Exploration of house data from Arizona state

This dataset has 5000 observations and 16 variables which contains information about houses built in Arizona state in US. This dataset is quite challenging to process for modelling due to lack of metadata such as no description about data and fields in the data. There are some missing value in the dataset. In exploration, I worked on finding different relationship amonge the features.

There are few major parts of this notebook that are outlined as following:

a.) Data input

b.) Dealing with missing values

c.) Exploratory Data Analysis

In [757…
```
# Impport the packages
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import folium
```

Data input

In [758…
```
# Read the data
data = pd.read_csv('raw_house_data.csv')
```

In [759…
```
# # Display the few rows of the data
data.head(5)
```

Out[759…

	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms
0	21530491	5300000.0	85637	-110.378200	31.356362	2154.00	5272.00	1941	13
1	21529082	4200000.0	85646	-111.045371	31.594213	1707.00	10422.36	1997	2
2	3054672	4200000.0	85646	-111.040707	31.594844	1707.00	10482.00	1997	2
3	21919321	4500000.0	85646	-111.035925	31.645878	636.67	8418.58	1930	7
4	21306357	3411450.0	85750	-110.813768	32.285162	3.21	15393.00	1995	4

In [760…
```
# Check the dimensions of the data
data.shape
```

Out[760… (5000, 16)

In [761…
```
# show the concise summary about data and count missing values in the dataframe
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 16 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   MLS               5000 non-null   int64  
 1   sold_price        5000 non-null   float64
 2   zipcode           5000 non-null   int64  
 3   longitude         5000 non-null   float64
 4   latitude          5000 non-null   float64
 5   lot_acres         4990 non-null   float64
 6   taxes             5000 non-null   float64
 7   year_built        5000 non-null   int64  
 8   bedrooms          5000 non-null   int64  
 9   bathrooms         5000 non-null   object 
 10  sqrt_ft           5000 non-null   object 
 11  garage            5000 non-null   object 
 12  kitchen_features  5000 non-null   object 
 13  fireplaces        5000 non-null   object 
 14  floor_covering    5000 non-null   object 
 15  HOA               5000 non-null   object 
dtypes: float64(5), int64(4), object(7)
memory usage: 625.1+ KB
```

In [762…
```
# Descriptive statistical meaures
data.describe()
```

Out[762…

	MLS	sold_price	zipcode	longitude	latitude	lot_acres	ta
count	5.000000e+03	5.000000e+03	5000.000000	5000.000000	5000.000000	4990.000000	5.000000e·
mean	2.127070e+07	7.746262e+05	85723.025600	-110.912107	32.308512	4.661317	9.402828e·
std	2.398508e+06	3.185556e+05	38.061712	0.120629	0.178028	51.685230	1.729385e·
min	3.042851e+06	1.690000e+05	85118.000000	-112.520168	31.356362	0.000000	0.000000e·
25%	2.140718e+07	5.850000e+05	85718.000000	-110.979260	32.277484	0.580000	4.803605e·
50%	2.161469e+07	6.750000e+05	85737.000000	-110.923420	32.318517	0.990000	6.223760e·
75%	2.180480e+07	8.350000e+05	85749.000000	-110.859078	32.394334	1.757500	8.082830e·
max	2.192856e+07	5.300000e+06	86323.000000	-109.454637	34.927884	2154.000000	1.221508e·

In [763…
```
# check the duplicate values
data.duplicated().sum()
```

Out[763… 0

Dealing with Missing values

In [764…
```
# check the missing value in each column
data.isna().sum()
```

Out[764…
```
MLS                 0
sold_price          0
zipcode             0
longitude           0
latitude            0
lot_acres          10
taxes               0
year_built          0
bedrooms            0
bathrooms           0
sqrt_ft             0
garage              0
kitchen_features    0
fireplaces          0
floor_covering      0
HOA                 0
dtype: int64
```

In [765…
```
# Replace the nan values from the dataframe
data1 = data.replace(['None'], np.nan)
```

In [766…
```
# Check few rows of the data
data1.head(3)
```

Out[766…

	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms
0	21530491	5300000.0	85637	-110.378200	31.356362	2154.0	5272.00	1941	13
1	21529082	4200000.0	85646	-111.045371	31.594213	1707.0	10422.36	1997	2
2	3054672	4200000.0	85646	-111.040707	31.594844	1707.0	10482.00	1997	2

In [767…
```
# fill the missing value in column HOA with 0
data1['HOA'].fillna(0, inplace = True)
```

In [768…
```
# check the missing values in each column
data1.isna().sum()
```

Out[768…
```
MLS                 0
sold_price          0
zipcode             0
longitude           0
latitude            0
lot_acres          10
taxes               0
```

```
year_built           0
bedrooms             0
bathrooms            6
sqrt_ft             56
garage               7
kitchen_features    33
fireplaces           0
floor_covering       1
HOA                  0
dtype: int64
```

In [769…

```
# Delete all missing values from the dataframe
data2 = data1.dropna()
data2 = data2.drop(data2[data2['year_built']==0].index)
data2.head()
```

Out[769…

	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms
0	21530491	5300000.0	85637	-110.378200	31.356362	2154.00	5272.00	1941	13
1	21529082	4200000.0	85646	-111.045371	31.594213	1707.00	10422.36	1997	2
3	21919321	4500000.0	85646	-111.035925	31.645878	636.67	8418.58	1930	7
4	21306357	3411450.0	85750	-110.813768	32.285162	3.21	15393.00	1995	4
5	21528016	3250000.0	85718	-110.910593	32.339090	1.67	27802.84	1999	3

In [770…

```
# To check the missing value again
data2.isna().sum()
```

Out[770…

```
MLS                 0
sold_price          0
zipcode             0
longitude           0
latitude            0
lot_acres           0
taxes               0
year_built          0
bedrooms            0
bathrooms           0
sqrt_ft             0
garage              0
kitchen_features    0
fireplaces          0
floor_covering      0
HOA                 0
dtype: int64
```

In [771…
```
# convert the object type in to float type
data2['bathrooms'] = pd.to_numeric(data2['bathrooms'])
data2['garage'] = pd.to_numeric(data2['garage'])
data2['sqrt_ft'] = pd.to_numeric(data2['sqrt_ft'])
```

In [772…
```
# Calculate the sold price per square foot
data2["sold_price_per_sqrt_ft"] = data2["sold_price"]/data2["sqrt_ft"]

#data2.head()
```

