import libraries and read the data

In [1]:
```
#Import libraries
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(color_codes=True)
```

In [2]:
```
df=pd.read_csv('C:/Users/Consultant/Documents/machine-learning/regression/new_df.csv')
```

In [3]:
```
df
```

Out[3]:

	Unnamed: 0	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	sqrt_ft
0	3	21919321	4500000.0	85646	-111.035925	31.645878	636.67	8418.58	1930	7	5.0	9019.0
1	4	21306357	3411450.0	85750	-110.813768	32.285162	3.21	15393.00	1995	4	6.0	6396.0
2	5	21528016	3250000.0	85718	-110.910593	32.339090	1.67	27802.84	1999	3	4.0	6842.0
3	6	21610478	2400000.0	85712	-110.883315	32.261069	2.10	19038.42	2001	9	8.0	12025.0
4	7	21211741	2500000.0	85750	-110.861002	32.331603	1.07	21646.00	2011	6	8.0	8921.0
...	...	...	...	...	...	...	...	...	...	...	...	...
4777	4993	21908358	565000.0	85750	-110.820216	32.307646	0.83	4568.71	1986	4	3.0	2813.0
4778	4994	21909379	535000.0	85718	-110.922291	32.317496	0.18	4414.00	2002	3	2.0	2106.0
4779	4995	21810382	495000.0	85641	-110.661829	31.907917	4.98	2017.00	2005	5	3.0	3601.0
4780	4996	21908591	550000.0	85750	-110.858556	32.316373	1.42	4822.01	1990	4	3.0	2318.0

	Unnamed: 0	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	sqrt_ft
4781	4998	21900515	550000.0	85745	-111.055528	32.296871	1.01	5822.93	2009	4	4.0	3724.0

4782 rows × 19 columns

In [4]:
```
df= df.drop(df.columns[[0]], axis='columns')
```

In [5]:
```
df
```

Out[5]:

	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	sqrt_ft	garage	l
0	21919321	4500000.0	85646	-111.035925	31.645878	636.67	8418.58	1930	7	5.0	9019.0	4.0	
1	21306357	3411450.0	85750	-110.813768	32.285162	3.21	15393.00	1995	4	6.0	6396.0	3.0	
2	21528016	3250000.0	85718	-110.910593	32.339090	1.67	27802.84	1999	3	4.0	6842.0	3.0	
3	21610478	2400000.0	85712	-110.883315	32.261069	2.10	19038.42	2001	9	8.0	12025.0	4.0	
4	21211741	2500000.0	85750	-110.861002	32.331603	1.07	21646.00	2011	6	8.0	8921.0	4.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
4777	21908358	565000.0	85750	-110.820216	32.307646	0.83	4568.71	1986	4	3.0	2813.0	2.0	
4778	21909379	535000.0	85718	-110.922291	32.317496	0.18	4414.00	2002	3	2.0	2106.0	2.0	
4779	21810382	495000.0	85641	-110.661829	31.907917	4.98	2017.00	2005	5	3.0	3601.0	3.0	
4780	21908591	550000.0	85750	-110.858556	32.316373	1.42	4822.01	1990	4	3.0	2318.0	3.0	

	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	sqrt_ft	garage	
4781	21900515	550000.0	85745	-111.055528	32.296871	1.01	5822.93	2009	4	4.0	3724.0	3.0	

4782 rows × 18 columns

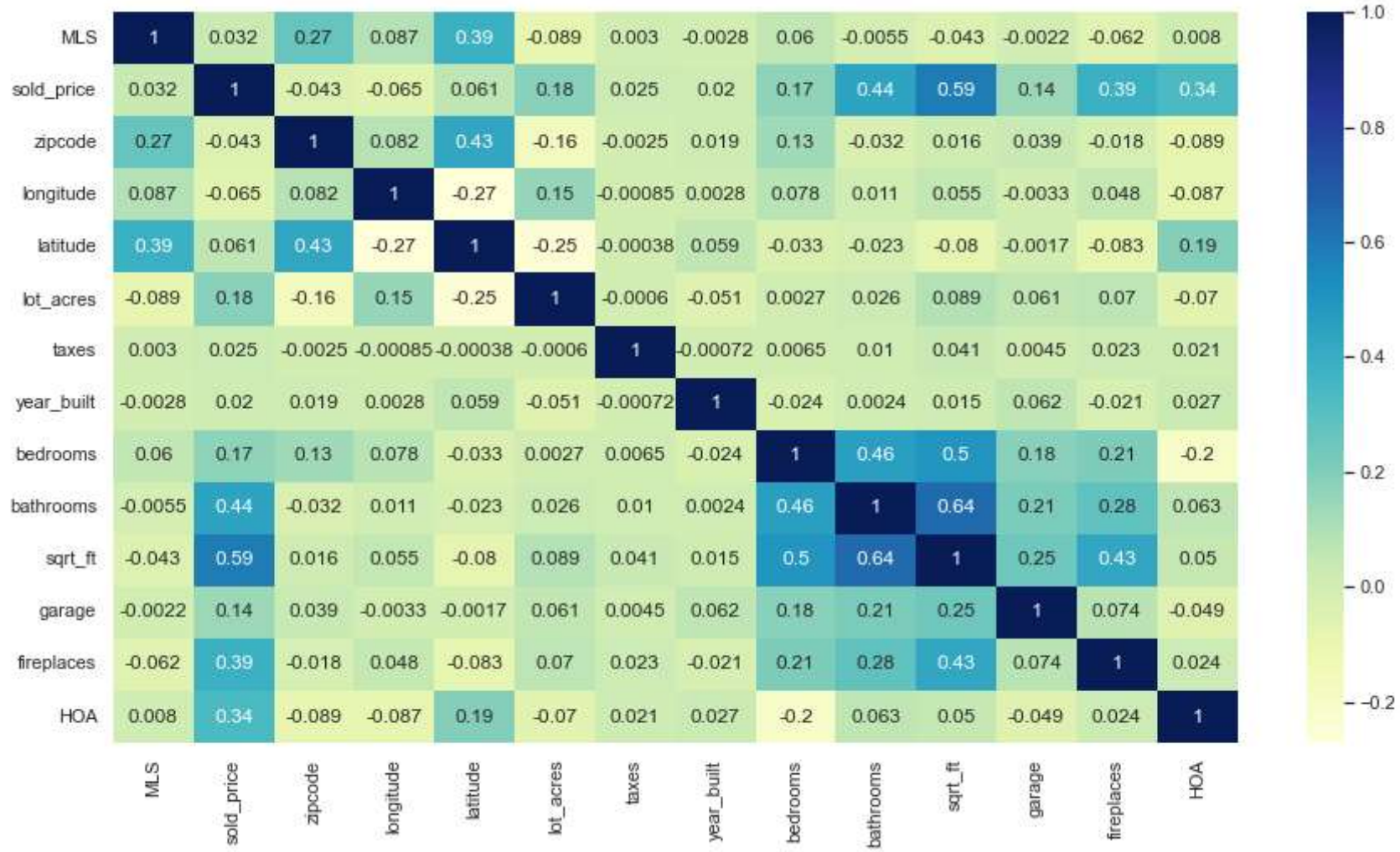
In [6]:
```
df.describe()
```

Out[6]:

	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bath
count	4.782000e+03	4.782000e+03	4782.000000	4782.000000	4782.000000	4782.000000	4.782000e+03	4782.000000	4782.000000	4782.0
mean	2.138991e+07	7.728091e+05	85724.498327	-110.913257	32.317169	2.737066	9.551398e+03	1993.933292	3.899624	3.8
std	1.898486e+06	2.987144e+05	36.319366	0.117324	0.166950	14.939447	1.768330e+05	59.852681	0.867482	1.1
min	3.047349e+06	1.690000e+05	85118.000000	-112.520168	31.361562	0.000000	0.000000e+00	0.000000	1.000000	1.0
25%	2.140806e+07	5.871250e+05	85718.000000	-110.980234	32.283876	0.590000	4.856648e+03	1989.000000	3.000000	3.0
50%	2.161509e+07	6.797500e+05	85737.000000	-110.922933	32.320090	0.990000	6.280395e+03	2000.000000	4.000000	4.0
75%	2.180485e+07	8.400000e+05	85750.000000	-110.859118	32.398420	1.670000	8.144557e+03	2006.000000	4.000000	4.0
max	2.192856e+07	4.500000e+06	86323.000000	-109.454637	34.927884	636.670000	1.221508e+07	2019.000000	10.000000	36.0

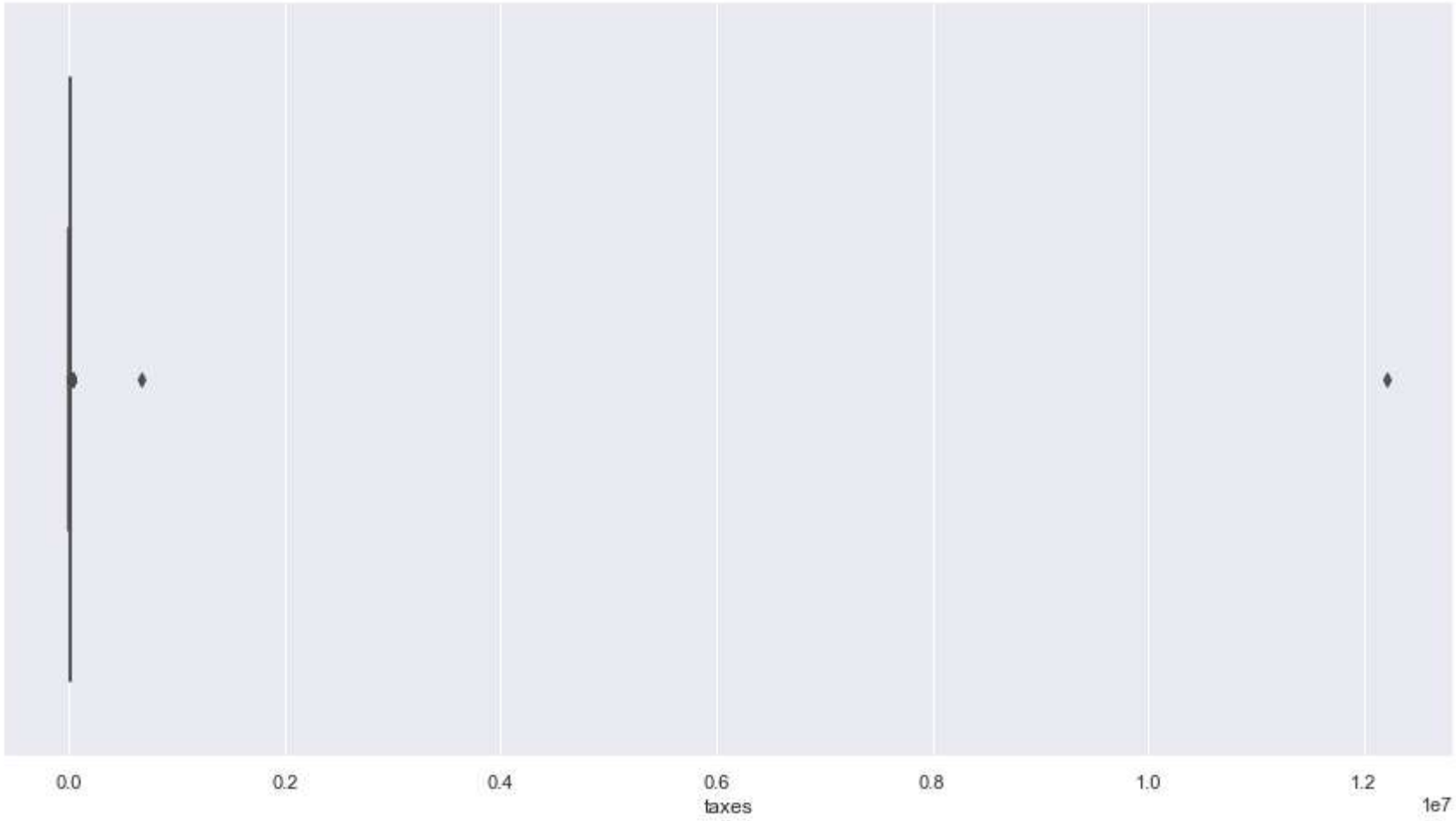
In [7]:
```
sns.set(rc={"figure.figsize":(15, 8)})
dfplot1 = sns.heatmap(df.corr(), cmap="YlGnBu", annot=True)
dfplot1
```

Out[7]: <AxesSubplot:>



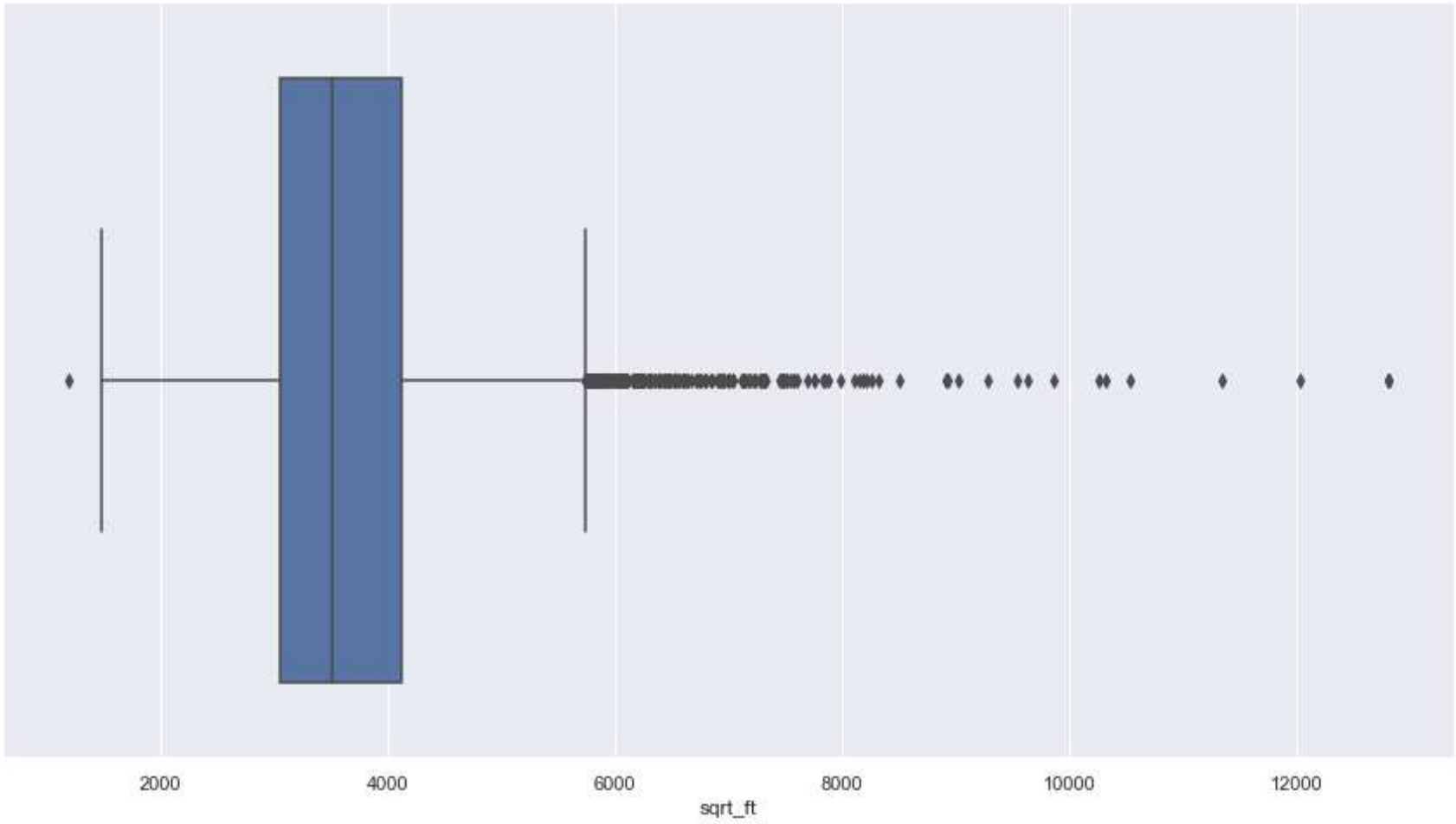
In [8]:
```
sns.boxplot(x=df['taxes'])
```

Out[8]: <AxesSubplot:xlabel='taxes'>



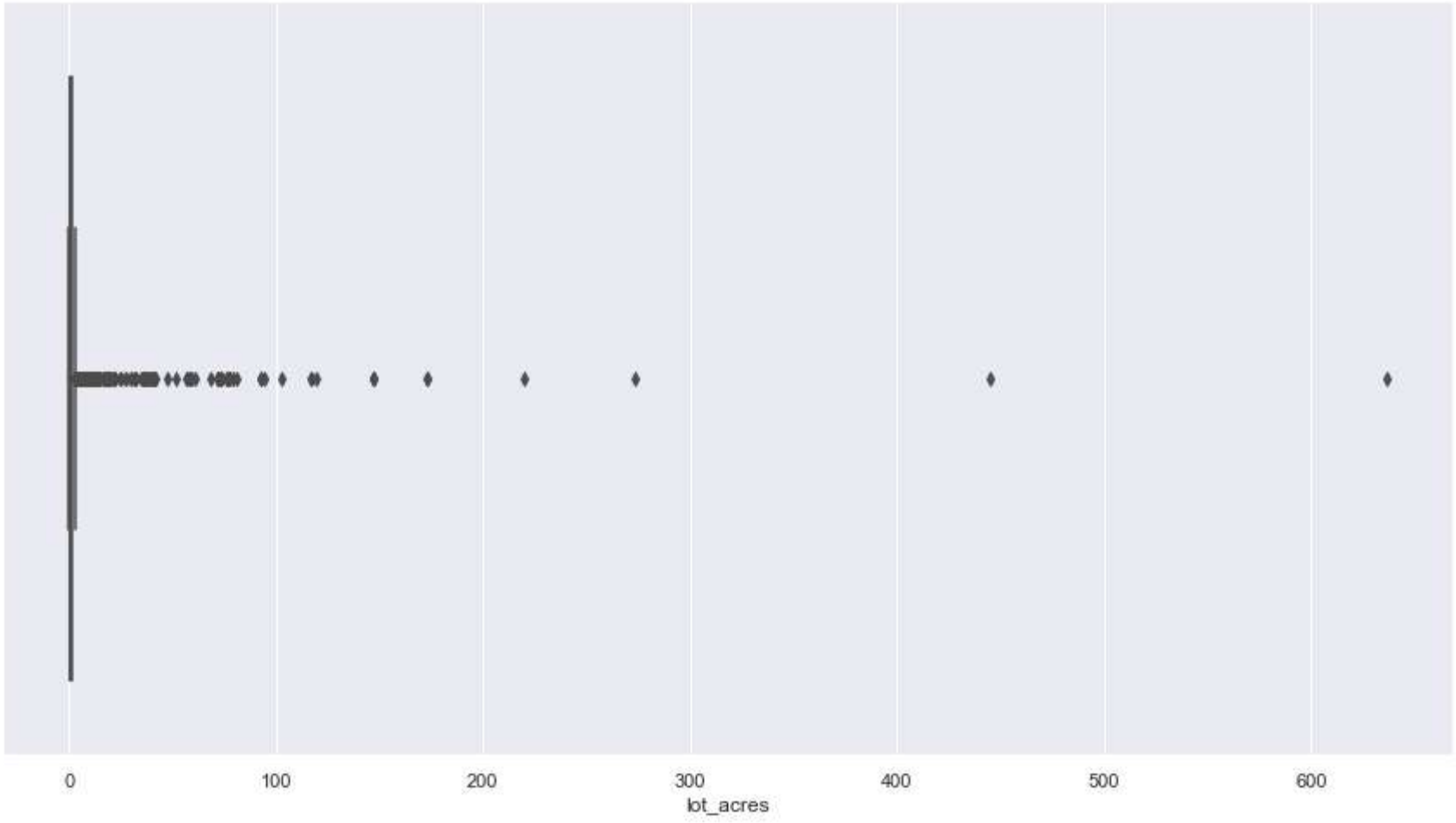
In [9]:
```
sns.boxplot(x=df['sqrt_ft'])
```

