import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import mpl\_toolkits

%matplotlib inline

data=pd.read\_csv("kc\_house\_data.csv")

data.head()

data['bedrooms'].value\_counts().plot(kind='bar')

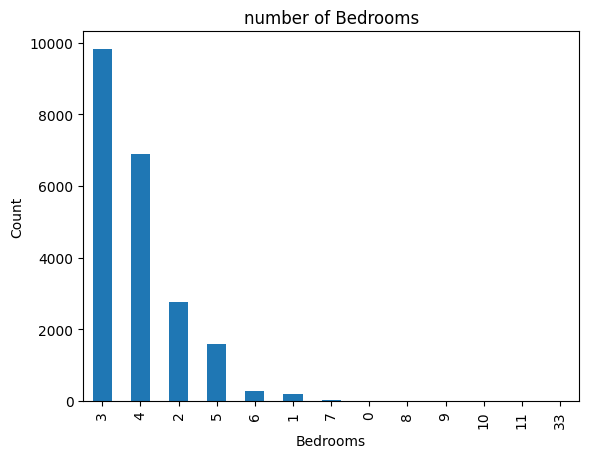
plt.title('number of Bedrooms')

plt.xlabel('Bedrooms')

plt.ylabel('Count')

sns.despine

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



plt.figure(figsize=(10,10))

sns.jointplot(x=data.lat.values,y=data.long.values,size=10)

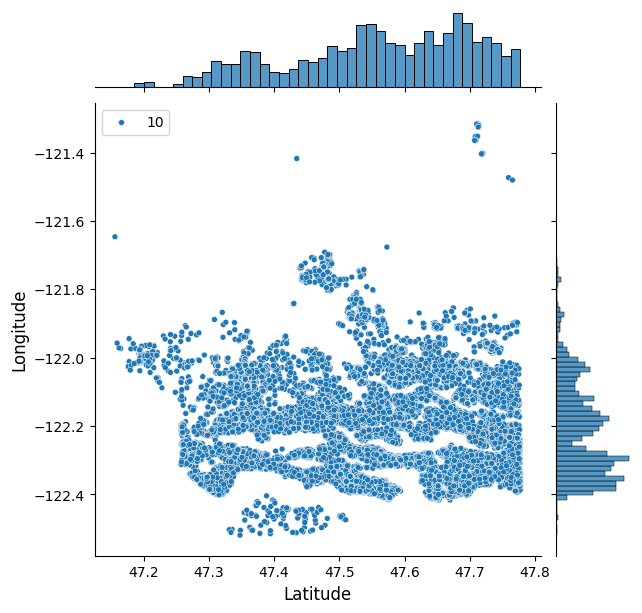
plt.ylabel('Longitude',fontsize=12)

plt.xlabel('Latitude',fontsize=12)

plt.show()

sns.despine

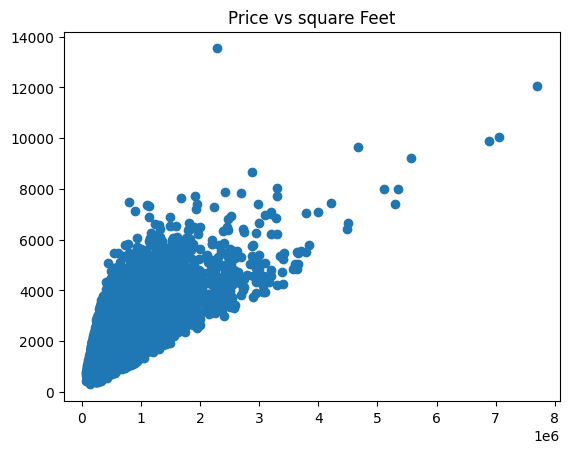
<Figure size 1000x1000 with 0 Axes>



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

plt.scatter(data.price,data.sqft\_living)

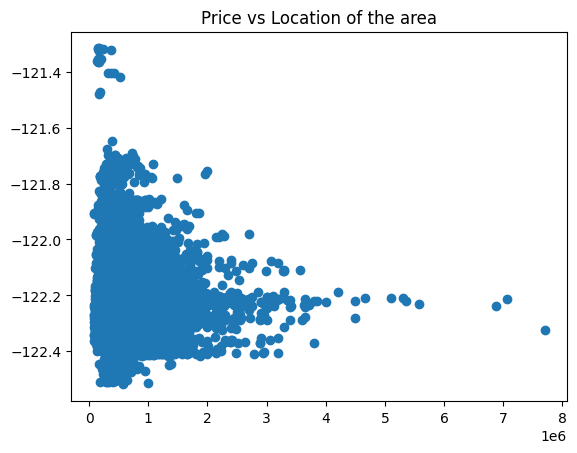
plt.title("Price vs square Feet")

Text(0.5, 1.0, 'Price vs square Feet') 

plt.scatter(data.price,data.long)

plt.title("Price vs Location of the area")

Text(0.5, 1.0, 'Price vs Location of the area')



plt.scatter(data.bedrooms,data.price)

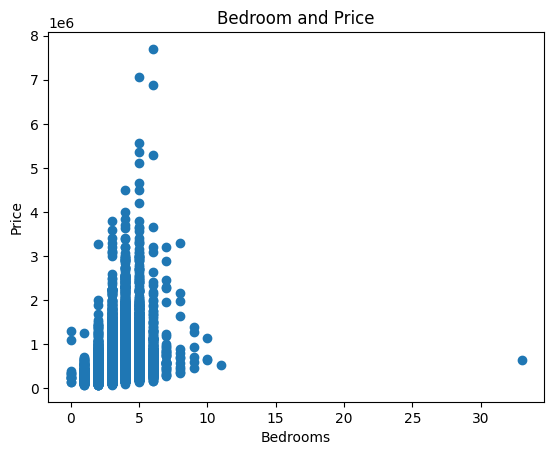
plt.title("Bedroom and Price")

plt.xlabel("Bedrooms")

plt.ylabel("Price")

plt.show()

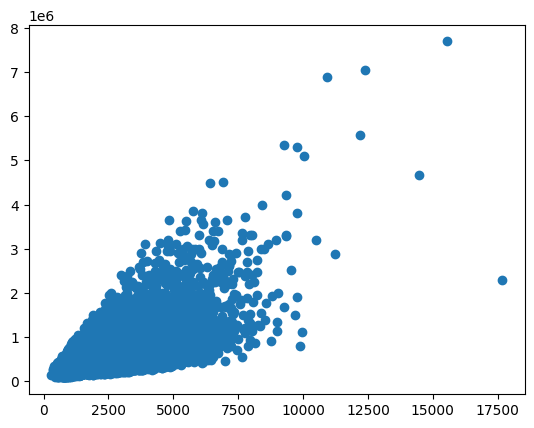
sns.despine



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

plt.scatter((data['sqft\_living']+data['sqft\_basement']),data['price'])

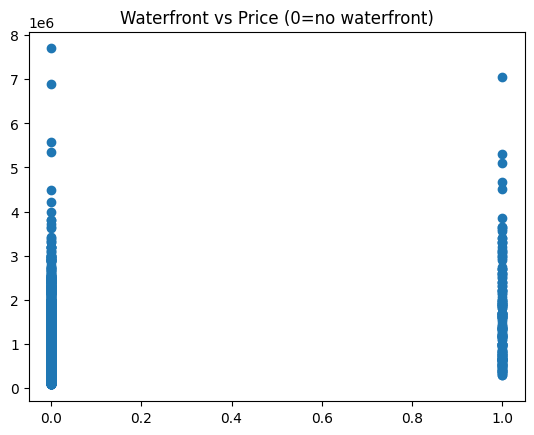
<matplotlib.collections.PathCollection at 0x1e69141a9f0>



plt.scatter(data.waterfront,data.price)

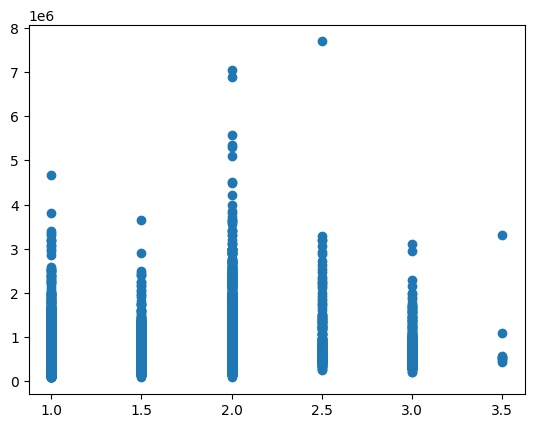
plt.title("Waterfront vs Price (0=no waterfront)")

Text(0.5, 1.0, 'Waterfront vs Price (0=no waterfront)')



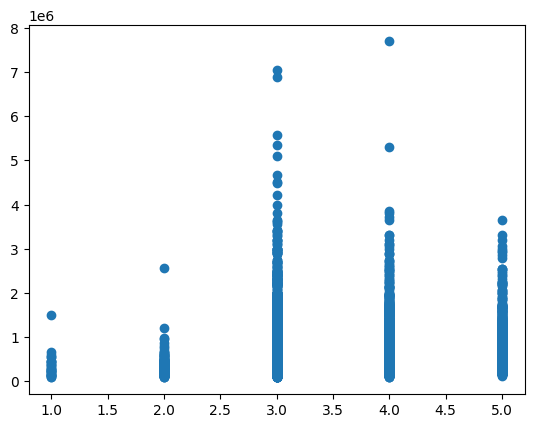
plt.scatter(data.floors,data.price)