Exploratory Data Analysis

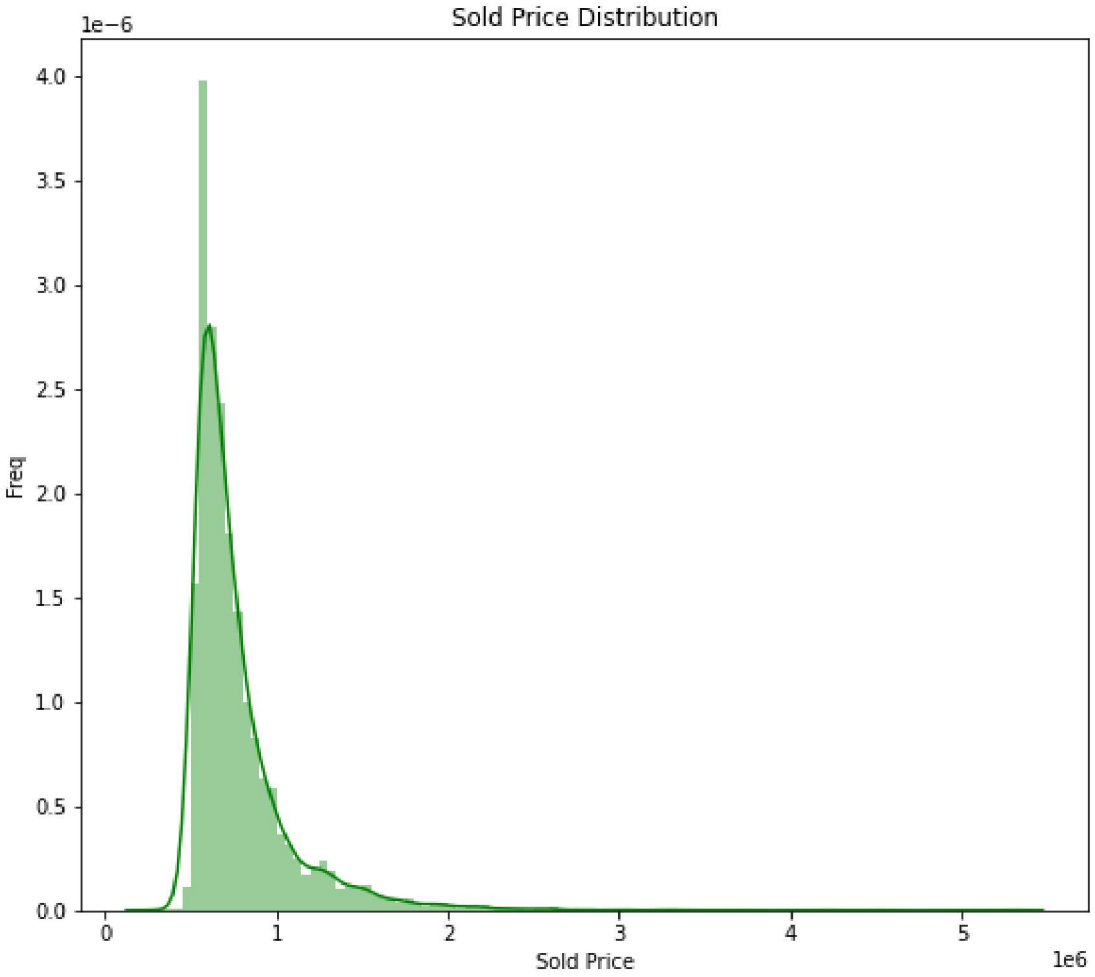
In [773…
```
data2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4903 entries, 0 to 4998
Data columns (total 17 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   MLS                    4903 non-null   int64  
 1   sold_price             4903 non-null   float64
 2   zipcode                4903 non-null   int64  
 3   longitude              4903 non-null   float64
 4   latitude               4903 non-null   float64
 5   lot_acres              4903 non-null   float64
 6   taxes                  4903 non-null   float64
 7   year_built             4903 non-null   int64  
 8   bedrooms               4903 non-null   int64  
 9   bathrooms              4903 non-null   float64
 10  sqrt_ft                4903 non-null   float64
 11  garage                 4903 non-null   float64
 12  kitchen_features       4903 non-null   object 
 13  fireplaces             4903 non-null   object 
 14  floor_covering         4903 non-null   object 
 15  HOA                    4903 non-null   object 
 16  sold_price_per_sqrt_ft  4903 non-null   float64
dtypes: float64(9), int64(4), object(4)
memory usage: 689.5+ KB
```

In [774…
```
# Disply the distribution of sold price column
plt.figure(figsize=(9, 8))
sns.distplot(data2['sold_price'], color='g', bins=100, hist_kws={'alpha': 0.4});
plt.title('Sold Price Distribution')
plt.xlabel('Sold Price')
plt.ylabel('Freq')
```

```
C:\Users\User\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarni
ng:

`distplot` is a deprecated function and will be removed in a future version. Please
adapt your code to use either `displot` (a figure-level function with similar flexib
ility) or `histplot` (an axes-level function for histograms).
```

Out[774…
```
Text(0, 0.5, 'Freq')
```

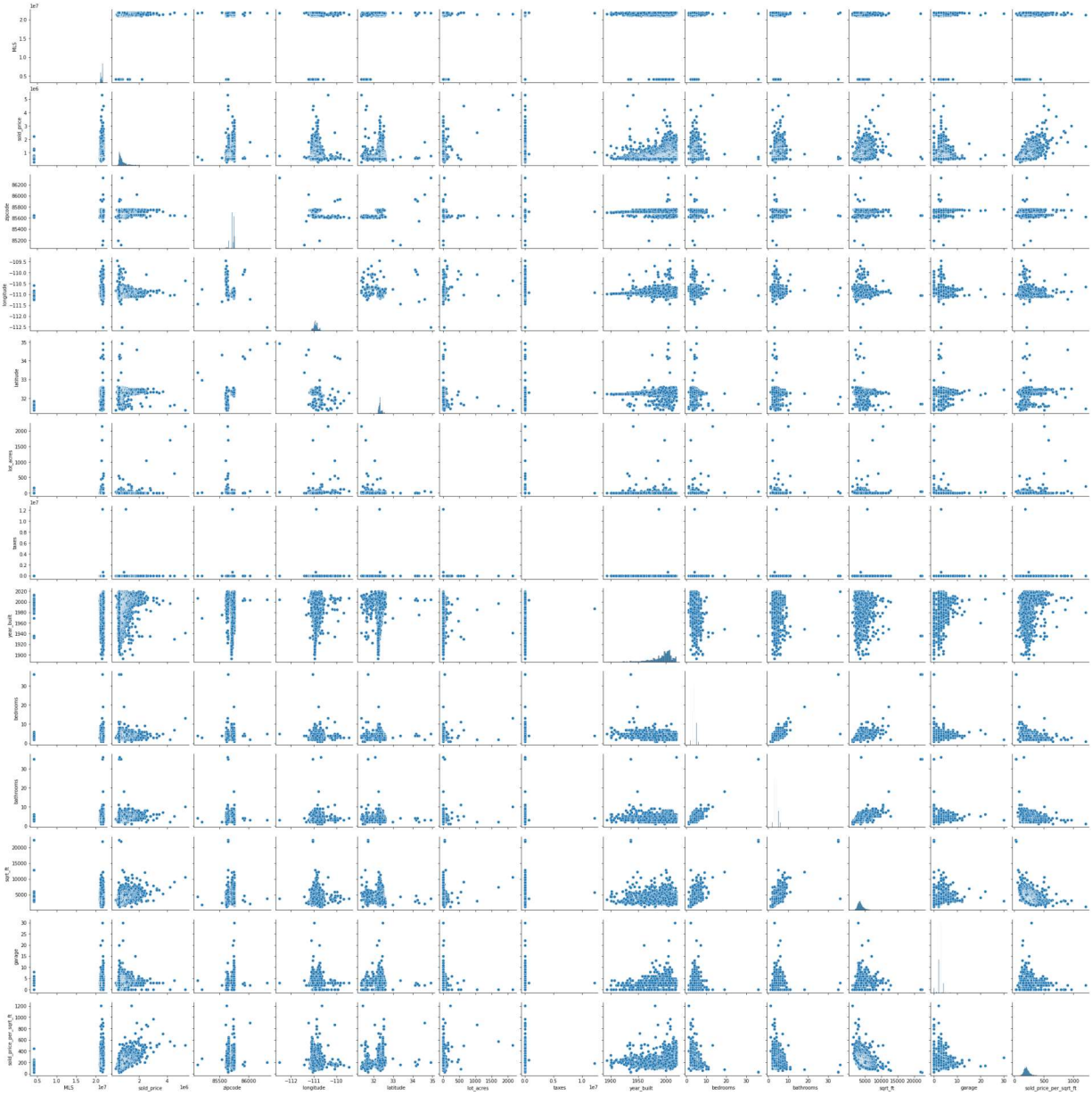


In [775…

```
# Check the relationship of the features
plt.figure(figsize=(16, 12))
sns.pairplot(data2)
plt.xticks(fontsize=14, rotation=90)
```

Out[775…

```
(array([-500.,    0.,  500., 1000., 1500.]),
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
<Figure size 1152x864 with 0 Axes>
```



