Dataset description

Dataset
This dataset contains house sale prices for King County, which includes Seattle. It includes homes sold between May 2014 and May 2015.

Objective:
Explore and preprocess the data set for regression models.

Data Set Description
There are 21 variables in this data set:
- 3 categorical variables (waterfront, view, condition),
- 16 continuous variables,
- 1 variable to store house ID (Id)
- 1 variable to store date house sold (date)

Structure of the data set

Variable Name	Description	Sample Data
Id	House ID	7129300520; ...
date	Date house sold	20141013T000000; ...
price	House price	221900; 538000; ...
bedrooms	Number of bedrooms	3; 2; ...
bathrooms	Number of bathrooms	1; 2.25; ...
sqft_living	Living room size (in sqft)	1180; 2570
sqft_lot	Lot size (in sqft)	5650; 7242; ...
floors	Number of floors	1; 2; ...
waterfront	Has access to waterway (0 = no; 1 = yes)	0; 1; ...
view	View	0; 2; ...
condition	House condition (1 = bad; 5 = perfect)	1; 5; ...
grade	House grade	7; 6; ...
sqft_above	Above size (in sqft)	1180; 2170; ...
sqft_basement	Basement size (in sqft)	0; 400; ...

yr_built	Year when house was built	1955; 1951; ...
yr_renovated	Year when house was renovated	0; 1991; ...
zipcode	House zipcode	98178; 98125; ...
lat	House latitude	47.5112; 47.721; ...
long	House longitude	-122.257; -122.319; ...
sqft_living15	Living15 (in sqft)	1340; 1690; ...
sqft_lot15	Lot15 (in sqft)	5650; 7639; ...

According to the output below, we have **21613 entries**, 0 to 21612, as well as **21 features**. The "Non-Null Count" column shows the number of non-null entries. In this dataset there is no null entry.
Our target or response variable is 'price' and the rest of the features are our predictor variables.

We also have a mix of numerical (20 int64 or float64) and 1 object data type.

```
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   id             21613 non-null  int64  
 1   date           21613 non-null  object 
 2   price          21613 non-null  float64
 3   bedrooms       21613 non-null  int64  
 4   bathrooms      21613 non-null  float64
 5   sqft_living    21613 non-null  int64  
 6   sqft_lot       21613 non-null  int64  
 7   floors         21613 non-null  float64
 8   waterfront     21613 non-null  int64  
 9   view           21613 non-null  int64  
 10  condition      21613 non-null  int64  
 11  grade          21613 non-null  int64  
 12  sqft_above     21613 non-null  int64  
 13  sqft_basement  21613 non-null  int64  
 14  yr_built       21613 non-null  int64  
 15  yr_renovated   21613 non-null  int64  
 16  zipcode        21613 non-null  int64  
 17  lat            21613 non-null  float64
 18  long           21613 non-null  float64
 19  sqft_living15  21613 non-null  int64  
 20  sqft_lot15     21613 non-null  int64  
dtypes: float64(5), int64(15), object(1)
```

