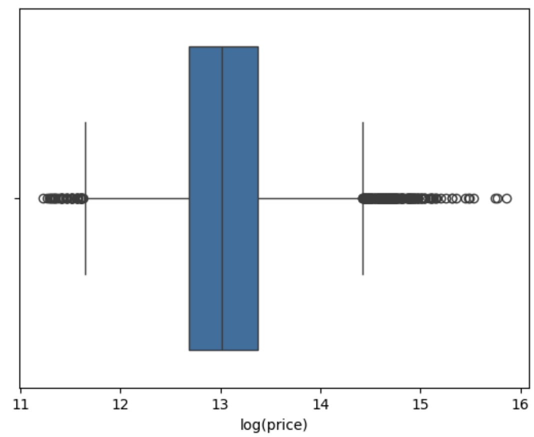
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.boxplot(x=data['price'])
#plt.xlim(0, 1000000)
```

```
<Axes: xlabel='price'>
```



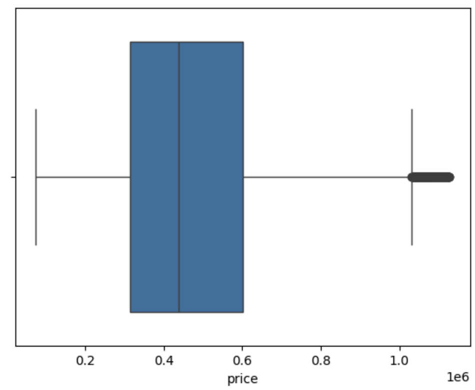
The data points have a fairly left-skewed distribution, and there are many outliers. So we will log data first, then drop some outliers.

```python
import numpy as np
sns.boxplot(x=np.log1p(data['price']))
plt.xlabel("log(price)")
plt.show()
```

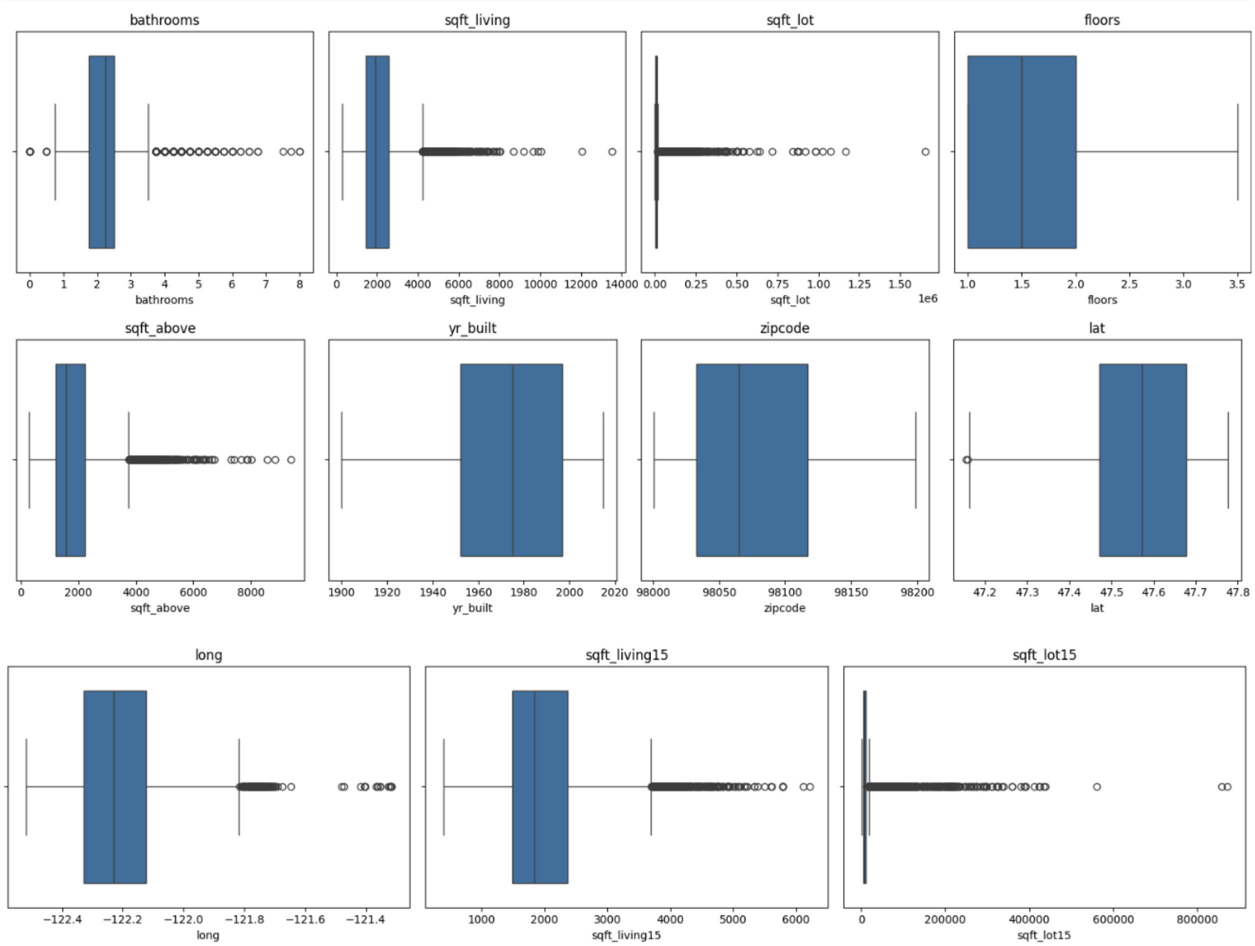