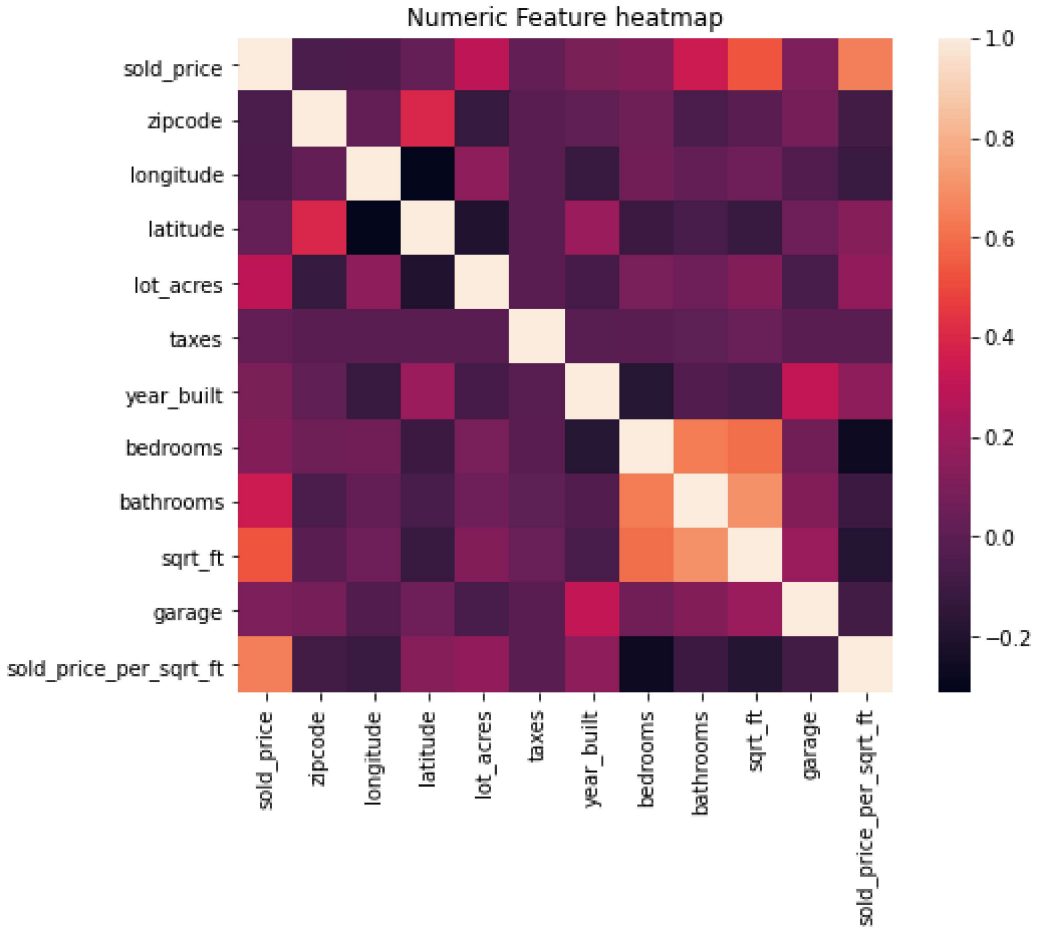
With this information we can see that the prices are skewed right and some outliers lies above ~3,000,000.

In [776…

```
# To check the correlation amonge the variables
fig,ax = plt.subplots(figsize=(8,6))
correlation = data2.select_dtypes(include=['float64','int64']).iloc[:,1:].corr()
sns.heatmap(correlation,ax=ax,vmax=1,square=True)
plt.title('Numeric Feature heatmap')
```

Out[776…

```
Text(0.5, 1.0, 'Numeric Feature heatmap')
```



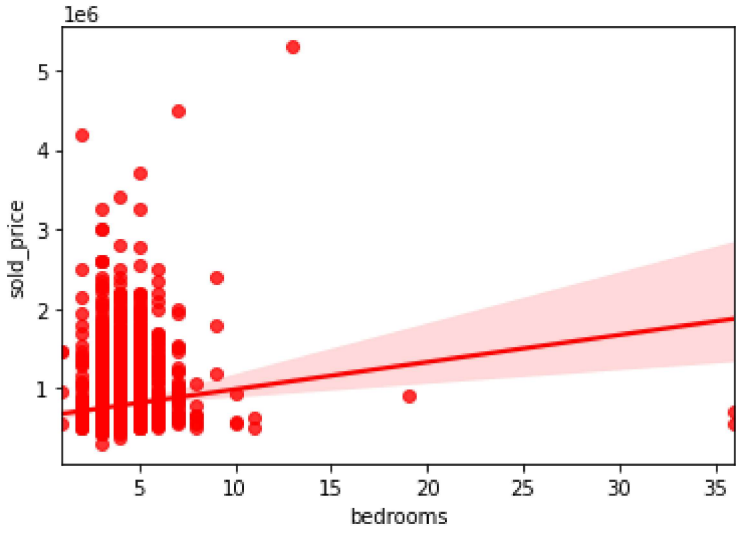
In [777…

```
corr_dict = correlation['sold_price'].to_dict()
del correlation['sold_price']
## descending
## correlation with sold price
for key,val in sorted(corr_dict.items(),key=lambda x:-abs(x[1])):
    print('{0} \t : {1}' .format(key,val))
```

```
sold_price       : 1.0
sold_price_per_sqrt_ft   : 0.6467866269604817
sqrt_ft          : 0.533660017650321
bathrooms        : 0.34869426019063465
lot_acres        : 0.2988167370502996
bedrooms         : 0.12315355094202046
garage   : 0.10392860678544359
year_built       : 0.10309148858595892
zipcode          : -0.051999571838080544
longitude        : -0.041089567244066275
latitude         : 0.031388410666668476
taxes    : 0.023605188873612976
```

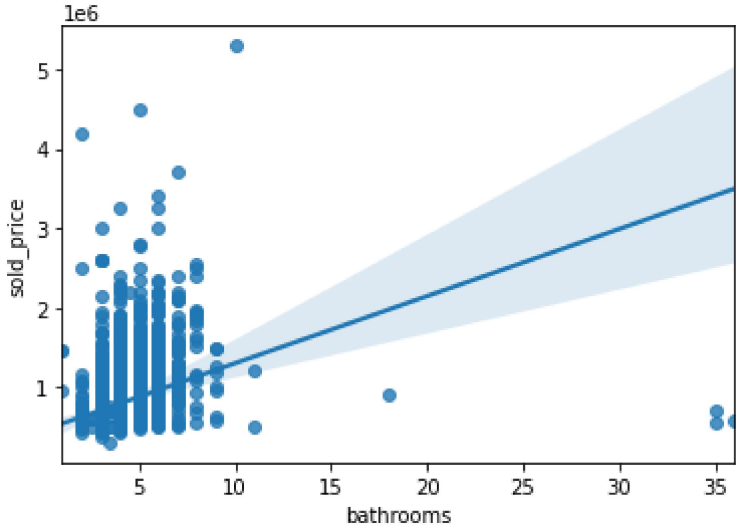
In [778…

```
# plot the scatter plot between sold_price and bedrooms to find the relationship
sns.regplot(x = "bedrooms",
            y = "sold_price",
            data = data2, color = 'red')
# show the plot
plt.show()
```



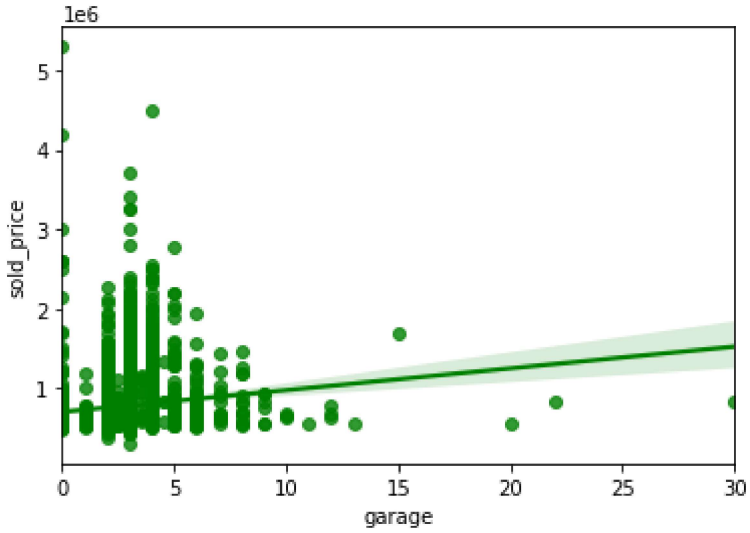
In [779…

```
# plot the scatter plot between sold_price and bathrooms
sns.regplot(x = "bathrooms",
            y = "sold_price",
            data = data2)
# show the plot
plt.show()
```



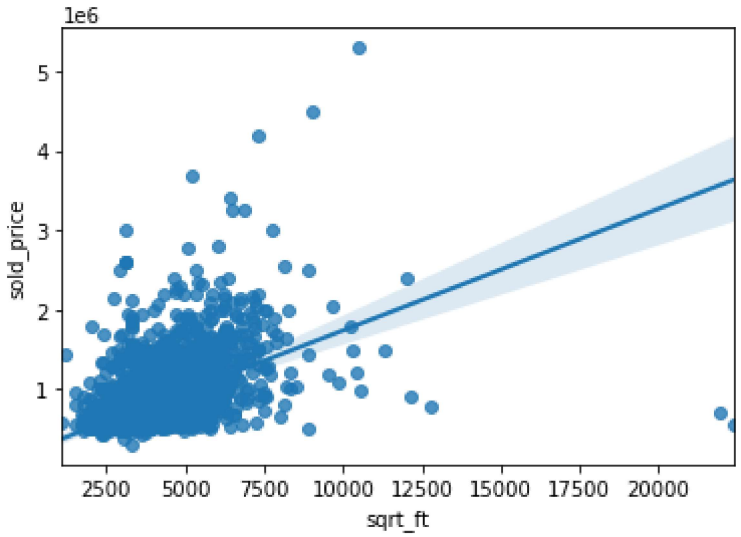
In [780…

```
sns.regplot(x = "garage",
            y = "sold_price",
            data = data2, color = 'green')
# show the plot
plt.show()
```



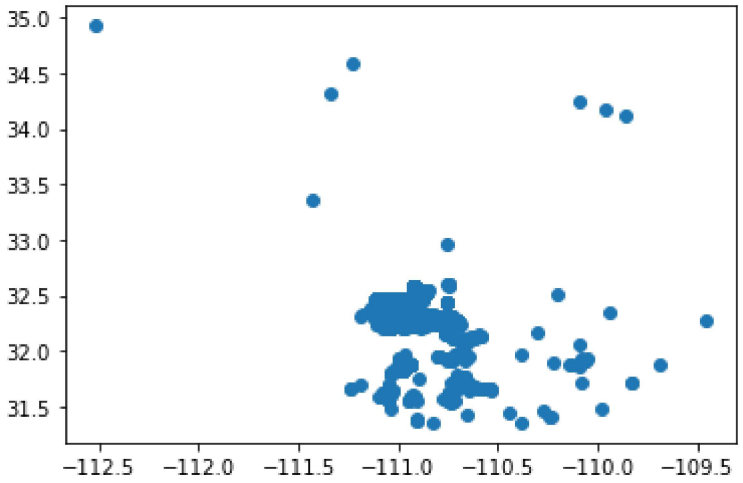
In [781…

```
# plot the scatter plot between sold_price and sqrt_ft
sns.regplot(x = "sqrt_ft",
            y = "sold_price",
            data = data2)
# show the plot
plt.show()
```



In [782…

```
# scatter plot between logitude and latitude
plt.scatter(x=data2['longitude'], y=data2['latitude'])
plt.show()
```



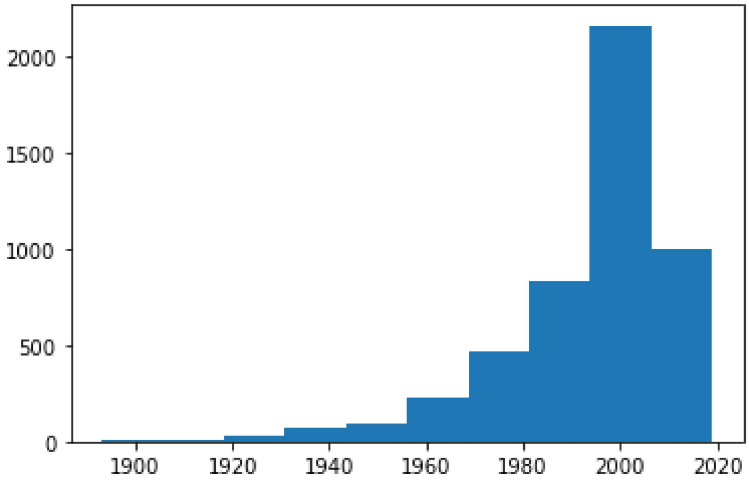
In [783…
```
# create the cloropleth map
map_world = folium.Map(location = [31, -110], tiles = 'stamenterrain', zoom_start =
#  add Locations to map
for lat, lng, label in zip(data2.latitude, data2.longitude, data2.sold_price):
    folium.CircleMarker(
        [lat, lng],
        columns= ['sold_price', 'bedrooms'],
        radius=2,
        popup=label,
        fill=True,
        color='Red',
        fill_color='PuBu',
        fill_opacity=0.6
        ).add_to(map_world)
```

In [784…
```
map_world
```

Out[784… Make this Notebook Trusted to load map: File -> Trust Notebook

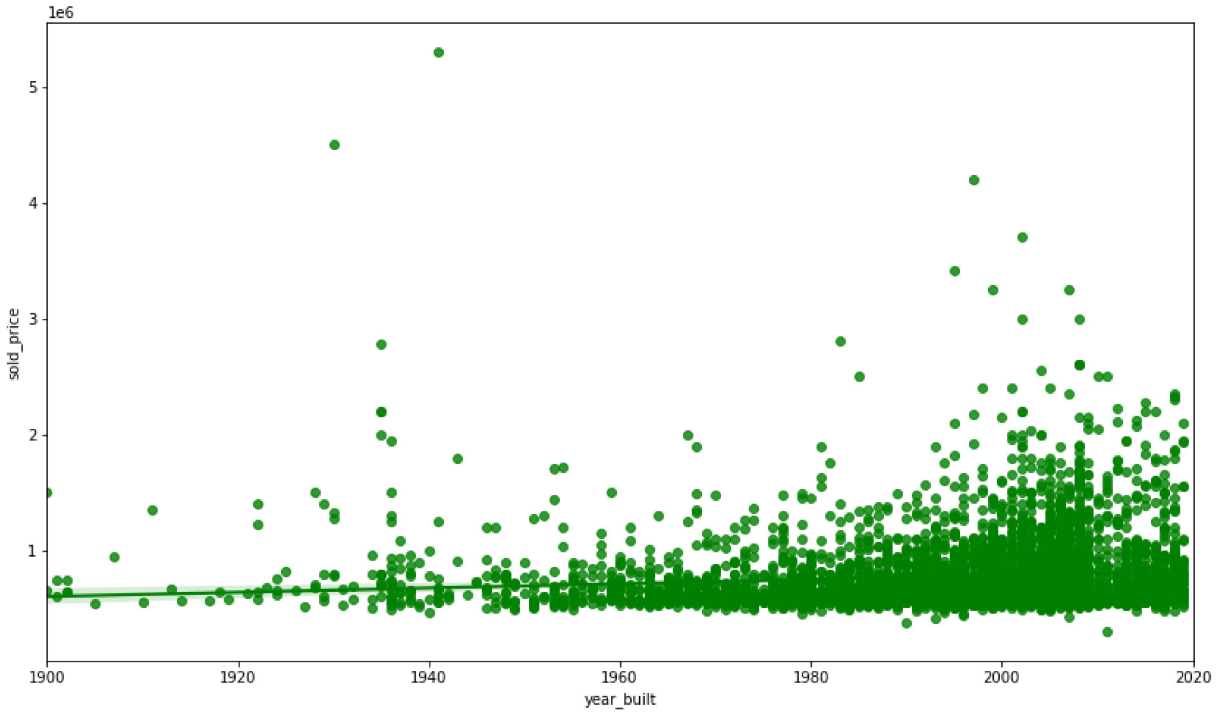
In [785…
```
# Distribution of zipcode
df= data2['year_built']
plt.hist(x=df)
```

Out[785…
```
(array([   9.,    7.,   25.,   75.,   94.,  223.,  468.,  834., 2165.,
        1003.]),
 array([1893. , 1905.6, 1918.2, 1930.8, 1943.4, 1956. , 1968.6, 1981.2,
        1993.8, 2006.4, 2019. ]),
 <BarContainer object of 10 artists>)
```



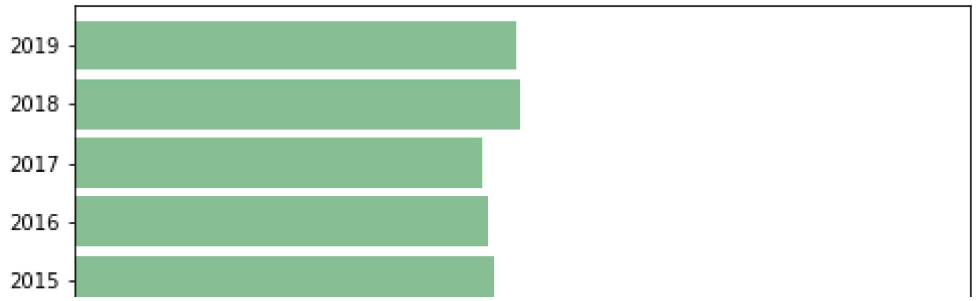
In [786…

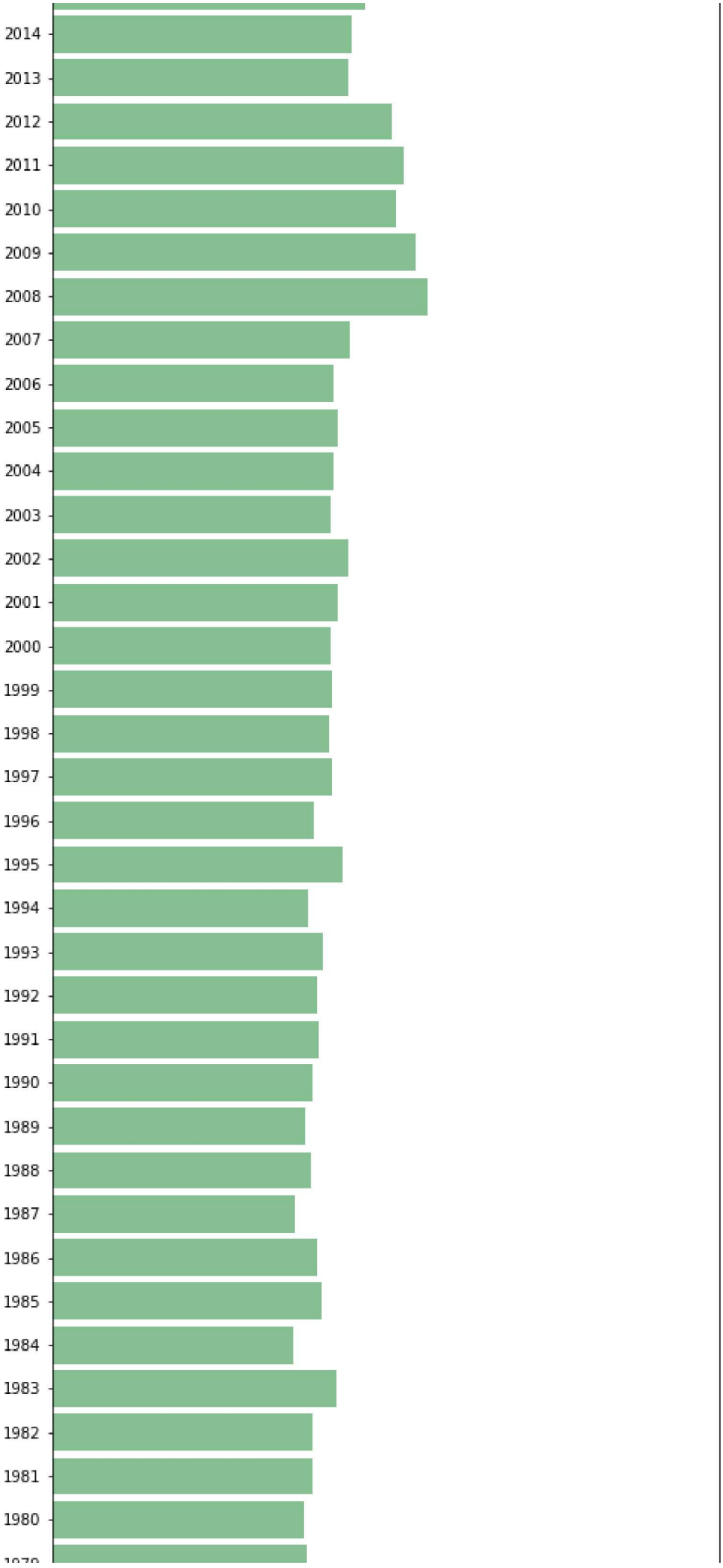
```
# correlation between sold price and year_built
plt.figure(figsize=[14,8])
plt.xlim([1900, 2020])
sns.regplot(x = "year_built",
            y = "sold_price",
            data = data2, color = 'green')

# show the plot
plt.show()
```

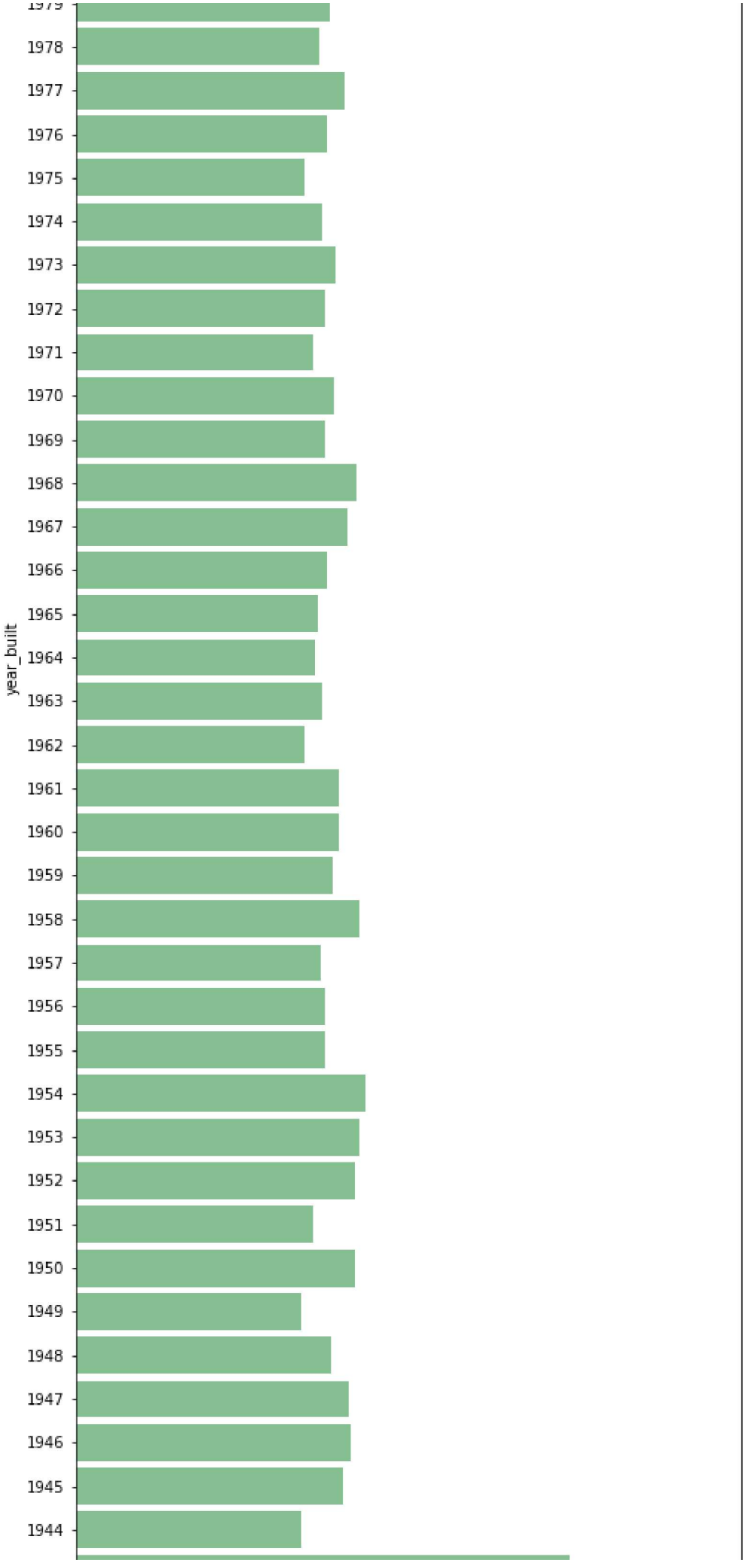


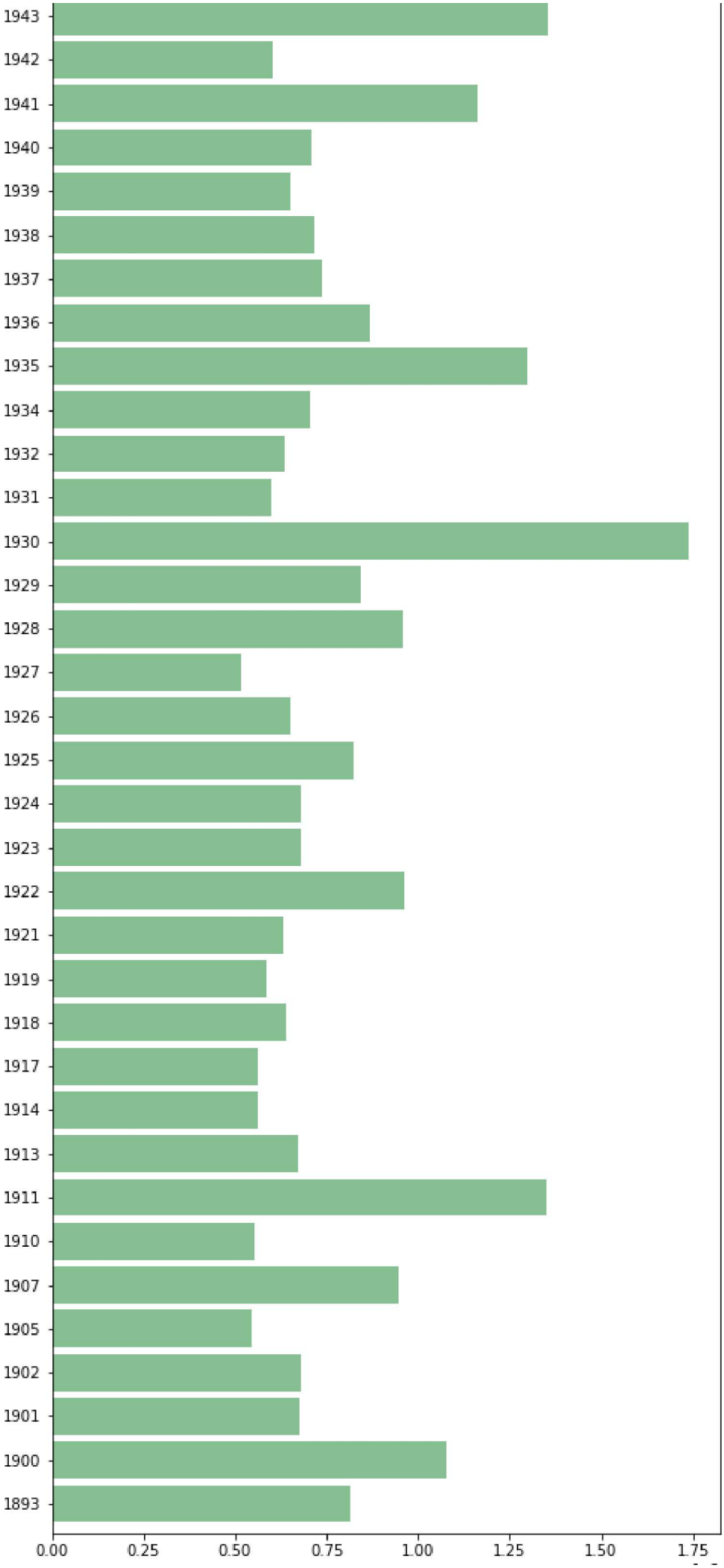
In [787…

```
# bar chat between year)built and sold_price
x = data2.groupby('year_built')['sold_price'].mean()
ax = x.plot(kind='barh', figsize=(8, 60), color='#86bf91', zorder=2, width=0.85)
```




2014
2013
2012
2011
2010
2009
2008
2007
2006
2005
2004
2003
2002
2001
2000
1999
1998
1997
1996
1995
1994
1993
1992
1991
1990
1989
1988
1987
1986
1985
1984
1983
1982
1981
1980

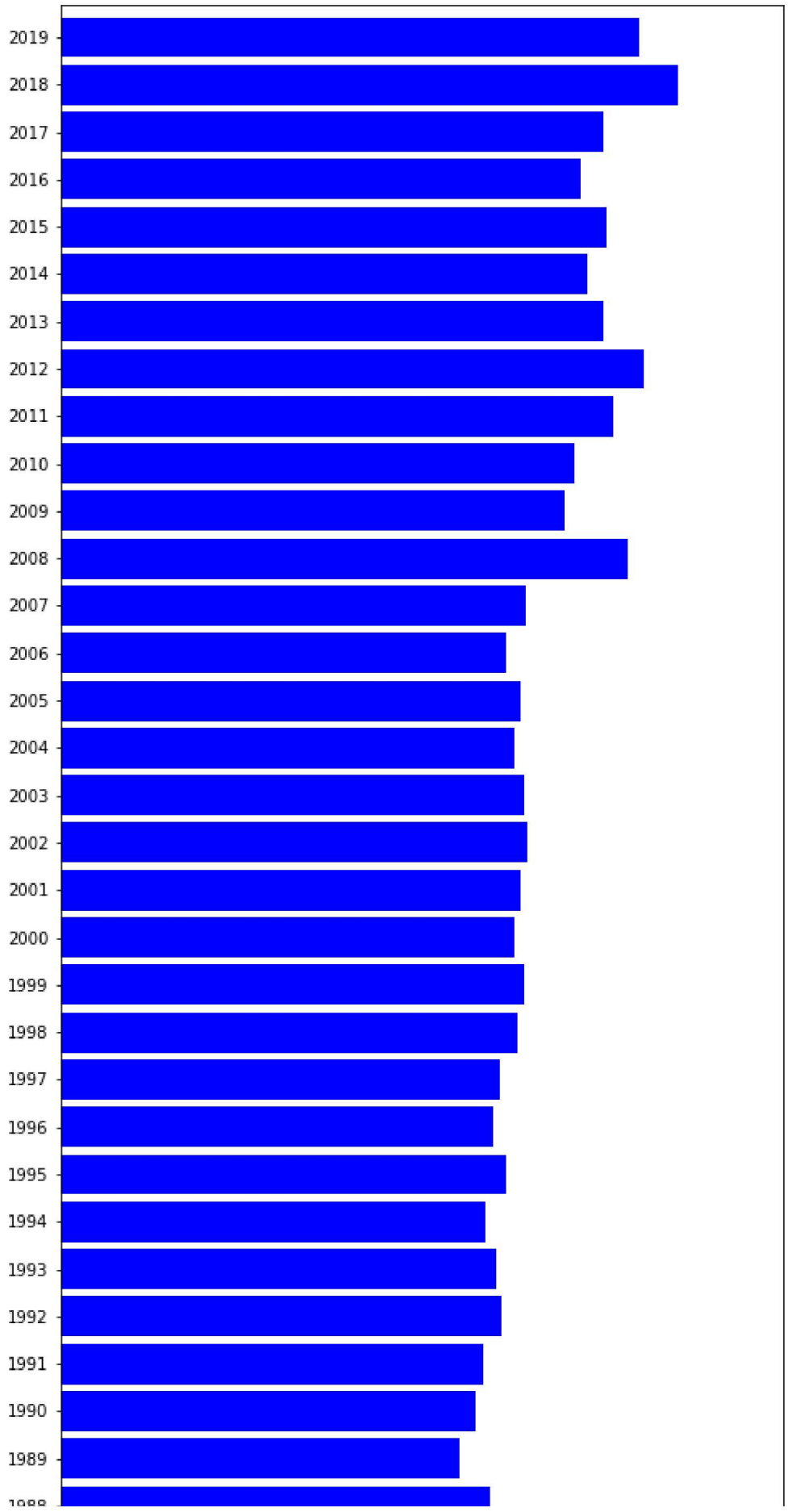

year_built
1978
1977
1976
1975
1974
1973
1972
1971
1970
1969
1968
1967
1966
1965
1964
1963
1962
1961
1960
1959
1958
1957
1956
1955
1954
1953
1952
1951
1950
1949
1948
1947
1946
1945
1944

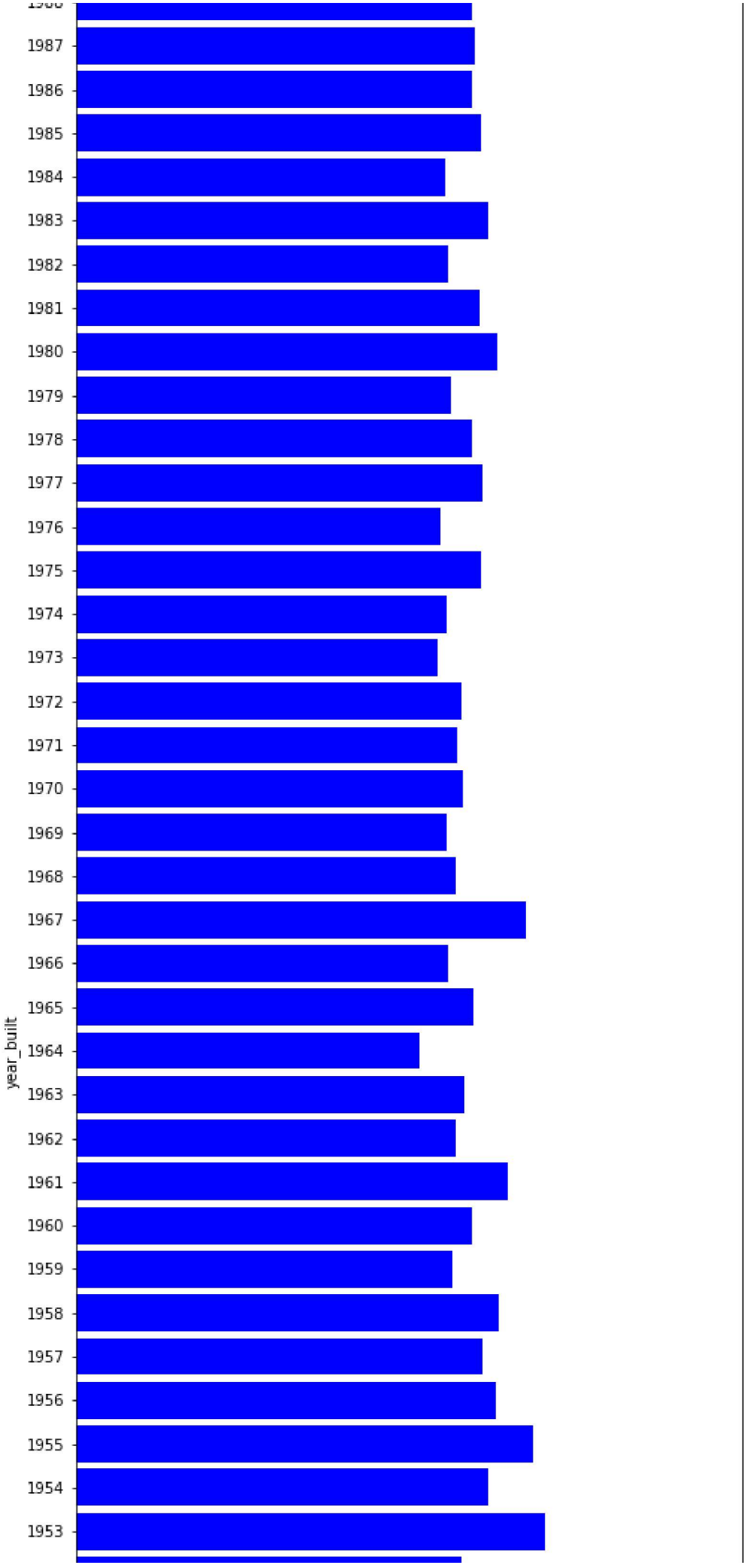

1943
1942
1941
1940
1939
1938
1937
1936
1935
1934
1932
1931
1930
1929
1928
1927
1926
1925
1924
1923
1922
1921
1919
1918
1917
1914
1913
1911
1910
1907
1905
1902
1901
1900
1893
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75

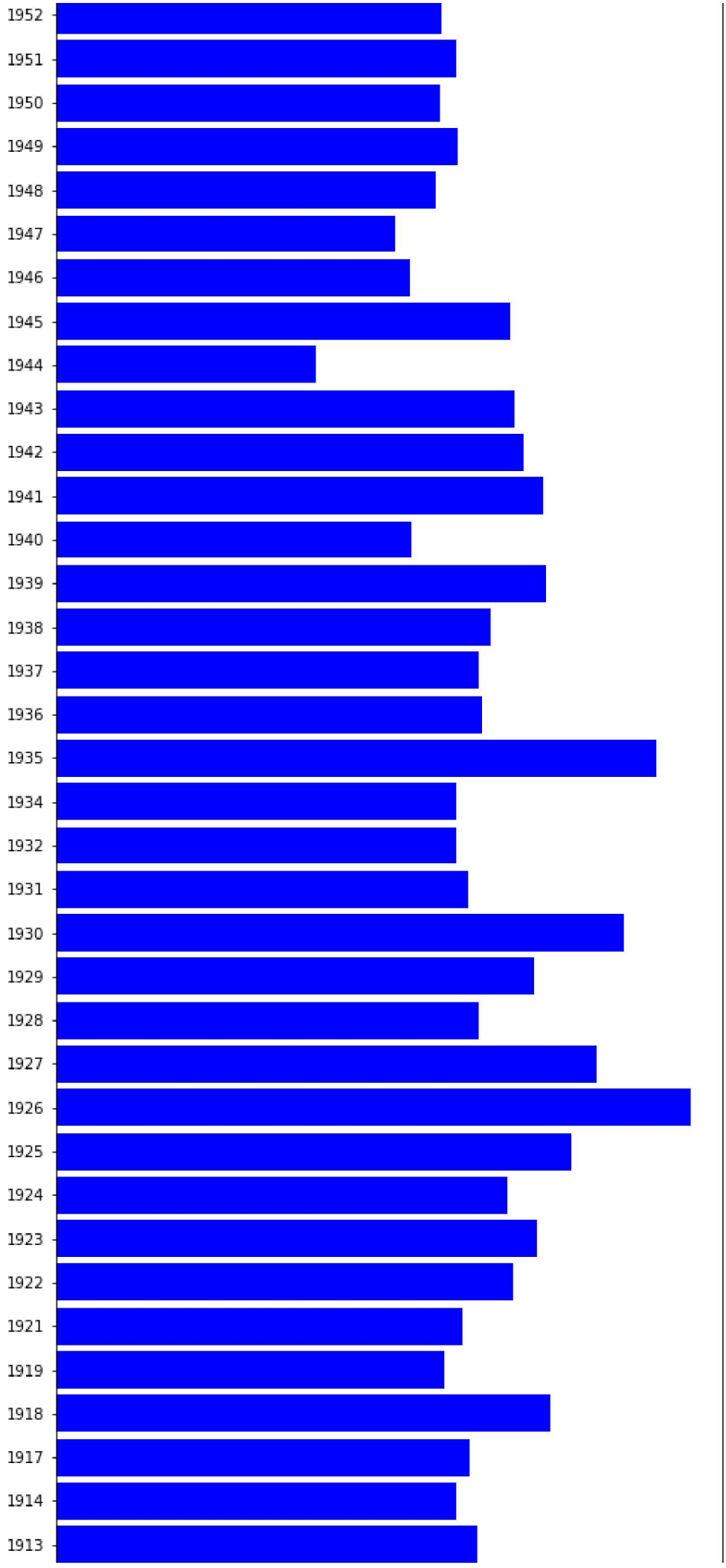
1e6

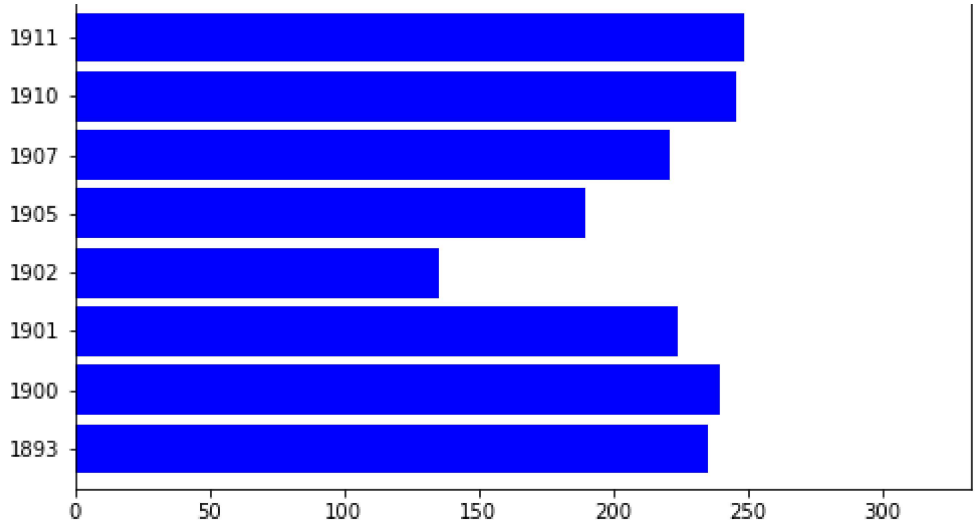
In [795…

```
# bar chat between year_built and sold_price_sqrt_per_ft
x = data2.groupby('year_built')['sold_price_per_sqrt_ft'].mean()
ax = x.plot(kind='barh', figsize=(8, 60), color='blue', zorder=2, width=0.85)
```



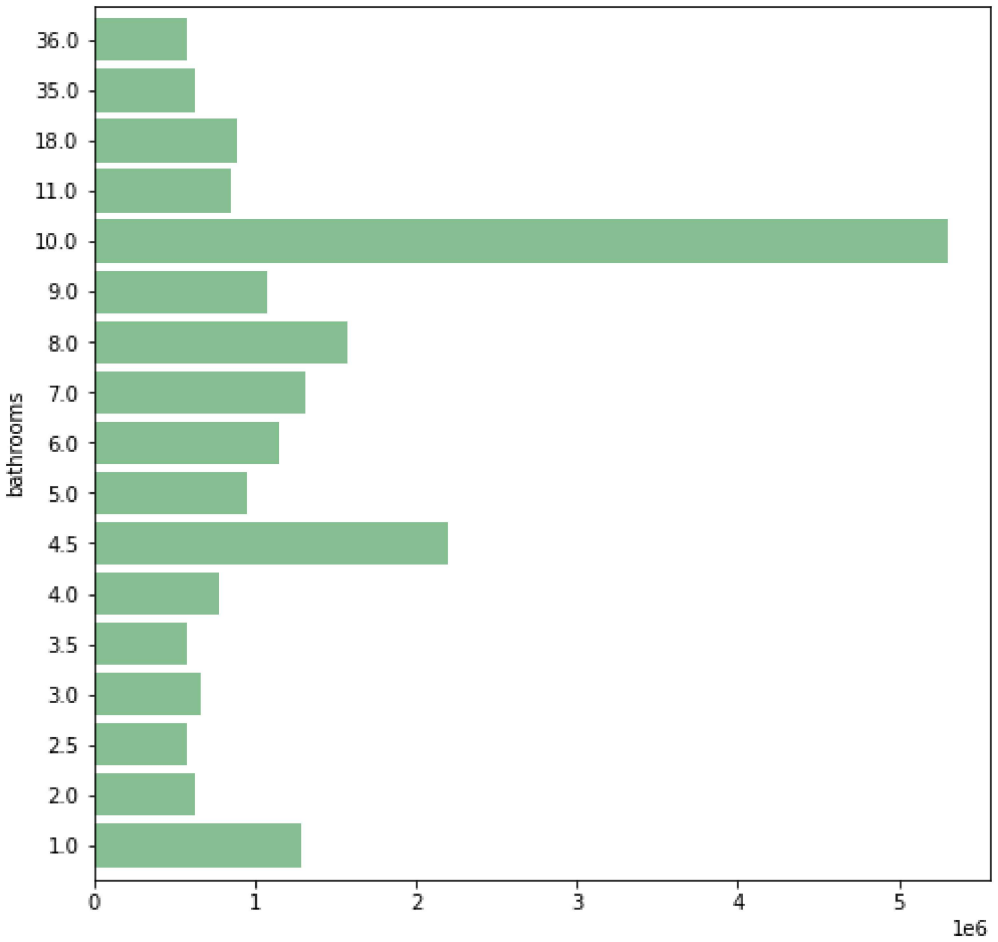

1987
1986
1985
1984
1983
1982
1981
1980
1979
1978
1977
1976
1975
1974
1973
1972
1971
1970
1969
1968
1967
1966
1965
1964
1963
1962
1961
1960
1959
1958
1957
1956
1955
1954
1953
year_built


1952
1951
1950
1949
1948
1947
1946
1945
1944
1943
1942
1941
1940
1939
1938
1937
1936
1935
1934
1932
1931
1930
1929
1928
1927
1926
1925
1924
1923
1922
1921
1919
1918
1917
1914
1913



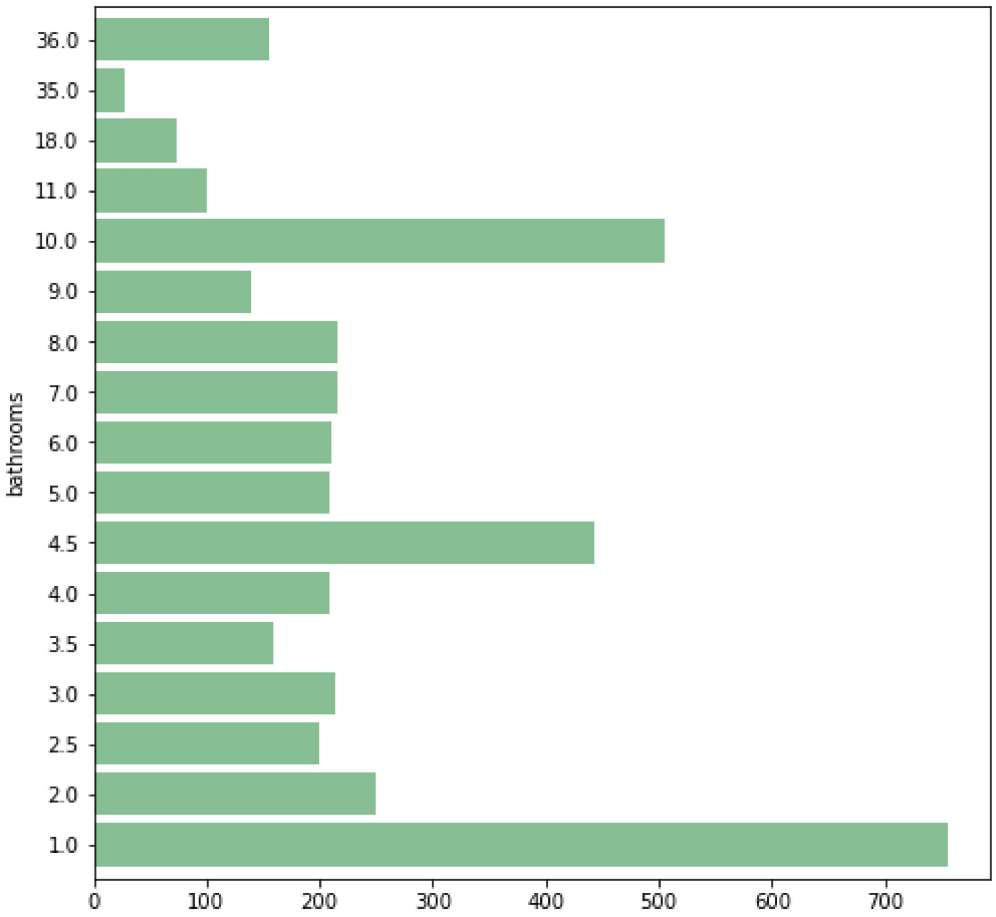
In [789…

```
# # bar chat between bathroom and sold_price
x = data2.groupby('bathrooms')['sold_price'].mean()
ax = x.plot(kind='barh', figsize=(8, 8), color='#86bf91', zorder=2, width=0.85)
```



In [790…

```
#  bar chat between bathroom and sold_price_per_sqrt_ft
x = data2.groupby('bathrooms')['sold_price_per_sqrt_ft'].mean()
ax = x.plot(kind='barh', figsize=(8, 8), color='#86bf91', zorder=2, width=0.85)
```



In [792…

```
# count the number of features in kitchen
data2['comma_count'] = data2.kitchen_features.str.count(',')
```

In [793…

```
data2.head()
```

Out[793…

	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms
0	21530491	5300000.0	85637	-110.378200	31.356362	2154.00	5272.00	1941	13
1	21529082	4200000.0	85646	-111.045371	31.594213	1707.00	10422.36	1997	2
3	21919321	4500000.0	85646	-111.035925	31.645878	636.67	8418.58	1930	7
4	21306357	3411450.0	85750	-110.813768	32.285162	3.21	15393.00	1995	4
5	21528016	3250000.0	85718	-110.910593	32.339090	1.67	27802.84	1999	3

In [794…

```
x = data2.groupby('comma_count')['sold_price'].max()
ax = x.plot(kind='barh', figsize=(8, 8), color='r', zorder=2, width=0.85)
```