Out[9]:
```
<AxesSubplot:xlabel='sqrt_ft'>
```



In [10]:
```
sns.boxplot(x=df['lot_acres'])
```

Out[10]:
```
<AxesSubplot:xlabel='lot_acres'>
```



In [11]:
```
df.shape
```

Out[11]:
```
(4782, 18)
```

In [12]:
```
df = df.drop(df[df['taxes']==df['taxes'].max()].index)
```

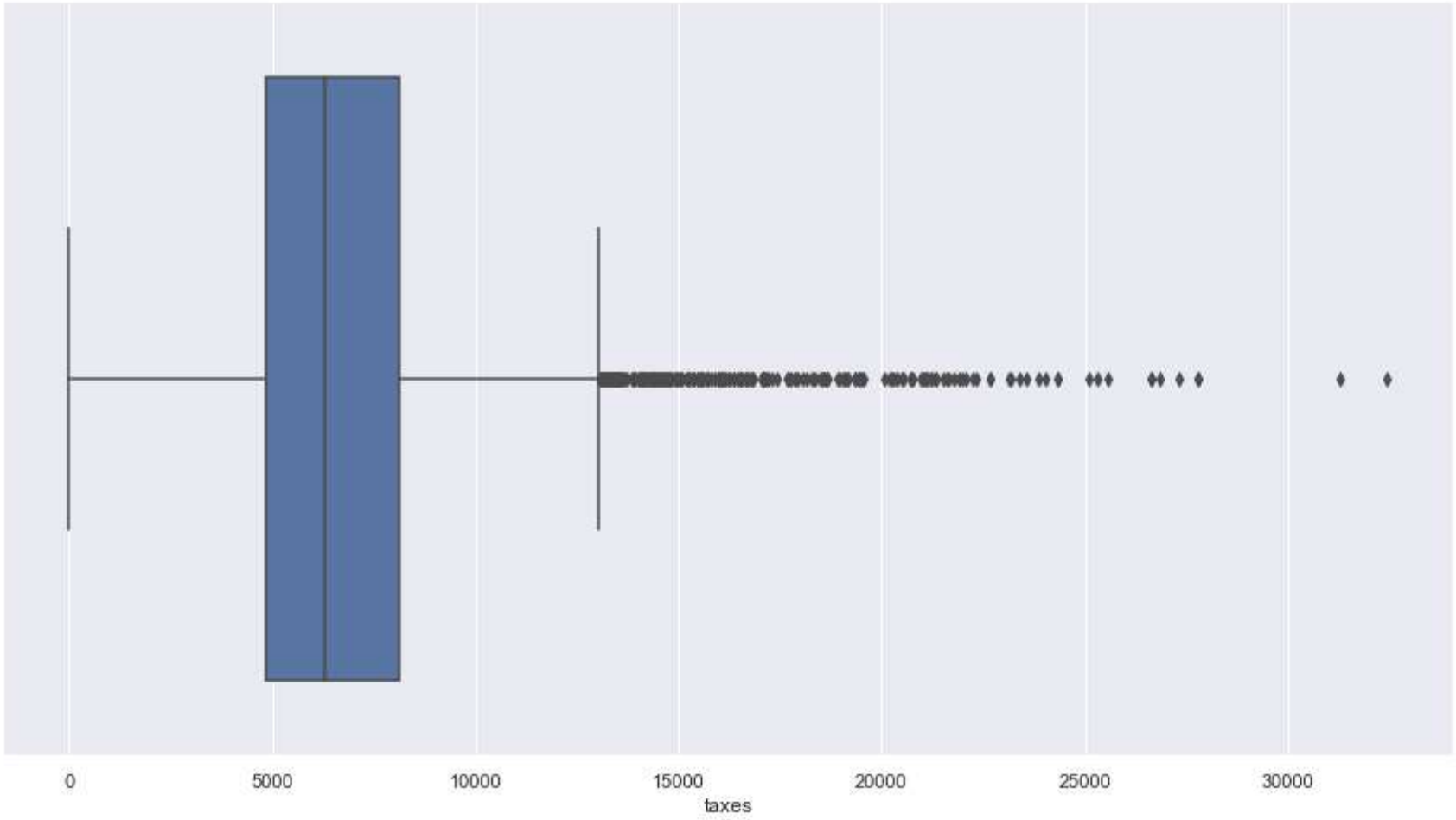
In [13]:
```
df = df.drop(df[df['taxes']==df['taxes'].max()].index)
```

In [14]:
```
df.shape
```

Out[14]:
```
(4780, 18)
```

In [15]:
```
sns.boxplot(x=df['taxes'])
```

Out[15]:
```
<AxesSubplot:xlabel='taxes'>
```



In [16]:
```
df['taxes'].max()
```

Out[16]:
```
32442.22
```

In [17]:
```
df = df.drop(df[df['sqrt_ft']>8000].index)
```

In [18]:
```
df
```

Out[18]:

	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	sqrt_ft	garage	kitch
1	21306357	3411450.0	85750	-110.813768	32.285162	3.21	15393.00	1995	4	6.0	6396.0	3.0	Refr
2	21528016	3250000.0	85718	-110.910593	32.339090	1.67	27802.84	1999	3	4.0	6842.0	3.0	Refr
5	21324646	3700000.0	85718	-110.912156	32.343601	6.73	25094.39	2002	5	7.0	5238.0	3.0	Free
6	21812010	3250000.0	85750	-110.837950	32.327575	3.53	18936.11	2007	5	6.0	6480.0	3.0	El
7	21900396	2776518.0	85640	-111.045441	31.562121	147.18	7330.36	1935	5	5.0	5067.0	5.0	Apı
...	...	...	...	...	...	...	...	...	...	...	...	...	
4777	21908358	565000.0	85750	-110.820216	32.307646	0.83	4568.71	1986	4	3.0	2813.0	2.0	El
4778	21909379	535000.0	85718	-110.922291	32.317496	0.18	4414.00	2002	3	2.0	2106.0	2.0	El
4779	21810382	495000.0	85641	-110.661829	31.907917	4.98	2017.00	2005	5	3.0	3601.0	3.0	Dis
4780	21908591	550000.0	85750	-110.858556	32.316373	1.42	4822.01	1990	4	3.0	2318.0	3.0	El

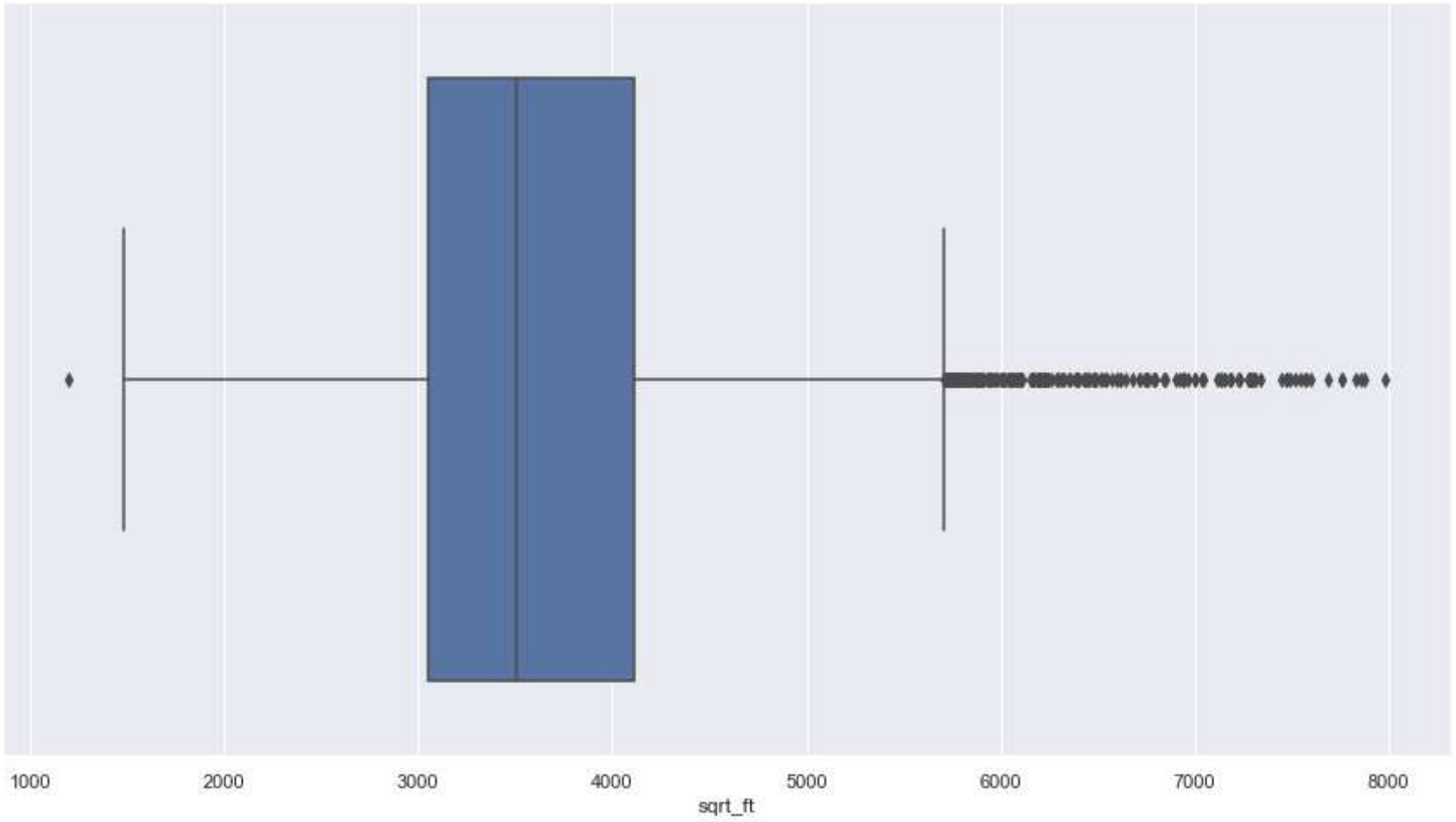
	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	sqrt_ft	garage	kitch
4781	21900515	550000.0	85745	-111.055528	32.296871	1.01	5822.93	2009	4	4.0	3724.0	3.0	

Dis

4757 rows × 18 columns

In [19]:
```
sns.boxplot(x=df['sqrt_ft'])
```

Out[19]:
```
<AxesSubplot:xlabel='sqrt_ft'>
```



In [20]:
```
df = df.drop(df[df['lot_acres']>200].index)
```

In [21]:
```
df
```

Out[21]:

	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	sqrt_ft	garage	kitch
1	21306357	3411450.0	85750	-110.813768	32.285162	3.21	15393.00	1995	4	6.0	6396.0	3.0	Refr
2	21528016	3250000.0	85718	-110.910593	32.339090	1.67	27802.84	1999	3	4.0	6842.0	3.0	Refr
5	21324646	3700000.0	85718	-110.912156	32.343601	6.73	25094.39	2002	5	7.0	5238.0	3.0	Free
6	21812010	3250000.0	85750	-110.837950	32.327575	3.53	18936.11	2007	5	6.0	6480.0	3.0	El
7	21900396	2776518.0	85640	-111.045441	31.562121	147.18	7330.36	1935	5	5.0	5067.0	5.0	Apı
...	...	...	...	...	...	...	...	...	...	...	...	...	
4777	21908358	565000.0	85750	-110.820216	32.307646	0.83	4568.71	1986	4	3.0	2813.0	2.0	El
4778	21909379	535000.0	85718	-110.922291	32.317496	0.18	4414.00	2002	3	2.0	2106.0	2.0	El
4779	21810382	495000.0	85641	-110.661829	31.907917	4.98	2017.00	2005	5	3.0	3601.0	3.0	Dis
4780	21908591	550000.0	85750	-110.858556	32.316373	1.42	4822.01	1990	4	3.0	2318.0	3.0	El

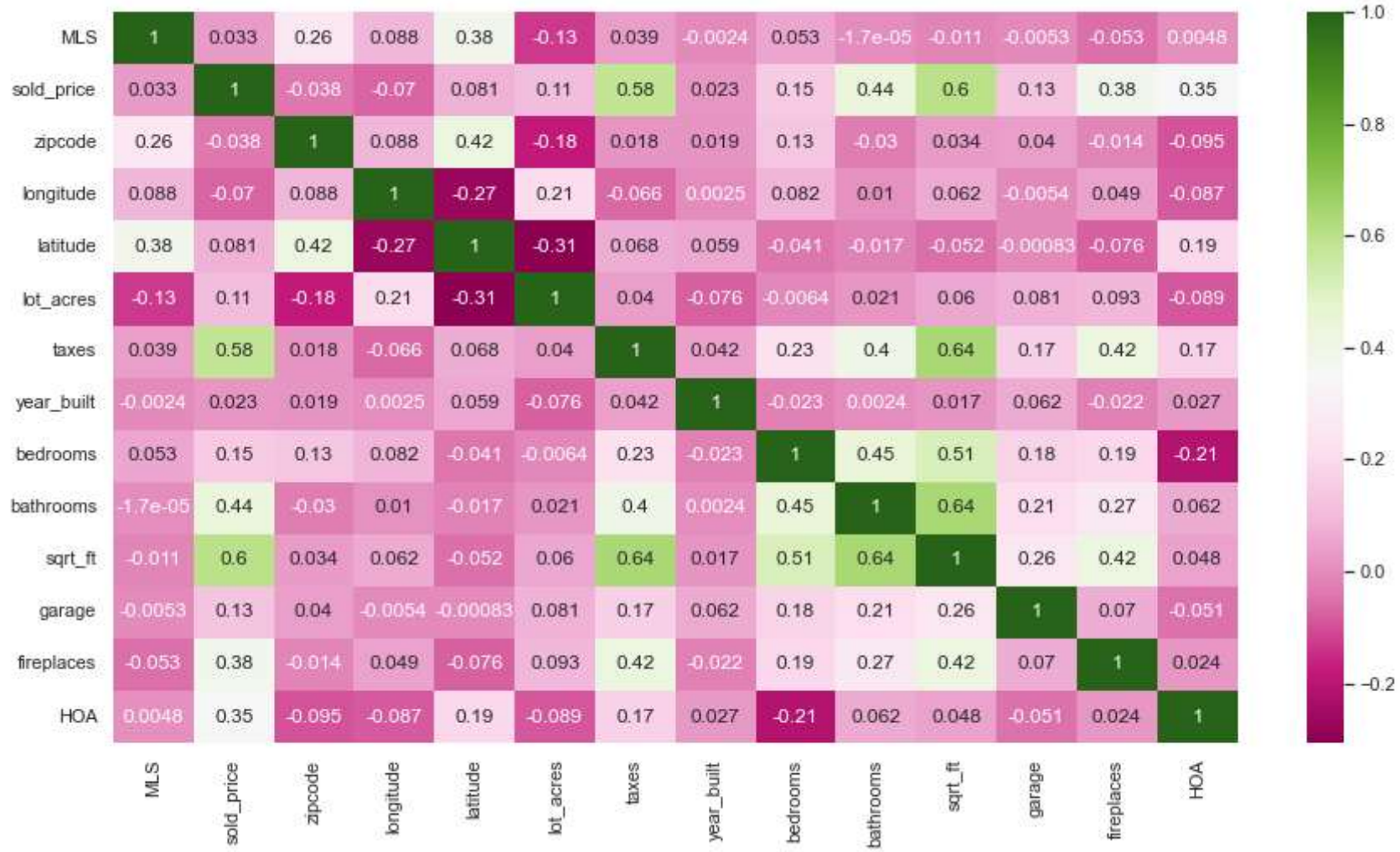
	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	sqrt_ft	garage	kitch
4781	21900515	550000.0	85745	-111.055528	32.296871	1.01	5822.93	2009	4	4.0	3724.0	3.0	

Dis

4754 rows × 18 columns

In [22]:
```
sns.set(rc={"figure.figsize":(15, 8)})
dfplot1 = sns.heatmap(df.corr(), cmap="PiYG", annot=True)
dfplot1
```

Out[22]: <AxesSubplot:>



In [23]:
```
df['year_built'].value_counts()
```

Out[23]:
```
2006    243
2002    233
2005    227
2007    225
2004    198
       ...
1900      1
1926      1
1919      1
1914      1
1939      1
Name: year_built, Length: 102, dtype: int64
```

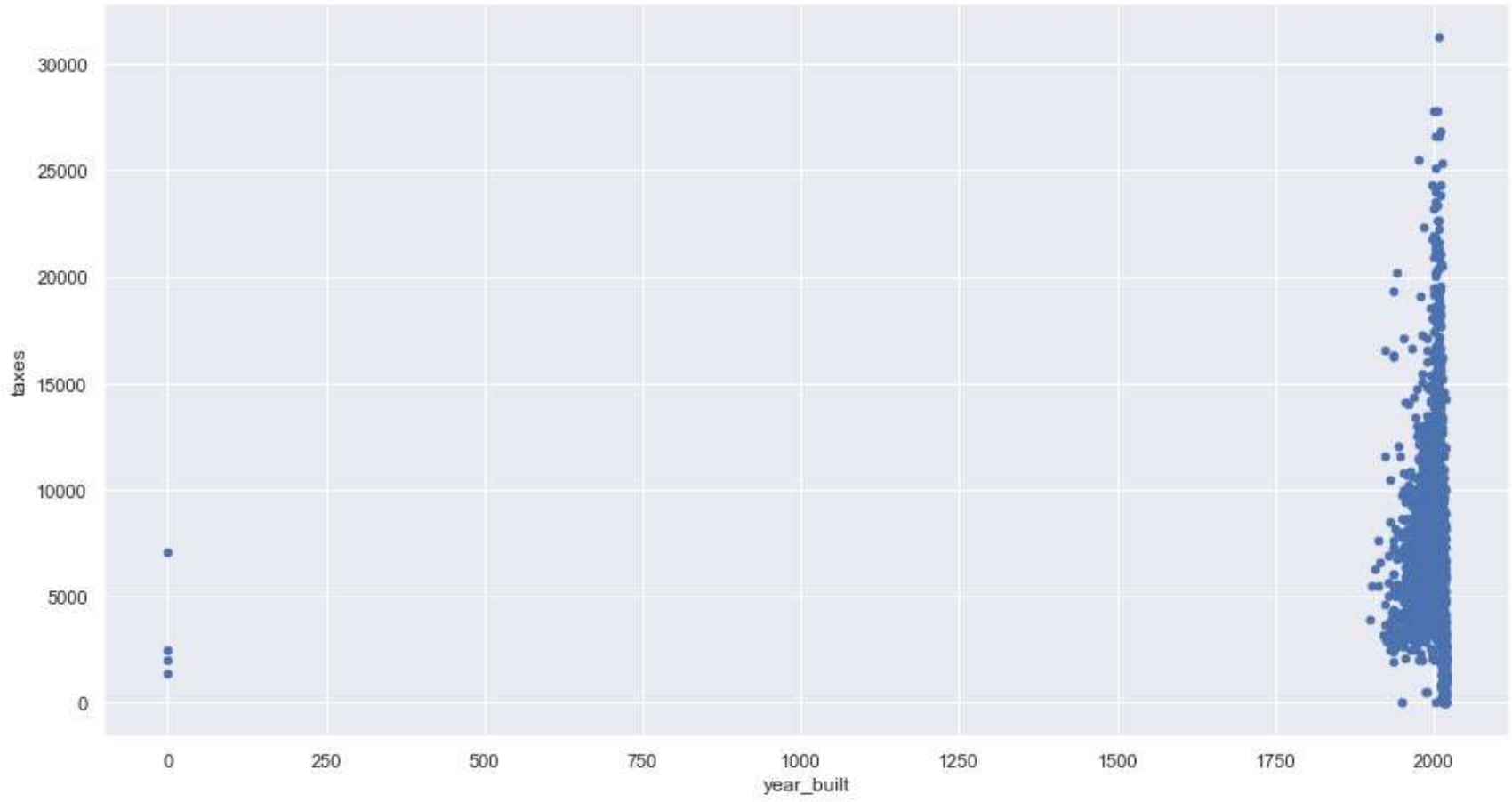
In [24]:
```
df['MLS'].nunique()
```

Out[24]:
```
4754
```

In [25]:
```
df.plot(kind='scatter',x='year_built',y='taxes')
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will h
ave precedence in case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide
a 2D array with a single row if you intend to specify the same RGB or RGBA value for all points.
```

Out[25]:
```
<AxesSubplot:xlabel='year_built', ylabel='taxes'>
```



In [26]:
```
df.loc[df['year_built'] ==0]
```

Out[26]:

	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	sqrt_ft	garage	kitche
146	21207587	1210520.0	85658	-111.101588	32.468488	0.33	1412.75	0	4	5.0	3334.0	2.0	I F Micro
910	21608590	695000.0	85645	-111.183593	31.702330	72.00	2480.58	0	4	4.0	2272.0	2.0	I F
1387	21702126	877170.0	85755	-110.977158	32.459429	0.60	2005.57	0	3	4.0	3078.0	3.0	I Micro
1707	21719857	700000.0	85701	-110.963672	32.216996	0.13	7059.56	0	3	3.0	2500.0	3.0	I Ele

df['sqrt_ft']=np.where(df['sqrt_ft']==0,df['sqrt_ft'].replace(mode),df['sqrt_ft'])