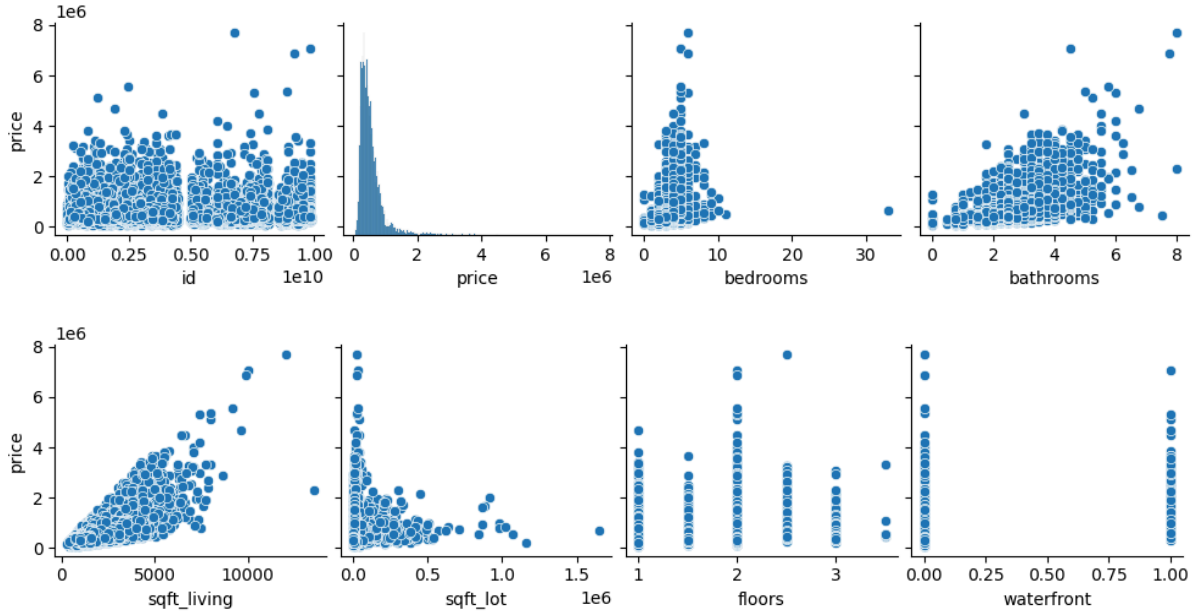
For the "bedrooms", "sqft_living", "sqft_above", "sqft_basement", "sqft_living15" and "sqft_lot15" features, there is a big difference between the 75th percentile and the max value. This could indicate there may be some outliers and/or the data is not normally distributed.

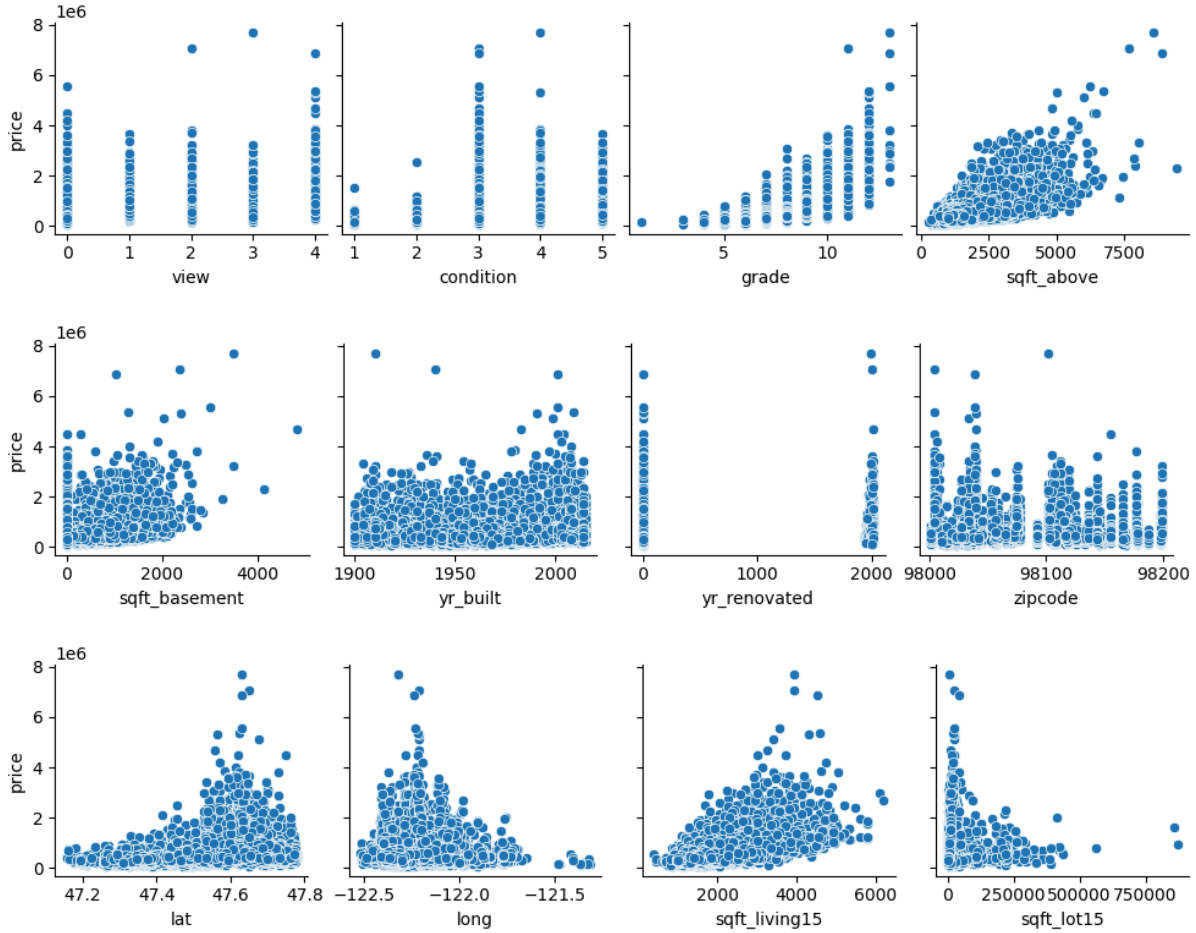
	bedrooms	sqft_living	sqft_above	sqft_basement	sqft_living15
count	21436.000000	21436.000000	21436.000000	21436.000000	21436.000000
mean	3.371571	2082.704936	1790.960440	291.744495	1988.314378
std	0.929205	919.146469	829.026491	442.781983	685.699093
min	0.000000	290.000000	290.000000	0.000000	399.000000
25%	3.000000	1430.000000	1200.000000	0.000000	1490.000000
50%	3.000000	1920.000000	1560.000000	0.000000	1840.000000
75%	4.000000	2550.000000	2220.000000	560.000000	2370.000000
max	33.000000	13540.000000	9410.000000	4820.000000	6210.000000

