In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import mpl_toolkits
%matplotlib inline
```

In [75]:
```
df = pd.read_csv("kc_house_data.csv")
```

In [76]:
```
df.head()
```

Out[76]:

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

5 rows × 21 columns

In [77]:
```
df.describe()
```

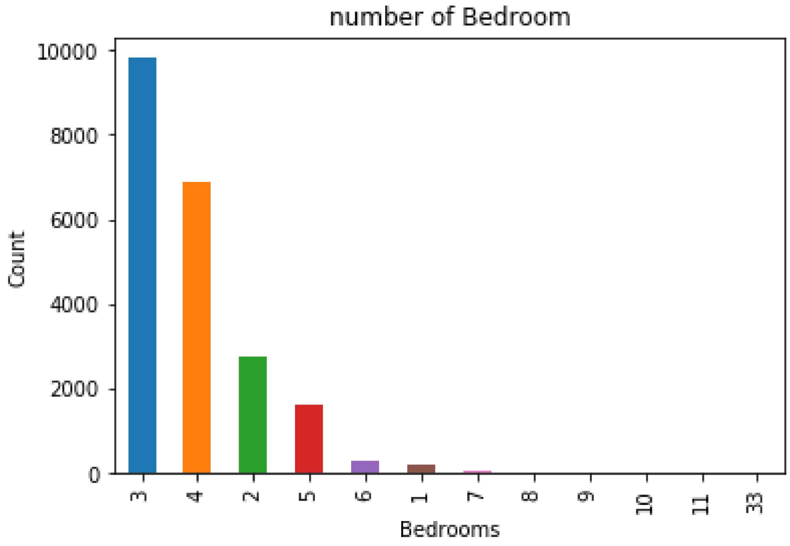
Out[77]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	
count	2.159700e+04	2.159700e+04	21597.000000	21597.000000	21597.000000	2.159700e+04	21597
mean	4.580474e+09	5.402966e+05	3.373200	2.115826	2080.321850	1.509941e+04	1
std	2.876736e+09	3.673681e+05	0.926299	0.768984	918.106125	4.141264e+04	0
min	1.000102e+06	7.800000e+04	1.000000	0.500000	370.000000	5.200000e+02	1
25%	2.123049e+09	3.220000e+05	3.000000	1.750000	1430.000000	5.040000e+03	1
50%	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	1
75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000	1.068500e+04	2
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3

In [78]:
```
data=df.dropna()
```

In [79]:
```
data['bedrooms'].value_counts().plot(kind='bar')
plt.title('number of Bedroom')
plt.xlabel('Bedrooms')
plt.ylabel('Count')
sns.despine
```

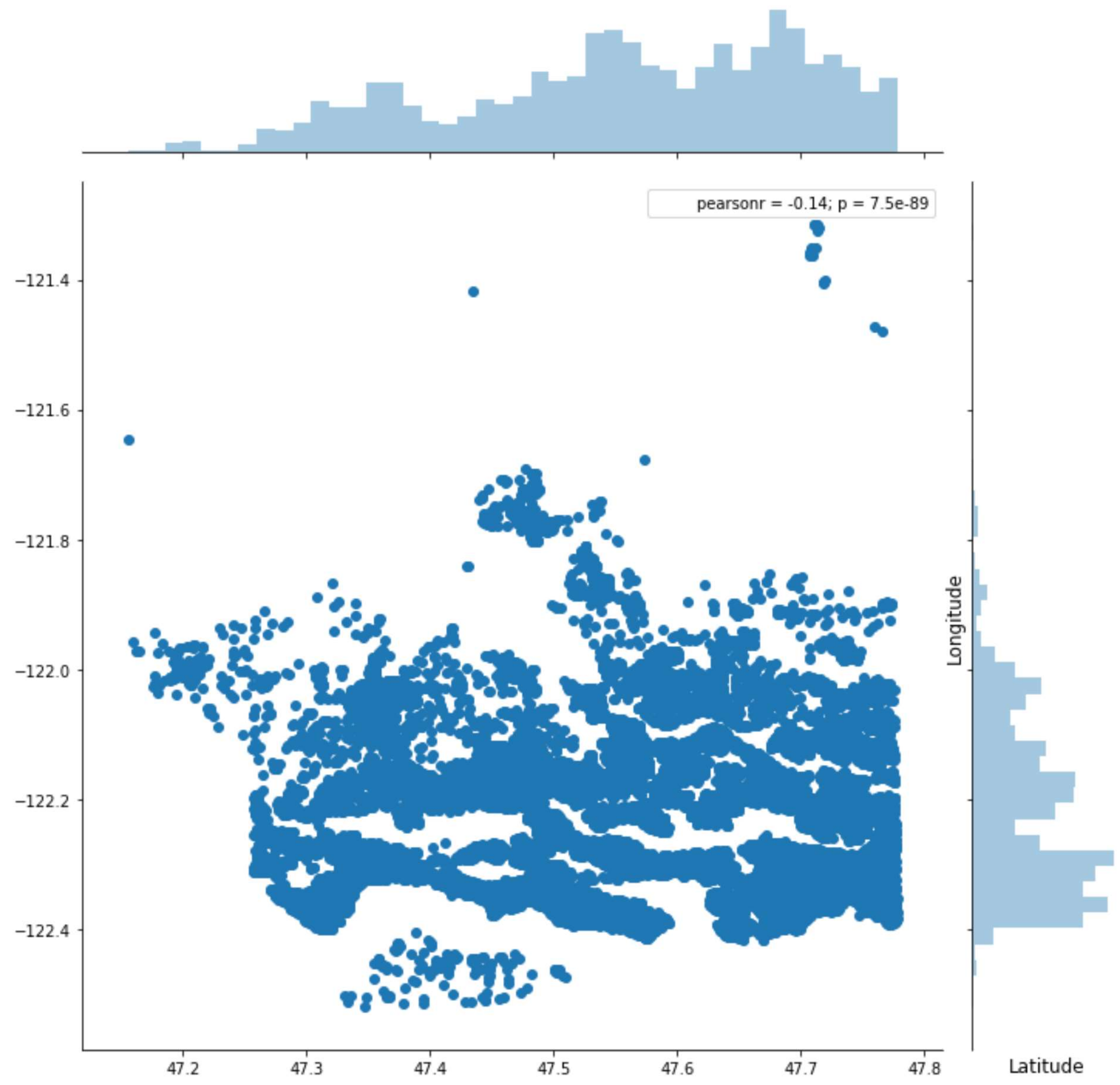
Out[79]:
```
<function seaborn.utils.despine(fig=None, ax=None, top=True, right=True, left=F
alse, bottom=False, offset=None, trim=False)>
```



In [80]:

```
plt.figure(figsize=(10,10))
sns.jointplot(x=data.lat.values, y=data.long.values, size=10)
plt.ylabel('Longitude', fontsize=12)
plt.xlabel('Latitude', fontsize=12)
plt.show()
plt1 = plt()
sns.despine
```

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-pac
kages\matplotlib\axes\_axes.py:6462: UserWarning: The 'normed' kwarg is depreca
ted, and has been replaced by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-pac
kages\matplotlib\axes\_axes.py:6462: UserWarning: The 'normed' kwarg is depreca
ted, and has been replaced by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "

<Figure size 720x720 with 0 Axes>
```



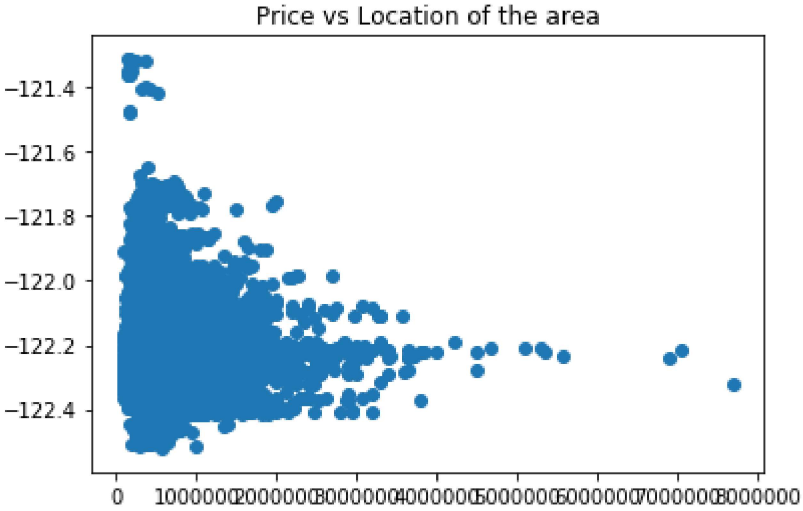
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-80-67b21dd0040b> in <module>()
      4 plt.xlabel('Latitude', fontsize=12)
      5 plt.show()
----> 6 plt1 = plt()
      7 sns.despine

TypeError: 'module' object is not callable
```

In [ ]:

```
plt.scatter(data.price,data.sqft_living)
plt.title("Price vs Square Feet")
```

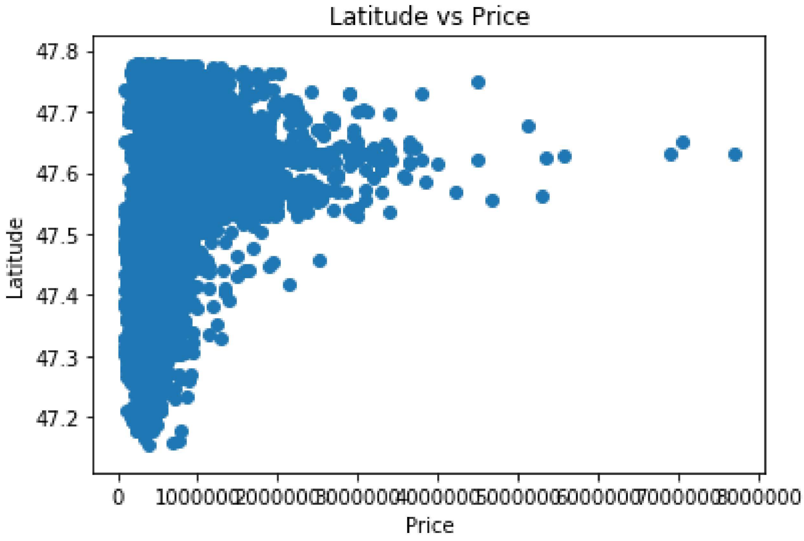
In [81]:
```
plt.scatter(data.price,data.long)
plt.title("Price vs Location of the area")
```

Out[81]:
```
Text(0.5,1,'Price vs Location of the area')
```



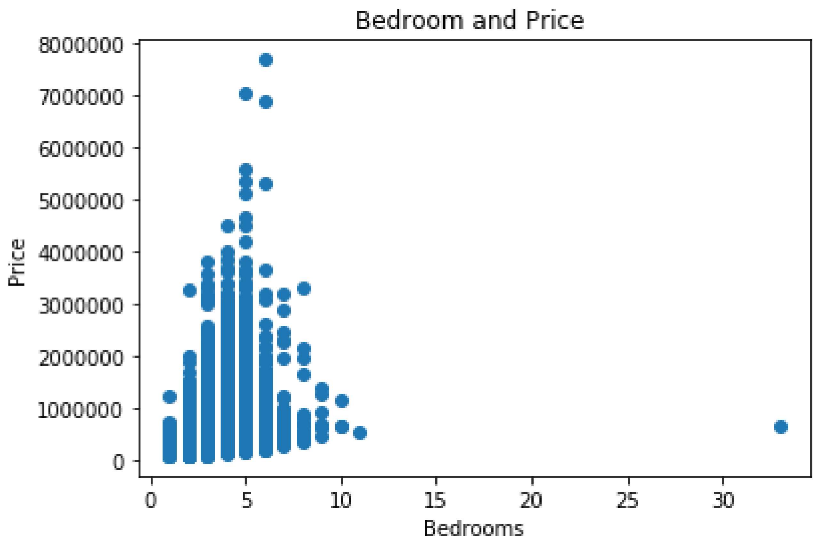
In [82]:
```
plt.scatter(data.price,data.lat)
plt.xlabel("Price")
plt.ylabel('Latitude')
plt.title("Latitude vs Price")
```

Out[82]:
```
Text(0.5,1,'Latitude vs Price')
```



In [83]:

```
plt.scatter(data.bedrooms,data.price)
plt.title("Bedroom and Price ")
plt.xlabel("Bedrooms")
plt.ylabel("Price")
plt.show()
sns.despine
```



Out[83]:

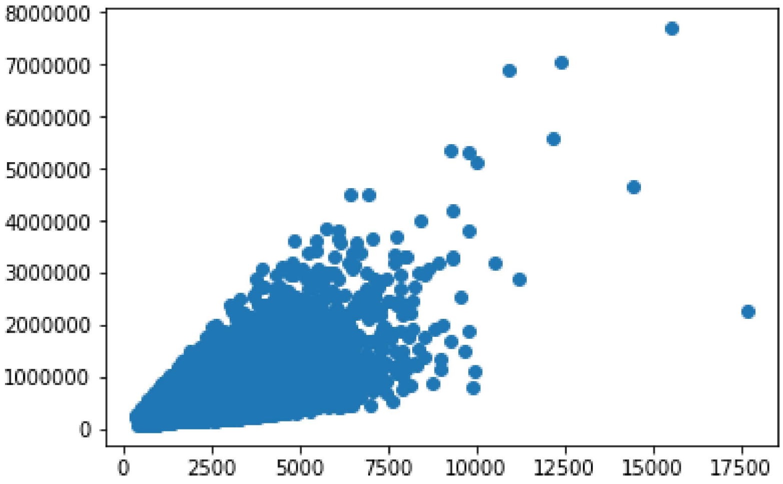
```
<function seaborn.utils.despine(fig=None, ax=None, top=True, right=True, left=F
alse, bottom=False, offset=None, trim=False)>
```

In [84]:

```
plt.scatter((data['sqft_living']+data['sqft_basement']),data['price'])
```

Out[84]:

```
<matplotlib.collections.PathCollection at 0x2349e28cac8>
```

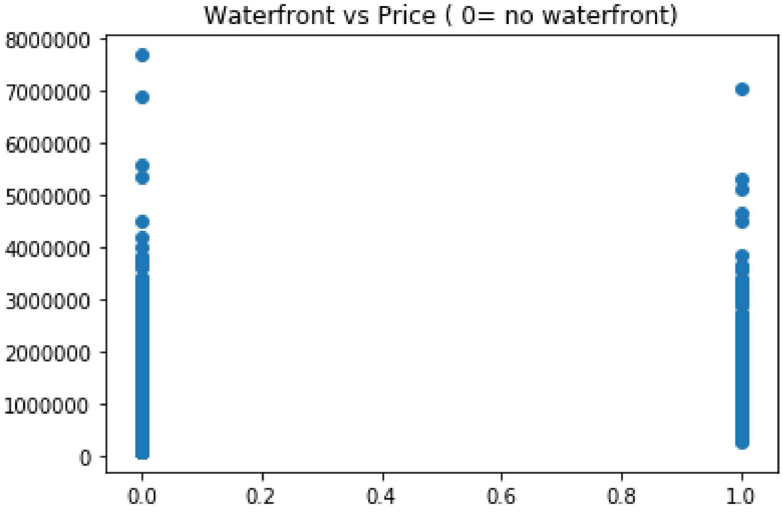


In [85]:

```
plt.scatter(data.waterfront,data.price)
plt.title("Waterfront vs Price ( 0= no waterfront)")
```

Out[85]:

```
Text(0.5,1,'Waterfront vs Price ( 0= no waterfront)')
```



In [86]:

```
train1 = data.drop(['id', 'price'],axis=1)
```

In [87]:

```
train1.head()
```

Out[87]:

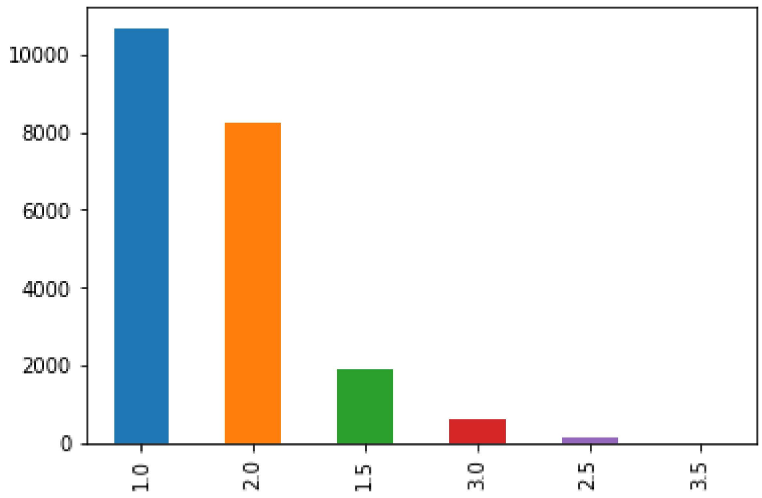
	date	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grad
0	10/13/2014	3	1.00	1180	5650	1.0	0	0	3	
1	12/9/2014	3	2.25	2570	7242	2.0	0	0	3	
2	2/25/2015	2	1.00	770	10000	1.0	0	0	3	
3	12/9/2014	4	3.00	1960	5000	1.0	0	0	5	
4	2/18/2015	3	2.00	1680	8080	1.0	0	0	3	

In [88]:

```
data.floors.value_counts().plot(kind='bar')
```

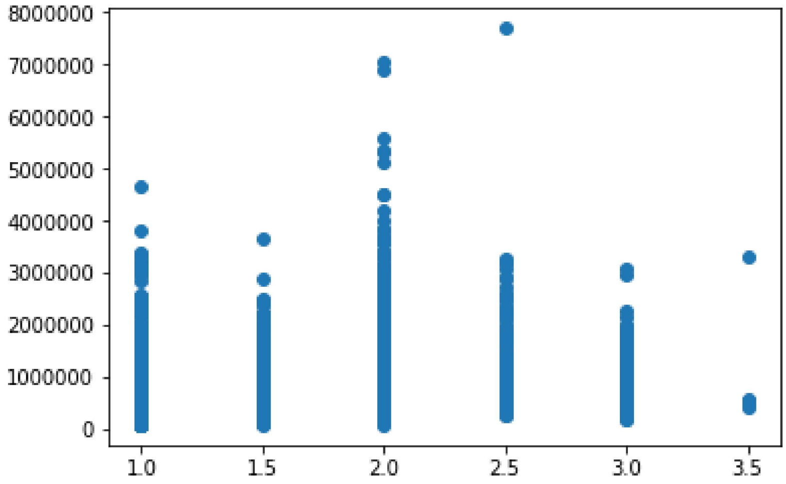
Out[88]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2349e3290b8>
```



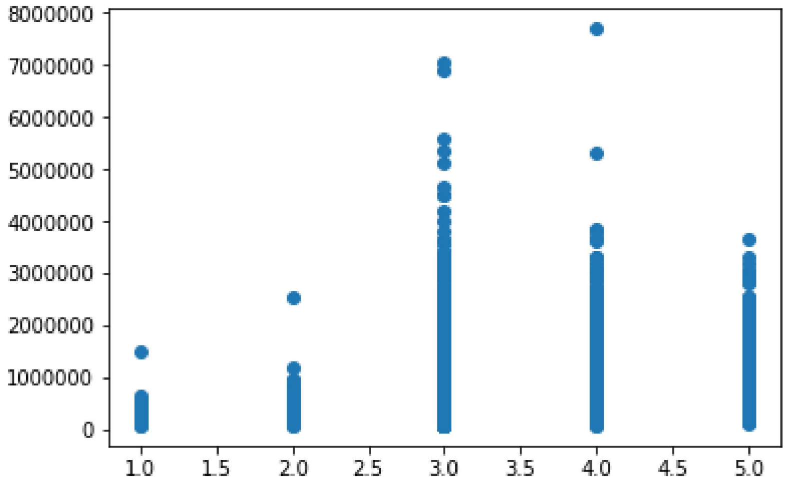
In [89]:
```
plt.scatter(data.floors,data.price)
```

Out[89]:
```
<matplotlib.collections.PathCollection at 0x2349e6d2898>
```



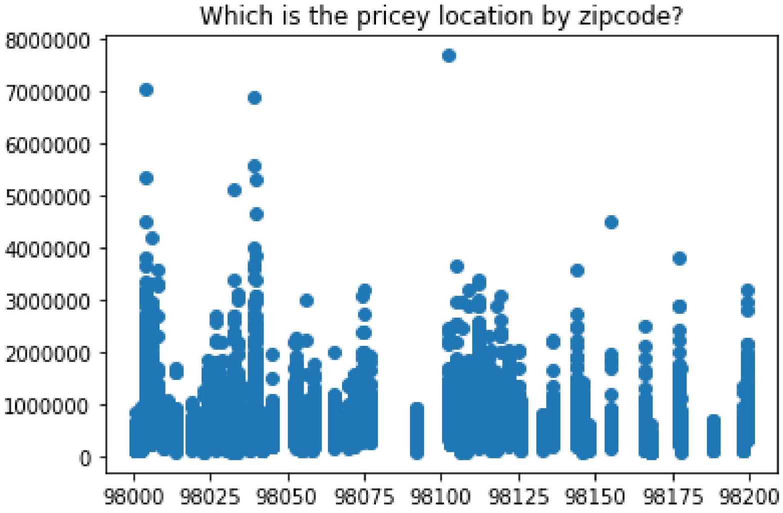
In [90]:
```
plt.scatter(data.condition,data.price)
```

Out[90]:
```
<matplotlib.collections.PathCollection at 0x2349e7341d0>
```



In [91]:
```
plt.scatter(data.zipcode,data.price)
plt.title("Which is the pricey location by zipcode?")
```

Out[91]:
```
Text(0.5,1,'Which is the pricey location by zipcode?')
```



In [92]:
```
from sklearn.linear_model import LinearRegression
```

In [93]:
```
reg = LinearRegression()
```

In [94]:
```
labels = data['price']
conv_dates = [1 if values == 2014 else 0 for values in data.date ]
data['date'] = conv_dates
train1 = data.drop(['id', 'price'],axis=1)
```

In [95]:
```
from sklearn.model_selection import train_test_split
```

In [96]:
```
train1.isnull().sum()
```

Out[96]:
```
date             0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
grade            0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
zipcode          0
lat              0
long             0
sqft_living15    0
sqft_lot15       0
dtype: int64
```

In [106]:
```
df=train1.dropna()
```

In [107]:
```
df.isnull().sum()
```

Out[107]:
```
date             0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
grade            0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
zipcode          0
lat              0
long             0
sqft_living15    0
sqft_lot15       0
dtype: int64
```

In [108]:
```
x_train , x_test , y_train , y_test = train_test_split(df , labels , test_size =
```

In [109]:

```
x_train.dtypes
```

Out[109]:

```
date               int64
bedrooms           int64
bathrooms        float64
sqft_living        int64
sqft_lot           int64
floors           float64
waterfront         int64
view               int64
condition          int64
grade              int64
sqft_above         int64
sqft_basement      int64
yr_built           int64
yr_renovated       int64
zipcode            int64
lat              float64
long             float64
sqft_living15      int64
sqft_lot15         int64
dtype: object
```

In [110]:

```
x_train.isnull().sum()
```

Out[110]:

```
date             0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
grade            0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
zipcode          0
lat              0
long             0
sqft_living15    0
sqft_lot15       0
dtype: int64
```

In [111]:
```
x_train.dropna()
```

Out[111]:

	date	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade
2980	0	4	2.50	3000	10392	2.0	0	0	3	9
11224	0	2	1.00	980	3600	1.0	0	0	3	6
15371	0	3	1.50	1340	7200	1.0	0	0	3	7
6039	0	4	3.25	4250	18000	2.0	0	3	5	10
2945	0	3	1.00	940	9272	1.0	0	0	3	7
7495	0	2	1.00	990	10556	2.0	0	0	3	7
16610	0	5	3.50	2950	7980	2.0	0	3	3	9
13496	0	4	2.75	2170	5988	2.0	0	0	3	8
8085	0	5	6.75	9640	13068	1.0	1	4	3	12
35	0	3	2.50	2300	3060	1.5	0	0	3	8
10432	0	5	2.50	2510	7525	1.5	0	0	4	7
11299	0	3	3.75	2380	3600	1.5	0	0	3	7
15723	0	2	1.50	920	1598	2.0	0	0	3	7
2604	0	2	1.00	1070	189486	1.0	0	0	3	6
6848	0	3	1.50	2330	11740	1.0	0	0	3	8
7773	0	3	2.00	2500	30056	1.0	0	0	5	8
14250	0	3	1.00	2030	4080	1.5	0	0	4	7
20378	0	3	2.25	1420	990	3.0	0	0	3	8
11508	0	4	1.75	2160	19283	2.0	0	0	3	8
10104	0	5	3.00	3640	6930	2.0	0	0	3	8
21199	0	3	2.50	1950	3825	2.0	0	0	3	7
9420	0	2	1.00	1140	7435	1.0	0	0	3	7
19655	0	4	2.50	3420	17038	2.0	0	0	3	9
6984	0	3	1.75	1720	15225	1.0	0	0	4	7
7508	0	3	2.00	1010	2820	1.5	0	0	3	7
2570	0	3	1.75	1720	223377	1.0	0	0	3	7
10342	0	3	1.75	2310	11200	1.0	0	0	4	8
12397	0	4	1.50	2070	7245	1.0	0	0	4	7
12353	0	4	4.00	5280	17677	2.0	0	3	3	11
9306	0	3	1.00	970	9583	1.0	0	0	4	6
...	...	...	...	...	...	...	...	...	...	...
7622	0	3	1.75	1360	16000	1.0	0	0	3	7
20164	0	4	3.50	2910	5260	2.0	0	0	3	9
9541	0	3	2.50	1940	8196	2.0	0	0	3	8

	date	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade
16639	0	4	2.50	1700	4268	2.0	0	0	3	7
21372	0	3	2.25	1330	1198	2.0	0	0	3	8
20026	0	4	2.50	3320	7429	2.0	0	0	3	9
15905	0	3	2.50	2530	8669	2.0	0	0	3	9
8170	0	4	2.75	2290	6120	2.0	0	0	4	7
14695	0	3	2.25	2090	15000	1.0	0	0	3	7
20531	0	3	3.50	1710	2212	2.0	0	0	3	7
21418	0	2	2.50	1430	923	3.0	0	0	3	8
13414	0	4	1.00	2080	3500	1.5	0	0	5	7
19111	0	2	1.00	890	8180	1.0	0	0	3	7
20084	0	4	2.50	1714	3080	2.0	0	0	3	8
8316	0	3	1.00	1240	5750	1.0	0	0	4	6
6548	0	3	1.75	1540	7490	1.0	0	0	5	7
19162	0	4	3.00	2370	3672	1.5	0	0	5	7
11071	0	3	2.00	2320	17688	1.0	0	0	3	8
5167	0	3	2.00	1280	14972	1.0	0	0	3	7
2773	0	3	1.00	1090	8520	1.0	0	0	3	7
10827	0	3	1.50	1810	14400	1.0	0	0	4	7
433	0	3	2.50	1490	2138	2.0	0	0	3	7
21154	0	4	2.50	1954	5075	2.0	0	0	3	8
11527	0	3	1.00	1150	2496	1.0	0	0	3	6
14696	0	3	2.25	1690	7292	1.0	0	0	3	7
1099	0	5	4.50	6070	14731	2.0	0	0	3	11
18898	0	3	1.75	1100	10125	1.0	0	0	4	7
11798	0	4	1.75	1700	10230	1.0	0	0	3	8
6637	0	4	2.25	2330	8994	2.0	0	0	3	8
2575	0	4	2.25	2080	7526	1.0	0	0	4	7

19437 rows × 19 columns

In [112]:
```
x_train.isnull().sum()
```

Out[112]:
```
date              0
bedrooms          0
bathrooms         0
sqft_living       0
sqft_lot          0
floors            0
waterfront        0
view              0
condition         0
grade             0
sqft_above        0
sqft_basement     0
yr_built          0
yr_renovated      0
zipcode           0
lat               0
long              0
sqft_living15     0
sqft_lot15        0
dtype: int64
```

In [113]:
```
x_train.fillna(method='ffill', inplace=True)
```

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-pac
kages\pandas\core\frame.py:3787: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stab
le/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy)
  downcast=downcast, **kwargs)
```

In [115]:
```
reg.fit(x_train,y_train)
```

Out[115]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

In [116]:
```
reg.score(x_test,y_test)
```

Out[116]:
```
0.6795469478306214
```

In [117]:
```
from sklearn import ensemble
clf = ensemble.GradientBoostingRegressor(n_estimators = 400, max_depth = 5, min_s
          learning_rate = 0.1, loss = 'ls')
```

```
In [119]: clf.fit(x_train, y_train)
```

```
Out[119]: GradientBoostingRegressor(alpha=0.9, criterion='friedman_mse', init=None,
                                  learning_rate=0.1, loss='ls', max_depth=5, max_features=None,
                                  max_leaf_nodes=None, min_impurity_decrease=0.0,
                                  min_impurity_split=None, min_samples_leaf=1,
                                  min_samples_split=2, min_weight_fraction_leaf=0.0,
                                  n_estimators=400, presort='auto', random_state=None,
                                  subsample=1.0, verbose=0, warm_start=False)
```

```
In [120]: clf.score(x_test,y_test)
```

```
Out[120]: 0.9140041072409933
```

```
In [ ]:
```