In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:

```
df= pd.read_csv('kc_house_data.csv')
df.head()
```

Out[2]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterf
0	7129300520	10/13/2014	221900.0	3	1.00	1180	5650	1.0	
1	6414100192	12/9/2014	538000.0	3	2.25	2570	7242	2.0	
2	5631500400	2/25/2015	180000.0	2	1.00	770	10000	1.0	
3	2487200875	12/9/2014	604000.0	4	3.00	1960	5000	1.0	
4	1954400510	2/18/2015	510000.0	3	2.00	1680	8080	1.0	

5 rows × 21 columns

In [3]:

```
df.describe()
```

Out[3]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	
count	2.159700e+04	2.159700e+04	21597.000000	21597.000000	21597.000000	2.159700e+04	2
mean	4.580474e+09	5.402966e+05	3.373200	2.115826	2080.321850	1.509941e+04	
std	2.876736e+09	3.673681e+05	0.926299	0.768984	918.106125	4.141264e+04	
min	1.000102e+06	7.800000e+04	1.000000	0.500000	370.000000	5.200000e+02	
25%	2.123049e+09	3.220000e+05	3.000000	1.750000	1430.000000	5.040000e+03	
50%	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	
75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000	1.068500e+04	
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	

In [4]:

```
df['zipcode'].isnull()
```

Out[4]:

```
0        False
1        False
2        False
3        False
4        False
         ...
21592    False
21593    False
21594    False
21595    False
21596    False
Name: zipcode, Length: 21597, dtype: bool
```

In [5]:

```
df['zipcode'].dtypes
```

Out[5]:

```
dtype('int64')
```

In [6]:

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

Out[6]:

```
id              -0.016772
price            1.000000
bedrooms         0.308787
bathrooms        0.525906
sqft_living      0.701917
sqft_lot         0.089876
floors           0.256804
waterfront       0.276295
view             0.395734
condition        0.036056
grade            0.667951
sqft_above       0.605368
yr_built         0.053953
yr_renovated     0.129599
zipcode         -0.053402
lat              0.306692
long             0.022036
sqft_living15    0.585241
sqft_lot15       0.082845
Name: price, dtype: float64
```

In [7]:

```
# dropping Longitude, ID since they are not greatly correlated to price for analysis.
# Dropping "date" since it is in date and time format, which will not be analysis friendly.


df = df.drop(['long','id','date'], axis=1)
df
```

Out[7]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition
0	221900.0	3	1.00	1180	5650	1.0	NaN	0.0	3
1	538000.0	3	2.25	2570	7242	2.0	0.0	0.0	3
2	180000.0	2	1.00	770	10000	1.0	0.0	0.0	3
3	604000.0	4	3.00	1960	5000	1.0	0.0	0.0	5
4	510000.0	3	2.00	1680	8080	1.0	0.0	0.0	3
...	...	...	...	...	...	...	...	...	..
21592	360000.0	3	2.50	1530	1131	3.0	0.0	0.0	3
21593	400000.0	4	2.50	2310	5813	2.0	0.0	0.0	3
21594	402101.0	2	0.75	1020	1350	2.0	0.0	0.0	3
21595	400000.0	3	2.50	1600	2388	2.0	NaN	0.0	3
21596	325000.0	2	0.75	1020	1076	2.0	0.0	0.0	3

21597 rows × 18 columns

In [8]:

```
df.isna().sum()
```

Out[8]:

```
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
sqft_living15       0
sqft_lot15          0
dtype: int64
```

In [9]:

```
#  cleaning sqft_basement column, by converting it to float and replacing ? with 0.
#  we are considering 0 as no basement available for the property

df['sqft_basement'] = df['sqft_basement'].str.replace("?", "0", regex=True).astype(float)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 18 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   price          21597 non-null  float64
 1   bedrooms       21597 non-null  int64  
 2   bathrooms      21597 non-null  float64
 3   sqft_living    21597 non-null  int64  
 4   sqft_lot       21597 non-null  int64  
 5   floors         21597 non-null  float64
 6   waterfront     19221 non-null  float64
 7   view           21534 non-null  float64
 8   condition      21597 non-null  int64  
 9   grade          21597 non-null  int64  
 10  sqft_above     21597 non-null  int64  
 11  sqft_basement  21597 non-null  float64
 12  yr_built       21597 non-null  int64  
 13  yr_renovated   17755 non-null  float64
 14  zipcode        21597 non-null  int64  
 15  lat            21597 non-null  float64
 16  sqft_living15  21597 non-null  int64  
 17  sqft_lot15     21597 non-null  int64  
dtypes: float64(8), int64(10)
memory usage: 3.0 MB
```

In [10]:

```
#  Cleaning other columns by filling 0 replacing NA

df ['yr_renovated'] = df ['yr_renovated'].fillna(0)
df['waterfront'] = df['waterfront'].fillna(0)
df['view'] = df['view'].fillna(0)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 18 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   price          21597 non-null  float64
 1   bedrooms       21597 non-null  int64  
 2   bathrooms      21597 non-null  float64
 3   sqft_living    21597 non-null  int64  
 4   sqft_lot       21597 non-null  int64  
 5   floors         21597 non-null  float64
 6   waterfront     21597 non-null  float64
 7   view           21597 non-null  float64
 8   condition      21597 non-null  int64  
 9   grade          21597 non-null  int64  
 10  sqft_above     21597 non-null  int64  
 11  sqft_basement  21597 non-null  float64
 12  yr_built       21597 non-null  int64  
 13  yr_renovated   21597 non-null  float64
 14  zipcode        21597 non-null  int64  
 15  lat            21597 non-null  float64
 16  sqft_living15  21597 non-null  int64  
 17  sqft_lot15     21597 non-null  int64  
dtypes: float64(8), int64(10)
memory usage: 3.0 MB
```

In [11]:

```
df['view'].value_counts()
```

Out[11]:

```
0.0    19485
2.0      957
3.0      508
1.0      330
4.0      317
Name: view, dtype: int64
```

In [12]:

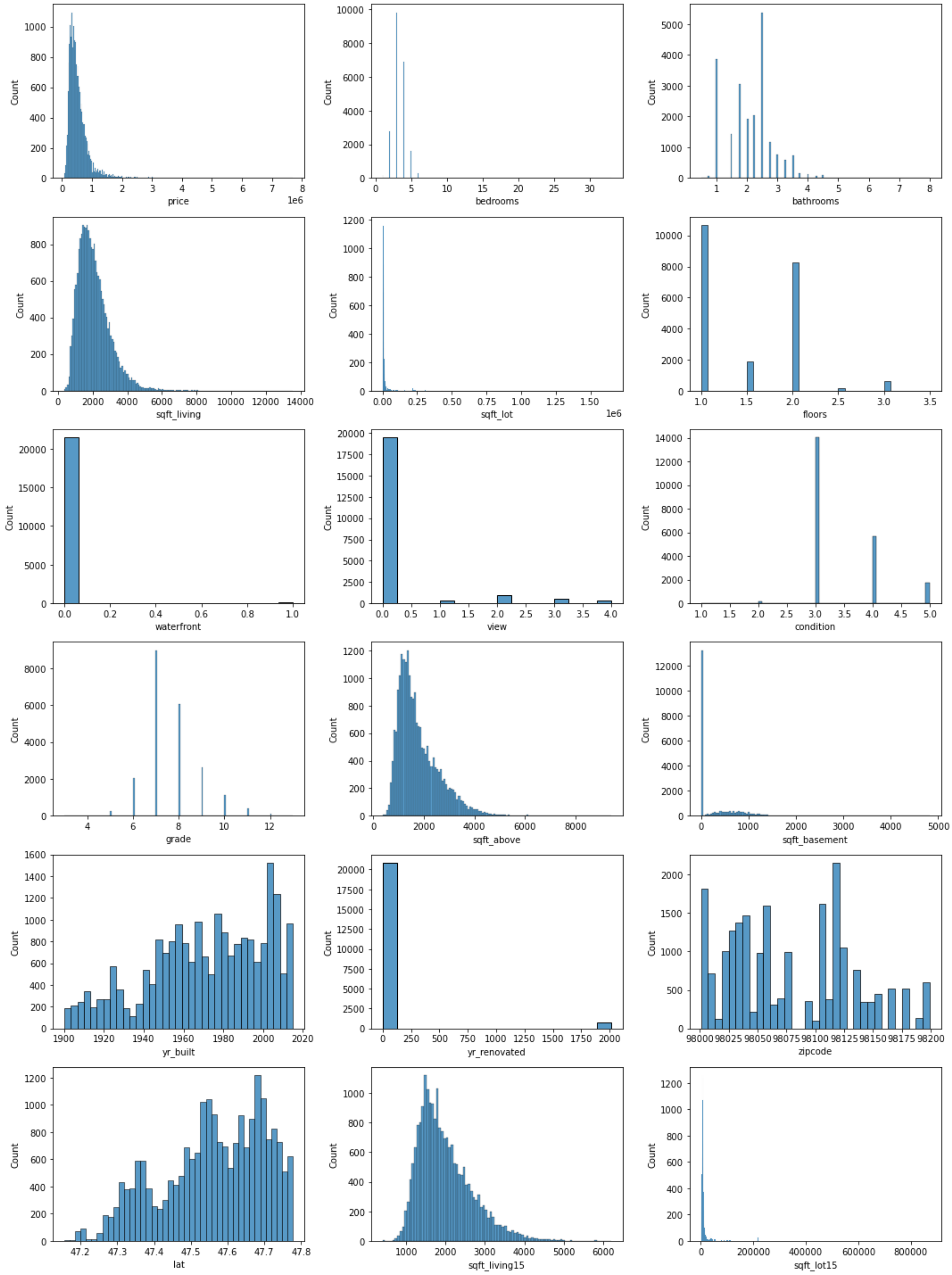
```
#  recognising continuous variables

conti_var = df.describe(include = np.number).columns.tolist()
conti_var
```

Out[12]:

```
['price',
 'bedrooms',
 'bathrooms',
 'sqft_living',
 'sqft_lot',
 'floors',
 'waterfront',
 'view',
 'condition',
 'grade',
 'sqft_above',
 'sqft_basement',
 'yr_built',
 'yr_renovated',
 'zipcode',
 'lat',
 'sqft_living15',
 'sqft_lot15']
```

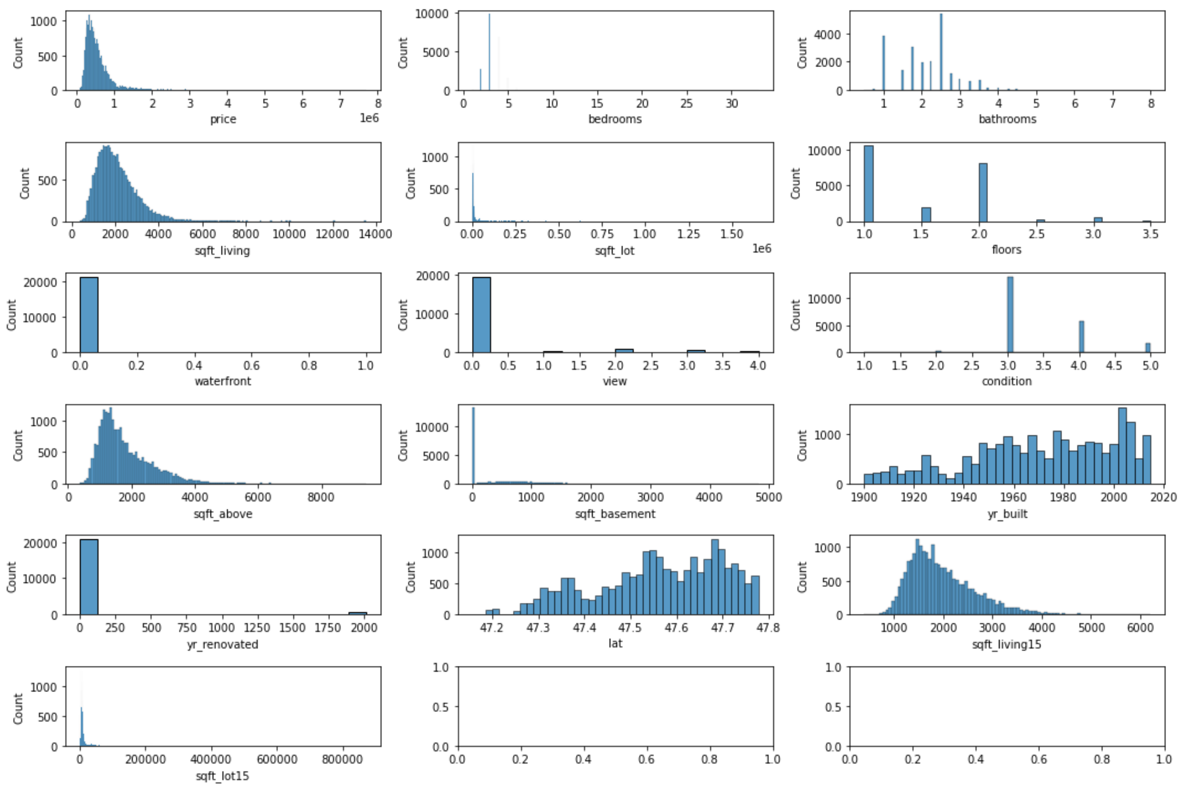
In [13]:

```
fig, ax = plt.subplots(len(conti_var) //3 + int(np.ceil(len(conti_var)%3)),3, figsize=(15,
for i, c in enumerate(conti_var):
    sns.histplot(df[c], ax=ax[i//3, i%3])
fig.tight_layout()
```