In this dataset, 'condition' feature has type int64 but it has only 5 values (1-5) so we can treat it like a categorical feature. The same for "view" (values from 0 to 4) and "grade" (values from 1 to 13).

Correlations

Next, let's generate some pair plots to visually inspect the correlation between the features (excluding 'date') and the target variable. Also, building pair plots is one of the possible ways to spot the outliers that might be present in the data.





From the plots above we can draw some conclusions:
- there is a correlation between 'price' and 'bathrooms', 'sqft_living', 'sqft_above', 'grade', 'sqft_living15'
- there may be outliers in 'sqft_lot15', 'sqft_basement', 'sqft_above', 'sqft_lot', 'sqft_living' and 'bedrooms'

Correlations can also be calculated using the Pearson Correlation Coefficient:
There are 5 strongly correlated values with “price”:

sqft_living	0.702035
grade	0.667434
sqft_above	0.605567
sqft_living15	0.585379
bathrooms	0.525138

The correlations calculated with the Pearson Correlation Coefficient corespond with the conclusions drawn from the pair plots.

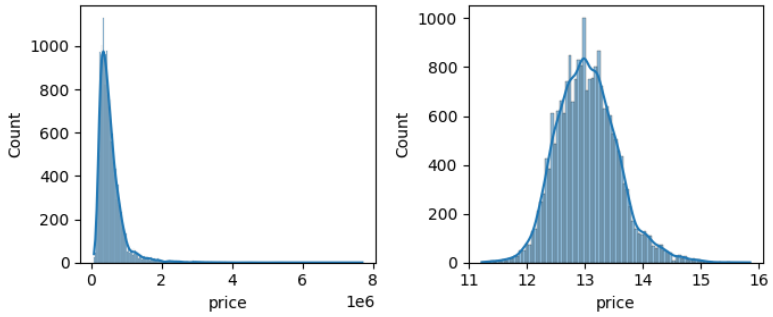
Log transformations

In this section, we are going to inspect whether our 'price' data are normally distributed. The assumption of the normal distribution must be met in order to perform any type of regression analysis.

