



Learn Git and GitHub without any code!


Using the Hello World guide, you'll start a branch, write comments, and open a pull request.

[Read the guide](#)

 [JosephDenney](#) / [KingCountyRealEstate](#)

 Code

 Issues


 Pull requests

 Actions

 Projects

 Wiki

 Security

 master ▾



[KingCountyRealEstate](#) / [Mod2HousingReg.md](#)



JosephDenney monday

 History

 1 contributor

Raw

Blame



2986 lines (2316 sloc) 92 KB

Housing Analysis in King County, Washington

Linear Regression Analysis

Business Challenge

A local realtor wants to know which features are affecting home prices the most in King County. The realtor does work in all of the areas of King County. A regression will help the client better understand which homes to show and when to highlight the important features to prospective home buyers.

Important features will be qualified as those features which most affect the price of a home.

```
import warnings

warnings.filterwarnings('ignore')

import pandas as pd
import numpy as np
import csv

import scipy.stats as scs
import matplotlib.patches as patches
import statsmodels.api as sm
import statsmodels.formula.api as sms
import scipy.stats as stats
from pltfunctions import hist_kde_plots as hk
from statsFunctions import adjusted_r_squared as clc
from statsFunctions import check_model as cm
from statsFunctions import check_vif_feature_space
from statsFunctions import build_sm_ols
from statsFunctions import check_residuals_normal
from statsFunctions import check_residuals_homoskedasticity
from statsmodels.formula.api import ols
from haversine import haversine

from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import f_regression

import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv(r'data\CleanHousing.csv', index_col=0)
pd.options.display.float_format = '{:.2f}'.format

# checking for OLS assumptions:
# 1) the regression model is 'linear in parameters'
# 2) There is a random sampling of observations
# 3) The conditional mean should be zero, ie the expected value of the mean of the e
# 4) There is no multi-collinearity (no features can be derived from other features'
# 5) There is homoscedasticity and no autocorrelation
# 6) Error terms should be normally distributed
```

Data description per following image

* The following data has been removed from the dataset, however: id, date, sqft_above, sqft_basement, yr_renovated, and sqft_living15

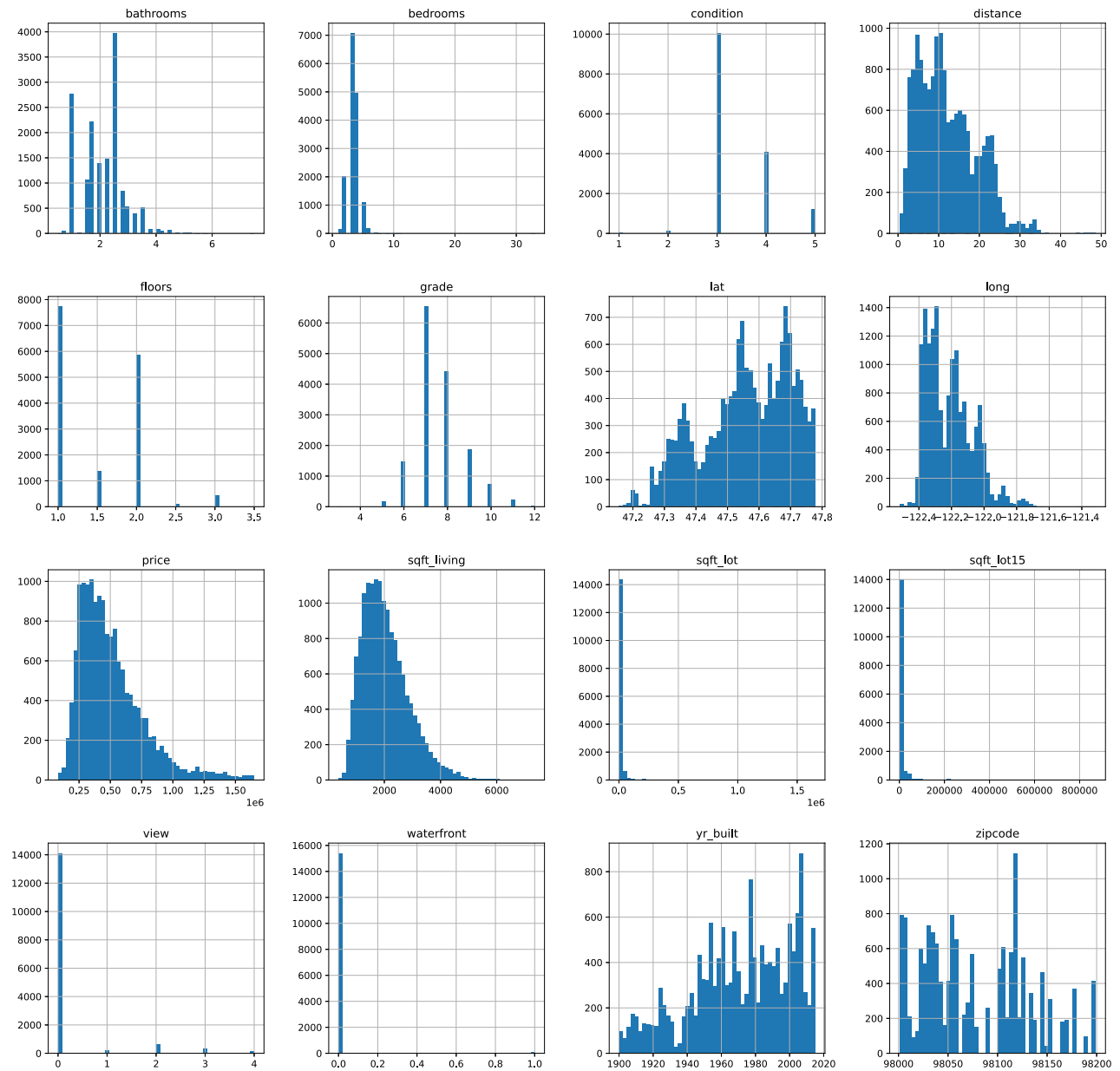
* Please refer to Mod2ProjectEDA.ipynb for an explanation of exclusions

* There is an added feature that indicates the distance of the home from downtown Seattle in miles

Variable	Description
Id	Unique ID for each home sold
Date	Date of the home sale
Price	Price of each home sold
Bedrooms	Number of bedrooms
Bathrooms	Number of bathrooms, where .5 accounts for a room with a toilet but no shower
Sqft_living	Square footage of the apartments interior living space
Sqft_lot	Square footage of the land space
Floors	Number of floors
Waterfront	A dummy variable for whether the apartment was overlooking the waterfront or not
View	An index from 0 to 4 of how good the view of the property was
Condition	An index from 1 to 5 on the condition of the apartment,
Grade	An index from 1 to 13, where 1-3 falls short of building construction and design, 7 has an average level of construction and design, and 11-13 have a high quality level of construction and design
Sqft_above	The square footage of the interior housing space that is above ground level
Sqft_basement	The square footage of the interior housing space that is below ground level
Yr_built	The year the house was initially built
Yr_renovated	The year of the house's last renovation
Zipcode	What zipcode area the house is in
Lat	Latitude
Long	Longitude
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

```
df.hist(figsize=(20,20),bins=50)
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x00000214B36F4C10>,  
      <matplotlib.axes._subplots.AxesSubplot object at 0x00000214B3962160>,  
      <matplotlib.axes._subplots.AxesSubplot object at 0x00000214B3793550>,  
      <matplotlib.axes._subplots.AxesSubplot object at 0x00000214B3FEE9D0>],  
      [<matplotlib.axes._subplots.AxesSubplot object at 0x00000214B4016E20>,  
      <matplotlib.axes._subplots.AxesSubplot object at 0x00000214B404D1F0>,  
      <matplotlib.axes._subplots.AxesSubplot object at 0x00000214B404D2E0>,  
      <matplotlib.axes._subplots.AxesSubplot object at 0x00000214B4074790>],  
      [<matplotlib.axes._subplots.AxesSubplot object at 0x00000214B40C8FA0>,  
      <matplotlib.axes._subplots.AxesSubplot object at 0x00000214B40FE430>,  
      <matplotlib.axes._subplots.AxesSubplot object at 0x00000214B41288B0>,  
      <matplotlib.axes._subplots.AxesSubplot object at 0x00000214B4154D00>],  
      [<matplotlib.axes._subplots.AxesSubplot object at 0x00000214B418C190>,  
      <matplotlib.axes._subplots.AxesSubplot object at 0x00000214B41BA5E0>,  
      <matplotlib.axes._subplots.AxesSubplot object at 0x00000214B41E4A30>,  
      <matplotlib.axes._subplots.AxesSubplot object at 0x00000214B4212EB0>]],  
      dtype=object)
```



`df.describe()` # a look at the data, data has been cleaned and imported

```
<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }
```

```
.dataframe tbody tr th {
  vertical-align: top;
}
```

```
.dataframe thead th {
  text-align: right;
}
```

```
</style>
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot	flo

count	15480.00	15480.00	15480.00	15480.00	15480.00	15480.00
price						
mean	508430.96	3.36	2.09	2037.26	15073.20	1.49
std	261518.09	0.93	0.73	834.96	41238.50	0.54
min	82000.00	1.00	0.50	370.00	520.00	1.00
25%	320000.00	3.00	1.50	1420.00	5012.75	1.00
50%	449000.00	3.00	2.25	1900.00	7560.00	1.50
75%	627500.00	4.00	2.50	2510.00	10500.00	2.00
max	1650000.00	33.00	7.50	7350.00	1651359.00	3.50

Check for Multicollinearity in the data

```
features = ['price','bedrooms','bathrooms','sqft_living','sqft_lot','sqft_lot15','floors']
```

```
corr1 = df[features].corr()  
corr1  
# correlation table with target variable
```

```
<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }
```

```
.dataframe tbody tr th {  
    vertical-align: top;  
}
```

```
.dataframe thead th {  
    text-align: right;  
}
```

```
</style>
```

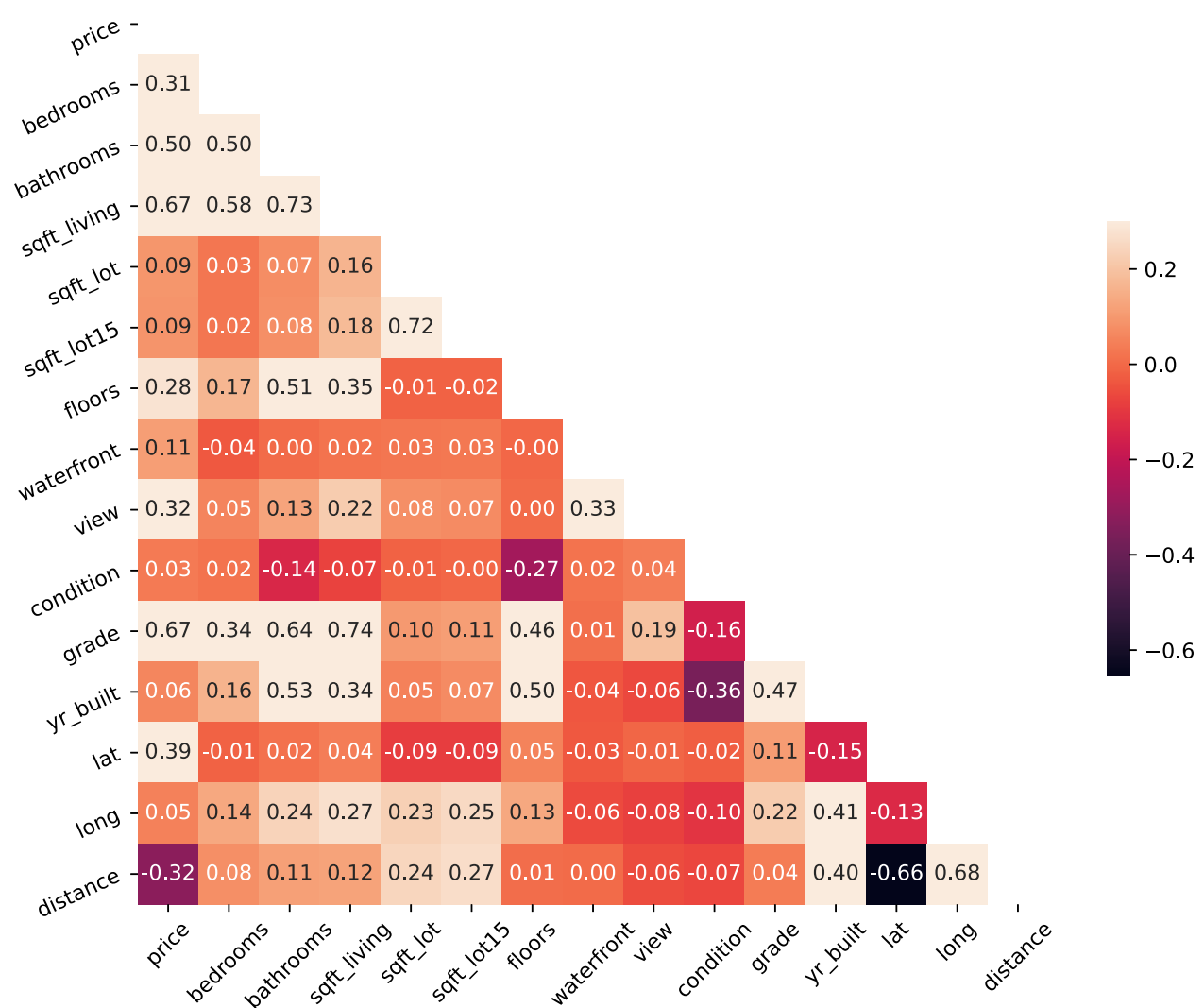
	price	bedrooms	bathrooms	sqft_living	sqft_lot	sqft_lot15
price	1.00	0.31	0.50	0.67	0.09	0.09
bedrooms	0.31	1.00	0.50	0.58	0.03	0.02
bathrooms	0.50	0.50	1.00	0.73	0.07	0.08
saft living	0.67	0.58	0.73	1.00	0.16	0.18

	price	bedrooms	bathrooms	sqft_living	sqft_lot	sqft_lot15
sqft_lot	0.09	0.03	0.07	0.16	1.00	0.72
sqft_lot15	0.09	0.02	0.08	0.18	0.72	1.00
floors	0.28	0.17	0.51	0.35	-0.01	-0.02
waterfront	0.11	-0.04	0.00	0.02	0.03	0.03
view	0.32	0.05	0.13	0.22	0.08	0.07
condition	0.03	0.02	-0.14	-0.07	-0.01	-0.00
grade	0.67	0.34	0.64	0.74	0.10	0.11
yr_built	0.06	0.16	0.53	0.34	0.05	0.07
lat	0.39	-0.01	0.02	0.04	-0.09	-0.09
long	0.05	0.14	0.24	0.27	0.23	0.25
distance	-0.32	0.08	0.11	0.12	0.24	0.27

```
plt.figure(figsize=(10,8))
# cmap = sns.diverging_palette(300,10,as_cmap=True)
mask = np.triu(np.ones_like(corr1, dtype=bool))
chart = sns.heatmap(corr1, fmt='0.2f', mask=mask, annot=True, vmax=0.3, square=True,
chart.set_xticklabels(chart.get_xticklabels(),rotation=45)
chart.set_yticklabels(chart.get_yticklabels(),rotation=25)
plt.title("Heat Map with Target",fontsize=20)
```

```
Text(0.5, 1.0, 'Heat Map with Target')
```

Heat Map with Target



```
features2 = ['bedrooms','bathrooms','sqft_living','sqft_lot','floors','waterfront','  
corr2 = df[features2].corr()  
corr2  
# correlation table with target variable
```

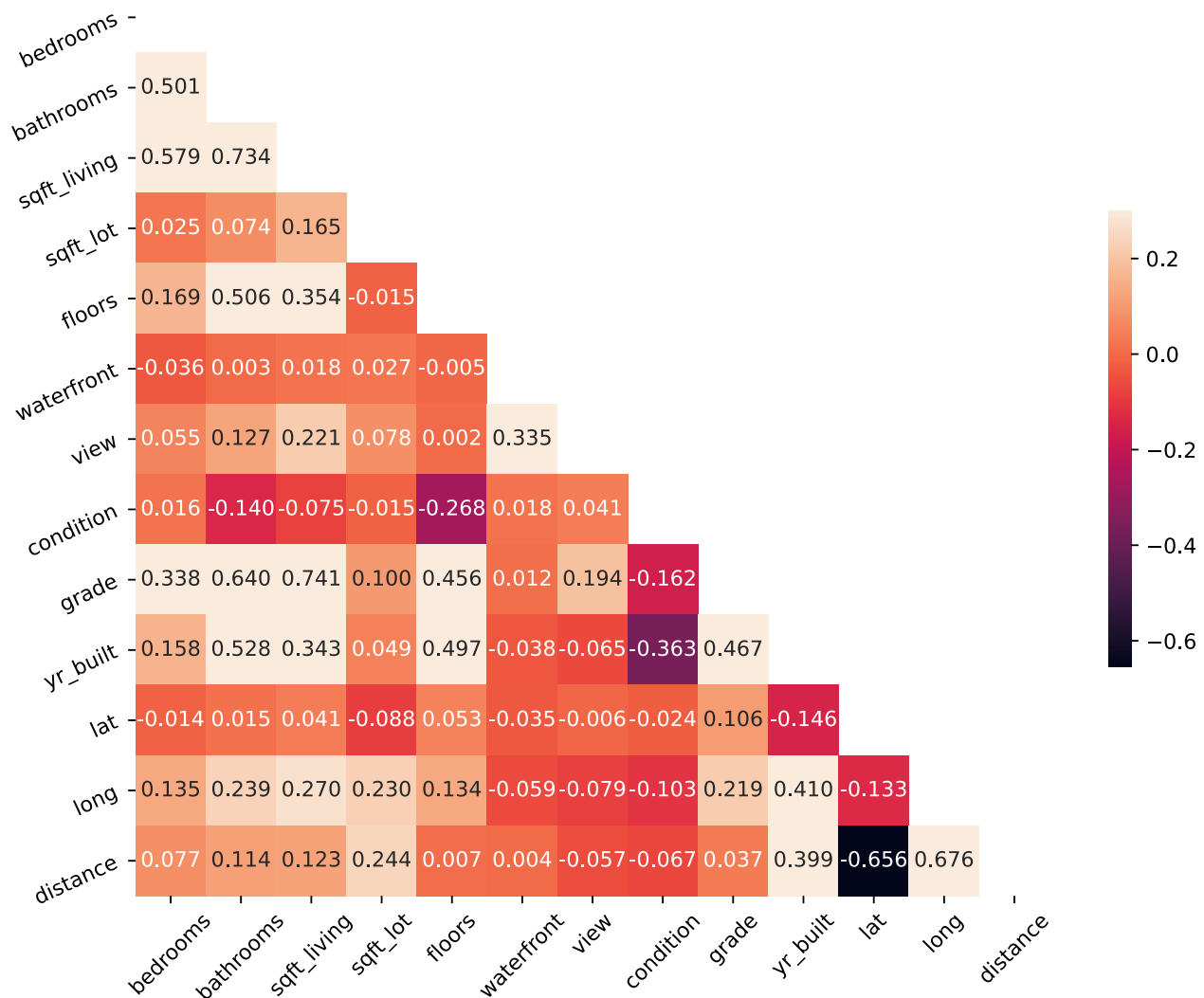
```
<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }  
  
 .dataframe tbody tr th {  
     vertical-align: top;  
 }  
  
 .dataframe thead th {  
     text-align: right;  
 }  
  
</style>
```


	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront
bedrooms	1.00	0.50	0.58	0.03	0.17	-0.04
bathrooms	0.50	1.00	0.73	0.07	0.51	0.00
sqft_living	0.58	0.73	1.00	0.16	0.35	0.02
sqft_lot	0.03	0.07	0.16	1.00	-0.01	0.03
floors	0.17	0.51	0.35	-0.01	1.00	-0.00
waterfront	-0.04	0.00	0.02	0.03	-0.00	1.00
view	0.05	0.13	0.22	0.08	0.00	0.33
condition	0.02	-0.14	-0.07	-0.01	-0.27	0.02
grade	0.34	0.64	0.74	0.10	0.46	0.01
yr_built	0.16	0.53	0.34	0.05	0.50	-0.04
lat	-0.01	0.02	0.04	-0.09	0.05	-0.03
long	0.14	0.24	0.27	0.23	0.13	-0.06
distance	0.08	0.11	0.12	0.24	0.01	0.00

```
plt.figure(figsize=(10,8))
# cmap = sns.diverging_palette(300,10,as_cmap=True)
mask = np.triu(np.ones_like(corr2, dtype=bool))
chart = sns.heatmap(corr2,fmt='0.3f', mask=mask, annot=True, vmax=0.3, square=True,
chart.set_xticklabels(chart.get_xticklabels(),rotation=45)
chart.set_yticklabels(chart.get_xticklabels(),rotation=25)
plt.title("Heat Map without Target",fontsize=20)
```

```
Text(0.5, 1.0, 'Heat Map without Target')
```

Heat Map without Target



Sqft_living and bathrooms are highly correlated, as are grade and sqft_living.

Keeping in mind those two sets of highly correlated features, we can continue with creating our model and answering our questions!

This next section will create a linear regression to model the data using a test/train split of 20/80.

```
# remove lat and long from here - will clearly be redundant since we feature engineer
X = df[['bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot15', 'floors', 'waterfront', 'view', 'condition', 'grade', 'yr_built']]
y = df['price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .20, random_state = 42)
```

```
reg = LinearRegression(fit_intercept=False) # create linear regression
reg.fit(X_train,y_train) # fit the model
```

```
LinearRegression(fit_intercept=False)
```

```
# evaluate the model on the testing data
a1 = reg.score(X_test, y_test)
X_train.shape
a1
```

```
0.6851223744258852
```

```
r2train = reg.score(X_train,y_train)
r2test = reg.score(X_test, y_test)
num_obtrain = X_train.shape[0]
num_obtest = X_test.shape[0]
ptrain = X_train.shape[1]
ptest = X_test.shape[1]
```

```
# compare the r2 scores of the model on the training and test data
r2tra, r2tes = reg.score(X_train,y_train),reg.score(X_test,y_test)
print(r2tra)
print(r2tes)
```

```
#a adjusted r^2 for the data is telling us that this model's features explain ~66% c
```

```
0.6894476298240686
0.6851223744258852
```

```
# calc adjusted r squared and VIF score for training data
clc(r2tra,num_obtrain,ptrain) # low VIF score is good but we can probably reduce it
```

```
Adjusted R^2 is: 0.6891966378494658
VIF score is: 3.217468411798136
```

```
(3.217468411798136, 0.6891966378494658)
```

```
# calc adjusted r squared and VIF score for test data  
clc(r2tes,num_obtest,ptest) # same as above test data is giving us similar indicator
```

```
Adjusted R^2 is: 0.6841017014094375  
VIF score is: 3.16557577062517
```

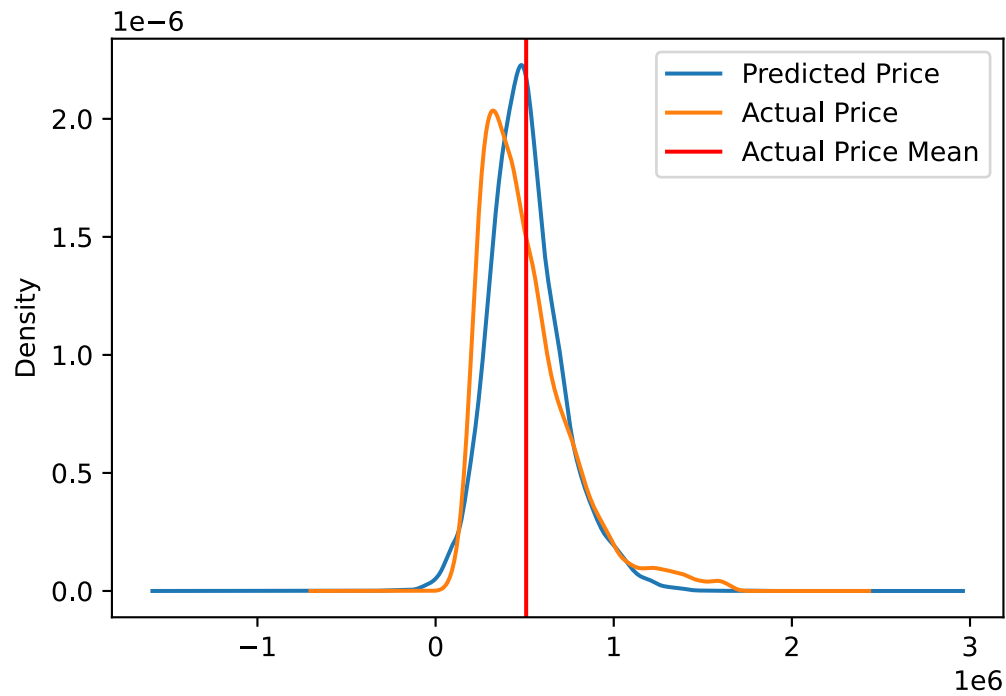
```
(3.16557577062517, 0.6841017014094375)
```

```
df['PredictedPrice'] = reg.predict(X) # create a column with a new predicted price t
```

```
x_mean = df['price'].mean()  
df.head()  
df['PredictedPrice'].plot.kde(label='Predicted Price')  
df['price'].plot.kde(label='Actual Price')  
plt.axvline(x_mean, 0, 5, label='Actual Price Mean', c='r')  
plt.legend()
```

```
# graph below shows that our model is missing some information and is anticipating a  
# the distribution of home prices leans to and peaks to the left of the mean home pr  
# peaks right at the actual price mean
```

```
<matplotlib.legend.Legend at 0x214b6dbc5b0>
```



```
# quick feature engineering of price per square foot - if we use this in a regressio
df['$_sqft'] = df['price']/df['sqft_living']
df.head()
```

```
<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }
```

```
.dataframe tbody tr th {
    vertical-align: top;
}
```

```
.dataframe thead th {
    text-align: right;
}
```

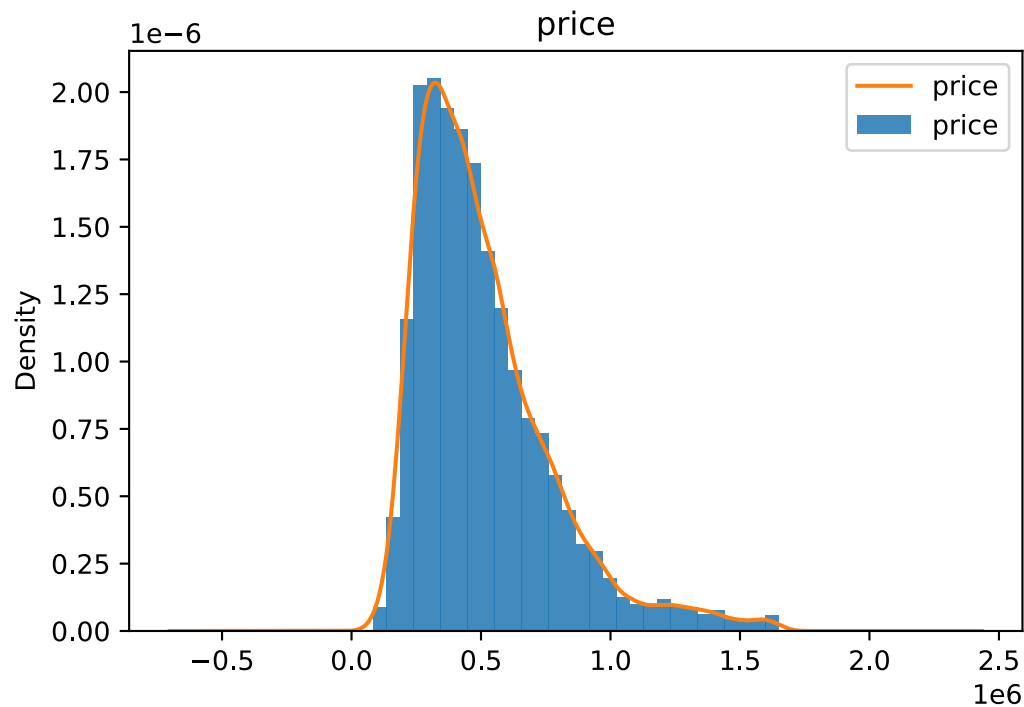
```
</style>
```

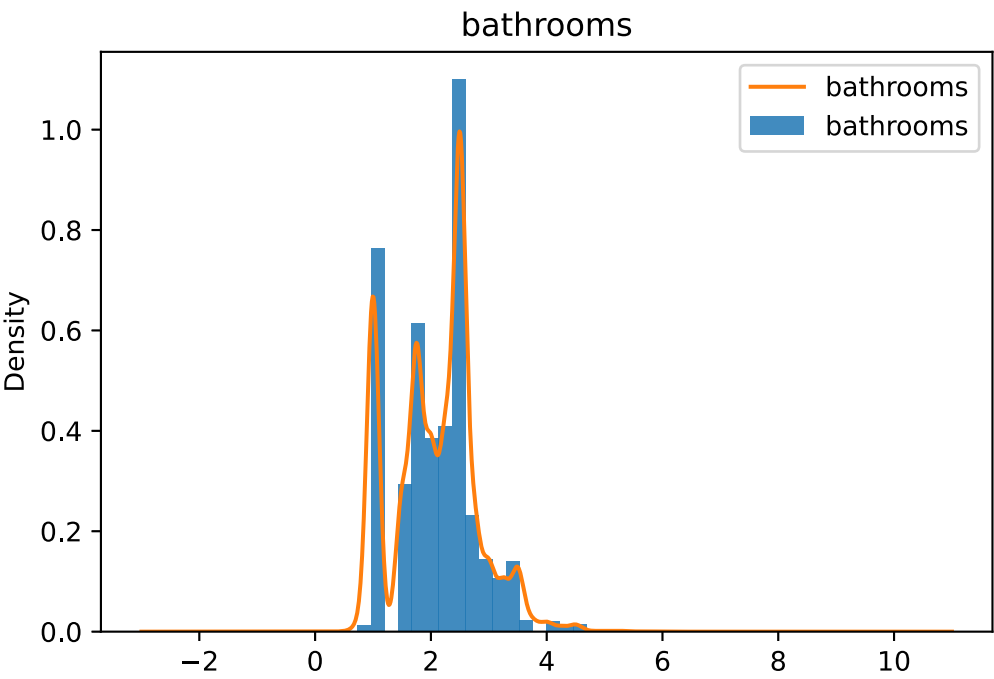
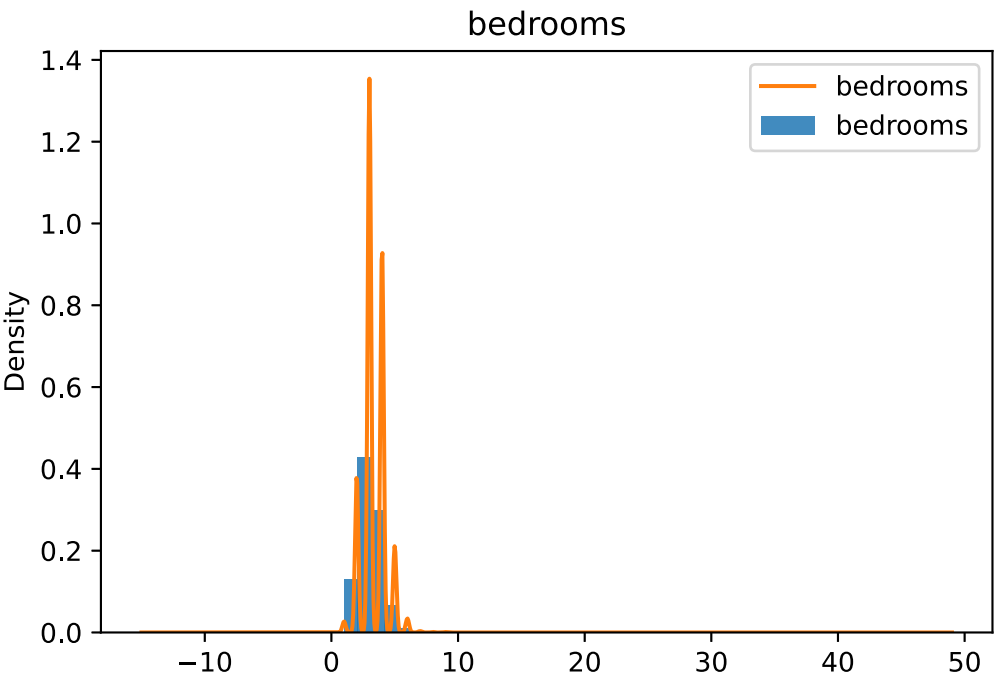
	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	wa
1	538000.00	3	2.25	2570	7242	2.00	0.0
3	604000.00	4	3.00	1960	5000	1.00	0.0
4	510000.00	3	2.00	1680	8080	1.00	0.0

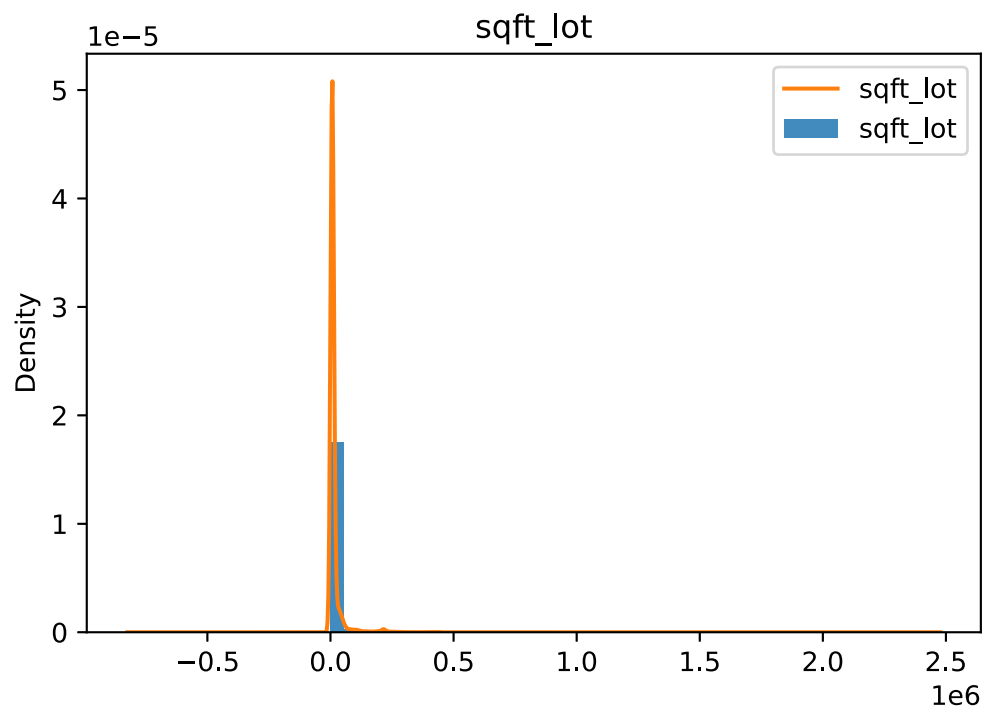
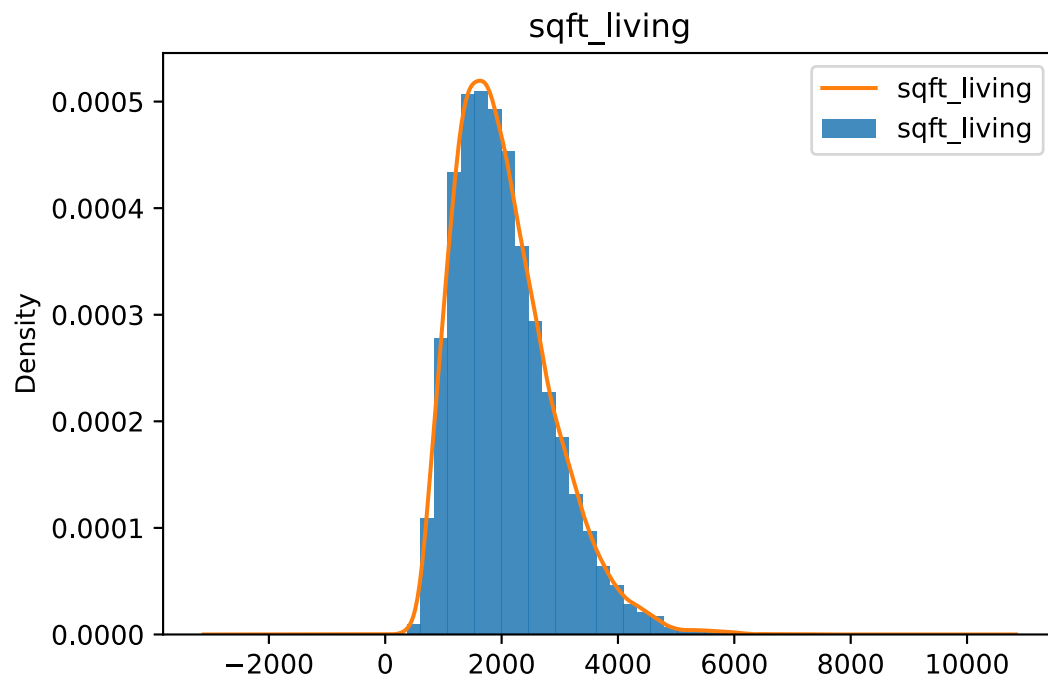
5	1230000.00	4	4.50	5420	9150	1.00	0.0
6	257500.00	3	2.25	1715	6819	2.00	0.0

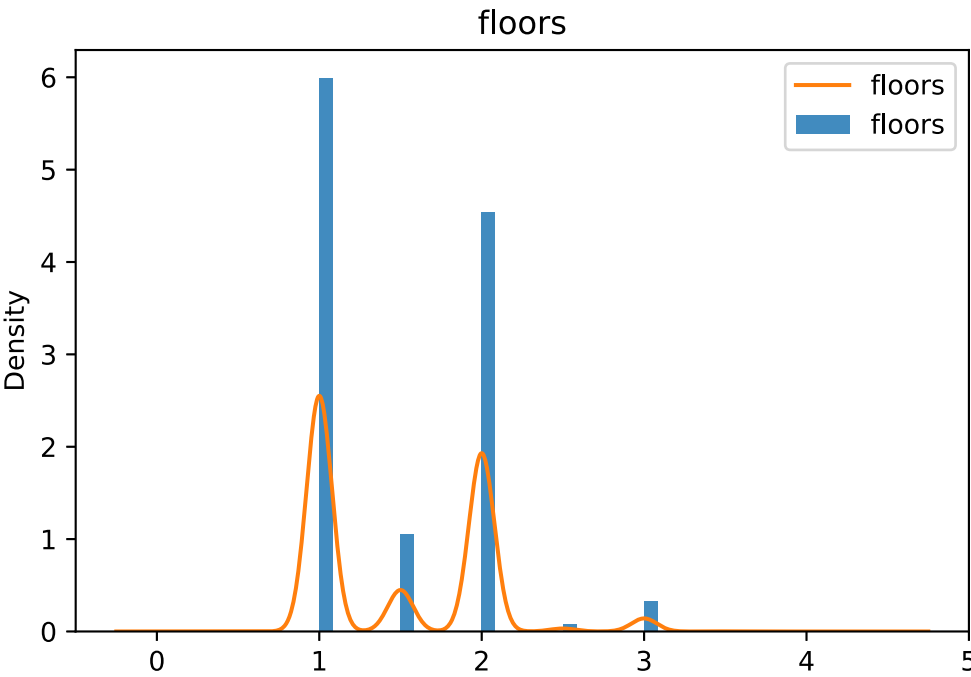
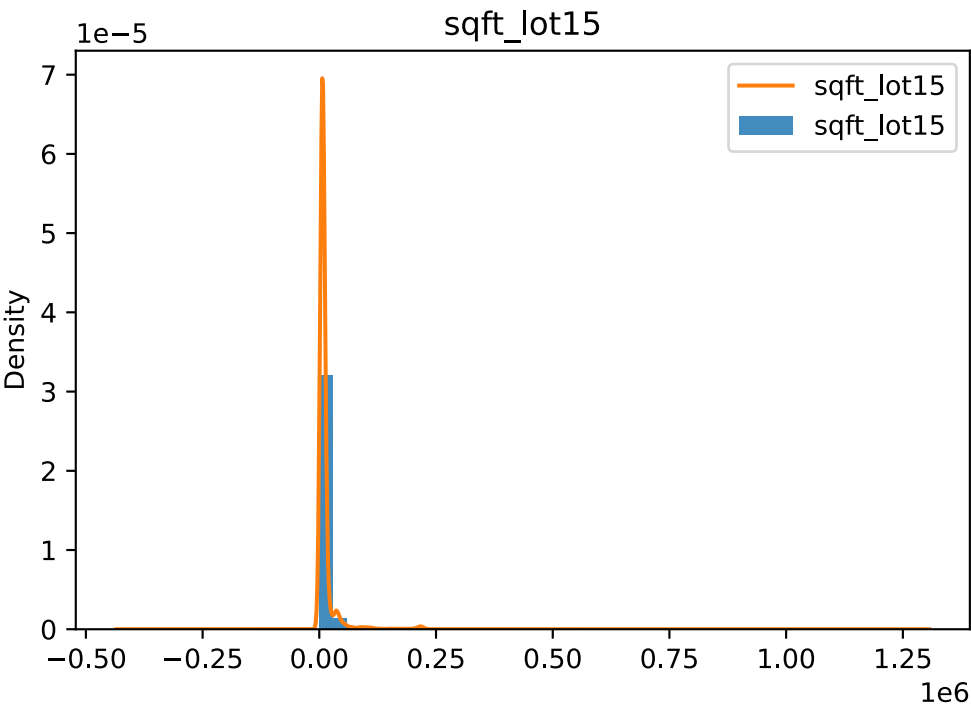
what can we remove from the model?

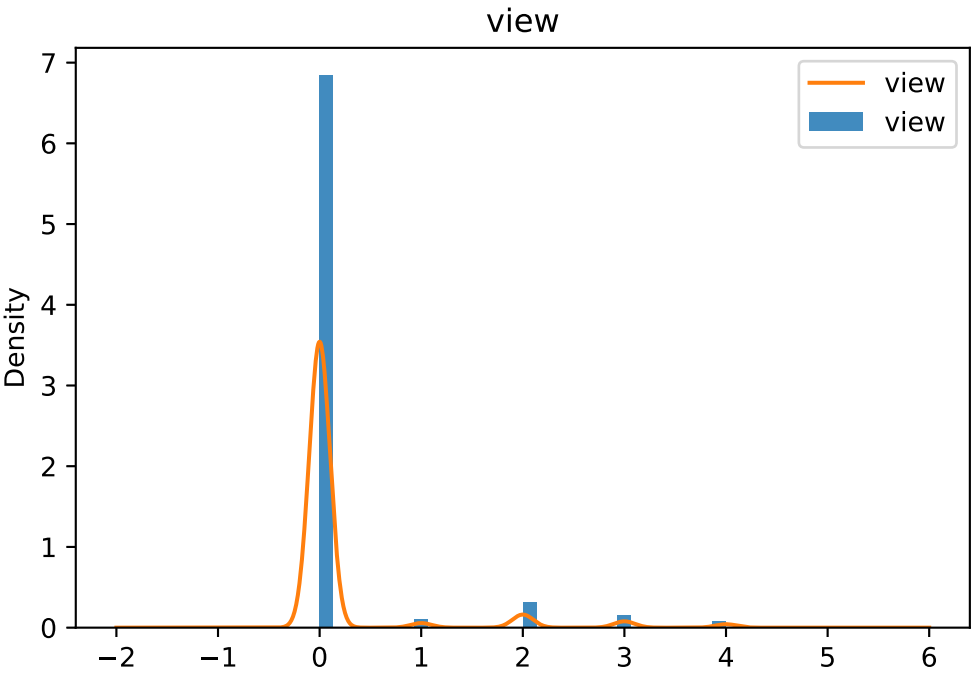
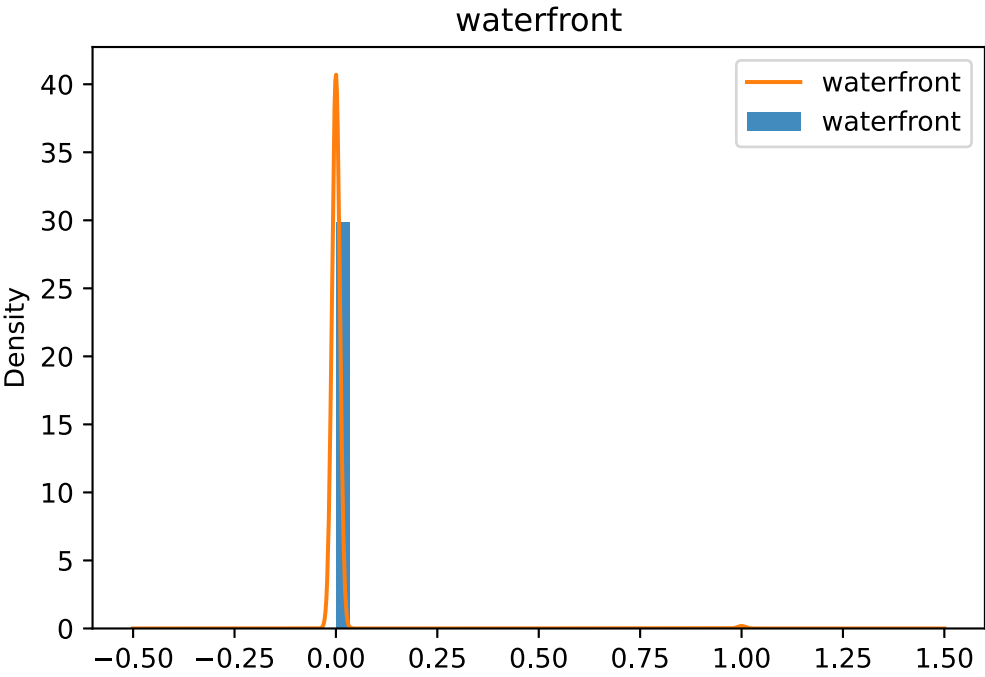
```
for column in ['price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'sqft_lot15',
df[column].plot.hist(density = True, bins = 30, alpha=.85)
df[column].plot.kde()
plt.title(column)
plt.legend()
plt.show()
```

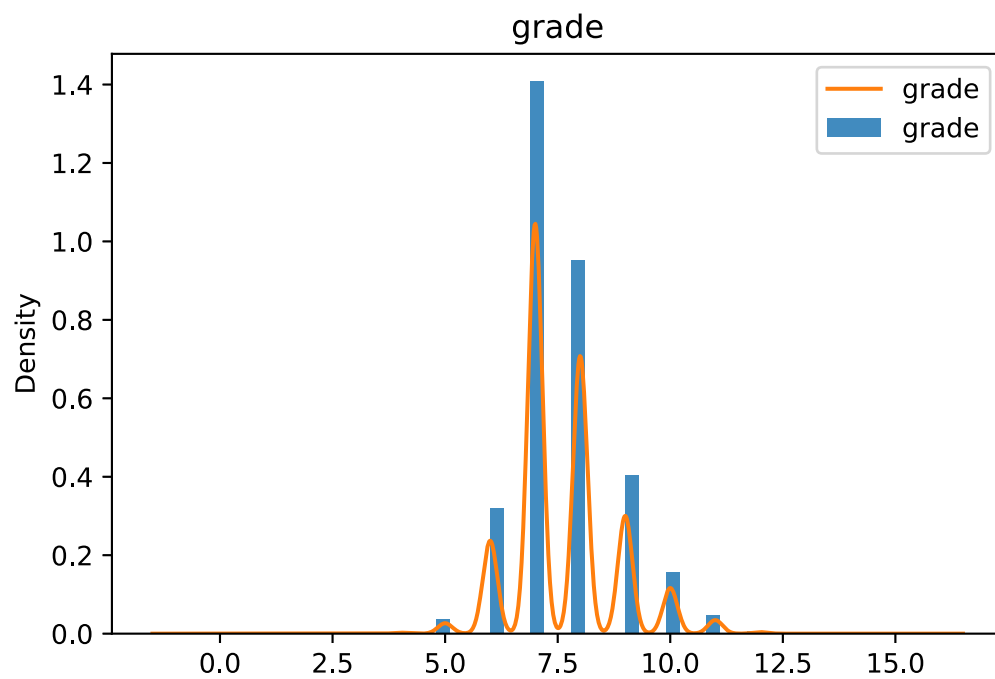
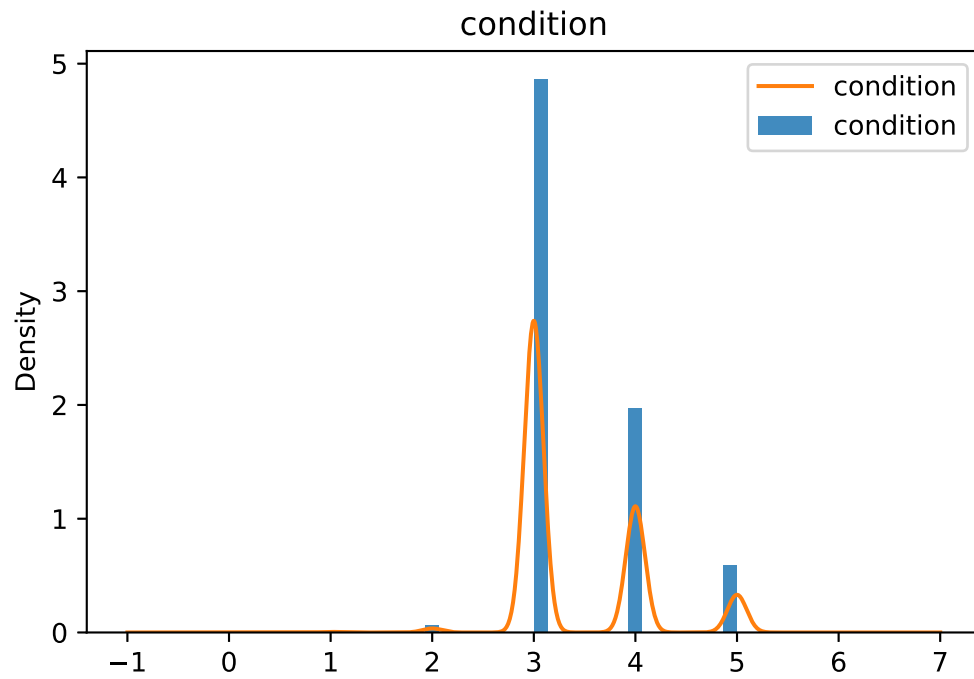


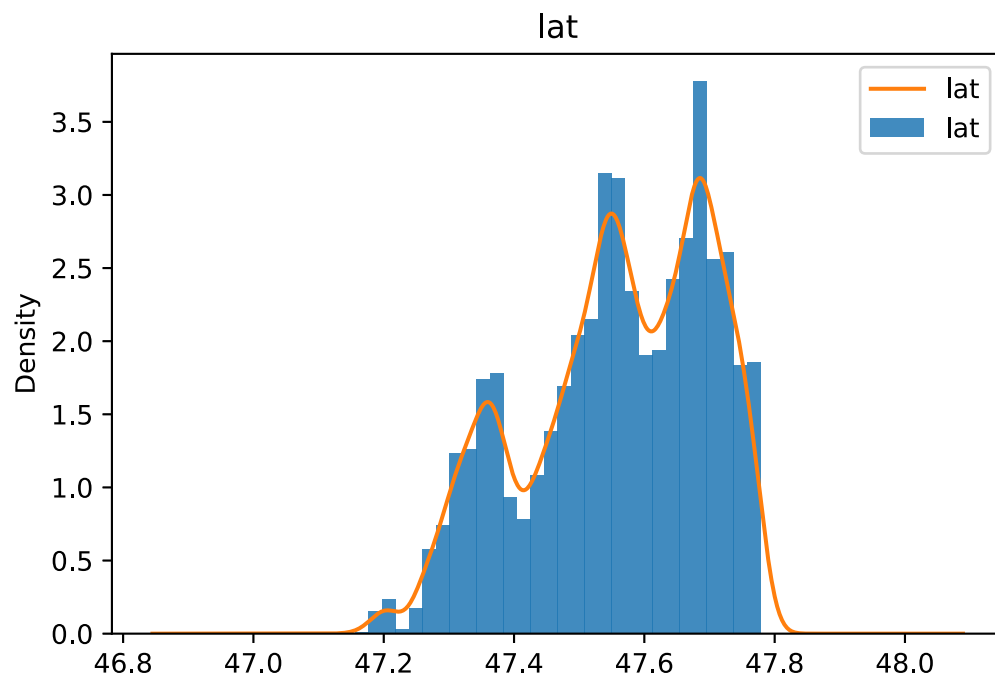
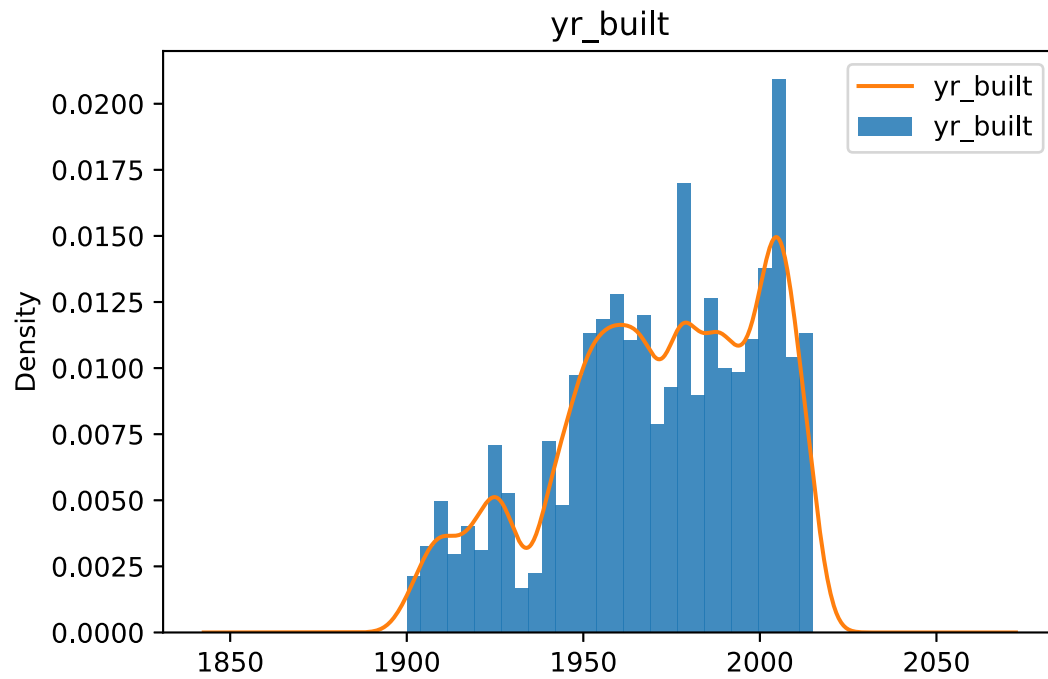


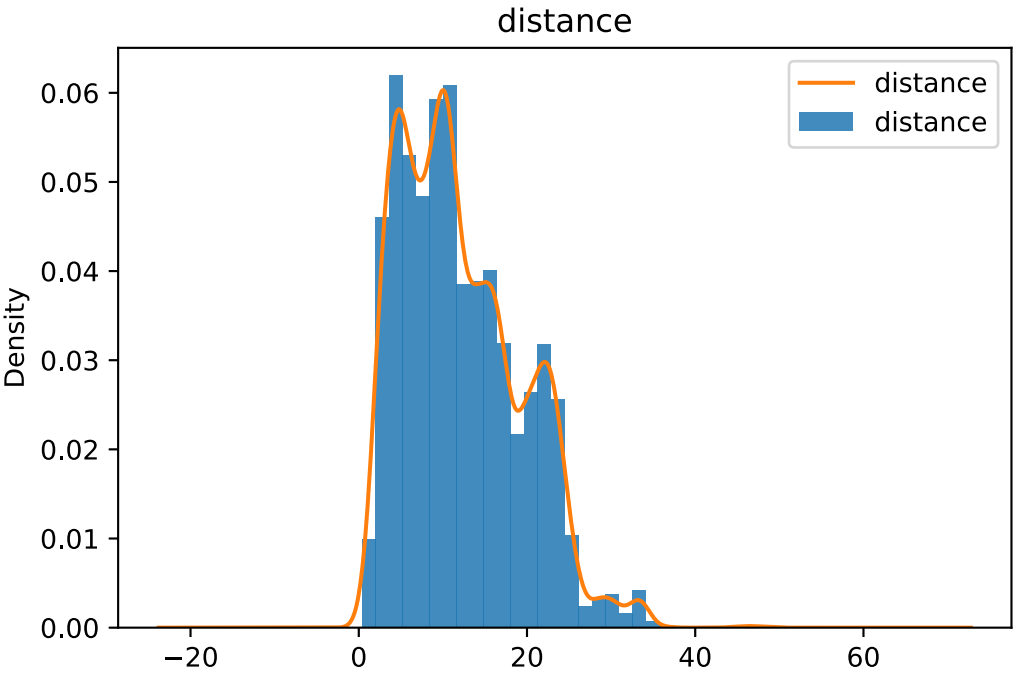
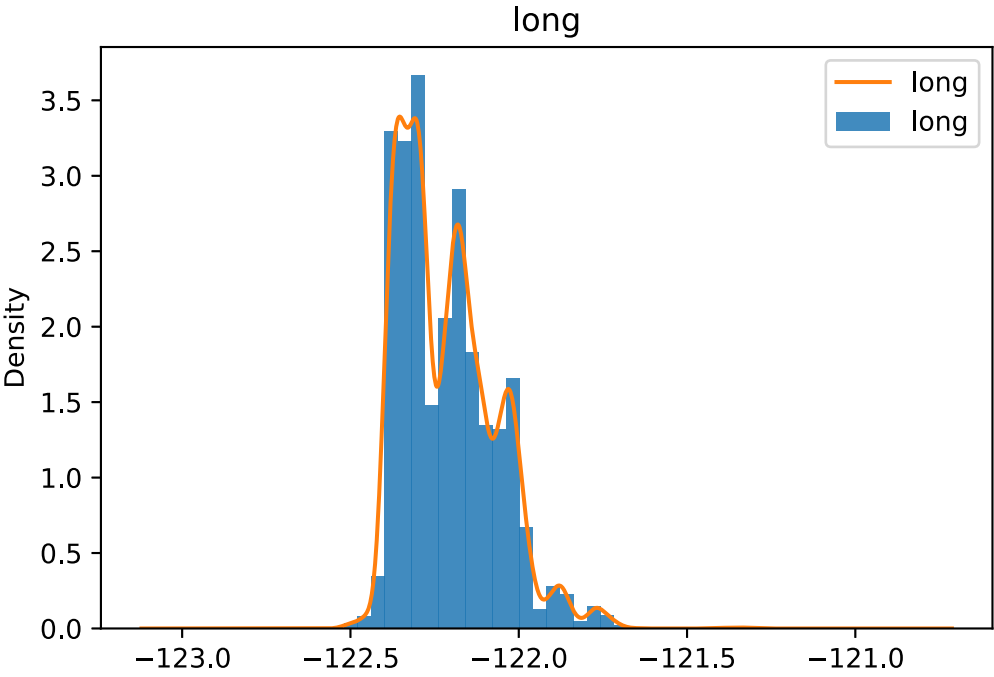


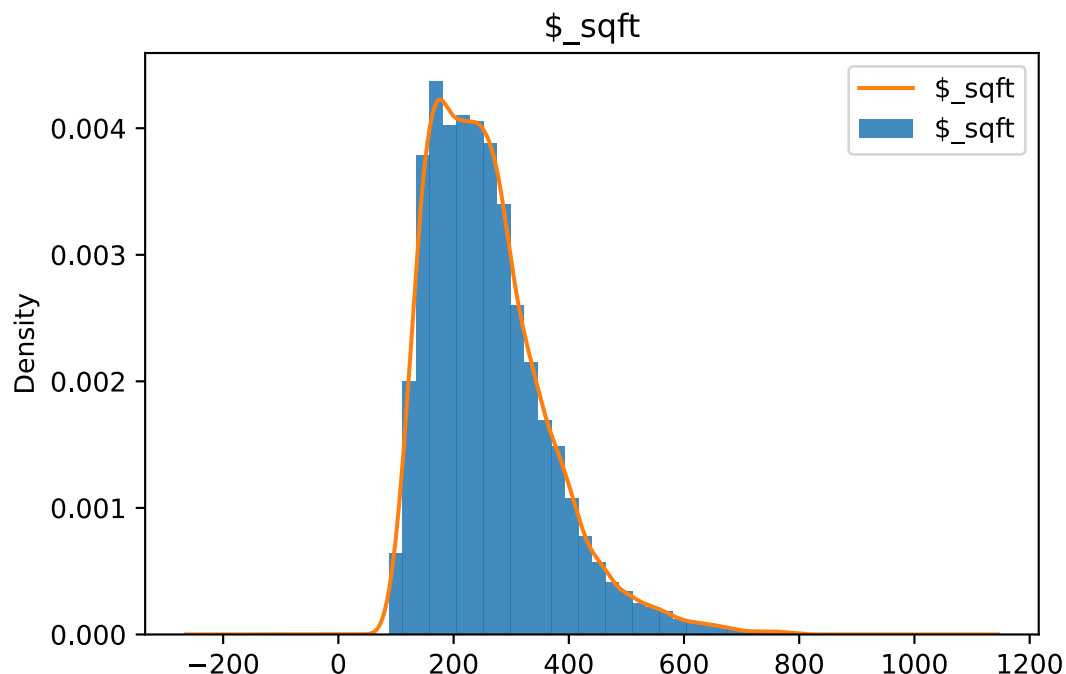












In the above graphs we can see similarity in the shape of latitude histogram as well as year built. This means that as year built increases so does how far north the home was built. This implies multicollinearity between these two features and for a final predictive model one or both will need to be removed. Let's look at a coefficient table for the data.

```
# continue to develop the model
# sqft_lot15 is going to represent the lots around the property
y_target = 'price'
formula = 'price~bedrooms+bathrooms+sqft_living+sqft_lot15+floors+waterfront+view+cc
lm = ols(formula=formula, data=df).fit()
```

```
print(lm.summary())
```

OLS Regression Results

=====

Dep. Variable:	price	R-squared (uncentered):	0.938
Model:	OLS	Adj. R-squared (uncentered):	0.938
Method:	Least Squares	F-statistic:	2.128e+04

Date: Sat 17 Oct 2020 Prob (F-statistic):

```
Date:          Sat, 17 Oct 2020 11:00 (PST)
0.00
Time:          12:04:47    Log-Likelihood:
-2.0565e+05
No. Observations:          15480    AIC:
4.113e+05
Df Residuals:              15469    BIC:
4.114e+05
Df Model:                  11
Covariance Type:          nonrobust
```

```
=====
              coef      std err          t      P>|t|      [0.025      0.975]
-----
bedrooms    -1.305e+04    1575.988     -8.279     0.000    -1.61e+04    -9959.186
bathrooms     4529.5018    2555.053      1.773     0.076     -478.703     9537.706
sqft_living   136.5889       2.700     50.589     0.000      131.297     141.881
sqft_lot15      0.5819       0.044     13.369     0.000       0.497       0.667
floors       -1658.3807    2635.354     -0.629     0.529    -6823.984     3507.222
waterfront    2.31e+05     1.82e+04     12.720     0.000     1.95e+05     2.67e+05
view          3.896e+04     1792.143     21.742     0.000     3.55e+04     4.25e+04
condition     3.737e+04     1804.636     20.710     0.000     3.38e+04     4.09e+04
grade         8.831e+04     1698.403     51.998     0.000     8.5e+04      9.16e+04
yr_built      -192.1698        6.855    -28.034     0.000     -205.606    -178.733
distance     -1.406e+04     172.466    -81.551     0.000    -1.44e+04    -1.37e+04
=====
Omnibus:                2830.898    Durbin-Watson:                1.972
Prob(Omnibus):           0.000    Jarque-Bera (JB):            8853.815
Skew:                    0.940    Prob(JB):                     0.00
Kurtosis:                6.192    Cond. No.                    4.88e+05
=====
```

Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The condition number is large, 4.88e+05. This might indicate that there are strong multicollinearity or other numerical problems.

```
# Let's repeat this analysis by addressing features with a high likelihood of multi
```

```
y_target = 'price'
formula = 'price~bedrooms+sqft_living+sqft_lot15+waterfront+view+condition+grade+yr_
lm = ols(formula=formula, data=df).fit()
```

```
print(lm.summary())
```

OLS Regression Results

```

=====
Dep. Variable:          price    R-squared (uncentered):
0.938
Model:                  OLS      Adj. R-squared (uncentered):
0.938
Method:                 Least Squares    F-statistic:
2.601e+04
Date:                   Sat, 17 Oct 2020    Prob (F-statistic):
0.00
Time:                   12:04:47    Log-Likelihood:
-2.0565e+05
No. Observations:       15480    AIC:
4.113e+05
Df Residuals:           15471    BIC:
4.114e+05
Df Model:                9
Covariance Type:        nonrobust
=====

```

	coef	std err	t	P> t	[0.025	0.975]
bedrooms	-1.257e+04	1552.612	-8.095	0.000	-1.56e+04	-9525.414
sqft_living	138.5070	2.464	56.215	0.000	133.678	143.336
sqft_lot15	0.5782	0.043	13.334	0.000	0.493	0.663
waterfront	2.31e+05	1.82e+04	12.725	0.000	1.95e+05	2.67e+05
view	3.896e+04	1785.078	21.824	0.000	3.55e+04	4.25e+04
condition	3.723e+04	1747.861	21.303	0.000	3.38e+04	4.07e+04
grade	8.861e+04	1641.762	53.974	0.000	8.54e+04	9.18e+04
yr_built	-192.4194	6.778	-28.389	0.000	-205.705	-179.134
distance	-1.404e+04	171.979	-81.664	0.000	-1.44e+04	-1.37e+04

```

=====
Omnibus:                2830.756    Durbin-Watson:                1.972
Prob(Omnibus):           0.000    Jarque-Bera (JB):              8853.903
Skew:                    0.940    Prob(JB):                      0.00
Kurtosis:                6.192    Cond. No.                      4.88e+05
=====

```

Warnings:

- [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
- [2] The condition number is large, 4.88e+05. This might indicate that there are strong multicollinearity or other numerical problems.

```

y_target = 'price'
formula = 'price~bedrooms+sqft_living+sqft_lot15+waterfront+view+condition+grade+dis

```



```
X = ['price', 'bedrooms', 'sqft_living', 'sqft_lot15', 'waterfront', 'view', 'condition', '']
lm = ols(formula=formula, data=df).fit()
```

```
print(lm.summary())
```

OLS Regression Results

```
=====
```

Dep. Variable:	price	R-squared (uncentered):	0.935
Model:	OLS	Adj. R-squared (uncentered):	0.935
Method:	Least Squares	F-statistic:	2.772e+04
Date:	Sat, 17 Oct 2020	Prob (F-statistic):	0.00
Time:	12:04:47	Log-Likelihood:	-2.0604e+05
No. Observations:	15480	AIC:	4.121e+05
Df Residuals:	15472	BIC:	4.122e+05
Df Model:	8		
Covariance Type:	nonrobust		

```
=====
```

	coef	std err	t	P> t	[0.025	0.975]
bedrooms	-2.66e+04	1509.610	-17.622	0.000	-2.96e+04	-2.36e+04
sqft_living	173.7200	2.184	79.558	0.000	169.440	178.000
sqft_lot15	0.5894	0.044	13.253	0.000	0.502	0.677
waterfront	2.172e+05	1.86e+04	11.668	0.000	1.81e+05	2.54e+05
view	4.251e+04	1826.415	23.276	0.000	3.89e+04	4.61e+04
condition	9590.4387	1488.772	6.442	0.000	6672.270	1.25e+04
grade	5.029e+04	958.474	52.469	0.000	4.84e+04	5.22e+04
distance	-1.535e+04	170.000	-90.279	0.000	-1.57e+04	-1.5e+04

```
=====
```

Omnibus:	2922.542	Durbin-Watson:	1.970
Prob(Omnibus):	0.000	Jarque-Bera (JB):	9177.120
Skew:	0.968	Prob(JB):	0.00
Kurtosis:	6.237	Cond. No.	4.87e+05

```
=====
```

Warnings:

- [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
- [2] The condition number is large, 4.87e+05. This might indicate that there are strong multicollinearity or other numerical problems.

Regression output analysis for model.

For each of the above OLS Regression outputs we can reject the Jarque Bera null hypothesis and conclude that the distribution is non-normal.

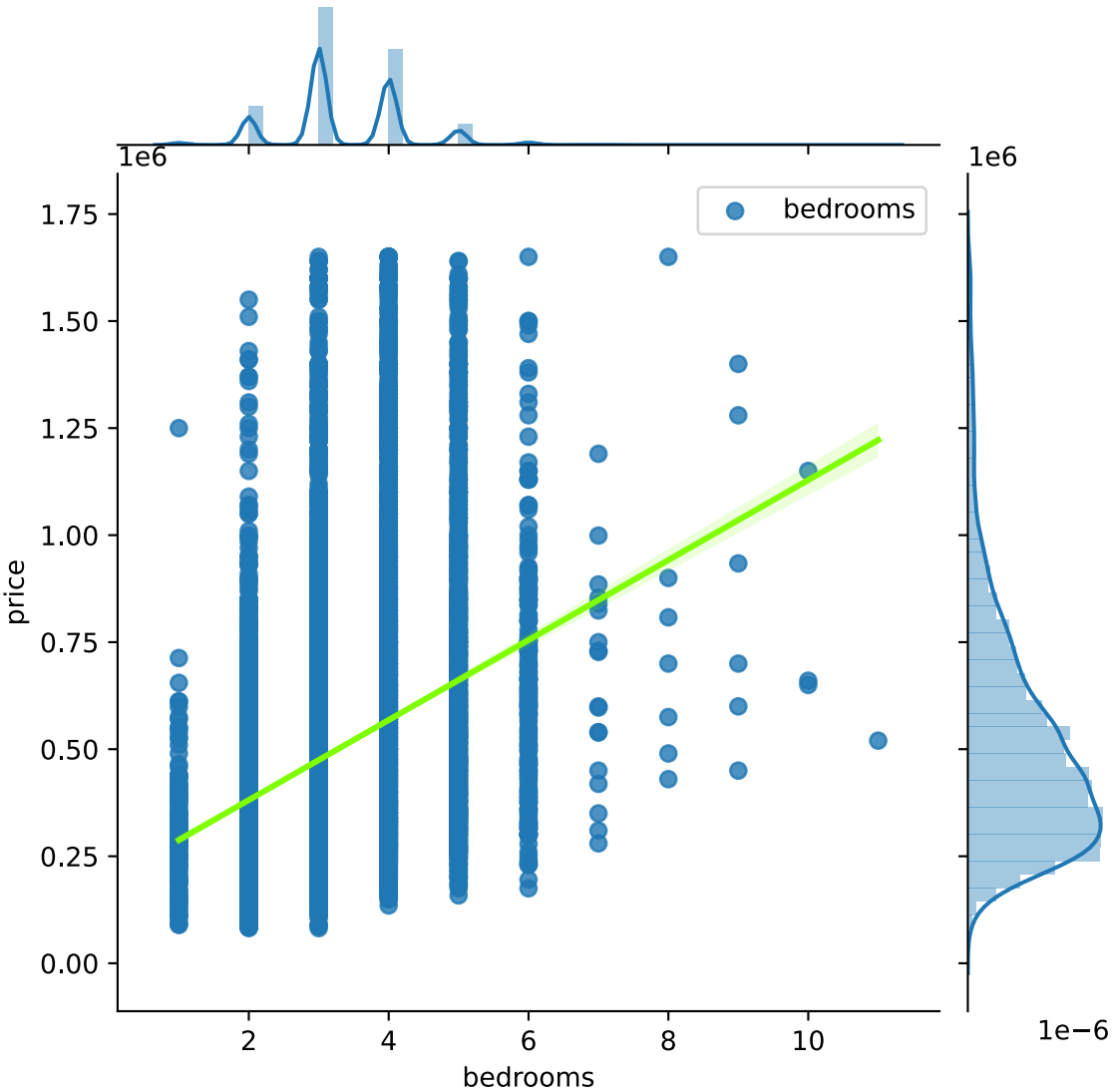
Additionally, we can see that our adjusted R^2 indicates for all regressions that our model is accounting for most (likely too much, or overfitting) of the variation in the dependent variable (price).

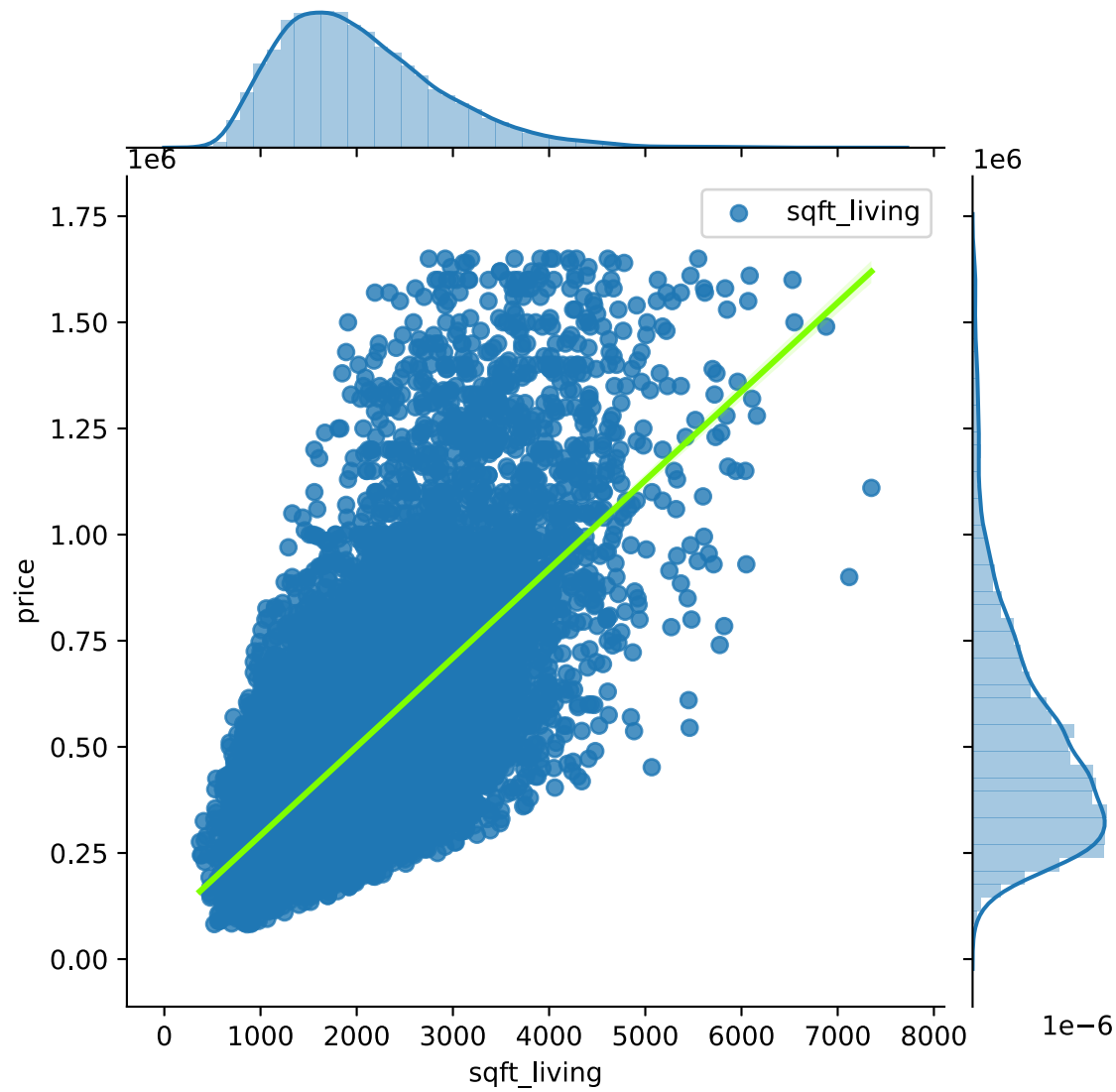
Durbin-Watson test around 2 indicates that there is neither positive autocorrelation nor negative autocorrelation present in the model.

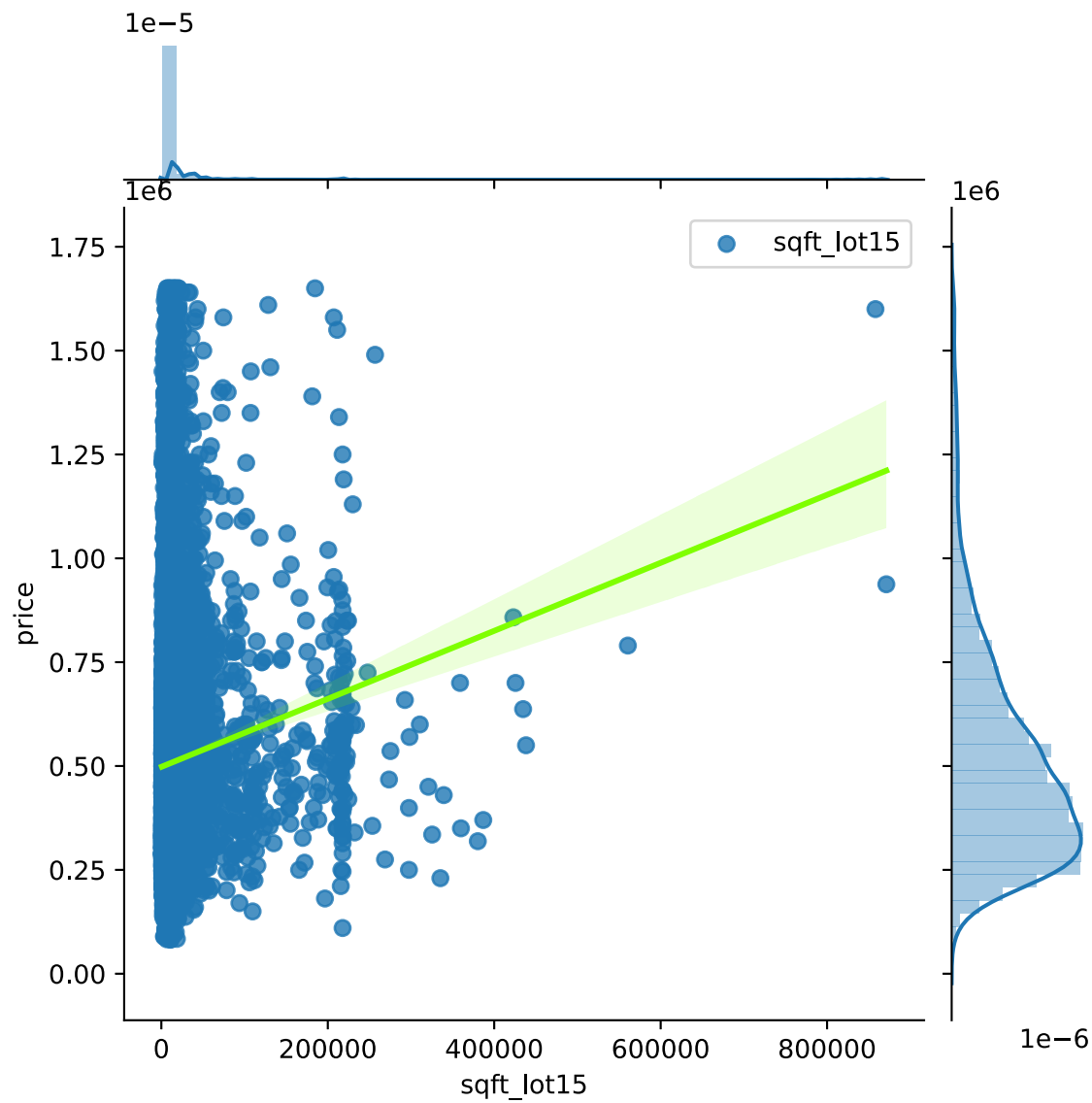
```
df = df[df['bedrooms']<30] # remove the large large high bedroom home from the datas

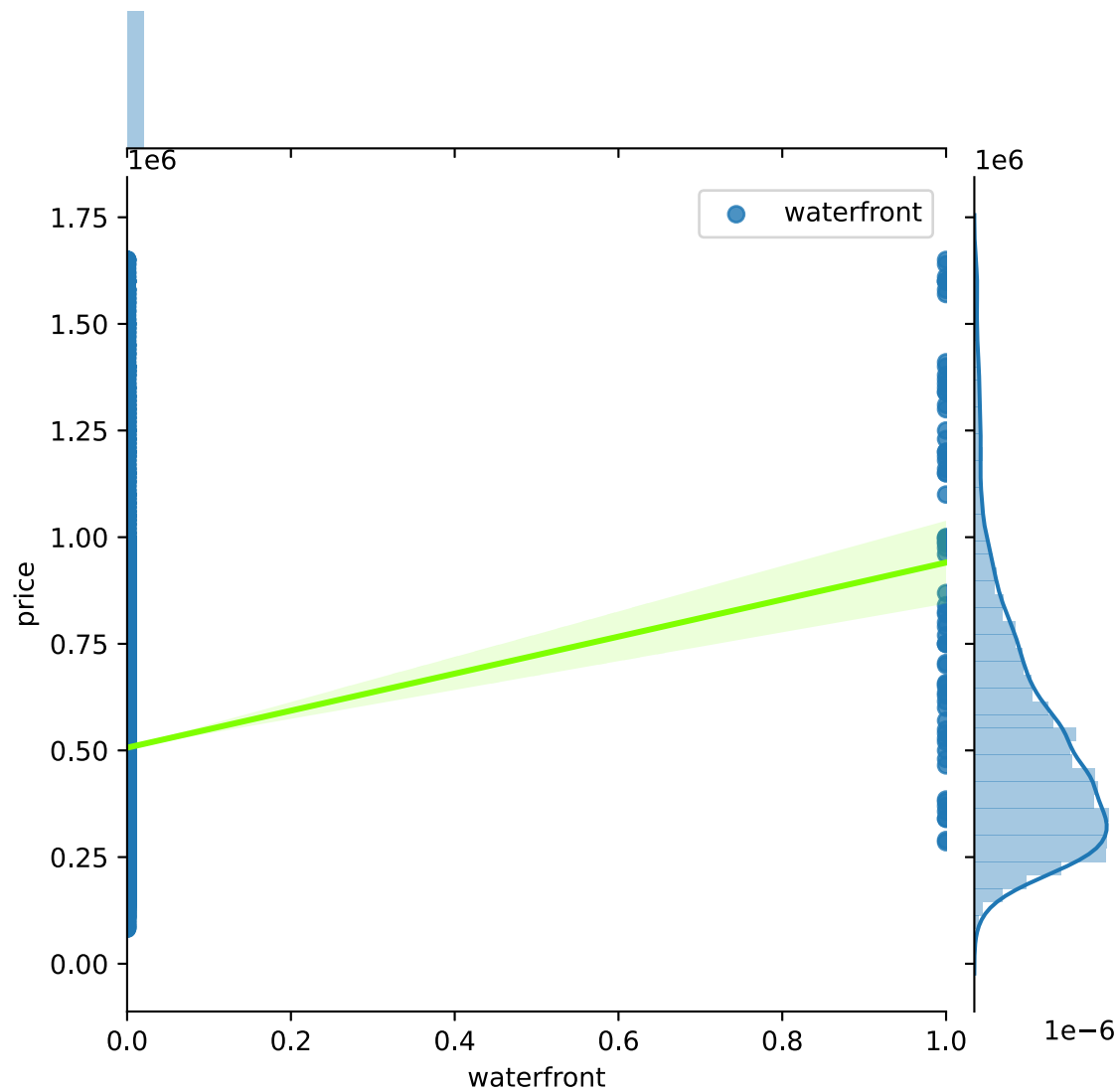
for column in ['bedrooms', 'sqft_living', 'sqft_lot15', 'waterfront', 'view', 'condition']
    sns.jointplot(x = column, y = 'price', data = df, kind = 'reg', label = column, j
    plt.legend()
    plt.show()

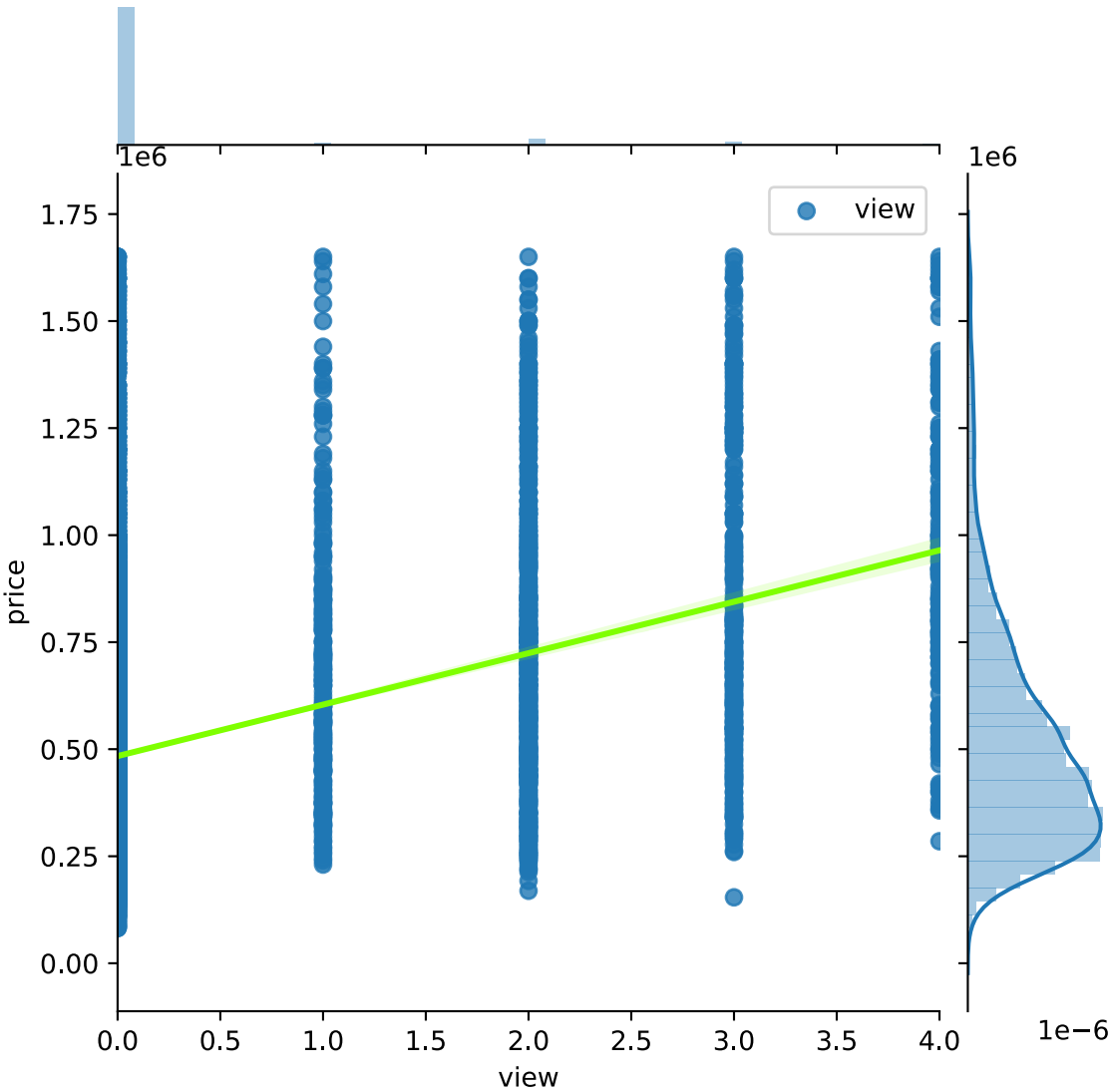
# remove single 30+ bedroom home from data
```

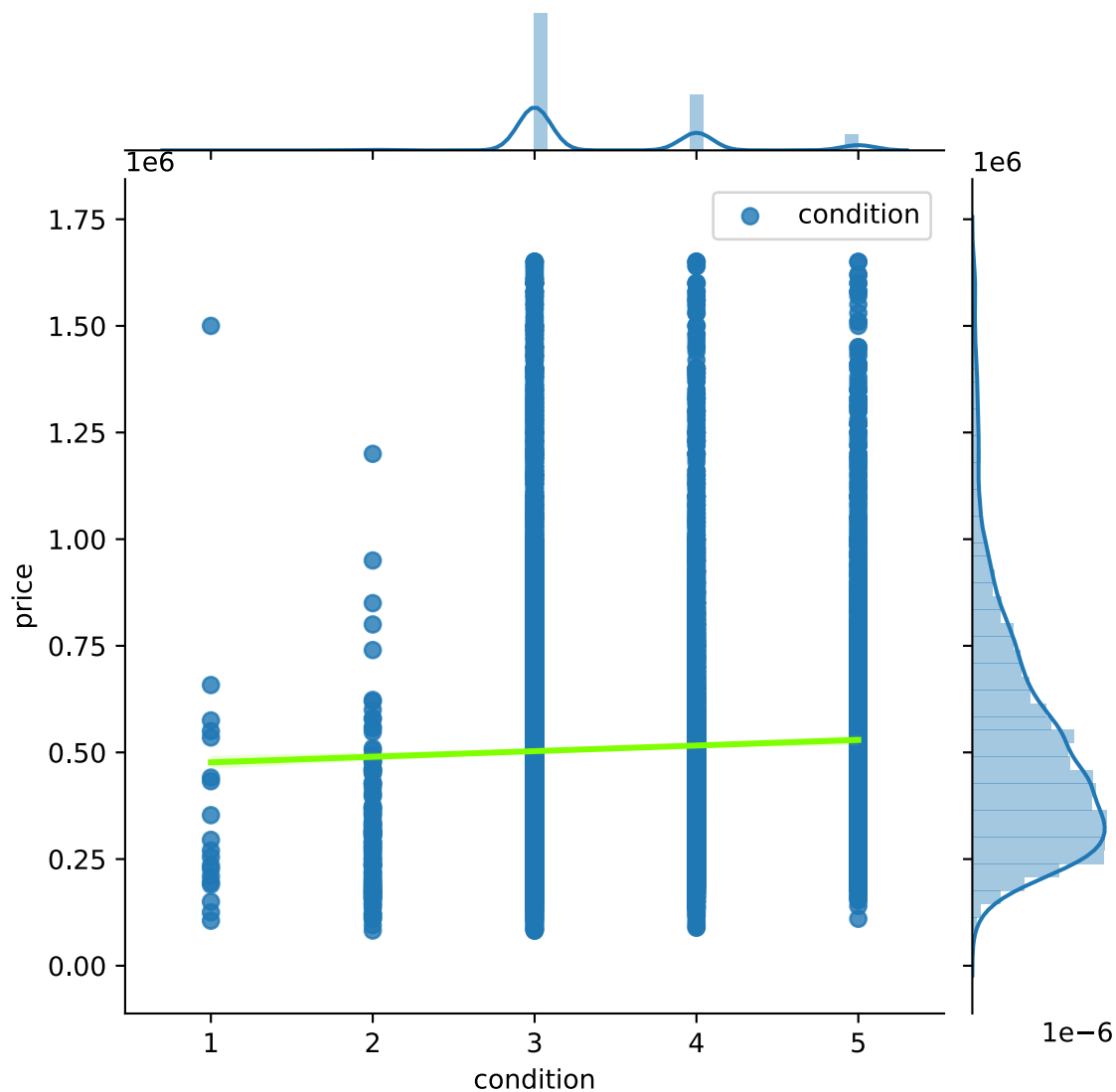












```
# Waterfront is a binary distribution and very few homes in the dataset are on the w
# Remove waterfront from the reg model
# Remove condition from the reg model, flat line/no correlation to price
# Removed gigantic 30+ bedroom house from the data set.
```

```
y_target = 'price'
formula = 'price~bedrooms+sqft_living+sqft_lot15+view+grade+distance-1'
lm = ols(formula=formula, data=df).fit()
```

```
print(lm.summary()) # fit to the data
```



```

                                OLS Regression Results
=====
Dep. Variable:                  price    R-squared (uncentered):
0.934
Model:                          OLS    Adj. R-squared (uncentered):
0.934
Method:                        Least Squares    F-statistic:
3.659e+04
Date:                          Sat, 17 Oct 2020    Prob (F-statistic):
0.00
Time:                          12:05:12    Log-Likelihood:
-2.0610e+05
No. Observations:              15479    AIC:
4.122e+05
Df Residuals:                  15473    BIC:
4.123e+05
Df Model:                      6
Covariance Type:               nonrobust
=====
                                coef    std err          t      P>|t|      [0.025    0.975]
-----
bedrooms      -2.64e+04    1503.098    -17.565     0.000    -2.93e+04    -2.35e+04
sqft_living    168.6491         2.033     82.961     0.000     164.664     172.634
sqft_lot15      0.5922         0.045     13.254     0.000         0.505         0.680
view           4.992e+04    1727.088     28.906     0.000     4.65e+04     5.33e+04
grade          5.538e+04     720.191     76.893     0.000     5.4e+04     5.68e+04
distance      -1.515e+04     170.442    -88.908     0.000    -1.55e+04    -1.48e+04
=====
Omnibus:                 2930.596    Durbin-Watson:           1.971
Prob(Omnibus):           0.000    Jarque-Bera (JB):       9163.325
Skew:                    0.972    Prob(JB):                0.00
Kurtosis:                 6.229    Cond. No.                4.58e+04
=====
```

```

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.

[2] The condition number is large, 4.58e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

```
cm(df,['bedrooms', 'sqft_living', 'sqft_lot15','view','grade','distance'],'price',ad
```

```
bedrooms surpassed threshold with vif=19.64880112008724
sqft_living surpassed threshold with vif=14.403396641134645
```

```
sqft_living surpassed threshold with vif=17.70330041137073
```

```
grade surpassed threshold with vif=22.07347655102637
```

```
distance surpassed threshold with vif=4.124077291265315
```

```
Model contains multicollinear features
```

OLS Regression Results

```
=====
```

```
Dep. Variable:          price    R-squared (uncentered):
0.934
Model:                  OLS      Adj. R-squared (uncentered):
0.934
Method:                 Least Squares    F-statistic:
3.659e+04
Date:                   Sat, 17 Oct 2020    Prob (F-statistic):
0.00
Time:                   12:05:12    Log-Likelihood:
-2.0610e+05
No. Observations:      15479    AIC:
4.122e+05
Df Residuals:          15473    BIC:
4.123e+05
Df Model:               6
Covariance Type:       nonrobust
```

```
=====
```

	coef	std err	t	P> t	[0.025	0.975]
bedrooms	-2.64e+04	1503.098	-17.565	0.000	-2.93e+04	-2.35e+04
sqft_living	168.6491	2.033	82.961	0.000	164.664	172.634
sqft_lot15	0.5922	0.045	13.254	0.000	0.505	0.680
view	4.992e+04	1727.088	28.906	0.000	4.65e+04	5.33e+04
grade	5.538e+04	720.191	76.893	0.000	5.4e+04	5.68e+04
distance	-1.515e+04	170.442	-88.908	0.000	-1.55e+04	-1.48e+04

```
=====
```

```
Omnibus:                2930.596    Durbin-Watson:           1.971
Prob(Omnibus):           0.000    Jarque-Bera (JB):        9163.325
Skew:                    0.972    Prob(JB):                 0.00
Kurtosis:                6.229    Cond. No.                 4.58e+04
```

```
=====
```

Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The condition number is large, 4.58e+04. This might indicate that there are strong multicollinearity or other numerical problems.

Residuals failed test/tests

```
<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x214b6244910>
```

```
# remove grade from model and recheck (high VIF scores)
y_target = 'price'
formula = 'price~bedrooms+sqft_living+sqft_lot15+view+distance-1'
lm = ols(formula=formula, data=df).fit()
```

```
print(lm.summary())
```

OLS Regression Results

```
=====
```

Dep. Variable:	price	R-squared (uncentered):	0.909
Model:	OLS	Adj. R-squared (uncentered):	0.909
Method:	Least Squares	F-statistic:	3.092e+04
Date:	Sat, 17 Oct 2020	Prob (F-statistic):	0.00
Time:	12:05:12	Log-Likelihood:	-2.0861e+05
No. Observations:	15479	AIC:	4.172e+05
Df Residuals:	15474	BIC:	4.173e+05
Df Model:	5		
Covariance Type:	nonrobust		

```
=====
```

	coef	std err	t	P> t	[0.025	0.975]
bedrooms	4.171e+04	1427.557	29.220	0.000	3.89e+04	4.45e+04
sqft_living	232.4534	2.182	106.548	0.000	228.177	236.730
sqft_lot15	0.5104	0.053	9.719	0.000	0.407	0.613
view	5.542e+04	2028.625	27.317	0.000	5.14e+04	5.94e+04
distance	-1.161e+04	192.899	-60.177	0.000	-1.2e+04	-1.12e+04

```
=====
```

Omnibus:	1004.663	Durbin-Watson:	1.951
Prob(Omnibus):	0.000	Jarque-Bera (JB):	3327.768
Skew:	0.290	Prob(JB):	0.00
Kurtosis:	5.196	Cond. No.	4.51e+04

```
=====
```

Warnings:

- [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
- [2] The condition number is large, 4.51e+04. This might indicate that there are strong multicollinearity or other numerical problems.

```
cm(df,['bedrooms', 'sqft_living', 'sqft_lot15', 'view', 'distance'], 'price', add_cons
# lets also remove bedrooms (double other features VIF scores)
```

```
bedrooms surpassed threshold with vif=12.824201566498408
sqft_living surpassed threshold with vif=12.0036147073731
distance surpassed threshold with vif=3.822219717502709
Model contains multicollinear features
```

OLS Regression Results

```
=====
Dep. Variable:          price    R-squared (uncentered):
0.909
Model:                  OLS      Adj. R-squared (uncentered):
0.909
Method:                 Least Squares    F-statistic:
3.092e+04
Date:                  Sat, 17 Oct 2020    Prob (F-statistic):
0.00
Time:                  12:05:12    Log-Likelihood:
-2.0861e+05
No. Observations:      15479    AIC:
4.172e+05
Df Residuals:          15474    BIC:
4.173e+05
Df Model:               5
Covariance Type:       nonrobust
```

```
=====
              coef      std err          t      P>|t|      [0.025      0.975]
-----
bedrooms      4.171e+04    1427.557     29.220     0.000     3.89e+04     4.45e+04
sqft_living   232.4534       2.182    106.548     0.000     228.177     236.730
sqft_lot15     0.5104       0.053     9.719     0.000       0.407       0.613
view          5.542e+04    2028.625     27.317     0.000     5.14e+04     5.94e+04
distance     -1.161e+04     192.899    -60.177     0.000    -1.2e+04    -1.12e+04
=====
Omnibus:                 1004.663    Durbin-Watson:                 1.951
Prob(Omnibus):            0.000    Jarque-Bera (JB):                 3327.768
Skew:                     0.290    Prob(JB):                         0.00
Kurtosis:                 5.196    Cond. No.                        4.51e+04
=====
```

Warnings:

- [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
- [2] The condition number is large, 4.51e+04. This might indicate that there are

strong multicollinearity or other numerical problems.
Residuals failed test/tests

<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x214b7377be0>

cm(df,['sqft_living', 'sqft_lot15', 'view', 'distance'],'price',add_constant=False,s

OLS Regression Results						
=====						
Dep. Variable:	price		R-squared (uncentered):			
0.904						
Model:	OLS		Adj. R-squared (uncentered):			
0.904						
Method:	Least Squares		F-statistic:			
3.642e+04						
Date:	Sat, 17 Oct 2020		Prob (F-statistic):			
0.00						
Time:	12:05:13		Log-Likelihood:			
-2.0902e+05						
No. Observations:	15479		AIC:			
4.181e+05						
Df Residuals:	15475		BIC:			
4.181e+05						
Df Model:	4					
Covariance Type:	nonrobust					
=====						
	coef	std err	t	P> t	[0.025	0.975]

sqft_living	286.4363	1.192	240.303	0.000	284.100	288.773
sqft_lot15	0.2755	0.053	5.168	0.000	0.171	0.380
view	4.991e+04	2074.763	24.056	0.000	4.58e+04	5.4e+04
distance	-9534.1252	184.243	-51.748	0.000	-9895.263	-9172.987
=====						
Omnibus:	831.658	Durbin-Watson:		1.926		
Prob(Omnibus):	0.000	Jarque-Bera (JB):		3103.675		
Skew:	0.128	Prob(JB):		0.00		
Kurtosis:	5.179	Cond. No.		4.48e+04		
=====						

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The condition number is large, 4.48e+04. This might indicate that there are strong multicollinearity or other numerical problems.

Residuals failed test/tests

<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x214b6087ac0>

```
cm(df,['sqft_living','view', 'distance'],'price',add_constant=False,show_summary=True
# remove condition
```

OLS Regression Results

```
=====
Dep. Variable:          price    R-squared (uncentered):
0.904

Model:                  OLS      Adj. R-squared (uncentered):
0.904

Method:                 Least Squares    F-statistic:
4.848e+04

Date:                   Sat, 17 Oct 2020    Prob (F-statistic):
0.00

Time:                   12:05:13    Log-Likelihood:
-2.0904e+05

No. Observations:      15479    AIC:
4.181e+05

Df Residuals:          15476    BIC:
4.181e+05

Df Model:               3

Covariance Type:        nonrobust
=====
```

	coef	std err	t	P> t	[0.025	0.975]
sqft_living	286.9358	1.189	241.317	0.000	284.605	289.266
view	5.056e+04	2072.629	24.395	0.000	4.65e+04	5.46e+04
distance	-9315.7122	179.479	-51.904	0.000	-9667.513	-8963.911

```
=====
Omnibus:                829.237    Durbin-Watson:                1.927
Prob(Omnibus):           0.000    Jarque-Bera (JB):              3056.391
Skew:                    0.136    Prob(JB):                      0.00
Kurtosis:                5.160    Cond. No.                      3.20e+03
=====
```

Warnings:

[1] Standard errors assume that the covariance matrix of the errors is correctly specified.

[2] The condition number is large, 3.2e+03. This might indicate that there are strong multicollinearity or other numerical problems.

Residuals failed test/tests

<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x214b741a6a0>

```
cm(df,['sqft_living','sqft_lot15', 'distance'],'price',add_constant=False,show_summa
# remove condition
```

OLS Regression Results

=====

```
Dep. Variable:          price    R-squared (uncentered):
0.900
Model:                OLS      Adj. R-squared (uncentered):
0.900
Method:              Least Squares    F-statistic:
4.663e+04
Date:                Sat, 17 Oct 2020    Prob (F-statistic):
0.00
Time:                12:05:13    Log-Likelihood:
-2.0931e+05
No. Observations:      15479    AIC:
4.186e+05
Df Residuals:          15476    BIC:
4.186e+05
Df Model:                3
Covariance Type:      nonrobust
```

=====

	coef	std err	t	P> t	[0.025	0.975]
sqft_living	294.6556	1.163	253.342	0.000	292.376	296.935
sqft_lot15	0.3536	0.054	6.525	0.000	0.247	0.460
distance	-1.012e+04	185.982	-54.434	0.000	-1.05e+04	-9759.231

```
=====
Omnibus:                944.168    Durbin-Watson:                1.928
Prob(Omnibus):          0.000    Jarque-Bera (JB):              3217.515
Skew:                   0.249    Prob(JB):                      0.00
Kurtosis:               5.177    Cond. No.                      3.94e+03
=====
```

Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The condition number is large, 3.94e+03. This might indicate that there are strong multicollinearity or other numerical problems.

Residuals failed test/tests

<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x214b62cc580>

cm(df,['sqft_living', 'view'],'price',add_constant=False,show_summary=True,vif_thres

OLS Regression Results

```

=====
Dep. Variable:          price    R-squared (uncentered):
0.887
Model:                  OLS      Adj. R-squared (uncentered):
0.887
Method:                 Least Squares    F-statistic:
6.079e+04
Date:                   Sat, 17 Oct 2020    Prob (F-statistic):
0.00
Time:                   12:05:13    Log-Likelihood:
-2.1028e+05
No. Observations:       15479    AIC:
4.206e+05
Df Residuals:           15477    BIC:
4.206e+05
Df Model:                2
Covariance Type:        nonrobust
=====

```

	coef	std err	t	P> t	[0.025	0.975]
sqft_living	236.5802	0.745	317.614	0.000	235.120	238.040
view	6.375e+04	2228.793	28.602	0.000	5.94e+04	6.81e+04

```

=====
Omnibus:                1360.587    Durbin-Watson:           1.959
Prob(Omnibus):           0.000    Jarque-Bera (JB):        2779.417
Skew:                    0.582    Prob(JB):                 0.00
Kurtosis:                4.719    Cond. No.                 3.18e+03
=====

```


Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The condition number is large, 3.18e+03. This might indicate that there are strong multicollinearity or other numerical problems.

Residuals failed test/tests

<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x214b63071f0>

cm(df,['sqft_living', 'distance'],'price',add_constant=False,show_summary=True,vif_t

OLS Regression Results

```

=====
Dep. Variable:          price    R-squared (uncentered):
0.900
Model:                  OLS      Adj. R-squared (uncentered):
0.900
Method:                 Least Squares    F-statistic:
6.974e+04
Date:                  Sat, 17 Oct 2020    Prob (F-statistic):
0.00
Time:                  12:05:13    Log-Likelihood:
-2.0933e+05
No. Observations:      15479    AIC:
4.187e+05
Df Residuals:          15477    BIC:
4.187e+05
Df Model:               2
Covariance Type:       nonrobust
=====
               coef      std err          t      P>|t|      [0.025      0.975]
-----
sqft_living    295.4378      1.158     255.032      0.000      293.167      297.708
distance     -9852.3292     181.513     -54.279      0.000     -1.02e+04     -9496.542
=====
Omnibus:            947.081    Durbin-Watson:           1.929
Prob(Omnibus):      0.000    Jarque-Bera (JB):        3148.574
Skew:               0.261    Prob(JB):                 0.00
Kurtosis:           5.147    Cond. No.                 275.
=====

```

Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Residuals failed test/tests

<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x214b6226370>

```
# lets look at residuals
```

```
# time to redo the training and testing of our variables after having eliminated some
X = df[['sqft_living','distance']]
y = df['price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .20, random_state=42)
```

```
reg = LinearRegression(fit_intercept=False) # create linear regression
reg.fit(X_train,y_train) # fit the model
```

```
LinearRegression(fit_intercept=False)
```

```
# evaluate the model on the testing data
a1 = reg.score(X_test, y_test)
X_train.shape
a1 # R^2
```

```
0.5008896889890428
```

```
r2train = reg.score(X_train, y_train)
r2test = reg.score(X_test, y_test)
num_obtrain = X_train.shape[0]
num_obtest = X_test.shape[0]
ptrain = X_train.shape[1]
ptest = X_test.shape[1]
```

```
# compare the r2 scores of the model on the training and test data
r2tra, r2tes = reg.score(X_train,y_train),reg.score(X_test,y_test)
print(r2tra)
print(r2tes)
```

#a adjusted r^2 for the data is telling us that this model's features explain ~60% c

```
0.5276807436545992
0.5008896889890428
```

```
# calc adjusted r squared and VIF score for training data
clc(r2tra,num_obtrain,ptrain)
```

```
Adjusted R^2 is:  0.5276044400590667
VIF score is:  2.1168700233444966
```

```
(2.1168700233444966, 0.5276044400590667)
```

```
# calc adjusted r squared and VIF score for training data
clc(r2tes,num_obtest,ptest)
```

```
Adjusted R^2 is:  0.5005669535794011
VIF score is:  2.0022703887276356
```

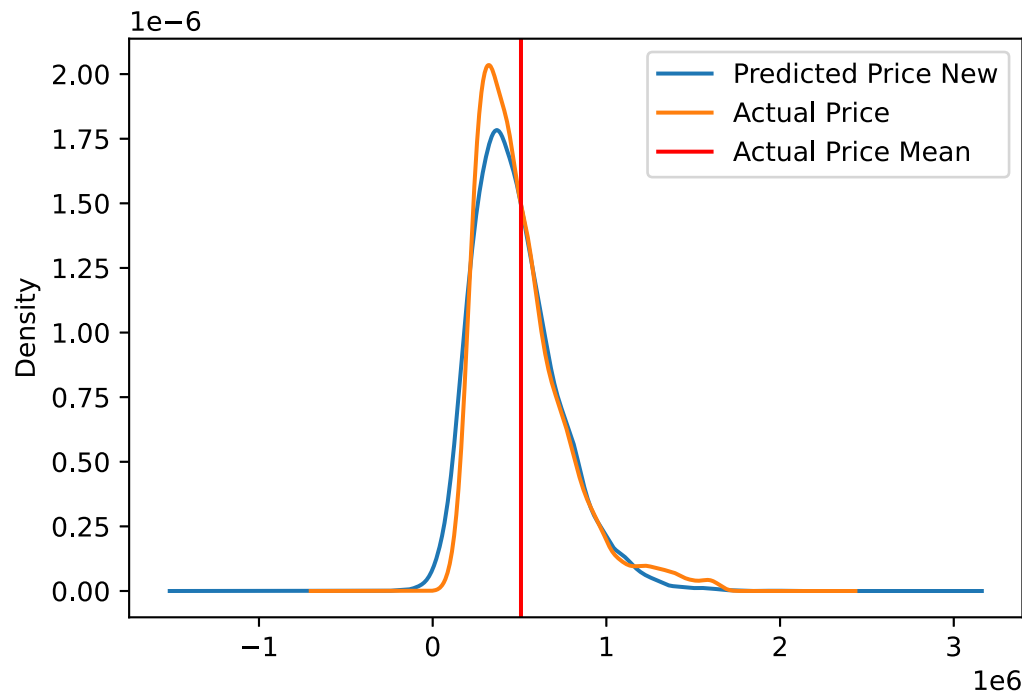
```
(2.0022703887276356, 0.5005669535794011)
```

```
df['PredictedPrice2'] = reg.predict(X) # create a column with a new predicted price
```

```
x_mean = df['price'].mean()
df.head()
```

```
df['PredictedPrice2'].plot.kde(label='Predicted Price New')
df['price'].plot.kde(label='Actual Price')
plt.axvline(x_mean, 0, 5, label='Actual Price Mean', c='r')
plt.legend()
# this creates a predictive curve much closer to the actual home prices and we can t
```

<matplotlib.legend.Legend at 0x214b611fee0>



Residuals

```
df['residuals'] = abs(df['price']-df['PredictedPrice'])
df['residuals2'] = abs(df['price']-df['PredictedPrice2'])
df_residuals = sum(df['residuals'])
df_residuals2 = sum(df['residuals2'])
df['residuals_sq'] = df['residuals']**2
df['residuals_sq2'] = df['residuals2']**2

print("Multicollinear model has a sum residual of: ",df_residuals)
print("Non multicollinear model has a sum residual of: ",df_residuals2)
print("Difference: ",df_residuals-df_residuals2)
print("SSR (multicollinear): ",sum(df['residuals_sq']))
print("SSR: ",sum(df['residuals_sq2']))
```

```
Multicollinear model has a sum residual of: 1666026507.9482143
Non multicollinear model has a sum residual of: 2089114835.3847826
Difference: -423088327.43656826
SSR (multicollinear): 328933405543480.8
SSR: 505457750084720.7
```

```
plt.figure(figsize=(15,12)) # model residual comparison
sns.scatterplot(data=df,x=df['price'],y=df['residuals'],alpha=.5,color='blue') # [['
sns.scatterplot(data=df,x=df['price'],y=df['residuals2'],alpha=.3,color='red') # [['
plt.legend(loc='lower right')
plt.title('Price of Homes Sold in King County')
plt.show()
```

```
# residuals graph for our 2 models to this point
```

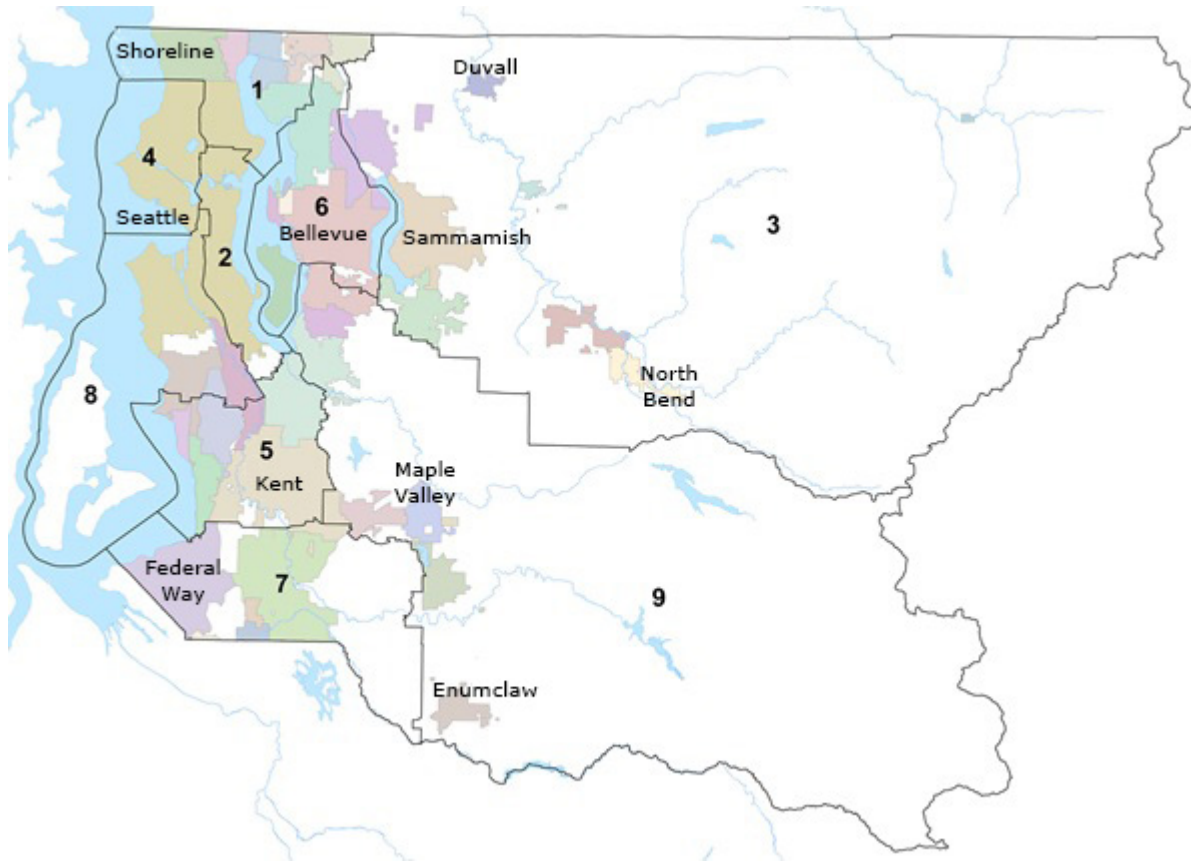
No handles with labels found to put in legend.

EDA and Modeling Findings

1. location (latitude and longitude) have a large affect on home price but are unfit features to contribute to a predictive model, and so distance is used instead.
2. condition and grade are likely collinear with one another, but not always
3. best fit predictive model to this point is based on beta coefficients of sq_footage and distance from Seattle

Question 1 - How does longitude and latitude affect home price in King County?

```
# Let's look at a map of King County and compare that to a grid of home prices
```



```
plt.figure(figsize=(10,7))
sns.scatterplot(x=df['long'],y=df['lat'],hue=df['price'],palette='Reds')
plt.legend(loc='lower right')
plt.title('Price of Homes Sold in King County')
plt.show()
```

If we compare these two maps, we can see that the areas in and around Bellevue (and Mercer Island) are demanding higher prices for homes based on those locations. Additionally, the immediate areas around downtown Seattle are higher priced homes. Let's see what a regression looks like when just taking into account distance, view, and lot size to determine price.

```
X = df[['distance','sqft_living']]
y = df['price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .20, random_st

# create test and train variables, assign random state for reproducibility
reg = LinearRegression(fit_intercept=False) # create linear regression
reg.fit(X_train,y_train) # fit the model

# evaluate the model on the testing data
a1 = reg.score(X_test, y_test)
```

```
X_train.shape
a1
r2train = reg.score(X_train,y_train)
r2test = reg.score(X_test, y_test)
num_obtrain = X_train.shape[0]
num_obtest = X_test.shape[0]
ptrain = X_train.shape[1]
ptest = X_test.shape[1]

# compare the r2 scores of the model on the training and test data
r2tra, r2tes = reg.score(X_train,y_train),reg.score(X_test,y_test)
print(r2tra)
print(r2tes)

# calc adjusted r squared and VIF score for training data
clc(r2tra,num_obtrain,ptrain) # low VIF score is good but we can probably reduce it

# calc adjusted r squared and VIF score for test data
clc(r2tes,num_obtest,ptest) # same as above test data is giving us similar indicator
df['Predicted_Price3'] = reg.predict(X) # create a column with a new predicted price
x_mean = df['Predicted_Price3'].mean()
df.head()

df['Predicted_Price3'].plot.kde(label='Predicted Price')
df['price'].plot.kde(label='Actual Price')
plt.axvline(x_mean, 0, 5,label='Actual Price Mean', c='r')
plt.legend()

# graph below shows that our model is missing some information and is anticipating a
# the distribution of home prices leans to and peaks to the left of the mean home pr

0.5276807436545992
0.5008896889890428
Adjusted R^2 is: 0.5276044400590667
VIF score is: 2.1168700233444966
Adjusted R^2 is: 0.5005669535794011
VIF score is: 2.0022703887276356

<matplotlib.legend.Legend at 0x214b62db220>
```

```
x_features = ['sqft_lot15', 'distance', 'sqft_living']
outcome = 'price'
X = df[x_features]
```

```
predictors = '+'.join(x_features)
formula = outcome + '~' + predictors + '-1'
model = ols(formula=formula, data=df).fit()
model.summary()
```

OLS Regression Results

Dep. Variable:	price	R-squared (uncentered):	0.900
Model:	OLS	Adj. R-squared (uncentered):	0.900
Method:	Least Squares	F-statistic:	4.663e+04
Date:	Sat, 17 Oct 2020	Prob (F-statistic):	0.00
Time:	12:05:24	Log-Likelihood:	-2.0931e+05
No. Observations:	15479	AIC:	4.186e+05
Df Residuals:	15476	BIC:	4.186e+05
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P> t	[0.025	0.975]
sqft_lot15	0.3536	0.054	6.525	0.000	0.247	0.460
distance	-1.012e+04	185.982	-54.434	0.000	-1.05e+04	-9759.231
sqft_living	294.6556	1.163	253.342	0.000	292.376	296.935

Omnibus:	944.168	Durbin-Watson:	1.928
Prob(Omnibus):	0.000	Jarque-Bera (JB):	3217.515
Skew:	0.249	Prob(JB):	0.00

Kurtosis:	5.177	Cond. No.	3.94e+03
-----------	-------	-----------	----------

Warnings:

- [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
- [2] The condition number is large, 3.94e+03. This might indicate that there are strong multicollinearity or other numerical problems.

Homoscedasticity

```
plt.scatter(model.predict(df[x_features]), model.resid)
plt.plot(model.predict(df[x_features]), [0 for i in range(len(df))])
```

```
[<matplotlib.lines.Line2D at 0x214b47c9f10>]
```

This graph appears to have a slight pattern as opposed to being randomly distributed around the flat 0 axis. I would say there is some heteroskedasticity, and that there may be some evidence that there should be concern that distance from Seattle and square footage are somehow correlated with one another.

Checking for Normality

```
fig = sm.graphics.qqplot(model.resid, dist=stats.norm, line='45', fit=True)
```

The qq plot for normality indicates that within our dataset are more extreme values than would be expected in a normal distribution.

Question 1 Answer and Further Investigation -

When latitude and longitude are included in the regression, their VIF threshold is surpassed by a large margin. A more measured approach might be to look at the data

after breaking down the dataset into smaller samples in their specific areas and creating neighborhood specific regressions. This would allow you to truly understand which specific features are affecting home prices, and therefore which to highlight. Location is a huge nonlinear factor on the price of the home.

Distance from Seattle, however, does introduce a better feature to the model and is helping to predict price. The graph above shows that our model is missing some information and is anticipating a less peaky distribution whereas the actual home sales are skewed towards the lower price range. Additionally, the distribution of home prices leans to and peaks to the left of the mean home price, indicating positive skewness in the actual home price sales. This is also why our predictive model also peaks just left of the actual price mean.

Question 2 - Is temporary condition or overall grade (build quality) affecting home price more significantly?

```
plt.figure(figsize=(11,9))
sns.jointplot(x='grade',y='price', data=df, kind = 'reg', label = column, joint_kws
plt.legend(loc='lower right')
plt.ylabel('Home Price in $100,000s',fontsize=12)
plt.xlabel('Build Grade',fontsize=12)
plt.show()
```

<Figure size 792x648 with 0 Axes>

```
plt.figure(figsize=(11,9))
sns.jointplot(x='condition',y='price', data = df, kind = 'reg', label = column, joi
plt.legend(loc='lower right')
plt.ylabel('Home Price in $100,000s',fontsize=12)
plt.xlabel('Home Condition', fontsize=12)
plt.show()
```

<Figure size 792x648 with 0 Axes>

Question 2 Answer and Further Investigation

While neither feature was used in our final regressive predictive model, it is clear that original build grade is strongly positively correlated with home prices. Condition may not affect sale price much. This is likely because condition is temporary and can be remedied fairly easily, whereas to change a home's build grade requires substantial renovation from the literal ground up. To further investigate, suggest breaking the data down by geographical location again. The analysis of the entire county is not apples to apples. Real estate in downtown Seattle with a view of the city is not the same as real estate 15 miles south of Seattle.

Question 3 - What parts of King County command the highest values per square foot?

```
df.head()
```

```
<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }
```

```
.dataframe tbody tr th {  
    vertical-align: top;  
}
```

```
.dataframe thead th {  
    text-align: right;  
}
```

```
</style>
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	wa
1	538000.00	3	2.25	2570	7242	2.00	0.0
3	604000.00	4	3.00	1960	5000	1.00	0.0
4	510000.00	3	2.00	1680	8080	1.00	0.0
5	1230000.00	4	4.50	5420	101930	1.00	0.0
6	257500.00	3	2.25	1715	6819	2.00	0.0

5 rows × 25 columns

```
fig1 = plt.figure(figsize=(12,9))
ax1 = sns.scatterplot(x=df['long'],y=df['lat'],hue=df['_sqft'],size=df['price'],pal
ax1.add_patch(patches.Circle((-122.225,47.565),.035,alpha=0.3,facecolor='green',edge
ax1.add_patch(patches.Circle((-122.22,47.625),.025,alpha=0.3,facecolor='red',edgecol
ax1.add_patch(patches.Circle((-122.30,47.633),.028,alpha=0.3,facecolor='orange',edge
ax1.add_patch(patches.Circle((-122.4,47.65),.027,alpha=0.3,facecolor='yellow',edgecc
plt.legend(loc='lower right')
plt.title('Price per Square Foot in King County')
plt.ylabel('Latitude')
plt.xlabel('Longitude')
plt.show()
```

Question 3 Answer and Further Investigation

Bellevue, Capitol Hill, Magnolia, and Mercer Island have homes that are more expensive per square foot. Mercer Island has large homes and you can get more for your dollar compared with Bellevue. Generally speaking, properties south of Seattle in King County are not as valuable, but we can't pinpoint or recommend why that may be other than location.

Isolating these specific locations and neighborhoods in separate regressions analyses will likely be an efficient way to build a better predictive model, the limitation being that the model can't be applied to different neighborhoods.

Question 4 - Our realtor client is most interested in Mercer Island. The client would like to know what features of homes on Mercer Island are most desirable. Let's investigate this specific portion of King County to provide them with an actionable recommendation.

```
# for this question we need to import the csv with zipcodes and remove everything ex
```

```
df_mercer = df[df['zipcode']==98040]
df_mercer.head() # Let's look at summary data and then run a regression with price a
```

```
<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }
```

```
.dataframe tbody tr th {
    vertical-align: top;
}
```

```
.dataframe thead th {
    text-align: right;
}
```

```
</style>
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	w
95	905000.00	4	2.50	3300	10250	1.00	0.
192	799000.00	3	2.50	2140	9897	1.00	0.
351	855000.00	4	2.75	2270	10460	2.00	0.
385	799000.00	4	2.25	2510	11585	2.00	0.
454	811000.00	3	1.75	1870	9897	1.00	0.

5 rows × 25 columns

```
x_features = ['sqft_living', 'sqft_lot']
outcome = 'price'
X = df_mercer[x_features]
```

```
predictors = '+'.join(x_features)
formula = outcome + '~' + predictors + '-1'
model = ols(formula=formula, data=df_mercer).fit()
model.summary()
```

OLS Regression Results

Dep. Variable:	price	R-squared (uncentered):	0.961
-----------------------	-------	--------------------------------	-------

Model:	OLS	Adj. R-squared (uncentered):	0.961
Method:	Least Squares	F-statistic:	2179.
Date:	Sun, 18 Oct 2020	Prob (F-statistic):	2.17e-124
Time:	19:06:43	Log-Likelihood:	-2415.6
No. Observations:	177	AIC:	4835.
Df Residuals:	175	BIC:	4842.
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P> t	[0.025	0.975]
sqft_living	325.8235	10.238	31.824	0.000	305.617	346.030
sqft_lot	5.0139	2.028	2.473	0.014	1.012	9.016

Omnibus:	32.535	Durbin-Watson:	1.962
Prob(Omnibus):	0.000	Jarque-Bera (JB):	92.188
Skew:	-0.721	Prob(JB):	9.59e-21
Kurtosis:	6.228	Cond. No.	10.1

Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Homoscedasticity

```
plt.scatter(model.predict(df_mercer[x_features]), model.resid)
plt.plot(model.predict(df_mercer[x_features]), [0 for i in range(len(df_mercer))])
```

```
[<matplotlib.lines.Line2D at 0x214bd231ac0>]
```

Near homoskedastic residual plot for Mercer Island but using grade feature in place of distance since Mercer Island was the focus. Results in a low condition number and high R-squared.

Checking for Normality

```
fig = sm.graphics.qqplot(model.resid, dist=stats.norm, line='45', fit=True)
```

QQ plot checks for normality - good overall test with indication that there are a few outliers in the predictive model.

Question 4 Answer and Further Investigation

Isolating the home sales on Mercer Island allowed us to isolate a smaller market to understand which features best predict the target. Let's run a quick check model formula for the Mercer Island data.

We recommend that the realtor highlight interior square footage, size of lot, and overall proximity to the city. The most important takeaway from our investigations is location, location, location. Can further isolate neighborhoods in further investigation and regressions, but many features that we started with are either trivial or autocorrelated and can't be used in regression.

```
cm(df_mercer,['sqft_living','sqft_lot'],'price',add_constant=False,show_summary=True)
```

```
sqft_living surpassed threshold with vif=3.776673540885498
```

```
sqft_lot surpassed threshold with vif=3.776673540885498
```

```
Model contains multicollinear features
```

```
OLS Regression Results
```

```
=====
```

```
Dep. Variable:                price    R-squared (uncentered):
```

```
0.961
```

```
Model:                        OLS    Adj. R-squared (uncentered):
```

```
0.961
```

```
Method:                        Least Squares    F-statistic:
```

```

2179.
Date:                Sun, 18 Oct 2020    Prob (F-statistic):
2.17e-124
Time:                19:08:48    Log-Likelihood:
-2415.6
No. Observations:    177    AIC:
4835.
Df Residuals:        175    BIC:
4842.
Df Model:            2
Covariance Type:     nonrobust
=====
              coef      std err          t      P>|t|      [0.025      0.975]
-----
sqft_living    325.8235     10.238     31.824     0.000     305.617     346.030
sqft_lot        5.0139      2.028      2.473     0.014       1.012       9.016
=====
Omnibus:                 32.535    Durbin-Watson:                 1.962
Prob(Omnibus):            0.000    Jarque-Bera (JB):                 92.188
Skew:                    -0.721    Prob(JB):                         9.59e-21
Kurtosis:                 6.228    Cond. No.                         10.1
=====

```

Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Residuals failed test/tests

```
<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x214bd214b80>
```

Somewhat higher VIF score, but acceptable at 3.78. For each addition square foot of living space, the model predicts (on Mercer Island) that the price of the home will go up by 325.8. For each square foot of lot space, the price will increase by 5.01.