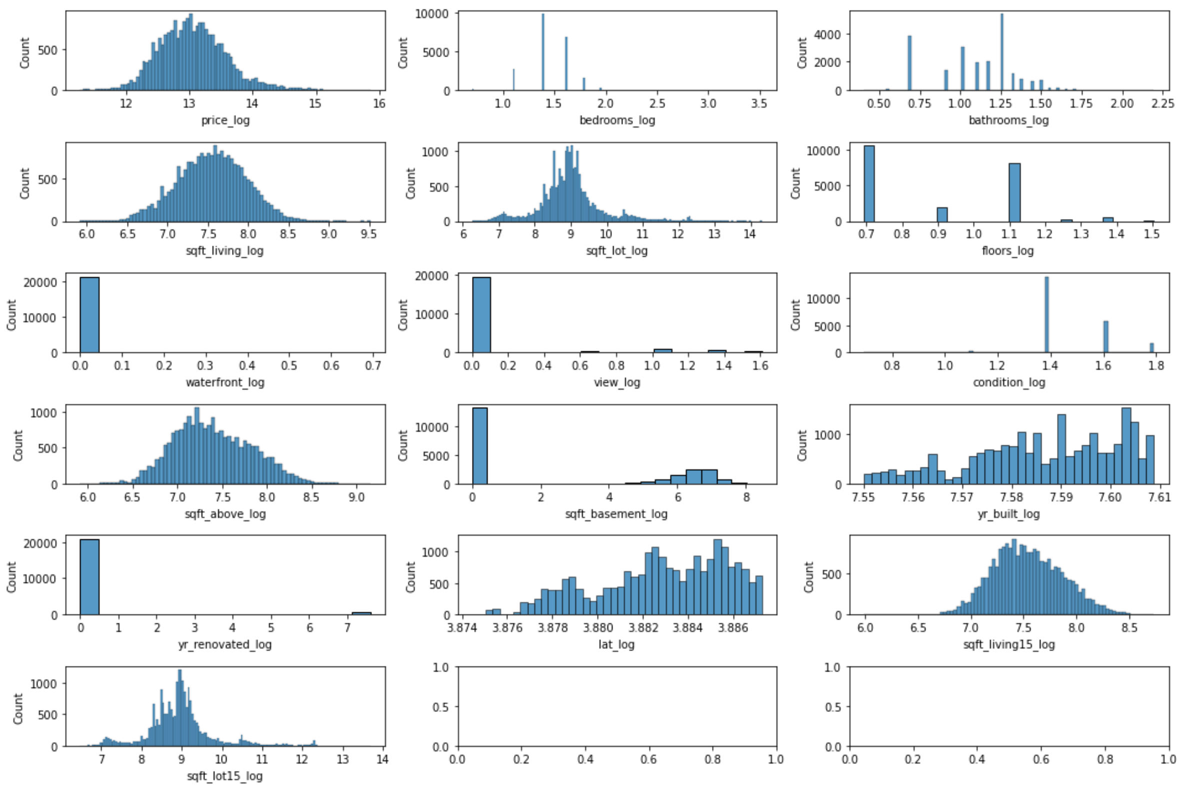
In [14]:

```
#  Selecting non zero continous variables for log transformation. Plotting to make sure.

nz_contivar = ['price','bedrooms','bathrooms','sqft_living', 'sqft_lot',
               'floors','waterfront','view','condition','sqft_above','sqft_basement','yr_bu
               'lat','sqft_living15','sqft_lot15']
df_conti = df[nz_contivar]

fig, ax = plt.subplots(len(nz_contivar)//3 + int(np.ceil(len(nz_contivar) %3)), 3, figsize=
for i, c in enumerate (nz_contivar):
    sns.histplot(df_conti[c], ax=ax [i//3, i%3])
fig.tight_layout()
```



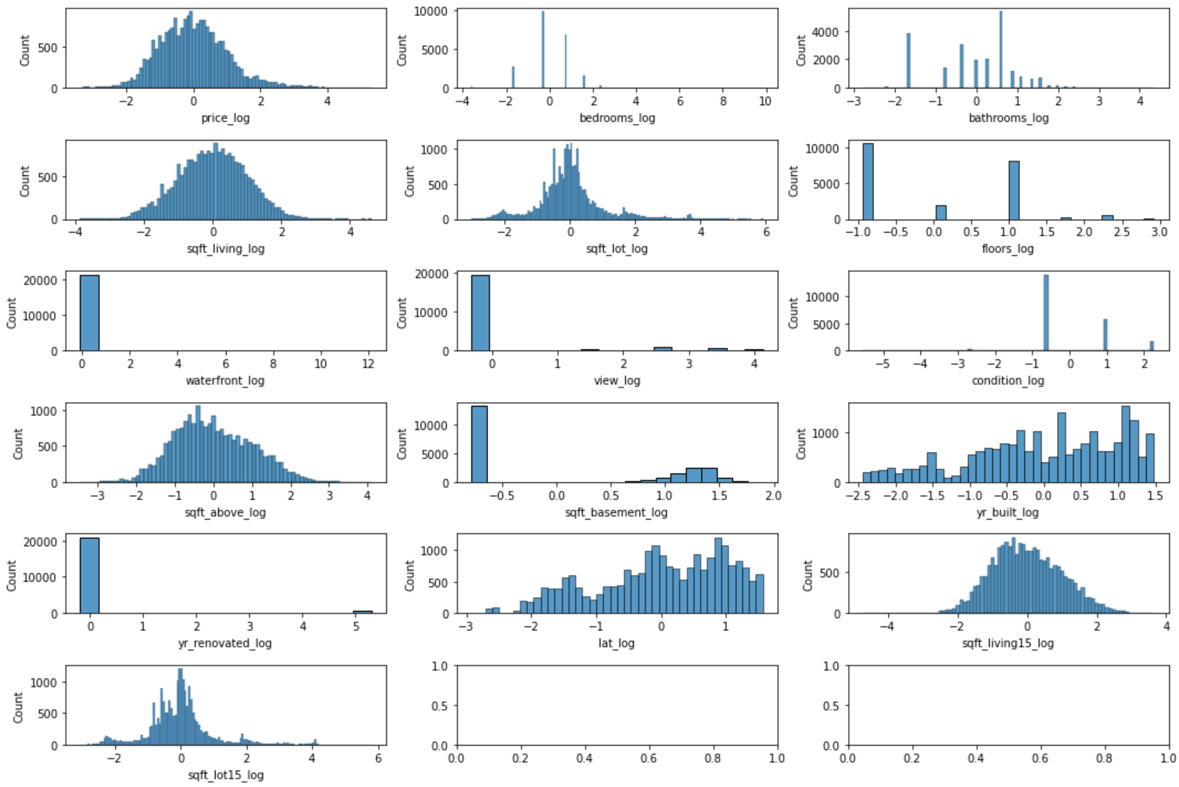
In [15]:

```
#  perform log transformations

log_var = [f'{c}_log'for c in nz_contivar]

df_log = np.log1p(df_conti)
df_log.columns = log_var

fig, ax = plt.subplots(len(log_var)//3 +int(np.ceil(len(log_var)% 3)),3, figsize=(15,10))
for i, c in enumerate(log_var):
    sns.histplot(df_log[c], ax=ax[i//3, i%3])
fig.tight_layout()
```



In [16]:

```
# standardising continuous variables

standardise_conti = df_log.apply(lambda x: (x - x.mean()) / x.std())
fig, ax = plt.subplots(len(log_var) // 3 + int(np.ceil(len(log_var) % 3)), 3, figsize=(15,
for i, c in enumerate(log_var):
    sns.histplot(standardise_conti[c], ax=ax[i // 3, i % 3])
fig.tight_layout()
```



In [17]:

```
# recognising Discreet variables
raw_var =df[['view','condition','grade','waterfront','bedrooms','bathrooms','floors']]
disc_var =['zipcode']
raw_var
```

Out[17]:

	view	condition	grade	waterfront	bedrooms	bathrooms	floors
0	0.0	3	7	0.0	3	1.00	1.0
1	0.0	3	7	0.0	3	2.25	2.0
2	0.0	3	6	0.0	2	1.00	1.0
3	0.0	5	7	0.0	4	3.00	1.0
4	0.0	3	8	0.0	3	2.00	1.0
...	...	...	...	...	...	...	...
21592	0.0	3	8	0.0	3	2.50	3.0
21593	0.0	3	8	0.0	4	2.50	2.0
21594	0.0	3	7	0.0	2	0.75	2.0
21595	0.0	3	8	0.0	3	2.50	2.0
21596	0.0	3	7	0.0	2	0.75	2.0

21597 rows × 7 columns

In [18]:

```
# converting into one hot encoding

from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder(drop='first')
enc.fit(df[disc_var])
ohe_cols = enc.transform(df[disc_var])
ohe_cols.toarray().shape
```

Out[18]:

```
(21597, 69)
```

In [19]:

```
#  turn this into dataframe

ohe_df =pd.DataFrame(ohe_cols.toarray(), columns=enc.get_feature_names_out(disc_var))
ohe_df.head()
```

Out[19]:

	zipcode_98002	zipcode_98003	zipcode_98004	zipcode_98005	zipcode_98006	zipcode_98007
0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 69 columns

In [20]:

```
# raw_df =pd.DataFrame(raw_var)
# raw_df
```

In [21]:

```
#  joining continuous standardised variables and one hot encoded variabels in a DataFrame

df = standardise_conti.join(ohe_df, how='left')

df = df.join(raw_var, how ='left')
df
```

Out[21]:

	price_log	bedrooms_log	bathrooms_log	sqft_living_log	sqft_lot_log	floors_log	wat
0	-1.401997	-0.320351	-1.644140	-1.125564	-0.388490	-0.947087	
1	0.279938	-0.320351	0.290975	0.709416	-0.113302	0.988608	
2	-1.799429	-1.687516	-1.644140	-2.131418	0.244426	-0.947087	
3	0.499698	0.740104	1.118575	0.070561	-0.523969	-0.947087	
4	0.178433	-0.320351	-0.028055	-0.292847	0.008081	-0.947087	
...	...	...	...	...	...	...	
21592	-0.483049	-0.320351	0.586352	-0.513318	-2.171142	2.362005	
21593	-0.282955	0.740104	0.586352	0.457935	-0.356962	0.988608	
21594	-0.273006	-1.687516	-2.176364	-1.468954	-1.975055	0.988608	
21595	-0.282955	-0.320351	0.586352	-0.407862	-1.343051	0.988608	
21596	-0.677291	-1.687516	-2.176364	-1.468954	-2.226364	0.988608	

21597 rows × 92 columns

In [22]:

```
#  defininig Y variable

num_vars = df.describe().columns.tolist()
y_var = 'price_log'
num_vars = [c for c in df if c != y_var]
print(num_vars)
```

```
['bedrooms_log', 'bathrooms_log', 'sqft_living_log', 'sqft_lot_log', 'floors
_log', 'waterfront_log', 'view_log', 'condition_log', 'sqft_above_log', 'sqf
t_basement_log', 'yr_built_log', 'yr_renovated_log', 'lat_log', 'sqft_living
15_log', 'sqft_lot15_log', 'zipcode_98002', 'zipcode_98003', 'zipcode_9800
4', 'zipcode_98005', 'zipcode_98006', 'zipcode_98007', 'zipcode_98008', 'zip
code_98010', 'zipcode_98011', 'zipcode_98014', 'zipcode_98019', 'zipcode_980
22', 'zipcode_98023', 'zipcode_98024', 'zipcode_98027', 'zipcode_98028', 'zi
pcode_98029', 'zipcode_98030', 'zipcode_98031', 'zipcode_98032', 'zipcode_98
033', 'zipcode_98034', 'zipcode_98038', 'zipcode_98039', 'zipcode_98040', 'z
ipcode_98042', 'zipcode_98045', 'zipcode_98052', 'zipcode_98053', 'zipcode_9
8055', 'zipcode_98056', 'zipcode_98058', 'zipcode_98059', 'zipcode_98065',
'zipcode_98070', 'zipcode_98072', 'zipcode_98074', 'zipcode_98075', 'zipcode
_98077', 'zipcode_98092', 'zipcode_98102', 'zipcode_98103', 'zipcode_98105',
'zipcode_98106', 'zipcode_98107', 'zipcode_98108', 'zipcode_98109', 'zipcode
_98112', 'zipcode_98115', 'zipcode_98116', 'zipcode_98117', 'zipcode_98118',
'zipcode_98119', 'zipcode_98122', 'zipcode_98125', 'zipcode_98126', 'zipcode
_98133', 'zipcode_98136', 'zipcode_98144', 'zipcode_98146', 'zipcode_98148',
'zipcode_98155', 'zipcode_98166', 'zipcode_98168', 'zipcode_98177', 'zipcode
_98178', 'zipcode_98188', 'zipcode_98198', 'zipcode_98199', 'view', 'conditi
on', 'grade', 'waterfront', 'bedrooms', 'bathrooms', 'floors']
```

In [23]:

```
#  analysing highest Pearson correlated variables with respect to price

cor = df[num_vars +[y_var]].corr()['price_log'].reset_index()
cor
```

Out[23]:

	index	price_log
0	bedrooms_log	0.347550
1	bathrooms_log	0.532899
2	sqft_living_log	0.674829
3	sqft_lot_log	0.138268
4	floors_log	0.319657
...	...	...
87	waterfront	0.170720
88	bedrooms	0.343360
89	bathrooms	0.551249
90	floors	0.310630
91	price_log	1.000000

92 rows × 2 columns

In [24]:

```
cor[abs(cor['price_log'])>0.6]
```

Out[24]:

	index	price_log
2	sqft_living_log	0.674829
13	sqft_living15_log	0.607167
86	grade	0.703720
91	price_log	1.000000

In [26]:

```
# creating a simple baseline model


simple_var = ['sqft_living_log','sqft_living15_log', 'grade']

simple_df = df[simple_var +[y_var]]
simple_df.head()
```

Out[26]:

	sqft_living_log	sqft_living15_log	grade	price_log
0	-1.125564	-1.035420	7	-1.401997
1	0.709416	-0.326861	7	0.279938
2	-2.131418	1.126525	6	-1.799429
3	0.070561	-0.990188	7	0.499698
4	-0.292847	-0.134305	8	0.178433

In [27]:

```
simple_df.describe()
```

Out[27]:

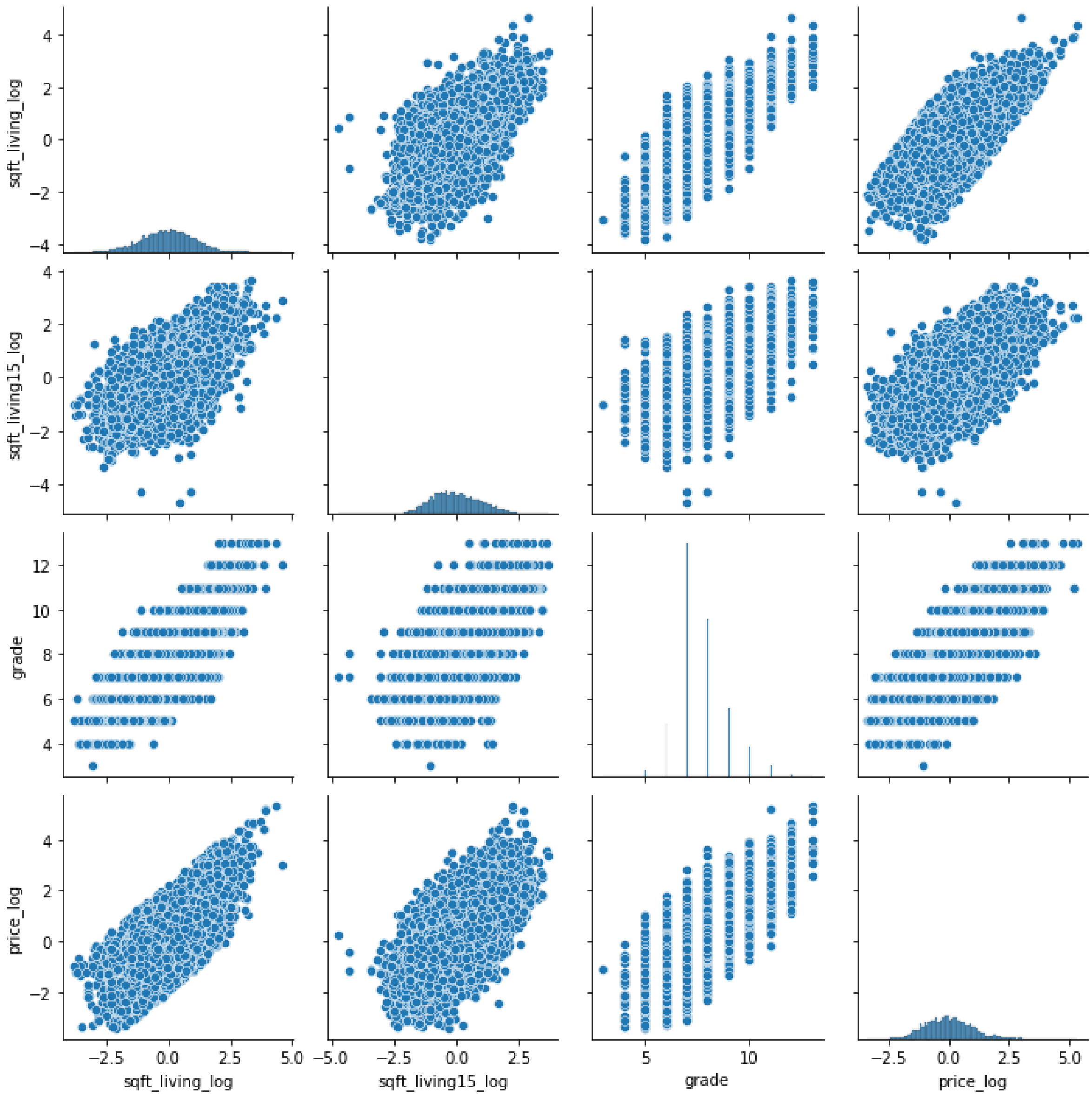
	sqft_living_log	sqft_living15_log	grade	price_log
count	2.159700e+04	2.159700e+04	21597.000000	2.159700e+04
mean	4.115159e-13	6.204906e-13	7.657915	-2.778020e-13
std	1.000000e+00	1.000000e+00	1.173200	1.000000e+00
min	-3.856839e+00	-4.731544e+00	3.000000	-3.387568e+00
25%	-6.726480e-01	-7.114525e-01	7.000000	-6.949025e-01
50%	9.638878e-03	-6.718734e-02	7.000000	-5.926901e-02
75%	6.909955e-01	6.929214e-01	8.000000	6.244259e-01
max	4.628371e+00	3.648206e+00	13.000000	5.333773e+00

In [28]:

```
#  checking out graphs

sns.pairplot(simple_df)
```

Out[28]:

<seaborn.axisgrid.PairGrid at 0x12ffcd8a760>



In [29]:

```
# defining X and y variable
X = simple_df.drop(y_var, axis=1)
y = simple_df[y_var]
```

In [30]:

```
# train test split with 75% - 25%

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, shuffle=True, test_size =0.25, ran
```

In [31]:

```
X_train.head()
```

Out[31]:

	sqft_living_log	sqft_living15_log	grade
6465	1.619186	2.187764	11
10332	0.046381	-2.053465	7
17878	-1.423195	-1.838640	6
18830	0.653716	0.030803	8
12147	0.210659	-0.151317	8

In [32]:

```
y_train.head()
```

Out[32]:

```
6465     1.722635
10332   -0.076226
17878   -1.261012
18830    0.389666
12147   -0.199361
Name: price_log, dtype: float64
```

In [33]:

```
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(16197, 3) (5400, 3) (16197,) (5400,)
```

In [34]:

```
#  fitting X and Y in linear Regression model

from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
linreg.fit(X_train, y_train)
```

Out[34]:

```
LinearRegression()
```

In [35]:

```
# predicting

y_trainpreds = linreg.predict(X_train)
y_testpreds = linreg.predict(X_test)
```

In [36]:

```
#  finding out R Squared and Mean Squared error for test and train to see how our Regressio

from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error

rsq_train = r2_score(y_train, y_trainpreds)
mse_train = mean_squared_error(y_train, y_trainpreds)
rsq_test = r2_score(y_test, y_testpreds)
mse_test = mean_squared_error(y_test, y_testpreds)

print('rsq_train', rsq_train)
print('rsq_test:', rsq_test)
print('mse_train:', mse_train)
print('mse_test:', mse_test)
```

```
rsq_train 0.5513267203029137
rsq_test: 0.5509620181145147
mse_train: 0.4500213787486459
mse_test: 0.44484314066150926
```

In [37]:

```
# calculate adjusted r squared value fro test and train data
# Adj r2 = 1- (1-R2)* (n-1)/ (n-p-1)

def adj_r2(r2, sample, major_X):
    part = (sample -1)/ (sample - major_X -1)
    return 1-((1-r2) * part)
```

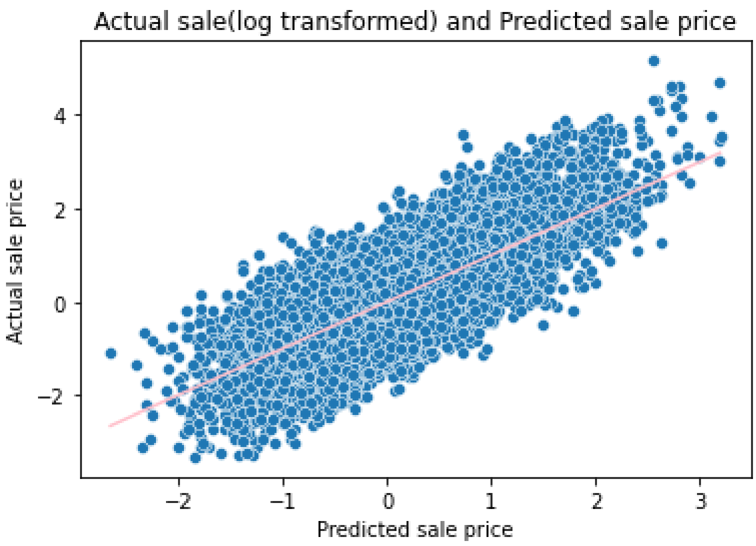
In [38]:

```
adj_rsqtrain = adj_r2(rsq_train, len(X_train), len(X_train.columns))
adj_rsqtest = adj_r2(rsq_test, len(X_test), len(X_test.columns))

print('adj r2 train:', adj_rsqtrain)
print('adj r2 test:', adj_rsqtest)
```

```
adj r2 train: 0.5512435967409368
adj r2 test: 0.5507123676427474
```

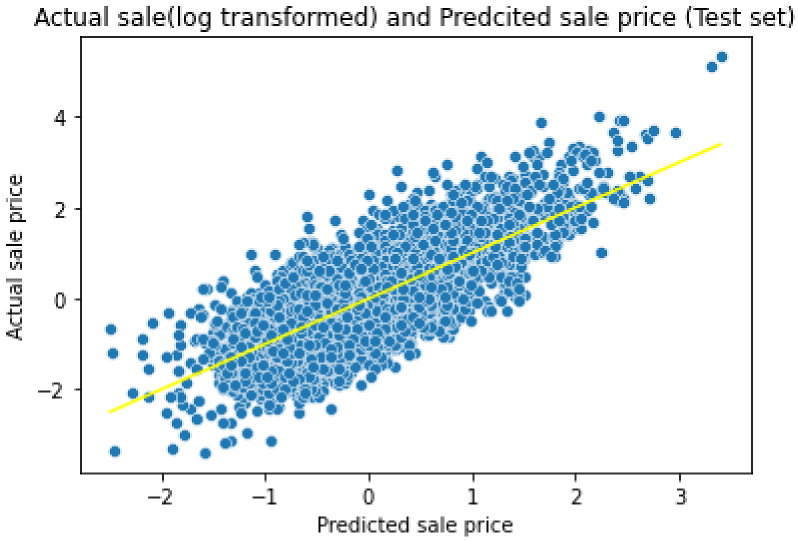
In [39]:

```
# Plotting training Log transformed training set predictions

import matplotlib.pyplot as plt
sns.scatterplot(x = y_trainpreds, y= y_train)
sns.lineplot(x = y_trainpreds, y = y_trainpreds, color ='pink' )
plt.title('Actual sale(log transformed) and Predicted sale price')
plt.xlabel('Predicted sale price')
plt.ylabel('Actual sale price')
plt.show()
```



In [40]:

```
#  plotting Log transformed Test set predictions

sns.scatterplot(x = y_testpreds, y= y_test)
sns.lineplot(x = y_testpreds, y = y_testpreds, color ='yellow' )
plt.title('Actual sale(log transformed) and Predcited sale price (Test set)')
plt.xlabel('Predicted sale price')
plt.ylabel('Actual sale price')
plt.show()
```

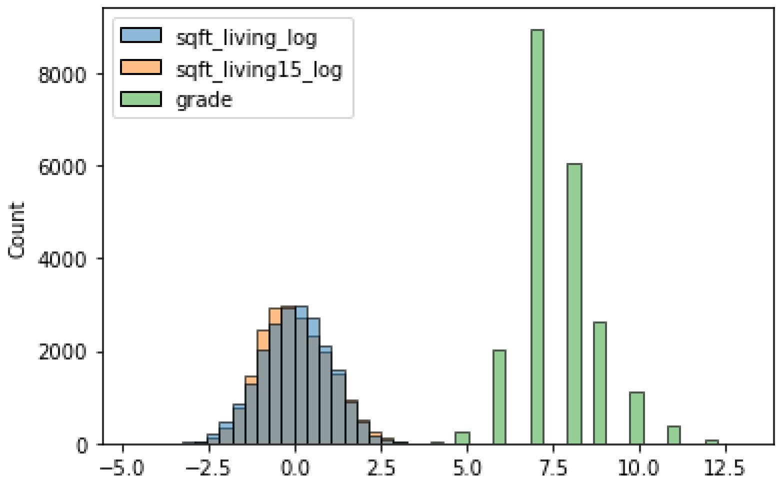


In [41]:

```
sns.histplot(X)
```

Out[41]:

```
<AxesSubplot:ylabel='Count'>
```

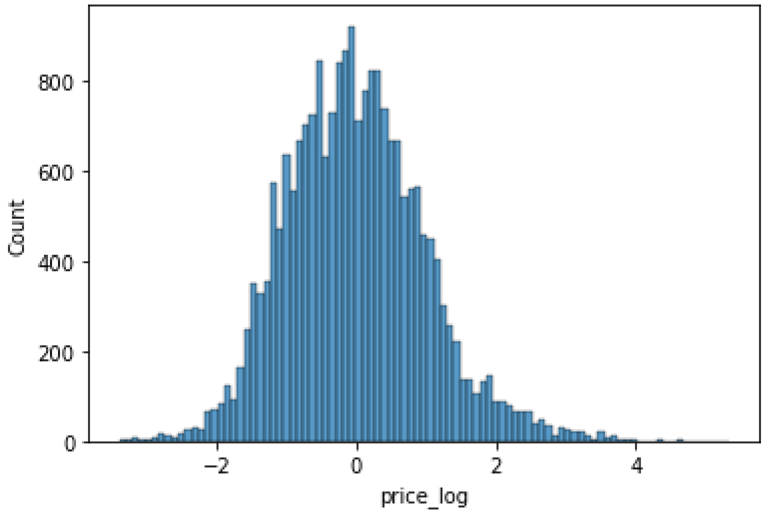


In [42]:

```
sns.histplot(y)
```

Out[42]:

```
<AxesSubplot:xlabel='price_log', ylabel='Count'>
```



In [43]:

```
linreg.fit(X_train, y_train)
y_trainpreds = linreg.predict(X_train)
y_testpreds = linreg.predict(X_test)
```

In [44]:

```
#  cross validate to refine the model

from sklearn.model_selection import cross_validate
cross_val_results = cross_validate(linreg, X, y, cv=10, scoring=["r2", "neg_mean_squared_er
cross_val_results
```

Out[44]:

```
{'fit_time': array([0.00197935, 0.00300026, 0.00200009, 0.00145984, 0.001917
36,
        0.00099492, 0.00100136, 0.00200009, 0.00200009, 0.00299954]),
 'score_time': array([0.00185347, 0.00099993, 0.00197577, 0.00126982, 0.0010
7574,
        0.0010066 , 0.0009973 , 0.00099969, 0.00100017, 0.00099993]),
 'test_r2': array([0.53089672, 0.55598678, 0.52471759, 0.55947389, 0.5049187
2,
        0.55137935, 0.54809031, 0.56337007, 0.54481047, 0.59273351]),
 'train_r2': array([0.55344941, 0.55065194, 0.55408527, 0.55011501, 0.555758
86,
        0.55117593, 0.55159859, 0.54975869, 0.55195954, 0.54447673]),
 'test_neg_mean_squared_error': array([-0.46746125, -0.48014626, -0.4586034
6, -0.45168389, -0.44714316,
        -0.45711415, -0.45639478, -0.46818639, -0.46752997, -0.34508016]),
 'train_neg_mean_squared_error': array([-0.44665062, -0.44525611, -0.4476269
2, -0.44838773, -0.44890232,
        -0.44778297, -0.44788342, -0.44657173, -0.4466672 , -0.46074132])}
```

In [45]:

```
sorted(cross_val_results.keys())
```

Out[45]:

```
['fit_time',
 'score_time',
 'test_neg_mean_squared_error',
 'test_r2',
 'train_neg_mean_squared_error',
 'train_r2']
```

In [46]:

```
#  getting avg values for the keys.

cross_val_avg = {k: np.mean(v) for k,v in cross_val_results.items()}
cross_val_avg
```

Out[46]:

```
{'fit_time': 0.001935291290283203,
 'score_time': 0.0012178421020507812,
 'test_r2': 0.5476377401115963,
 'train_r2': 0.551302997243093,
 'test_neg_mean_squared_error': -0.4499343463369372,
 'train_neg_mean_squared_error': -0.4486470351734878}
```