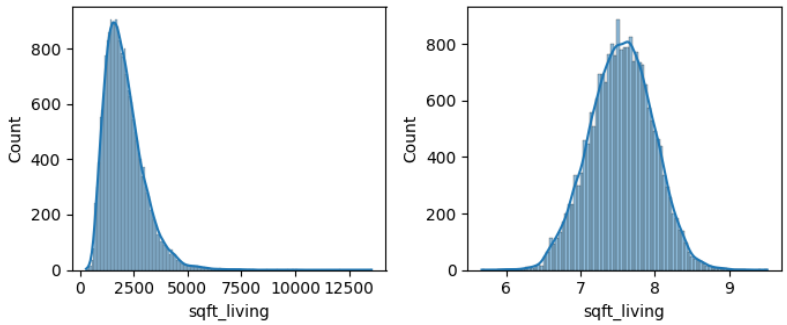
The normal distribution assumption is important because:
- **It helps with accuracy**: When we assume the errors follow a normal distribution, the math behind the scenes works out better, and we get more accurate predictions.
- **Confidence in results**: It helps us create reliable confidence intervals and p-values. These tell us how sure we are about our predictions and whether our findings are significant or just happened by chance.

Transforming features to be closer to a normal distribution can sometimes help, especially if:
- **Improving Model Fit**: Transformations like logarithmic, square root, or Box-Cox can stabilize variance and make relationships more linear, improving the overall fit of the model.
- **Reducing Skewness**: Highly skewed features can lead to less reliable estimates. Transforming them to be more normally distributed can improve the accuracy and interpretability of the regression coefficients.
- **Handling Outliers**: Transformations can mitigate the influence of outliers, making the model more robust.

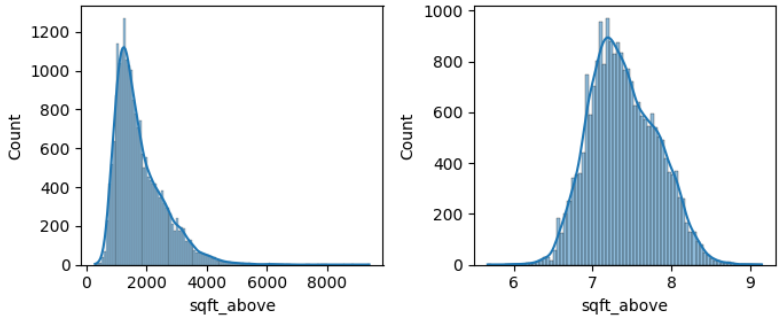
Initial skewness for price: 4.024069
Log transformed column skewness: 0.428072



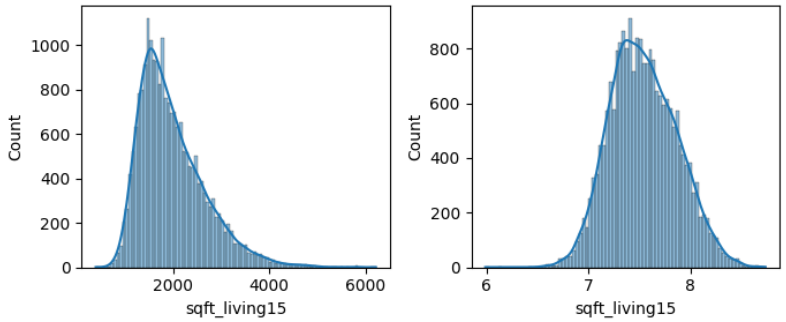
Initial skewness for sqft_living: 1.471555
Log transformed column skewness: -0.035438



Initial skewness: 1.446664
Log transformed column skewness: 0.253384



Initial skewness: 1.108181
Log transformed column skewness: 0.206497



Duplicates

Having duplicate values can effect our analysis, so it is good to check whether there are any duplicates in our data. We search by the 'Id' column, which contains a unique index number for each entry.

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...
94	6021501535	20141223T000000	700000.0	3	1.50	1580	5000	1.0	0	0	...
314	4139480200	20141209T000000	1400000.0	4	3.25	4290	12103	1.0	0	3	...
325	7520000520	20150311T000000	240500.0	2	1.00	1240	12092	1.0	0	0	...
346	3969300030	20141229T000000	239900.0	4	1.00	1000	7134	1.0	0	0	...
372	2231500030	20150324T000000	530000.0	4	2.25	2180	10754	1.0	0	0	...
...	...	...	...	...	...	...	...	...	...	...	...
20181	7853400250	20150219T000000	645000.0	4	3.50	2910	5260	2.0	0	0	...
20613	2724049222	20141201T000000	220000.0	2	2.50	1000	1092	2.0	0	0	...
20670	8564860270	20150330T000000	502000.0	4	2.50	2680	5539	2.0	0	0	...
20780	6300000226	20150504T000000	380000.0	4	1.00	1200	2171	1.5	0	0	...
21581	7853420110	20150504T000000	625000.0	3	3.00	2780	6000	2.0	0	0	...

177 rows × 21 columns

There are 177 duplicated rows which can be deleted.

Missing Values

Using `.isnull()` and `.sum()` functions from pandas we didn't found any missing values.

```
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
```

Feature Scaling

Feature scaling is important in many machine learning algorithms for a few key reasons:

- Improving Model Performance: Algorithms like Gradient Descent converge faster when features are scaled. It helps the algorithm reach the minimum loss more quickly, improving the efficiency of training.
- Equal Weight: Features with larger ranges can dominate the model's decision-making process. Scaling ensures all features contribute equally to the model, leading to better performance.
- Reducing Sensitivity: Models like k-Nearest Neighbors (k-NN) and Support Vector Machines (SVM) are sensitive to the scale of features because they rely on distance measurements. Unscaled features can distort these distances and degrade model accuracy.
- Consistency: Features on different scales can lead to inconsistent behavior in tree-based algorithms, especially when making splits. Scaling ensures more consistent and reliable splits.

Common methods of feature scaling include:

- Standardization: Transforming features to have a mean of 0 and a standard deviation of 1.
- Min-Max Scaling: Rescaling features to a fixed range, typically [0, 1].

Using Standard Scaler we can transform the features:

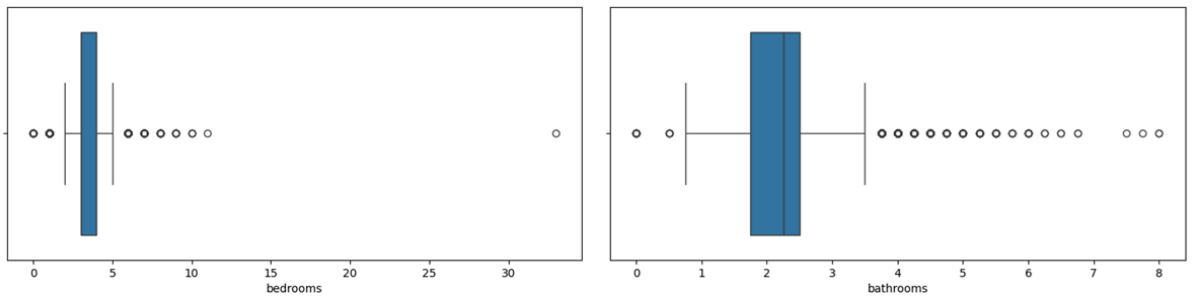
code:
```
data_std_scaled = StandardScaler().fit_transform(data[['price',
'sqft_living', 'sqft_above', 'sqft_living15', 'lat', 'long']])
```

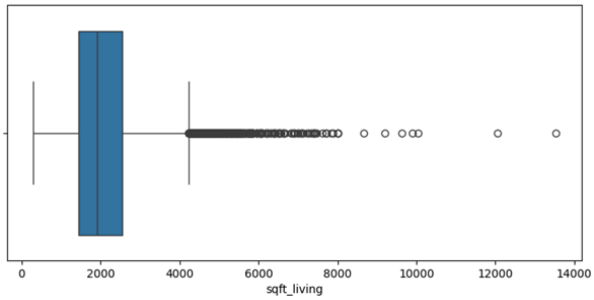
output:
```
[[-0.86659232 -0.98213508 -0.73697849 -0.94550147 -0.35322574
-0.30734733]
 [-0.00687903  0.53017277  0.4572211  -0.43506159  1.16050374
-0.74739853]
 [-0.98054988 -1.4282115  -1.231546    1.06709006  1.28243905
-0.13700494]
 ...
 [-0.37649047 -1.15621368 -0.92998044 -1.41218936  0.24707117
-0.60544653]
 [-0.38220467 -0.52517875 -0.23034836 -0.84341349 -0.18511375
1.02700145]
 [-0.58618599 -1.15621368 -0.92998044 -1.41218936  0.24490664
-0.60544653]]
```

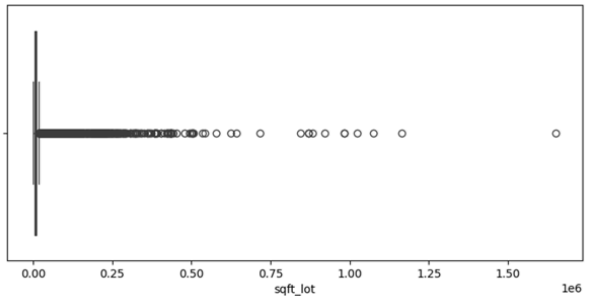
Outliers

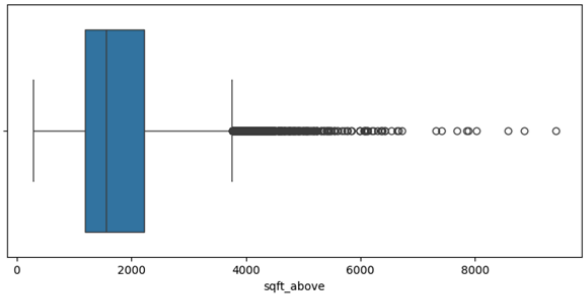
An outlier is a data point that is significantly different from the other data points in a dataset. It stands out because it lies far away from the majority of values. Outliers can occur due to variability in the data or because of measurement errors.
Impact on Analysis: Outliers can skew and mislead statistical analyses, affecting mean and standard deviation, and influencing the results of regression models and other analyses.

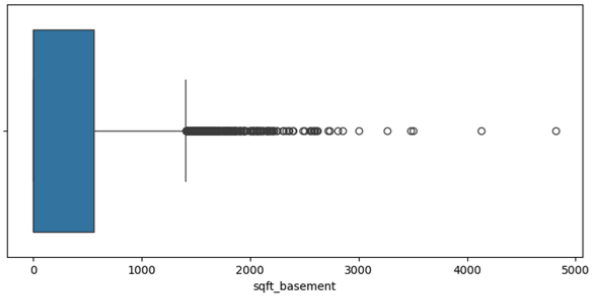
Below there are boxplots and scatterplots that can be used to identify visually the outliers from the dataset:

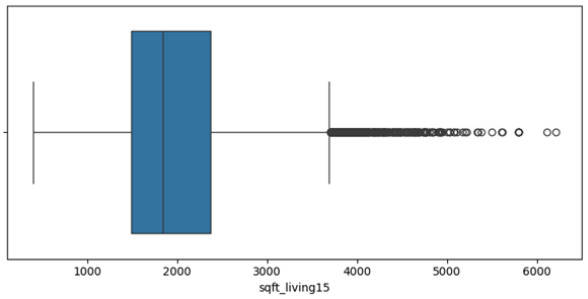


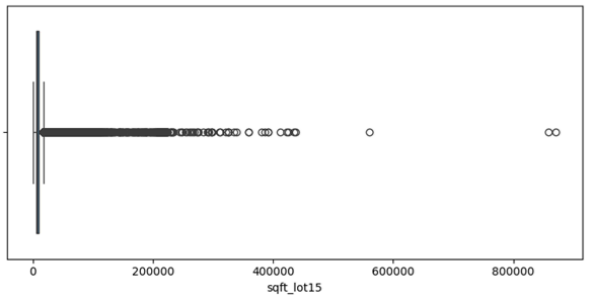

0
2000
4000
6000
8000
10000
12000
14000
sqft_living

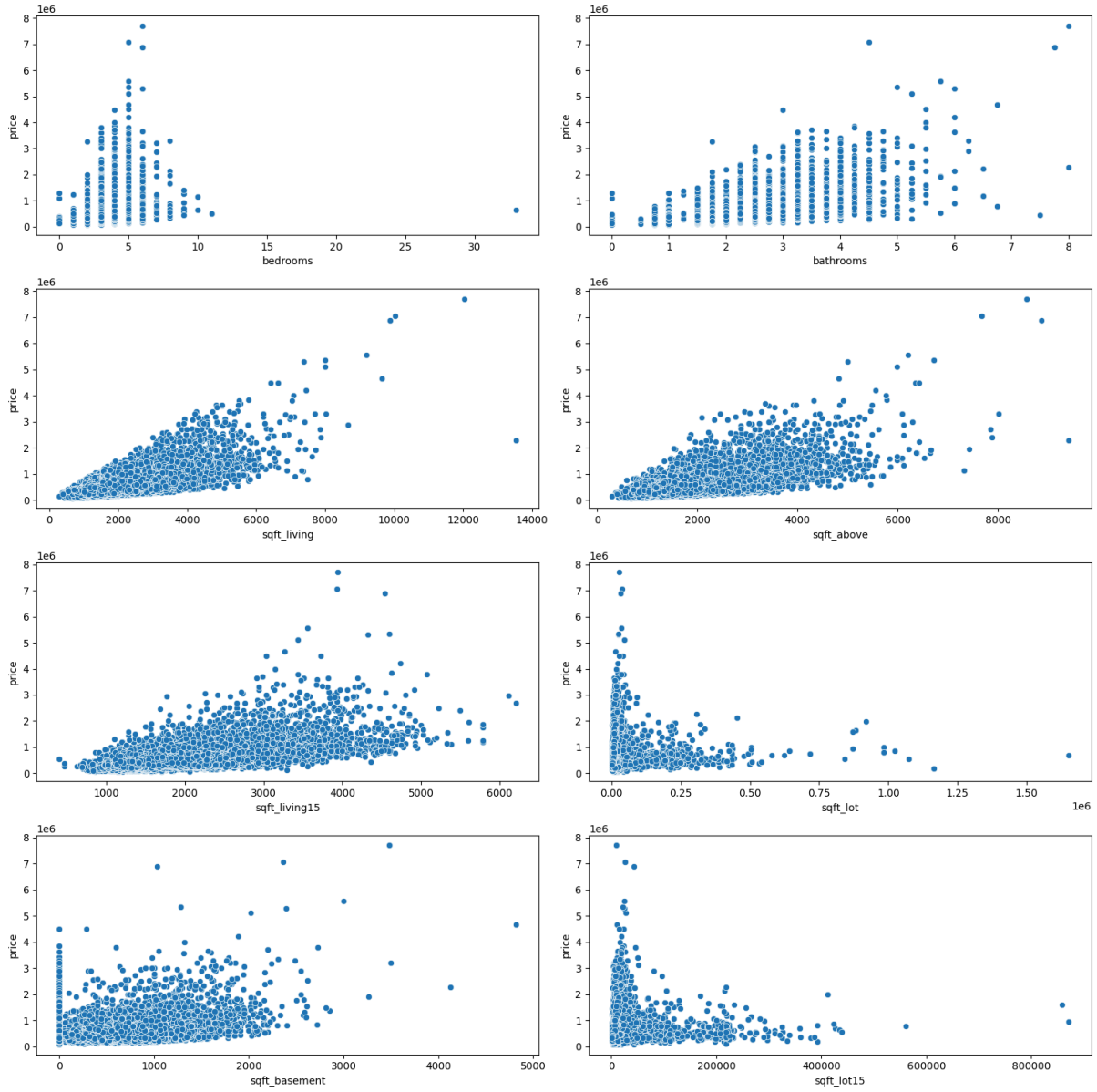

0.00
0.25
0.50
0.75
1.00
1.25
1.50
1e6
sqft_lot


0
2000
4000
6000
8000
sqft_above


0
1000
2000
3000
4000
5000
sqft_basement


1000
2000
3000
4000
5000
6000
sqft_living15


0
200000
400000
600000
800000
sqft_lot15

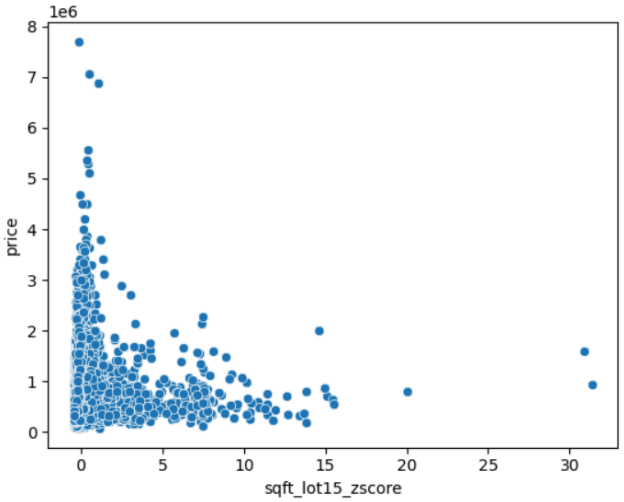


From the above plots, we can see there are some data points that deviate from the rest of the population. We can delete those outliers.

Z-score Analysis

Z-score is another way to identify outliers mathematically. Z-score is the signed number of standard deviations by which the value of an observation or data point is above the mean value of what is being observed or measured.

After calculating the z-score for column “sqft_lot15”, we can see there are 3 datapoint that deviate from the rest of the population. These datapoints can be considered outliers.



Feature Transformation

Categorical Variables

"View" and "condition" features have each 5 types of values so we can one hot encode them using pandas `get_dummies()` function. This will result in extra 10 columns, one for each type of value.

	view_0	view_1	view_2	view_3	view_4	condition_1	condition_2	condition_3	condition_4	condition_5
0	True	False	False	False	False	False	False	True	False	False
1	True	False	False	False	False	False	False	True	False	False
2	True	False	False	False	False	False	False	True	False	False
3	True	False	False	False	False	False	False	False	False	True
4	True	False	False	False	False	False	False	True	False	False

Dealing with Dates

Calculating the time since the houses have been constructed or renovated can help accuracy for machine learning algorithms like decision trees.

code:

```
current_year = datetime.now().year
clean_data['age_built'] = current_year - clean_data['yr_built']
clean_data['age_renovated'] =
clean_data['yr_renovated'].where(clean_data['yr_renovated'] == 0,
current_year - clean_data['yr_renovated'])
```

The code above will generate 2 new columns: 'age_built' and 'age_renovated':

	yr_built	age_built	yr_renovated	age_renovated
0	1955	70	0	0
1	1951	74	1991	34
2	1933	92	0	0
3	1965	60	0	0
4	1987	38	0	0

Hypothesis Testing

Hypothesis 1:

Prove (or disprove) if the average price of houses on waterfronts is bigger than the average price of houses not on waterfronts.

H_0: The average price of houses on waterfronts are less than or equal to prices of houses not on waterfronts.
H_A: The average price of houses on waterfronts are greater than prices of houses not on waterfronts.

The '>' sign in the alternate hypothesis indicates the test is right tailed. To compare the mean values of waterfront house and nonwaterfront house populations, we will use a t-test. If z-values (calculated from a t-test) fall into the area on the right side of a distribution curve, this would cause us to reject the null hypothesis.

Using `scipy.stats.ttest_ind()` we can calculate the z_values:

```
t_value = 40.55092950214794 , p_value =0.0 , p_value_onetail = 0.0
```

Conclusion: Since p_value 0.0 is less than alpha 0.05 , we reject the null hypothesis that the average price of houses on waterfronts are greater than average prices of houses not on waterfronts.

Other possible hypothesis to be tested:

Hypothesis 2:

H_0: The mean price of houses with different condition values (1-5) are the same.
H_A: At least one of the means is different.

Hypothesis 3:

H_0: Houses on waterfront proportions are not significantly different across the different grade values.
H_A: Houses on waterfront proportions are different across the different grade values.

Conclusion

The dataset has the advantage of no missing values. This makes the analysis a bit simpler and also the accuracy of a future machine learning can benefit from that.

We saw that some of the features can benefit from log transformation. Also any duplicated values are easy to find and delete.