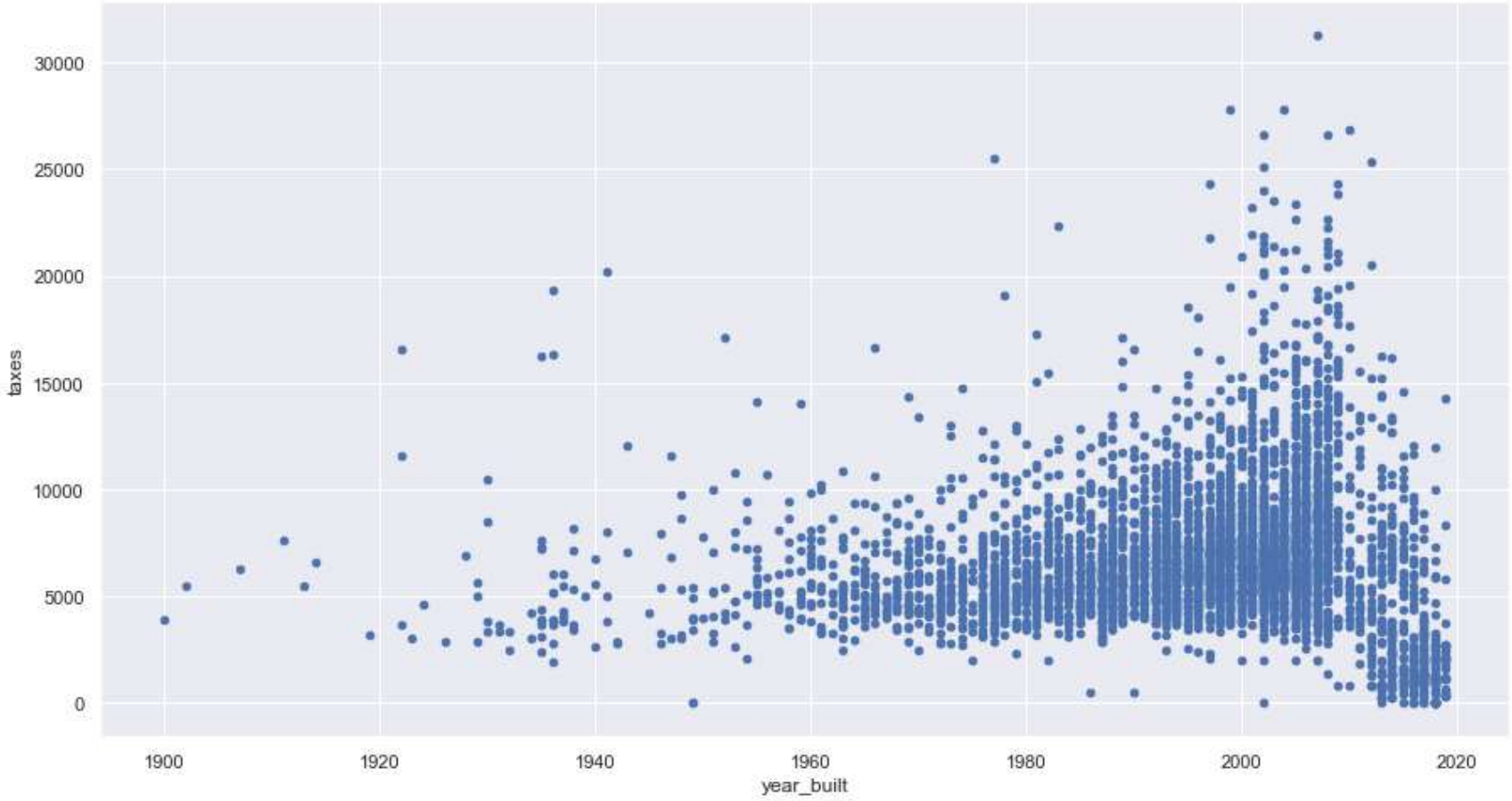
In [27]:
```
year_min=df['year_built'].min()
```

In [28]:
```
df['year_built']=np.where(df['year_built']==0,df['year_built'].replace(year_min),df['year_built'])
```

In [29]:
```
df.plot(kind='scatter',x='year_built',y='taxes')
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will h
ave precedence in case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide
a 2D array with a single row if you intend to specify the same RGB or RGBA value for all points.
```

Out[29]: <AxesSubplot:xlabel='year_built', ylabel='taxes'>

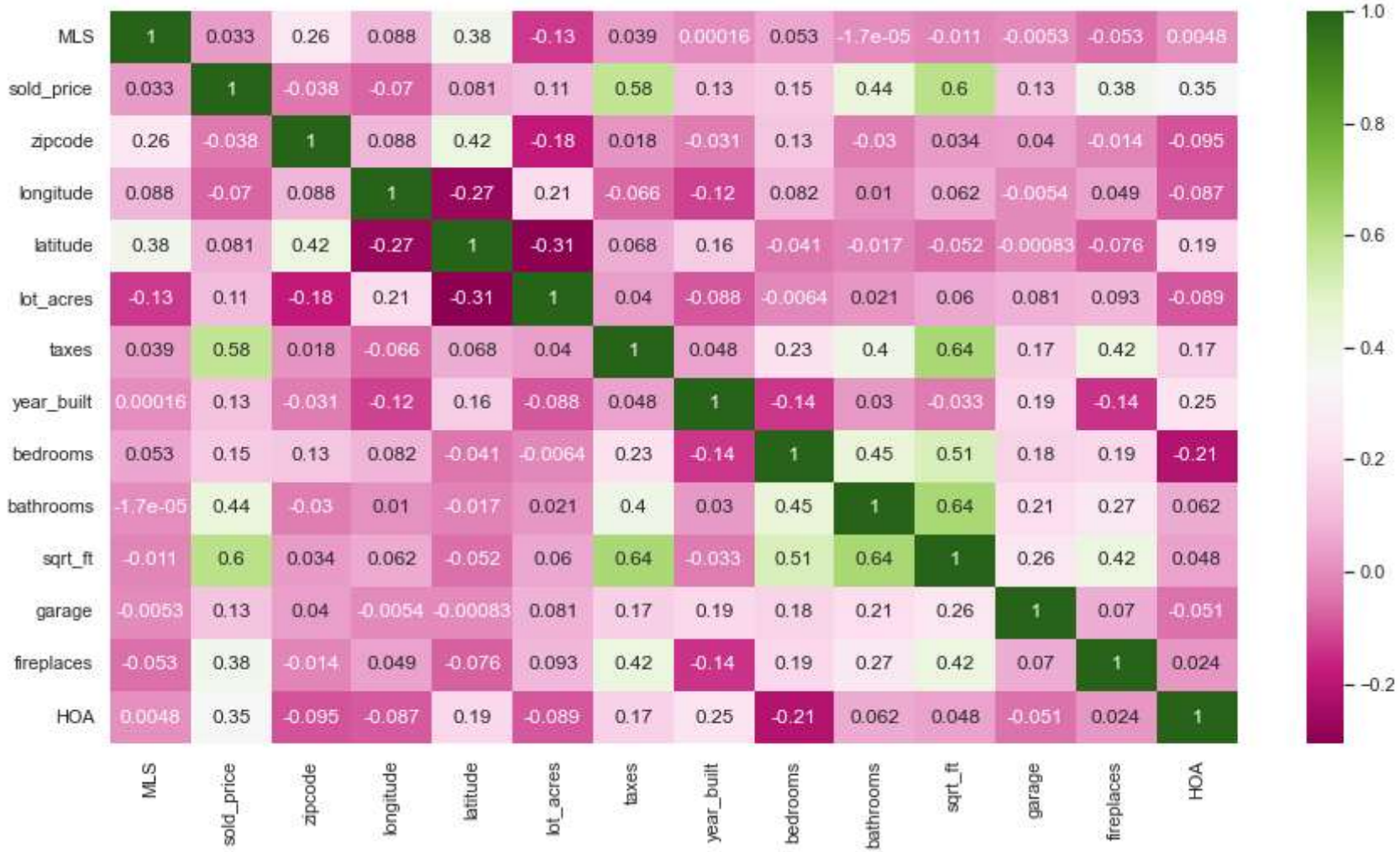


In [30]:
```
df.shape
```

Out[30]: (4754, 18)

In [31]:
```
sns.set(rc={"figure.figsize":(15, 8)})
dfplot1 = sns.heatmap(df.corr(), cmap="PiYG", annot=True)
dfplot1
```

Out[31]: <AxesSubplot:>



In [32]:
```
df= df.drop(df.columns[[0]], axis='columns')
```

In [33]:
```
df
```

Out[33]:

	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	sqrt_ft	garage	kitchen_featu
1	3411450.0	85750	-110.813768	32.285162	3.21	15393.00	1995	4	6.0	6396.0	3.0	Dishwas Garb Dispo Refrigerator, I
2	3250000.0	85718	-110.910593	32.339090	1.67	27802.84	1999	3	4.0	6842.0	3.0	Dishwas Garb Dispo Refrigerator, I
5	3700000.0	85718	-110.912156	32.343601	6.73	25094.39	2002	5	7.0	5238.0	3.0	Compa Dishwas Freezer, Garb Dispo
6	3250000.0	85750	-110.837950	32.327575	3.53	18936.11	2007	5	6.0	6480.0	3.0	Dishwas Double S Electric Ra Free
7	2776518.0	85640	-111.045441	31.562121	147.18	7330.36	1935	5	5.0	5067.0	5.0	Free Refrigera Appliance Co Sta
...	...	...	...	...	...	...	...	...	...	...	...	
4777	565000.0	85750	-110.820216	32.307646	0.83	4568.71	1986	4	3.0	2813.0	2.0	Dishwas Double S Electric Ra Garl
4778	535000.0	85718	-110.922291	32.317496	0.18	4414.00	2002	3	2.0	2106.0	2.0	Dishwas Double S Electric Ra Garl
4779	495000.0	85641	-110.661829	31.907917	4.98	2017.00	2005	5	3.0	3601.0	3.0	Dishwas Double S Garb Disposal, Ga
4780	550000.0	85750	-110.858556	32.316373	1.42	4822.01	1990	4	3.0	2318.0	3.0	Dishwas Double S Electric Ra Garl

	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	sqrt_ft	garage	kitchen_featu
4781	550000.0	85745	-111.055528	32.296871	1.01	5822.93	2009	4	4.0	3724.0	3.0	Dishwas Double S Garb Disposal, Ga

4754 rows × 17 columns

In [34]:
```
df = df.drop(['kitchen_features','kitchen_features','kitchen_vectors','floor_vectors'], axis=1)
df.head(5)
```

Out[34]:

	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	sqrt_ft	garage	fireplaces	floor_c
1	3411450.0	85750	-110.813768	32.285162	3.21	15393.00	1995	4	6.0	6396.0	3.0	5.0	C
2	3250000.0	85718	-110.910593	32.339090	1.67	27802.84	1999	3	4.0	6842.0	3.0	5.0	Natura Wood
5	3700000.0	85718	-110.912156	32.343601	6.73	25094.39	2002	5	7.0	5238.0	3.0	1.0	Carpet, Stone
6	3250000.0	85750	-110.837950	32.327575	3.53	18936.11	2007	5	6.0	6480.0	3.0	2.0	C
7	2776518.0	85640	-111.045441	31.562121	147.18	7330.36	1935	5	5.0	5067.0	5.0	5.0	C Natura

In [35]:
```
df = df.drop(['floor_covering'], axis=1)
df.head(5)
```

Out[35]:

	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	sqrt_ft	garage	fireplaces	HOA
1	3411450.0	85750	-110.813768	32.285162	3.21	15393.00	1995	4	6.0	6396.0	3.0	5.0	55.00
2	3250000.0	85718	-110.910593	32.339090	1.67	27802.84	1999	3	4.0	6842.0	3.0	5.0	422.00
5	3700000.0	85718	-110.912156	32.343601	6.73	25094.39	2002	5	7.0	5238.0	3.0	1.0	421.00
6	3250000.0	85750	-110.837950	32.327575	3.53	18936.11	2007	5	6.0	6480.0	3.0	2.0	141.67
7	2776518.0	85640	-111.045441	31.562121	147.18	7330.36	1935	5	5.0	5067.0	5.0	5.0	0.00

In [36]:
```
df.describe()
```

Out[36]:

	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	s
count	4.754000e+03	4754.000000	4754.000000	4754.000000	4754.000000	4754.000000	4754.000000	4754.000000	4754.000000	4754.00
mean	7.688585e+05	85724.643879	-110.913500	32.318397	2.370038	6807.774794	1995.625999	3.892512	3.788704	3678.10
std	2.886162e+05	36.214401	0.117124	0.164705	8.426050	3393.148877	15.896500	0.850004	1.084433	938.24
min	1.690000e+05	85118.000000	-112.520168	31.361562	0.000000	0.000000	1900.000000	1.000000	1.000000	1484.00
25%	5.850000e+05	85718.000000	-110.980423	32.284408	0.590000	4849.935000	1989.000000	3.000000	3.000000	3049.00
50%	6.779500e+05	85737.000000	-110.923244	32.320212	0.980000	6261.000000	2000.000000	4.000000	4.000000	3511.50
75%	8.350000e+05	85750.000000	-110.859118	32.398978	1.657500	8084.490000	2006.000000	4.000000	4.000000	4113.75
max	3.700000e+06	86323.000000	-109.454637	34.927884	172.760000	31275.000000	2019.000000	10.000000	36.000000	7983.00

In [37]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4754 entries, 1 to 4781
Data columns (total 13 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   sold_price  4754 non-null   float64
 1   zipcode     4754 non-null   int64  
 2   longitude   4754 non-null   float64
 3   latitude    4754 non-null   float64
 4   lot_acres   4754 non-null   float64
 5   taxes       4754 non-null   float64
 6   year_built  4754 non-null   int64  
 7   bedrooms    4754 non-null   int64  
 8   bathrooms   4754 non-null   float64
 9   sqrt_ft     4754 non-null   float64
 10  garage      4754 non-null   float64
 11  fireplaces  4754 non-null   float64
 12  HOA         4754 non-null   float64
dtypes: float64(10), int64(3)
memory usage: 520.0 KB
```

Regression

In [38]:
```
y=df['taxes']
```

In [39]:
```
y
```

Out[39]:
```
1       15393.00
2       27802.84
5       25094.39
6       18936.11
7        7330.36
           ...
4777     4568.71
4778     4414.00
4779     2017.00
4780     4822.01
4781     5822.93
Name: taxes, Length: 4754, dtype: float64
```

In [40]:
```
X=df[['lot_acres','year_built','sqrt_ft']]
```

In [41]:
```
X.head()
```

Out[41]:

	lot_acres	year_built	sqrt_ft
1	3.21	1995	6396.0
2	1.67	1999	6842.0
5	6.73	2002	5238.0
6	3.53	2007	6480.0
7	147.18	1935	5067.0

Normalize X

In [42]:
```
X=(X- X.min()) / ( X.max() - X.min())
```

In [43]:
```
X
```

Out[43]:

	lot_acres	year_built	sqrt_ft
1	0.018581	0.798319	0.755809
2	0.009667	0.831933	0.824435
5	0.038956	0.857143	0.577627
6	0.020433	0.899160	0.768734
7	0.851933	0.294118	0.551316
...	...	...	...
4777	0.004804	0.722689	0.204493
4778	0.001042	0.857143	0.095707
4779	0.028826	0.882353	0.325742
4780	0.008219	0.756303	0.128327
4781	0.005846	0.915966	0.344668

4754 rows × 3 columns

In [44]:
```
X.max()
```

Out[44]:
```
lot_acres     1.0
year_built    1.0
sqrt_ft       1.0
dtype: float64
```

In [45]:
```
X.min()
```

Out[45]:
```
lot_acres     0.0
year_built    0.0
sqrt_ft       0.0
dtype: float64
```

In [46]:
```
X= X.to_numpy()
y= y.to_numpy()
```

In [47]:
```
X
```

Out[47]:
```
array([[0.01858069, 0.79831933, 0.75580859],
       [0.00966659, 0.83193277, 0.82443453],
       [0.03895578, 0.85714286, 0.57762733],
       ...,
       [0.02882612, 0.88235294, 0.32574242],
       [0.0082195 , 0.75630252, 0.12832743],
       [0.00584626, 0.91596639, 0.34466841]])
```

In [48]:
```
class MultipleLinearRegression():
  def fit(self,X,y):
    self.W = np.linalg.solve(X.T@X,X.T@y)

  def predict(self,X):

    return np.matmul(X,self.W)
```

In [49]:
```
lr_mul=MultipleLinearRegression()
```

In [50]:
```
lr_mul.fit(X,y)
```

In [51]:
```
y_hat=lr_mul.predict(X)
```

In [52]:
```
plt.figure()
plt.scatter(y,y_hat,s=8)
plt.plot(y,y,color="#FF0070")
```

Out[52]:
```
[<matplotlib.lines.Line2D at 0x1fddb4a6940>]
```

In [53]:
```
def R2 (Y,Y_hat):
  return (1-(np.sum((Y-Y_hat)**2)/np.sum((Y-np.mean(Y))**2)))
```

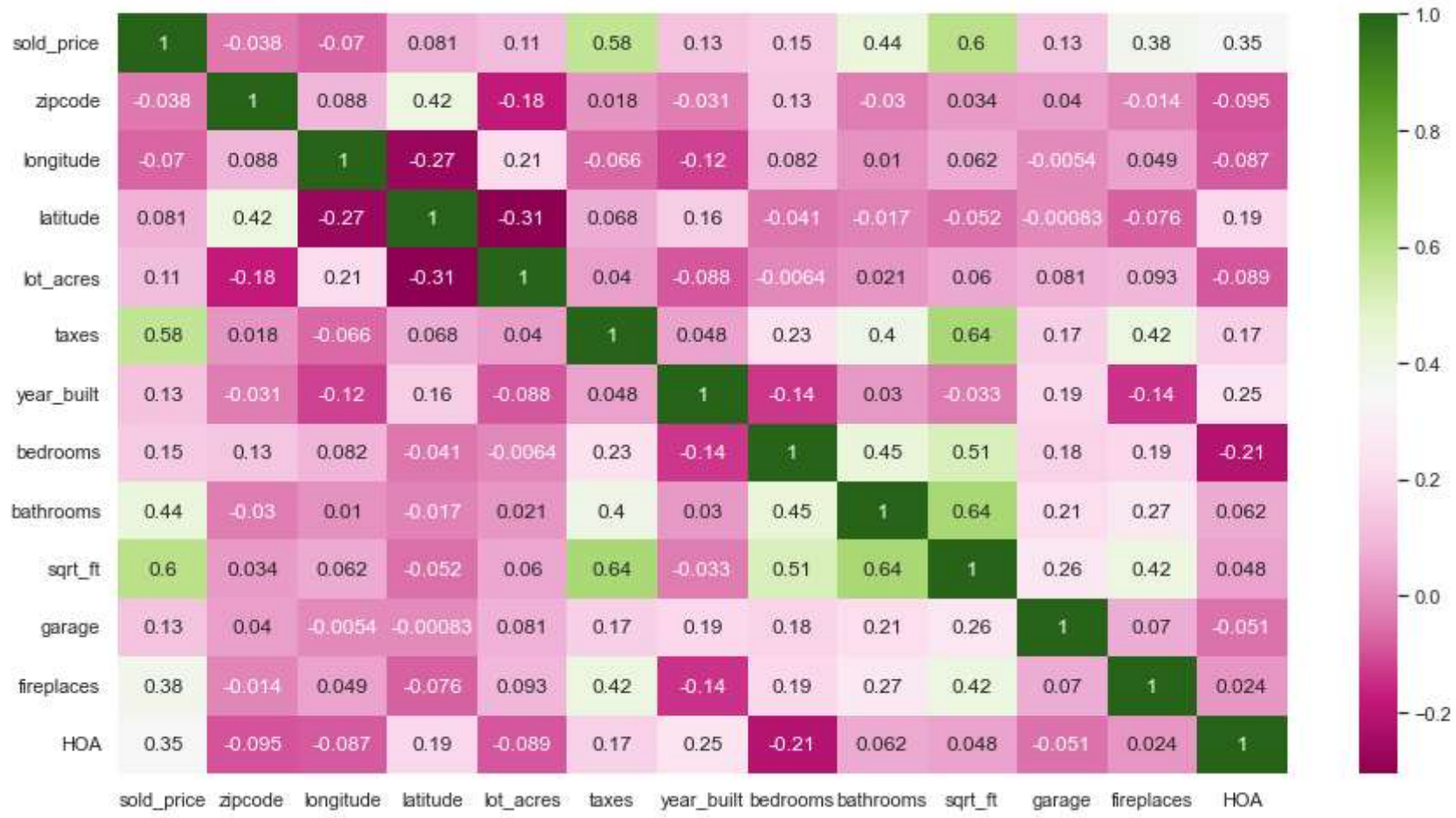
In [54]:
```
R2(y,y_hat)
```

Out[54]:
```
0.40831671249493773
```

In [55]:
```
## Change the independent variables, chose variables whit more correlation
```

In [56]:
```
sns.set(rc={"figure.figsize":(15, 8)})
dfplot1 = sns.heatmap(df.corr(), cmap="PiYG", annot=True)
dfplot1
```

Out[56]: <AxesSubplot:>



Now,we are gonna run the model with the most correlated variables

First delete the outliers

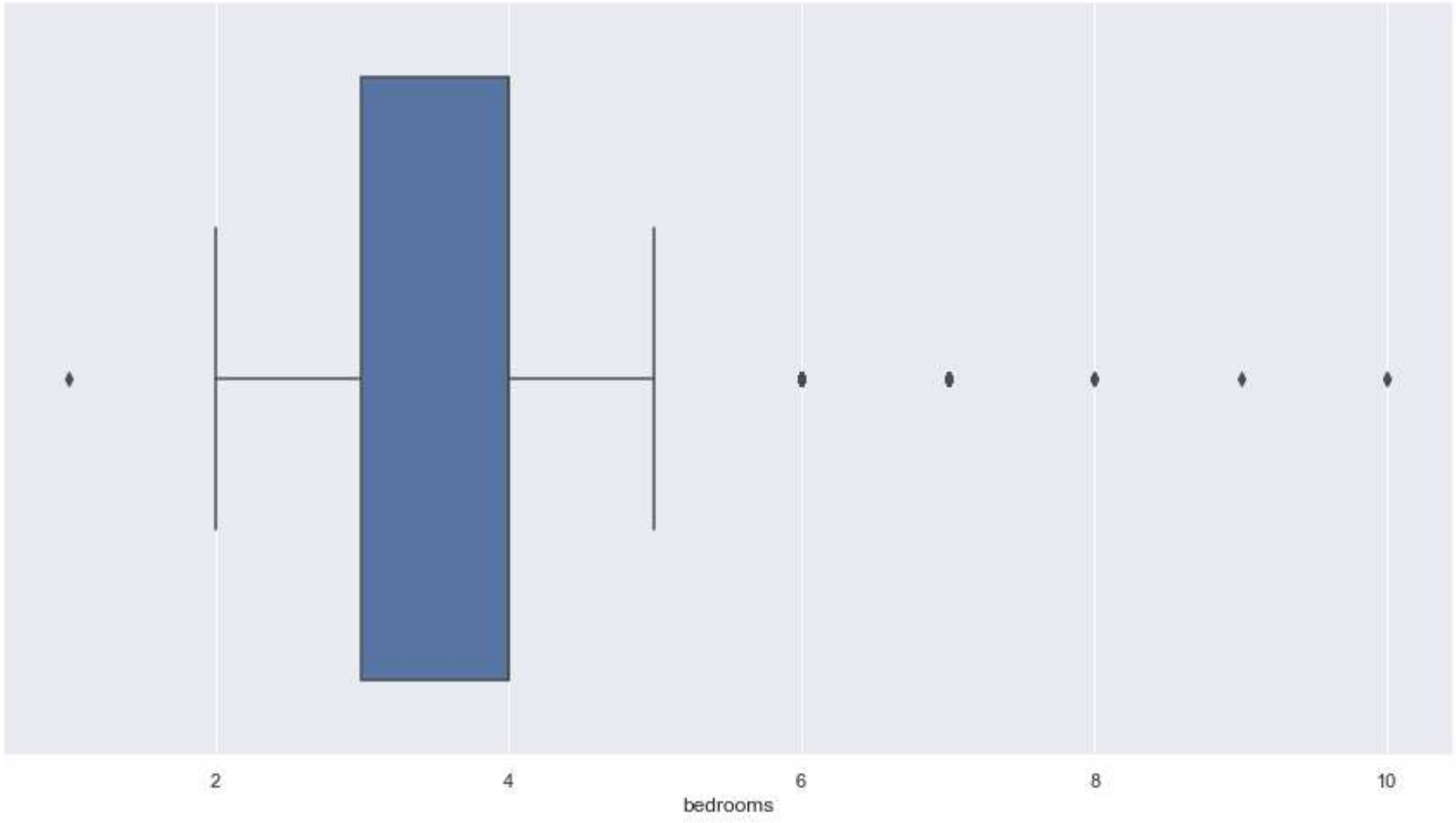
In [57]:
```
df.describe()
```

Out[57]:

	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	sq
count	4.754000e+03	4754.000000	4754.000000	4754.000000	4754.000000	4754.000000	4754.000000	4754.000000	4754.000000	4754.00
mean	7.688585e+05	85724.643879	-110.913500	32.318397	2.370038	6807.774794	1995.625999	3.892512	3.788704	3678.10
std	2.886162e+05	36.214401	0.117124	0.164705	8.426050	3393.148877	15.896500	0.850004	1.084433	938.24
min	1.690000e+05	85118.000000	-112.520168	31.361562	0.000000	0.000000	1900.000000	1.000000	1.000000	1484.00
25%	5.850000e+05	85718.000000	-110.980423	32.284408	0.590000	4849.935000	1989.000000	3.000000	3.000000	3049.00
50%	6.779500e+05	85737.000000	-110.923244	32.320212	0.980000	6261.000000	2000.000000	4.000000	4.000000	3511.50
75%	8.350000e+05	85750.000000	-110.859118	32.398978	1.657500	8084.490000	2006.000000	4.000000	4.000000	4113.75
max	3.700000e+06	86323.000000	-109.454637	34.927884	172.760000	31275.000000	2019.000000	10.000000	36.000000	7983.00

In [58]:
```
sns.boxplot(x=df['bedrooms'])
```

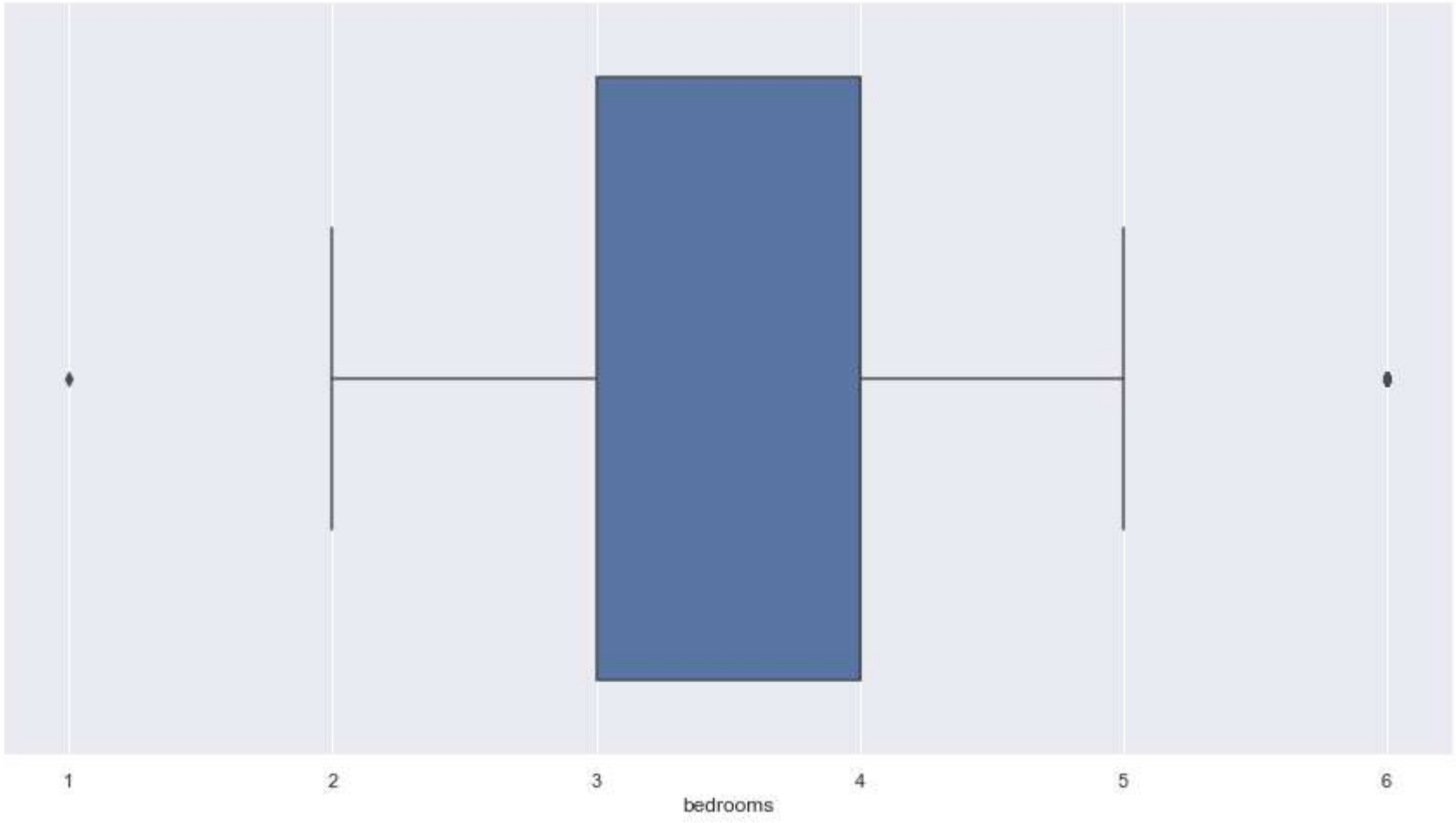
Out[58]:
```
<AxesSubplot:xlabel='bedrooms'>
```



In [59]:
```
df = df[~(df['bedrooms']>6)]
```

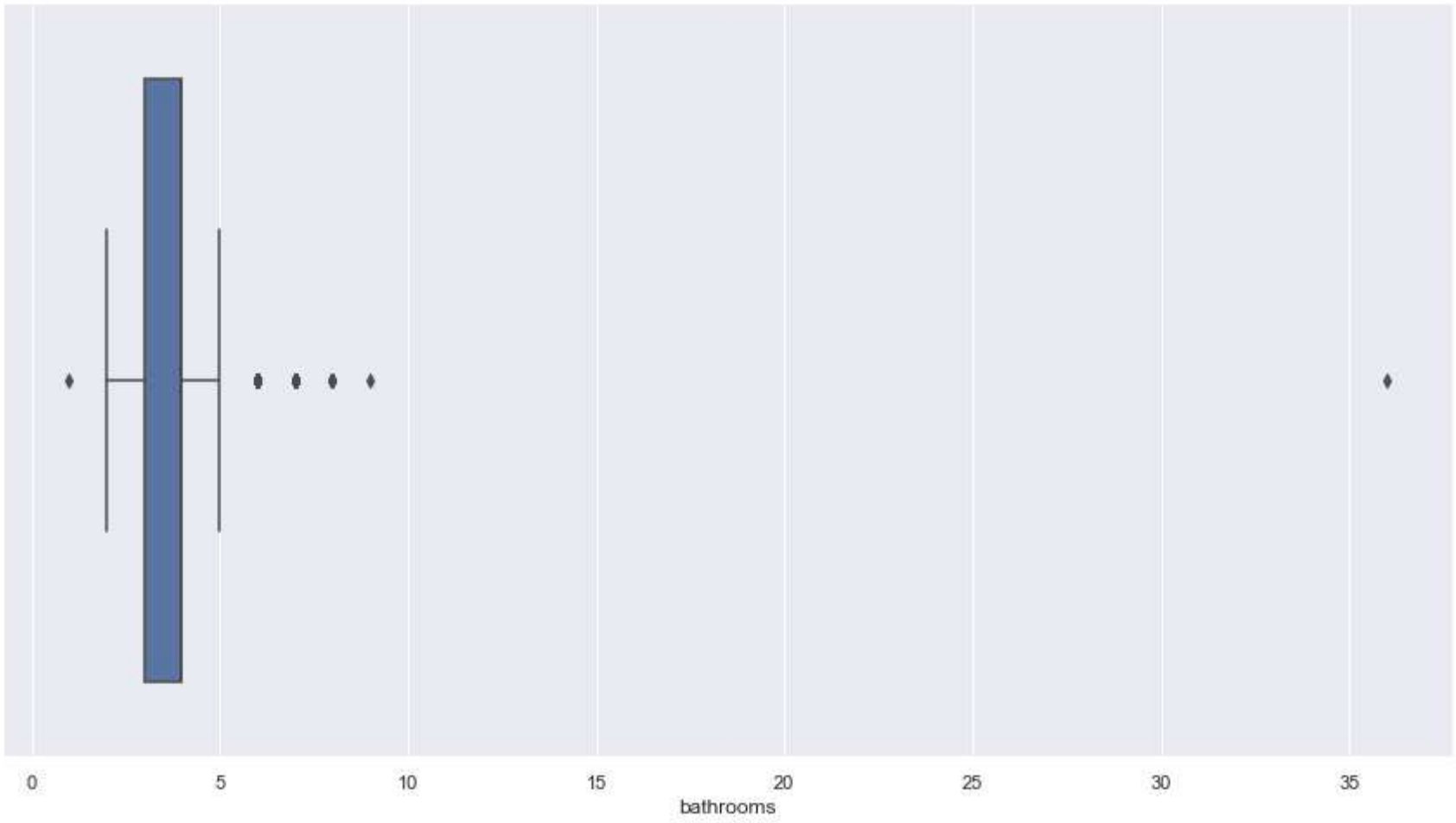
In [60]:
```
sns.boxplot(x=df['bedrooms'])
```

Out[60]:
```
<AxesSubplot:xlabel='bedrooms'>
```



In [61]:
```
sns.boxplot(x=df['bathrooms'])
```

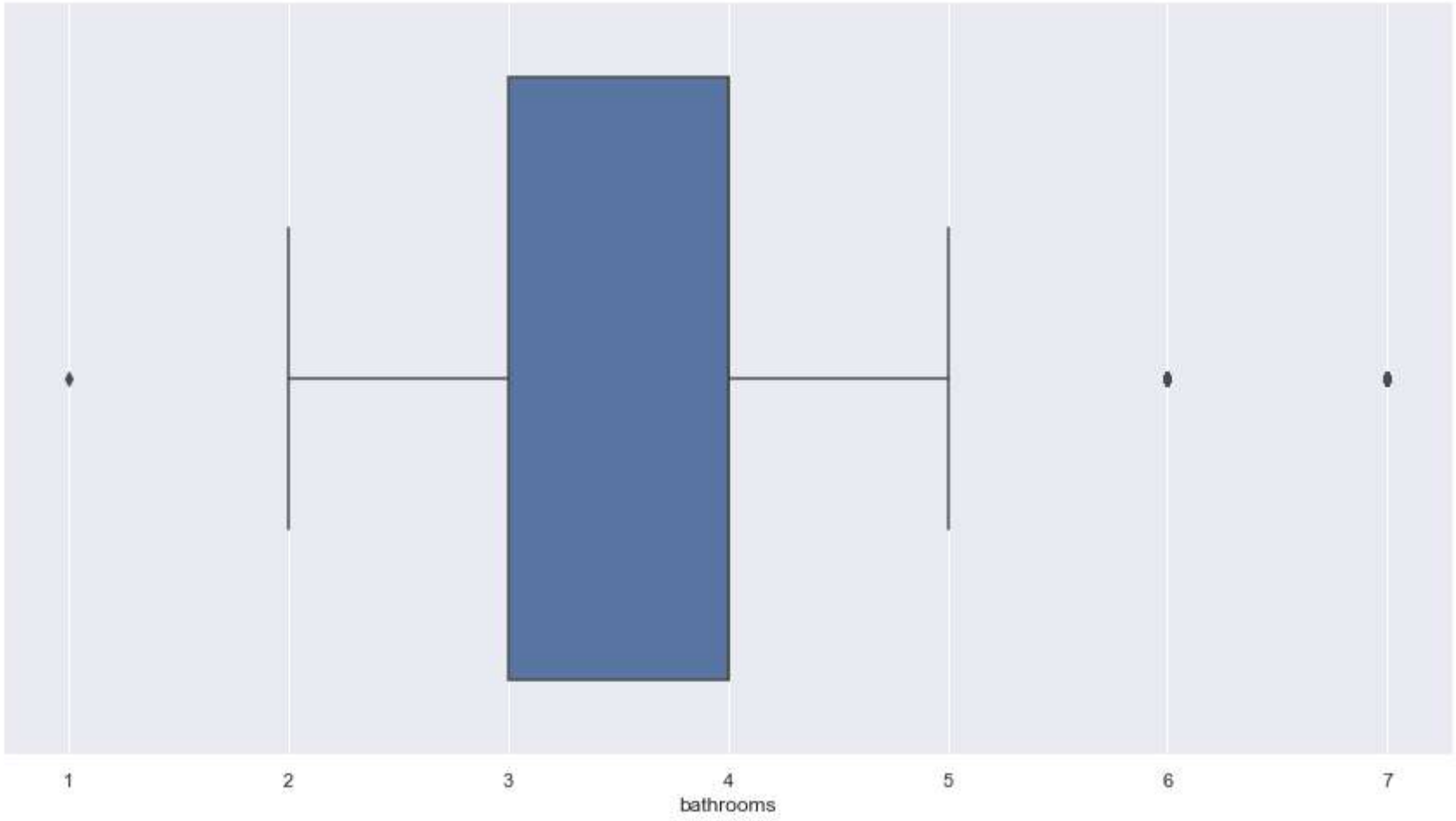
Out[61]:
```
<AxesSubplot:xlabel='bathrooms'>
```



In [62]:
```
df = df[~(df['bathrooms']>7)]
```

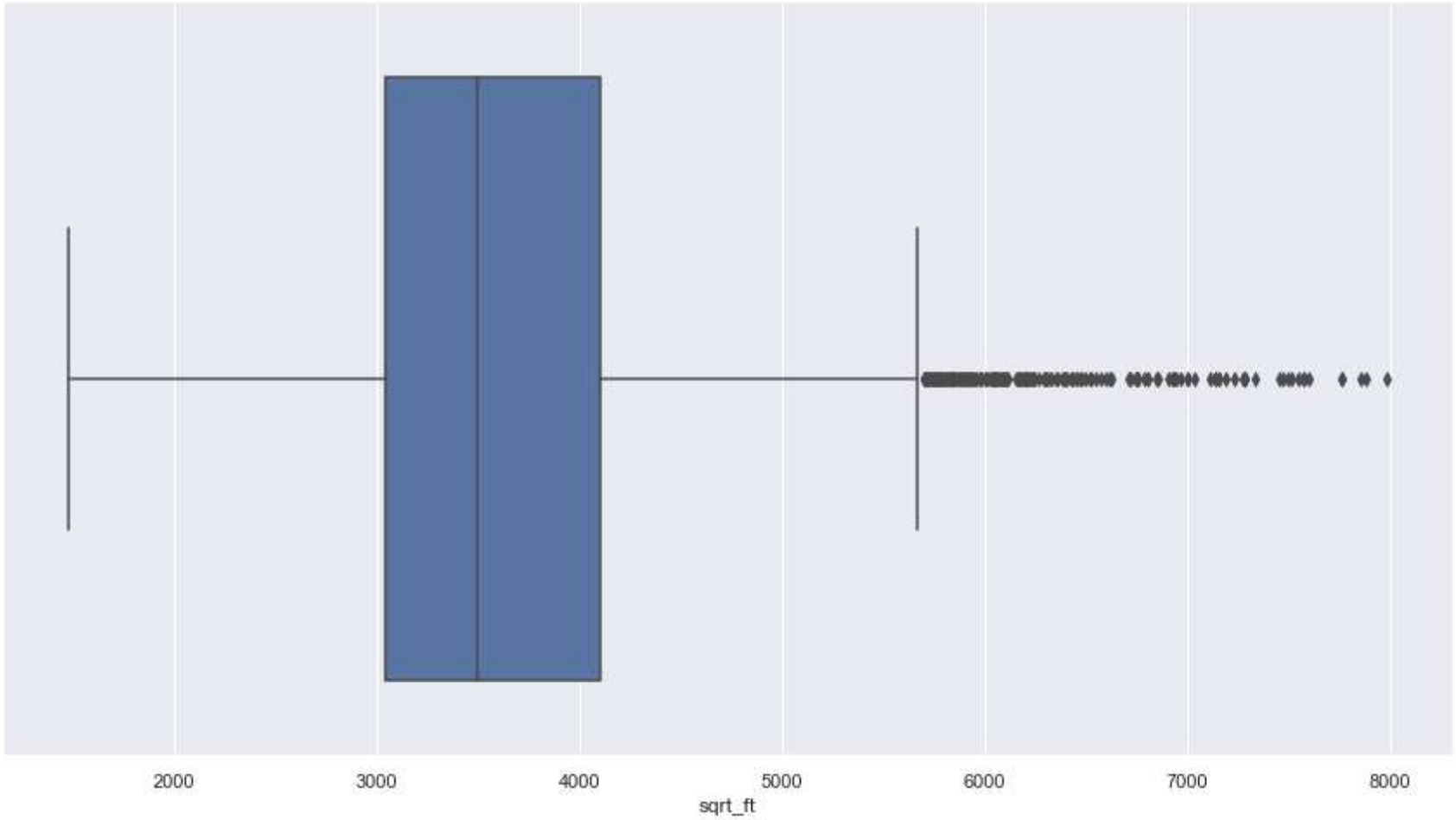
In [63]:
```
sns.boxplot(x=df['bathrooms'])
```

Out[63]:
```
<AxesSubplot:xlabel='bathrooms'>
```



In [64]:
```
sns.boxplot(x=df['sqrt_ft'])
```

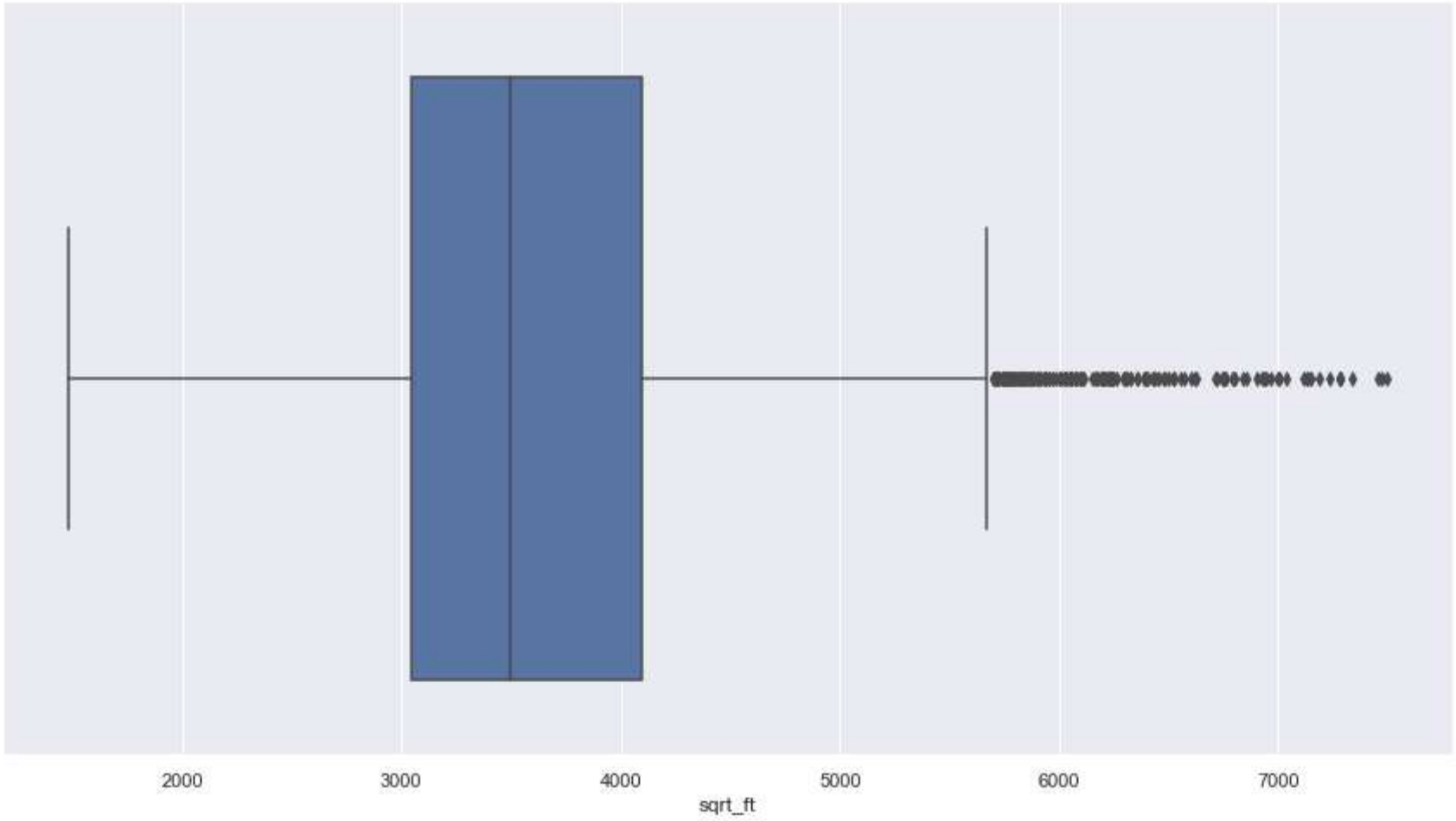
Out[64]: <AxesSubplot:xlabel='sqrt_ft'>



In [65]:
```
df = df[~(df['sqrt_ft']>7500)]
```

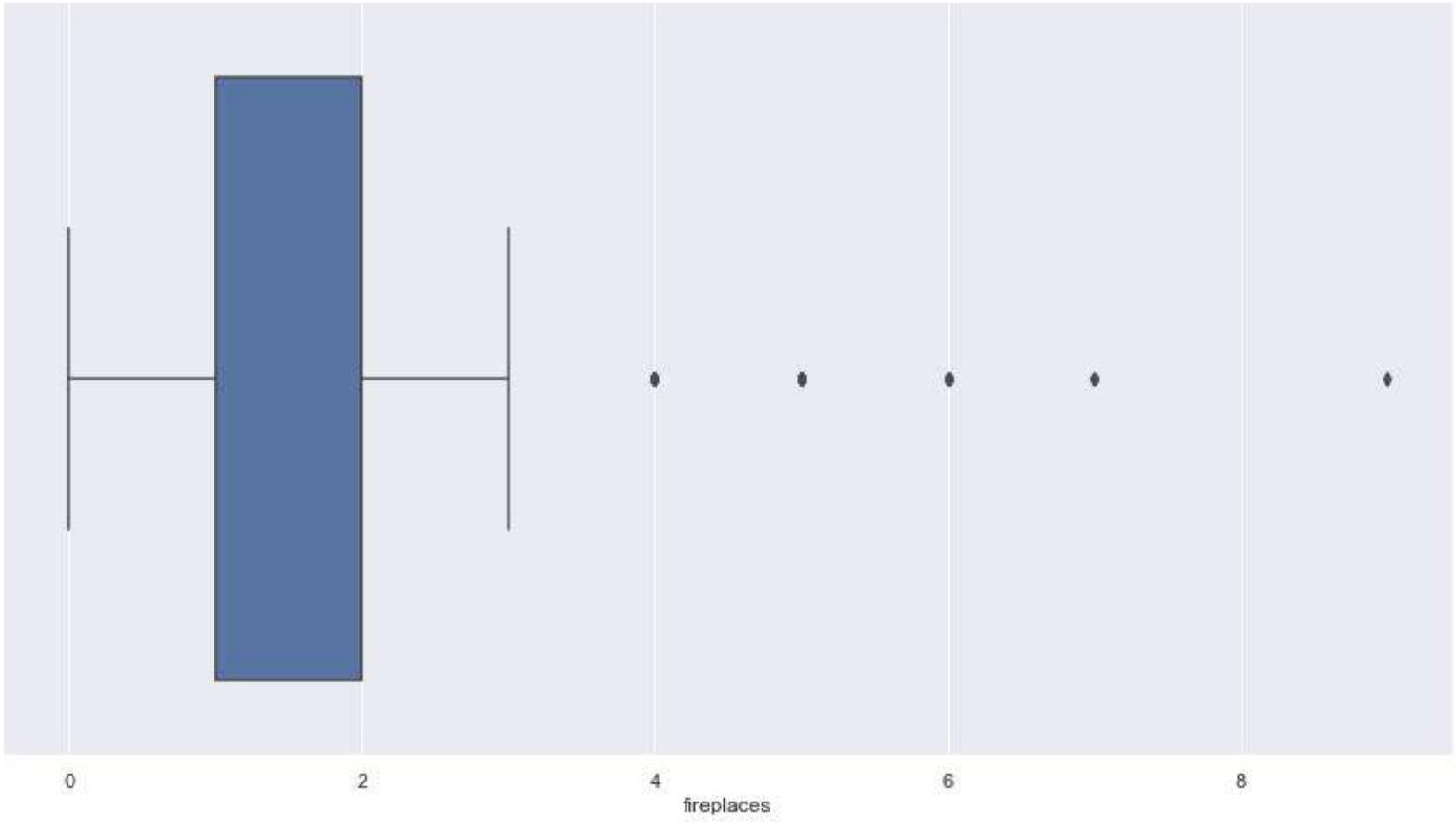
In [66]:
```
sns.boxplot(x=df['sqrt_ft'])
```

Out[66]:
```
<AxesSubplot:xlabel='sqrt_ft'>
```



In [67]:
```
sns.boxplot(x=df['fireplaces'])
```

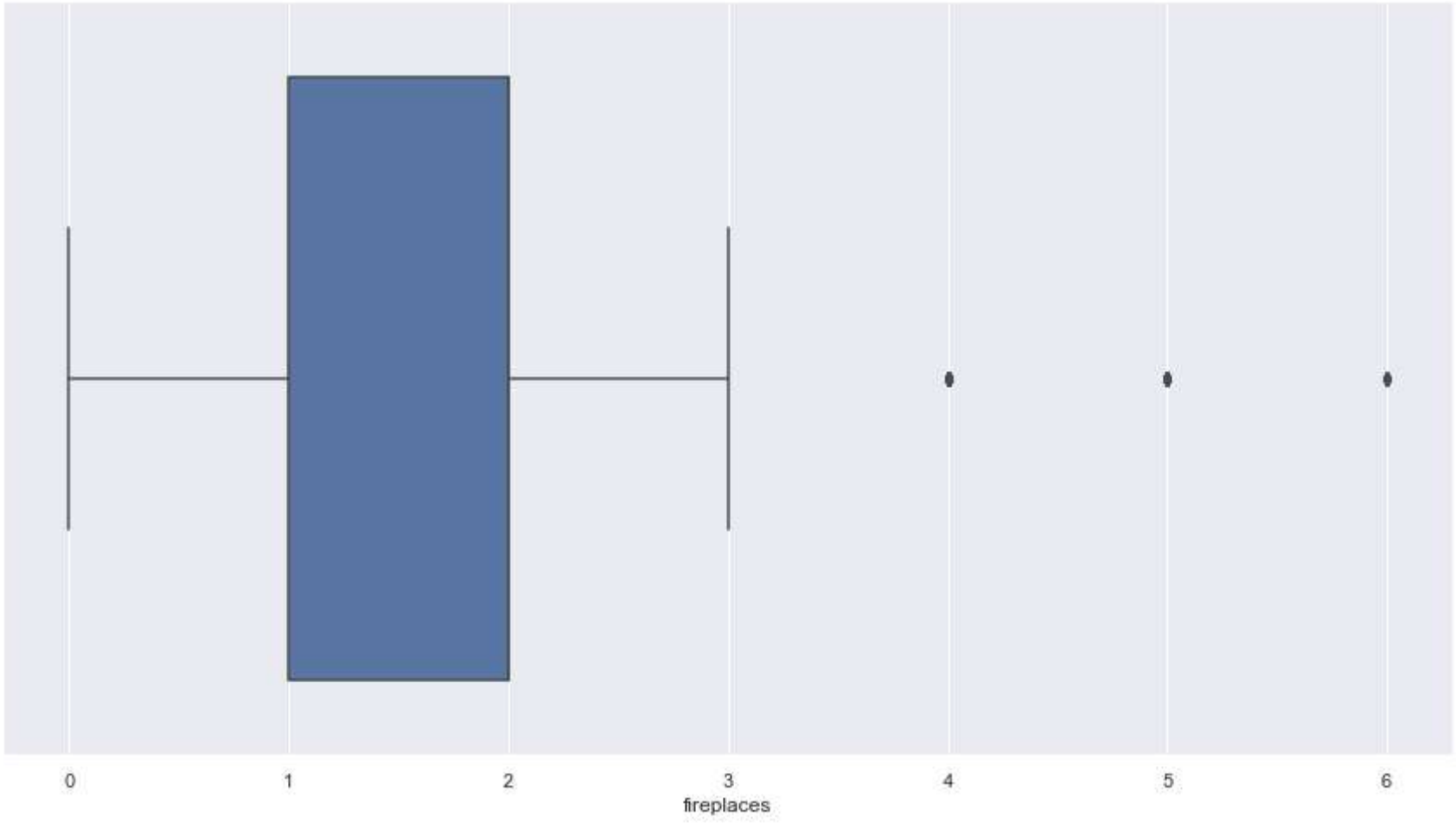
Out[67]: <AxesSubplot:xlabel='fireplaces'>



In [68]:
```
df = df[~(df['fireplaces']>6)]
```

In [69]:
```
sns.boxplot(x=df['fireplaces'])
```

Out[69]:
```
<AxesSubplot:xlabel='fireplaces'>
```



In [70]:
```
df.describe()
```

Out[70]:

	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	s
count	4.701000e+03	4701.000000	4701.000000	4701.000000	4701.000000	4701.000000	4701.000000	4701.000000	4701.000000	4701.00
mean	7.642429e+05	85724.722187	-110.913668	32.318953	2.326777	6751.911766	1995.839396	3.866411	3.752712	3650.55
std	2.816901e+05	36.286898	0.117339	0.164955	8.323000	3317.503530	15.544061	0.802139	0.930829	893.56
min	1.690000e+05	85118.000000	-112.520168	31.361562	0.000000	0.000000	1902.000000	1.000000	1.000000	1484.00
25%	5.850000e+05	85718.000000	-110.980646	32.284888	0.580000	4843.000000	1989.000000	3.000000	3.000000	3045.00
50%	6.750000e+05	85737.000000	-110.923314	32.320341	0.970000	6229.780000	2000.000000	4.000000	4.000000	3500.00
75%	8.300000e+05	85750.000000	-110.859163	32.399661	1.620000	8038.140000	2006.000000	4.000000	4.000000	4100.00
max	3.700000e+06	86323.000000	-109.454637	34.927884	172.760000	31275.000000	2019.000000	6.000000	7.000000	7495.00

In [71]:
```
df['taxes'].value_counts()
```

Out[71]:
```
0.00       20
900.22      9
1122.94     9
845.18      7
1288.81     7
           ..
8184.60     1
5096.00     1
5113.00     1
4662.64     1
5822.93     1
Name: taxes, Length: 4443, dtype: int64
```

In [72]:
```
df = df[~(df['taxes']==0)]
```

In [73]:
```
df.describe()
```

Out[73]:

	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	sq
count	4.681000e+03	4681.000000	4681.000000	4681.000000	4681.000000	4681.000000	4681.000000	4681.000000	4681.000000	4681.00
mean	7.637877e+05	85724.749840	-110.913487	32.318918	2.334125	6780.759925	1995.780816	3.869259	3.754860	3653.16
std	2.816862e+05	36.311785	0.117514	0.165161	8.339820	3295.027904	15.493527	0.801439	0.931804	894.27
min	1.690000e+05	85118.000000	-112.520168	31.361562	0.000000	1.000000	1902.000000	1.000000	1.000000	1484.00
25%	5.850000e+05	85718.000000	-110.980835	32.285254	0.590000	4855.530000	1989.000000	3.000000	3.000000	3047.00
50%	6.750000e+05	85737.000000	-110.922926	32.320392	0.980000	6247.000000	2000.000000	4.000000	4.000000	3503.00
75%	8.270000e+05	85750.000000	-110.859043	32.399467	1.620000	8048.000000	2006.000000	4.000000	4.000000	4101.00
max	3.700000e+06	86323.000000	-109.454637	34.927884	172.760000	31275.000000	2019.000000	6.000000	7.000000	7495.00

In [74]:
```
df.shape
```

Out[74]:
```
(4681, 13)
```

In [75]:
```
X1=df[['bedrooms','bathrooms','sqrt_ft','fireplaces']]
```

In [76]:
```
X1
```

Out[76]:

	bedrooms	bathrooms	sqrt_ft	fireplaces
1	4	6.0	6396.0	5.0
2	3	4.0	6842.0	5.0
5	5	7.0	5238.0	1.0
6	5	6.0	6480.0	2.0
7	5	5.0	5067.0	5.0
...	...	...	...	...
4777	4	3.0	2813.0	2.0
4778	3	2.0	2106.0	1.0
4779	5	3.0	3601.0	1.0
4780	4	3.0	2318.0	1.0
4781	4	4.0	3724.0	1.0

4681 rows × 4 columns

In [77]:
```
X1=(X1- X1.min()) / ( X1.max() - X1.min()) ## Normalize
```

In [78]:
```
X1
```

Out[78]:

	bedrooms	bathrooms	sqrt_ft	fireplaces
1	0.6	0.833333	0.817169	0.833333
2	0.4	0.500000	0.891366	0.833333
5	0.8	1.000000	0.624522	0.166667
6	0.8	0.833333	0.831143	0.333333
7	0.8	0.666667	0.596074	0.833333
...	...	...	...	...
4777	0.6	0.333333	0.221095	0.333333
4778	0.4	0.166667	0.103477	0.166667
4779	0.8	0.333333	0.352188	0.166667
4780	0.6	0.333333	0.138746	0.166667
4781	0.6	0.500000	0.372650	0.166667

4681 rows × 4 columns

In [79]:
```
X1=X1.to_numpy()
```

In [80]:
```
X1
```

Out[80]:
```
array([[0.6       , 0.83333333, 0.81716852, 0.83333333],
       [0.4       , 0.5       , 0.89136583, 0.83333333],
       [0.8       , 1.        , 0.62452171, 0.16666667],
       ...,
       [0.8       , 0.33333333, 0.35218766, 0.16666667],
       [0.6       , 0.33333333, 0.13874563, 0.16666667],
       [0.6       , 0.5       , 0.37265014, 0.16666667]])
```

In [81]:
```
y1=df['taxes']
```

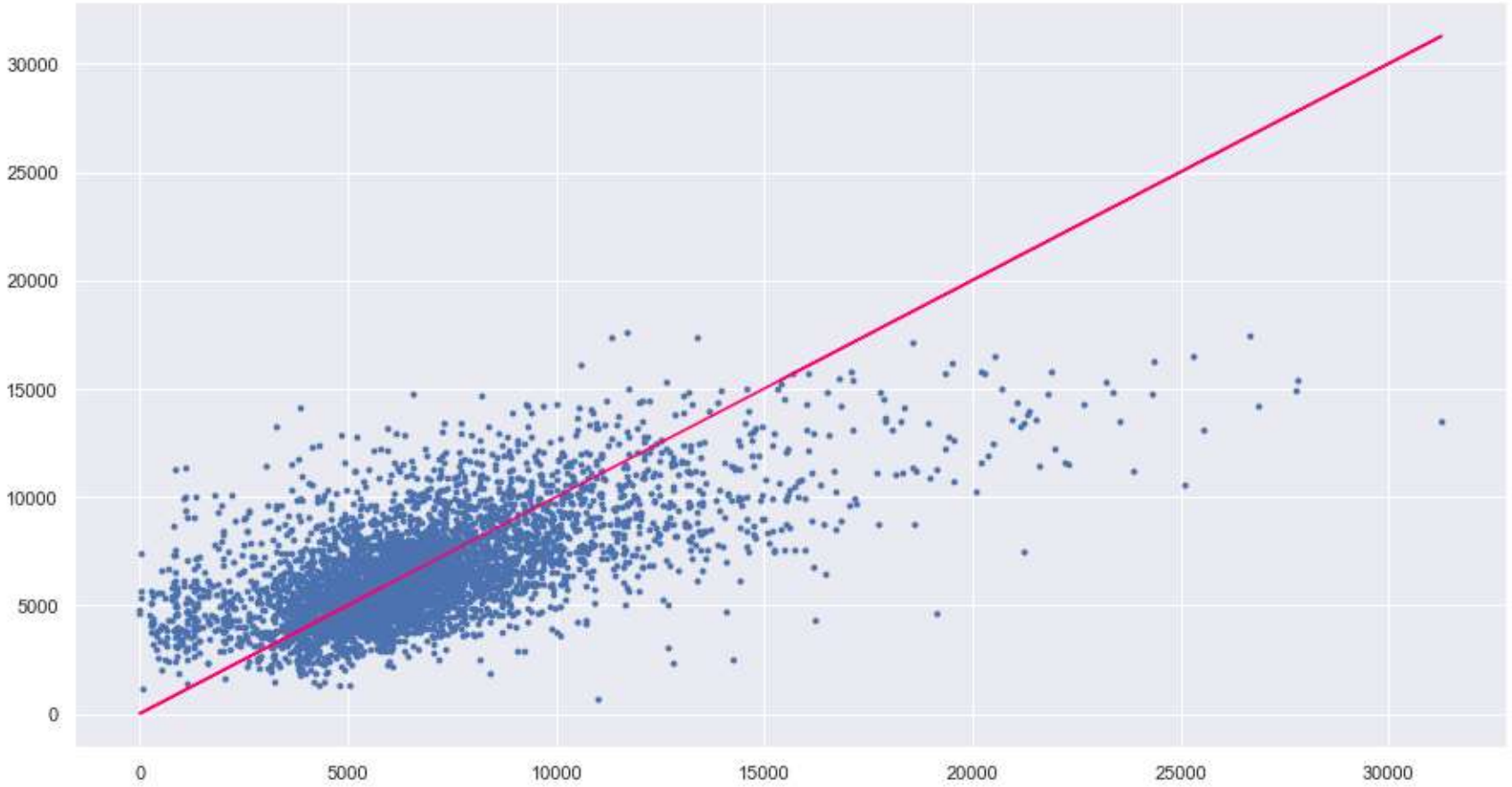
In [82]:
```
y1=y1.to_numpy()
```

In [83]:
```
lr_mul.fit(X1,y1)
```

In [84]:
```
y_hat1=lr_mul.predict(X1)
```

In [85]:
```
plt.figure()
plt.scatter(y1,y_hat1,s=8)
plt.plot(y1,y1,color="#FF0070")
```

Out[85]:
```
[<matplotlib.lines.Line2D at 0x1fddcf13c10>]
```

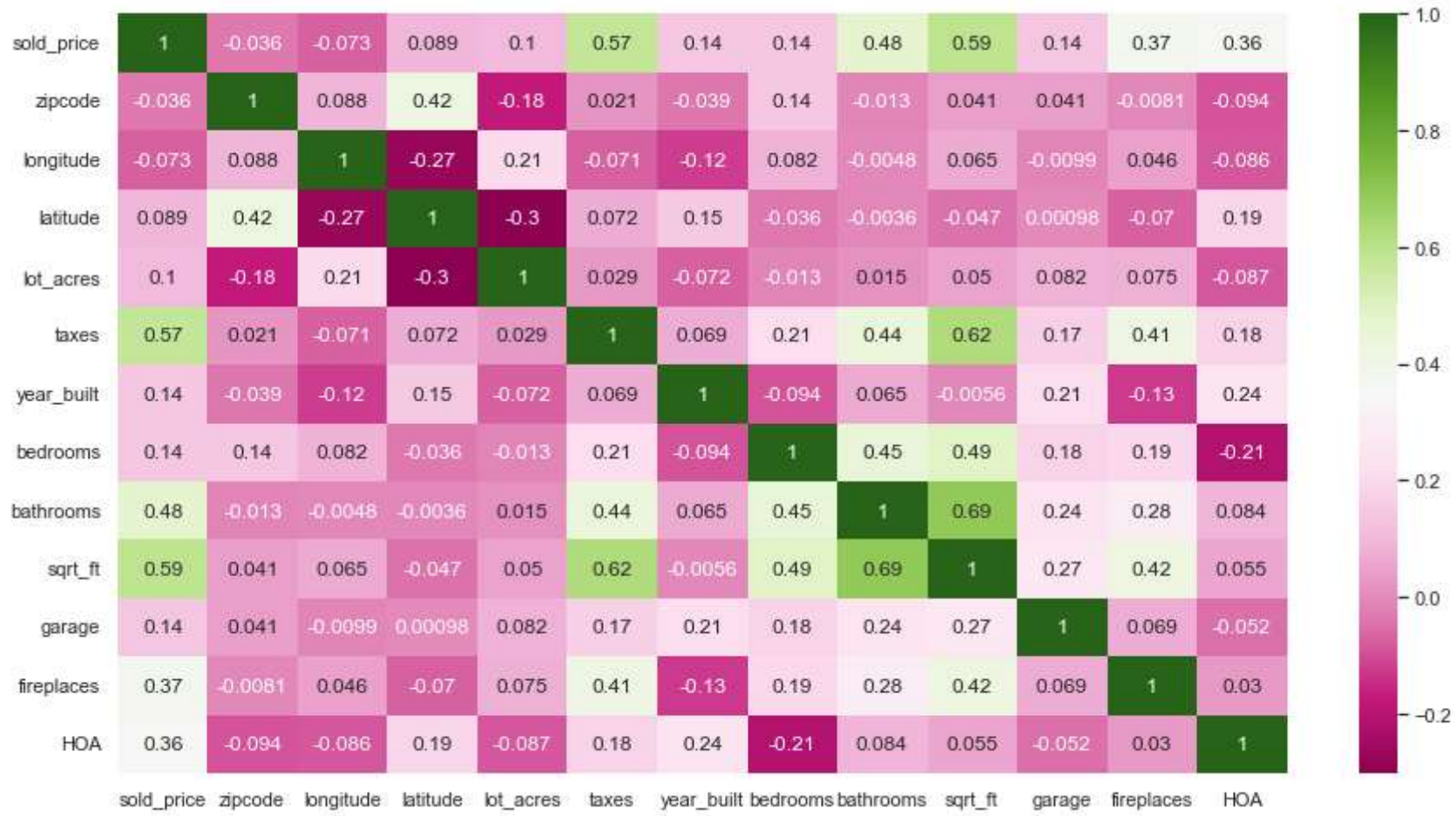


In [86]:
```
R2(y1,y_hat1)
```

Out[86]:
```
0.4028854630136791
```

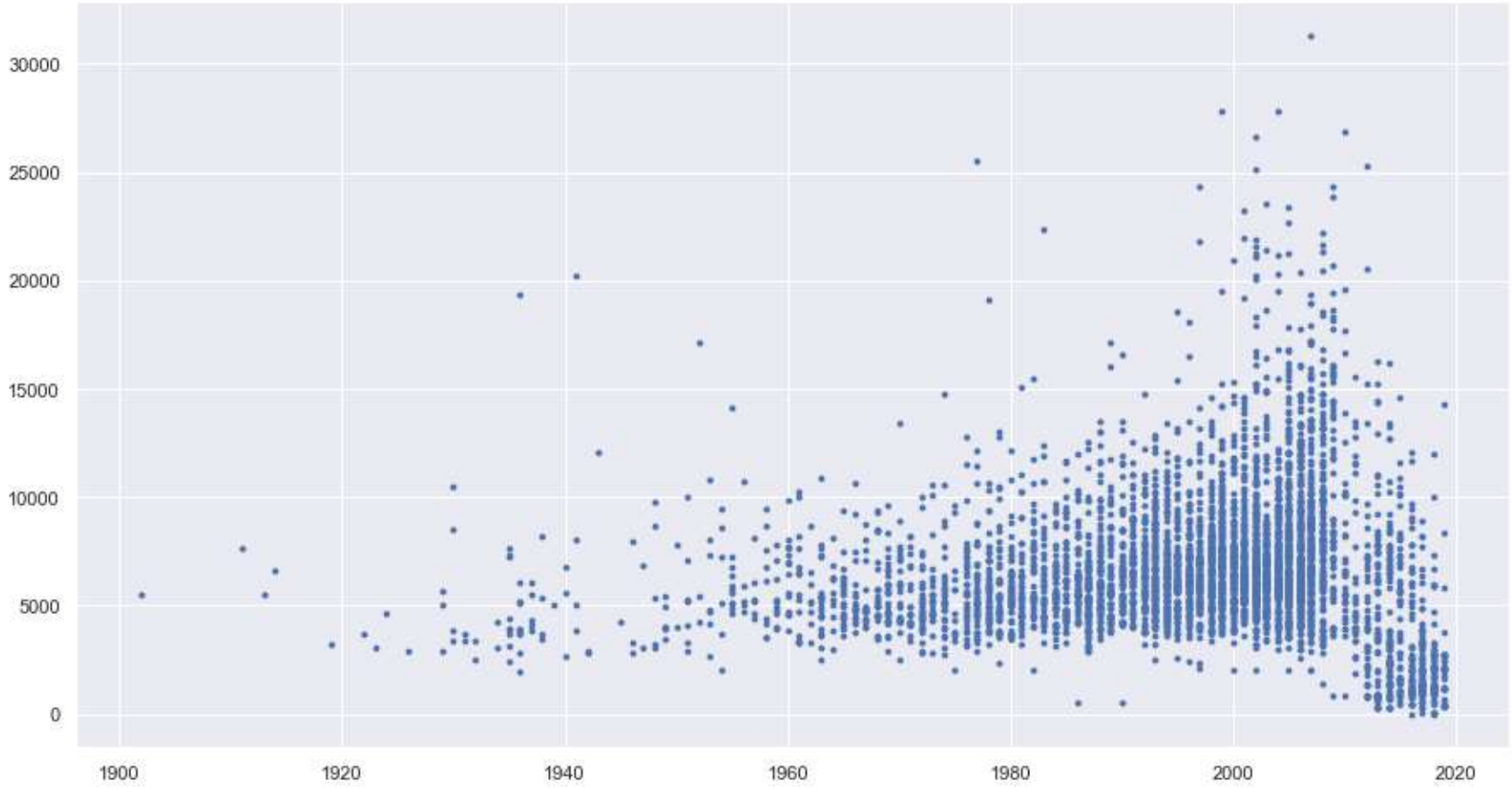
In [87]:
```
sns.set(rc={"figure.figsize":(15, 8)})
dfplot1 = sns.heatmap(df.corr(), cmap="PiYG", annot=True)
dfplot1
```

Out[87]:
```
<AxesSubplot:>
```



In [88]:
```
plt.figure()
plt.scatter(df['year_built'],df['taxes'],s=8)
```

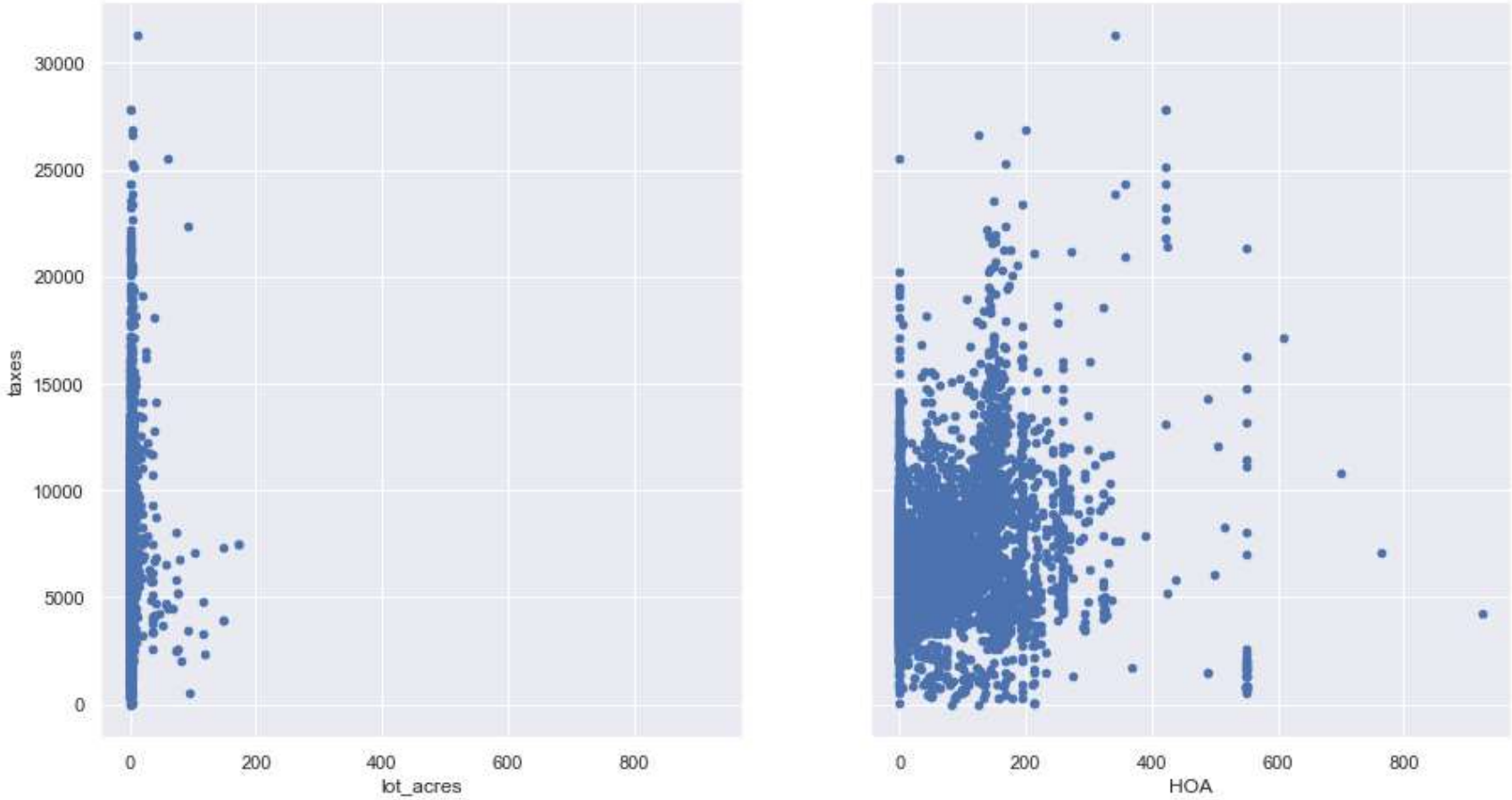
Out[88]:
```
<matplotlib.collections.PathCollection at 0x1fddd75aaf0>
```



In [89]:
```
figure,axs=plt.subplots(1,2,sharey=True,sharex=True,)
df.plot(kind='scatter',x='lot_acres',y='taxes',ax=axs[0])
df.plot(kind='scatter',x='HOA',y='taxes',ax=axs[1])
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will h
ave precedence in case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide
a 2D array with a single row if you intend to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will h
ave precedence in case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide
a 2D array with a single row if you intend to specify the same RGB or RGBA value for all points.
```

Out[89]:
```
<AxesSubplot:xlabel='HOA', ylabel='taxes'>
```



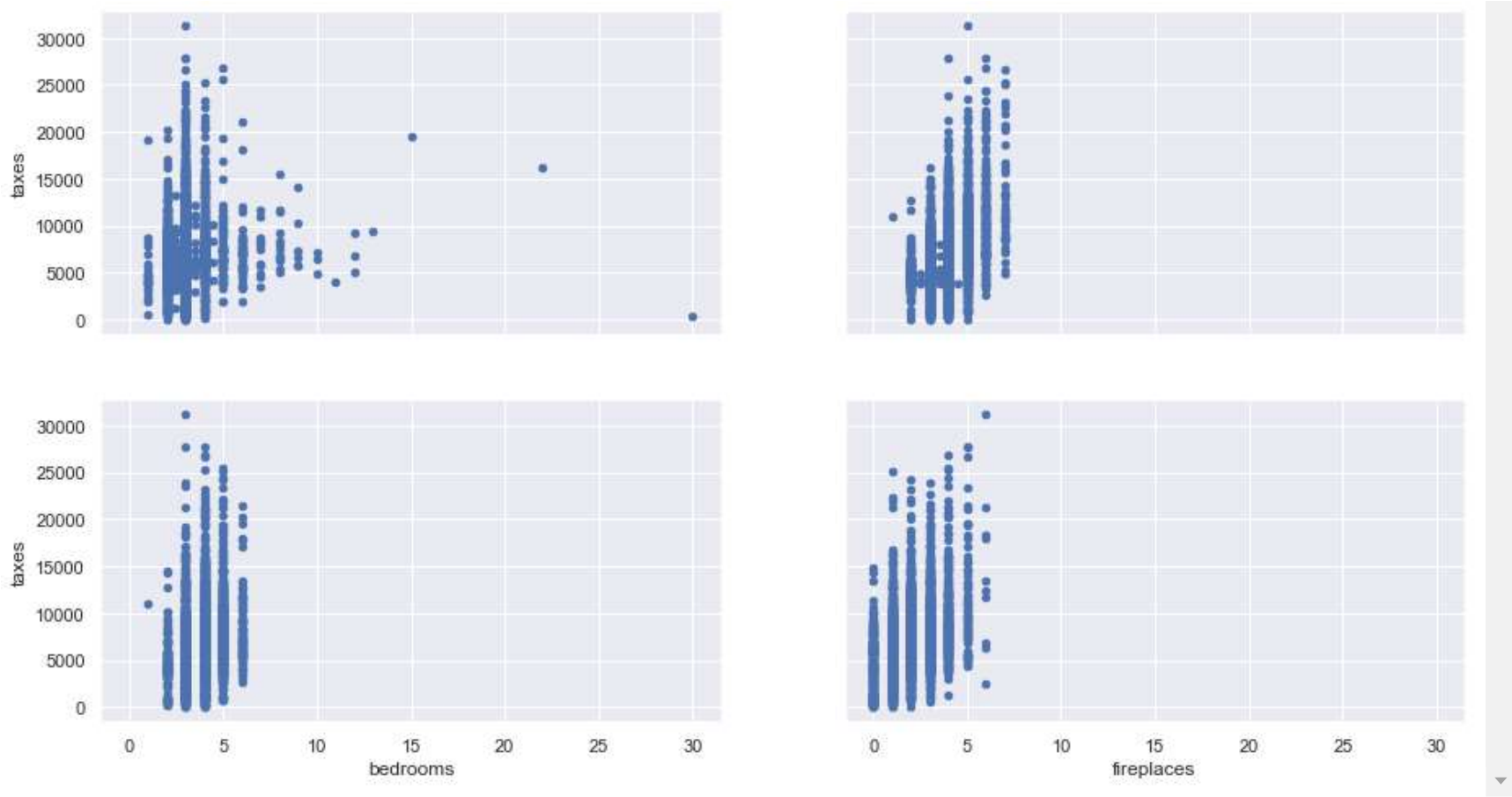
In [90]:

```python
figure,axs=plt.subplots(2,2,sharey=True,sharex=True,)
df.plot(kind='scatter',x='garage',y='taxes',ax=axs[0][0])
df.plot(kind='scatter',x='bathrooms',y='taxes',ax=axs[0][1])
df.plot(kind='scatter',x='bedrooms',y='taxes',ax=axs[1][0])
df.plot(kind='scatter',x='fireplaces',y='taxes',ax=axs[1][1])
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will h
ave precedence in case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide
a 2D array with a single row if you intend to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will h
ave precedence in case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide
a 2D array with a single row if you intend to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will h
ave precedence in case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide
a 2D array with a single row if you intend to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will h
ave precedence in case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide
a 2D array with a single row if you intend to specify the same RGB or RGBA value for all points.
```

Out[90]:

```
<AxesSubplot:xlabel='fireplaces', ylabel='taxes'>
```


taxes
30000
25000
20000
15000
10000
5000
0
taxes
30000
25000
20000
15000
10000
5000
0
0
5
10
15
20
25
30
bedrooms
0
5
10
15
20
25
30
fireplaces

In [91]:
```
df.describe()
```

Out[91]:

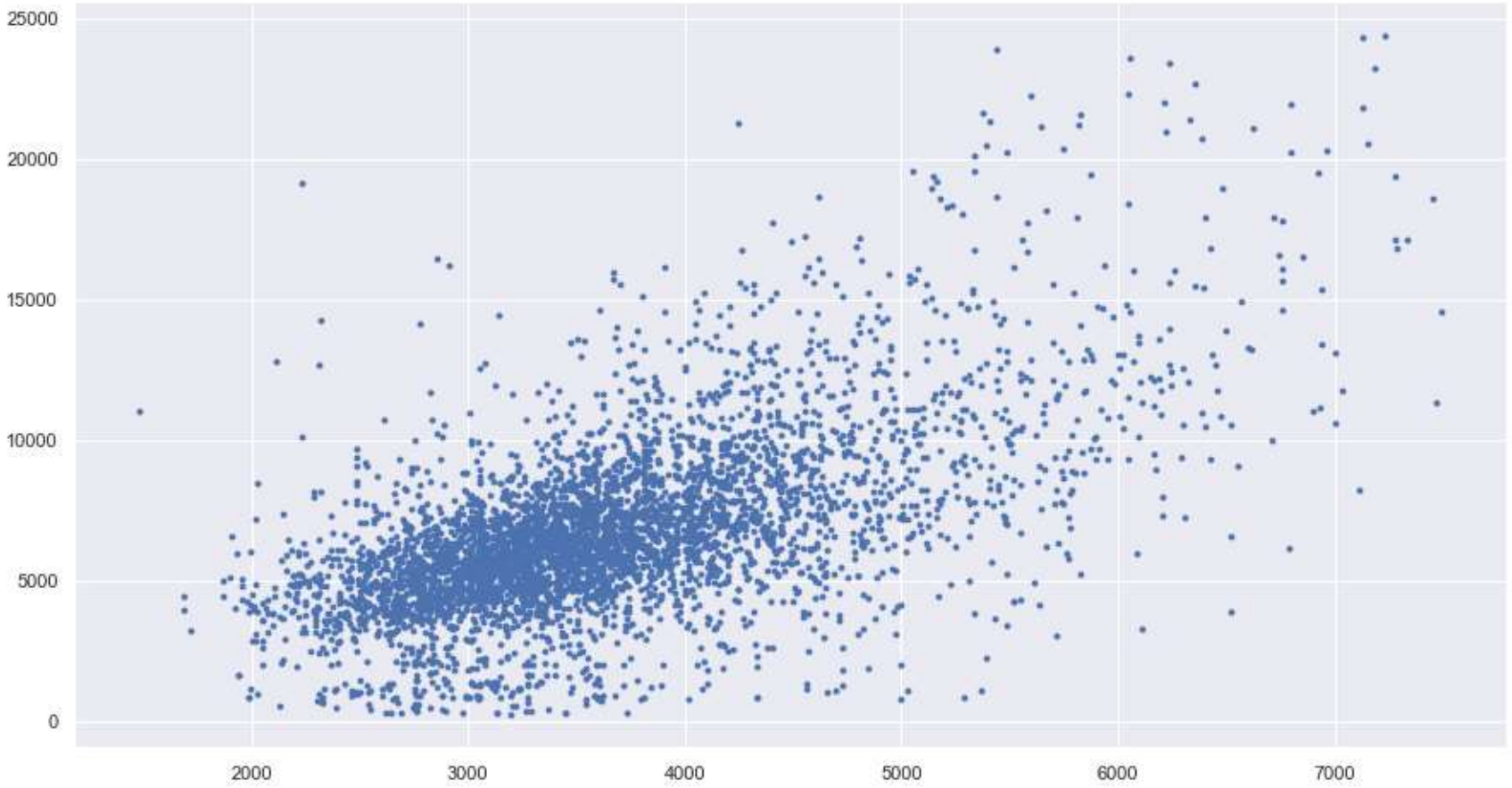
	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	s
count	4.681000e+03	4681.000000	4681.000000	4681.000000	4681.000000	4681.000000	4681.000000	4681.000000	4681.000000	4681.00
mean	7.637877e+05	85724.749840	-110.913487	32.318918	2.334125	6780.759925	1995.780816	3.869259	3.754860	3653.16
std	2.816862e+05	36.311785	0.117514	0.165161	8.339820	3295.027904	15.493527	0.801439	0.931804	894.27
min	1.690000e+05	85118.000000	-112.520168	31.361562	0.000000	1.000000	1902.000000	1.000000	1.000000	1484.00
25%	5.850000e+05	85718.000000	-110.980835	32.285254	0.590000	4855.530000	1989.000000	3.000000	3.000000	3047.00
50%	6.750000e+05	85737.000000	-110.922926	32.320392	0.980000	6247.000000	2000.000000	4.000000	4.000000	3503.00
75%	8.270000e+05	85750.000000	-110.859043	32.399467	1.620000	8048.000000	2006.000000	4.000000	4.000000	4101.00
max	3.700000e+06	86323.000000	-109.454637	34.927884	172.760000	31275.000000	2019.000000	6.000000	7.000000	7495.00

In [92]:
```
df = df[~(df['taxes']>25000)]
```

In [93]:
```
df = df[~(df['taxes']<200)]
```

In [94]:
```
plt.figure()
plt.scatter(df['sqrt_ft'],df['taxes'],s=8)
```

Out[94]:
```
<matplotlib.collections.PathCollection at 0x1fdddf0dfd0>
```



In [95]:
```
df['taxes'].value_counts()
```

Out[95]:
```
900.22     9
1122.94    9
845.18     7
1288.81    7
862.36     5
          ..
5285.00    1
5993.24    1
7641.49    1
8399.00    1
5822.93    1
Name: taxes, Length: 4429, dtype: int64
```

In [96]:
```
y2=df['taxes']
```

In [97]:
```
X2=df[['sqrt_ft','year_built','fireplaces','HOA']]
```

In [98]:
```
X2=(X2- X2.min()) / ( X2.max() - X2.min()) ## Normalize
```

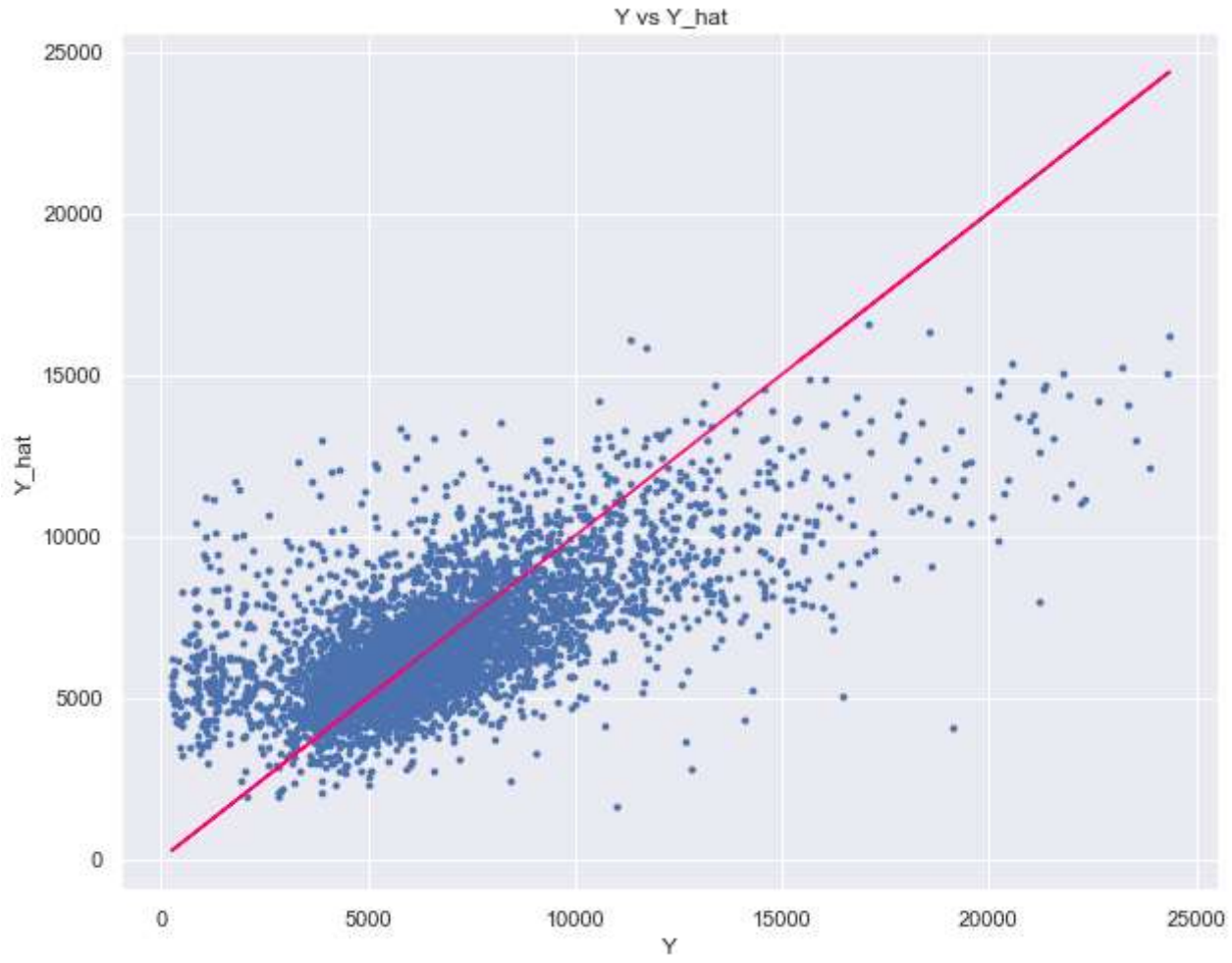
In [99]:
```
y2=y2.to_numpy()
X2=X2.to_numpy()
```

In [100]:
```
lr_mul.fit(X2,y2)
```

In [101]:
```
y_hat2=lr_mul.predict(X2)
```

In [102]:
```
plt.figure(figsize=(10,8))
plt.scatter(y2,y_hat2,s=8)
plt.plot(y2,y2,color="#FF0070")
plt.xlabel('Y')
plt.ylabel('Y_hat')
plt.title('Y vs Y_hat')
```

Out[102]: Text(0.5, 1.0, 'Y vs Y_hat')



In [103]:
```
R2(y2,y_hat2)
```

Out[103]:
```
0.4326730619154938
```

In [104]:
```
df.describe()
```

Out[104]:

	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	s
count	4.667000e+03	4667.000000	4667.000000	4667.000000	4667.000000	4667.000000	4667.000000	4667.000000	4667.000000	4667.00
mean	7.608633e+05	85724.772873	-110.913390	32.318923	2.319293	6754.724650	1995.742876	3.869509	3.751768	3649.21
std	2.722210e+05	36.309137	0.117595	0.164834	8.306809	3181.287317	15.489522	0.801778	0.927294	887.95
min	3.000000e+05	85118.000000	-112.520168	31.361562	0.000000	252.590000	1902.000000	1.000000	1.000000	1484.00
25%	5.850000e+05	85718.000000	-110.980857	32.285230	0.590000	4862.000000	1989.000000	3.000000	3.000000	3046.50
50%	6.750000e+05	85737.000000	-110.922752	32.320316	0.970000	6245.770000	2000.000000	4.000000	4.000000	3501.00
75%	8.250000e+05	85750.000000	-110.859005	32.399420	1.620000	8039.765000	2006.000000	4.000000	4.000000	4100.00
max	3.411450e+06	86323.000000	-109.454637	34.927884	172.760000	24353.000000	2019.000000	6.000000	7.000000	7495.00

split the data,apply regresion and KNN regressor

In [105]:
```
dforigin=df
```

In [106]:
```
## normalize the data
df=(df- df.min()) / ( df.max() - df.min())
```

In [107]:
```
df
```

Out[107]:

	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	sqrt_ft	garage	fireplaces	
1	1.000000	0.524481	0.556641	0.258978	0.018581	0.628222	0.794872	0.6	0.833333	0.817169	0.068966	0.833333	0.
6	0.948111	0.524481	0.548753	0.270871	0.020433	0.775237	0.897436	0.8	0.833333	0.831143	0.068966	0.333333	0.
7	0.795937	0.433195	0.481067	0.056237	0.851933	0.293678	0.282051	0.8	0.666667	0.596074	0.137931	0.833333	0.
8	0.562439	0.497925	0.522544	0.274806	0.009551	1.000000	0.914530	0.8	0.833333	0.955914	0.068966	0.666667	0.
9	0.578508	0.430705	0.581944	0.100884	0.171336	0.475943	0.794872	1.0	1.000000	0.826818	0.068966	0.666667	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	
4777	0.085169	0.524481	0.554538	0.265283	0.004804	0.179089	0.717949	0.6	0.333333	0.221095	0.034483	0.333333	0.
4778	0.075527	0.497925	0.521240	0.268045	0.001042	0.172670	0.854701	0.4	0.166667	0.103477	0.034483	0.166667	0.
4779	0.062672	0.434025	0.606205	0.153198	0.028826	0.073211	0.880342	0.8	0.333333	0.352188	0.068966	0.166667	0.
4780	0.080348	0.524481	0.542031	0.267730	0.008219	0.189599	0.752137	0.6	0.333333	0.138746	0.068966	0.166667	0.
4781	0.080348	0.520332	0.477777	0.262262	0.005846	0.231131	0.914530	0.6	0.500000	0.372650	0.068966	0.166667	0.

4667 rows × 13 columns

shuffle the data

In [108]:
```
np.random.seed(0)
##df = df.sample(frac=1)
## Split database
i = np.arange(len(df))
np.random.shuffle(i)
train = df.iloc[i[:int(len(df)*0.70)]]
validation= df.iloc[i[int(len(df)*0.70):int(len(df)*0.85)]]
test = df.iloc[i[int(len(df)*0.85):]]
```

In [109]:
```
df
```

Out[109]:

	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	sqrt_ft	garage	fireplaces	
1	1.000000	0.524481	0.556641	0.258978	0.018581	0.628222	0.794872	0.6	0.833333	0.817169	0.068966	0.833333	0.
6	0.948111	0.524481	0.548753	0.270871	0.020433	0.775237	0.897436	0.8	0.833333	0.831143	0.068966	0.333333	0.
7	0.795937	0.433195	0.481067	0.056237	0.851933	0.293678	0.282051	0.8	0.666667	0.596074	0.137931	0.833333	0.
8	0.562439	0.497925	0.522544	0.274806	0.009551	1.000000	0.914530	0.8	0.833333	0.955914	0.068966	0.666667	0.
9	0.578508	0.430705	0.581944	0.100884	0.171336	0.475943	0.794872	1.0	1.000000	0.826818	0.068966	0.666667	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	
4777	0.085169	0.524481	0.554538	0.265283	0.004804	0.179089	0.717949	0.6	0.333333	0.221095	0.034483	0.333333	0.
4778	0.075527	0.497925	0.521240	0.268045	0.001042	0.172670	0.854701	0.4	0.166667	0.103477	0.034483	0.166667	0.
4779	0.062672	0.434025	0.606205	0.153198	0.028826	0.073211	0.880342	0.8	0.333333	0.352188	0.068966	0.166667	0.
4780	0.080348	0.524481	0.542031	0.267730	0.008219	0.189599	0.752137	0.6	0.333333	0.138746	0.068966	0.166667	0.
4781	0.080348	0.520332	0.477777	0.262262	0.005846	0.231131	0.914530	0.6	0.500000	0.372650	0.068966	0.166667	0.

4667 rows × 13 columns

train data

In [110]:
```
y_train=train['taxes']
```

In [111]:
```
X_train=train[['sqrt_ft','year_built','fireplaces','HOA']]
```

In [112]:
```
y_train=y_train.to_numpy()
X_train=X_train.to_numpy()
```

In [113]:
```
lr_mul.fit(X_train,y_train)
```

In [114]:
```
y_hattrain=lr_mul.predict(X_train)
```

In [115]:
```
R2(y_train,y_hattrain)
```

Out[115]:
```
0.43206437750215887
```

validation data

In [116]:
```
y_val=validation['taxes']
```

In [117]:
```
X_val=validation[['sqrt_ft','year_built','fireplaces','HOA']]
```

In [118]:
```
y_val=y_val.to_numpy()
X_val=X_val.to_numpy()
```

In [119]:
```
y_hatval=lr_mul.predict(X_val)
```

In [120]:
```
R2(y_val,y_hatval)
```

Out[120]:
```
0.33925736010186636
```

test data

In [121]:
```
y_test=test['taxes']
```

In [122]:
```
X_test=test[['sqrt_ft','year_built','fireplaces','HOA']]
```

In [123]:
```
y_test=y_test.to_numpy()
X_test=X_test.to_numpy()
```

In [124]:
```
y_hattest=lr_mul.predict(X_test)
```

In [125]:
```
R2(y_test,y_hattest)
```

Out[125]:
```
0.5066082579936524
```

Implement the KNN to try getting a better accuracy

In [126]:
```
class KNNRegresor():

  def fit(self,X,y):
    self.X=X
    self.y=y

  def predict(self,X,K, epsilon = 1e-3):
    N=len(X)
    y_hat= np.zeros(N)

    for i in range(N):
      dist2 = np.sum((self.X-X[i])**2,axis=1)
      idxt= np.argsort(dist2)[:K]
      gamma_K = np.exp(-dist2[idxt])/np.exp(-dist2[idxt]).sum()
      y_hat[i]=gamma_K.dot(self.y[idxt])

    return y_hat
```

In [127]:
```
knn_reg=KNNRegresor()
```

In [128]:
```
knn_reg.fit(X_train,y_train)
```

In [129]:
```
y_hatknn = knn_reg.predict(X_train,4)
```

In [130]:
```
R2(y_train,y_hatknn)
```

Out[130]:
```
0.7090438997251874
```

validation

In [131]:
```
y_hatknnv = knn_reg.predict(X_val,15)
```

In [132]:
```
R2(y_val,y_hatknnv)
```

Out[132]:
```
0.513229382978138
```

test

In [133]:
```
y_hatknnt = knn_reg.predict(X_test,15)
```

In [134]:
```
R2(y_test,y_hatknnt)
```

Out[134]:
```
0.5989179033584271
```

longitude and latitude

In [135]:
```
import plotly.express as px
from sklearn.cluster import KMeans
```

In [136]:
```
dforigin
```

Out[136]:

	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	sqrt_ft	garage	fireplaces	HO
1	3411450.0	85750	-110.813768	32.285162	3.21	15393.00	1995	4	6.0	6396.0	3.0	5.0	55.
6	3250000.0	85750	-110.837950	32.327575	3.53	18936.11	2007	5	6.0	6480.0	3.0	2.0	141.
7	2776518.0	85640	-111.045441	31.562121	147.18	7330.36	1935	5	5.0	5067.0	5.0	5.0	0.
8	2050000.0	85718	-110.918294	32.341609	1.65	24353.00	2009	5	6.0	7230.0	3.0	4.0	357.
9	2100000.0	85637	-110.736202	31.721347	29.60	11723.00	1995	6	7.0	6454.0	3.0	4.0	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	
4777	565000.0	85750	-110.820216	32.307646	0.83	4568.71	1986	4	3.0	2813.0	2.0	2.0	6.
4778	535000.0	85718	-110.922291	32.317496	0.18	4414.00	2002	3	2.0	2106.0	2.0	1.0	198.
4779	495000.0	85641	-110.661829	31.907917	4.98	2017.00	2005	5	3.0	3601.0	3.0	1.0	0.
4780	550000.0	85750	-110.858556	32.316373	1.42	4822.01	1990	4	3.0	2318.0	3.0	1.0	43.
4781	550000.0	85745	-111.055528	32.296871	1.01	5822.93	2009	4	4.0	3724.0	3.0	1.0	0.

4667 rows × 13 columns

In [137]:
```
df
```

Out[137]:

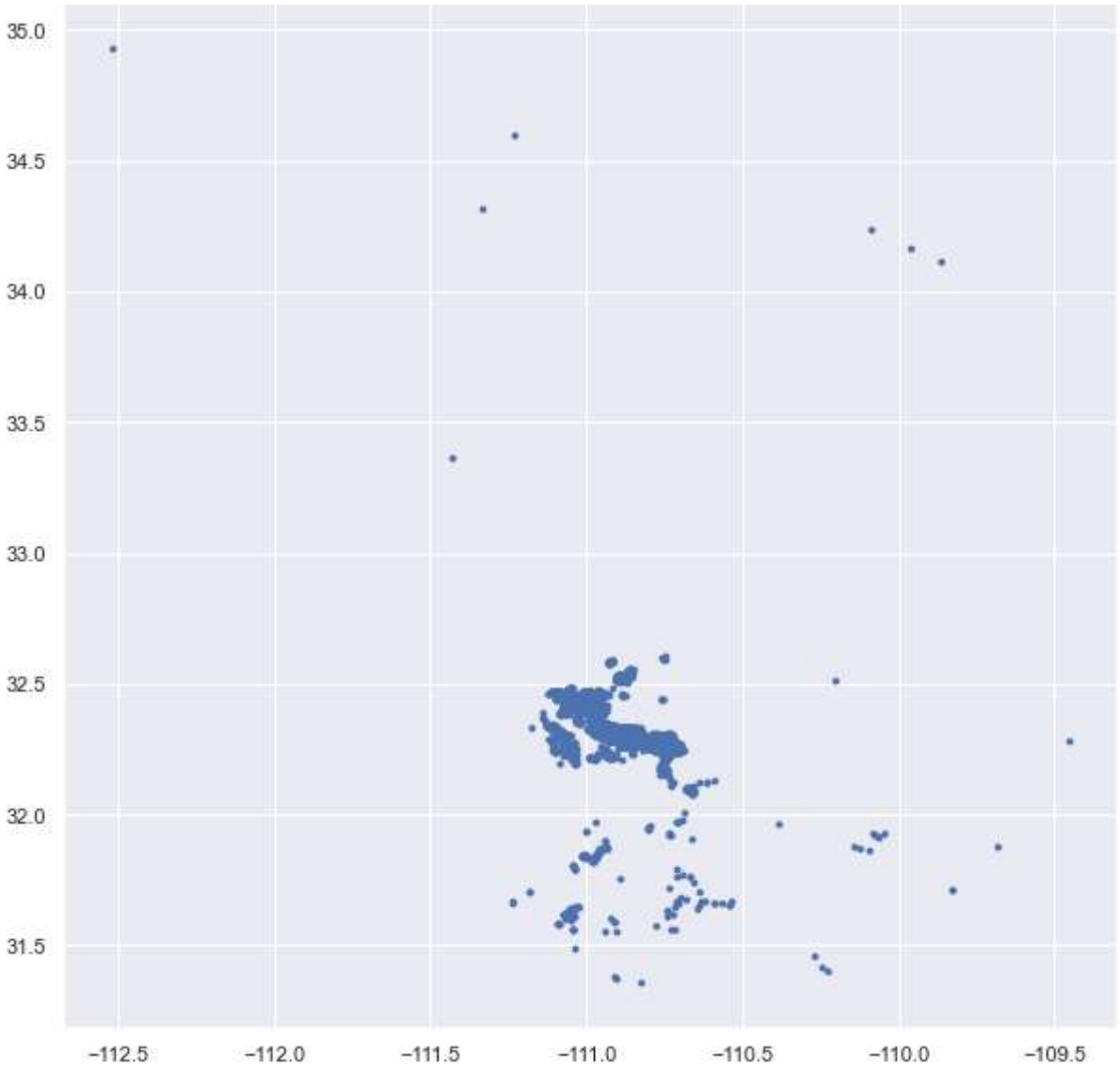
	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	sqrt_ft	garage	fireplaces	
1	1.000000	0.524481	0.556641	0.258978	0.018581	0.628222	0.794872	0.6	0.833333	0.817169	0.068966	0.833333	0.
6	0.948111	0.524481	0.548753	0.270871	0.020433	0.775237	0.897436	0.8	0.833333	0.831143	0.068966	0.333333	0.
7	0.795937	0.433195	0.481067	0.056237	0.851933	0.293678	0.282051	0.8	0.666667	0.596074	0.137931	0.833333	0.
8	0.562439	0.497925	0.522544	0.274806	0.009551	1.000000	0.914530	0.8	0.833333	0.955914	0.068966	0.666667	0.
9	0.578508	0.430705	0.581944	0.100884	0.171336	0.475943	0.794872	1.0	1.000000	0.826818	0.068966	0.666667	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	
4777	0.085169	0.524481	0.554538	0.265283	0.004804	0.179089	0.717949	0.6	0.333333	0.221095	0.034483	0.333333	0.
4778	0.075527	0.497925	0.521240	0.268045	0.001042	0.172670	0.854701	0.4	0.166667	0.103477	0.034483	0.166667	0.
4779	0.062672	0.434025	0.606205	0.153198	0.028826	0.073211	0.880342	0.8	0.333333	0.352188	0.068966	0.166667	0.
4780	0.080348	0.524481	0.542031	0.267730	0.008219	0.189599	0.752137	0.6	0.333333	0.138746	0.068966	0.166667	0.
4781	0.080348	0.520332	0.477777	0.262262	0.005846	0.231131	0.914530	0.6	0.500000	0.372650	0.068966	0.166667	0.

4667 rows × 13 columns

In [138]:
```
fig = px.scatter_mapbox(dforigin,lat = 'latitude', lon = 'longitude', color = 'sold_price',
                        center=dict(lon=-111, lat=32.37),
                       zoom = 9, mapbox_style = 'open-street-map')
fig
```

In [139]:
```
plt.figure(figsize=(10,10))
plt.scatter(dforigin['longitude'],dforigin['latitude'],s=8)
```

Out[139]:
```
<matplotlib.collections.PathCollection at 0x1fde2be4730>
```

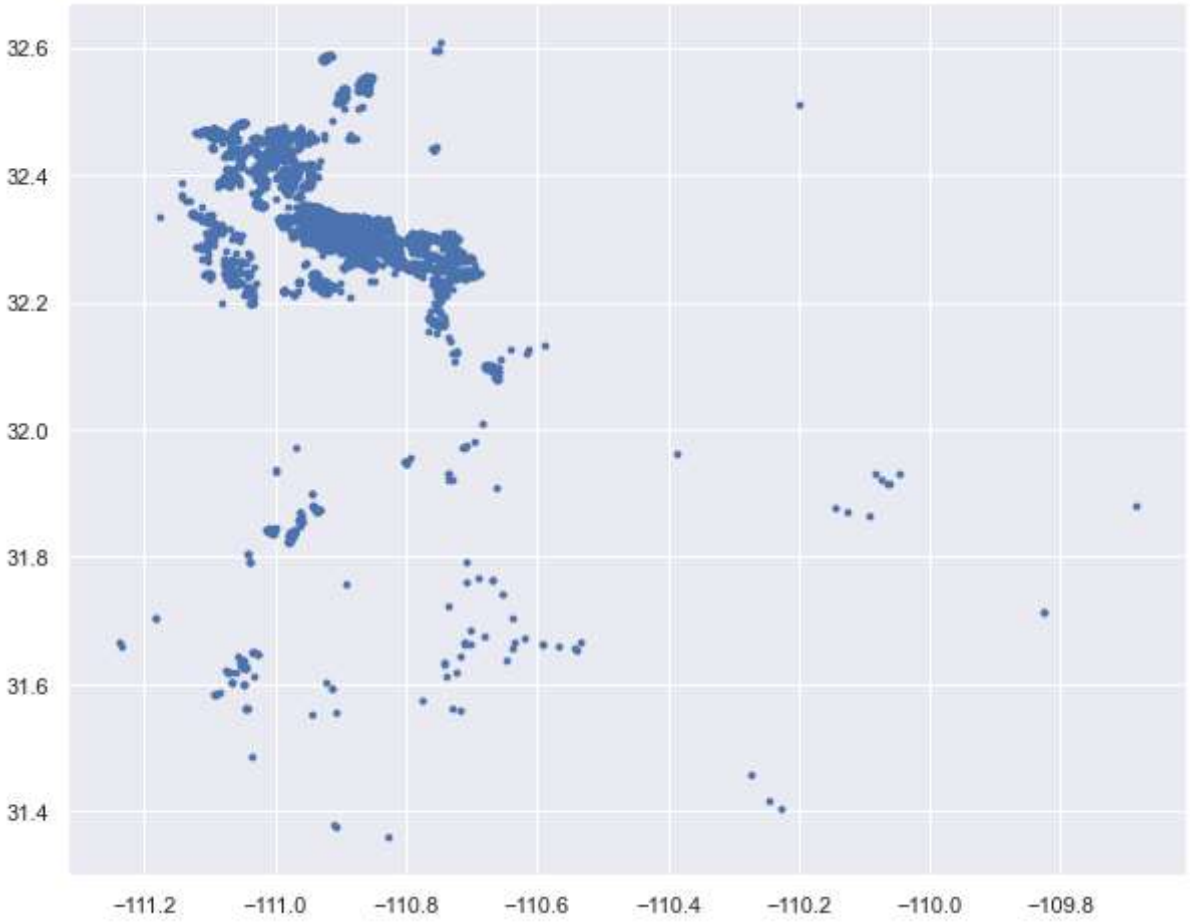


In [140]:
```
dforigin = dforigin[~(dforigin['latitude']>33)]
```

In [141]:
```
dforigin = dforigin[~(dforigin['longitude']>-109.5)]
```

In [142]:
```
plt.figure(figsize=(10,8))
plt.scatter(dforigin['longitude'],dforigin['latitude'],s=8)
```

Out[142]: <matplotlib.collections.PathCollection at 0x1fde2c5adf0>



In [143]:
```
dforigin.shape
```

Out[143]:
```
(4659, 13)
```

In [144]:
```
location= dforigin.iloc[:,2:4]
location
```

Out[144]:

	longitude	latitude
1	-110.813768	32.285162
6	-110.837950	32.327575
7	-111.045441	31.562121
8	-110.918294	32.341609
9	-110.736202	31.721347
...	...	...
4777	-110.820216	32.307646
4778	-110.922291	32.317496
4779	-110.661829	31.907917
4780	-110.858556	32.316373
4781	-111.055528	32.296871

4659 rows × 2 columns

Clustering

In [145]:
```
k_means = KMeans(6)
```

In [146]:
```
k_means.fit(location)
```

Out[146]:
```
KMeans(n_clusters=6)
```

In [147]:
```
k_means.labels_
```

Out[147]:
```
array([4, 1, 2, ..., 2, 1, 0])
```

Clustering Resutls

In [148]:
```
clusters = k_means.fit_predict(location)
```

In [149]:
```
clusters
```

Out[149]:
```
array([0, 4, 1, ..., 1, 4, 2])
```

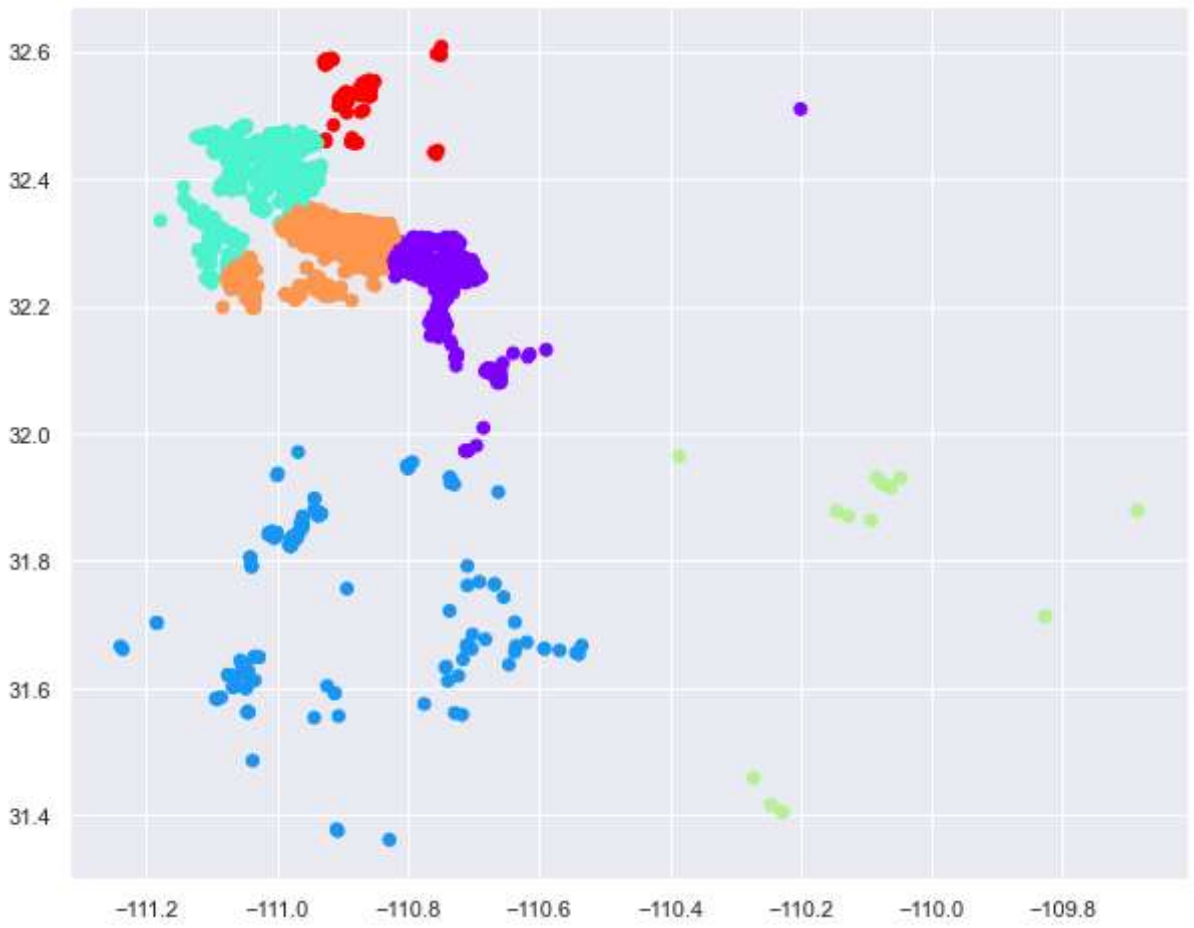
In [150]:
```
dforigin["Clusters"] = clusters
dforigin
```

Out[150]:

	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	sqrt_ft	garage	fireplaces	H(
1	3411450.0	85750	-110.813768	32.285162	3.21	15393.00	1995	4	6.0	6396.0	3.0	5.0	55.
6	3250000.0	85750	-110.837950	32.327575	3.53	18936.11	2007	5	6.0	6480.0	3.0	2.0	141.
7	2776518.0	85640	-111.045441	31.562121	147.18	7330.36	1935	5	5.0	5067.0	5.0	5.0	0.
8	2050000.0	85718	-110.918294	32.341609	1.65	24353.00	2009	5	6.0	7230.0	3.0	4.0	357.
9	2100000.0	85637	-110.736202	31.721347	29.60	11723.00	1995	6	7.0	6454.0	3.0	4.0	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	
4777	565000.0	85750	-110.820216	32.307646	0.83	4568.71	1986	4	3.0	2813.0	2.0	2.0	6.
4778	535000.0	85718	-110.922291	32.317496	0.18	4414.00	2002	3	2.0	2106.0	2.0	1.0	198.
4779	495000.0	85641	-110.661829	31.907917	4.98	2017.00	2005	5	3.0	3601.0	3.0	1.0	0.
4780	550000.0	85750	-110.858556	32.316373	1.42	4822.01	1990	4	3.0	2318.0	3.0	1.0	43.
4781	550000.0	85745	-111.055528	32.296871	1.01	5822.93	2009	4	4.0	3724.0	3.0	1.0	0.

4659 rows × 14 columns

In [151]:
```
plt.figure(figsize=(10,8))
plt.scatter(dforigin["longitude"], dforigin["latitude"], c = dforigin["Clusters"], cmap = "rainbow")
plt.show()
```



In [152]:
```
df_copy=dforigin.copy()
```

In [153]:
```
df_copy=(df_copy- df_copy.min()) / ( df_copy.max() - df_copy.min())
```

In [154]:
```
##df = df.sample(frac=1)
## Split database
j = np.arange(len(df_copy))
np.random.shuffle(j)
train1 = df_copy.iloc[j[:int(len(df_copy)*0.70)]]
validation1 = df_copy.iloc[j[int(len(df_copy)*0.70):int(len(df_copy)*0.85)]]
test1 = df_copy.iloc[j[int(len(df_copy)*0.85):]]
```

Train

In [155]:
```
y_train1=train1['taxes']
```

In [156]:
```
X_train1=train1[['sqrt_ft','HOA','fireplaces','year_built','latitude','longitude']]
```

In [157]:
```
y_train1=y_train1.to_numpy()
X_train1=X_train1.to_numpy()
```

In [158]:
```
knn_reg.fit(X_train1,y_train1)
```

In [159]:
```
y_hatknn1 = knn_reg.predict(X_train1,4)
```

In [160]:
```
R2(y_train1,y_hatknn1)
```

Out[160]:
```
0.7369313367816959
```

In [161]:
```
## X_test1=test1[['sqrt_ft','year_built','fireplaces','HOA','Clusters','bathrooms','bedrooms','garage','lot_acre
```

validation

In [162]:
```
y_val1=validation1['taxes']
```

In [163]:
```
X_val1=validation1[['sqrt_ft','HOA','fireplaces','year_built','latitude','longitude']]
```

In [164]:
```
y_val1=y_val1.to_numpy()
X_val1=X_val1.to_numpy()
```

In [187]:
```
y_hatknnv1 = knn_reg.predict(X_val1,10)
```

In [188]:
```
R2(y_val1,y_hatknnv1)
```

Out[188]: 0.5907467527876548

test

In [189]:
```
y_test1=test1['taxes']
```

In [190]:
```
X_test1=test1[['sqrt_ft','HOA','fireplaces','year_built','latitude','longitude']]
```

In [191]:
```
y_test1=y_test1.to_numpy()
X_test1=X_test1.to_numpy()
```

In [192]:
```
y_hatknnt1 = knn_reg.predict(X_test1,10)
```

In [193]:
```
R2(y_test1,y_hatknnt1)
```

Out[193]: 0.6410438019417197

more plots

In [194]:
```
R2_test1 = {k:0 for v,k in enumerate(range(1,21))}
for n in range(1,21):
    test_pred = knn_reg.predict(X_test1, n)
    R2_test1[n] = R2(y_test1,test_pred)
```

In [195]:
```
R2_test1
```

Out[195]:
```
{1: 0.44462146082675136,
 2: 0.59127556097871,
 3: 0.6198556328329908,
 4: 0.6443143266766629,
 5: 0.6532436159425421,
 6: 0.6483538361564356,
 7: 0.6356905708320137,
 8: 0.6332487518917216,
 9: 0.6327836663981214,
 10: 0.6410438019417197,
 11: 0.6428269104816,
 12: 0.6429090594502636,
 13: 0.6377737218014929,
 14: 0.6303980169117187,
 15: 0.6301789241594624,
 16: 0.6283768552480906,
 17: 0.623338777096091,
 18: 0.6232592637272715,
 19: 0.6211039766286401,
 20: 0.6172333337797711}
```

In [196]:
```
R2_train1 = {k:0 for v,k in enumerate(range(1,21))}
for n in range(1,21):
    train_pred = knn_reg.predict(X_train1, n)
    R2_train1[n] = R2(y_train1,train_pred)
```

In [197]:
```
R2_train1
```

Out[197]:
```
{1: 0.9999513579909443,
 2: 0.8355769688297043,
 3: 0.7754892988026754,
 4: 0.7369313367816959,
 5: 0.7173569967584047,
 6: 0.698516631180065,
 7: 0.6895973662304408,
 8: 0.6772827641221111,
 9: 0.6675734667143021,
 10: 0.6577496211240377,
 11: 0.6473770589142018,
 12: 0.6414294137714769,
 13: 0.6346331257030617,
 14: 0.6300966446235884,
 15: 0.6253474734811865,
 16: 0.6234673996773887,
 17: 0.6213927176198313,
 18: 0.6173400034739875,
 19: 0.6147193492449182,
 20: 0.6107780028816299}
```

In [199]:
```
dfR2= pd.DataFrame([[key, R2_test1[key]] for key in R2_test1.keys()], columns=['K neighbours', 'R2test'])
dfR2
```

Out[199]:

	K neighbours	R2test
0	1	0.444621
1	2	0.591276
2	3	0.619856
3	4	0.644314
4	5	0.653244
5	6	0.648354
6	7	0.635691
7	8	0.633249
8	9	0.632784
9	10	0.641044
10	11	0.642827
11	12	0.642909
12	13	0.637774
13	14	0.630398
14	15	0.630179
15	16	0.628377
16	17	0.623339
17	18	0.623259
18	19	0.621104
19	20	0.617233

In [200]:
```
dfR21= pd.DataFrame([[R2_train1[key]] for key in R2_train1.keys()], columns=['R2train'])
dfR21
```

Out[200]:

	R2train
0	0.999951
1	0.835577
2	0.775489
3	0.736931
4	0.717357
5	0.698517
6	0.689597
7	0.677283
8	0.667573
9	0.657750
10	0.647377
11	0.641429
12	0.634633
13	0.630097
14	0.625347
15	0.623467
16	0.621393
17	0.617340
18	0.614719
19	0.610778

In [201]:
```
dfR2['R2train']=dfR21['R2train']
```

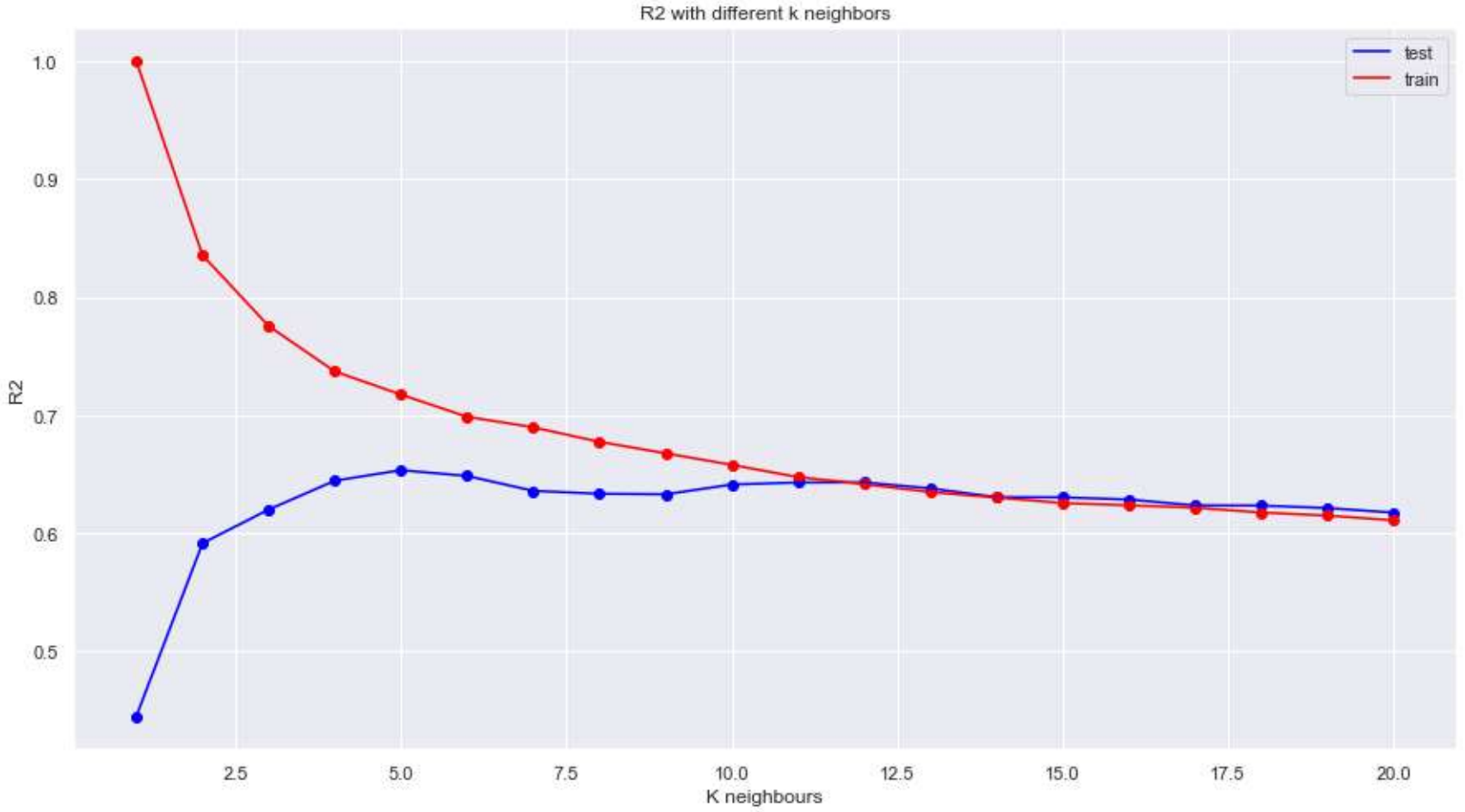
In [202]:
```
dfR2
```

Out[202]:

	K neighbours	R2test	R2train
0	1	0.444621	0.999951
1	2	0.591276	0.835577
2	3	0.619856	0.775489
3	4	0.644314	0.736931
4	5	0.653244	0.717357
5	6	0.648354	0.698517
6	7	0.635691	0.689597
7	8	0.633249	0.677283
8	9	0.632784	0.667573
9	10	0.641044	0.657750
10	11	0.642827	0.647377
11	12	0.642909	0.641429
12	13	0.637774	0.634633
13	14	0.630398	0.630097
14	15	0.630179	0.625347
15	16	0.628377	0.623467
16	17	0.623339	0.621393
17	18	0.623259	0.617340
18	19	0.621104	0.614719
19	20	0.617233	0.610778

In [207]:
```
plt.plot(dfR2['K neighbours'], dfR2['R2test'],label='test',color='blue')
plt.plot(dfR2['K neighbours'], dfR2['R2train'],label='train',color='red')
plt.scatter(dfR2['K neighbours'], dfR2['R2test'],color='blue')
plt.scatter(dfR2['K neighbours'], dfR2['R2train'],color='red')
plt.legend()
plt.xlabel('K neighbours')
plt.ylabel('R2')
plt.title('R2 with different k neighbors')
```

Out[207]:
```
Text(0.5, 1.0, 'R2 with different k neighbors')
```



In [ ]: