LOADING LIBERARY

In [209…

```
import itertools
import numpy as np
import pandas as pd
pd.set_option('display.max_columns', 50)
import matplotlib.pyplot as plt
from  matplotlib.ticker import PercentFormatter
%matplotlib inline

from geopy import distance, Nominatim
from uszipcode import SearchEngine, SimpleZipcode, Zipcode
import folium

from scipy import stats
from scipy.stats import norm
import seaborn as sns

import statsmodels.api as sm
from statsmodels.tools.eval_measures import rmse, meanabs
from statsmodels.formula.api import ols
from statsmodels.stats.outliers_influence import variance_inflation_factor

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split, cross_val_score, KFold
from sklearn.preprocessing import FunctionTransformer, quantile_transform, scale, StandardScaler, MinMaxScaler, Standards
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from sklearn import metrics

%run "./project_class.ipynb"
func=Master
```

LOADING DATASET

In [136…

```
# df.to_csv('./data/kc_house_data.gz', compression='gzip')
df = pd.read_csv("./data/kc_house_data.gz", index_col=None)
df.head()
```

Out[136…

	Unnamed: 0	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_abo
0	0	7129300520	10/13/2014	221900.0	3	1.00	1180	5650	1.0	NaN	0.0	3	7	11
1	1	6414100192	12/9/2014	538000.0	3	2.25	2570	7242	2.0	0.0	0.0	3	7	21
2	2	5631500400	2/25/2015	180000.0	2	1.00	770	10000	1.0	0.0	0.0	3	6	7
3	3	2487200875	12/9/2014	604000.0	4	3.00	1960	5000	1.0	0.0	0.0	5	7	10
4	4	1954400510	2/18/2015	510000.0	3	2.00	1680	8080	1.0	0.0	0.0	3	8	16

In [137…

```
print("the data set contains {0:,} rows and {1} columns".format(df.shape[0], df.shape[1]))
```

```
the data set contains 21,597 rows and 22 columns
```

Column Names and descriptions for Kings County Data Set

```
id             unique identified for a house
date           Date house was sold
price          Price  is prediction target
bedrooms       Number of Bedrooms/House
bathrooms      Number  of bathrooms/bedrooms
sqft_living    square footage of the home
sqft_lot       square footage of the lot
floors         Total floors (levels) in house
waterfront     House which has a view to a waterfront
view           Has been viewed
condition      How good the condition is ( Overall )
grade          overall grade given to the housing unit, based on King County grading system
sqft_above     square footage of house apart from basement
sqft_basement  square footage of the basement
yr_built       Built Year
yr_renovated   Year when house was renovated
zipcode        zip
lat            Latitude coordinate
long           Longitude coordinate
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
```

DESCRIPTIVE STATISTICS

In [138…

```
df.describe()
```

Out[138…

	Unnamed: 0	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	
count	21597.000000	2.159700e+04	2.159700e+04	21597.000000	21597.000000	21597.000000	2.159700e+04	21597.000000	19221.000000	21534
mean	10798.000000	4.580474e+09	5.402966e+05	3.373200	2.115826	2080.321850	1.509941e+04	1.494096	0.007596	0
std	6234.661218	2.876736e+09	3.673681e+05	0.926299	0.768984	918.106125	4.141264e+04	0.539683	0.086825	0

	Unnamed: 0	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	
min	0.000000	1.000102e+06	7.800000e+04	1.000000	0.500000	370.000000	5.200000e+02	1.000000	0.000000	0
25%	5399.000000	2.123049e+09	3.220000e+05	3.000000	1.750000	1430.000000	5.040000e+03	1.000000	0.000000	0
50%	10798.000000	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	1.500000	0.000000	0
75%	16197.000000	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000	1.068500e+04	2.000000	0.000000	0
max	21596.000000	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3.500000	1.000000	4

In [139…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 22 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Unnamed: 0     21597 non-null  int64  
 1   id             21597 non-null  int64  
 2   date           21597 non-null  object 
 3   price          21597 non-null  float64
 4   bedrooms       21597 non-null  int64  
 5   bathrooms      21597 non-null  float64
 6   sqft_living    21597 non-null  int64  
 7   sqft_lot       21597 non-null  int64  
 8   floors         21597 non-null  float64
 9   waterfront     19221 non-null  float64
 10  view           21534 non-null  float64
 11  condition      21597 non-null  int64  
 12  grade          21597 non-null  int64  
 13  sqft_above     21597 non-null  int64  
 14  sqft_basement  21597 non-null  object 
 15  yr_built       21597 non-null  int64  
 16  yr_renovated   17755 non-null  float64
 17  zipcode        21597 non-null  int64  
 18  lat            21597 non-null  float64
 19  long           21597 non-null  float64
 20  sqft_living15  21597 non-null  int64  
 21  sqft_lot15     21597 non-null  int64  
dtypes: float64(8), int64(12), object(2)
memory usage: 3.6+ MB
```

TOP!

DATA SCRUBBING

Check Missing Values

In [140…

```
for i in df.columns:
    total_nan = df[i].isnull().sum()
    if total_nan > 0:
        print("total missing value of {0:>15}is: {1:>5}".format(i, total_nan))
del total_nan
```

```
total missing value of      waterfrontis:  2376
total missing value of            viewis:    63
total missing value of    yr_renovatedis:  3842
```

both columns "view" and "waterfront" are categories where the "view" represented with the value 1 if the house has been seen and the "waterfront" represented with 1 if the house has waterfront, those will be filled with zeros

In [141…

```
df['waterfront'].fillna(value=0, axis=0, inplace=True)
df['view'].fillna(value=0, axis=0, inplace=True)
```

the column "yr_renovated" represents the year in which the house was renovated, we noticed that only 744 houses were renovated. since the proportion of the renovated houses is so few compared to the entire column, it would make more sense if we use it as a category where we assign 1 to the renovated houses and 0 to those that are not

In [142…

```
df['yr_renovated'].fillna(value=0, axis=0, inplace=True)
df.loc[df['yr_renovated']!=0, ['yr_renovated']] = 1
df.loc[:,'yr_renovated'] = df['yr_renovated'].apply(np.int)  #.astype('int')
df.rename(columns={'yr_renovated': 'renovated'}, inplace=True)
```

Drop the column "Unnamed: 0" from axis 1 matching the string by RegEx

In [143…

```
un_named_columns = df.iloc[:,df.columns.str.contains('^Unnamed', case=False, regex=True)]
df.drop(un_named_columns, axis=1, inplace=True)
del un_named_columns
```

Drop "ID column

In [144…

```
df.drop(columns=['id'], inplace=True)
df['date'] = pd.to_datetime(df['date'], utc=False)
```

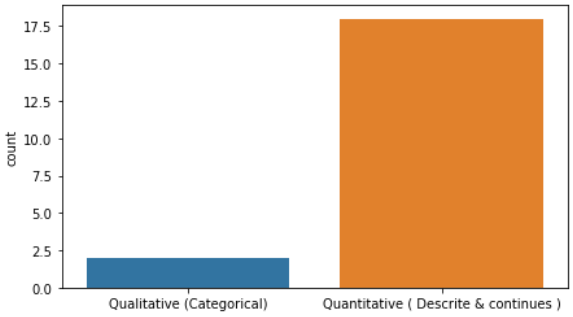
TOP!

DATA EXPLORATION

our first goal is to identify the types of variables we will deal with, so we start by iterating the dataframe columns in alphabetical order instead of analyzing each variable based on statistical type and scale of measurement.

In [145…
```
#define a list for the variables to exclude
var_excluded = set()
```

In [146…
```
# generate a list to categorize our variables type
fig, ax1 = plt.subplots(ncols=1, sharey=True)
fig.set_size_inches(7,4)
sns.axes_style()
variables_type = [ "Quantitative ( Descrite & continues )" if df[col].dtype in ['int64', 'float64'] else "Qualitative (C
plt.figure(figsize=(5,3))
sns.countplot(x=variables_type, ax=ax1)
plt.show()
```

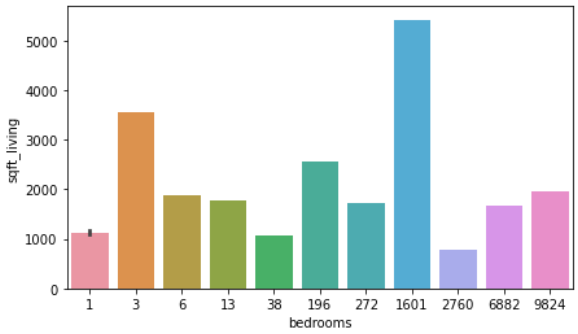



```
<Figure size 360x216 with 0 Axes>
```

BEDROOMS

In [148…
```
fig, ax1 = plt.subplots(ncols=1, sharey=True)
fig.set_size_inches(7,4)
sns.axes_style()
sns.barplot(x=df.bedrooms.value_counts(), y=df.sqft_living)
```

Out[148…
```
<AxesSubplot:xlabel='bedrooms', ylabel='sqft_living'>
```



SQFT_ABOVE & SQFT_BASEMENT

regarding the 2 variables "sqft_above" "sqft_basement" we noticed that the sum of both represents the actual square feet of the entire living area

in fact if we take for example line number 3 where "sqft_living" is 1960 the sqft_above "is 1050 it is easy to come out with the difference of "sqft_basement" which is 910.

in the real world we calculate the house price if the basement is finished since we do not have sufficient data to determine this data, we exclude this variable from our analyzes

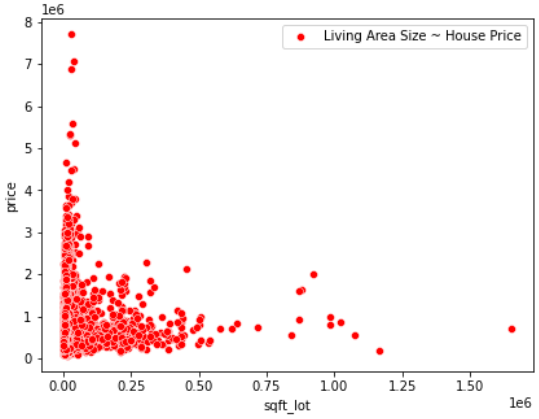
In [149…
```
var_excluded.update(('sqft_above', 'sqft_basement'))
```

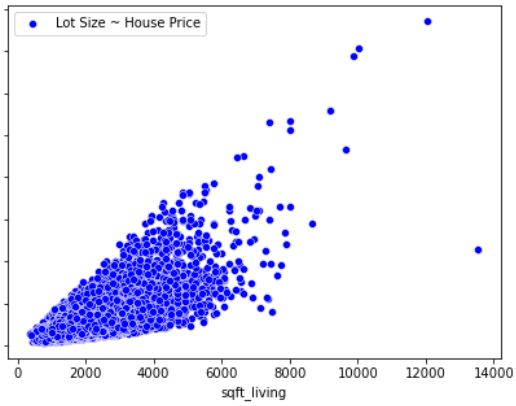
SQFT_LOT & SQFT_LIVING

Let us examine the importance of having a large lot. We will define a ratio of sqft_living over sqft_lot to understand if there is an ideal trade-off between lot size (presumably garden) and living space.

In [150…
```
fig, (ax1, ax2) = plt.subplots(ncols=2, sharey=True)
fig.set_size_inches(15,5)
sns.axes_style()
sns.scatterplot(x=df.sqft_lot, y=df.price, color='red',  ax=ax1, label='Living Area Size ~ House Price')
sns.scatterplot(x=df.sqft_living, y=df.price, color='blue', ax=ax2, label='Lot Size ~ House Price')
```

Out[150…
```
<AxesSubplot:xlabel='sqft_living', ylabel='price'>
```





In [151…
```
_ = np.size(np.where((np.array(df.sqft_living/df.sqft_lot)) >= 1))
print('total houses with sqft_living >= sqft_lot:', _)
```

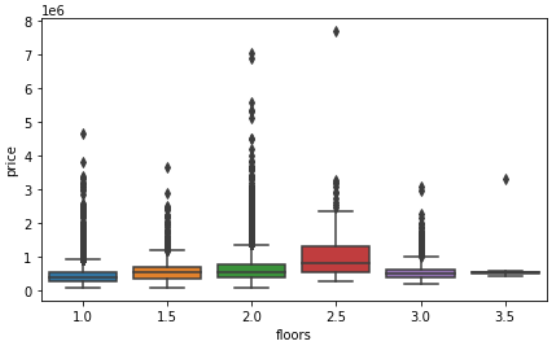
```
total houses with sqft_living >= sqft_lot: 788
```

unexpectedly we got 788 houses where the square footage of living aerea "sqft_living" is greater than lot square footage "sqft_lot", it's more likely about detached structures like garage, patios or even basement.

FLOORS

In [152…
```
fig, ax1 = plt.subplots(ncols=1, sharey=True)
fig.set_figwidth(7,4)
sns.boxplot(x=df['floors'], y=df['price'], ax=ax1)
```

Out[152… `<AxesSubplot:xlabel='floors', ylabel='price'>`



WATERFRONT

the waterfront column describes whether the house has waterfront with the value 1, otherwise the value 0, as observed only 146 houses has a waterfront, that is less than 1% of the entire dataset.

In [153…
```
df['waterfront'].value_counts().to_frame()
```

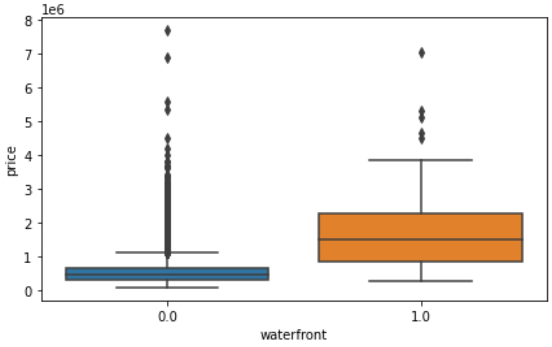
Out[153…

	waterfront
0.0	21451
1.0	146

In [154…
```
waterfront_1 = df.loc[df.waterfront==1,'price'].mean()
waterfront_0 = df.loc[df.waterfront==0,'price'].mean()
print(f"{'the waterfront house prices are higher by'} {(waterfront_1/waterfront_0)*100:.2f}%")
```

```
the waterfront house prices are higher by 322.61%
```

In [155…
```
fig, ax1 = plt.subplots(ncols=1, sharey=True)
fig.set_size_inches(7,4)
sns.axes_style()
waterfront_bp = sns.boxplot(x=df['waterfront'], y=df['price'], ax=ax1)
```



count the unique values of the 'floors' & 'waterfront' to determine the houses type.

In [156…
```
df['floors'].value_counts().to_frame()
```

Out[156…

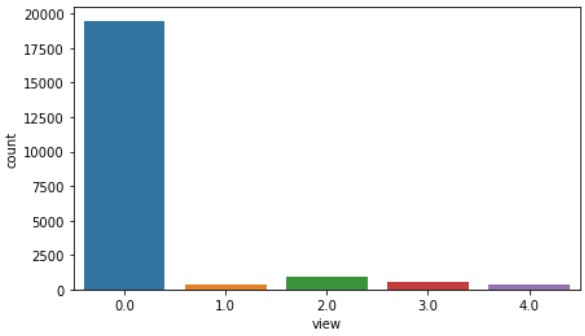
	floors
1.0	10673
2.0	8235
1.5	1910
3.0	611
2.5	161
3.5	7

VIEW

the variable "view" describes the times that the house has been seen, however we have noticed that 19485 of the data are quale to zero.

In [157…
```
fig, ax1 = plt.subplots(ncols=1, sharey=True)
fig.set_size_inches(7,4)
sns.axes_style()
sns.countplot(x=df['view'], ax=ax1)
```

Out[157… `<AxesSubplot:xlabel='view', ylabel='count'>`



In [158…
```
print("view: zeros value: {0:>10}".format((df.view==0).sum()))
print("renovated: zeros value: {0:>5}".format( (df.renovated==0).sum() ))
```

```
view: zeros value:      19485
renovated: zeros value: 20853
```

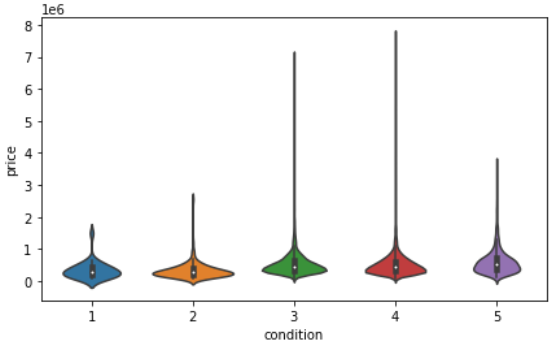
we're goin to exclude the 'view' column since it contains almost all null values.

CONDITION

Relative to age and grade. Coded 1-5.
1 = Poor- Worn out.
2 = Fair- Badly worn.
3 = Average
4 = Good
5= Very Good}

In [183…
```
fig, ax1 = plt.subplots(ncols=1, sharey=True)
fig.set_size_inches(7,4)
sns.axes_style()
sns.violinplot(x=df['condition'], y=df['price'], ax=ax1)
```

Out[183… `<AxesSubplot:xlabel='condition', ylabel='price'>`

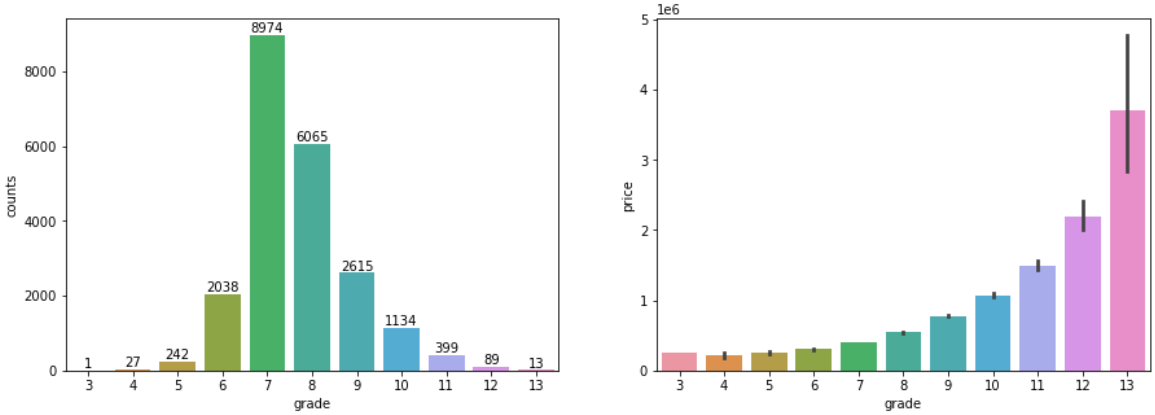


GRADE

Represents the construction quality of improvements. Grades run from grade 1 to 13.: King County link

In [161…

```
fig, (ax1, ax2) = plt.subplots(ncols=2, sharey=False)
fig.set_size_inches(15,5)
sns.axes_style()
df_by_grade = df.groupby('grade').size().reset_index(name='counts')
g = sns.barplot(x='grade', y='counts', data=df_by_grade, linewidth=3, errcolor='gray', ax=ax1)
for index, row in df_by_grade.iterrows():
    g.text(row.name, row.counts, round(row.counts, 2), color='black', va='bottom',  ha="center", fontsize=10)
g2=sns.barplot(x='grade', y='price', data=df, ax=ax2)
```



let's compare the lot size with the living area and the percentage of living aerea compared to the lot size

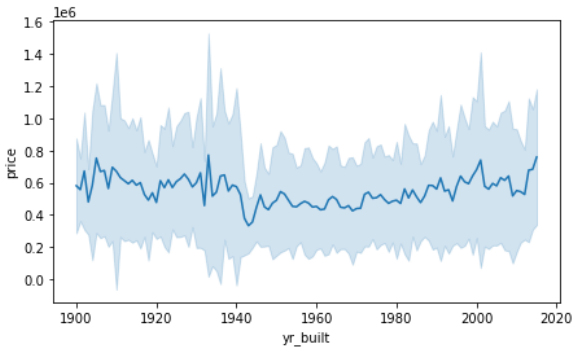
YR_BUILT

Let's count the houses based on the year of construction, it seems to have a strong correlation/p>

In [182…

```
fig, ax1 = plt.subplots(nrows=1)
fig.set_size_inches(7,4)
sns.axes_style()
sns.lineplot(x="yr_built", y="price", ci='sd', markers=True, data=df)
```

Out[182… <AxesSubplot:xlabel='yr_built', ylabel='price'>

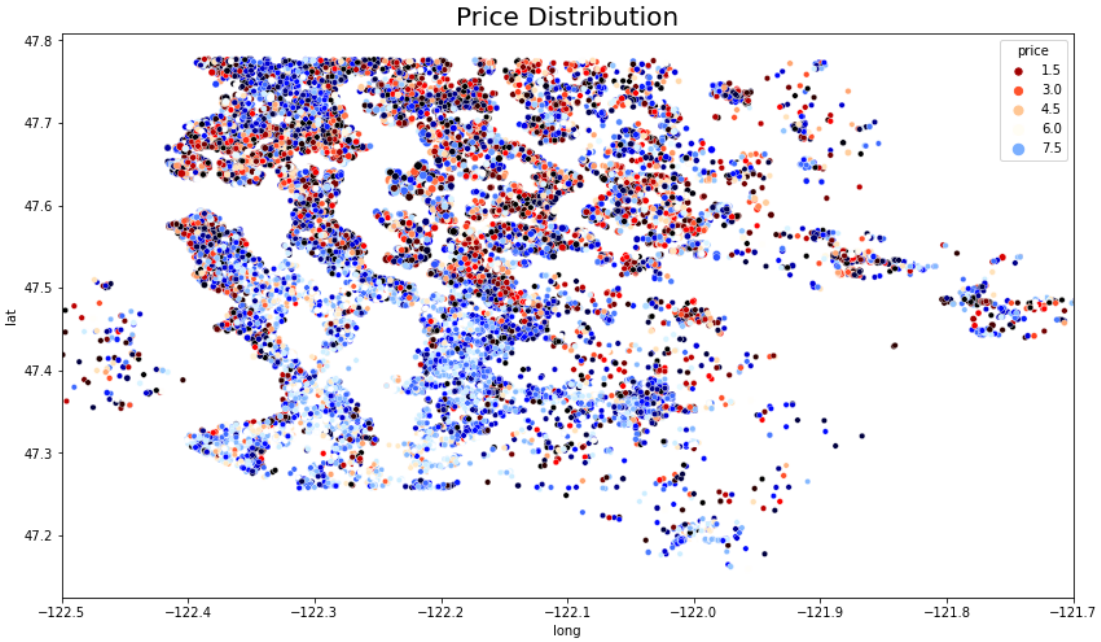


PRICE DISTRIBUTION

simple Seaborn scatterplot shows homes by their latitude and longitude, with price set as the hue.

In [163…

```
plt.figure(figsize=(14,8))
sns.scatterplot(x=df.long, y=df.lat, hue=df.price, size=df.price, palette='flag')
plt.xlim(-122.5, -121.7)
plt.title('Price Distribution', fontdict={'fontsize': 20})
plt.show()
```



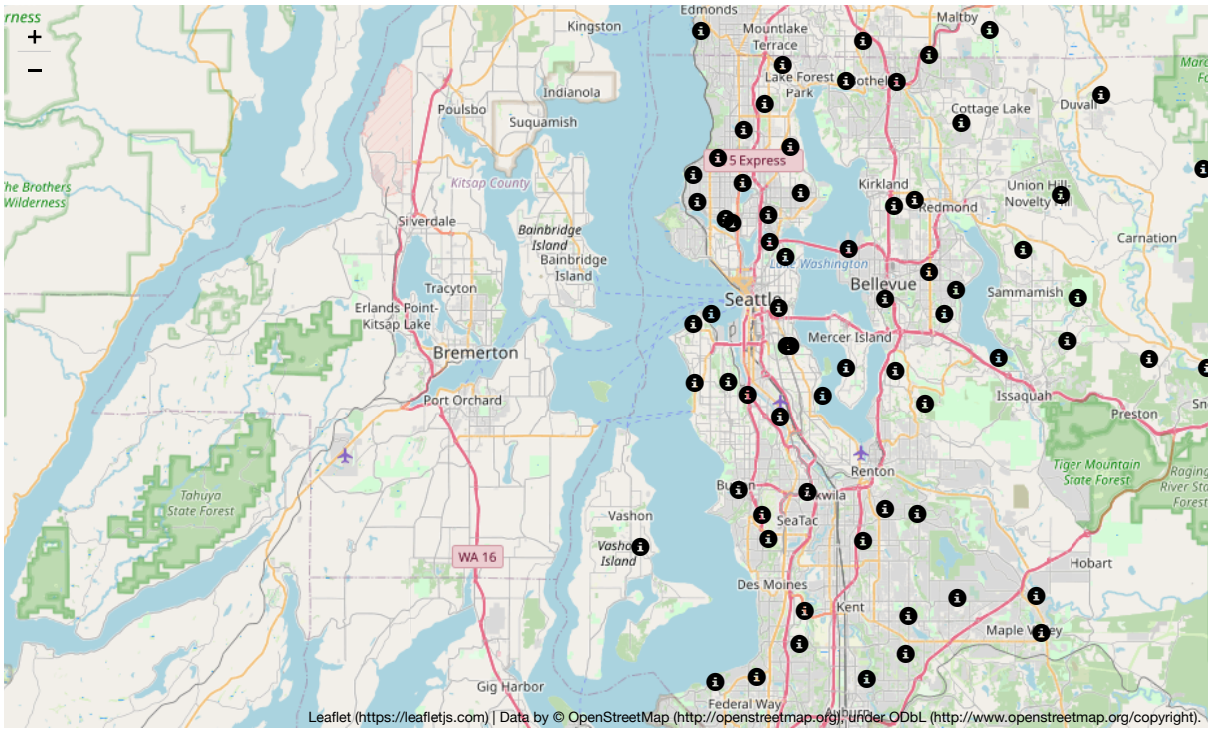
TOP!

let's visualize the top 70 zip codes with with an interactive geographic map

In [165…

```
dfzip = df.drop_duplicates('zipcode', keep='first')
centerlat = (dfzip['lat'].max() + dfzip['lat'].min()) / 2
centerlong = (dfzip['long'].max() + dfzip['long'].min()) / 2
map = folium.Map(location=[centerlat, centerlong], zoom_start=9)
#icon = folium.Icon(color='blue', icon_color='white', icon='info-sign',angle=0)

for i in range(dfzip.shape[0]):
    pup = '${:,}'.format(dfzip.iloc[i]['price'])
    if dfzip.iloc[0]['waterfront'] == 1:
        ic = folium.Icon(color='red', icon_color='red')
    else:
        ic = folium.Icon(color='blue', icon_color='white')
    folium.Marker([dfzip.iloc[i]['lat'], dfzip.iloc[i]['long']], icon=ic, popup=pup, radius=3).add_to(map)
# map.save('top_70_zipcode.html')
map
```

Out[165…



FEATURE ENGINEERING

exclude variables that are not relevant to the analysis, we're going to use the scatterplot matrix to evaluate the correlation and the Multicollinearity.

In [166…

```
var_excluded.update(('lat','long', 'zipcode', 'bathrooms','date'))
```

In [167…

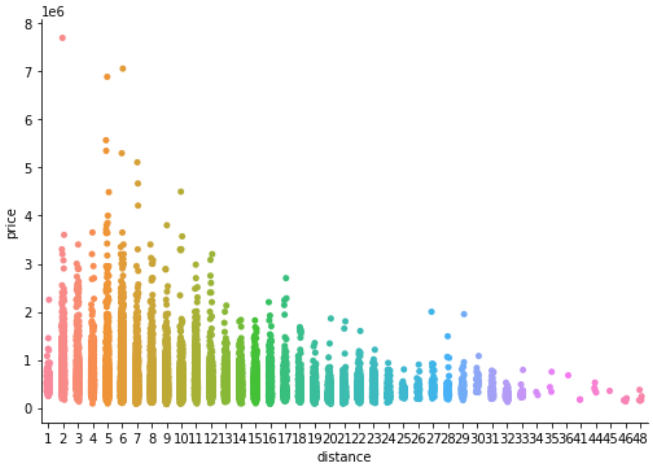
```
var_predictors = set(df.columns)-var_excluded
```

LONG & LAT

From previous plot we notice that the price seems to go down as houses are further from the center, It would be appropriate to create a new feature that represents distance from the center of King County. for feature we used the geopy library, which essentially calculates the distance in miles from specific latitude and longitude points..

In [168…
```
lat_long=[(x,y) for x,y in zip(df['lat'], df['long'])]
kc = (47.6062, -122.3321) # king county usa downtown lat long
miles = [int(round(distance.distance(i, kc).miles,0)) for i in lat_long ]
df['distance'] = miles
var_predictors.add(('distance'))
```

In [251…
```
fig.set_size_inches(12, 10)
distance = sns.catplot(x='distance', y='price', data=df, height=5, aspect=1.4)
```

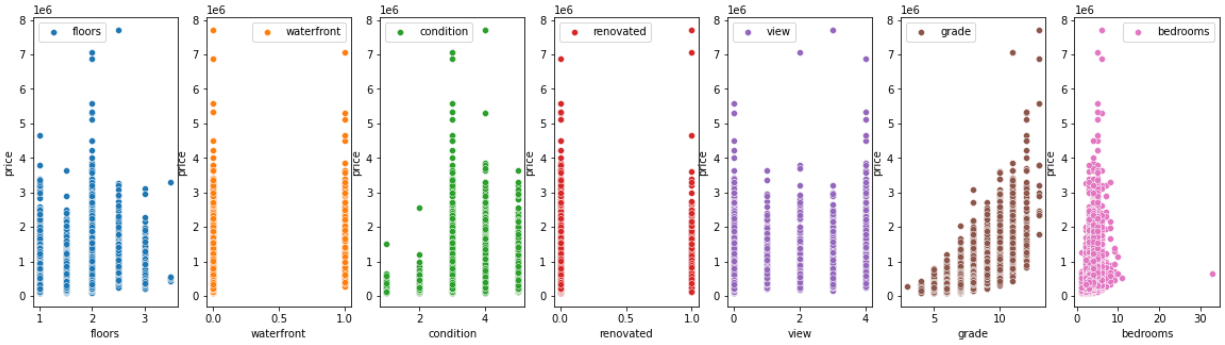


Categorical Variables

once we have identified the category variables, let's take a final visual look at the variables for further confirmation.

In [252…
```
var_categories =  {'condition', 'waterfront', 'floors', 'renovated', 'bedrooms' , 'view', 'grade'}
```

In [253…
```
palette = itertools.cycle(sns.color_palette())
fig, axes = plt.subplots(nrows=1, ncols=len(var_categories), figsize=(20,5))
for xcol, ax in zip(var_categories, axes):
    sns.scatterplot(x=df[xcol], y=df['price'] , ax=ax, label=xcol, color=next(palette))
```



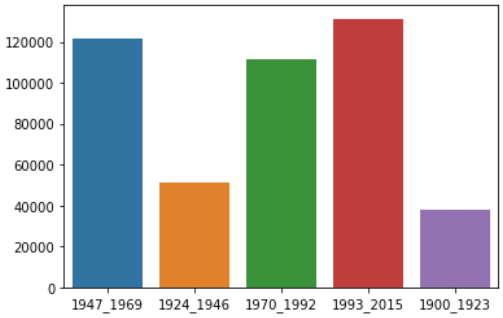
as we see from the graph it seems that variables are cattegoric type so we're going to use the pandas CUT method to segment these values into groups of "bins".

In [254…
```
yr_built_bins = [1900,1923,1946,1969,1992,2015]
yr_built_labels = ['1900_1923','1924_1946','1947_1969','1970_1992','1993_2015']
yr_built_cat = pd.cut(x=df['yr_built'], bins=yr_built_bins, labels=yr_built_labels, include_lowest=True)
df['yr_built'] = yr_built_cat.cat.as_unordered()
```

In [255…
```
var_categories.add(('yr_built'))
```

In [256…
```
yr_built_unique = df.yr_built.unique()
n_construction  = [df[df.yr_built == j].size for j in df.yr_built.unique()]
sns.barplot(x=yr_built_unique, y=n_construction)
```

Out[256… <AxesSubplot:>



Convert categorical features into Int.' dtype

In [40]:
```
df.condition  = df.condition.astype(int)
df.waterfront = df.waterfront.astype(int)
df.floors     = df.floors.astype(int)
df.renovated  = df.renovated.astype(int)
df.grade      = df.grade.astype(int)
df.view       = df.view.astype(int)
```

ONE-HOT-ENCODING

In [41]:
```
#create a dummy data by removing redundant columns when using get_dummies
df_categories = pd.DataFrame()

for cat in var_categories:
    df_categories[cat]=df[cat].astype('category')
    df_dummy = pd.get_dummies(df_categories[cat], prefix=cat, drop_first=True)
    df_categories = df_categories.join(df_dummy)
    df_categories.drop(labels=cat, axis=1, inplace=True)
```

using the USZIPCODE library we're going to decode the zip code in order to obtain a list of the corresponding neighborhoods. it's more efficient decoding by zipcode rather than coordinates since the unique values are only 70. we noticed that the 70 zip codes present in the dataframe refers to 24 neighborhoods in other words, the 21597 houses are all concentrated in 24 urban areas.

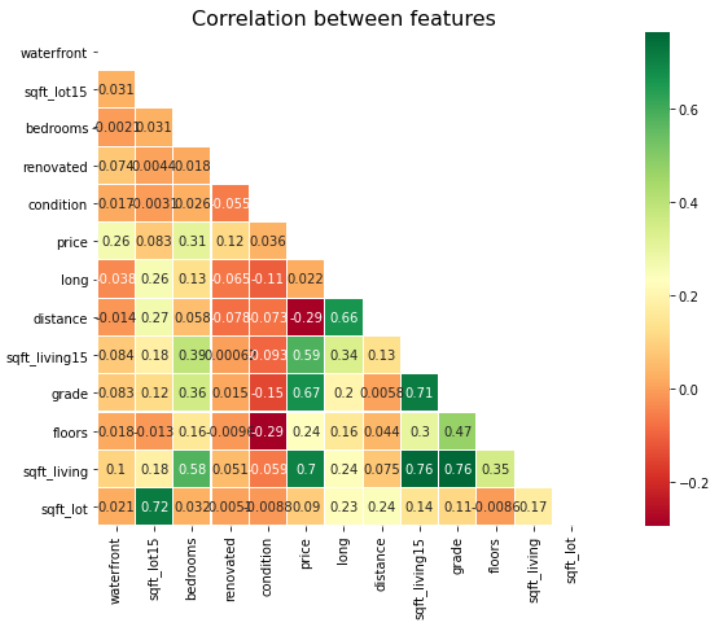
In [42]:
```
search = SearchEngine()
neighbourhoods = [search.by_zipcode(c).city for c in df.zipcode]
df['neighbourhoods'] = neighbourhoods

df_neighbourhoods = pd.DataFrame()
df_neighbourhoods = df['neighbourhoods'].astype('category')
df_neighbourhoods = pd.get_dummies(df_neighbourhoods, drop_first=True)
```

In [43]:
```
var_categories.add(('neighbourhoods'))
df_categories = df_categories.join(df_neighbourhoods)
```

CORRELATION MATRIX

In [44]:
```
cor_features = set(df.columns)-set(('zipcode', 'view', 'sqft_basement','sqft_above','lon','lat','bathrooms'))
corr = df[cor_features].corr(method='pearson')
# mask = corr[(corr !=1.000)]
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True
plt.figure(figsize=(15, 7))
sns.heatmap(corr, mask=mask, annot=True, linewidth=.1, cmap="RdYlGn", square=True)
plt.title('Correlation between features', fontdict={'fontsize': 16})
```

Out[44]:
```
Text(0.5, 1.0, 'Correlation between features')
```



TOP

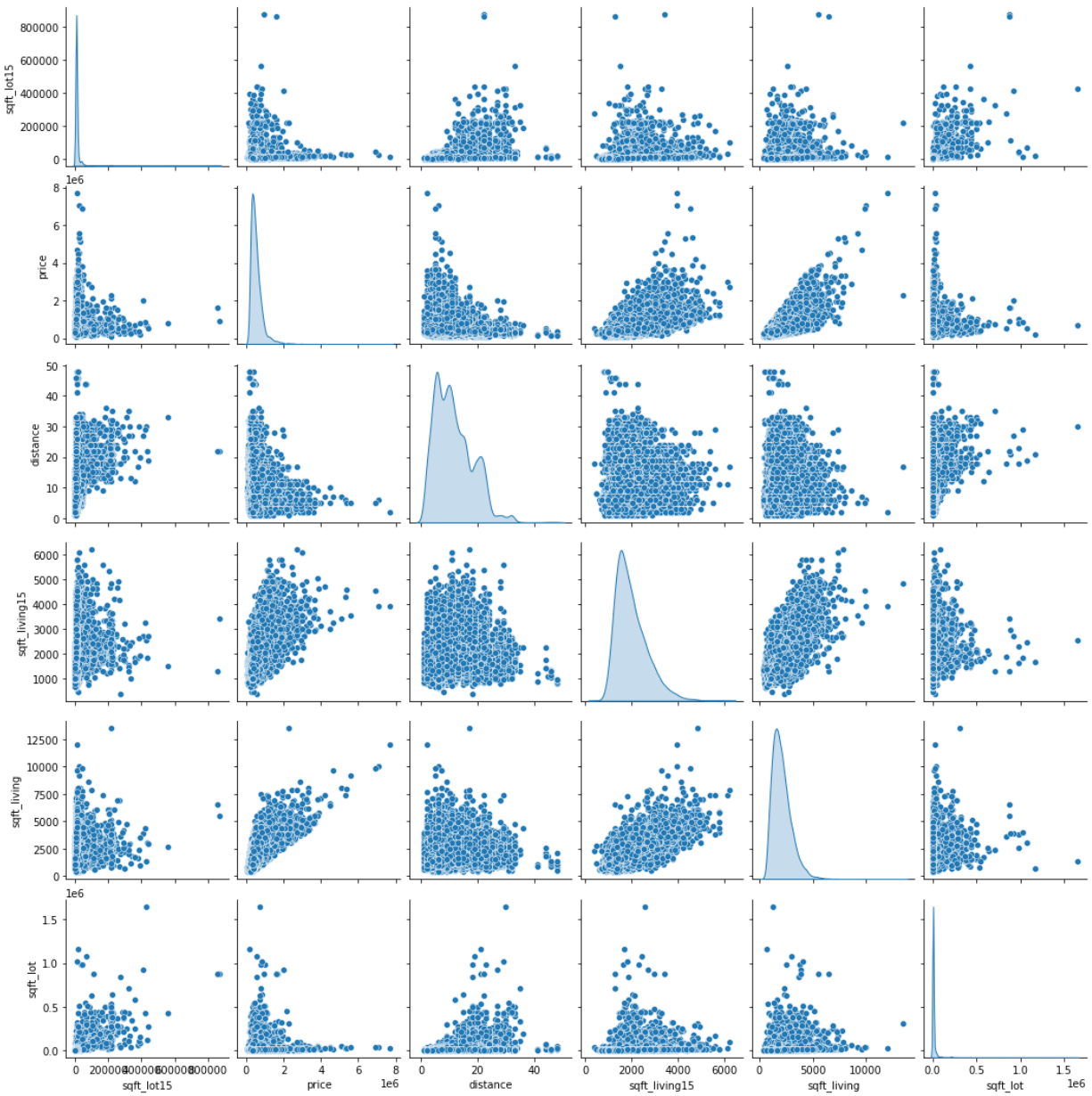
CHECK FEATURES DISTRIBUTION

We would like to investigate the relationship between our target variable price and the continuous feature variables in our dataset. We will make use of Seaborn's jointplot to simultaneously inspect linearity and distributions.

In [45]:
```
df_features = pd.DataFrame(df[var_predictors-var_categories])
```

In [46]:
```
sns.pairplot(df_features, diag_kind="kde")
```

Out[46]:
```
<seaborn.axisgrid.PairGrid at 0x7febd83de370>
```



Min Max SCALER

Using the standard function from sklearn to scale the indepedent variables, so that all the features hold a standard weight towards the depedent variable.

In [47]:

```
x_scaler    = MinMaxScaler()
y_scaler    = MinMaxScaler()

y           = y_scaler.fit_transform(df.price.values.reshape(-1,1))
x_scaled    = x_scaler.fit_transform(df_features.drop(labels=['price'], axis=1))
df_features = pd.DataFrame(x_scaled, columns=df_features.columns.difference(['price']))
df_features.head()
```

Out[47]:

	distance	sqft_living	sqft_living15	sqft_lot	sqft_lot15
0	0.005742	0.127660	0.161934	0.061503	0.003108
1	0.008027	0.148936	0.222165	0.167046	0.004072
2	0.008513	0.191489	0.399415	0.030372	0.005743
3	0.004996	0.127660	0.165376	0.120729	0.002714
4	0.007871	0.255319	0.241094	0.099468	0.004579

TOP

MODELING

as a starting point, we are trying to establish the following models, each responding to different criteria.

Model A

our first model aims to establish the correlation between continuous features to obtain as a first result a moderate value of coefficient of determination R2.

In [48]:
```
x           = df_features
x           = sm.add_constant(x)

model_a     = sm.OLS(y, x).fit()
model_a_pred = model_a.predict(x)
print(str(model_a.summary()))
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.626
Model:                            OLS   Adj. R-squared:                  0.626
Method:                 Least Squares   F-statistic:                     7221.
Date:                Tue, 16 Mar 2021   Prob (F-statistic):               0.00
Time:                        00:05:52   Log-Likelihood:                 45461.
No. Observations:               21597   AIC:                        -9.091e+04
Df Residuals:                   21591   BIC:                        -9.086e+04
Df Model:                           5
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const            0.0147      0.001     25.662      0.000       0.014       0.016
distance         0.0118      0.009      1.259      0.208      -0.007       0.030
sqft_living     -0.1259      0.001    -84.360      0.000      -0.129      -0.123
sqft_living15    0.0783      0.003     29.837      0.000       0.073       0.083
sqft_lot         0.3975      0.004     89.697      0.000       0.389       0.406
sqft_lot15       0.0892      0.012      7.720      0.000       0.067       0.112
==============================================================================
Omnibus:                    18450.351   Durbin-Watson:                   1.993
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          1496816.456
Skew:                           3.675   Prob(JB):                         0.00
Kurtosis:                      43.117   Cond. No.                         73.2
==============================================================================

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

Model A Scores

In [49]:
```
model_a_mae, model_a_mse, model_a_rmse = func.metrics(y, model_a_pred)
```

```
R-squared                0.626
Mean Absolute Error      0.019 $141,677.74
Mean Squared Error       0.001
Root Mean Squared Error  0.029 $224,726.54
```

Using Variance Inflation Factor (VIF) we measure of collinearity among predictor variables within Model A.

In [50]:
```
func.vif(x)
```

Out[50]:
```
const            8.185292
distance         2.129808
sqft_living      1.097942
sqft_living15    2.379905
sqft_lot         2.370869
sqft_lot15       2.088376
dtype: float64
```

MODEL B

the second model would be more accurate and complex given the numerous categorical variables. our goal is to obtain a better performing model with a high R2 value while maintaining a significant P-value below a threshold of 0.05

In [51]:
```
x = pd.concat([x, df_categories], axis= 1)
```

In [52]:
```
x             = sm.add_constant(x)
model_b       = sm.OLS(y,x).fit()
model_b_pred  = model_b.predict(x)

print(str(model_b.summary()))
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.785
Model:                            OLS   Adj. R-squared:                  0.785
Method:                 Least Squares   F-statistic:                     1211.
Date:                Tue, 16 Mar 2021   Prob (F-statistic):               0.00
Time:                        00:06:03   Log-Likelihood:                 51459.
No. Observations:               21597   AIC:                        -1.028e+05
Df Residuals:                   21531   BIC:                        -1.023e+05
Df Model:                          65
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const            0.0604      0.023      2.642      0.008       0.016       0.105
distance        -0.0161      0.007     -2.198      0.028      -0.030      -0.002
sqft_living     -0.1460      0.003    -47.654      0.000      -0.152      -0.140
sqft_living15    0.0215      0.002      9.324      0.000       0.017       0.026
sqft_lot         0.2673      0.005     57.172      0.000       0.258       0.276
sqft_lot15       0.0529      0.009      5.990      0.000       0.036       0.070
waterfront_1     0.0743      0.002     32.325      0.000       0.070       0.079
bedrooms_2       0.0018      0.002      1.051      0.293      -0.002       0.005
bedrooms_3       0.0013      0.002      0.765      0.444      -0.002       0.005
bedrooms_4      -0.0025      0.002     -1.425      0.154      -0.006       0.001
bedrooms_5      -0.0038      0.002     -2.106      0.035      -0.007      -0.000
bedrooms_6      -0.0052      0.002     -2.345      0.019      -0.010      -0.001
bedrooms_7      -0.0118      0.004     -2.893      0.004      -0.020      -0.004
```

```
bedrooms_8             0.0167      0.006      2.570      0.010       0.004       0.029
bedrooms_9            -0.0118      0.009     -1.262      0.207      -0.030       0.007
bedrooms_10           -0.0187      0.013     -1.435      0.151      -0.044       0.007
bedrooms_11           -0.0391      0.022     -1.743      0.081      -0.083       0.005
bedrooms_33            0.0194      0.022      0.863      0.388      -0.025       0.063
yr_built_1924_1946    -0.0017      0.001     -2.391      0.017      -0.003      -0.000
yr_built_1947_1969    -0.0092      0.001    -13.663      0.000      -0.011      -0.008
yr_built_1970_1992    -0.0123      0.001    -16.128      0.000      -0.014      -0.011
yr_built_1993_2015    -0.0102      0.001    -12.445      0.000      -0.012      -0.009
view_1                 0.0100      0.001      7.932      0.000       0.008       0.012
view_2                 0.0061      0.001      7.953      0.000       0.005       0.008
view_3                 0.0151      0.001     14.392      0.000       0.013       0.017
view_4                 0.0337      0.002     21.261      0.000       0.031       0.037
renovated_1            0.0089      0.001     10.025      0.000       0.007       0.011
condition_2            0.0096      0.005      2.118      0.034       0.001       0.018
condition_3            0.0117      0.004      2.786      0.005       0.003       0.020
condition_4            0.0158      0.004      3.766      0.000       0.008       0.024
condition_5            0.0216      0.004      5.122      0.000       0.013       0.030
grade_4               -0.0160      0.023     -0.699      0.485      -0.061       0.029
grade_5               -0.0200      0.023     -0.889      0.374      -0.064       0.024
grade_6               -0.0209      0.022     -0.928      0.353      -0.065       0.023
grade_7               -0.0139      0.022     -0.620      0.535      -0.058       0.030
grade_8               -0.0078      0.022     -0.346      0.729      -0.052       0.036
grade_9                0.0042      0.023      0.185      0.853      -0.040       0.048
grade_10               0.0228      0.023      1.010      0.312      -0.021       0.067
grade_11               0.0506      0.023      2.245      0.025       0.006       0.095
grade_12               0.1043      0.023      4.599      0.000       0.060       0.149
grade_13               0.2547      0.023     10.873      0.000       0.209       0.301
floors_2            7.831e-05      0.000      0.168      0.866      -0.001       0.001
floors_3               0.0017      0.001      1.547      0.122      -0.000       0.004
Bellevue               0.0070      0.001      5.417      0.000       0.004       0.010
Black Diamond          0.0179      0.002      7.552      0.000       0.013       0.023
Bothell               -0.0118      0.002     -6.277      0.000      -0.015      -0.008
Carnation              0.0187      0.002      8.585      0.000       0.014       0.023
Duvall                 0.0079      0.002      4.378      0.000       0.004       0.011
Enumclaw               0.0316      0.002     17.650      0.000       0.028       0.035
Fall City              0.0171      0.003      6.474      0.000       0.012       0.022
Federal Way           -0.0049      0.001     -4.444      0.000      -0.007      -0.003
Issaquah               0.0037      0.001      3.079      0.002       0.001       0.006
Kenmore               -0.0153      0.002     -9.168      0.000      -0.019      -0.012
Kent                  -0.0092      0.001     -9.048      0.000      -0.011      -0.007
Kirkland              -0.0020      0.001     -1.525      0.127      -0.005       0.001
Maple Valley           0.0055      0.001      4.613      0.000       0.003       0.008
Medina                 0.1060      0.003     30.628      0.000       0.099       0.113
Mercer Island          0.0153      0.002      8.206      0.000       0.012       0.019
North Bend             0.0350      0.002     20.081      0.000       0.032       0.038
Redmond                0.0027      0.001      2.248      0.025       0.000       0.005
Renton                -0.0209      0.001    -18.724      0.000      -0.023      -0.019
Sammamish             -0.0023      0.001     -1.882      0.060      -0.005    9.52e-05
Seattle               -0.0179      0.001    -14.389      0.000      -0.020      -0.015
Snoqualmie             0.0147      0.002      9.809      0.000       0.012       0.018
Vashon                -0.0181      0.002     -7.804      0.000      -0.023      -0.014
Woodinville           -0.0035      0.001     -2.557      0.011      -0.006      -0.001
==============================================================================
Omnibus:                    15658.426   Durbin-Watson:                   1.993
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          1231493.060
Skew:                           2.809   Prob(JB):                         0.00
Kurtosis:                      39.564   Cond. No.                          834.
==============================================================================

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

Model b Scores #1

In [53]:
```
model_b_mae, model_b_mse, model_b_rmse = func.metrics(y, model_b_pred)
```

```
R-squared                0.785
Mean Absolute Error      0.014 $106,303.35
Mean Squared Error       0.000
Root Mean Squared Error  0.022 $170,235.73
```

For an initial fit the model looks good obtaining a R-Squared of 0.785 and as well as Adj. R-squared of 0.785. The contribution is attributed to the categorical variabers that make the model more stable and positive.

Surprisingly the P-Value of the continuous variables is lower than the threshold of 0.05. Furthermore, some categories that exceed the cut-off threshold, so we begin to discard these variables while maintaining those with a value of less than .05.

In [54]:
```
stepwise_result = func.stepwise_selection(x, y, verbose=False)
print('resulting features:')
print(stepwise_result)
```

```
resulting features:
['view_4', 'grade_10', 'grade_11', 'sqft_living', 'sqft_living15', 'const', 'sqft_lot', 'Medina', 'grade_12', 'grade_13',
'waterfront_1', 'Renton', 'grade_9', 'Bellevue', 'Mercer Island', 'Enumclaw', 'North Bend', 'Seattle', 'condition_5', 're
novated_1', 'Kent', 'view_3', 'grade_8', 'condition_4', 'Snoqualmie', 'Kenmore', 'Carnation', 'yr_built_1924_1946', 'grad
e_6', 'yr_built_1970_1992', 'Black Diamond', 'Fall City', 'view_2', 'yr_built_1947_1969', 'yr_built_1993_2015', 'bedrooms
_3', 'view_1', 'Vashon', 'bedrooms_2', 'Bothell', 'Federal Way', 'sqft_lot15', 'Maple Valley', 'Duvall', 'Issaquah', 'Red
mond', 'grade_5', 'bedrooms_8']
```

In [55]:
```
x           = x[stepwise_result]
x           = sm.add_constant(x)

model_c     = sm.OLS(y,x).fit()
model_c_pred = model_c.predict(x)
print(str(model_c.summary()))
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.785
Model:                            OLS   Adj. R-squared:                  0.784
Method:                 Least Squares   F-statistic:                     1672.
Date:                Tue, 16 Mar 2021   Prob (F-statistic):               0.00
Time:                        00:06:37   Log-Likelihood:                 51436.
```

```
No. Observations:               21597   AIC:                          -1.028e+05
Df Residuals:                   21549   BIC:                          -1.024e+05
Df Model:                          47
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
view_4           0.0339      0.002     21.407      0.000       0.031       0.037
grade_10         0.0372      0.001     39.510      0.000       0.035       0.039
grade_11         0.0654      0.001     45.263      0.000       0.063       0.068
sqft_living     -0.1434      0.002    -59.275      0.000      -0.148      -0.139
sqft_living15    0.0207      0.002      9.131      0.000       0.016       0.025
const            0.0538      0.001     44.602      0.000       0.051       0.056
sqft_lot         0.2609      0.004     59.552      0.000       0.252       0.269
Medina           0.1084      0.003     33.226      0.000       0.102       0.115
grade_12         0.1194      0.003     44.619      0.000       0.114       0.125
grade_13         0.2703      0.006     41.806      0.000       0.258       0.283
waterfront_1     0.0743      0.002     32.383      0.000       0.070       0.079
Renton          -0.0191      0.001    -26.836      0.000      -0.021      -0.018
grade_9          0.0185      0.001     28.269      0.000       0.017       0.020
Bellevue         0.0090      0.001     11.215      0.000       0.007       0.011
Mercer Island    0.0175      0.001     11.770      0.000       0.015       0.020
Enumclaw         0.0322      0.002     18.182      0.000       0.029       0.036
North Bend       0.0359      0.002     20.924      0.000       0.033       0.039
Seattle         -0.0157      0.001    -23.292      0.000      -0.017      -0.014
condition_5      0.0102      0.001     16.479      0.000       0.009       0.011
renovated_1      0.0090      0.001     10.171      0.000       0.007       0.011
Kent            -0.0077      0.001     -9.831      0.000      -0.009      -0.006
view_3           0.0154      0.001     14.703      0.000       0.013       0.017
grade_8          0.0064      0.000     14.976      0.000       0.006       0.007
condition_4      0.0043      0.000     10.761      0.000       0.003       0.005
Snoqualmie       0.0160      0.001     11.297      0.000       0.013       0.019
Kenmore         -0.0134      0.001     -9.512      0.000      -0.016      -0.011
Carnation        0.0196      0.002      9.239      0.000       0.015       0.024
yr_built_1924_1946 -0.0017   0.001     -2.469      0.014      -0.003      -0.000
grade_6         -0.0071      0.001    -11.842      0.000      -0.008      -0.006
yr_built_1970_1992 -0.0123   0.001    -16.668      0.000      -0.014      -0.011
Black Diamond    0.0188      0.002      8.062      0.000       0.014       0.023
Fall City        0.0181      0.003      7.023      0.000       0.013       0.023
view_2           0.0063      0.001      8.155      0.000       0.005       0.008
yr_built_1947_1969 -0.0094   0.001    -14.248      0.000      -0.011      -0.008
yr_built_1993_2015 -0.0098   0.001    -12.960      0.000      -0.011      -0.008
bedrooms_3       0.0039      0.000     10.326      0.000       0.003       0.005
view_1           0.0101      0.001      8.037      0.000       0.008       0.013
Vashon          -0.0170      0.002     -7.831      0.000      -0.021      -0.013
bedrooms_2       0.0041      0.001      6.954      0.000       0.003       0.005
Bothell         -0.0099      0.002     -5.977      0.000      -0.013      -0.007
Federal Way     -0.0035      0.001     -3.706      0.000      -0.005      -0.002
sqft_lot15       0.0397      0.007      5.957      0.000       0.027       0.053
Maple Valley     0.0068      0.001      6.279      0.000       0.005       0.009
Duvall           0.0091      0.002      5.344      0.000       0.006       0.012
Issaquah         0.0053      0.001      5.698      0.000       0.003       0.007
Redmond          0.0045      0.001      5.382      0.000       0.003       0.006
grade_5         -0.0064      0.002     -4.238      0.000      -0.009      -0.003
bedrooms_8       0.0204      0.006      3.279      0.001       0.008       0.033
==============================================================================
Omnibus:                    15686.240   Durbin-Watson:                   1.992
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          1246529.118
Skew:                           2.813   Prob(JB):                         0.00
Kurtosis:                      39.791   Cond. No.                         63.3
==============================================================================

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

In [56]:
```
model_b_mae, model_b_mse, model_b_rmse = func.metrics(y, model_b_pred)
```

```
R-squared                0.785
Mean Absolute Error      0.014 $106,303.35
Mean Squared Error       0.000
Root Mean Squared Error  0.022 $170,235.73
```

SKLEARN MODEL

Regression Model Validation

In [57]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=.3, random_state=7)

linreg = LinearRegression()
linreg.fit(x_train, y_train)

#Calculating predictions on the train set, and test set
y_hat_train = linreg.predict(x_train)
y_hat_test  = linreg.predict(x_test)

#Calculating your residuals
train_residuals = y_hat_train - y_train
test_residuals = y_hat_test - y_test

#Calculating the Mean Squared Error
train_mse = mean_squared_error(y_train, y_hat_train)
test_mse  = mean_squared_error(y_test, y_hat_test)

print("\033[94m"f"{'R^2 Score':<30}{round(linreg.score(x, y),2):>5}")
print(f"{'Train Mean Squarred Error':<30} {train_mse:>5}")
print(f"{'Test Mean Squarred Error':<30} {test_mse:>5}")
```

```
R^2 Score                      0.78
Train Mean Squarred Error      0.0005325592772809952
Test Mean Squarred Error       0.0004267273157622785
```

In [58]:
```
model_b_mae, model_b_mse, model_b_rmse = func.metrics(y_test, y_hat_test)
```

```
R-squared                0.801
Mean Absolute Error      0.014 $104,132.96
Mean Squared Error       0.000
Root Mean Squared Error  0.021 $157,450.54
```

CROSS VALIDATION SCORE

In [59]:
```
kf = KFold(n_splits=10, shuffle=True, random_state=74)

msw    = cross_val_score(linreg, x_test, y_test, scoring='neg_mean_squared_error', cv=kf, n_jobs=1)
scores  = cross_val_score(linreg, x, y, scoring='r2', cv=kf, n_jobs=1)
```

In [60]:
```
model_b_mae, model_b_mse, model_b_rmse = func.metrics(y_test, y_hat_test)
```

```
R-squared                0.801
Mean Absolute Error      0.014 $104,132.96
Mean Squared Error       0.000
Root Mean Squared Error  0.021 $157,450.54
```