lab1

October 6, 2023

[184]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt # for plotting
from sklearn.linear_model import LinearRegression # for linear regression
from sklearn.model_selection import train_test_split # for splitting data␣
↪between train and test
```

1 Part 1: Intro to Pandas with Housing Data

1.1 House Sales for King County (May 2014 - May 2015)

This dataset contains house sale prices for King County, which includes Seattle. It includes homes sold between May 2014 and May 2015.

- `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

1.2 Loading & Exploring data

[189]:
```
df = pd.read_csv('lab1_data.csv')
df.head() # look at first few rows
```

[189]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	…	grade	sqft_above	sqft_basement	yr_built	yr_renovated	zipcode	lat	long	sqft_living15	sqft_lot15
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	…	7	1180	0	1955	0	98178	47.5112	-122.257	1340	5650
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	…	7	2170	400	1951	1991	98125	47.7210	-122.319	1690	7639
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	…	6	770	0	1933	0	98028	47.7379	-122.233	2720	8062
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	…	7	1050	910	1965	0	98136	47.5208	-122.393	1360	5000
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	…	8	1680	0	1987	0	98074	47.6168	-122.045	1800	7503

[5 rows x 21 columns]

[191]:
```
df.shape # dimensions (num rows, num columns)
```

[191]: (21613, 21)

[193]:
```
df.describe() # summary data for each column (some don't make sense)
```

[193]:

	id	price	bedrooms	bathrooms	sqft_living
count	2.161300e+04	2.161300e+04	21613.000000	21613.000000	21613.000000
mean	4.580302e+09	5.400881e+05	3.370842	2.114757	2079.899736
std	2.876566e+09	3.671272e+05	0.930062	0.770163	918.440897

```
min    1.000102e+06  7.500000e+04      0.000000      0.000000    290.000000
25%    2.123049e+09  3.219500e+05      3.000000      1.750000   1427.000000
50%    3.904930e+09  4.500000e+05      3.000000      2.250000   1910.000000
75%    7.308900e+09  6.450000e+05      4.000000      2.500000   2550.000000
max    9.900000e+09  7.700000e+06     33.000000      8.000000  13540.000000

           sqft_lot        floors    waterfront          view     condition  \
count  2.161300e+04  21613.000000  21613.000000  21613.000000  21613.000000
mean   1.510697e+04      1.494309      0.007542      0.234303      3.409430
std    4.142051e+04      0.539989      0.086517      0.766318      0.650743
min    5.200000e+02      1.000000      0.000000      0.000000      1.000000
25%    5.040000e+03      1.000000      0.000000      0.000000      3.000000
50%    7.618000e+03      1.500000      0.000000      0.000000      3.000000
75%    1.068800e+04      2.000000      0.000000      0.000000      4.000000
max    1.651359e+06      3.500000      1.000000      4.000000      5.000000

              grade    sqft_above  sqft_basement      yr_built  yr_renovated  \
count  21613.000000  21613.000000   21613.000000  21613.000000  21613.000000
mean       7.656873   1788.390691     291.509045   1971.005136     84.402258
std        1.175459    828.090978     442.575043     29.373411    401.679240
min        1.000000    290.000000       0.000000   1900.000000      0.000000
25%        7.000000   1190.000000       0.000000   1951.000000      0.000000
50%        7.000000   1560.000000       0.000000   1975.000000      0.000000
75%        8.000000   2210.000000     560.000000   1997.000000      0.000000
max       13.000000   9410.000000    4820.000000   2015.000000   2015.000000

            zipcode           lat          long  sqft_living15     sqft_lot15
count  21613.000000  21613.000000  21613.000000   21613.000000   21613.000000
mean   98077.939805     47.560053   -122.213896    1986.552492   12768.455652
std       53.505026      0.138564      0.140828     685.391304   27304.179631
min    98001.000000     47.155900   -122.519000     399.000000     651.000000
25%    98033.000000     47.471000   -122.328000    1490.000000    5100.000000
50%    98065.000000     47.571800   -122.230000    1840.000000    7620.000000
75%    98118.000000     47.678000   -122.125000    2360.000000   10083.000000
max    98199.000000     47.777600   -121.315000    6210.000000  871200.000000
```

** Think #1 ** For which columns are these summary statistics actually useful?

1.3 Selecting data

[197]:
```
df.loc[25:100] # get rows 25 - 100 (kind of pointless...)
```

[197]:
```
             id             date     price  bedrooms  bathrooms  sqft_living  \
25   1202000200  20141103T000000  233000.0         3       2.00         1710
26   1794500383  20140626T000000  937000.0         3       1.75         2450
27   3303700376  20141201T000000  667000.0         3       1.00         1400
28   5101402488  20140624T000000  438000.0         3       1.75         1520
```

```
29   1873100390  20150302T000000  719000.0         4       2.50         2570
..          …               …       …       …        …          …
96   3422049190  20150330T000000  247500.0         3       1.75         1960
97   1099611230  20140912T000000  199000.0         4       1.50         1160
98    722079104  20140711T000000  314000.0         3       1.75         1810
99   7338200240  20140516T000000  437500.0         3       2.50         2320
100  1952200240  20140611T000000  850830.0         3       2.50         2070

     sqft_lot  floors  waterfront  view  …  grade  sqft_above  \
25       4697     1.5           0     0  …      6        1710
26       2691     2.0           0     0  …      8        1750
27       1581     1.5           0     0  …      8        1400
28       6380     1.0           0     0  …      7         790
29       7173     2.0           0     0  …      8        2570
..        …     …          …   …  …    …         …
96      15681     1.0           0     0  …      7        1960
97       6400     1.0           0     0  …      7        1160
98      41800     1.0           0     0  …      7        1210
99      36847     2.0           0     2  …      9        2320
100     13241     1.5           0     0  …      9        1270

     sqft_basement  yr_built  yr_renovated  zipcode      lat     long  \
25               0      1941             0    98002  47.3048 -122.218
26             700      1915             0    98119  47.6386 -122.360
27               0      1909             0    98112  47.6221 -122.314
28             730      1948             0    98115  47.6950 -122.304
29               0      2005             0    98052  47.7073 -122.110
..             …       …            …       …      …      …
96               0      1967             0    98032  47.3576 -122.277
97               0      1975             0    98023  47.3036 -122.378
98             600      1980             0    98038  47.4109 -121.958
99               0      1992             0    98045  47.4838 -121.714
100            800      1910             0    98102  47.6415 -122.315

     sqft_living15  sqft_lot15
25            1030        4705
26            1760        3573
27            1860        3861
28            1520        6235
29            2630        6026
..             …         …
96            1750       15616
97            1160        6400
98            1650      135036
99            2550       35065
100           2200        4500
```

```
[76 rows x 21 columns]
```

[199]:
```
df.loc[371, 'price'] # get price of 371st row
```

[199]:
```
315000.0
```

[201]:
```
df.iloc[25:100, 0:5] # get 25th-100th rows and first 5 columns
```

[201]:

	id	date	price	bedrooms	bathrooms
25	1202000200	20141103T000000	233000.0	3	2.00
26	1794500383	20140626T000000	937000.0	3	1.75
27	3303700376	20141201T000000	667000.0	3	1.00
28	5101402488	20140624T000000	438000.0	3	1.75
29	1873100390	20150302T000000	719000.0	4	2.50
..	…	…	…	…	…
95	1483300570	20140908T000000	905000.0	4	2.50
96	3422049190	20150330T000000	247500.0	3	1.75
97	1099611230	20140912T000000	199000.0	4	1.50
98	722079104	20140711T000000	314000.0	3	1.75
99	7338200240	20140516T000000	437500.0	3	2.50

```
[75 rows x 5 columns]
```

1.4 Sorting data & filtering (w/ Boolean indexing)

[204]:
```
df_by_price = df.sort_values(by='price') # sort data by price column
df_by_price.head() # show cheapest homes
```

[204]:

	id	date	price	bedrooms	bathrooms	sqft_living	\
1149	3421079032	20150217T000000	75000.0	1	0.00	670	
15293	40000362	20140506T000000	78000.0	2	1.00	780	
465	8658300340	20140523T000000	80000.0	1	0.75	430	
16198	3028200080	20150324T000000	81000.0	2	1.00	730	
8274	3883800011	20141105T000000	82000.0	3	1.00	860	

	sqft_lot	floors	waterfront	view	…	grade	sqft_above	\
1149	43377	1.0	0	0	…	3	670	
15293	16344	1.0	0	0	…	5	780	
465	5050	1.0	0	0	…	4	430	
16198	9975	1.0	0	0	…	5	730	
8274	10426	1.0	0	0	…	6	860	

	sqft_basement	yr_built	yr_renovated	zipcode	lat	long	\
1149	0	1966	0	98022	47.2638	-121.906	
15293	0	1942	0	98168	47.4739	-122.280	
465	0	1912	0	98014	47.6499	-121.909	
16198	0	1943	0	98168	47.4808	-122.315	

```
8274               0      1954             0    98146  47.4987 -122.341

       sqft_living15  sqft_lot15
1149            1160       42882
15293           1700       10387
465             1200        7500
16198            860        9000
8274            1140       11250

[5 rows x 21 columns]
```

[206]:
```
max_price = 100000
df_by_price[df_by_price['price'] <= max_price] # show homes with price <=␣
 ↪max_price
```

[206]:

	id	date	price	bedrooms	bathrooms	\
1149	3421079032	20150217T000000	75000.0	1	0.00	
15293	40000362	20140506T000000	78000.0	2	1.00	
465	8658300340	20140523T000000	80000.0	1	0.75	
16198	3028200080	20150324T000000	81000.0	2	1.00	
8274	3883800011	20141105T000000	82000.0	3	1.00	
2141	1623049041	20140508T000000	82500.0	2	1.00	
18468	7999600180	20140529T000000	83000.0	2	1.00	
3767	1523049188	20150430T000000	84000.0	2	1.00	
16714	1322049150	20150305T000000	85000.0	2	1.00	
10253	2422049104	20140915T000000	85000.0	2	1.00	
13756	1788900230	20140722T000000	86500.0	3	1.00	
5866	9320900420	20141014T000000	89000.0	3	1.00	
3108	1721801591	20150219T000000	89950.0	1	1.00	
16530	2114700500	20150418T000000	90000.0	1	1.00	
7992	2734100835	20150303T000000	90000.0	1	1.00	
12551	1049010620	20140513T000000	90000.0	2	1.00	
18939	4239400300	20141129T000000	90000.0	3	1.00	
17580	1423049019	20140523T000000	90000.0	2	1.00	
10770	795000765	20140616T000000	92000.0	2	1.00	
3321	2724200705	20141212T000000	95000.0	2	1.00	
14581	6929602721	20150408T000000	95000.0	2	1.00	
3805	7335400215	20150505T000000	95000.0	1	0.75	
10585	6198400218	20140919T000000	95000.0	2	1.00	
5723	1788800630	20141029T000000	96500.0	3	1.00	
5303	5128000010	20150105T000000	99000.0	2	1.00	
1218	3751600030	20140717T000000	100000.0	2	1.00	
16340	6146600170	20140703T000000	100000.0	2	0.75	
5639	7224000980	20140610T000000	100000.0	4	1.00	
15456	8856000545	20140507T000000	100000.0	2	1.00	
4563	1900000195	20140630T000000	100000.0	2	1.00	
3444	7813200115	20140904T000000	100000.0	2	1.00	

	sqft_living	sqft_lot	floors	waterfront	view	…	grade	\
1149	670	43377	1.0	0	0	…	3	
15293	780	16344	1.0	0	0	…	5	
465	430	5050	1.0	0	0	…	4	
16198	730	9975	1.0	0	0	…	5	
8274	860	10426	1.0	0	0	…	6	
2141	520	22334	1.0	0	0	…	5	
18468	900	8580	1.0	0	0	…	5	
3767	700	20130	1.0	0	0	…	6	
16714	910	9753	1.0	0	0	…	5	
10253	830	9000	1.0	0	0	…	6	
13756	840	9480	1.0	0	0	…	6	
5866	900	4750	1.0	0	0	…	6	
3108	570	4080	1.0	0	0	…	5	
16530	560	4120	1.0	0	0	…	4	
7992	780	4000	1.0	0	0	…	5	
12551	790	2640	1.0	0	0	…	7	
18939	980	2490	2.0	0	0	…	6	
17580	580	7500	1.0	0	0	…	5	
10770	760	5500	1.5	0	0	…	5	
3321	800	8550	1.0	0	0	…	7	
14581	960	7000	1.0	0	0	…	4	
3805	760	5746	1.0	0	0	…	5	
10585	1070	20450	1.0	0	0	…	6	
5723	840	12091	1.0	0	0	…	6	
5303	960	8236	1.0	0	0	…	6	
1218	770	17334	1.0	0	0	…	7	
16340	660	5240	1.0	0	0	…	4	
5639	1120	2685	1.0	0	0	…	5	
15456	910	22000	1.0	0	0	…	6	
4563	930	7623	1.0	0	0	…	6	
3444	790	6426	1.0	0	0	…	6	

	sqft_above	sqft_basement	yr_built	yr_renovated	zipcode	lat	\
1149	670	0	1966	0	98022	47.2638	
15293	780	0	1942	0	98168	47.4739	
465	430	0	1912	0	98014	47.6499	
16198	730	0	1943	0	98168	47.4808	
8274	860	0	1954	0	98146	47.4987	
2141	520	0	1951	0	98168	47.4799	
18468	900	0	1918	0	98168	47.4727	
3767	700	0	1949	0	98168	47.4752	
16714	910	0	1947	0	98032	47.3897	
10253	830	0	1939	0	98032	47.3813	
13756	840	0	1960	0	98023	47.3277	
5866	900	0	1969	0	98023	47.3026	

3108	570	0	1942	0	98146	47.5098
16530	560	0	1947	0	98106	47.5335
7992	780	0	1905	0	98108	47.5424
12551	790	0	1973	0	98034	47.7351
18939	980	0	1969	0	98092	47.3170
17580	580	0	1943	0	98178	47.4852
10770	760	0	1947	0	98168	47.5045
3321	800	0	1947	0	98198	47.4075
14581	960	0	1918	0	98198	47.3864
3805	760	0	1915	0	98002	47.3046
10585	1070	0	1948	0	98058	47.4338
5723	840	0	1959	0	98023	47.3281
5303	960	0	1948	0	98058	47.4698
1218	770	0	1978	0	98001	47.2997
16340	660	0	1912	0	98032	47.3881
5639	860	260	1939	0	98055	47.4904
15456	910	0	1956	0	98001	47.2777
4563	930	0	1942	0	98166	47.4670
3444	790	0	1944	0	98178	47.4933

	long	sqft_living15	sqft_lot15
1149	-121.906	1160	42882
15293	-122.280	1700	10387
465	-121.909	1200	7500
16198	-122.315	860	9000
8274	-122.341	1140	11250
2141	-122.296	1572	10570
18468	-122.270	2060	6533
3767	-122.271	1490	18630
16714	-122.236	1160	7405
10253	-122.243	1160	7680
13756	-122.341	840	9420
5866	-122.363	900	3404
3108	-122.334	890	5100
16530	-122.348	980	4120
7992	-122.321	1150	4000
12551	-122.178	1310	2064
18939	-122.182	980	3154
17580	-122.251	1700	11250
10770	-122.329	1040	5515
3321	-122.294	1490	8550
14581	-122.307	1850	8120
3805	-122.215	970	6696
10585	-122.183	1360	15581
5723	-122.343	840	9324
5303	-122.166	1260	8236
1218	-122.269	1480	17334

```
16340 -122.234           850        5080
5639  -122.203          1120        4838
15456 -122.252          1326        9891
4563  -122.349          1300        7641
3444  -122.245          1380        6946

[31 rows x 21 columns]
```

[208]:
```
df_by_price[df_by_price['waterfront'] == 1] # show homes with waterfront view
```

[208]:
```
              id             date      price  bedrooms  bathrooms  \
18275  2781600195  20141117T000000   285000.0         1       1.00
1168   3523029041  20141009T000000   290000.0         2       0.75
16570  2923039243  20141113T000000   340000.0         4       1.00
6102    222029026  20140917T000000   340000.0         2       0.75
11556  2013802030  20140911T000000   357000.0         3       2.00
…            …               …         …        …          …
2626   7738500731  20140815T000000  4500000.0         5       5.50
8092   1924059029  20140617T000000  4668000.0         5       6.75
1164   1247600105  20141020T000000  5110800.0         5       5.25
1315   7558700030  20150413T000000  5300000.0         6       6.00
3914   9808700762  20140611T000000  7062500.0         5       4.50

       sqft_living  sqft_lot  floors  waterfront  view  …  grade  \
18275         1060     54846     1.0           1     4  …      5
1168           440      8313     1.0           1     3  …      5
16570         1200     11834     1.0           1     3  …      6
6102          1060     48292     1.0           1     2  …      6
11556         2460     53882     1.0           1     4  …      7
…              …        …        …           …    …  …    …
2626          6640     40014     2.0           1     4  …     12
8092          9640     13068     1.0           1     4  …     12
1164          8010     45517     2.0           1     4  …     12
1315          7390     24829     2.0           1     4  …     12
3914         10040     37325     2.0           1     2  …     11

       sqft_above  sqft_basement  yr_built  yr_renovated  zipcode      lat  \
18275        1060              0      1935             0    98070  47.4716
1168          440              0      1943             0    98070  47.4339
16570        1200              0      1972             0    98070  47.4557
6102          560            500      1947             0    98070  47.4285
11556        2460              0      1955             0    98198  47.3811
…             …              …         …             …        …        …
2626         6350            290      2004             0    98155  47.7493
8092         4820           4820      1983          2009    98040  47.5570
1164         5990           2020      1999             0    98033  47.6767
1315         5000           2390      1991             0    98040  47.5631
```

```
3914         7680           2360      1940          2001    98004  47.6500

          long  sqft_living15  sqft_lot15
18275 -122.445           2258       31762
1168  -122.512            880       26289
16570 -122.443           1670       47462
6102  -122.511            750       80201
11556 -122.325           2660       32625
…        …             …          …
2626  -122.280           3030       23408
8092  -122.210           3270       10454
1164  -122.211           3430       26788
1315  -122.210           4320       24619
3914  -122.214           3930       25449

[163 rows x 21 columns]
```

1.4.1 TODO #1

Get all homes with more than 8 bedrooms and sort by price. What do you notice about the relationship between price and bedrooms? Looking at these homes, what columns seem to associate with higher price?

[211]:
```
homes_over_8_rooms = df_by_price[df_by_price['bedrooms'] > 8]
homes_over_8_rooms
```

[211]:
```
                id             date      price  bedrooms  bathrooms  \
8546    424049043  20140811T000000   450000.0         9       7.50
8757   1773100755  20140821T000000   520000.0        11       3.00
4096   1997200215  20140507T000000   599999.0         9       4.50
15870  2402100895  20140625T000000   640000.0        33       1.75
15161  5566100170  20141029T000000   650000.0        10       2.00
19254  8812401450  20141229T000000   660000.0        10       3.00
4235   2902200015  20150106T000000   700000.0         9       3.00
18443  8823901445  20150313T000000   934000.0         9       3.00
13314   627300145  20140814T000000  1148000.0        10       5.25
6079   9822700190  20140808T000000  1280000.0         9       4.50
16844  8823900290  20150317T000000  1400000.0         9       4.00

       sqft_living  sqft_lot  floors  waterfront  view  …  grade  \
8546          4050      6504     2.0           0     0  …      7
8757          3000      4960     2.0           0     0  …      7
4096          3830      6988     2.5           0     0  …      7
15870         1620      6000     1.0           0     0  …      7
15161         3610     11914     2.0           0     0  …      7
19254         2920      3745     2.0           0     0  …      7
4235          3680      4400     2.0           0     0  …      7
```

```
18443         2820      4480     2.0           0     0  …      7
13314         4590     10920     1.0           0     2  …      9
6079          3650      5000     2.0           0     0  …      8
16844         4620      5508     2.5           0     0  …     11

       sqft_above  sqft_basement  yr_built  yr_renovated  zipcode      lat  \
8546         4050              0      1996             0    98144  47.5923
8757         2400            600      1918          1999    98106  47.5560
4096         2450           1380      1938             0    98103  47.6927
15870        1040            580      1947             0    98103  47.6878
15161        3010            600      1958             0    98006  47.5705
19254        1860           1060      1913             0    98105  47.6635
4235         2830            850      1908             0    98102  47.6374
18443        1880            940      1918             0    98105  47.6654
13314        2500           2090      2008             0    98004  47.5861
6079         2530           1120      1915          2010    98105  47.6604
16844        3870            750      1915             0    98105  47.6684

          long  sqft_living15  sqft_lot15
8546  -122.301           1448        3866
8757  -122.363           1420        4960
4096  -122.338           1460        6291
15870 -122.331           1330        4700
15161 -122.175           2040       11914
19254 -122.320           1810        3745
4235  -122.324           1960        2450
18443 -122.307           2460        4400
13314 -122.113           2730       10400
6079  -122.289           2510        5000
16844 -122.309           2710        4320

[11 rows x 21 columns]
```

There doesn't seem to be a straightforward relationship between the number of bedrooms and the price within this subset, as the most expensive house does not have the most bedrooms. It seems that the grade show some relation to the price - the highest-priced homes in the above subset also have the highest grades. While grade might influence price, it's not the sole determinant.

1.5 Operations & adding columns

[215]:
```
df['price'].mean() # mean price of home
```

[215]:
```
540088.1417665294
```

[217]:
```
df['price'].describe() # more summary statistics for home prices
```

[217]:
```
count    2.161300e+04
mean     5.400881e+05
std      3.671272e+05
min      7.500000e+04
25%      3.219500e+05
50%      4.500000e+05
75%      6.450000e+05
max      7.700000e+06
Name: price, dtype: float64
```

[219]:
```
df['price_per_sqft'] = df['price'] / df['sqft_living'] # add new column to get price / sqft (interior)
```

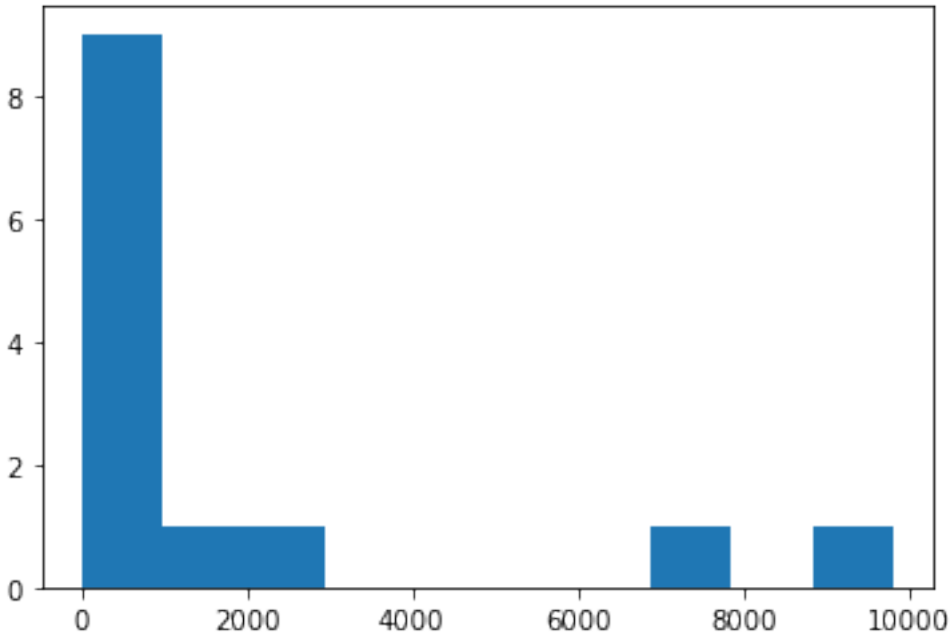
[221]:
```
current_year = 2021
df['age'] = df['yr_built'].apply(lambda yr: current_year - yr) # get age of house by applying function to each value in series

# df['age'] = current_year - df['yr_built'] # equivalent command (but I had to show y'all apply)
```

[223]:
```
hist_bed = df['bedrooms'].value_counts() # frequency count of number of houses with same number of bedrooms (about that mansion...)
hist_bed = hist_bed.sort_index() # sort by index (num of bedrooms)
```

[225]:
```
plt.hist(df['bedrooms'].value_counts()) # plot histogram
```

[225]:
```
(array([9., 1., 1., 0., 0., 0., 0., 1., 0., 1.]),
 array([1.0000e+00, 9.8330e+02, 1.9656e+03, 2.9479e+03, 3.9302e+03,
        4.9125e+03, 5.8948e+03, 6.8771e+03, 7.8594e+03, 8.8417e+03,
        9.8240e+03]),
 <BarContainer object of 10 artists>)
```



1.6 Grouping

[228]:
```
df.groupby('waterfront').apply(lambda grp: grp.mean())# group by waterfront and␣
↪look at means for each column
```

```
/tmp/ipykernel_1050/3393052474.py:1: FutureWarning: Dropping of nuisance columns
in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future
version this will raise TypeError.  Select only valid columns before calling the
reduction.
  df.groupby('waterfront').apply(lambda grp: grp.mean())# group by waterfront
and look at means for each column
```

[228]:

waterfront	id	price	bedrooms	bathrooms	sqft_living	\
0	4.580984e+09	5.315636e+05	3.371375	2.110478	2071.587972	
1	4.490512e+09	1.661876e+06	3.300613	2.677914	3173.687117	

waterfront	sqft_lot	floors	waterfront	view	condition	…	\
0	15028.964196	1.493193	0.0	0.207459	3.408485	…	
1	25371.828221	1.641104	1.0	3.766871	3.533742	…	

waterfront	sqft_basement	yr_built	yr_renovated	zipcode	lat	\
0	288.400000	1971.072121	81.149930	98077.798555	47.560225	

```
1              700.644172  1962.190184    512.392638  98096.527607  47.537364

                   long  sqft_living15    sqft_lot15  price_per_sqft        age
waterfront
0           -122.213382    1981.386667  12695.378089      262.302879  49.927879
1           -122.281601    2666.349693  22385.104294      508.096412  58.809816

[2 rows x 22 columns]
```

1.6.1 TODO #2

1) Group homes by number of floors and get median of each row for each group. What do you notice?

2) For homes that have been renovated, plot a histogram grouping homes by year built. (assuming 0.0 means home was never renovated)

[231]:
```
# 2.1
median_floor = df.groupby('floors').apply(lambda grp: grp.median())
median_floor
```

```
/tmp/ipykernel_1050/3950524138.py:2: FutureWarning: Dropping of nuisance columns
in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future
version this will raise TypeError.  Select only valid columns before calling the
reduction.
  median_floor = df.groupby('floors').apply(lambda grp: grp.median())
```

[231]:
```
                  id     price  bedrooms  bathrooms  sqft_living  sqft_lot  \
floors
1.0     3.884801e+09  390000.0       3.0      1.750       1630.0    8337.0
1.5     3.856905e+09  524475.0       3.0      1.750       1760.0    5962.5
2.0     3.982700e+09  542950.0       4.0      2.500       2440.0    7089.0
2.5     3.751602e+09  799200.0       4.0      2.500       2850.0    5474.0
3.0     3.448000e+09  490000.0       3.0      2.500       1500.0    1323.0
3.5     1.972201e+09  534500.0       3.0      2.625       1730.0    1331.0

        floors  waterfront  view  condition  …  sqft_basement  yr_built  \
floors                                       …
1.0        1.0         0.0   0.0        3.0  …          200.0    1962.0
1.5        1.5         0.0   0.0        4.0  …            0.0    1928.0
2.0        2.0         0.0   0.0        3.0  …            0.0    1998.0
2.5        2.5         0.0   0.0        3.0  …           60.0    1977.0
3.0        3.0         0.0   0.0        3.0  …            0.0    2007.0
3.5        3.5         0.0   0.0        3.0  …            0.0    2005.5

        yr_renovated  zipcode       lat      long  sqft_living15  sqft_lot15  \
floors
1.0              0.0  98070.0  47.56150 -122.2650         1680.0      8173.5
```

```
1.5              0.0  98115.0  47.62850 -122.3085          1660.0       5700.0
2.0              0.0  98055.0  47.56410 -122.1680          2260.0       7113.0
2.5              0.0  98106.0  47.61870 -122.2950          2240.0       5352.0
3.0              0.0  98109.0  47.67120 -122.3460          1470.0       1466.0
3.5              0.0  98104.0  47.65295 -122.3335          1405.0       1331.0

        price_per_sqft   age
floors
1.0         242.187500  59.0
1.5         288.289458  93.0
2.0         233.644860  23.0
2.5         297.235023  44.0
3.0         325.963719  14.0
3.5         333.233444  15.5

[6 rows x 22 columns]
```

After grouping homes by the number of floors and calculating the median for each attribute, we observed the following trends:
(1) Homes with 2.5 floors tend to have the highest median price(799200.0).
(2) A higher number of floors doesn't necessarily mean more bedrooms or bathrooms.
(3) The median living space tends to increase with the number of floors, but homes with 3 and 3.5 floors are an exception with smaller living spaces.
(4) The lot size generally decreases as the number of floors increases.

[234]:
```
# 2.2
renovated = df[df['yr_renovated'] > 0]
renovated
```

[234]:
```
                id             date      price  bedrooms  bathrooms  \
1       6414100192  20141209T000000   538000.0         3       2.25
35      9547205180  20140613T000000   696000.0         3       2.50
95      1483300570  20140908T000000   905000.0         4       2.50
103     2450000295  20141007T000000  1088000.0         3       2.50
115     3626039325  20141121T000000   740500.0         3       3.50
…              …              …         …       …         …
19622   7351200295  20150114T000000  1150000.0         3       1.75
20057    126039256  20140904T000000   434900.0         3       2.00
20444   4305600360  20150225T000000   500012.0         4       2.50
20447   3319500628  20150212T000000   356999.0         3       1.50
20962   1278000210  20150311T000000   110000.0         2       1.00

        sqft_living  sqft_lot  floors  waterfront  view  …  sqft_basement  \
1              2570      7242     2.0           0     0  …            400
35             2300      3060     1.5           0     0  …            790
95             3300     10250     1.0           0     0  …            910
103            2920      8113     2.0           0     0  …              0
```

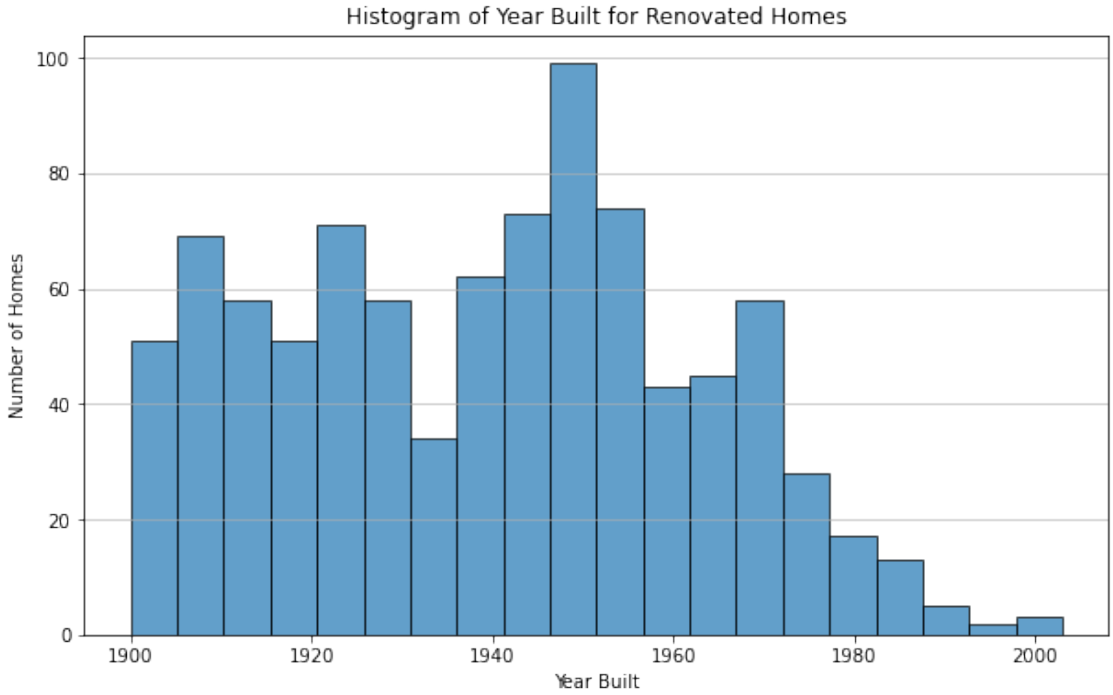
```
115           4380      6350     2.0           0     0  …           1600
…              …         …        …          …    …   …             …
19622         1760      6788     2.0           1     4  …              0
20057         1520      5040     2.0           0     0  …              0
20444         2400      9612     1.0           0     0  …           1170
20447         1010      1546     2.0           0     0  …              0
20962          828      4524     1.0           0     0  …              0

       yr_built  yr_renovated  zipcode      lat     long  sqft_living15  \
1          1951          1991    98125  47.7210 -122.319           1690
35         1930          2002    98115  47.6827 -122.310           1590
95         1946          1991    98040  47.5873 -122.249           1950
103        1950          2010    98004  47.5814 -122.196           2370
115        1900          1999    98117  47.6981 -122.368           1830
…           …             …       …      …       …              …
19622      1940          1960    98125  47.7336 -122.284           1630
20057      1977          2006    98177  47.7770 -122.362           1860
20444      1962          2009    98059  47.4799 -122.127           2430
20447      1971          2014    98144  47.5998 -122.311           1010
20962      1968          2007    98001  47.2655 -122.244            828

       sqft_lot15  price_per_sqft  age
1            7639      209.338521   70
35           3264      302.608696   91
95           6045      274.242424   75
103          8113      372.602740   71
115          6350      169.063927  121
…             …              …   …
19622        7588      653.409091   81
20057        8710      286.118421   44
20444        5539      208.338333   59
20447        1517      353.464356   50
20962        5402      132.850242   53

[914 rows x 23 columns]
```

[236]:
```
plt.figure(figsize=(10, 6))
plt.hist(renovated['yr_built'], bins=20, edgecolor='black', alpha=0.7)
plt.title('Histogram of Year Built for Renovated Homes')
plt.xlabel('Year Built')
plt.ylabel('Number of Homes')
plt.grid(axis='y', alpha=0.75)
plt.show()
```



From the above histogram, we observed:
(1)An increasing trend of renovations for homes built from the 1940s onwards, peaking around the 1950s.
(2)Fewer renovations for homes built in more recent decades(from 1975s to 2000s), which may be expected since newer homes might require fewer renovations.

1.7 What wasn't covered

- handling missing data
- categoricals (setting values to be qualitative categories (e.g. "a", "b", "c" for letter grades)
- merging data (combining multiple dataframes)
- string methods (e.g. lower() to make string lower case)
- time series

2 Part 2: Basic Linear Regression (predicting housing prices)

2.1 Simple linear regression: $Y = B_0 + B_1 x$

where Y is `price`, and x is `sqft_living`

[241]:
```
x_data = df[['sqft_living']]
y_data = df['price']

# split data between train and test set (20% of data for test, rest for␣
↪training)
```

```
x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.
↪33, random_state=371)

m = LinearRegression() # model

m.fit(x_train, y_train) # fit model to data

intercept = m.intercept_
coefs = m.coef_ # on average, price increases by this much for every increase␣
↪of 1 x_vars (sqft_living). B/c just 1 x var, this is slope.
print(intercept)
print(coefs)
```

```
-57405.355907862424
[287.46247817]
```

[242]:
```
y_pred = m.predict(x_test)
y_pred # predicted prices for homes of certain square foot
```

[242]:
```
array([393910.73482066, 779110.45556984, 419782.35785605, …,
       698620.96168195, 287549.61789737, 244430.24617172])
```

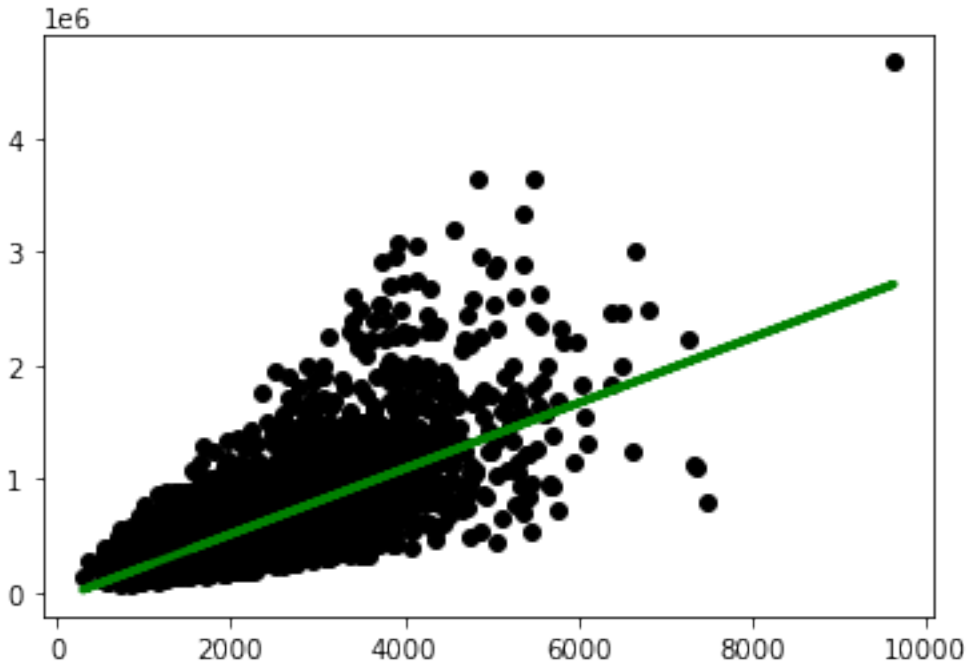
[243]:
```
plt.scatter(x_test, y_test, color='black') # plot data

print(x_test.shape)
print(y_test.shape)

plt.plot(x_test.squeeze(), y_pred, color='green', linewidth = 3) # plot␣
↪regression line of best fit
```

```
(7133, 1)
(7133,)
```

[243]:
```
[<matplotlib.lines.Line2D at 0x7f11ab2bdc70>]
```



2.2 Multiple linear regression: $Y = B_0 + B_1X_1 + B_2X_2 + B_3X_3$

where Y is `price` and X_{1-3} is `[sqft_living, bedrooms, waterfront]`

[246]:
```
x_cols = ['sqft_living', 'bedrooms', 'waterfront', 'view', 'condition']

x = df[x_cols]
y = df['price']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,␣
↪random_state=371)

m2 = LinearRegression()

m2.fit(x_train, y_train)

print(m2.intercept_)
print(m2.coef_)
```

```
-68250.91795627621
[ 2.89032231e+02 -4.65209791e+04  5.77843284e+05  6.90160559e+04
  4.20027000e+04]
```

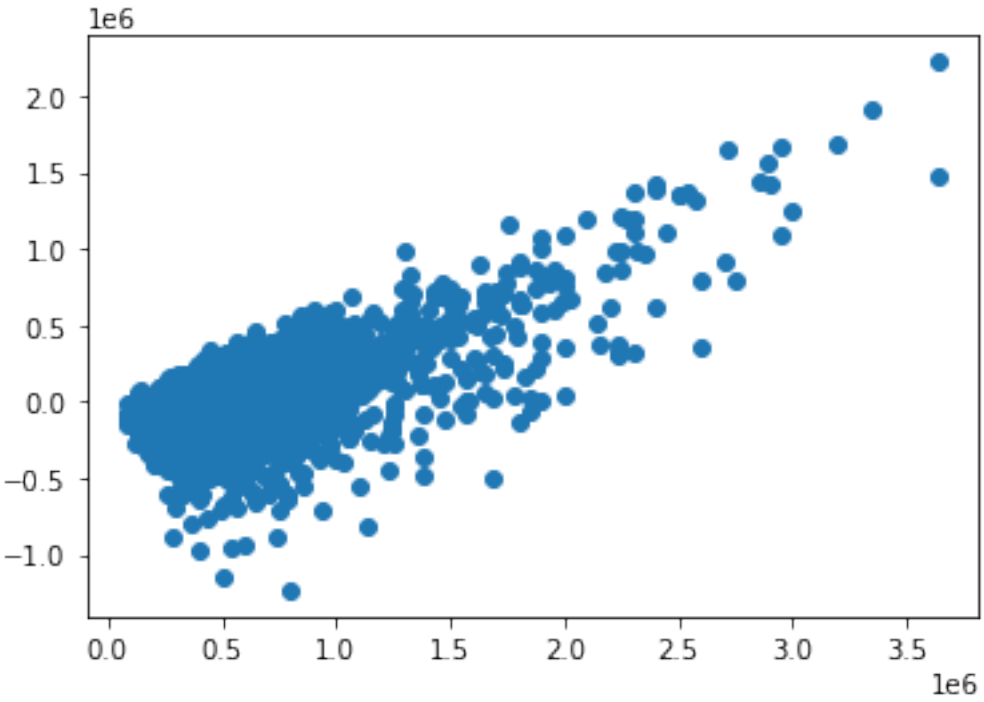
[251]:
```
y_pred = m2.predict(x_test) # use test data to get predicted values
y_pred
```

[251]:
```
array([371974.84700568, 666236.07797343, 397987.74776931, …,
       600034.48708751, 736155.45769648, 614486.09862286])
```

[253]:
```
residuals = y_test - y_pred # residuals are error between predicted and actual
```

[255]:
```
plt.scatter(y_test, residuals) # plot residuals
```

[255]:
```
<matplotlib.collections.PathCollection at 0x7f11ac006ee0>
```



2.2.1 TODO #3

1) Create a linear model that predicts `sqft_living` using one other other column (your choice). Test your model with 33% of your data (which should not be used in training). Plot the data and line of best fit. Plot the residuals. How well does a linear model represent this relationship?

2) Create a new linear model using multiple columns to predict `sqft_living`. Test your model with 33% of your data (which should not be used in training). Plot the residuals. How well does a linear model represent this relationship?

[258]:
```
# 3.1: fit model
x_data = df[['yr_built']]
y_data = df['sqft_living']

# split data between train and test set
```

```
x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.
↪33, random_state=371)

m = LinearRegression() # model

m.fit(x_train, y_train) # fit model to data

intercept = m.intercept_
coefs = m.coef_
print(intercept)
print(coefs)
```

```
-17588.06947662306
[9.98054485]
```

[260]:
```
y_pred = m.predict(x_test)
y_pred
```

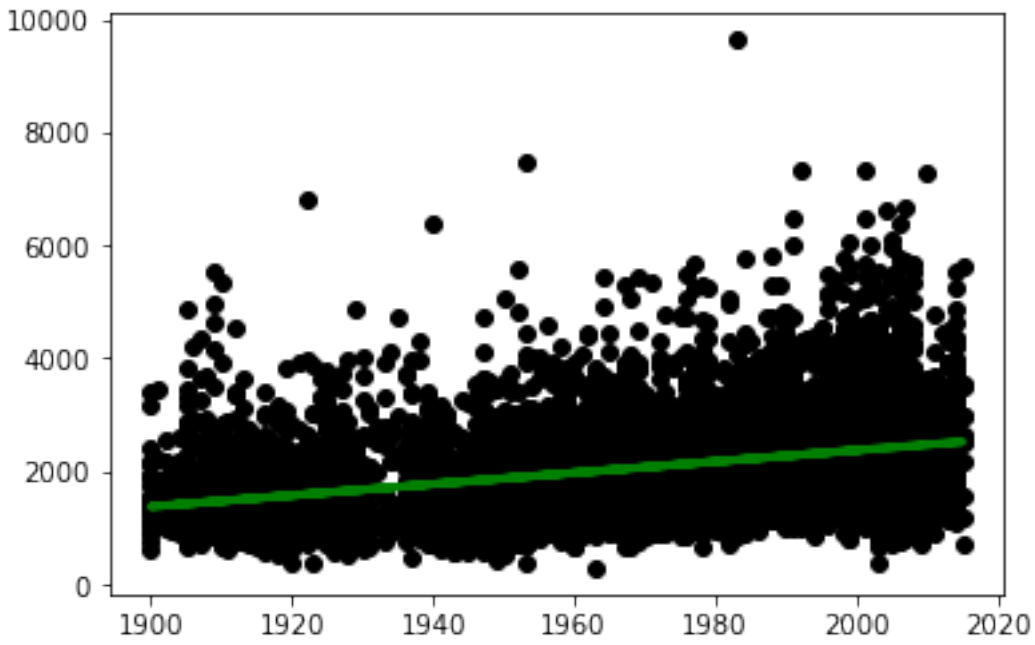
[260]:
```
array([2373.02021429, 2143.46768285, 2203.35095192, …, 1953.83733078,
       1893.95406171, 1414.88790913])
```

[262]:
```
# plot data w/ line of best fit
plt.scatter(x_test, y_test, color='black') # plot data

print(x_test.shape)
print(y_test.shape)
4
plt.plot(x_test.squeeze(), y_pred, color='green', linewidth = 3) # plot␣
↪regression line of best fit
```

```
(7133, 1)
(7133,)
```

[262]:
```
[<matplotlib.lines.Line2D at 0x7f11ac188520>]
```

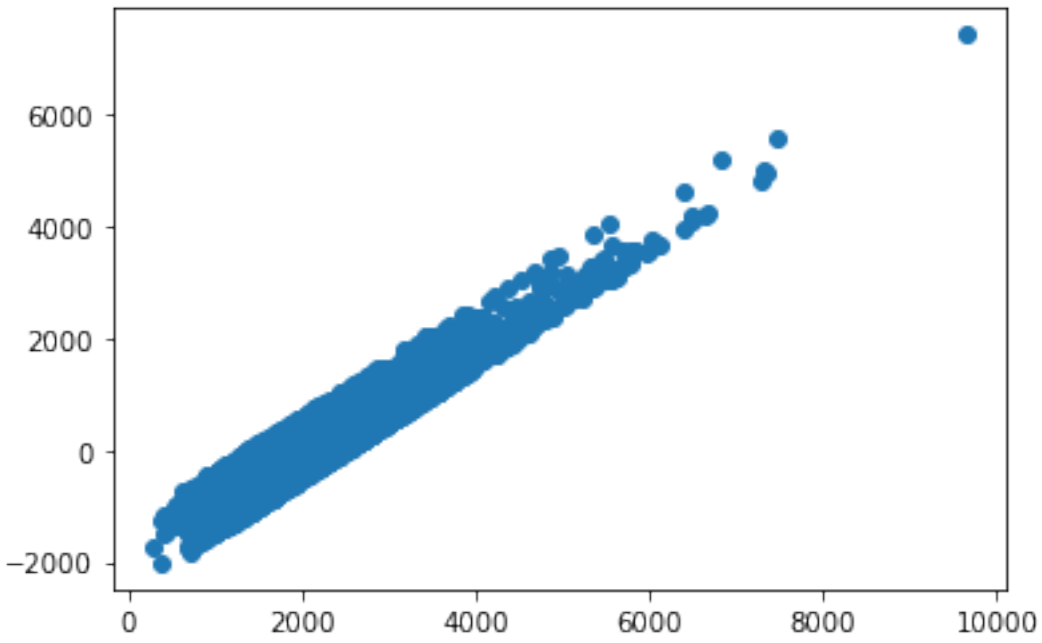


[264]:
```
residuals = y_test - y_pred # residuals are error between predicted and actual
residuals
```

[264]:
```
9857     -803.020214
16196     766.532317
3024     -543.350952
13550    1206.726869
5613     -823.701145
             …
19257     326.726869
3661     -924.187524
3570      676.162669
1445     -693.954062
8288     -364.887909
Name: sqft_living, Length: 7133, dtype: float64
```

[266]:
```
plt.scatter(y_test, residuals) # plot residuals
```

[266]:
```
<matplotlib.collections.PathCollection at 0x7f11ab0d1f70>
```



According to the graph, it indicates a positive correlation between residuals and predicted values, suggesting that the linear model may not fully capture the underlying relationship between sqft_living and year built. Ideally, residuals should be randomly scattered around zero with no discernible pattern. The clustering and pattern observed in the residual plot might imply that the model's predictive accuracy is inconsistent across different levels of predicted values.

[269]:
```
#3.2: fit model
x_cols = ['price', 'bedrooms', 'waterfront', 'view', 'condition']

x = df[x_cols]
y = df['sqft_living']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.33,␣
 ↪random_state=371)

m3 = LinearRegression()

m3.fit(x_train, y_train)

print(m3.intercept_)
print(m3.coef_)
```

```
447.9469427109857
[ 1.45111640e-03  3.76308691e+02 -7.42896862e+02  6.80039046e+01
 -1.26463475e+02]
```

[271]:
```
y_pred = m3.predict(x_test) # use test data to get predicted values
y_pred
```

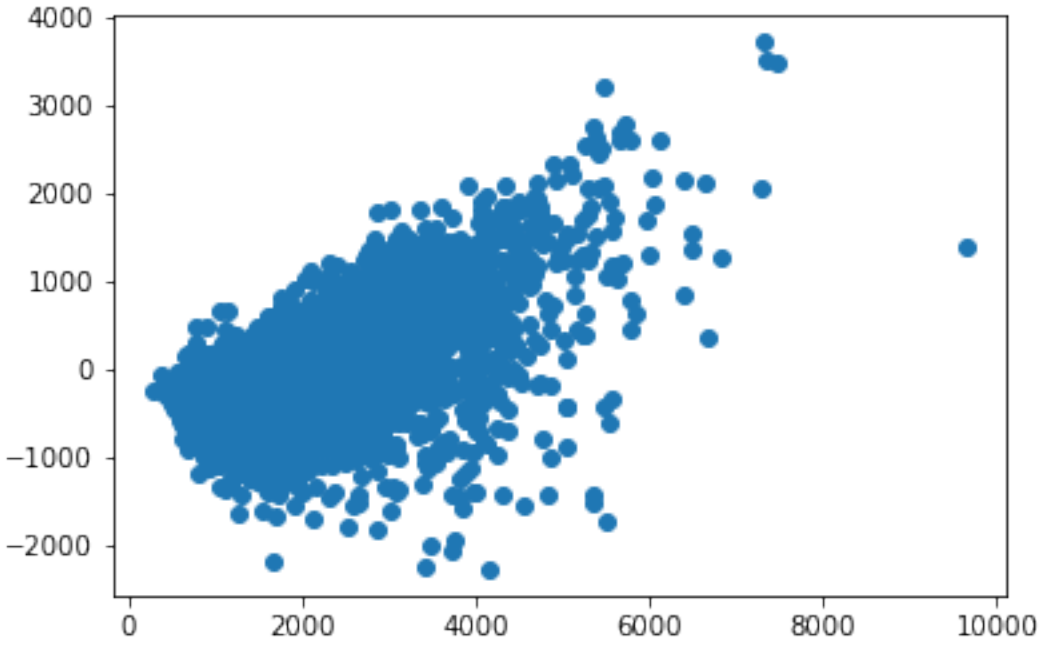
[271]:
```
array([1914.33409489, 2799.12351247, 1854.83832243, …, 1603.79518497,
       1607.20662833, 1764.39956233])
```

[273]:
```
residuals = y_test - y_pred # residuals are error between predicted and actual
residuals
```

[273]:
```
9857     -344.334095
16196     110.876488
3024     -194.838322
13550     939.257064
5613     -222.914845
            …
19257    -359.603546
3661     -106.892952
3570     1026.204815
1445     -407.206628
8288     -714.399562
Name: sqft_living, Length: 7133, dtype: float64
```

[275]:
```
# plot residuals
plt.scatter(y_test, residuals) # plot residuals
```

[275]:
```
<matplotlib.collections.PathCollection at 0x7f11ab0bf880>
```



According to the graph, it indicates a moderate positive correlation between residuals and predicted values, suggesting that the linear model may not fully capture the underlying relationship between sqft_living and other independent variables. Ideally, residuals should be randomly scattered around zero with no discernible pattern. The clustering and pattern observed in the residual plot might imply that the model's predictive accuracy is inconsistent across different levels of predicted values.