In [47]:

```
#  summary list output

summary = list()
cross_val_avg['n_features'] = len(X.columns)
cross_val_avg['dataset'] = 'simple'

summary.append(cross_val_avg)
pd.DataFrame(summary)
```

Out[47]:

	fit_time	score_time	test_r2	train_r2	test_neg_mean_squared_error	train_neg_mean_squ
0	0.001935	0.001218	0.547638	0.551303	-0.449934	

In [48]:

```
# Introducing more X variables
num2_var = num_vars[:8]
```

In [49]:

```
df[num2_var].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   bedrooms_log     21597 non-null  float64
 1   bathrooms_log    21597 non-null  float64
 2   sqft_living_log  21597 non-null  float64
 3   sqft_lot_log     21597 non-null  float64
 4   floors_log       21597 non-null  float64
 5   waterfront_log   21597 non-null  float64
 6   view_log         21597 non-null  float64
 7   condition_log    21597 non-null  float64
dtypes: float64(8)
memory usage: 1.3 MB
```

In [50]:

```
df[num2_var].describe()
```

Out[50]:

	bedrooms_log	bathrooms_log	sqft_living_log	sqft_lot_log	floors_log	waterfront_l
count	2.159700e+04	2.159700e+04	2.159700e+04	2.159700e+04	2.159700e+04	2.159700e+
mean	1.322714e-14	-7.403432e-14	4.115159e-13	-9.210487e-14	-8.540480e-14	-1.048729e-
std	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+
min	-3.614426e+00	-2.790771e+00	-3.856839e+00	-3.031500e+00	-9.470866e-01	-8.249784e-
25%	-3.203510e-01	-3.748615e-01	-6.726480e-01	-5.151367e-01	-9.470866e-01	-8.249784e-
50%	-3.203510e-01	2.909753e-01	9.638878e-03	-5.718997e-02	1.182029e-01	-8.249784e-
75%	7.401043e-01	5.863517e-01	6.909955e-01	3.178784e-01	9.886077e-01	-8.249784e-
max	9.849983e+00	4.350745e+00	4.628371e+00	5.906296e+00	2.924302e+00	1.212097e+

In [51]:

```
#  Check for multicollinearity
df[num2_var].corr()[abs(df[num2_var].corr())>0.6]
```

Out[51]:

	bedrooms_log	bathrooms_log	sqft_living_log	sqft_lot_log	floors_log	waterfro
bedrooms_log	1.000000	NaN	0.649916	NaN	NaN	
bathrooms_log	NaN	1.000000	0.765945	NaN	NaN	
sqft_living_log	0.649916	0.765945	1.000000	NaN	NaN	
sqft_lot_log	NaN	NaN	NaN	1.0	NaN	
floors_log	NaN	NaN	NaN	NaN	1.0	
waterfront_log	NaN	NaN	NaN	NaN	NaN	
view_log	NaN	NaN	NaN	NaN	NaN	
condition_log	NaN	NaN	NaN	NaN	NaN	

In [52]:

```
#  dropping two variables from our simple model. adding more variables to our model.
temp_df = df[num2_var].drop(['sqft_living_log'], axis=1)
temp_df
```

Out[52]:

	bedrooms_log	bathrooms_log	sqft_lot_log	floors_log	waterfront_log	view_log	conditi
0	-0.320351	-1.644140	-0.388490	-0.947087	-0.082498	-0.319476	-0.
1	-0.320351	0.290975	-0.113302	0.988608	-0.082498	-0.319476	-0.
2	-1.687516	-1.644140	0.244426	-0.947087	-0.082498	-0.319476	-0.
3	0.740104	1.118575	-0.523969	-0.947087	-0.082498	-0.319476	2.
4	-0.320351	-0.028055	0.008081	-0.947087	-0.082498	-0.319476	-0.
...	...	...	...	...	...	...	
21592	-0.320351	0.586352	-2.171142	2.362005	-0.082498	-0.319476	-0.
21593	0.740104	0.586352	-0.356962	0.988608	-0.082498	-0.319476	-0.
21594	-1.687516	-2.176364	-1.975055	0.988608	-0.082498	-0.319476	-0.
21595	-0.320351	0.586352	-1.343051	0.988608	-0.082498	-0.319476	-0.
21596	-1.687516	-2.176364	-2.226364	0.988608	-0.082498	-0.319476	-0.

21597 rows × 7 columns

In [53]:

```
X = temp_df
X_train, X_test, y_train, y_test = train_test_split(X,y, shuffle=True, test_size =0.25, ran
print(X_train.shape, X_test.shape)
print(y_train.shape, y_test.shape)
```

```
(16197, 7) (5400, 7)
(16197,) (5400,)
```

In [54]:

```
#  Define function

def get_cross_val_avg(X, y, regmodel, setname, scoring=None):
    if scoring is None:
        scoring=["r2", "neg_mean_squared_error"]
    cross_val_results = cross_validate(linreg, X, y, cv=10, scoring= scoring, return_train_
    cross_val_avg = {k: np.mean(v) for k,v in cross_val_results.items()}
    cross_val_avg['n_features'] = len(X.columns)
    cross_val_avg['dataset'] = setname
    return cross_val_avg
```

In [55]:

```
scores = get_cross_val_avg(X, y, linreg, 'with 7 more variables')
summary.append(scores)
pd.DataFrame(summary)
```

Out[55]:

	fit_time	score_time	test_r2	train_r2	test_neg_mean_squared_error	train_neg_mean_squ
0	0.001935	0.001218	0.547638	0.551303	-0.449934	
1	0.004135	0.001094	0.375939	0.381973	-0.619628	

In [56]:

```
#  adding more X variables
num3_var = num_vars[8:]
```

In [60]:

```
temp_num3_df = df[num3_var].drop(['grade','sqft_living15_log'], axis=1)
temp_num3_df
```

Out[60]:

	sqft_above_log	sqft_basement_log	yr_built_log	yr_renovated_log	lat_log	sqft_lot15_
0	-0.753624	-0.785188	-0.537412	-0.188883	-0.351390	-0.395
1	0.672625	1.112322	-0.674329	5.292283	1.160045	-0.024
2	-1.752585	-0.785188	-1.293945	-0.188883	1.281512	0.041
3	-1.026820	1.372094	-0.196342	-0.188883	-0.282087	-0.545
4	0.073385	-0.785188	0.547945	-0.188883	0.410186	-0.046
...	...	...	...	...	...	
21592	-0.145574	-0.785188	1.284041	-0.188883	1.004017	-2.019
21593	0.819022	-0.785188	1.450211	-0.188883	-0.355000	-0.097
21594	-1.094661	-0.785188	1.284041	-0.188883	0.248778	-1.668
21595	-0.040842	-0.785188	1.117456	-0.188883	-0.183210	-2.215
21596	-1.094661	-0.785188	1.250757	-0.188883	0.246616	-2.150

