JosephDenney / **KingCountyRealEstate**

<> Code   Issues   Pull requests   Actions   Projects   Wiki   Security

master ▾

KingCountyRealEstate / Mod2ProjectEDA.ipynb

JosephDenney sunday final nb   History

1 contributor

<>   Raw   Blame

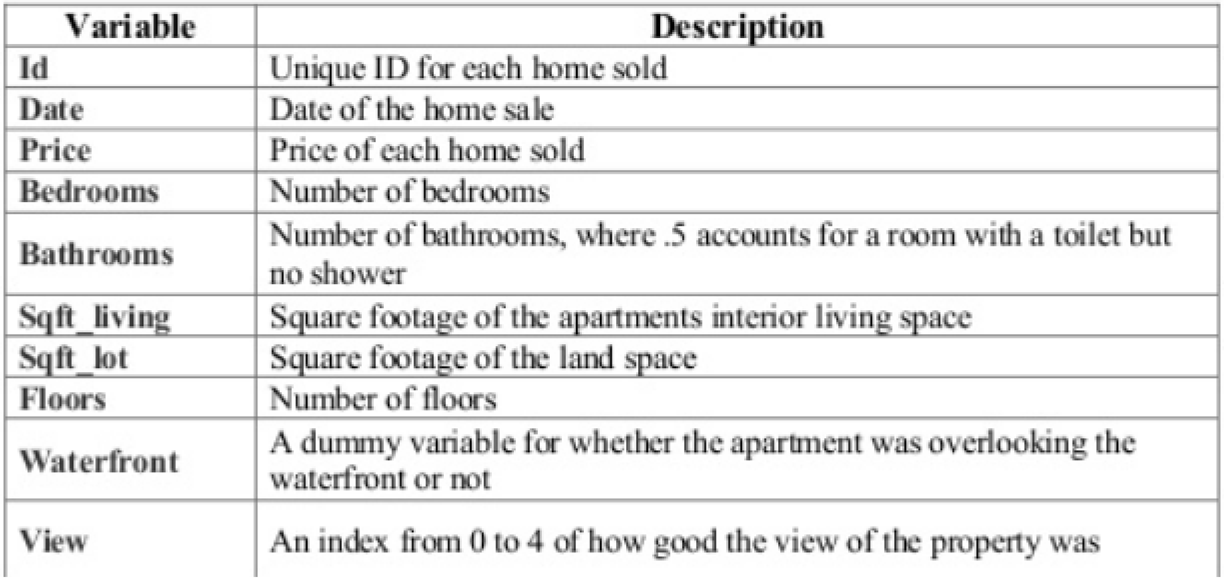
```
987 lines (987 sloc)   437 KB
```

Housing Analysis in King County, Washington

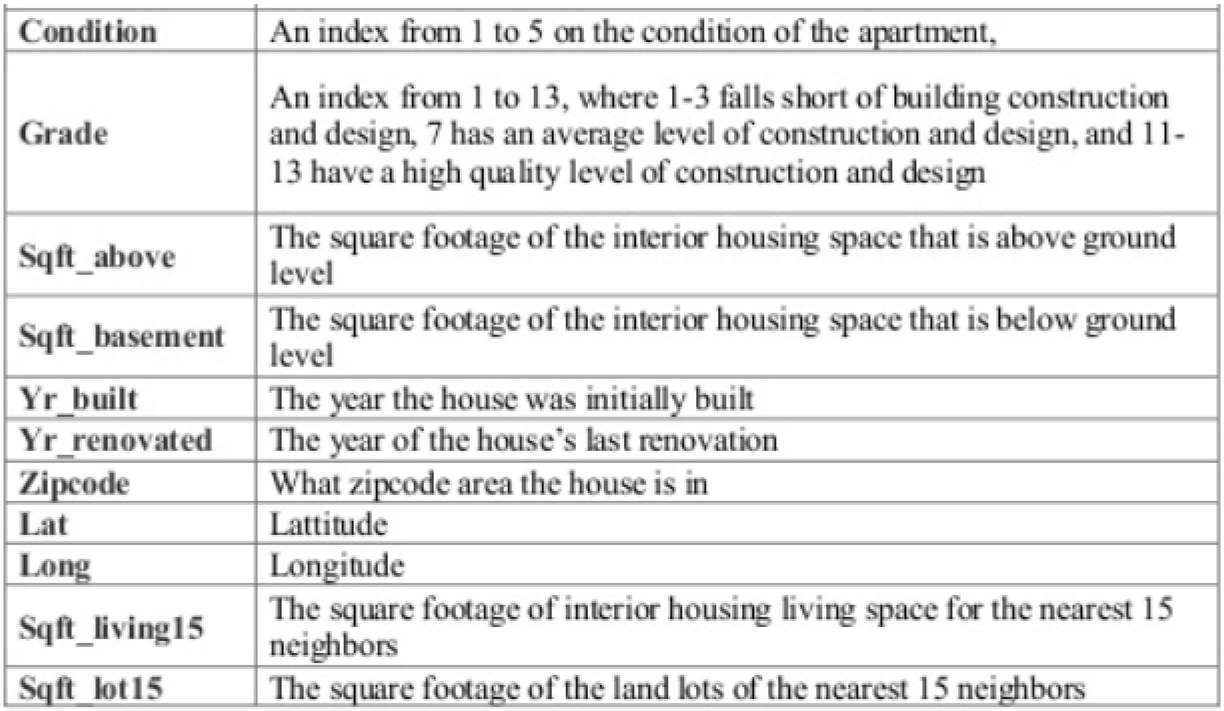
EDA, data cleaning, feature engineering notebook

In [24]:
```
import warnings

warnings.filterwarnings('ignore')

import pandas as pd
import numpy as np
import csv

import scipy.stats as scs
import statsmodels.api as sm
import statsmodels.formula.api as sms
import scipy.stats as stats
from statsFunctions import check_model as sf
from pltfunctions import hist_kde_plots
from haversine import haversine
from math import sqrt

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 sklearn.metrics as metrics

import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv(r'data\kc_house_data.csv')
```

Data description as follows



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	Lattitude
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

In [25]:

```
df.info() # a good initial picture of the data
# initial thoughts for necessary data and a regression upon seeing the
 data -
# price is y target
# 1) Unique identifiers (column= id) are unnecessary for a regression
# 2) lat and long wont be needed if include zipcode and will just be no
ise (decided to keep lat/long over zip)
# 3) square footage and home quality is likely to change based on locat
ion
# 4) anticipate autocorrelation between location and home features - dr
opping zipcode and keeping lat long
# 5) date - as time goes on, home prices are likely to go up - if the s
ales are all within a close time frame then we can ignore date
# 6) nearest 15 neighbors data - will autocorrelate with homes nearby,
 remove data
# 7) already have square footage of home, remove above and below ground
sqft
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   id             21597 non-null  int64  
 1   date           21597 non-null  object 
 2   price          21597 non-null  float64
 3   bedrooms       21597 non-null  int64  
 4   bathrooms      21597 non-null  float64
 5   sqft_living    21597 non-null  int64  
 6   sqft_lot       21597 non-null  int64  
 7   floors         21597 non-null  float64
 8   waterfront     19221 non-null  float64
 9   view           21534 non-null  float64
 10  condition      21597 non-null  int64  
 11  grade          21597 non-null  int64  
 12  sqft_above     21597 non-null  int64  
 12  sqft_above     21597 non-null  int64
 13  sqft_basement  21597 non-null  object
 14  yr_built       21597 non-null  int64
 15  yr_renovated   17755 non-null  float64
 16  zipcode        21597 non-null  int64
 17  lat            21597 non-null  float64
 18  long           21597 non-null  float64
 19  sqft_living15  21597 non-null  int64
 20  sqft_lot15     21597 non-null  int64
dtypes: float64(8), int64(11), object(2)
memory usage: 3.5+ MB
```

In [26]:

```
# before we drop any null values (in yr_renovated and waterfront specif
ically) we should address the issues above.
df_init = df.drop(columns=['id','sqft_living15','sqft_above','sqft_base
ment'])
# filter for more recent home sales
df_init['date'].unique()
# can remove date and null values
print(df['yr_renovated'].unique())
print(df.groupby(df['yr_renovated']).count())
# given that these dates are very spread out and also that 17011 of the
homes sold out have no renovation year at all, I will remove the data f
or the purposes of a linear regression
```

```
[   0. 1991.   nan 2002. 2010. 1992. 2013. 1994. 1978. 2005. 2003. 198
4.
 1954. 2014. 2011. 1983. 1945. 1990. 1988. 1977. 1981. 1995. 2000. 199
9.
 1998. 1970. 1989. 2004. 1986. 2007. 1987. 2006. 1985. 2001. 1980. 197
1.
 1979. 1997. 1950. 1969. 1948. 2009. 2015. 1974. 2008. 1968. 2012. 196
3.
 1951. 1962. 1953. 1993. 1996. 1955. 1982. 1956. 1940. 1976. 1946. 197
5.
 1964. 1973. 1957. 1959. 1960. 1967. 1965. 1934. 1972. 1944. 1958.]
                 id   date  price  bedrooms  bathrooms  sqft_living  sq
ft_lot  \
yr_renovated
0.0           17011  17011  17011     17011      17011        17011
17011
1934.0            1      1      1         1          1            1
1
1940.0            2      2      2         2          2            2
2
1944.0            1      1      1         1          1            1
1
1945.0            3      3      3         3          3            3
3
...             ...    ...    ...       ...        ...          ...
...
2011.0            9      9      9         9          9            9
9
2012.0            8      8      8         8          8            8
8
2013.0           31     31     31        31         31           31
31
2014.0           73     73     73        73         73           73
73
```

```
73
2015.0           14     14     14        14          14            14
14

              floors  waterfront   view  condition  grade  sqft_above
\
yr_renovated
0.0            17011       15157  16961      17011  17011       17011
1934.0             1           1      1          1      1           1
1940.0             2           2      2          2      2           2
1944.0             1           1      1          1      1           1
1945.0             3           2      3          3      3           3
...              ...         ...    ...        ...    ...         ...
2011.0             9           7      9          9      9           9
2012.0             8           7      8          8      8           8
2013.0            31          29     31         31     31          31
2014.0            73          64     73         73     73          73
2015.0            14          13     14         14     14          14

              sqft_basement  yr_built  zipcode    lat   long  sqft_livi
ng15  \
yr_renovated
0.0                   17011     17011    17011  17011  17011          1
7011
1934.0                    1         1        1      1      1
1
1940.0                    2         2        2      2      2
2
1944.0                    1         1        1      1      1
1
1945.0                    3         3        3      3      3
3
...                     ...       ...      ...    ...    ...
...
2011.0                    9         9        9      9      9
9
2012.0                    8         8        8      8      8
8
2013.0                   31        31       31     31     31
31
2014.0                   73        73       73     73     73
73
2015.0                   14        14       14     14     14
14

              sqft_lot15
yr_renovated
0.0                17011
1934.0                 1
1940.0                 2
1944.0                 1
1945.0                 3
...                  ...
2011.0                 9
2012.0                 8
2013.0                31
2014.0                73
2015.0                14
```

```
[70 rows x 20 columns]
```

In [27]:

```
df_init = df_init.dropna() # drop null values
df_init = df_init.drop(columns=['date','yr_renovated']) # drop date col
umn
```

In [28]:

```
df_init.head()
```

Out[28]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view
1	538000.0	3	2.25	2570	7242	2.0	0.0	0.0
3	604000.0	4	3.00	1960	5000	1.0	0.0	0.0
4	510000.0	3	2.00	1680	8080	1.0	0.0	0.0
5	1230000.0	4	4.50	5420	101930	1.0	0.0	0.0
6	257500.0	3	2.25	1715	6819	2.0	0.0	0.0

In [29]:

```
df_init.info() # data is all in integer or float dtype
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15762 entries, 1 to 21596
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   price        15762 non-null  float64
 1   bedrooms     15762 non-null  int64  
 2   bathrooms    15762 non-null  float64
 3   sqft_living  15762 non-null  int64  
 4   sqft_lot     15762 non-null  int64  
 5   floors       15762 non-null  float64
 6   waterfront   15762 non-null  float64
 7   view         15762 non-null  float64
 8   condition    15762 non-null  int64  
 9   grade        15762 non-null  int64  
 10  yr_built     15762 non-null  int64  
 11  zipcode      15762 non-null  int64  
 12  lat          15762 non-null  float64
 13  long         15762 non-null  float64
 14  sqft_lot15   15762 non-null  int64  
dtypes: float64(7), int64(8)
memory usage: 1.9 MB
```

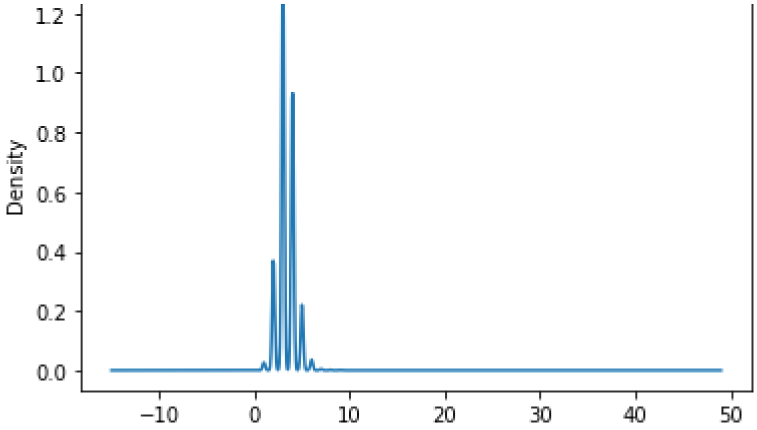
In [30]:

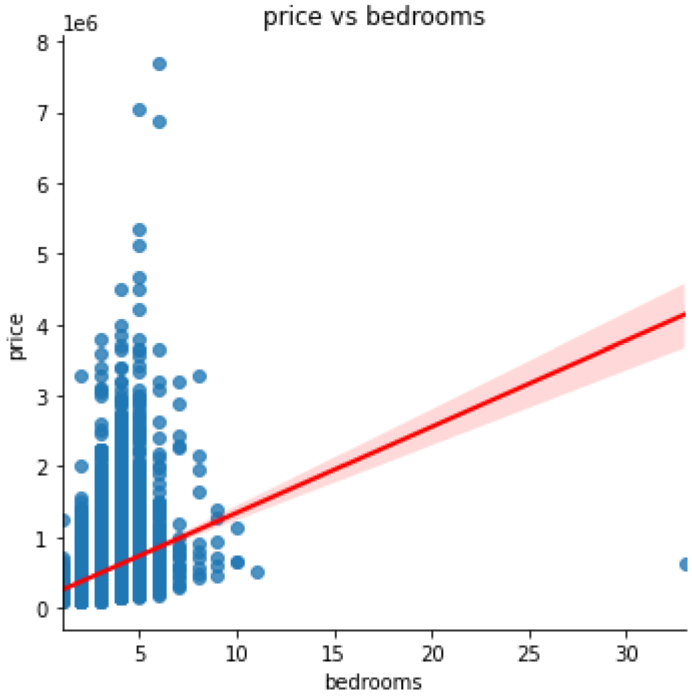
```
features = ['bedrooms','bathrooms','sqft_living','sqft_lot','sqft_lot1
5','floors','waterfront','view','condition','grade','yr_built','yr_reno
vated','lat','long']
```

In [9]:

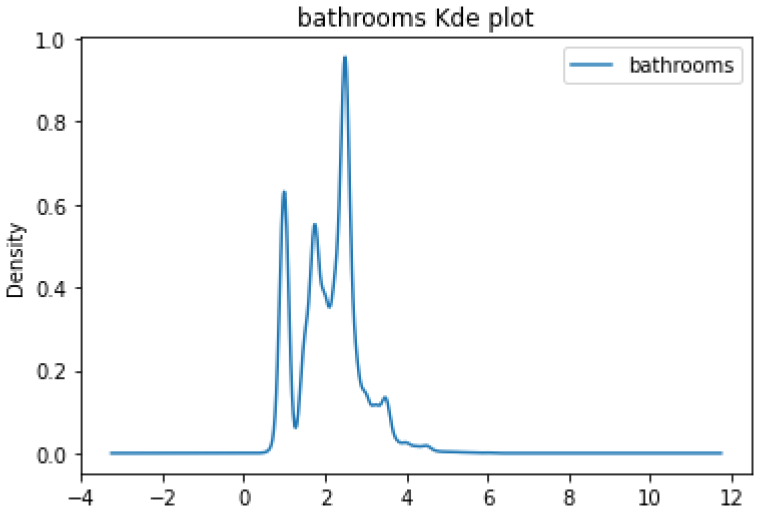
```
for feature in features:
    hist_kde_plots(feature,'price',df_init)
```

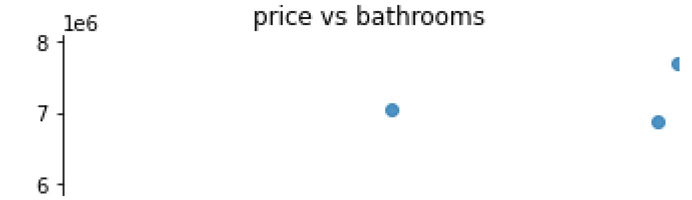
```
bedrooms
```

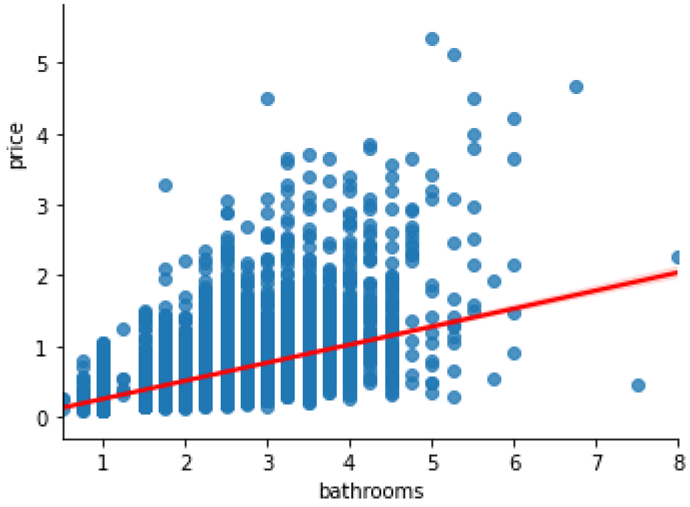

Density
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−10
0
10
20
30
40
50


price vs bedrooms
1e6
8
7
6
5
4
3
2
1
0
price
5
10
15
20
25
30
bedrooms

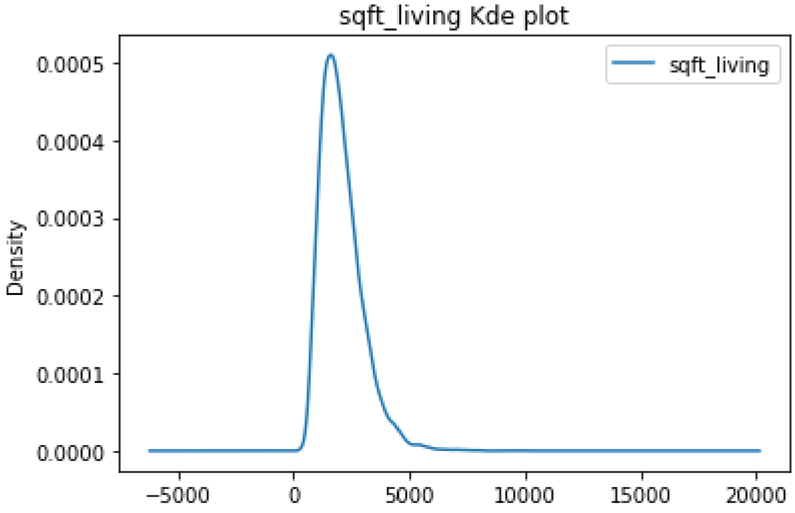
bathrooms

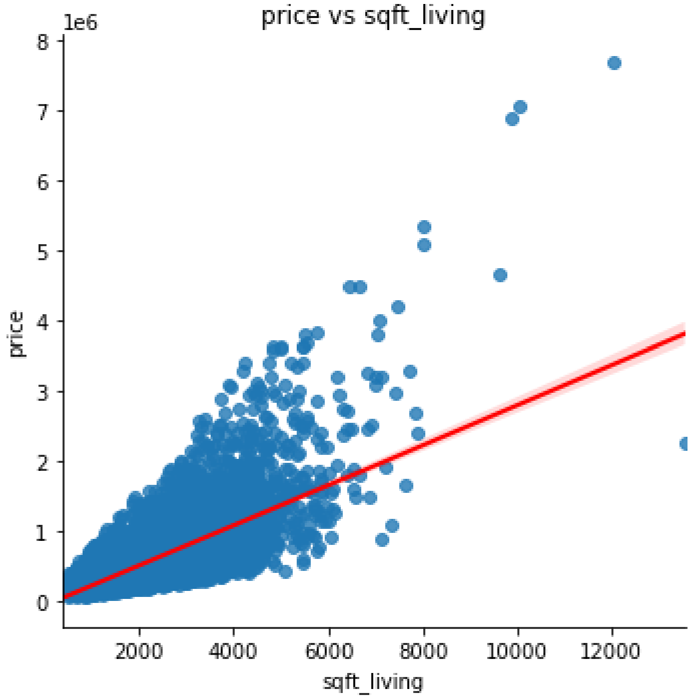

bathrooms Kde plot
bathrooms
1.0
0.8
0.6
0.4
0.2
0.0
Density
−4
−2
0
2
4
6
8
10
12


price vs bathrooms
1e6
8
7
6



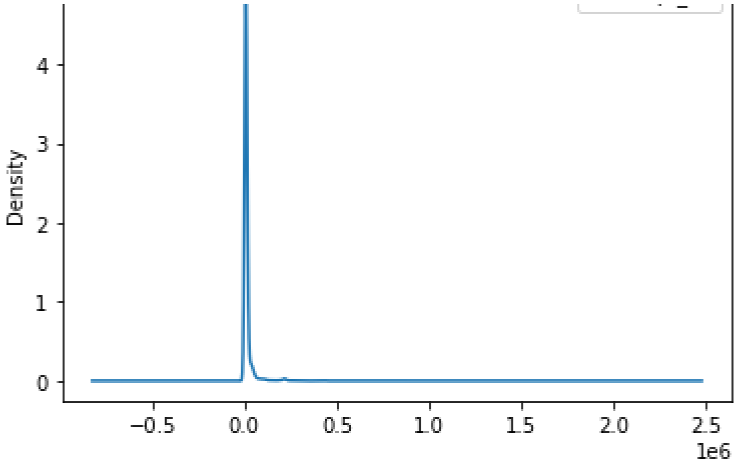
sqft_living

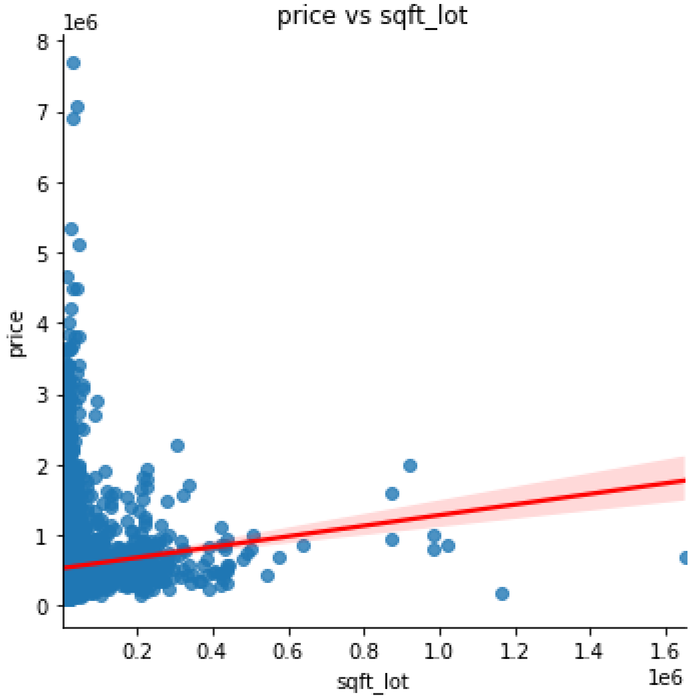

sqft_living Kde plot


price vs sqft_living

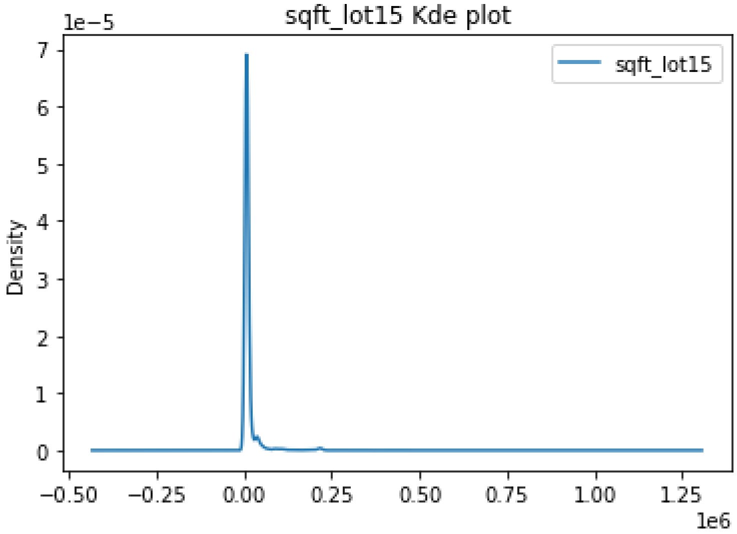
sqft_lot

sqft_lot Kde plot

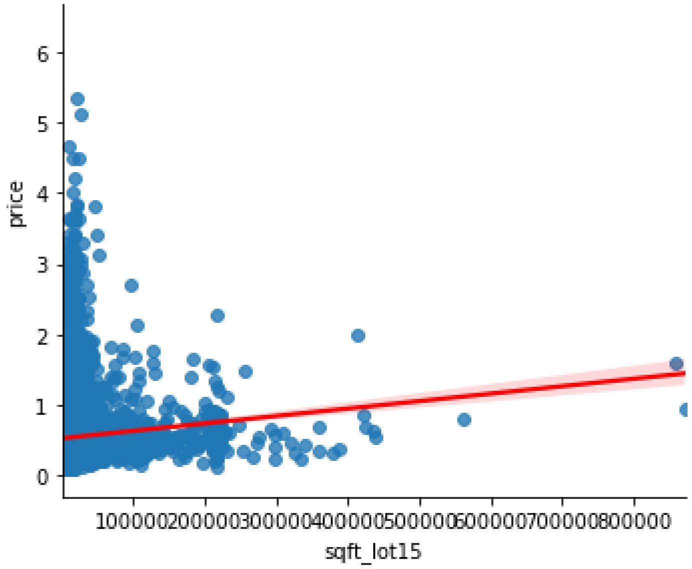

Density
4
3
2
1
0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
1e6


price vs sqft_lot
1e6
8
7
6
5
4
3
2
1
0
price
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
sqft_lot
1e6

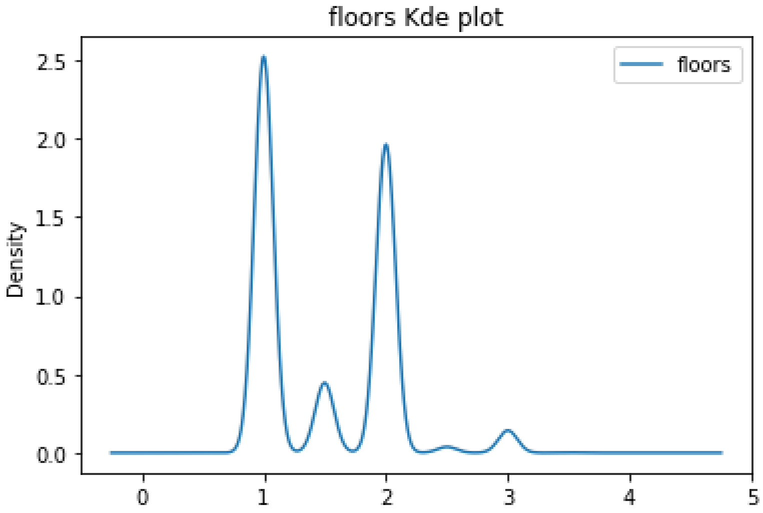
sqft_lot15

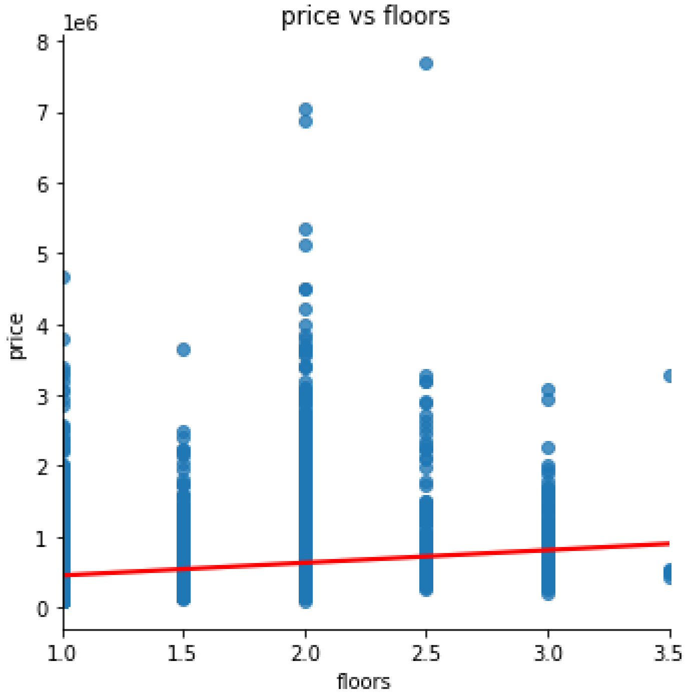

sqft_lot15 Kde plot
1e−5
sqft_lot15
7
6
5
4
3
2
1
0
Density
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
1e6

price vs sqft_lot15
1e6
8
7



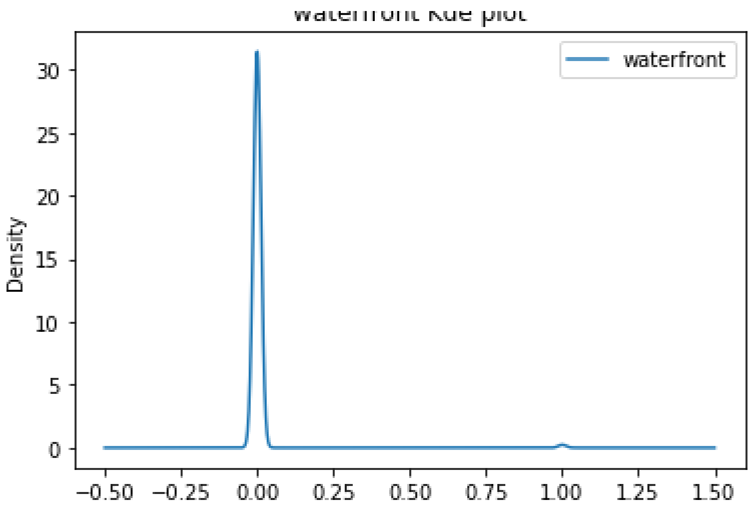
floors

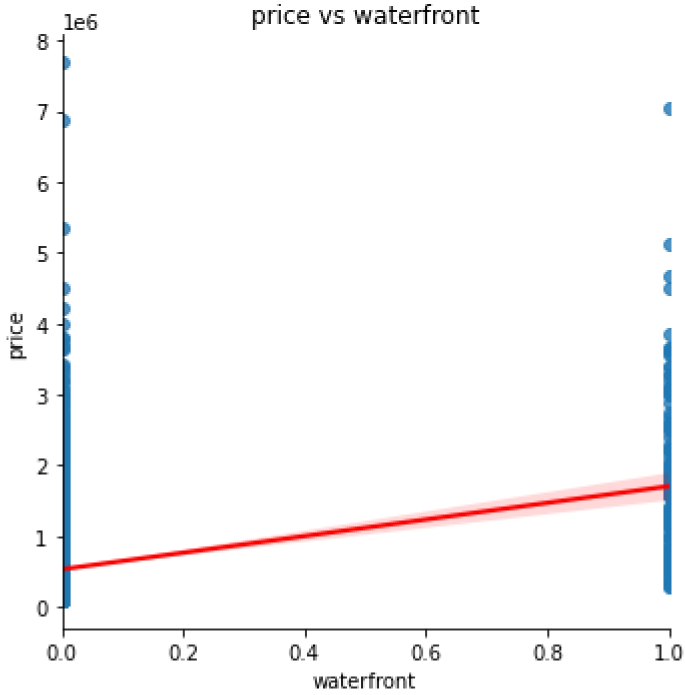




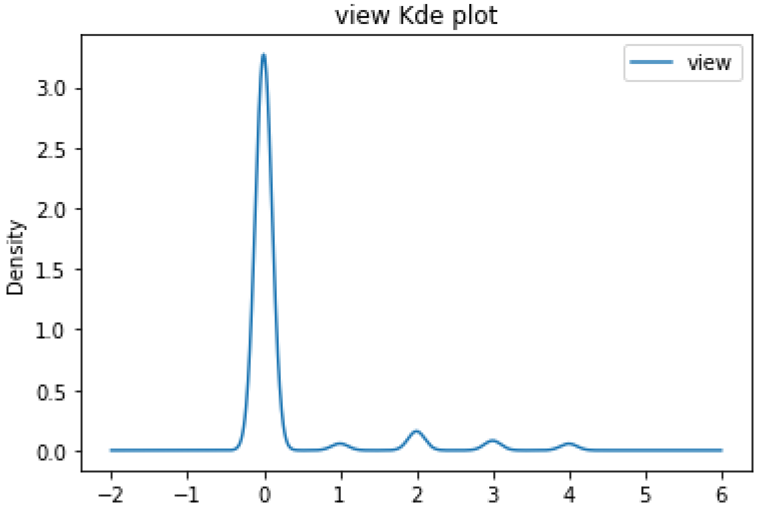
waterfront

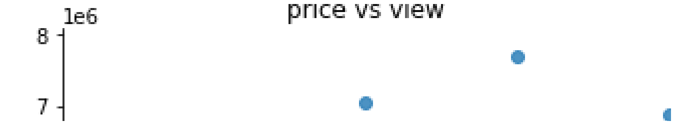
waterfront Kde plot

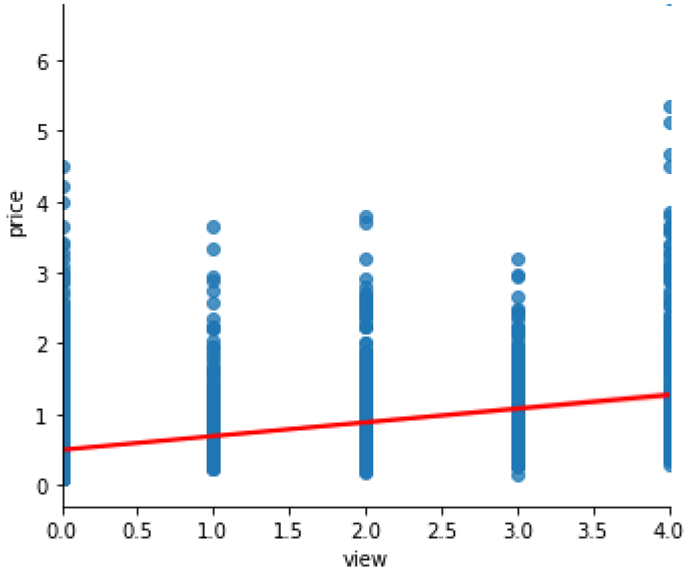

waterfront Kde plot
waterfront
Density
30
25
20
15
10
5
0
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50


price vs waterfront
1e6
8
7
6
5
4
3
2
1
0
price
0.0
0.2
0.4
0.6
0.8
1.0
waterfront

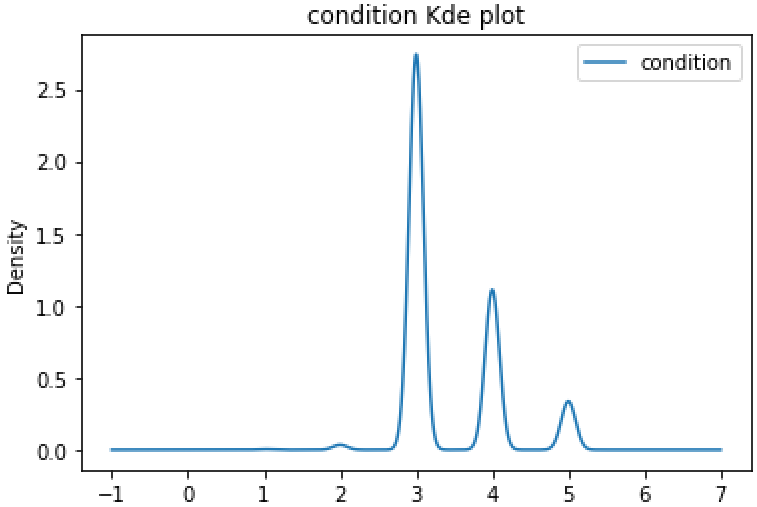
view

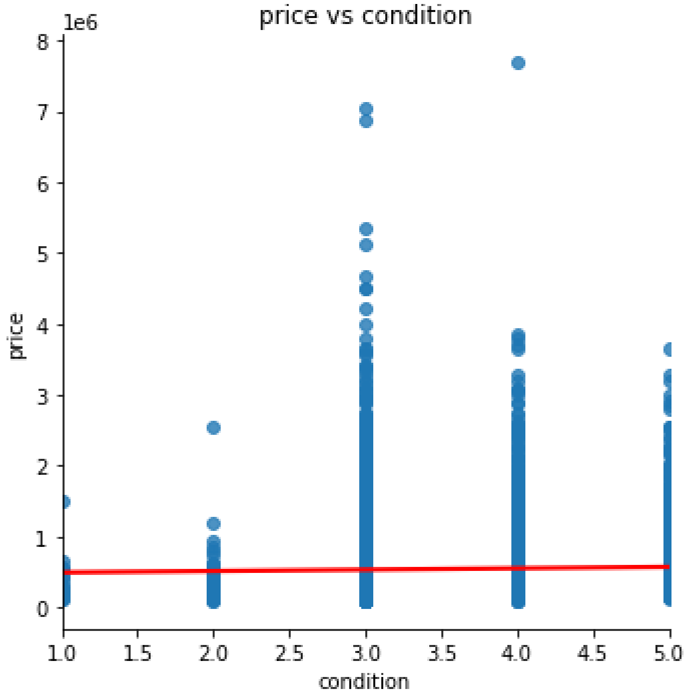

view Kde plot
view
Density
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−2
−1
0
1
2
3
4
5
6


price vs view
1e6
8
7

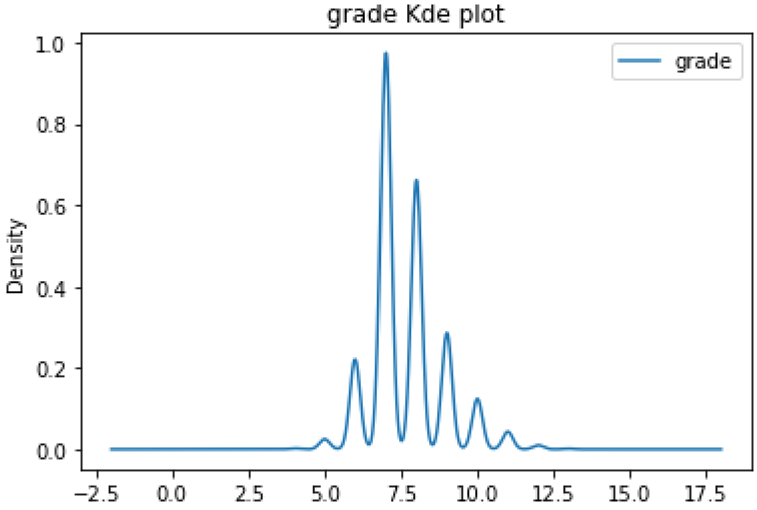


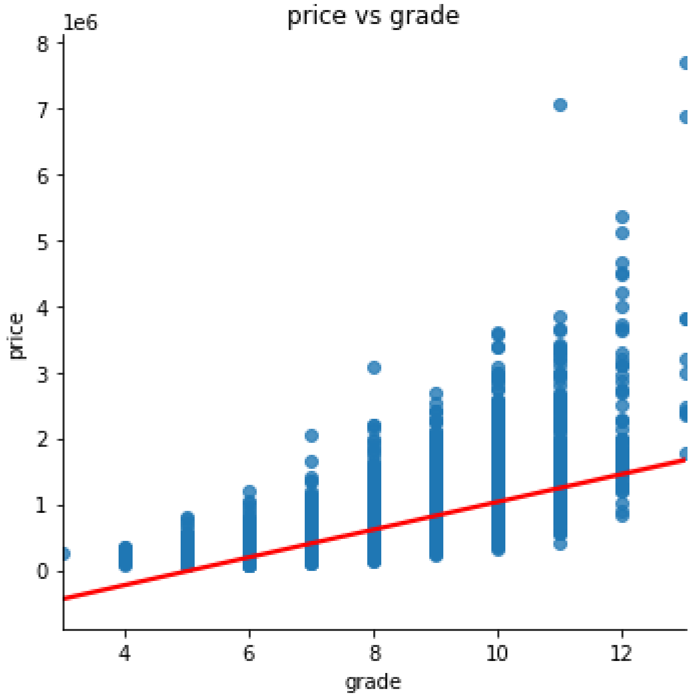
condition



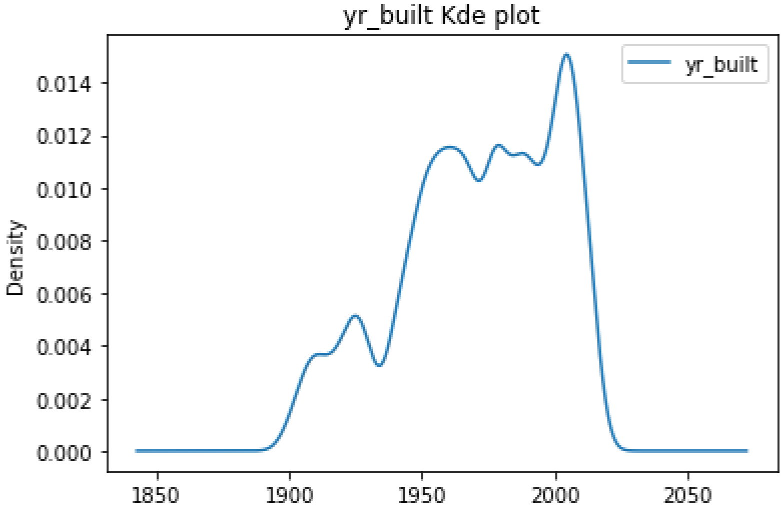


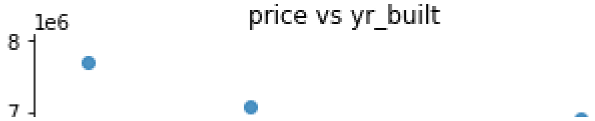
grade

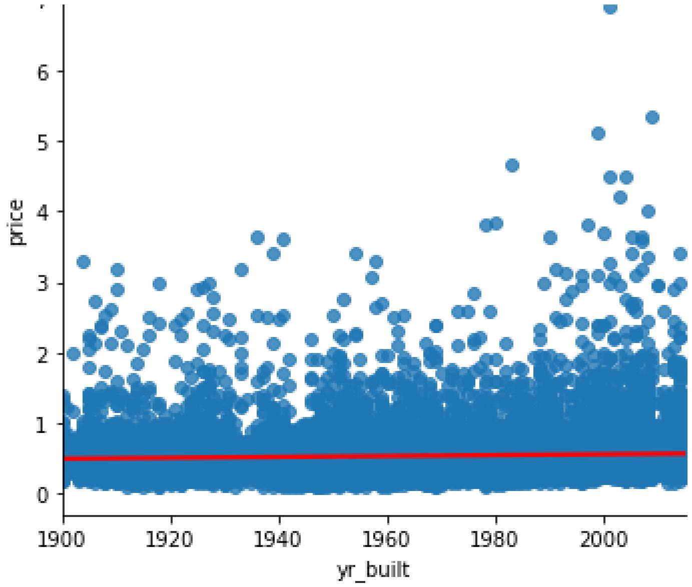

grade Kde plot
grade
Density
1.0
0.8
0.6
0.4
0.2
0.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


price vs grade
1e6
8
7
6
5
4
3
2
1
0
price
4
6
8
10
12
grade

yr_built


yr_built Kde plot
yr_built
Density
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
1850
1900
1950
2000
2050


price vs yr_built
1e6
8
7



```
yr_renovated
------------------------------------------------------------------------
----
KeyError                                  Traceback (most recent call l
ast)
~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(se
lf, key, method, tolerance)
   2645             try:
-> 2646                 return self._engine.get_loc(key)
   2647             except KeyError:

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObj
ectHashTable.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObj
ectHashTable.get_item()

KeyError: 'yr_renovated'

During handling of the above exception, another exception occurred:

KeyError                                  Traceback (most recent call l
ast)
<ipython-input-9-ca5a9d20d9bc> in <module>
      1 for feature in features:
----> 2     hist_kde_plots(feature,'price',df_init)

~\desktop\coursework\phase_1\Phase2\Phase_2_Project\pltfunctions.py in
hist_kde_plots(feature, target, df)
     52     print(feature)
     53     #kde plot
---> 54     df[feature].plot.kde(label=feature)
     55     plt.title("{} Kde plot".format(feature))
     56     plt.legend()

~\anaconda3\lib\site-packages\pandas\core\frame.py in __getitem__(self,
~\anaconda3\lib\site-packages\pandas\core\frame.py in __getitem__(self,
key)
   2798             if self.columns.nlevels > 1:
   2799                 return self._getitem_multilevel(key)
-> 2800             indexer = self.columns.get_loc(key)
   2801             if is_integer(indexer):
   2802                 indexer = [indexer]

~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(se
lf, key, method, tolerance)
   2646                 return self._engine.get_loc(key)
   2647             except KeyError:
-> 2648                 return self._engine.get_loc(self._maybe_cast_in
dexer(key))
   2649         indexer = self.get_indexer([key], method=method, tolera
nce=tolerance)
   2650         if indexer.ndim > 1 or indexer.size > 1:

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObj
ectHashTable.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObj
ectHashTable.get_item()

KeyError: 'yr_renovated'
```

In [31]:

```
# funky data in the following columns - 1) yr_renovated - a lot of home
s that haven't been
# renovated and they are all listed as 0's. Need to reconcile this data
- because of groupby count analysis above decided to just remove this d
ata (<17000 homes were never renovated per original dataset)
# 2) remove any price data that is greater than 3 standard deviations f
rom the mean
df_init.describe()
```

Out[31]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	fl
count	1.576200e+04	15762.000000	15762.000000	15762.000000	1.576200e+04	1
mean	5.413172e+05	3.378949	2.120797	2084.512372	1.528082e+04	1
std	3.722258e+05	0.935301	0.766772	918.617686	4.182288e+04	0
min	8.200000e+04	1.000000	0.500000	370.000000	5.200000e+02	1
25%	3.210000e+05	3.000000	1.750000	1430.000000	5.048500e+03	1
50%	4.500000e+05	3.000000	2.250000	1920.000000	7.602000e+03	1
75%	6.448750e+05	4.000000	2.500000	2550.000000	1.072000e+04	2
max	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3

In [32]:

```
df_init.isna().sum()
```

Out[32]: price 0

```
Out[32]: price          0
         bedrooms       0
         bathrooms      0
         sqft_living    0
         sqft_lot       0
         floors         0
         waterfront     0
         view           0
         condition      0
         grade          0
         yr_built       0
         zipcode        0
         lat            0
         long           0
         sqft_lot15     0
         dtype: int64
```

```
In [33]: # remove outliers and data will be cleaned, can move on to feature engi
         neering
         # we can standartize the data as well to do this
         # we can see that there are no homes below 3 standard deviations with t
         he describe table
         # however there are homes above 3 standard deviations of the mean price
         z = abs(stats.zscore(df_init))
         print(z)
         threshold = 3
         print(np.where(z>3))
         avg_price = df_init['price'].mean()
         stdev_price = df_init['price'].std()
         upper_bound = avg_price + 3*stdev_price

         # df_clean = df_init[(z < 3).all(axis=1)] # this doesnt work because we
         remove some data (waterfront) that we don't want to remove - remove onl
         y homes sold with a price above 3 standard deviations
         df_clean = df_init[df_init['price']<upper_bound]
```

```
         [[0.00891201 0.40517583 0.16850812 ... 1.16734842 0.74967289 0.1880666
         ]
          [0.16840532 0.66403252 1.14666608 ... 0.2768393  1.27560722 0.2823963
         ]
          [0.08413755 0.40517583 0.15754454 ... 0.41567829 1.19770555 0.1929278
         5]
          ...
          [0.37966644 0.66403252 0.49456077 ... 0.34969792 1.05528338 0.2037584
         3]
          [0.37402184 1.47438418 1.78780781 ... 0.25409085 0.60752848 0.3893795
         5]
          [0.58116341 1.47438418 1.78780781 ... 0.25192673 0.60752848 0.4126134
         6]]
         (array([    3,     3,     3, ..., 15740, 15740, 15751], dtype=int64), a
         rray([ 2,  3, 14, ...,  6,  7,  3], dtype=int64))
```

```
In [34]: df_clean.isna().sum()
```

```
Out[34]: price          0
         bedrooms       0
         bathrooms      0
         sqft_living    0
```

```
sqft_lot        0
floors          0
waterfront      0
view            0
condition       0
grade           0
yr_built        0
zipcode         0
lat             0
long            0
sqft_lot15      0
dtype: int64
```

In [35]:

```
df_clean.price.max() # have removed absurdly high priced homes from df
df_clean.waterfront.max() # where we discover we removed data that is b
inary with low standard deviation compared to the max, which is 1.
# lets try a different method
df_clean.describe()
```

Out[35]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	fl
count	1.548000e+04	15480.000000	15480.000000	15480.000000	1.548000e+04	1
mean	5.084310e+05	3.362532	2.093169	2037.260401	1.507320e+04	1
std	2.615181e+05	0.926927	0.733848	834.955963	4.123850e+04	0
min	8.200000e+04	1.000000	0.500000	370.000000	5.200000e+02	1
25%	3.200000e+05	3.000000	1.500000	1420.000000	5.012750e+03	1
50%	4.490000e+05	3.000000	2.250000	1900.000000	7.560000e+03	1
75%	6.275000e+05	4.000000	2.500000	2510.000000	1.050000e+04	2
max	1.650000e+06	33.000000	7.500000	7350.000000	1.651359e+06	3

In [36]:

```
# after having completed a regression, discovered that lat and long are
not worthwhile in a regression analysis
# as a result, will create a distance from seattle feature to create a
 regression
```

In [85]:

```
df_clean['lat'] = [round(i,4) for i in df_clean['lat']]
df_clean['long'] = [round(i,4) for i in df_clean['long']]
```

In [86]:

```
df_clean['geo_loc'] = list(zip(df_clean['lat'], df_clean['long']))
```

In [87]:

```
Seattle = [47.6219, -122.3517] # defining Seattle's location
```

In [88]:

```
distance = []
for i in df_clean['geo_loc']:
    distance.append((haversine((Seattle),(i), unit='mi')))
rounded = [round(i,4) for i in distance]

df_clean['distance'] = rounded
```

In [89]:

```
df_clean.info()
```

```
In [89]: df_clean.info()
         df_clean.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15480 entries, 1 to 21596
Data columns (total 17 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   price         15480 non-null  float64
 1   bedrooms      15480 non-null  int64  
 2   bathrooms     15480 non-null  float64
 3   sqft_living   15480 non-null  int64  
 4   sqft_lot      15480 non-null  int64  
 5   floors        15480 non-null  float64
 6   waterfront    15480 non-null  float64
 7   view          15480 non-null  float64
 8   condition     15480 non-null  int64  
```