Import Libraries

170
```
# Import all libraries used including the libraries that were used to determine the best model
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sn
import os
import glob
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score, explained_variance_score
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler
import openpyxl
import pickle
```

171
```
#set working directory to where data is stored
os.chdir(r"C:\Users\belbi\OneDrive\Documents\GCU\Capstone\Data\All Data")
```

172
```
#get filenames of all datafiles
csv_files = glob.glob('*.{}'.format('csv'))
csv_files
```

172
```
['redfin_2022-11-06-15-36-04.csv',
 'redfin_2022-11-06-15-54-57.csv',
 'redfin_2022-11-06-15-55-50.csv',
 'redfin_2022-11-06-15-56-31.csv',
 'redfin_2022-11-06-15-57-35.csv',
 'redfin_2022-11-06-15-58-16.csv',
 'redfin_2022-11-06-15-59-07.csv',
 'redfin_2022-11-06-15-59-57.csv',
 'redfin_2022-11-06-16-00-40.csv',
 'redfin_2022-11-06-16-01-03.csv',
 'redfin_2022-11-06-16-02-21.csv',
 'redfin_2022-11-06-16-04-07.csv',
 'redfin_2022-11-06-16-04-47.csv',
 'redfin_2022-11-06-16-07-10.csv',
 'redfin_2022-11-06-16-07-54.csv',
 'redfin_2022-11-06-16-08-30.csv',
 'redfin_2022-11-06-16-08-59.csv',
 'redfin_2022-11-06-16-09-39.csv',
 'redfin_2022-11-06-16-09-57.csv',
 'redfin_2022-11-06-16-10-12.csv',
 'redfin_2022-11-06-16-10-30.csv',
 'redfin_2022-11-06-16-11-00.csv',
 'redfin_2022-11-06-16-11-16.csv',
 'redfin_2022-11-06-16-11-32.csv',
 'redfin_2022-11-06-16-12-13.csv',
 'redfin_2022-11-06-16-12-24.csv',
 'redfin_2022-11-06-16-12-50.csv',
 'redfin_2022-11-06-16-13-03.csv',
 'redfin_2022-11-06-16-13-27.csv',
 'redfin_2022-11-06-16-13-41.csv',
 'redfin_2022-11-06-16-13-55.csv',
 'redfin_2022-11-06-16-14-05.csv',
 'redfin_2022-11-06-16-14-33.csv',
 'redfin_2022-11-06-16-15-04.csv',
 'redfin_2022-11-06-16-15-27.csv',
```

```
 'redfin_2022-11-06-16-15-47.csv',
 'redfin_2022-11-06-16-16-03.csv',
 'redfin_2022-11-06-16-16-52.csv',
 'redfin_2022-11-06-16-17-14.csv',
 'redfin_2022-11-06-16-17-28.csv',
 'redfin_2022-11-06-16-17-42.csv',
 'redfin_2022-11-09-17-52-28.csv',
 'redfin_2022-11-09-17-53-28.csv',
 'redfin_2022-11-09-17-54-08.csv',
 'redfin_2022-11-09-17-54-22.csv',
 'redfin_2022-11-09-17-54-53.csv',
 'redfin_2022-11-09-17-55-09.csv',
 'redfin_2022-11-09-17-56-15.csv',
 'redfin_2022-11-09-17-56-44.csv',
 'redfin_2022-11-09-17-56-56.csv',
 'redfin_2022-11-09-17-57-10.csv',
 'redfin_2022-11-09-17-58-27.csv',
 'redfin_2022-11-09-17-59-02.csv',
 'redfin_2022-11-09-17-59-25.csv',
 'redfin_2022-11-09-17-59-44.csv',
 'redfin_2022-11-09-17-59-59.csv',
 'redfin_2022-11-09-18-00-50.csv',
 'redfin_2022-11-09-18-02-05.csv',
 'redfin_2022-11-09-18-02-32.csv']
```

173

```
#build the dataframe by appending each file to the dataframe
df_append = pd.DataFrame()
for file in csv_files:
    df_temp = pd.read_csv(file)
    df_append = df_append.append(df_temp, ignore_index = True)
df_append
```

```
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
```

```
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
```

```
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1639621125.py:5: FutureWarning: The frame.append method
  df_append = df_append.append(df_temp, ignore_index = True)
```

173

	SALE TYPE	SOLD DATE	PROPERTY TYPE	ADDRESS	CITY	STATE OR PROVINCE	ZIP OR POSTAL CODE	PRICE
0	PAST SALE	NaN	Single Family Residential	1846 E Willetta St	Phoenix	AZ	85006.0	210000.0
1	PAST SALE	NaN	Single Family Residential	1515 E Brill St	Phoenix	AZ	85006.0	360000.0
2	PAST SALE	NaN	Single Family Residential	1632 E Granada Rd	Phoenix	AZ	85006.0	245000.0
3	PAST SALE	September-16-2022	Single Family Residential	1113 W PORTLAND St	Phoenix	AZ	85007.0	780000.0
4	PAST SALE	NaN	Single Family Residential	946 W Cocopah St	Phoenix	AZ	85007.0	258000.0
...	...	...	...	...	...	...	...	...
14081	PAST SALE	November-12-2021	Single Family Residential	3543 W STEINBECK Ct	Anthem	AZ	85086.0	460000.0
14082	PAST SALE	November-12-2021	Single Family Residential	34702 N 25TH Ln	Phoenix	AZ	85086.0	498000.0

	SALE TYPE	SOLD DATE	PROPERTY TYPE	ADDRESS	CITY	STATE OR PROVINCE	ZIP OR POSTAL CODE	PRICE
14083	PAST SALE	November-12-2021	Single Family Residential	3728 W MEDINAH Way	Anthem	AZ	85086.0	473000.0
14084	PAST SALE	November-10-2021	Single Family Residential	4642 W ROLLING ROCK Dr	Phoenix	AZ	85086.0	469000.0
14085	PAST SALE	November-9-2021	Single Family Residential	39829 N RIVER BEND Rd	Phoenix	AZ	85086.0	460000.0

14086 rows × 28 columns

174
```
df = df_append
```

175
```
#explore the first five rows of data
df.head()
```

175

	SALE TYPE	SOLD DATE	PROPERTY TYPE	ADDRESS	CITY	STATE OR PROVINCE	ZIP OR POSTAL CODE	PRICE	BE
0	PAST SALE	NaN	Single Family Residential	1846 E Willetta St	Phoenix	AZ	85006.0	210000.0	Na
1	PAST SALE	NaN	Single Family Residential	1515 E Brill St	Phoenix	AZ	85006.0	360000.0	Na
2	PAST SALE	NaN	Single Family Residential	1632 E Granada Rd	Phoenix	AZ	85006.0	245000.0	Na
3	PAST SALE	September-16-2022	Single Family Residential	1113 W PORTLAND St	Phoenix	AZ	85007.0	780000.0	4.0
4	PAST SALE	NaN	Single Family Residential	946 W Cocopah St	Phoenix	AZ	85007.0	258000.0	Na

5 rows × 28 columns

176
```
#view statistics on all columns in the data
df.describe(include='all')
```

176

	SALE TYPE	SOLD DATE	PROPERTY TYPE	ADDRESS	CITY	STATE OR PROVINCE	ZIP OR POSTAL CODE	
count	14086	11310	14086	14073	14073	14086	14073.000000	1.40700
unique	1	296	7	9841	8	1	NaN	NaN
top	PAST SALE	February-28-2022	Single Family Residential	3329 E Palm Ln	Phoenix	AZ	NaN	NaN
freq	14086	121	14029	9	13105	14086	NaN	NaN
mean	NaN	NaN	NaN	NaN	NaN	NaN	85047.239324	3.80337
std	NaN	NaN	NaN	NaN	NaN	NaN	71.250392	7.88927
min	NaN	NaN	NaN	NaN	NaN	NaN	85003.000000	4.50000
25%	NaN	NaN	NaN	NaN	NaN	NaN	85019.000000	3.38000
50%	NaN	NaN	NaN	NaN	NaN	NaN	85032.000000	3.87000
75%	NaN	NaN	NaN	NaN	NaN	NaN	85042.000000	4.36550
max	NaN	NaN	NaN	NaN	NaN	NaN	85392.000000	1.82500

11 rows × 28 columns

177
```
#look for any null values contained in columns
df.isnull().sum()
```

177
```
SALE TYPE                        0
SOLD DATE                     2776
PROPERTY TYPE                    0
ADDRESS                         13
CITY                            13
STATE OR PROVINCE                0
ZIP OR POSTAL CODE              13
PRICE                           16
BEDS                          2714
BATHS                           72
LOCATION                      2822
SQUARE FEET                     30
LOT SIZE                        18
YEAR BUILT                      30
DAYS ON MARKET               14086
$/SQUARE FEET                   46
HOA/MONTH                    11184
STATUS                        2760
NEXT OPEN HOUSE START TIME   14086
```

```
NEXT OPEN HOUSE END TIME                                                                14086
URL (SEE https://www.redfin.com/buy-a-home/comparative-market-analysis FOR INFO ON PRICING)      0
SOURCE                                                                                  2760
MLS#                                                                                    2776
FAVORITE                                                                                   0
INTERESTED                                                                                 0
LATITUDE                                                                                   0
LONGITUDE                                                                                  0
POOL                                                                                       0
dtype: int64
```

178
```
#view the data types for each of the columns in teh data
df.dtypes
```

178
```
SALE TYPE                                                                              object
SOLD DATE                                                                              object
PROPERTY TYPE                                                                          object
ADDRESS                                                                                object
CITY                                                                                   object
STATE OR PROVINCE                                                                      object
ZIP OR POSTAL CODE                                                                    float64
PRICE                                                                                 float64
BEDS                                                                                  float64
BATHS                                                                                 float64
LOCATION                                                                               object
SQUARE FEET                                                                           float64
LOT SIZE                                                                              float64
YEAR BUILT                                                                            float64
DAYS ON MARKET                                                                        float64
$/SQUARE FEET                                                                         float64
HOA/MONTH                                                                             float64
STATUS                                                                                 object
NEXT OPEN HOUSE START TIME                                                            float64
NEXT OPEN HOUSE END TIME                                                              float64
URL (SEE https://www.redfin.com/buy-a-home/comparative-market-analysis FOR INFO ON PRICING)    object
SOURCE                                                                                 object
MLS#                                                                                  float64
FAVORITE                                                                               object
INTERESTED                                                                             object
LATITUDE                                                                              float64
LONGITUDE                                                                             float64
POOL                                                                                   object
dtype: object
```

179
```
#convert the sold date to a date
date_string = df['SOLD DATE']
df['SOLDDATE'] = pd.to_datetime(date_string)
```

180
```
#removing data without MLS# as these most likely represent private sales
df = df.dropna(subset=['MLS#'],axis=0)
```

181
```
#after removing the data without MLS# determine how many columns still contain null values
df.isnull().sum()
```

181
```
SALE TYPE                                                                                  0
SOLD DATE                                                                                  0
PROPERTY TYPE                                                                              0
ADDRESS                                                                                    0
CITY                                                                                       0
STATE OR PROVINCE                                                                          0
ZIP OR POSTAL CODE                                                                         0
```

```
PRICE                                                                                  0
BEDS                                                                                   0
BATHS                                                                                  5
LOCATION                                                                              46
SQUARE FEET                                                                            0
LOT SIZE                                                                               4
YEAR BUILT                                                                             0
DAYS ON MARKET                                                                     11310
$/SQUARE FEET                                                                          0
HOA/MONTH                                                                           8408
STATUS                                                                                 0
NEXT OPEN HOUSE START TIME                                                         11310
NEXT OPEN HOUSE END TIME                                                           11310
URL (SEE https://www.redfin.com/buy-a-home/comparative-market-analysis FOR INFO ON PRICING)    0
SOURCE                                                                                 0
MLS#                                                                                   0
FAVORITE                                                                               0
INTERESTED                                                                             0
LATITUDE                                                                               0
LONGITUDE                                                                              0
POOL                                                                                   0
SOLDDATE                                                                               0
dtype: int64
```

182
```
#remove the rows of data where the bed, baths or lotsize data is missing
df = df.dropna(subset=['BEDS'],axis=0)
df = df.dropna(subset=['BATHS'],axis=0)
df = df.dropna(subset=['LOT SIZE'],axis=0)
```

183
```
#build an HOA data object based on whether there is an amount in the HOA/month column
df['HOA'] = np.where(df['HOA/MONTH'] > 0 ,1, 0)
```

184
```
#convert the Pool Y/N data to 1/0 values based on Y =1 and all else is 0
df['POOL'] = np.where(df['POOL'] == 'Y', 1,0)
```

185
```
#view the dataframe after making the changes above
df
```

185

	SALE TYPE	SOLD DATE	PROPERTY TYPE	ADDRESS	CITY	STATE OR PROVINCE	ZIP OR POSTAL CODE	PRICE
3	PAST SALE	September-16-2022	Single Family Residential	1113 W PORTLAND St	Phoenix	AZ	85007.0	780000.0
5	PAST SALE	September-9-2022	Single Family Residential	1425 E PORTLAND St	Phoenix	AZ	85006.0	385000.0
6	PAST SALE	August-8-2022	Single Family Residential	1501 W Willetta St	Phoenix	AZ	85007.0	649999.0
8	PAST SALE	August-29-2022	Single Family Residential	1515 W YAVAPAI St W	Phoenix	AZ	85007.0	350000.0

	SALE TYPE	SOLD DATE	PROPERTY TYPE	ADDRESS	CITY	STATE OR PROVINCE	ZIP OR POSTAL CODE	PRICE
9	PAST SALE	September-30-2022	Single Family Residential	810 S 3RD Ave	Phoenix	AZ	85003.0	614000.0
...	...	...	...	...	...	...	...	...
14081	PAST SALE	November-12-2021	Single Family Residential	3543 W STEINBECK Ct	Anthem	AZ	85086.0	460000.0
14082	PAST SALE	November-12-2021	Single Family Residential	34702 N 25TH Ln	Phoenix	AZ	85086.0	498000.0
14083	PAST SALE	November-12-2021	Single Family Residential	3728 W MEDINAH Way	Anthem	AZ	85086.0	473000.0
14084	PAST SALE	November-10-2021	Single Family Residential	4642 W ROLLING ROCK Dr	Phoenix	AZ	85086.0	469000.0
14085	PAST SALE	November-9-2021	Single Family Residential	39829 N RIVER BEND Rd	Phoenix	AZ	85086.0	460000.0

11301 rows × 30 columns

186
```
# select the columns to be used in the model as well as the app for analysis
#rename long column names and remove spaces in column names
data = df[['SOLDDATE','ADDRESS','CITY','STATE OR PROVINCE','ZIP OR POSTAL CODE','PRICE','BEDS','BATHS','S
data = data.rename(columns = {'ZIP OR POSTAL CODE':'ZIPCODE'})
data = data.rename(columns = {'SQUARE FEET':'SQFT'})
data = data.rename(columns = {'LOT SIZE':'LOTSIZE'})
data = data.rename(columns = {'YEAR BUILT':'YEARBUILT'})
```

187
```
#limit the dataset to sold price of less than $500k
data = data[data.PRICE <500000]
```

188
```
#Remove zipcodes captured that are not Phoenix
data = data[data.ZIPCODE < 85099]
```

189
```
#Remove Duplicates and view shape of the data
data.duplicated().sum()
data.drop_duplicates
data.shape
```

189
```
(10476, 16)
```

```
190 #Verify that there are no missing values in the new dataset
    data.isnull().sum()
```

```
190 SOLDDATE             0
    ADDRESS              0
    CITY                 0
    STATE OR PROVINCE    0
    ZIPCODE              0
    PRICE                0
    BEDS                 0
    BATHS                0
    SQFT                 0
    LOTSIZE              0
    YEARBUILT            0
    MLS#                 0
    POOL                 0
    HOA                  0
    LATITUDE             0
    LONGITUDE            0
    dtype: int64
```

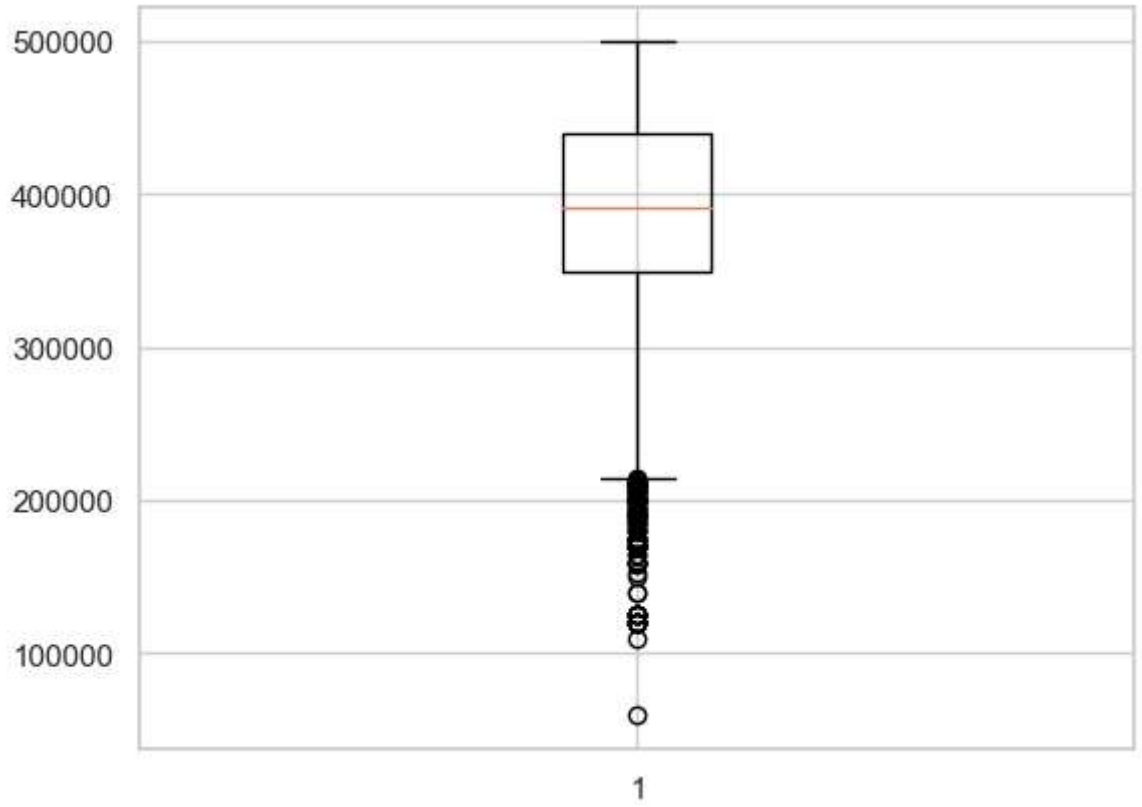
```
191 #covert zipcode to category data for one hot encoding
    data['ZIPCODE']=data['ZIPCODE'].astype('int64')
    data['YEARBUILT']=data['YEARBUILT'].astype('int64')
    data['SQFT']=data['SQFT'].astype('int64')
    data['LOTSIZE']=data['LOTSIZE'].astype('int64')
    data['BEDS']=data['BEDS'].astype('int32')
```

Exploratory Data Analysis

Explore Price

```
192 #original view of price to determine outliers that need to be removed

    plt.boxplot(data['PRICE'])
```

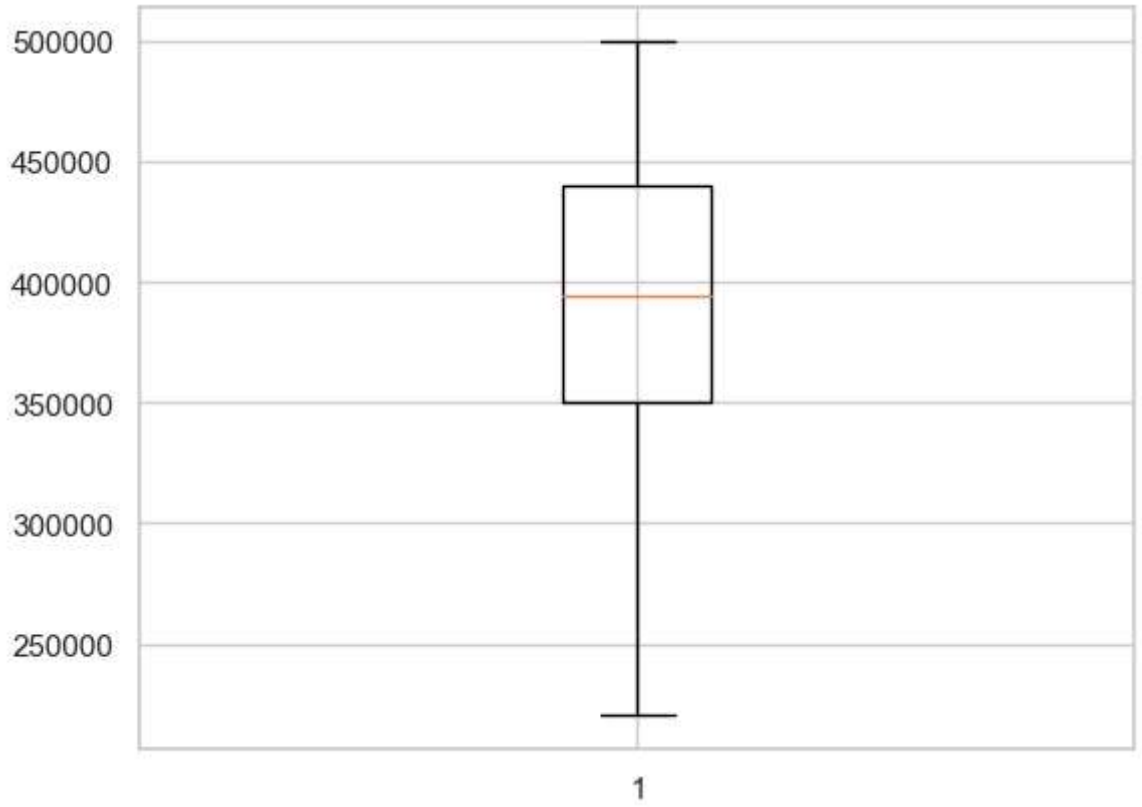
```
192 {'whiskers': [<matplotlib.lines.Line2D at 0x1b4d24a6880>,
      <matplotlib.lines.Line2D at 0x1b4d24a6370>],
     'caps': [<matplotlib.lines.Line2D at 0x1b4d24c0190>,
      <matplotlib.lines.Line2D at 0x1b4d24a63a0>],
     'boxes': [<matplotlib.lines.Line2D at 0x1b4d24a0f40>],
     'medians': [<matplotlib.lines.Line2D at 0x1b4d24a6a60>],
     'fliers': [<matplotlib.lines.Line2D at 0x1b4d24a6be0>],
     'means': []}
```



193
```
#remove outliers less than 220,000
data = data[data.PRICE>220000]
```

194
```
#view after removing the outliers
plt.boxplot(data['PRICE'])
```

194
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1b49a2e1910>,
  <matplotlib.lines.Line2D at 0x1b49a2e1bb0>],
 'caps': [<matplotlib.lines.Line2D at 0x1b49a2e1e80>,
  <matplotlib.lines.Line2D at 0x1b49a2f0160>],
 'boxes': [<matplotlib.lines.Line2D at 0x1b49a2e1670>],
 'medians': [<matplotlib.lines.Line2D at 0x1b49a2f0400>],
 'fliers': [<matplotlib.lines.Line2D at 0x1b49a2f06a0>],
 'means': []}
```



Compare house size to price

```
195 x=data['PRICE']
    y=data['SQFT']
    plt.scatter(x,y)
    plt.title('House Sq Ft versus Price')
    plt.xlabel('Price')
    plt.ylabel('Sq Ft')

    z=np.polyfit(x,y,1)
    p=np.poly1d(z)
    plt.plot(x,p(x), color ='red', linewidth=3)
```

```
195 [<matplotlib.lines.Line2D at 0x1b49a642370>]
```

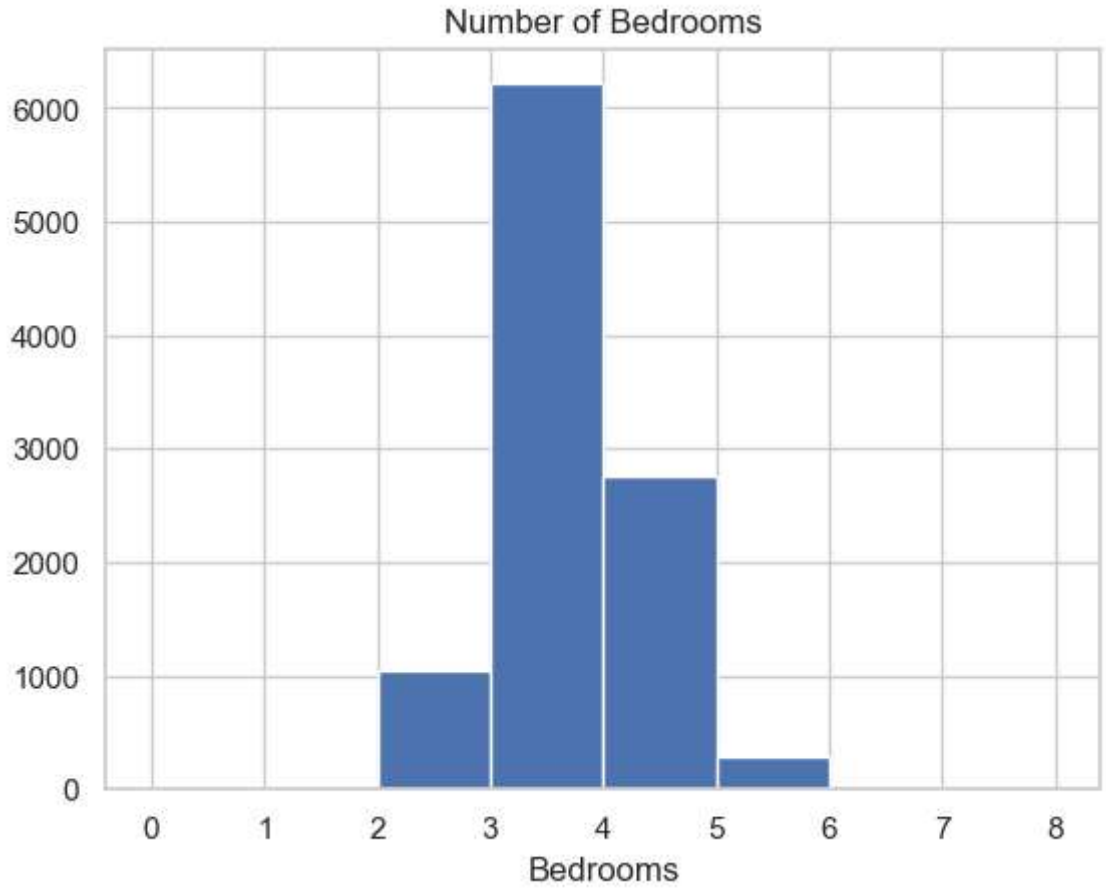
Compare number of bedrooms to price

196
```
y=data['PRICE']
x=data['BEDS']
plt.scatter(x,y)
plt.title('Number of Bedrooms versus Price')
plt.xlabel('Bedrooms')
plt.ylabel('Price')

z=np.polyfit(x,y,1)
p=np.poly1d(z)
plt.plot(x,p(x), color ='red', linewidth=3)
```

196
```
[<matplotlib.lines.Line2D at 0x1b49a6b5a30>]
```

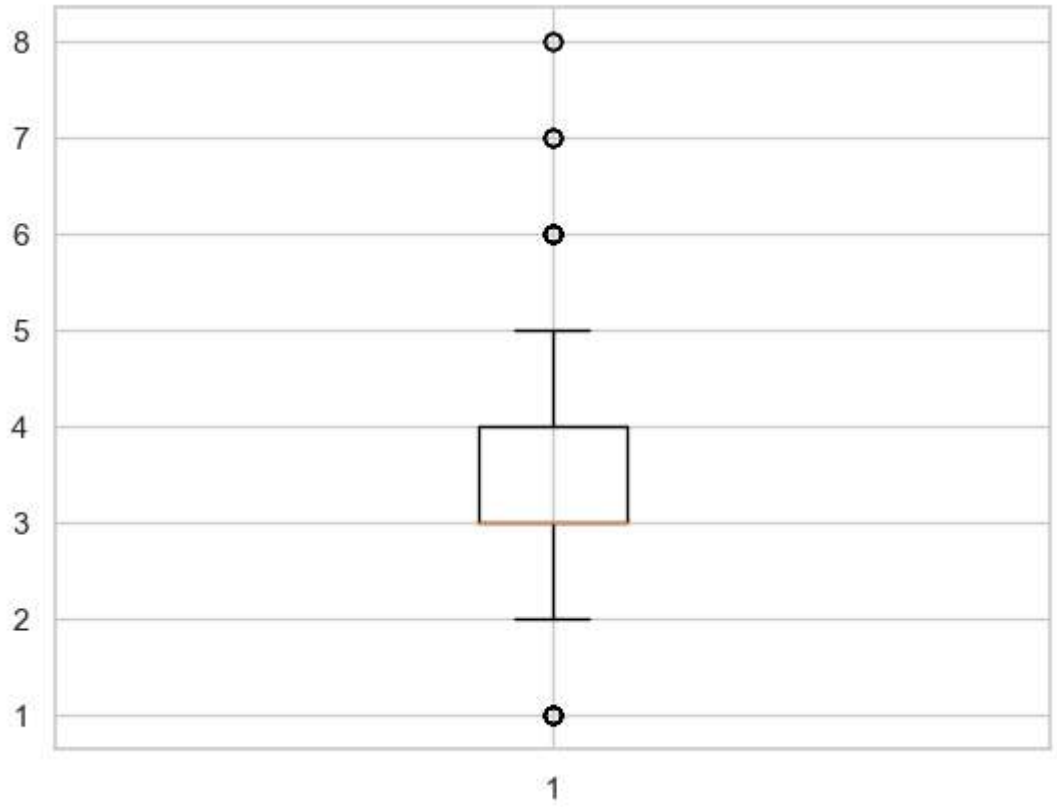
Histogram of number of Bedrooms

```
197 plt.hist(data['BEDS'], bins= [0,1,2,3,4,5,6,7,8])
    plt.title('Number of Bedrooms')
    plt.xlabel('Bedrooms')
    plt.show()
```



198
```
#View the distribution of the number of bedrooms to identify outliers
plt.boxplot(data['BEDS'])
```

198
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1b49a9a5190>,
  <matplotlib.lines.Line2D at 0x1b49a9a5430>],
 'caps': [<matplotlib.lines.Line2D at 0x1b49a9a56d0>,
  <matplotlib.lines.Line2D at 0x1b49a9a5970>],
 'boxes': [<matplotlib.lines.Line2D at 0x1b49a996eb0>],
 'medians': [<matplotlib.lines.Line2D at 0x1b49a9a5c10>],
 'fliers': [<matplotlib.lines.Line2D at 0x1b49a9a5eb0>],
 'means': []}
```

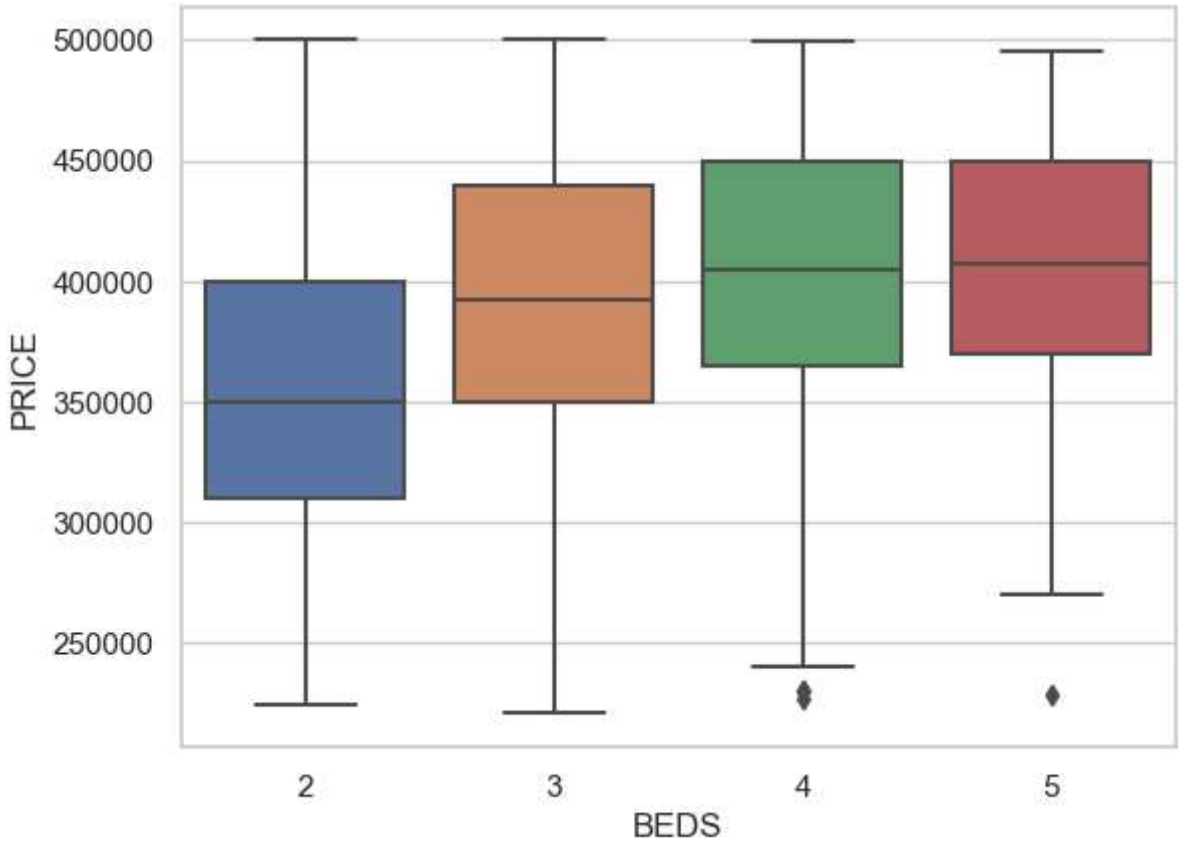



```
199 #remove the 1 bedroom and more than 5 bedrooms
    data = data[data.BEDS <6]
```

```
200 data = data[data.BEDS>1]
```

```
201 # view the beds by price to see relationship and identify outliers
    sn.set(style = 'whitegrid')
    sn.boxplot(x='BEDS',
               y='PRICE',
               data = data)
```

```
201 <AxesSubplot: xlabel='BEDS', ylabel='PRICE'>
```



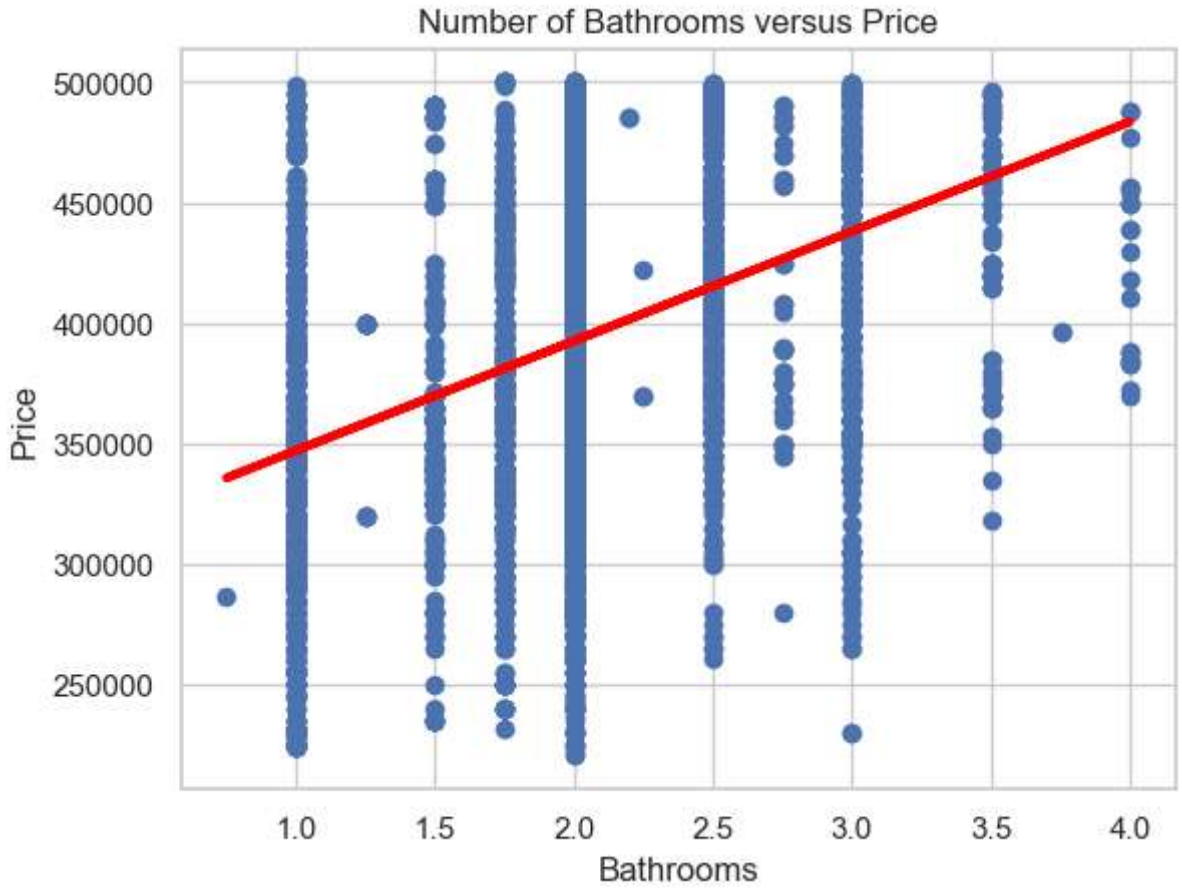
202
```
#Remove houses with 4+ bedrooms and sold for less than 250,000
dataremove = data[(data['BEDS']>=4) & (data['PRICE']< 250000)].index
data.drop(dataremove,inplace=True)
```

sn.set(style = 'whitegrid') sn.boxplot(x='BEDS', y='PRICE', data = data)

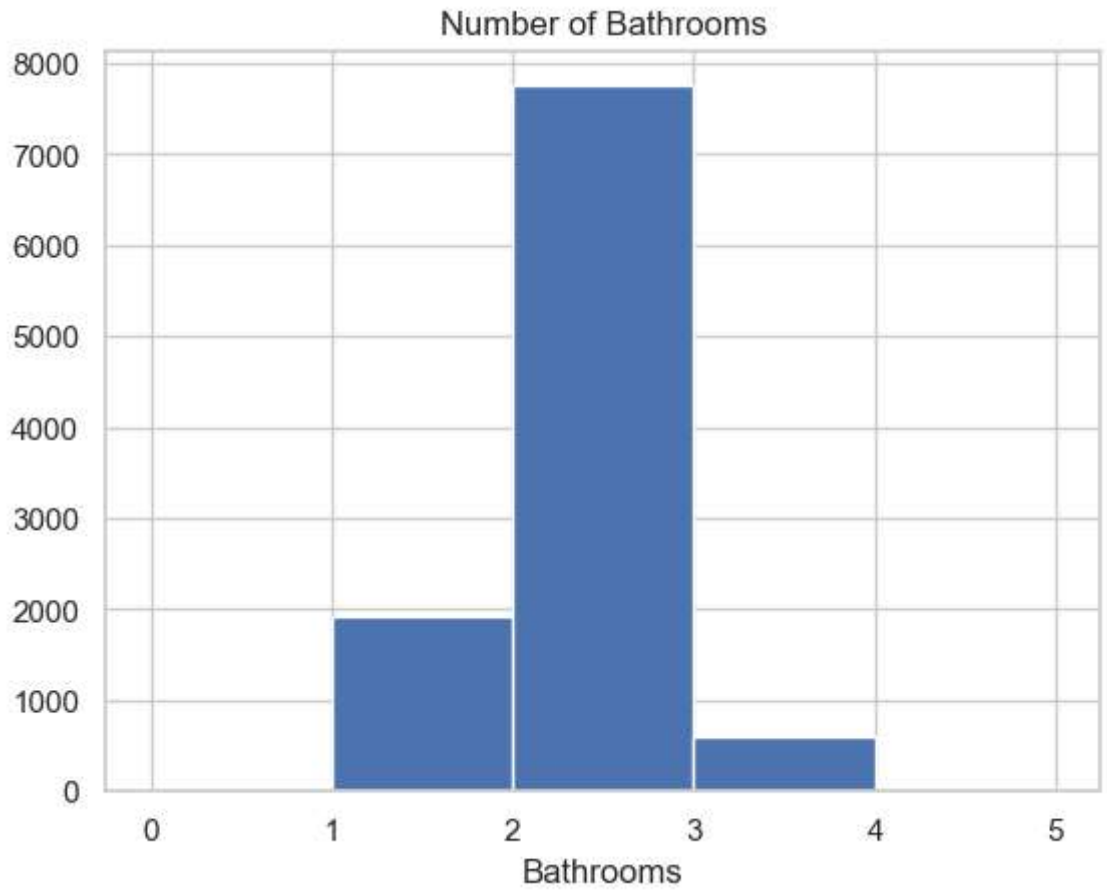
Compare number of bathrooms to price

203
```
y=data['PRICE']
x=data['BATHS']
plt.scatter(x,y)
plt.title('Number of Bathrooms versus Price')
plt.xlabel('Bathrooms')
plt.ylabel('Price')

z=np.polyfit(x,y,1)
p=np.poly1d(z)
plt.plot(x,p(x), color ='red', linewidth=3)
```

203
```
[<matplotlib.lines.Line2D at 0x1b49ace7220>]
```

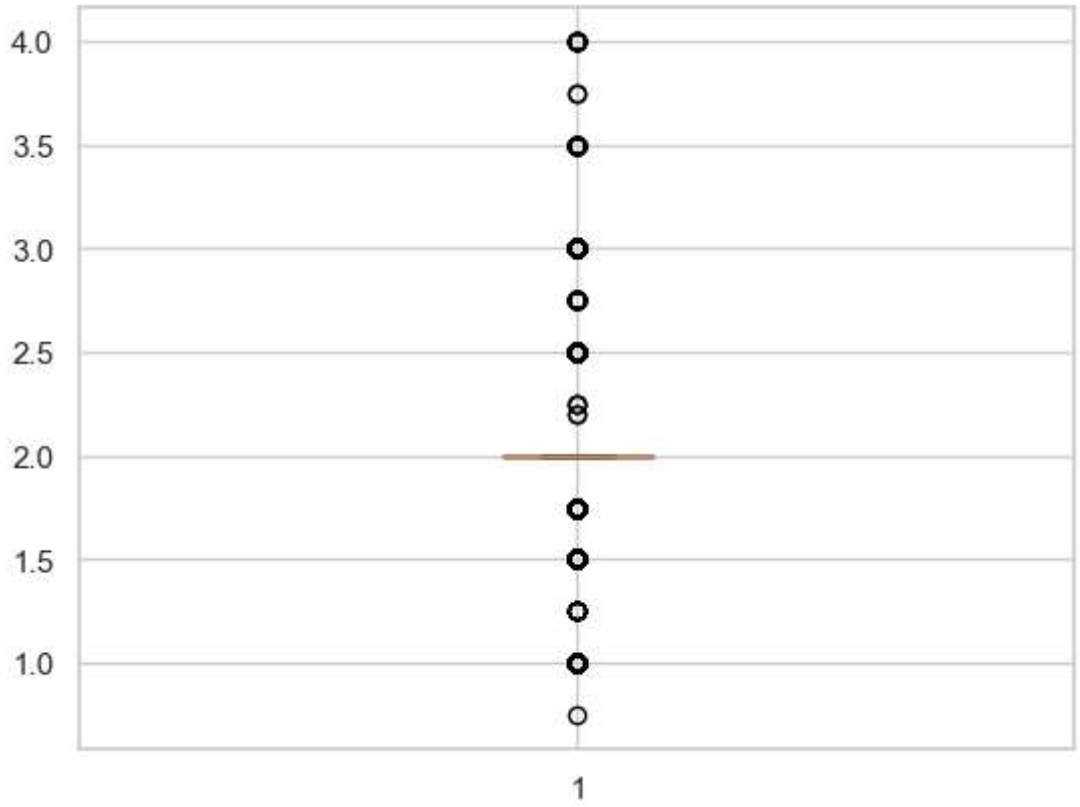



```
204 # view the distribution of bathrooms
    plt.hist(data['BATHS'], bins= [0,1,2,3,4,5])
    plt.title('Number of Bathrooms')
    plt.xlabel('Bathrooms')
    plt.show()
```



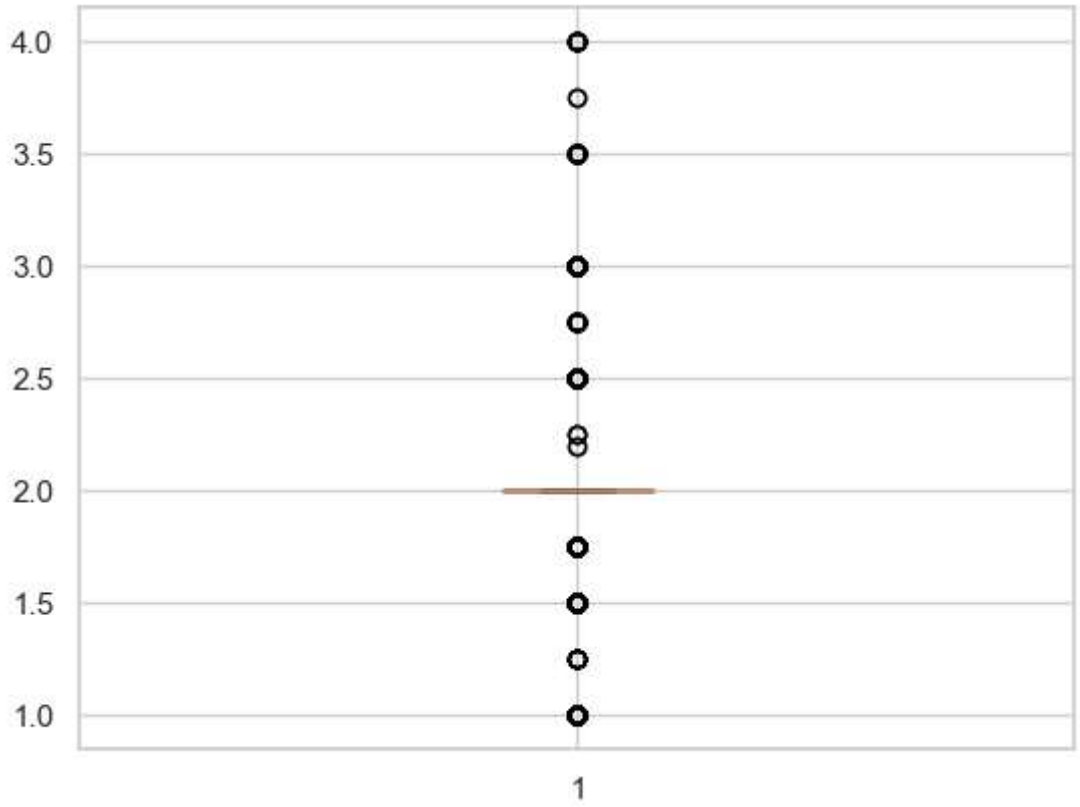
205
```
#identify outliers in the number of bathrooms
plt.boxplot(data['BATHS'])
```

205
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1b49ade2040>,
  <matplotlib.lines.Line2D at 0x1b49ade22e0>],
 'caps': [<matplotlib.lines.Line2D at 0x1b49ade2580>,
  <matplotlib.lines.Line2D at 0x1b49ade27c0>],
 'boxes': [<matplotlib.lines.Line2D at 0x1b49add2d60>],
 'medians': [<matplotlib.lines.Line2D at 0x1b49ade2a60>],
 'fliers': [<matplotlib.lines.Line2D at 0x1b49ade2d00>],
 'means': []}
```



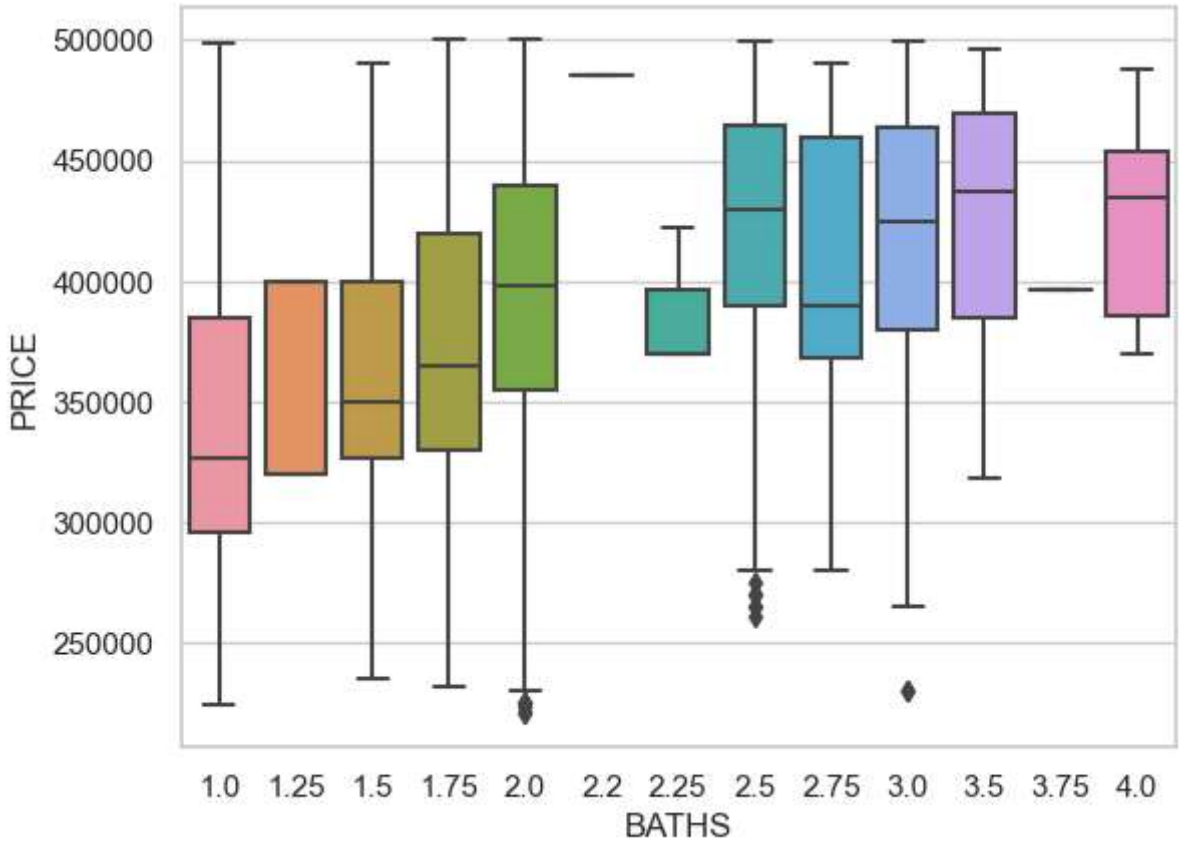
206
```
#remove less than 1 bathroom
data = data[data.BATHS >= 1]
plt.boxplot(data['BATHS'])
```

206
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1b49af51eb0>,
  <matplotlib.lines.Line2D at 0x1b49af65190>],
 'caps': [<matplotlib.lines.Line2D at 0x1b49af65430>,
  <matplotlib.lines.Line2D at 0x1b49af656d0>],
 'boxes': [<matplotlib.lines.Line2D at 0x1b49af51c10>],
 'medians': [<matplotlib.lines.Line2D at 0x1b49af65970>],
 'fliers': [<matplotlib.lines.Line2D at 0x1b49af65c10>],
 'means': []}
```

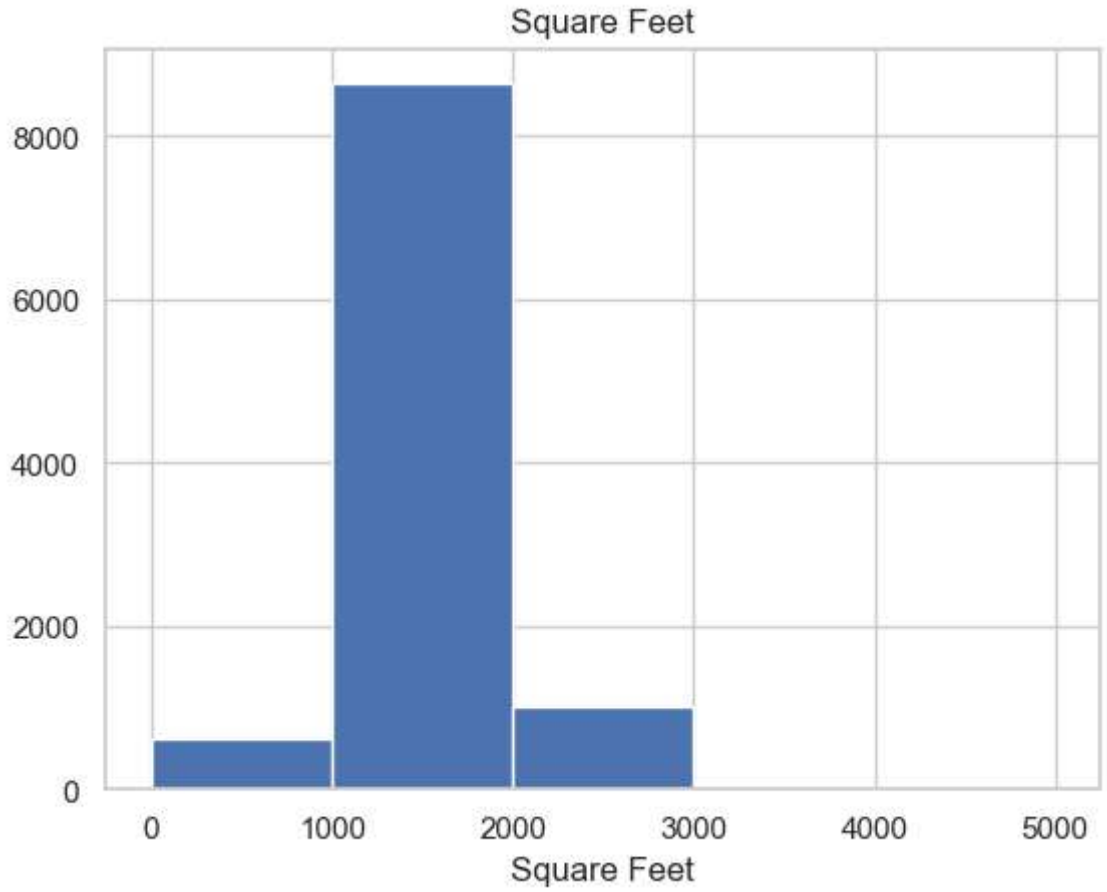



```
207 #view the relationship between bathrooms and price after removing outliers
    sn.set(style = 'whitegrid')
    sn.boxplot(x='BATHS',
               y='PRICE',
               data = data)
```

```
207 <AxesSubplot: xlabel='BATHS', ylabel='PRICE'>
```

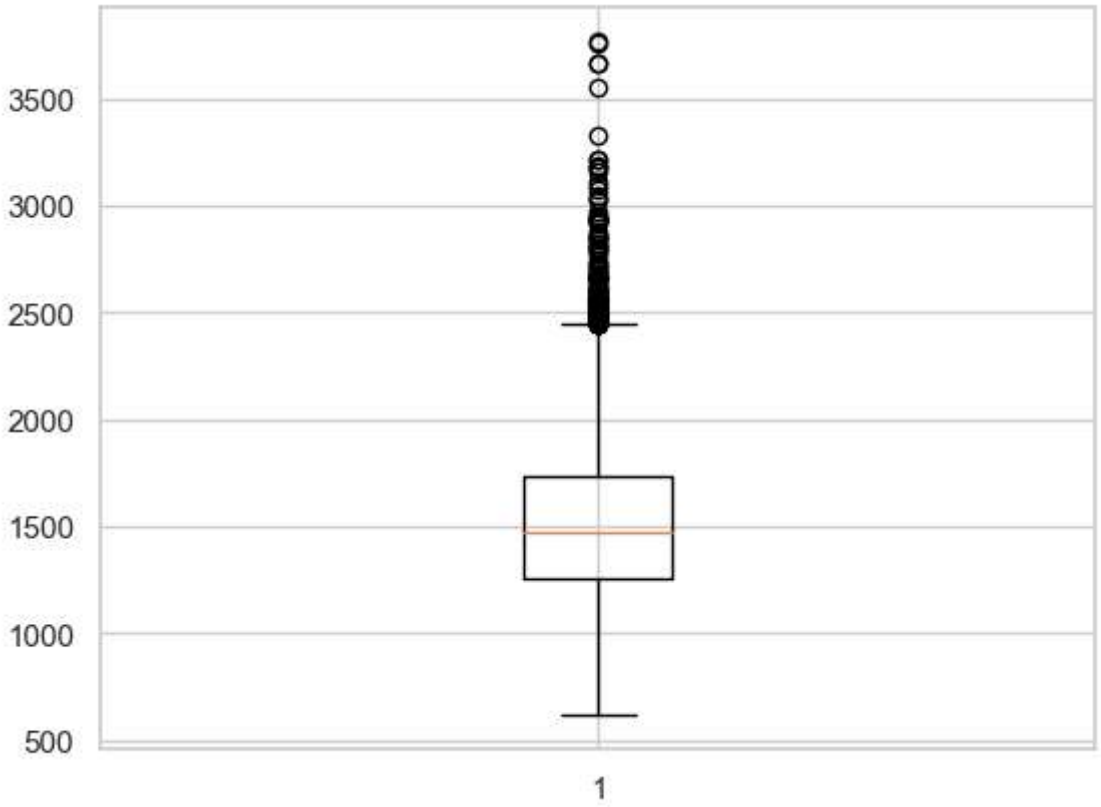


208
```
#view distribution of house square footage
plt.hist(data['SQFT'], bins= [0,1000,2000,3000,4000,5000])
plt.title('Square Feet')
plt.xlabel('Square Feet')
plt.show()
```



209
```
#view square footage as a boxplot
plt.boxplot(data['SQFT'])
```

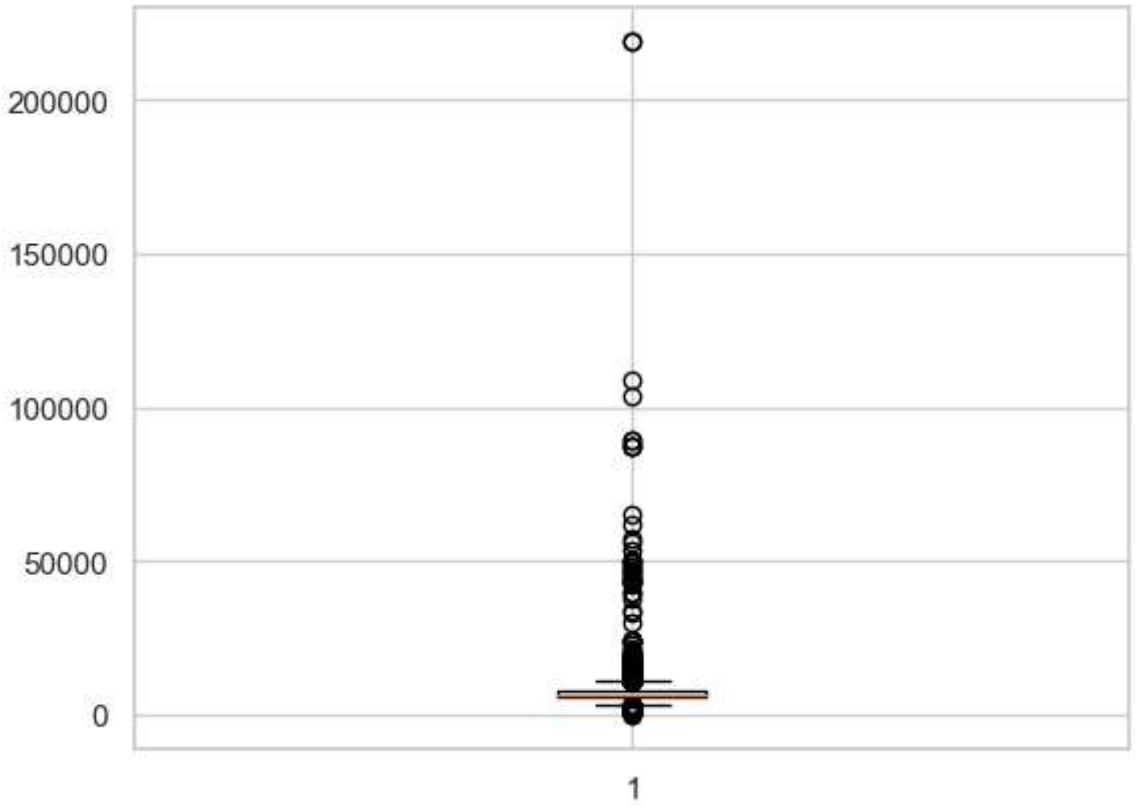
209
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1b49b2c2640>,
  <matplotlib.lines.Line2D at 0x1b49b2c28e0>],
 'caps': [<matplotlib.lines.Line2D at 0x1b49b2c2b80>,
  <matplotlib.lines.Line2D at 0x1b49b2c2e20>],
 'boxes': [<matplotlib.lines.Line2D at 0x1b49b2c23a0>],
 'medians': [<matplotlib.lines.Line2D at 0x1b49b2d2130>],
 'fliers': [<matplotlib.lines.Line2D at 0x1b49b2d23d0>],
 'means': []}
```



Explore Lot Size

```
210 #view lot size in a boxplot to identify outliers
    plt.boxplot(data['LOTSIZE'])
```

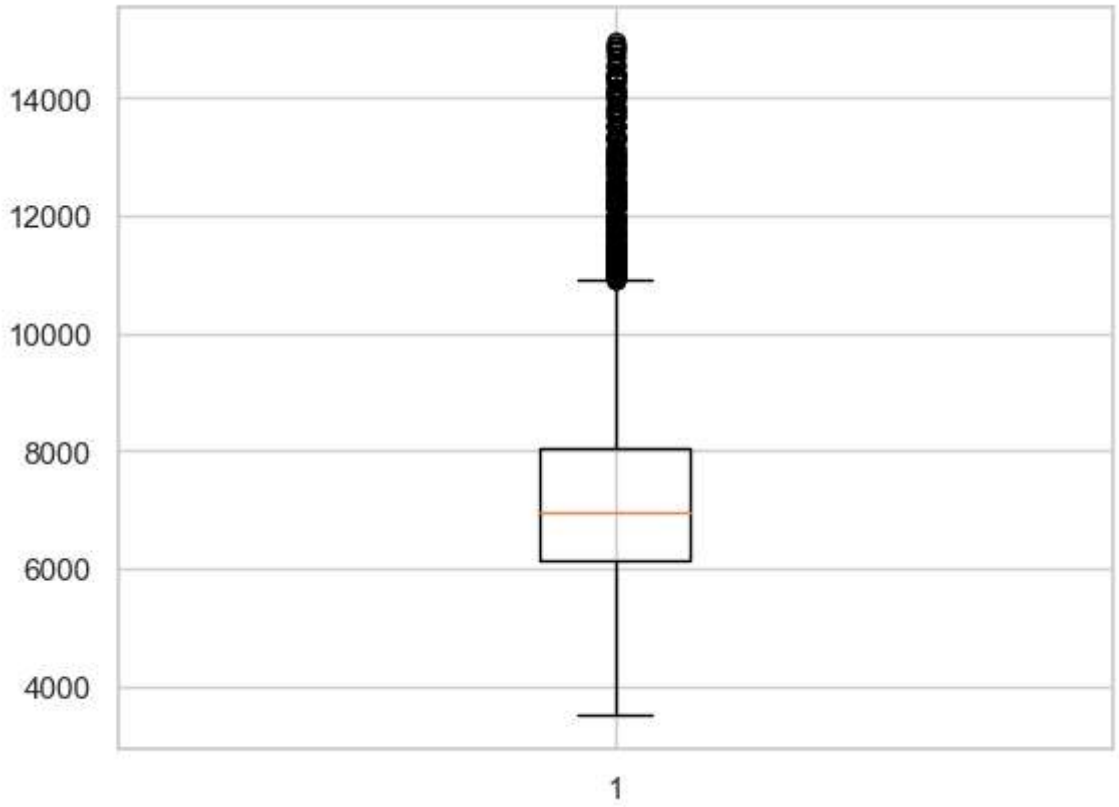
```
210 {'whiskers': [<matplotlib.lines.Line2D at 0x1b49b3357c0>,
      <matplotlib.lines.Line2D at 0x1b49b335a60>],
     'caps': [<matplotlib.lines.Line2D at 0x1b49b335d00>,
      <matplotlib.lines.Line2D at 0x1b49b335fa0>],
     'boxes': [<matplotlib.lines.Line2D at 0x1b49b335520>],
     'medians': [<matplotlib.lines.Line2D at 0x1b49b342280>],
     'fliers': [<matplotlib.lines.Line2D at 0x1b49b342520>],
     'means': []}
```



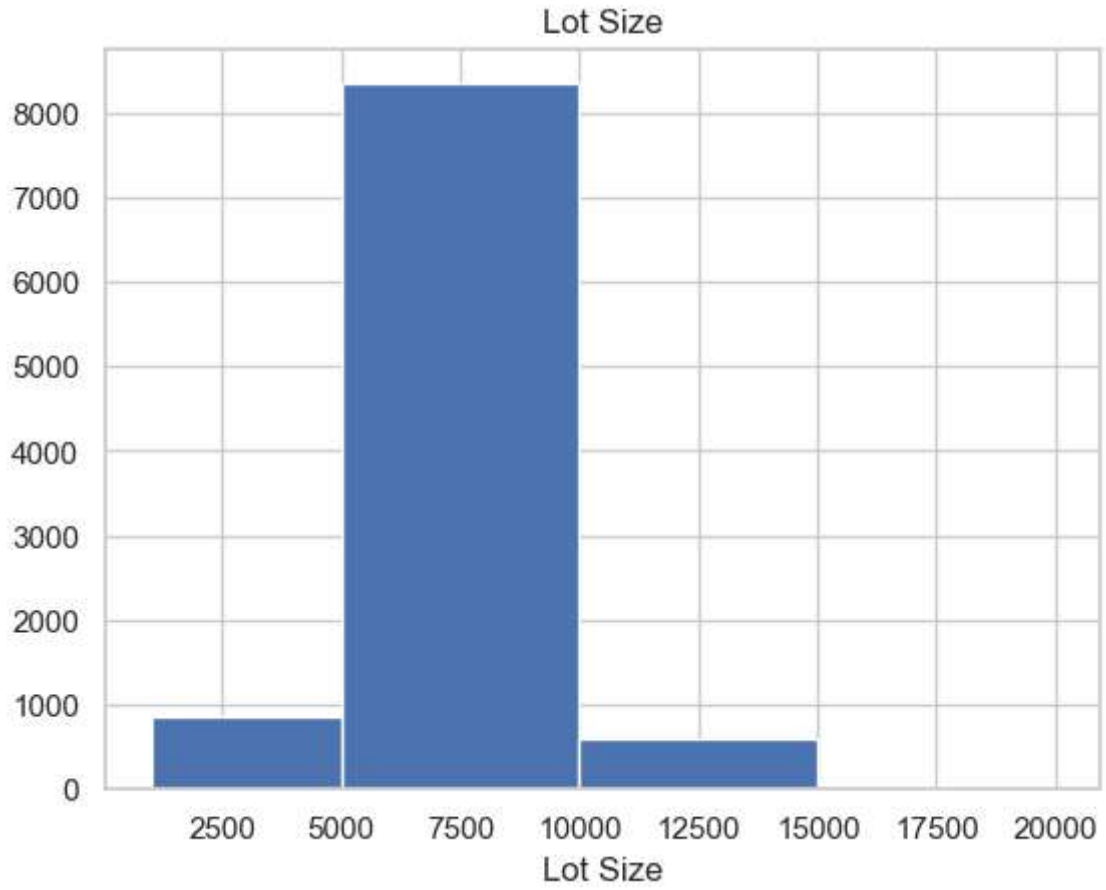
211
```
#Remove greater than 15,000sq ft and less than 3500 as unrealistic for price range
data = data[data.LOTSIZE < 15000]
data = data[data.LOTSIZE > 3500]
```

212
```
#view lotsize as a boxplot after removing outliers identified above
plt.boxplot(data['LOTSIZE'])
```

212
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1b4a50d6d90>,
  <matplotlib.lines.Line2D at 0x1b4a50e6070>],
 'caps': [<matplotlib.lines.Line2D at 0x1b4a50e6340>,
  <matplotlib.lines.Line2D at 0x1b4a50e65e0>],
 'boxes': [<matplotlib.lines.Line2D at 0x1b4a50d6af0>],
 'medians': [<matplotlib.lines.Line2D at 0x1b4a50e6880>],
 'fliers': [<matplotlib.lines.Line2D at 0x1b4a50e6b20>],
 'means': []}
```

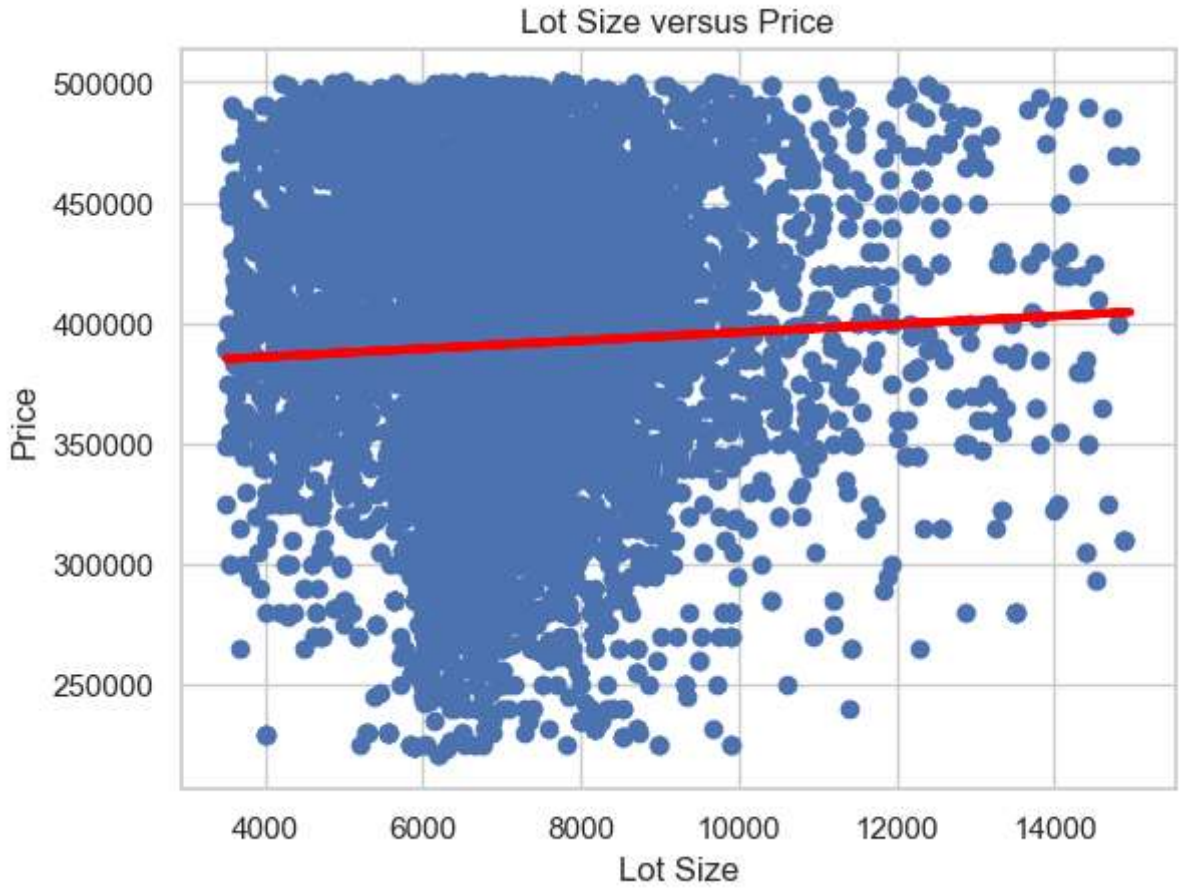



```
213 #view the distribution of houses by lotsize binning at 5000 sq ft intervals
    plt.hist(data['LOTSIZE'], bins= [1000,5000,10000,15000,20000])
    plt.title('Lot Size')
    plt.xlabel('Lot Size')
    plt.show()
```



214
```
y=data['PRICE']
x=data['LOTSIZE']
plt.scatter(x,y)
plt.title('Lot Size versus Price')
plt.xlabel('Lot Size')
plt.ylabel('Price')

z=np.polyfit(x,y,1)
p=np.poly1d(z)
plt.plot(x,p(x), color ='red', linewidth=3)
```

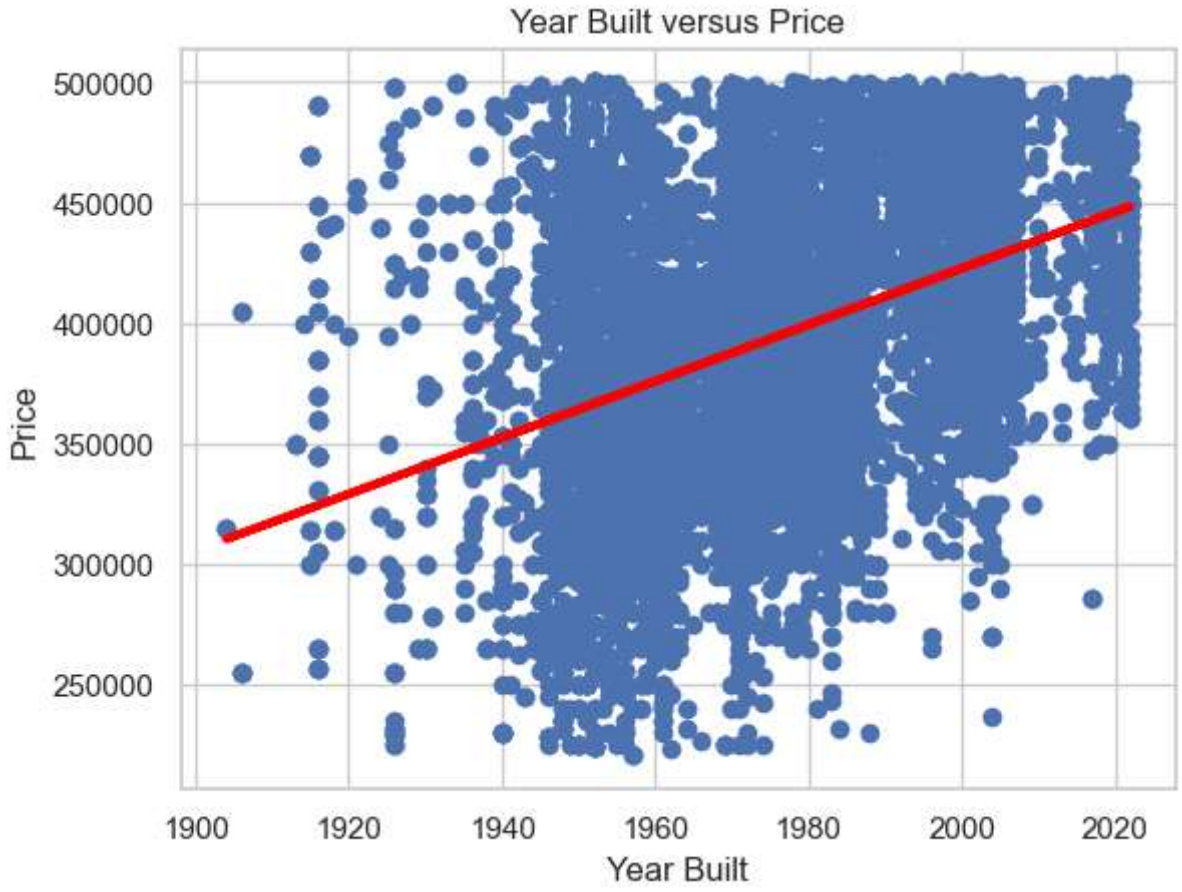
214
```
[<matplotlib.lines.Line2D at 0x1b4d2716760>]
```



Explore Year Built

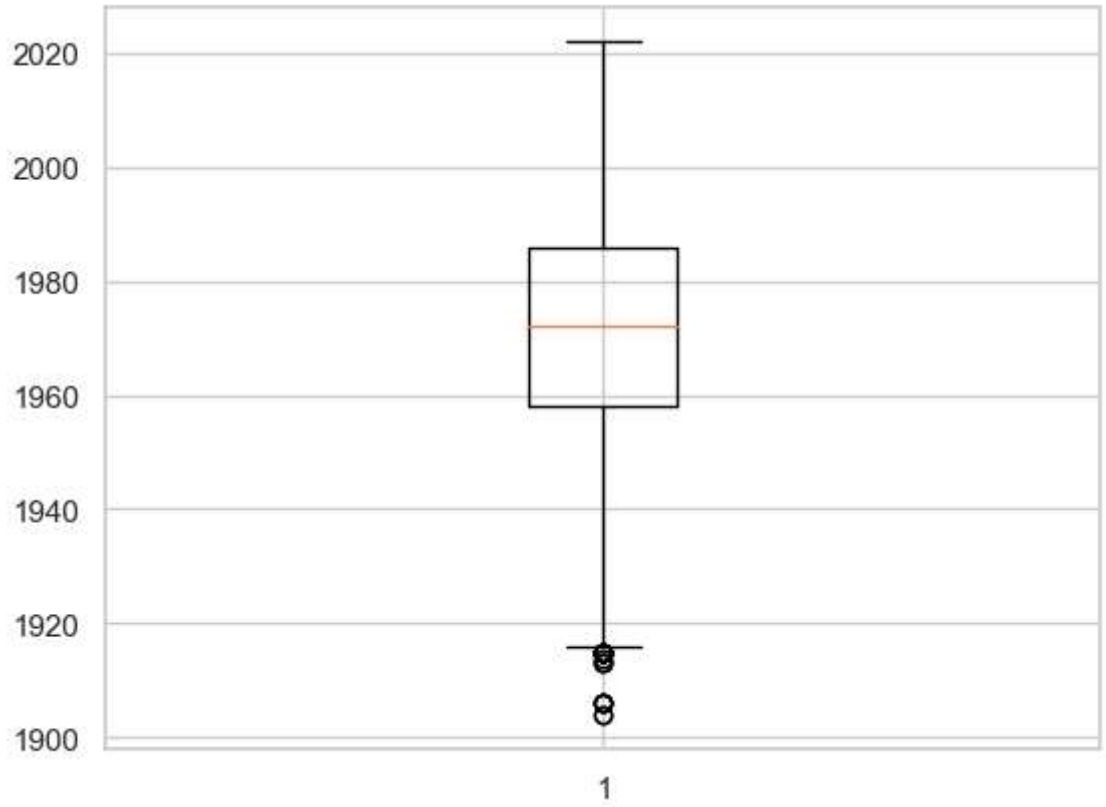
215
```
y=data['PRICE']
x=data['YEARBUILT']
plt.scatter(x,y)
plt.title('Year Built versus Price')
plt.xlabel('Year Built')
plt.ylabel('Price')

z=np.polyfit(x,y,1)
p=np.poly1d(z)
plt.plot(x,p(x), color ='red', linewidth=3)
```

215
```
[<matplotlib.lines.Line2D at 0x1b4d38c6940>]
```



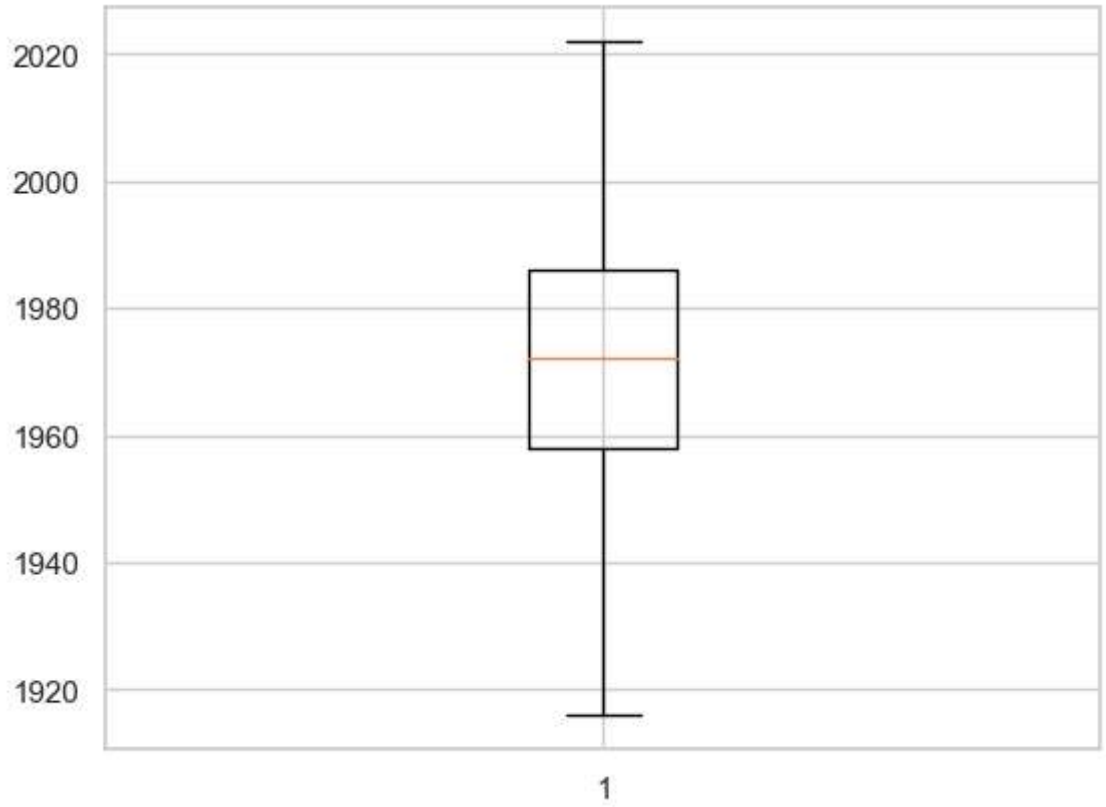
216
```
#view houses by year built to identify outliers
plt.boxplot(data['YEARBUILT'])
```

216
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1b4d37c8220>,
  <matplotlib.lines.Line2D at 0x1b4d37c84c0>],
 'caps': [<matplotlib.lines.Line2D at 0x1b4d37c8760>,
  <matplotlib.lines.Line2D at 0x1b4d37c88b0>],
 'boxes': [<matplotlib.lines.Line2D at 0x1b4d37b9f40>],
 'medians': [<matplotlib.lines.Line2D at 0x1b4d37c8b80>],
 'fliers': [<matplotlib.lines.Line2D at 0x1b4d37c8e20>],
 'means': []}
```



217
```
#Remove anything built before 1915
data = data[data.YEARBUILT > 1915]
plt.boxplot(data['YEARBUILT'])
```

217
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1b49d737850>,
  <matplotlib.lines.Line2D at 0x1b49d737760>],
 'caps': [<matplotlib.lines.Line2D at 0x1b49d737130>,
  <matplotlib.lines.Line2D at 0x1b49d67b1c0>],
 'boxes': [<matplotlib.lines.Line2D at 0x1b4d1b7da90>],
 'medians': [<matplotlib.lines.Line2D at 0x1b4a4727430>],
 'fliers': [<matplotlib.lines.Line2D at 0x1b4a4cc9e50>],
 'means': []}
```



218
```
#Remove Sold Date, Address, City, State or Provice, MLS#, Latitude and Longitude

model_data = data[['ZIPCODE','PRICE','BEDS','BATHS','SQFT','LOTSIZE','YEARBUILT','POOL','HOA']]
```

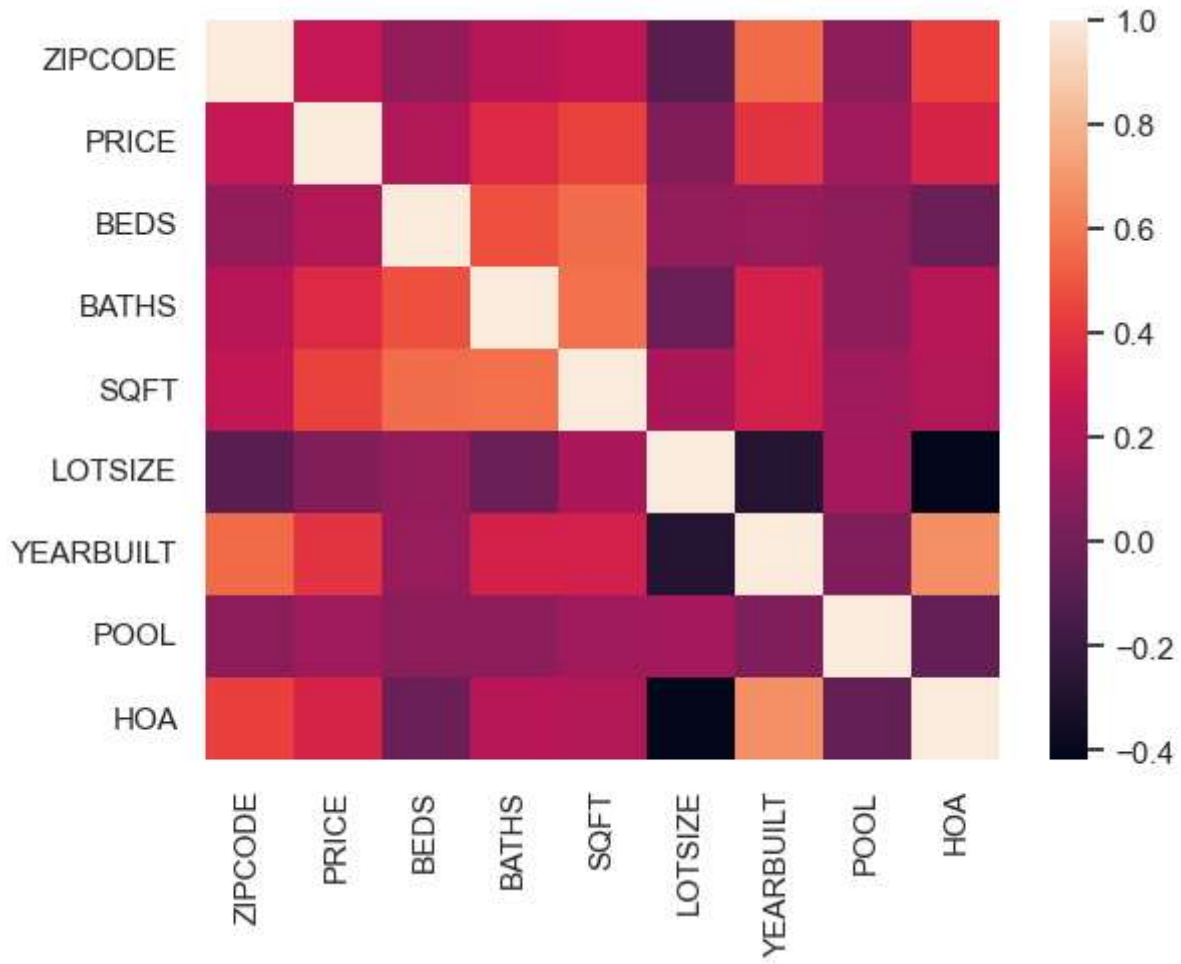
219
```
#view the first five rows of the model data and view the statistical data
model_data.head()
model_data.describe()
```

219

	ZIPCODE	PRICE	BEDS	BATHS	SQFT	LOTSIZE	YE
count	9788.000000	9788.000000	9788.000000	9788.000000	9788.000000	9788.000000	978
mean	85030.849816	391403.682162	3.232938	1.962725	1510.014508	7163.449734	197
std	16.180351	60204.959469	0.654691	0.453272	368.838202	1724.357887	20.
min	85003.000000	220500.000000	2.000000	1.000000	618.000000	3508.000000	191
25%	85019.000000	350000.000000	3.000000	2.000000	1256.750000	6125.000000	195
50%	85032.000000	395000.000000	3.000000	2.000000	1474.000000	6946.000000	197
75%	85041.000000	440000.000000	4.000000	2.000000	1734.000000	8031.250000	198
max	85087.000000	499999.000000	5.000000	4.000000	3770.000000	14963.000000	202

Heatmap of numerical values

```
220 ax = sn.heatmap(data = model_data.corr())
```




```
221 #correlation matrix of the model data
    corr_matrix = model_data.corr()
    corr_matrix['PRICE'].sort_values(ascending=False)
```

```
221 PRICE        1.000000
    SQFT         0.444500
    YEARBUILT    0.397496
    BATHS        0.361879
    HOA          0.338139
    ZIPCODE      0.262972
    BEDS         0.206007
    POOL         0.151689
    LOTSIZE      0.050020
    Name: PRICE, dtype: float64
```

```
222 #view data types of the model data
    model_data.dtypes
```

```
222 ZIPCODE        int64
    PRICE        float64
    BEDS           int32
    BATHS        float64
    SQFT           int64
    LOTSIZE        int64
    YEARBUILT      int64
    POOL           int32
```

```
HOA            int32
dtype: object
```

Build Models

```
223 #build the datasets removing the dependent variable PRICE from the set of independent variables
    X = model_data.drop('PRICE', axis = 1)
    y = model_data['PRICE']
```

```
224 #use one hot encoding on the feature variables

    encoder = OneHotEncoder(categories="auto", handle_unknown="ignore",sparse=False)
    X=encoder.fit_transform(X)
```

```
225 #scale the feature variables
    scaler = StandardScaler()
    X=scaler.fit_transform(X)
    X
```

```
225 array([[-0.05544728, -0.02860063,  7.20472818, ...,  2.30804781,
            0.53086999, -0.53086999],
           [-0.05544728, -0.02860063, -0.13879774, ...,  2.30804781,
            0.53086999, -0.53086999],
           [-0.05544728, -0.02860063, -0.13879774, ...,  2.30804781,
            0.53086999, -0.53086999],
           ...,
           [-0.05544728, -0.02860063, -0.13879774, ..., -0.43326659,
            -1.88370038,  1.88370038],
           [-0.05544728, -0.02860063, -0.13879774, ..., -0.43326659,
            -1.88370038,  1.88370038],
           [-0.05544728, -0.02860063, -0.13879774, ..., -0.43326659,
            -1.88370038,  1.88370038]])
```

```
226 #build the test and train datasets using 80% of the dataset for training and 20% for testing
    X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_state=4)
```

After testing various models including Linear Regression, XGBoost Regression, KNeighbor Regression, Random Forest Regression and ANN, the model with the highest R^2 score and the lowest MAE was the Random Forest Regression model. After selecting the Random Forest Regessor model, GridSearchCV was used to find the best parameters.

```
227 #Using GridSearchCV found the best parameters for the random forest model
    RandFor = RandomForestRegressor(bootstrap=False, max_features=3, n_estimators=300, random_state=4)
    RandFor.fit(X_train,y_train)

    y_pred = RandFor.predict(X_test)
```

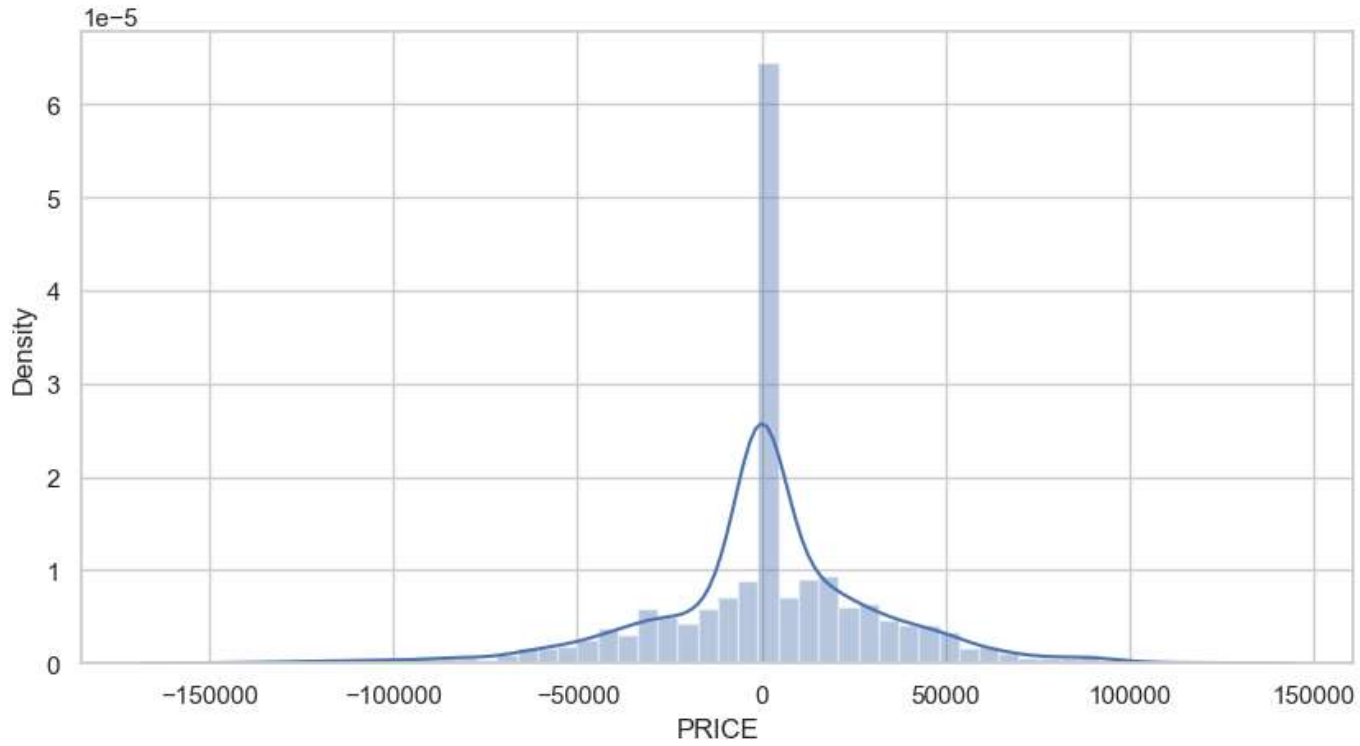
```
251 print('MAE: ', mean_absolute_error(y_test,y_pred))
    print('MSE: ', mean_squared_error(y_test,y_pred))
    print('R2: ', r2_score(y_test,y_pred))
    print('VarScore: ',explained_variance_score(y_test,y_pred) )
    print('RMSE: ', np.sqrt(mean_squared_error(y_test,y_pred)))
```

```
MAE:  20136.20126530149
MSE:  986204738.5211413
R2:  0.7296995053618887
VarScore:  0.730118413636752
RMSE:  31403.896868400603
score:  0.7296995053618887
```

229
```
fig=plt.figure(figsize=(10,5))
residuals = (y_test - y_pred)
sn.distplot(residuals)
```

```
C:\Users\belbi\AppData\Local\Temp\ipykernel_90980\1357820327.py:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sn.distplot(residuals)
```

229
```
<AxesSubplot: xlabel='PRICE', ylabel='Density'>
```



233
```
#change working directory
os.chdir(r"C:\Users\belbi\PycharmProject\Capstone")
```

234
```
## to test the model

testdata =pd.read_excel(r'C:\Users\belbi\PycharmProject\Capstone\testdata.xlsx',sheet_name='Sheet1')
datatest = encoder.transform(testdata)
```

235
```
datatest = encoder.transform(testdata)
```

```
datatest = scaler.transform(datatest)
prediction = RandFor.predict(datatest)
prediction
```

```
array([411074.33      , 422146.5       , 440055.76666667])
```

Save trained model

```
# pickle file to be used share model data with python pages used for Streamlit app

model = {"model": RandFor}
with open('trained_model.pkl','wb') as file:
    pickle.dump(model,file)
```

```
with open('trained_model.pkl','rb')as file:
    model = pickle.load(file)
```

```
# pickle file to be used to share encoder with python pages used for Streamlit app

with open('encoder.pkl' ,'wb') as f:
    pickle.dump(encoder,f)
```

```
with open('encoder.pkl','rb') as f:
    encoder = pickle.load(f)
```

```
# pickle file to be used to share standard scaler with python pages used for Streamlit app

with open('scaler.pkl' ,'wb') as f:
    pickle.dump(scaler,f)
```

```
with open('scaler.pkl','rb') as f:
    scaler=pickle.load(f)
```

```

# pickle file to be used to share data with python pages used for Streamlit app

with open('data.pkl','wb') as f:
    pickle.dump(data,f)
```

```
with open('data.pkl','rb') as f:
    data=pickle.load(f)
```

```
# pickle file to be used to share data with python pages used for Streamlit app

with open('modeldata.pkl','wb') as f:
    pickle.dump(model_data,f)
```

```
with open('modeldata.pkl','rb') as f:
```

```
modeldata=pickle.load(f)
```