21597 rows × 81 columns

In [61]:

```
temp_num3_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 81 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   sqft_above_log      21597 non-null  float64
 1   sqft_basement_log   21597 non-null  float64
 2   yr_built_log        21597 non-null  float64
 3   yr_renovated_log    21597 non-null  float64
 4   lat_log             21597 non-null  float64
 5   sqft_lot15_log      21597 non-null  float64
 6   zipcode_98002       21597 non-null  float64
 7   zipcode_98003       21597 non-null  float64
 8   zipcode_98004       21597 non-null  float64
 9   zipcode_98005       21597 non-null  float64
 10  zipcode_98006       21597 non-null  float64
 11  zipcode_98007       21597 non-null  float64
 12  zipcode_98008       21597 non-null  float64
 13  zipcode_98010       21597 non-null  float64
 14  zipcode_98011       21597 non-null  float64
 15  zipcode_98014       21597 non-null  float64
 16  zipcode_98019       21597 non-null  float64
 17  zipcode_98022       21597 non-null  float64
 18  zipcode_98023       21597 non-null  float64
 19  zipcode_98024       21597 non-null  float64
 20  zipcode_98027       21597 non-null  float64
 21  zipcode_98028       21597 non-null  float64
 22  zipcode_98029       21597 non-null  float64
 23  zipcode_98030       21597 non-null  float64
 24  zipcode_98031       21597 non-null  float64
 25  zipcode_98032       21597 non-null  float64
 26  zipcode_98033       21597 non-null  float64
 27  zipcode_98034       21597 non-null  float64
 28  zipcode_98038       21597 non-null  float64
 29  zipcode_98039       21597 non-null  float64
 30  zipcode_98040       21597 non-null  float64
 31  zipcode_98042       21597 non-null  float64
 32  zipcode_98045       21597 non-null  float64
 33  zipcode_98052       21597 non-null  float64
 34  zipcode_98053       21597 non-null  float64
 35  zipcode_98055       21597 non-null  float64
 36  zipcode_98056       21597 non-null  float64
 37  zipcode_98058       21597 non-null  float64
 38  zipcode_98059       21597 non-null  float64
 39  zipcode_98065       21597 non-null  float64
 40  zipcode_98070       21597 non-null  float64
 41  zipcode_98072       21597 non-null  float64
 42  zipcode_98074       21597 non-null  float64
 43  zipcode_98075       21597 non-null  float64
 44  zipcode_98077       21597 non-null  float64
 45  zipcode_98092       21597 non-null  float64
 46  zipcode_98102       21597 non-null  float64
 47  zipcode_98103       21597 non-null  float64
 48  zipcode_98105       21597 non-null  float64
 49  zipcode_98106       21597 non-null  float64
 50  zipcode_98107       21597 non-null  float64
```

```
 51  zipcode_98108      21597 non-null  float64
 52  zipcode_98109      21597 non-null  float64
 53  zipcode_98112      21597 non-null  float64
 54  zipcode_98115      21597 non-null  float64
 55  zipcode_98116      21597 non-null  float64
 56  zipcode_98117      21597 non-null  float64
 57  zipcode_98118      21597 non-null  float64
 58  zipcode_98119      21597 non-null  float64
 59  zipcode_98122      21597 non-null  float64
 60  zipcode_98125      21597 non-null  float64
 61  zipcode_98126      21597 non-null  float64
 62  zipcode_98133      21597 non-null  float64
 63  zipcode_98136      21597 non-null  float64
 64  zipcode_98144      21597 non-null  float64
 65  zipcode_98146      21597 non-null  float64
 66  zipcode_98148      21597 non-null  float64
 67  zipcode_98155      21597 non-null  float64
 68  zipcode_98166      21597 non-null  float64
 69  zipcode_98168      21597 non-null  float64
 70  zipcode_98177      21597 non-null  float64
 71  zipcode_98178      21597 non-null  float64
 72  zipcode_98188      21597 non-null  float64
 73  zipcode_98198      21597 non-null  float64
 74  zipcode_98199      21597 non-null  float64
 75  view               21597 non-null  float64
 76  condition          21597 non-null  int64
 77  waterfront         21597 non-null  float64
 78  bedrooms           21597 non-null  int64
 79  bathrooms          21597 non-null  float64
 80  floors             21597 non-null  float64
dtypes: float64(79), int64(2)
memory usage: 13.3 MB
```

In [62]:

```
temp_dff = temp_num3_df
temp_dff
```

Out[62]:

	sqft_above_log	sqft_basement_log	yr_built_log	yr_renovated_log	lat_log	sqft_lot15_
0	-0.753624	-0.785188	-0.537412	-0.188883	-0.351390	-0.395
1	0.672625	1.112322	-0.674329	5.292283	1.160045	-0.024
2	-1.752585	-0.785188	-1.293945	-0.188883	1.281512	0.041
3	-1.026820	1.372094	-0.196342	-0.188883	-0.282087	-0.545
4	0.073385	-0.785188	0.547945	-0.188883	0.410186	-0.046
...	...	...	...	...	...	
21592	-0.145574	-0.785188	1.284041	-0.188883	1.004017	-2.019
21593	0.819022	-0.785188	1.450211	-0.188883	-0.355000	-0.097
21594	-1.094661	-0.785188	1.284041	-0.188883	0.248778	-1.668
21595	-0.040842	-0.785188	1.117456	-0.188883	-0.183210	-2.215
21596	-1.094661	-0.785188	1.250757	-0.188883	0.246616	-2.150

21597 rows × 81 columns

In [63]:

```
type(temp_dff)
```

Out[63]:

```
pandas.core.frame.DataFrame
```

In [64]:

```
temp_df.join(temp_dff)
```

Out[64]:

	bedrooms_log	bathrooms_log	sqft_lot_log	floors_log	waterfront_log	view_log	conditi
0	-0.320351	-1.644140	-0.388490	-0.947087	-0.082498	-0.319476	-0.
1	-0.320351	0.290975	-0.113302	0.988608	-0.082498	-0.319476	-0.
2	-1.687516	-1.644140	0.244426	-0.947087	-0.082498	-0.319476	-0.
3	0.740104	1.118575	-0.523969	-0.947087	-0.082498	-0.319476	2.
4	-0.320351	-0.028055	0.008081	-0.947087	-0.082498	-0.319476	-0.
...	...	...	...	...	...	...	
21592	-0.320351	0.586352	-2.171142	2.362005	-0.082498	-0.319476	-0.
21593	0.740104	0.586352	-0.356962	0.988608	-0.082498	-0.319476	-0.
21594	-1.687516	-2.176364	-1.975055	0.988608	-0.082498	-0.319476	-0.
21595	-0.320351	0.586352	-1.343051	0.988608	-0.082498	-0.319476	-0.
21596	-1.687516	-2.176364	-2.226364	0.988608	-0.082498	-0.319476	-0.

21597 rows × 88 columns

In [65]:

```
X = temp_df.join(temp_dff)
X_train, X_test, y_train, y_test = train_test_split(X,y, shuffle=True, test_size =0.25, ran
print(X_train.shape, X_test.shape)
print(y_train.shape, y_test.shape)
```

```
(16197, 88) (5400, 88)
(16197,) (5400,)
```

In [66]:

```
type(X)
```

Out[66]:

```
pandas.core.frame.DataFrame
```

In [67]:

```
#  run model again
scores = get_cross_val_avg(X, y, linreg, 'with 88 more variables')
summary.append(scores)
pd.DataFrame(summary)
```

Out[67]:

	fit_time	score_time	test_r2	train_r2	test_neg_mean_squared_error	train_neg_mean_squ
0	0.001935	0.001218	0.547638	0.551303	-0.449934	
1	0.004135	0.001094	0.375939	0.381973	-0.619628	
2	0.043893	0.002366	0.860045	0.863197	-0.138990	

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: