**Import software libraries**

In [1]:

**import** sys *# Read system parameters*

**import** os *# Interact with the operating system*

**import** numpy **as** np *# Work with multi-dimensional arrays*

**import** pandas **as** pd *# Manipulate and analyze data*

**import** matplotlib *# Create 2D charts*

**import** scipy **as** sp *# Perform scientific computing and*

**import** sklearn *# Perform data mining and analysis*

**import** seaborn **as** sb *# Perform data visualization*

*# Summarize software libraries used*

print('Libraries used in this project:')

print('- NumPy {}'**.**format(np**.**\_\_version\_\_))

print('- Pandas {}'**.**format(pd**.**\_\_version\_\_))

print('- Matplotlib {}'**.**format(matplotlib**.**\_\_version\_\_))

print('- SciPy {}'**.**format(sp**.**\_\_version\_\_))

print('- Scikit-learn {}'**.**format(sklearn**.**\_\_version\_\_))

print('- Python {}\n'**.**format(sys**.**version))

Libraries used in this project:

- NumPy 1.16.2

- Pandas 0.24.2

- Matplotlib 3.0.3

- SciPy 1.2.1

- Scikit-learn 0.20.3

- Python 3.7.6 | packaged by conda-forge | (default, Mar 23 2020, 23:03:20)

[GCC 7.3.0]

**Load the dataset**

In [2]:

PROJECT\_ROOT\_DIR **=** '.'

DATA\_PATH **=** os**.**path**.**join(PROJECT\_ROOT\_DIR, 'housing\_data')

print('Data files in this project:', os**.**listdir(DATA\_PATH) )

*# Read the raw dataset*

data\_raw\_file **=** os**.**path**.**join( DATA\_PATH, 'kc\_house\_data.csv' )

data\_raw **=** pd**.**read\_csv( data\_raw\_file )

print('Loaded {} records from {}.\n'**.**format(len(data\_raw), data\_raw\_file))

Data files in this project: ['kc\_house\_data.csv']

Loaded 21613 records from ./housing\_data/kc\_house\_data.csv.

**Get acquainted with the dataset**

In [3]:

print(data\_raw**.**info()) *# View features and data types*

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 21613 entries, 0 to 21612

Data columns (total 21 columns):

id 21613 non-null int64

date 21613 non-null object

price 21613 non-null float64

bedrooms 21613 non-null int64

bathrooms 21613 non-null float64

sqft\_living 21613 non-null int64

sqft\_lot 21613 non-null int64

floors 21613 non-null float64

waterfront 21613 non-null int64

view 21613 non-null int64

condition 21613 non-null int64

grade 21613 non-null int64

sqft\_above 21613 non-null int64

sqft\_basement 21613 non-null int64

yr\_built 21613 non-null int64

yr\_renovated 21613 non-null int64

zipcode 21613 non-null int64

lat 21613 non-null float64

long 21613 non-null float64

sqft\_living15 21613 non-null int64

sqft\_lot15 21613 non-null int64

dtypes: float64(5), int64(15), object(1)

memory usage: 3.5+ MB

None

**Show example records**

In [4]:

*# View first ten records*

print(data\_raw**.**head(10))

id date price bedrooms bathrooms sqft\_living \

0 7129300520 20141013T000000 221900.0 3 1.00 1180

1 6414100192 20141209T000000 538000.0 3 2.25 2570

2 5631500400 20150225T000000 180000.0 2 1.00 770

3 2487200875 20141209T000000 604000.0 4 3.00 1960

4 1954400510 20150218T000000 510000.0 3 2.00 1680

5 7237550310 20140512T000000 1225000.0 4 4.50 5420

6 1321400060 20140627T000000 257500.0 3 2.25 1715

7 2008000270 20150115T000000 291850.0 3 1.50 1060

8 2414600126 20150415T000000 229500.0 3 1.00 1780

9 3793500160 20150312T000000 323000.0 3 2.50 1890

sqft\_lot floors waterfront view ... grade sqft\_above sqft\_basement \

0 5650 1.0 0 0 ... 7 1180 0

1 7242 2.0 0 0 ... 7 2170 400

2 10000 1.0 0 0 ... 6 770 0

3 5000 1.0 0 0 ... 7 1050 910

4 8080 1.0 0 0 ... 8 1680 0

5 101930 1.0 0 0 ... 11 3890 1530

6 6819 2.0 0 0 ... 7 1715 0

7 9711 1.0 0 0 ... 7 1060 0

8 7470 1.0 0 0 ... 7 1050 730

9 6560 2.0 0 0 ... 7 1890 0

yr\_built yr\_renovated zipcode lat long sqft\_living15 \

0 1955 0 98178 47.5112 -122.257 1340

1 1951 1991 98125 47.7210 -122.319 1690

2 1933 0 98028 47.7379 -122.233 2720

3 1965 0 98136 47.5208 -122.393 1360

4 1987 0 98074 47.6168 -122.045 1800

5 2001 0 98053 47.6561 -122.005 4760

6 1995 0 98003 47.3097 -122.327 2238

7 1963 0 98198 47.4095 -122.315 1650

8 1960 0 98146 47.5123 -122.337 1780

9 2003 0 98038 47.3684 -122.031 2390

sqft\_lot15

0 5650

1 7639

2 8062

3 5000

4 7503

5 101930

6 6819

7 9711

8 8113

9 7570

[10 rows x 21 columns]

**Examine descriptive statistics**

In [5]:

**with** pd**.**option\_context('float\_format', '{:.2f}'**.**format):

print( data\_raw**.**describe() )

id price bedrooms bathrooms sqft\_living sqft\_lot \

count 21613.00 21613.00 21613.00 21613.00 21613.00 21613.00

mean 4580301520.86 540088.14 3.37 2.11 2079.90 15106.97

std 2876565571.31 367127.20 0.93 0.77 918.44 41420.51

min 1000102.00 75000.00 0.00 0.00 290.00 520.00

25% 2123049194.00 321950.00 3.00 1.75 1427.00 5040.00

50% 3904930410.00 450000.00 3.00 2.25 1910.00 7618.00

75% 7308900445.00 645000.00 4.00 2.50 2550.00 10688.00

max 9900000190.00 7700000.00 33.00 8.00 13540.00 1651359.00

floors waterfront view condition grade sqft\_above \

count 21613.00 21613.00 21613.00 21613.00 21613.00 21613.00

mean 1.49 0.01 0.23 3.41 7.66 1788.39

std 0.54 0.09 0.77 0.65 1.18 828.09

min 1.00 0.00 0.00 1.00 1.00 290.00

25% 1.00 0.00 0.00 3.00 7.00 1190.00

50% 1.50 0.00 0.00 3.00 7.00 1560.00

75% 2.00 0.00 0.00 4.00 8.00 2210.00

max 3.50 1.00 4.00 5.00 13.00 9410.00

sqft\_basement yr\_built yr\_renovated zipcode lat long \

count 21613.00 21613.00 21613.00 21613.00 21613.00 21613.00

mean 291.51 1971.01 84.40 98077.94 47.56 -122.21

std 442.58 29.37 401.68 53.51 0.14 0.14

min 0.00 1900.00 0.00 98001.00 47.16 -122.52

25% 0.00 1951.00 0.00 98033.00 47.47 -122.33

50% 0.00 1975.00 0.00 98065.00 47.57 -122.23

75% 560.00 1997.00 0.00 98118.00 47.68 -122.12

max 4820.00 2015.00 2015.00 98199.00 47.78 -121.31

sqft\_living15 sqft\_lot15

count 21613.00 21613.00

mean 1986.55 12768.46

std 685.39 27304.18

min 399.00 651.00

25% 1490.00 5100.00

50% 1840.00 7620.00

75% 2360.00 10083.00

max 6210.00 871200.00

**Summarize the most common values**

In [6]:

*# Summarize most common values for features with non-continuous or categorical values*

features\_to\_summarize **=** ['view','waterfront','grade','zipcode','bedrooms','bathrooms','floors']

data\_raw[features\_to\_summarize]**.**mode()

Out[6]:

|  | **view** | **waterfront** | **grade** | **zipcode** | **bedrooms** | **bathrooms** | **floors** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 0 | 0 | 7 | 98103 | 3 | 2.5 | 1.0 |

**Show correlations with price**

In [7]:

*# Look for correlations with price*

print('Pearson correlations with price')

corr\_matrix **=** data\_raw**.**corr()

corr\_matrix['price']**.**sort\_values(ascending**=False**)

Pearson correlations with price

Out[7]:

price 1.000000

sqft\_living 0.702035

grade 0.667434

sqft\_above 0.605567

sqft\_living15 0.585379

bathrooms 0.525138

view 0.397293

sqft\_basement 0.323816

bedrooms 0.308350

lat 0.307003

waterfront 0.266369

floors 0.256794

yr\_renovated 0.126434

sqft\_lot 0.089661

sqft\_lot15 0.082447

yr\_built 0.054012

condition 0.036362

long 0.021626

id -0.016762

zipcode -0.053203

Name: price, dtype: float64

**Analyze cross correlations**

In [8]:

*# Use Matplotlib for visualization*

**%matplotlib** inline

**import** matplotlib **as** mpl

**import** matplotlib.pyplot **as** plt

*# Specify size and title for the visualization*

f, axes **=** plt**.**subplots(figsize**=**(20, 20))

plt**.**title('All Correlations',fontsize**=**32)

*# For the purpose of visualization, we'll use a different order for the features.*

*# We'll start with price, to make it easier to compare all other features with it.*

features **=** ['price','bedrooms','bathrooms',

'sqft\_living','sqft\_living15','sqft\_lot','sqft\_lot15','sqft\_above','sqft\_basement',

'floors','waterfront',

'view','condition','grade',

'yr\_built','yr\_renovated',

'zipcode','lat','long']

*# Use Seaborn library to plot the correlation matrix as a heatmap*

sb**.**heatmap(data\_raw[features]**.**corr(),

linewidths **=** 3.0,

square **=** **True**,

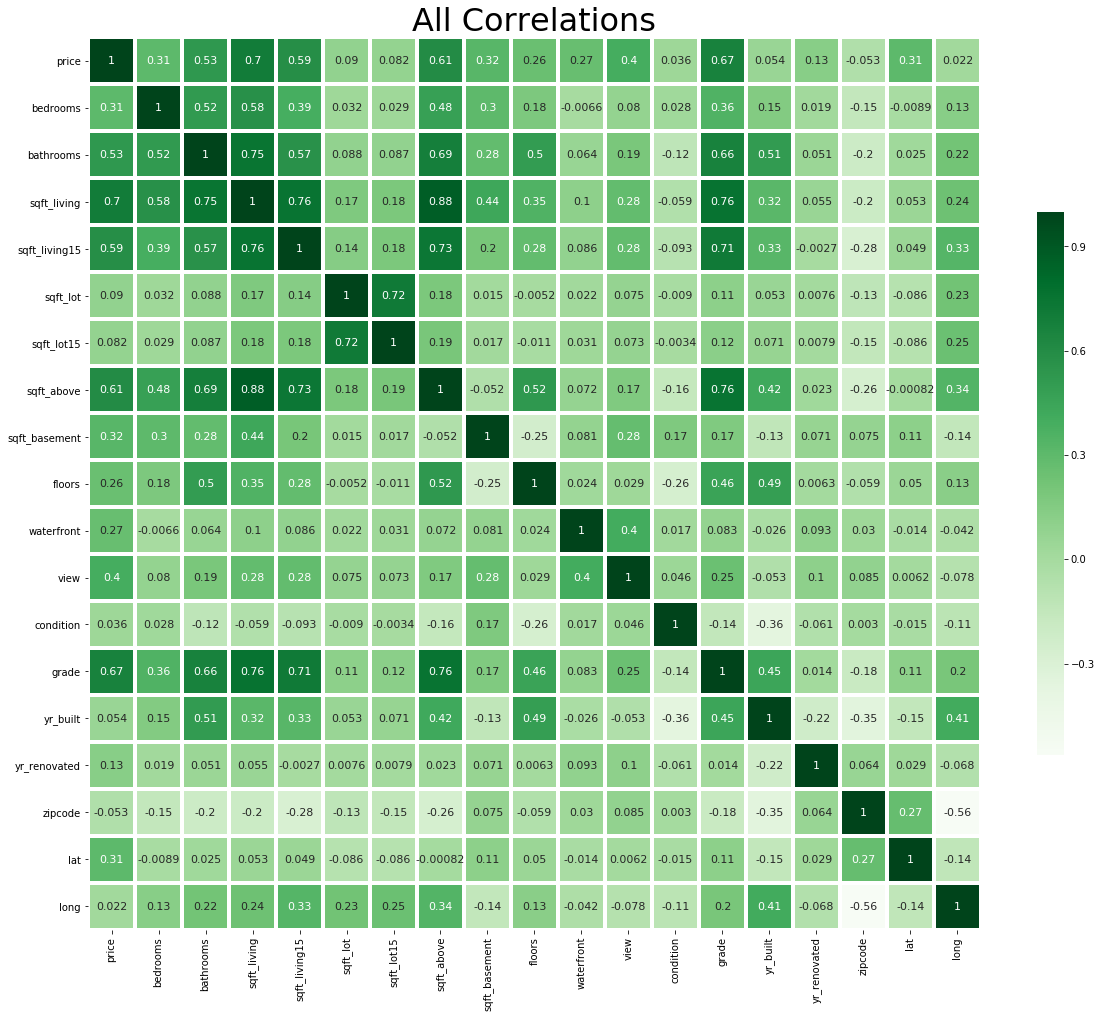
cmap **=** 'Greens',

linecolor**=**'w',

annot**=True**,

annot\_kws**=**{'size':11},

cbar\_kws**=**{'shrink': .5});



**Use histograms to visualize the distribution of various features**

In [9]:

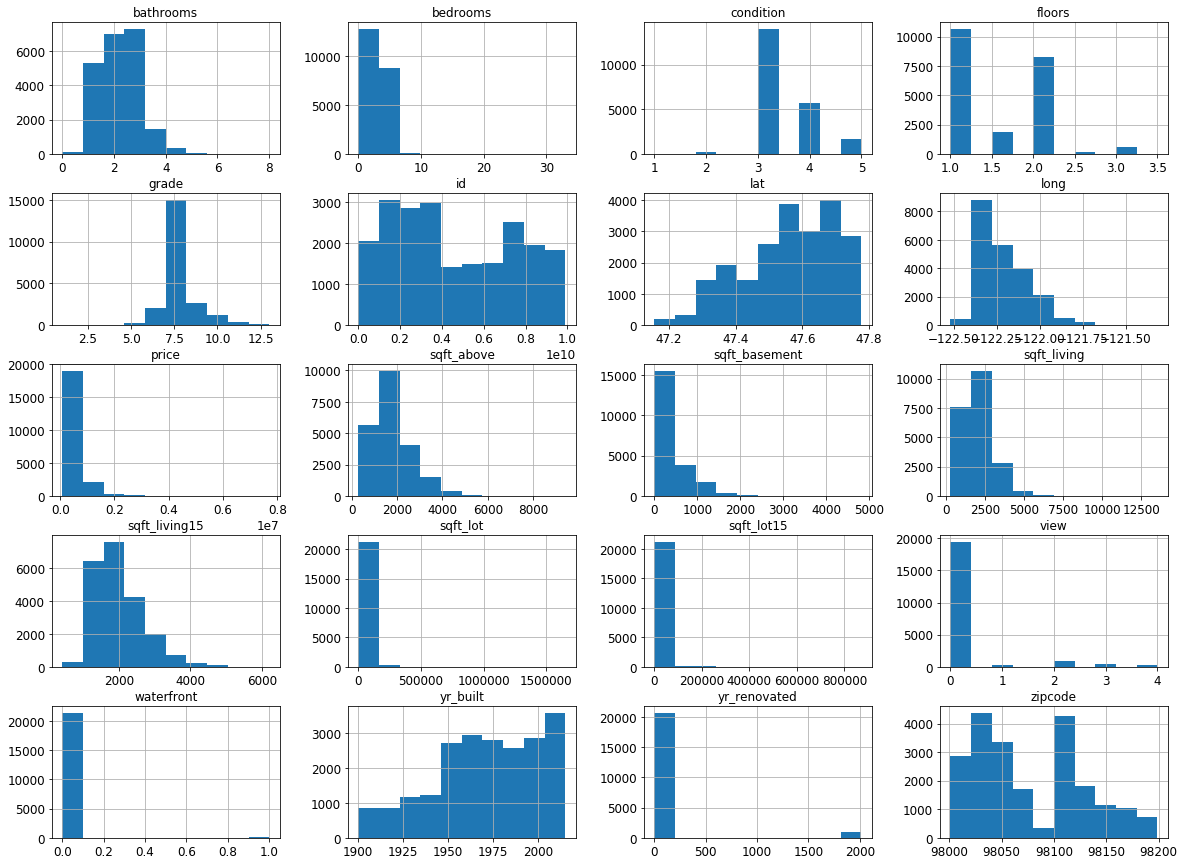
mpl**.**rc('axes', labelsize**=**14)

mpl**.**rc('xtick', labelsize**=**12)

mpl**.**rc('ytick', labelsize**=**12)

data\_raw**.**hist(figsize**=**(20,15));

plt**.**figure();



<Figure size 432x288 with 0 Axes>

**Visualize with a geographic map to gain insights regarding location**

In [10]:

*# To avoid overwhelming the visualization tool, we'll only plot every Nth house.*

n\_homes **=** 20

data\_raw\_subset **=** data\_raw**.**sort\_values(by **=**['price'], ascending **=** **False**)[::n\_homes]

*# Output highest house price*

max\_price **=** data\_raw\_subset**.**loc[data\_raw\_subset['price']**.**idxmax()]['price']

print(f'The highest home price in this dataset is ${max\_price:,.0f}')

*# Descriptions of the building grades used in King County*

*# Values obtained from http://www5.kingcounty.gov/sdc/FGDCDocs/resbldg\_extr\_faq.htm*

bldg\_grades **=** ['Unknown','Cabin','Substandard','Poor','Low','Fair',

'Low Average','Average','Good','Better',

'Very Good','Excellent','Luxury','Mansion','Exceptional Properties']

*# Use Folium library to plot values on a map.*

**import** folium

*# Generate the base map, centering on King County.*

base\_map **=** folium**.**Map(location **=** [47.5300, **-**122.2000],

control\_scale **=** **True**,

max\_zoom **=** 20,

zoom\_start **=** 10,

zoom\_control **=** **True**)

*# Plot homes by price.*

**for** index, row **in** data\_raw\_subset**.**iterrows():

*# Get the grade description for this row.*

grade\_desc **=** bldg\_grades[row['grade']]

waterfront\_desc **=** "Yes" **if** (row['waterfront'] **==** 1) **else** "No"

*# Add popup text. Click each point to show details.*

popup\_text **=** '<br>'**.**join(['King&nbsp;County&nbsp;Housing&nbsp;Sales&nbsp;Data',

'Price:&nbsp;${:,.0f}',

'Sqft&nbsp;Living:&nbsp;{:,.0f}',

'Grade:&nbsp;{}&nbsp;({})',

'Location:&nbsp;[{:.3f},{:.3f}]',

'Waterfront:&nbsp;{}',

'Zipcode:&nbsp;{}'])

popup\_text **=** popup\_text**.**format(row['price'],

row['sqft\_living'],

row['grade'], grade\_desc,

row['lat'], row['long'],

waterfront\_desc,

row['zipcode'])

*# Add each home to the map, but show larger dots for higher prices.*

scaling\_value **=** (row['price'] **/** max\_price) *# 1.0 for highest price.*

folium**.**CircleMarker([row['lat'], row['long']],

radius **=** 25 **\*** scaling\_value,

weight **=** 1,

fill **=** **True**,

fillColor **=** '#0000FF',

fillOpacity **=** 0.7,

color **=** '#0000FF',

opacity **=** 0.7,

popup **=** popup\_text)**.**add\_to(base\_map)

base\_map

The highest home price in this dataset is $7,700,000

Out[10]:

Make this Notebook Trusted to load map: File -> Trust Notebook

**Split the data into training and testing sets and labels**

In [11]:

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

*# Price is the dependent variable (value to be predicted), so it will be*

*# removed from the training data and put into a separate dataframe for labels.*

label\_columns **=** ['price']

training\_columns **=** ['sqft\_living',

'grade',

'bathrooms',

'view',

'sqft\_basement',

'bedrooms',

'lat',

'waterfront',

'floors',

'yr\_renovated',

'sqft\_lot',

'yr\_built',

'condition',

'long',

'zipcode']

*# Split independent and dependent variables.*

data\_train,data\_test,data\_train\_labels,data\_test\_labels **=** train\_test\_split(data\_raw[training\_columns],

data\_raw[label\_columns],

random\_state **=** 42)

*# Compare the number of rows and columns in the original data to the training and testing sets*

print(f'Original Set: {data\_raw**.**shape}')

print('------------------------------')

print(f'Training Features: {data\_train**.**shape}')

print(f'Testing Features: {data\_test**.**shape}')

print(f'Training Labels: {data\_train\_labels**.**shape}')

print(f'Testing Labels: {data\_test\_labels**.**shape}')

Original Set: (21613, 21)

------------------------------

Training Features: (16209, 15)

Testing Features: (5404, 15)

Training Labels: (16209, 1)

Testing Labels: (5404, 1)

**Build and test a linear regression model - Round 1**

In [12]:

**from** sklearn.linear\_model **import** LinearRegression

**from** time **import** time

*# Create a linear regression model*

regressor **=** LinearRegression()

*# Fit the model using training data and labels*

start **=** time()

regressor**.**fit(data\_train, data\_train\_labels);

end**=**time()

train\_time **=** (end **-** start) **\*** 1000

print('Model took {:,.2f} milliseconds to fit.'**.**format(train\_time))

Model took 5.67 milliseconds to fit.

**Use the holdout dataset to test the model**

In [13]:

*# Evaluate the model's performance using test data and labels*

score **=** regressor**.**score(data\_test, data\_test\_labels)

'Score: {}%'**.**format(int(round(score **\*** 100)))

Out[13]:

'Score: 70%'

**Compare predicted values to actual values**

In [14]:

predicted\_prices **=** regressor**.**predict(data\_test)

predictions **=** data\_test\_labels**.**copy()

predictions['predicted'] **=** predicted\_prices

*# View examples comparing actual prices to predicted prices*

**with** pd**.**option\_context('float\_format', '${:,.2f}'**.**format): print( predictions**.**head(10) )

price predicted

735 $365,000.00 $451,576.87

2830 $865,000.00 $745,528.73

4106 $1,038,000.00 $1,234,144.11

16218 $1,490,000.00 $1,659,505.67

19964 $711,000.00 $737,851.90

1227 $211,000.00 $284,352.79

18849 $790,000.00 $832,187.44

19369 $680,000.00 $490,462.65

20164 $384,500.00 $392,922.89

7139 $605,000.00 $471,310.48

In [15]:

**def** compare\_pred\_to\_actual(chart\_description):

N **=** 10 *# Plot every Nth value to save time and space*

pred\_df **=** predictions**.**sort\_values('price')[::N]

pred\_df['diff'] **=** pred\_df['price'] **-** pred\_df['predicted']

pred\_df['recnum'] **=** np**.**arange(len(pred\_df))

pred\_df['error\_pct'] **=** abs(pred\_df['diff']**/**pred\_df['price'])**\***100

ax **=** plt**.**figure(figsize**=**[18,10])

plt**.**ylabel('Price')

plt**.**xlabel('House')

plt**.**plot(pred\_df['recnum'], pred\_df['price'], color**=**'blue');

plt**.**scatter(pred\_df['recnum'],

pred\_df['predicted'],

pred\_df['error\_pct'],

color**=**'red');

ax**.**legend(['Actual','Predicted'],

loc**=**'lower center',

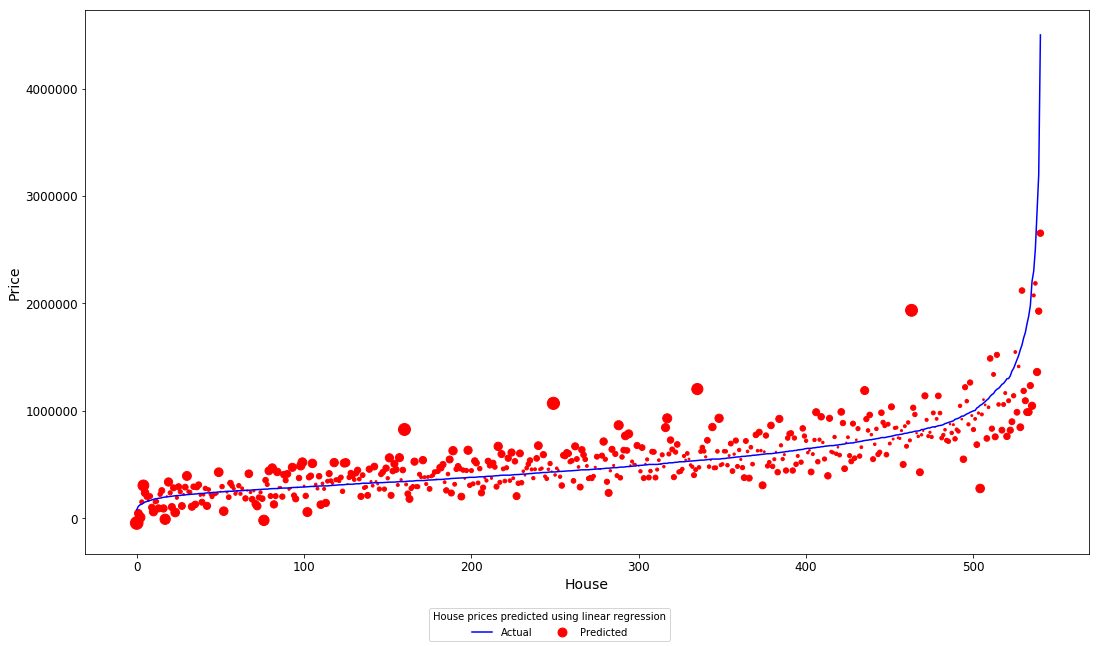
ncol**=**2,

title**=**chart\_description)

plt**.**show()

*# Compare the predicted prices to actual prices*

compare\_pred\_to\_actual('House prices predicted using linear regression')



**Identify outliers**

In [16]:

feature\_list **=** ['price','bedrooms']

**for** feature **in** feature\_list:

plt**.**figure(figsize**=**(20,2))

bplot **=** sb**.**boxplot(x**=**feature, data**=**data\_raw, orient**=**"h", fliersize**=**7)





**Examine data values in the outliers**

In [17]:

*# Houses with a value above $6,000,000*

data\_train**.**loc[data\_train\_labels['price'] **>** 6000000]

Out[17]:

|  | **sqft\_living** | **grade** | **bathrooms** | **view** | **sqft\_basement** | **bedrooms** | **lat** | **waterfront** | **floors** | **yr\_renovated** | **sqft\_lot** | **yr\_built** | **condition** | **long** | **zipcode** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **3914** | 10040 | 11 | 4.50 | 2 | 2360 | 5 | 47.6500 | 1 | 2.0 | 2001 | 37325 | 1940 | 3 | -122.214 | 98004 |
| **9254** | 9890 | 13 | 7.75 | 4 | 1030 | 6 | 47.6305 | 0 | 2.0 | 0 | 31374 | 2001 | 3 | -122.240 | 98039 |
| **7252** | 12050 | 13 | 8.00 | 3 | 3480 | 6 | 47.6298 | 0 | 2.5 | 1987 | 27600 | 1910 | 4 | -122.323 | 98102 |

In [18]:

*# Houses with more than 11 bedrooms*

data\_train**.**loc[data\_train['bedrooms'] **>** 11]

Out[18]:

|  | **sqft\_living** | **grade** | **bathrooms** | **view** | **sqft\_basement** | **bedrooms** | **lat** | **waterfront** | **floors** | **yr\_renovated** | **sqft\_lot** | **yr\_built** | **condition** | **long** | **zipcode** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **15870** | 1620 | 7 | 1.75 | 0 | 580 | 33 | 47.6878 | 0 | 1.0 | 0 | 6000 | 1947 | 5 | -122.331 | 98103 |

**Drop outliers from the training dataset**

In [19]:

print(f'{len(data\_train):6d} houses in the training dataset')

*# Keep only the rows for houses priced $6M or less*

data\_train **=** data\_train**.**loc[data\_train\_labels['price'] **<=** 6000000]

data\_train\_labels **=** data\_train\_labels**.**loc[data\_train\_labels['price'] **<=** 6000000]

print(f'{len(data\_train):6d} houses remain after dropping those priced over $6M')

*# Keep only the rows for houses with 11 or fewer bedrooms*

data\_train\_labels **=** data\_train\_labels**.**loc[data\_train['bedrooms'] **<=** 11]

data\_train **=** data\_train**.**loc[data\_train['bedrooms'] **<=** 11]

print(f'{len(data\_train):6d} houses remain after dropping those with more than 11 bedrooms')

16209 houses in the training dataset

16206 houses remain after dropping those priced over $6M

16205 houses remain after dropping those with more than 11 bedrooms

**Show statistics for the training features**

In [20]:

*# Show statistics for the features we'll be using, to prepare for feature scaling.*

**with** pd**.**option\_context('float\_format', '{:.2f}'**.**format):

print(data\_train['sqft\_living']**.**describe(), '\n')

print(data\_train\_labels['price']**.**describe())

count 16205.00

mean 2071.71

std 899.46

min 290.00

25% 1427.00

50% 1910.00

75% 2544.00

max 9640.00

Name: sqft\_living, dtype: float64

count 16205.00

mean 536227.60

std 348666.79

min 75000.00

25% 320000.00

50% 450000.00

75% 640000.00

max 5110800.00

Name: price, dtype: float64

**Compare the scale and distribution of price and sqft\_living**

In [21]:

*# Compare scale and distribution of price and sqft\_living*

**def** compare\_price\_sqft():

print('Maximum price =', data\_train\_labels**.**loc[data\_train\_labels['price']**.**idxmax()]['price']);

print('Maximum sqft\_living =', data\_train**.**loc[data\_train['sqft\_living']**.**idxmax()]['sqft\_living']);

fig **=** plt**.**figure(figsize**=**(15,3))

fig**.**subplots\_adjust(wspace**=**.4)

plt**.**rc('axes', titlesize**=**9) *# fontsize of the axes title*

plt**.**rc('axes', labelsize**=**11) *# fontsize of the x and y labels*

plt**.**rc('xtick', labelsize**=**8) *# fontsize of the tick labels*

plt**.**rc('ytick', labelsize**=**8) *# fontsize of the tick labels*

ax1**=**fig**.**add\_subplot(1, 3, 1)

plt**.**xlabel('price')

plt**.**hist(data\_train\_labels['price'], label**=**'price');

ax2**=**fig**.**add\_subplot(1, 3, 2)

plt**.**xlabel('sqft\_living')

plt**.**hist(data\_train['sqft\_living'], label**=**'sqft\_living');

*# View relationship between price and sqft\_living*

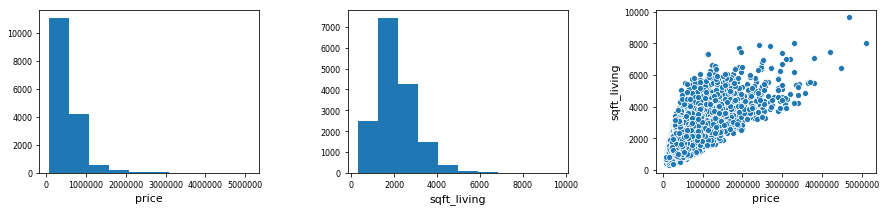
ax2**=**fig**.**add\_subplot(1, 3, 3)

sb**.**scatterplot(x**=**data\_train\_labels['price'], y**=**data\_train['sqft\_living']);

compare\_price\_sqft()

Maximum price = 5110800.0

Maximum sqft\_living = 9640.0



**Transform price and sqft\_living, and compare results**

In [22]:

*# Apply a log transformation to scale price and sqft\_living*

data\_train['sqft\_living'] **=** np**.**log(data\_train['sqft\_living'])

data\_train\_labels['price'] **=** np**.**log(data\_train\_labels['price'])

*# Log transformation must be applied to test dataset as well*

data\_test['sqft\_living'] **=** np**.**log(data\_test['sqft\_living'])

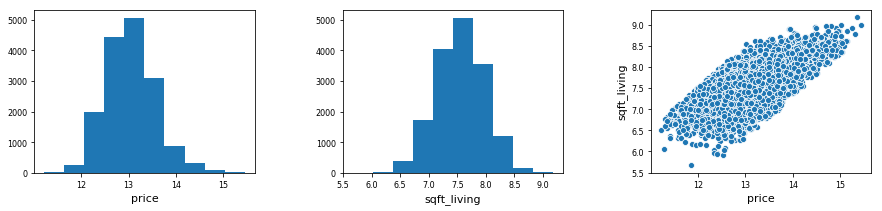
data\_test\_labels['price'] **=** np**.**log(data\_test\_labels['price'])

*# Compare scale and distribution of price and sqft\_living*

compare\_price\_sqft()

Maximum price = 15.446866505699305

Maximum sqft\_living = 9.17367638760459



**Build and test a linear regression model - Round 2**

In [23]:

**from** sklearn.linear\_model **import** LinearRegression

*# Create a linear regression model and fit it using the training data*

regressor **=** LinearRegression()

start **=** time()

regressor**.**fit(data\_train, data\_train\_labels);

end**=**time()

train\_time **=** (end **-** start) **\*** 1000

print('Model took {:,.2f} milliseconds to fit.'**.**format(train\_time))

*# Evaluate the model's performance*

score **=** regressor**.**score(data\_test, data\_test\_labels)

'Score: {}%'**.**format(int(round(score **\*** 100)))

Model took 5.95 milliseconds to fit.

Out[23]:

'Score: 77%'

**Compare the first ten predictions to actual values**

In [24]:

*# y\_pred is the predicted prices that will be produced by testing*

predicted\_prices **=** regressor**.**predict(data\_test)

predictions **=** data\_test\_labels**.**copy()

predictions['predicted'] **=** predicted\_prices

*# View examples of the transformed prices*

**with** pd**.**option\_context('float\_format', '${:,.2f}'**.**format): print( predictions**.**head(10) )

price predicted

735 $12.81 $12.95

2830 $13.67 $13.44

4106 $13.85 $14.01

16218 $14.21 $14.58

19964 $13.47 $13.45

1227 $12.26 $12.62

18849 $13.58 $13.59

19369 $13.43 $13.04

20164 $12.86 $12.92

7139 $13.31 $12.90

**Convert the prices back to actual values**

In [25]:

*# Need to call exp() function to convert back from log value to actual price.*

**import** math

predictions **=** predictions**.**applymap(math**.**exp)

*# View examples of the actual and predicted prices*

**with** pd**.**option\_context('float\_format', '${:,.2f}'**.**format): print( predictions**.**head(10) )

price predicted

735 $365,000.00 $420,860.44

2830 $865,000.00 $689,455.76

4106 $1,038,000.00 $1,220,057.79

16218 $1,490,000.00 $2,139,532.72

19964 $711,000.00 $696,523.60

1227 $211,000.00 $301,303.20

18849 $790,000.00 $800,320.80

19369 $680,000.00 $462,471.84

20164 $384,500.00 $408,170.22

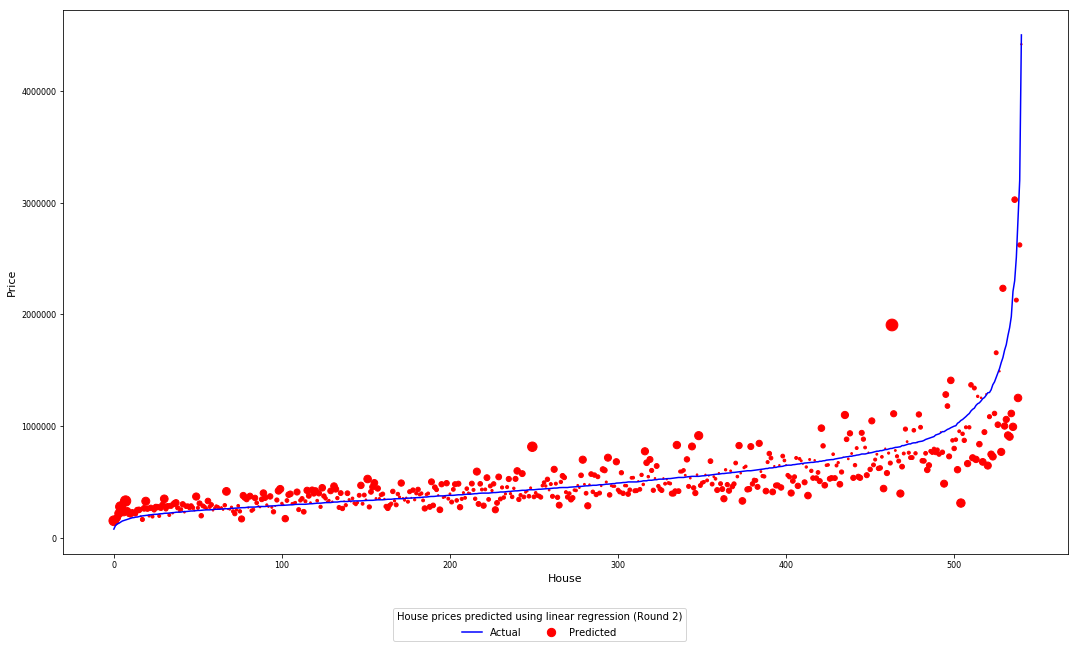
7139 $605,000.00 $398,681.64

**Compare predicted values to actual values (Round 2)**

In [26]:

*# Compare the predicted prices to actual prices*

compare\_pred\_to\_actual('House prices predicted using linear regression (Round 2)')



**Try a different algorithm**

In [27]:

*# Create a model using the random forest algorithm.*

**from** sklearn.ensemble **import** RandomForestRegressor

rnd\_forest **=** RandomForestRegressor(n\_estimators**=**100,random\_state**=**0)

start **=** time()

rnd\_forest**.**fit(data\_train, data\_train\_labels**.**values**.**ravel())

end**=**time()

train\_time **=** (end **-** start) **\*** 1000

print('Model took {:,.2f} milliseconds to fit.'**.**format(train\_time))

score **=** rnd\_forest**.**score(data\_test, data\_test\_labels)

print('Score: {}%'**.**format(int(round(score **\*** 100))))

Model took 9,180.19 milliseconds to fit.

Score: 89%

**View examples of the actual and predicted prices**

In [28]:

predicted\_prices **=** rnd\_forest**.**predict(data\_test)

predictions **=** data\_test\_labels**.**copy()

predictions['predicted'] **=** predicted\_prices

*# Scale the prices back to actual values.*

predictions **=** predictions**.**applymap(math**.**exp)

*# View examples of the actual and predicted prices*

**with** pd**.**option\_context('float\_format', '${:,.2f}'**.**format): print( predictions**.**head(10) )

price predicted

735 $365,000.00 $367,229.33

2830 $865,000.00 $797,830.56

4106 $1,038,000.00 $1,112,227.86

16218 $1,490,000.00 $1,866,465.53

19964 $711,000.00 $709,144.08

1227 $211,000.00 $251,855.28

18849 $790,000.00 $870,082.54

19369 $680,000.00 $562,736.77

20164 $384,500.00 $413,123.06

7139 $605,000.00 $534,529.34

In [29]:

*# Compare the new predicted prices to actual prices*

compare\_pred\_to\_actual('House prices predicted using random forest')