<matplotlib.collections.PathCollection at 0x1e6ad2e8320>



plt.scatter(data.condition,data.price)

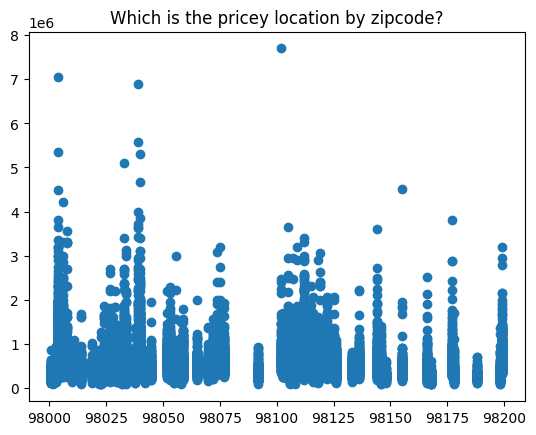
<matplotlib.collections.PathCollection at 0x1e6a8b07770>



plt.scatter(data.zipcode,data.price)

plt.title("Which is the pricey location by zipcode?")

Text(0.5, 1.0, 'Which is the pricey location by zipcode?')



##Prediction

from sklearn.linear\_model import LinearRegression

reg=LinearRegression()

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)

from sklearn.model\_selection import train\_test\_split

x\_train,x\_test,y\_train,y\_test=train\_test\_split(train1 , labels ,test\_size=0.10,random\_state=2)

reg.fit(x\_train,y\_train)

reg.score(x\_test,y\_test)

from sklearn import ensemble

params = {

    'n\_estimators': 400,

    'max\_depth': 5,

    'min\_samples\_split': 2,

    'learning\_rate': 0.1,

    'loss': 'squared\_error'

}

clf = ensemble.GradientBoostingRegressor(n\_estimators=400, max\_depth=5, min\_samples\_split=2,

                                         learning\_rate=0.1, loss='squared\_error')

clf.fit(x\_train, y\_train)

0.7320342760357805

 clf.score(x\_test,y\_test)

t\_sc = np.zeros((params['n\_estimators']),dtype=np.float64)

y\_pred = reg.predict(x\_test)

from sklearn.metrics import mean\_squared\_error

for i,y\_pred in enumerate(clf.staged\_predict(x\_test)):

    t\_sc[i]=mean\_squared\_error(y\_test,y\_pred)

testsc = np.arange((params['n\_estimators']))+1

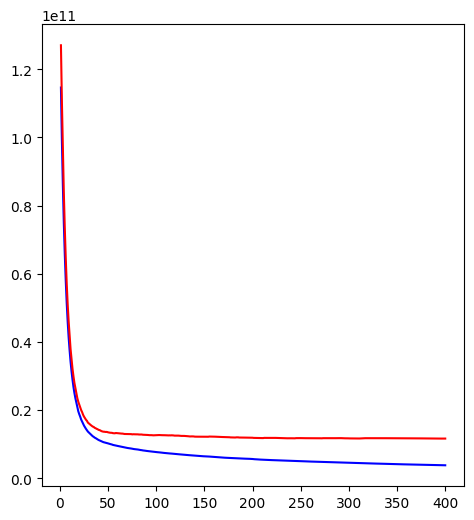
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)

plt.plot(testsc,clf.train\_score\_,'b-',label= 'Set dev train')

plt.plot(testsc,t\_sc,'r-',label = 'set dev test')

[<matplotlib.lines.Line2D at 0x1e6a8c4d8b0>]



from sklearn.preprocessing import scale

from sklearn.decomposition import PCA

pca = PCA()

pca.fit\_transform(scale(train1))

array([[-2.64785461e+00, -4.54699955e-02, -3.16665762e-01, ...,

-7.94687728e-02, -3.00457936e-16, 0.00000000e+00],

[-2.34485164e-01, 1.68297114e+00, -7.61521725e-01, ...,

9.81487761e-01, -1.89750938e-14, -0.00000000e+00],

[-2.57007792e+00, -6.14344122e-01, 3.49292423e-01, ...,

-1.38570764e-01, 3.34241218e-15, 0.00000000e+00],

...,

[-2.41985641e+00, -1.10027662e+00, -1.46293798e+00, ...,

9.66785881e-01, 1.07418809e-16, -0.00000000e+00],

[ 3.32183025e-01, -1.88043103e+00, -1.04412760e+00, ...,

-3.97449542e-01, -1.49949512e-18, 0.00000000e+00],

[-2.43180432e+00, -1.08505981e+00, -1.47248379e+00, ...,

9.53674385e-01, 7.17606664e-17, -0.00000000e+00]])