```python
Q1 = data['price'].quantile(0.25)
Q3 = data['price'].quantile(0.75)
IQR = Q3 - Q1
filtered = data[(data['price'] >= Q1 - 1.5*IQR) & (data['price'] <= Q3 + 1.5*IQR)]

sns.boxplot(x=filtered['price'])
plt.show()
```



Below are box-plots for continuous data features.



Calculate skewness value

```
float_cols = [ 'bathrooms',
 'sqft_living',
 'sqft_lot',
 'floors',
 'sqft_above',
 'yr_built',
 'zipcode',
 'lat',
 'long',
 'sqft_living15',
 'sqft_lot15']
skew_limit = 0.75
skew_vals = data[float_cols[2:]].skew()
skew_cols = (skew_vals.sort_values(ascending=False).
             to_frame().rename(columns={0:"Skew"})
             .query('abs(Skew) > {}'.format(skew_limit)))
skew_cols
```

	Skew
sqft_lot	13.043673
sqft_lot15	9.495380
sqft_above	1.444221
sqft_living15	1.105845
long	0.881923

We will apply log transform to all skewed columns.

```
to_skew=[ 'bathrooms',
 'sqft_living',
 'sqft_lot',
 'floors',
 'sqft_above',
 'yr_built',
 'zipcode',
 'lat',
 'long',
 'sqft_living15',
 'sqft_lot15']
for i in to_skew:
  data[i] = np.log1p(data[i])
```

```
float_cols = to_skew
skew_limit = 0.75
skew_vals = data[float_cols[2:]].skew()
skew_cols = (skew_vals.sort_values(ascending=False)
.to_frame().rename(columns={0:"Skew"})
.query('abs(Skew) > {}'.format(skew_limit)))
skew_cols
```

	Skew
sqft_lot15	0.965831
sqft_lot	0.960706

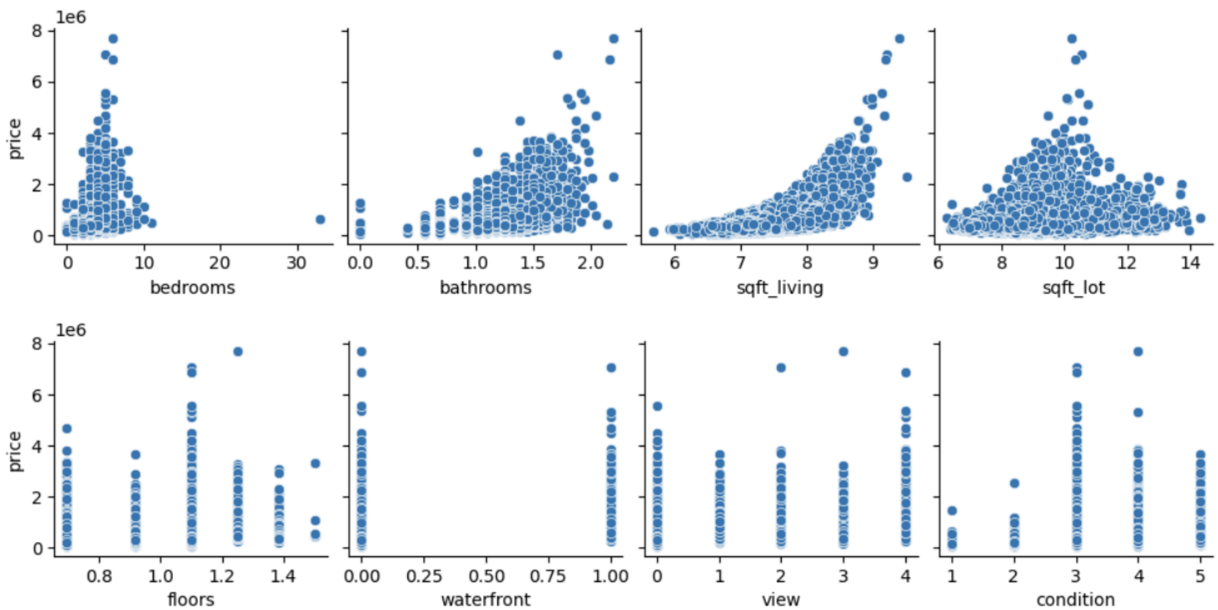
After applying log transformation to our columns, skewness values are mostly corrected. We end up with only 2 columns with skewed values instead of the initial 11 columns.

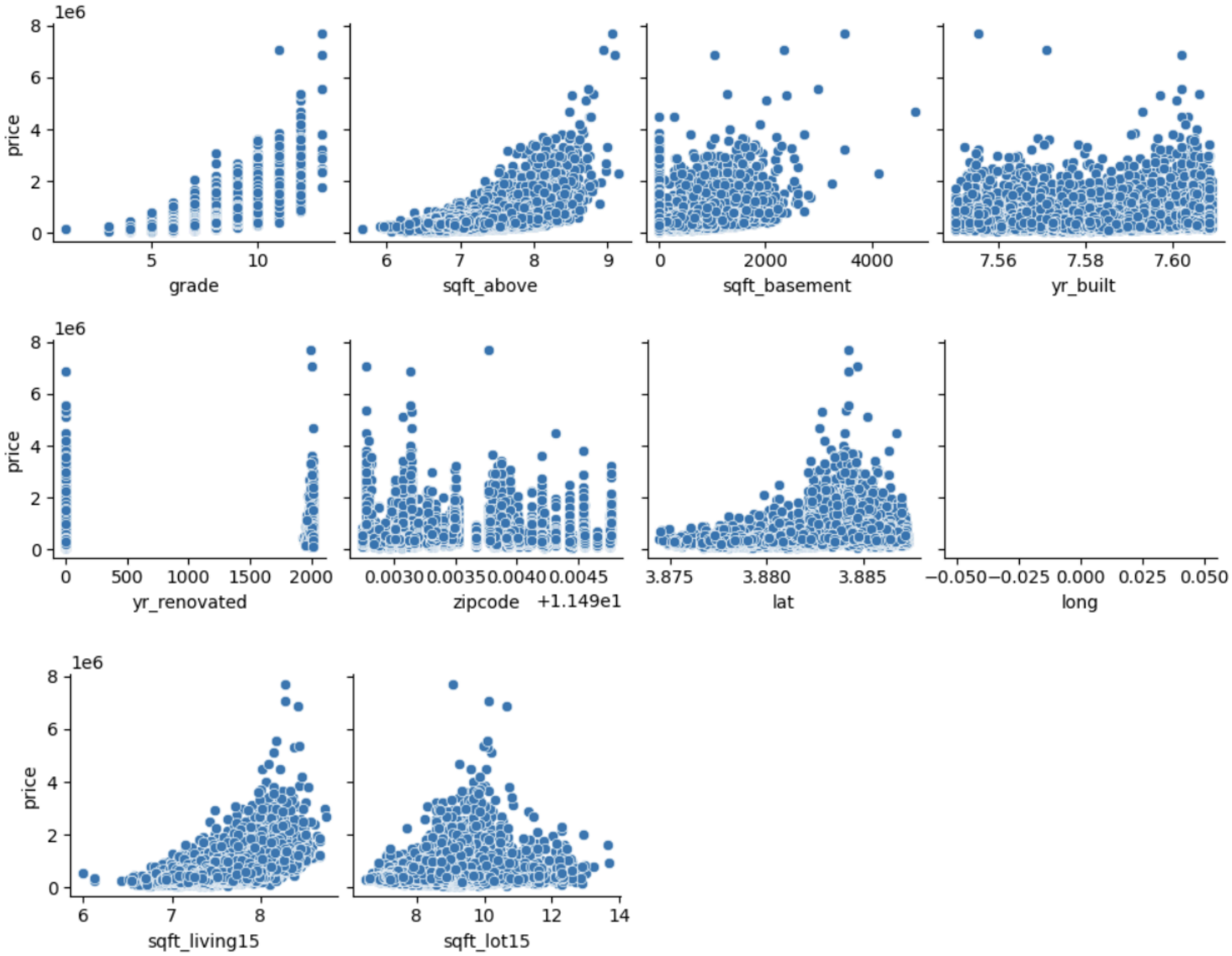
Since some features such as number of rooms, floors, waterfront, view, condition, grade, basement and year renovated are indicator variables, we will not apply skewness and transform on them.

Feature Engineering

There are some notable features in this dataset:

- sqft_living = sqft_above + sqft_basement .
- latitude and longitude features indicate geographical coordinates.
- sqft_living ↔ sqft_living15, sqft_lot ↔ sqft_lot15 often have high correlation.





Calculate statistics (before log transformation)

```
stats_df = data.describe()
stats_df.loc['range'] = stats_df.loc['max'] - stats_df.loc['min']

out_fields = ['mean', '25%','50%','75%','range']
stats_df = stats_df.loc[out_fields]
stats_df.rename({'50%':'median'},inplace = True)
stats_df
```

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above
mean	4.580302e+09	5.400881e+05	3.370842	2.114757	2079.899736	1.510697e+04	1.494309	0.007542	0.234303	3.40943	7.656873	1788.390691
25%	2.123049e+09	3.219500e+05	3.000000	1.750000	1427.000000	5.040000e+03	1.000000	0.000000	0.000000	3.00000	7.000000	1190.000000
median	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	1.500000	0.000000	0.000000	3.00000	7.000000	1560.000000
75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000	1.068800e+04	2.000000	0.000000	0.000000	4.00000	8.000000	2210.000000
range	9.899000e+09	7.625000e+06	33.000000	8.000000	13250.000000	1.650839e+06	2.500000	1.000000	4.000000	4.00000	12.000000	9120.000000

Calculate statistics (after log transformation)

```
stats_df = data.describe()
stats_df.loc['range'] = stats_df.loc['max'] - stats_df.loc['min']

out_fields = ['mean', '25%','50%','75%','range']
stats_df = stats_df.loc[out_fields]
stats_df.rename({'50%':'median'},inplace = True)
stats_df
```

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above
mean	4.580302e+09	5.400881e+05	3.370842	1.105107	7.550910	8.990134	0.891598	0.007542	0.234303	3.40943	7.656873	7.395548
25%	2.123049e+09	3.219500e+05	3.000000	1.011601	7.264030	8.525360	0.693147	0.000000	0.000000	3.00000	7.000000	7.082549
median	3.904930e+09	4.500000e+05	3.000000	1.178655	7.555382	8.938400	0.916291	0.000000	0.000000	3.00000	7.000000	7.353082
75%	7.308900e+09	6.450000e+05	4.000000	1.252763	7.844241	9.276970	1.098612	0.000000	0.000000	4.00000	8.000000	7.701200
range	9.899000e+09	7.625000e+06	33.000000	2.197225	3.840154	8.061360	0.810930	1.000000	4.000000	4.00000	12.000000	3.476311

Insights

We can tell from the pair plots there are many features that have a positive correlation with the price of a house. Let's calculate the correlation values.

```
data_num = data.select_dtypes(include = ['float64','int64'])
corr = data_num.corr()['price'][2:]
top_features = corr[abs(corr) > 0.5].sort_values(ascending=False)
print(f'{len(top_features)} Strongly correlated value: \n {top_features}')
```

```
4 Strongly correlated value:
 grade             0.667434
sqft_living       0.611757
sqft_living15     0.544014
sqft_above        0.542774
Name: price, dtype: float64
```

Hypothesis Testing

We can hypothesize about the data set in several ways. Here are some of the hypotheses we can have about our data set:

- H_0: Average house price in waterfront area (waterfront=1) = Average house price in non-waterfront area (waterfront=0).
- H_1: Average house price in two different groups.

```python
# 1. Waterfront vs Non-Waterfront
waterfront_price = data[data['waterfront'] == 1]['price']
non_waterfront_price = data[data['waterfront'] == 0]['price']

t_stat, p_val = stats.ttest_ind(waterfront_price, non_waterfront_price, equal_var=False)

print("Waterfront vs Non-Waterfront")
print("t-stat =", t_stat, "p-value =", p_val)
if p_val < alpha:
    print("→ Reject H0: There is difference in price between 2 groups\n")
else:
    print("→ Fail to reject H0: There is no significant difference in price\n")
```

```
Waterfront vs Non-Waterfront
t-stat = 12.871572568701405 p-value = 1.4173166497371035e-26
→ Reject H0: There is difference in price between 2 groups
```

- H_0: The average house price of the bedroom groups is the same.
- H_1: There is at least one bedroom group with a different average price.

```python
# 2. Bedrooms vs Price (ANOVA)
groups = [group['price'].values for name, group in data.groupby('bedrooms')]

f_stat, p_val = stats.f_oneway(*groups)

print("Price vs Bedrooms (ANOVA)")
print("F-stat =", f_stat, "p-value =", p_val)
if p_val < alpha:
    print("→ Reject H0: There is difference in price between groups\n")
else:
    print("→ Fail to reject H0: There is no significant difference in price\n")
```

```
Price vs Bedrooms (ANOVA)
F-stat = 213.72837603264605 p-value = 0.0
→ Reject H0: There is difference in price between groups
```

- H_0: There is no linear correlation between sqft_living and price.
- H_1: There is a linear correlation between them.

```python
# 3. Sqft_living vs Price (Pearson correlation)
corr, p_val = stats.pearsonr(data['sqft_living'], data['price'])

print("Sqft_living vs Price (Pearson)")
print("Correlation =", corr, "p-value =", p_val)
if p_val < alpha:
    print("→ Reject H0: There is linear correlation between Square footage of living and Price\n")
else:
    print("→ Fail to reject H0: There is no significant linear correlation between Square footage of living and Price\n")
```

```
Sqft_living vs Price (Pearson)
Correlation = 0.6116498961802913 p-value = 0.0
→ Reject H0: There is linear correlation between Square footage of living and Price
```

Suggestion

The analysis we did on the dataset is just the first step of all possible analysis methods that can be applied to this dataset. The quality of raw data also affects this process. We can try to formulate more features by applying feature engineering such as adding polynomial features, applying scaling methods such as standard scaling or min max scaling. Different visualization methods like heatmap, scatter plot, area chart can also be used to find more statistics about our dataset.

Summary

Predicting the price of a house could help companies choose appropriate features to focus on to increase price and help buyers choose appropriate prices for houses.

In conclusion, I believe that there is much potential in this data set. Although further EDA could be done on this data set and fine-tune it better, we managed to stick to the initial plan at the beginning of this project.