Import Libraries

1
```
# 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
```

2
```
#set working directory to where data is stored
os.chdir(r"C:\Users\belbi\OneDrive\Documents\GCU\Capstone\Data\All Data")
```

3
```
#get filenames of all datafiles
csv_files = glob.glob('*.{}'.format('csv'))
csv_files
```

3
```
['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',
 'redfin_2023-01-09-17-56-29.csv',
 'redfin_2023-01-09-18-01-10.csv',
 'redfin_2023-01-09-18-01-53.csv',
 'redfin_2023-01-09-18-02-33.csv',
 'redfin_2023-01-09-18-04-37.csv',
 'redfin_2023-01-09-18-05-17.csv',
 'redfin_2023-01-09-18-06-07.csv',
 'redfin_2023-01-09-18-07-13.csv',
 'redfin_2023-01-09-18-07-50.csv',
 'redfin_2023-01-09-18-08-32.csv',
 'redfin_2023-01-09-18-09-02.csv']
```

4

```
#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_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
```

```
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
```

```
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1639621125.py:5: FutureWarning: The frame.append metho
  df_append = df_append.append(df_temp, ignore_index = True)
```

4

	SALE TYPE	SOLD DATE	PROPERTY TYPE	ADDRESS	CITY	STATE OR PROVINCE	ZIP OR POSTAL CODE	PRICE

	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
...	...	...	...	...	...	...	...	...
15952	PAST SALE	October-31-2022	Single Family Residential	41338 N PARKER Ln	Anthem	AZ	85086.0	445000.0
15953	PAST SALE	December-12-2022	Single Family Residential	1646 W OWENS Way	Anthem	AZ	85086.0	375000.0
15954	PAST SALE	December-16-2022	Single Family Residential	39807 N CROSS TIMBERS Way	Anthem	AZ	85086.0	430000.0
15955	PAST SALE	NaN	Single Family Residential	1811 W Jomax Rd	Phoenix	AZ	85085.0	453492.0
15956	PAST SALE	NaN	Single Family Residential	35638 N 10th St	Phoenix	AZ	85086.0	161000.0

15957 rows × 28 columns

```
5 df = df_append

#explore the first five rows of data
```

6
```
df.head()
```

6

	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

7
```
#view statistics on all columns in the data
df.describe(include='all')
```

7

	SALE TYPE	SOLD DATE	PROPERTY TYPE	ADDRESS	CITY	STATE OR PROVINCE	ZIP OR POSTAL CODE	
count	15957	12844	15957	15944	15944	15957	15944.000000	1.594
unique	1	348	7	11098	11	1	NaN	NaN
top	PAST SALE	February-28-2022	Single Family Residential	4115 E CORONADO Rd	Phoenix	AZ	NaN	NaN
freq	15957	121	15892	10	14879	15957	NaN	NaN
mean	NaN	NaN	NaN	NaN	NaN	NaN	85046.772579	3.775
std	NaN	NaN	NaN	NaN	NaN	NaN	69.586668	7.977
min	NaN	NaN	NaN	NaN	NaN	NaN	85003.000000	1.500

	SALE TYPE	SOLD DATE	PROPERTY TYPE	ADDRESS	CITY	STATE OR PROVINCE	ZIP OR POSTAL CODE	
25%	NaN	NaN	NaN	NaN	NaN	NaN	85019.000000	3.35(
50%	NaN	NaN	NaN	NaN	NaN	NaN	85032.000000	3.85(
75%	NaN	NaN	NaN	NaN	NaN	NaN	85042.000000	4.35(
max	NaN	NaN	NaN	NaN	NaN	NaN	85392.000000	1.825

11 rows × 28 columns

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

```
8  SALE TYPE                                                                          0
   SOLD DATE                                                                       3113
   PROPERTY TYPE                                                                      0
   ADDRESS                                                                           13
   CITY                                                                              13
   STATE OR PROVINCE                                                                  0
   ZIP OR POSTAL CODE                                                                13
   PRICE                                                                             16
   BEDS                                                                            3049
   BATHS                                                                             81
   LOCATION                                                                        3162
   SQUARE FEET                                                                       35
   LOT SIZE                                                                          21
   YEAR BUILT                                                                        35
   DAYS ON MARKET                                                                 15957
   $/SQUARE FEET                                                                     51
   HOA/MONTH                                                                      12587
   STATUS                                                                          3097
   NEXT OPEN HOUSE START TIME                                                     15957
   NEXT OPEN HOUSE END TIME                                                       15957
   URL (SEE https://www.redfin.com/buy-a-home/comparative-market-analysis FOR INFO ON PRICING)        0
   SOURCE                                                                          3097
   MLS#                                                                            3113
   FAVORITE                                                                           0
   INTERESTED                                                                         0
   LATITUDE                                                                           0
   LONGITUDE                                                                          0
   POOL                                                                               0
   dtype: int64
```

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

```
9  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
```

10
```
#convert the sold date to a date
date_string = df['SOLD DATE']
df['SOLDDATE'] = pd.to_datetime(date_string)
df.drop('SOLD DATE', axis = 1)
```

10

	SALE TYPE	PROPERTY TYPE	ADDRESS	CITY	STATE OR PROVINCE	ZIP OR POSTAL CODE	PRICE	BEDS	BA
0	PAST SALE	Single Family Residential	1846 E Willetta St	Phoenix	AZ	85006.0	210000.0	NaN	1.0
1	PAST SALE	Single Family Residential	1515 E Brill St	Phoenix	AZ	85006.0	360000.0	NaN	1.0
2	PAST SALE	Single Family Residential	1632 E Granada Rd	Phoenix	AZ	85006.0	245000.0	NaN	1.0
3	PAST SALE	Single Family Residential	1113 W PORTLAND St	Phoenix	AZ	85007.0	780000.0	4.0	2.0
4	PAST SALE	Single Family Residential	946 W Cocopah St	Phoenix	AZ	85007.0	258000.0	NaN	1.0
...	...	...	...	...	...	...	...	...	...
15952	PAST SALE	Single Family Residential	41338 N PARKER Ln	Anthem	AZ	85086.0	445000.0	3.0	2.5

	SALE TYPE	PROPERTY TYPE	ADDRESS	CITY	STATE OR PROVINCE	ZIP OR POSTAL CODE	PRICE	BEDS	BA
15953	PAST SALE	Single Family Residential	1646 W OWENS Way	Anthem	AZ	85086.0	375000.0	2.0	2.0
15954	PAST SALE	Single Family Residential	39807 N CROSS TIMBERS Way	Anthem	AZ	85086.0	430000.0	3.0	2.0
15955	PAST SALE	Single Family Residential	1811 W Jomax Rd	Phoenix	AZ	85085.0	453492.0	NaN	NaN
15956	PAST SALE	Single Family Residential	35638 N 10th St	Phoenix	AZ	85086.0	161000.0	NaN	2.0

15957 rows × 28 columns

11
```
#add weekly posted mortgage rate for each sale
df2 = (pd.read_excel(r"C:\Users\belbi\PycharmProject\Capstone\historicaldata.xlsx", sheet_name='Sheet1'))
```

12
```
df2.head()
```

12

	Day	Rate
0	2021-08-01	2.77
1	2021-08-02	2.77
2	2021-08-03	2.77
3	2021-08-04	2.77
4	2021-08-05	2.87

13
```
df2.dtypes
```

13
```
Day     datetime64[ns]
Rate           float64
dtype: object
```

14
```
left_join = (pd.merge(df,
                      df2,
                      left_on='SOLDDATE', right_on='Day',
```

```
                    how='left')
                .drop('Day',axis=1))
```

15
```
df = left_join
df = df.rename(columns = {'Rate':'RATE'})
```

16
```
df.head()
```

16

	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 × 30 columns

17
```
#removing data without MLS# as these most likely represent private sales
df = df.dropna(subset=['MLS#'],axis=0)
```

18
```
#after removing the data without MLS# determine how many columns still contain null values
df.isnull().sum()
```

18
```
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                    6
LOCATION                49
SQUARE FEET              0
LOT SIZE                 5
YEAR BUILT               0
```

```
DAYS ON MARKET                                                                  12844
$/SQUARE FEET                                                                       0
HOA/MONTH                                                                        9474
STATUS                                                                              0
NEXT OPEN HOUSE START TIME                                                      12844
NEXT OPEN HOUSE END TIME                                                        12844
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
RATE                                                                                0
dtype: int64
```

19
```
#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)
```

20
```
#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)
```

21
```
#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)
```

22
```
#view the dataframe after making the changes above
df
```

22

	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
9	PAST SALE	September-30-2022	Single Family Residential	810 S 3RD Ave	Phoenix	AZ	85003.0	614000.0

	SALE TYPE	SOLD DATE	PROPERTY TYPE	ADDRESS	CITY	STATE OR PROVINCE	ZIP OR POSTAL CODE	PRICE
...	...	...	...	...	...	...	...	...
15950	PAST SALE	December-6-2022	Single Family Residential	40759 N TRAILHEAD Way	Phoenix	AZ	85086.0	375000.0
15951	PAST SALE	December-30-2022	Single Family Residential	39768 N HIGH NOON Way	Phoenix	AZ	85086.0	460000.0
15952	PAST SALE	October-31-2022	Single Family Residential	41338 N PARKER Ln	Anthem	AZ	85086.0	445000.0
15953	PAST SALE	December-12-2022	Single Family Residential	1646 W OWENS Way	Anthem	AZ	85086.0	375000.0
15954	PAST SALE	December-16-2022	Single Family Residential	39807 N CROSS TIMBERS Way	Anthem	AZ	85086.0	430000.0

12833 rows × 31 columns

23
```
# 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'})
```

24
```
#limit the dataset to sold price of less than $500k
data = data[data.PRICE <500000]
```

25
```
#Remove zipcodes captured that are not Phoenix
data = data[data.ZIPCODE < 85099]
```

26
```
#Remove Duplicates and view shape of the data
data.duplicated().sum()
data.drop_duplicates
data.shape
```

26
```
(11936, 17)
```

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

```
27 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
   RATE                 0
   dtype: int64
```

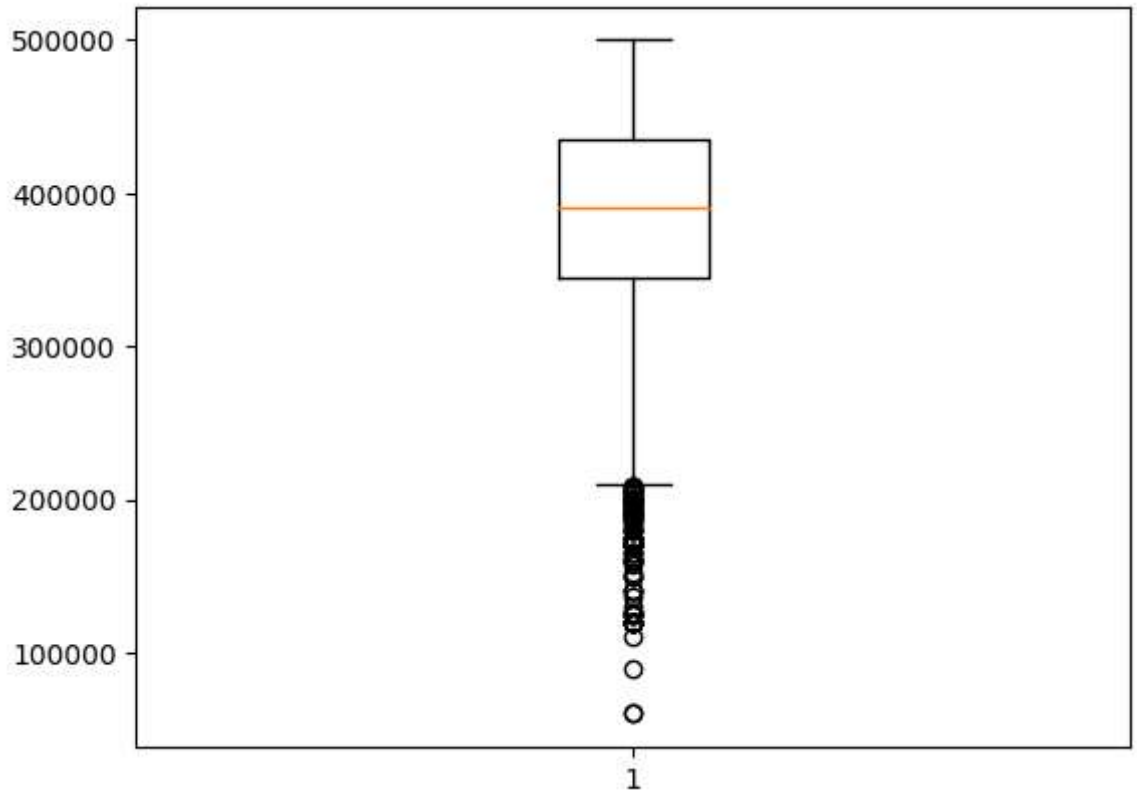
```
28 #covert zipcode to category data for one hot encoding
   data['ZIPCODE']=data['ZIPCODE'].astype('int64')
   data['YEARBUILT']=data['YEARBUILT'].astype('int64')
   data['SQFT']=data['SQFT'].astype('int64')
   data['LOTSIZE']=data['LOTSIZE'].astype('int64')
   data['BEDS']=data['BEDS'].astype('int32')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

Exploratory Data Analysis

Explore Price

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

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

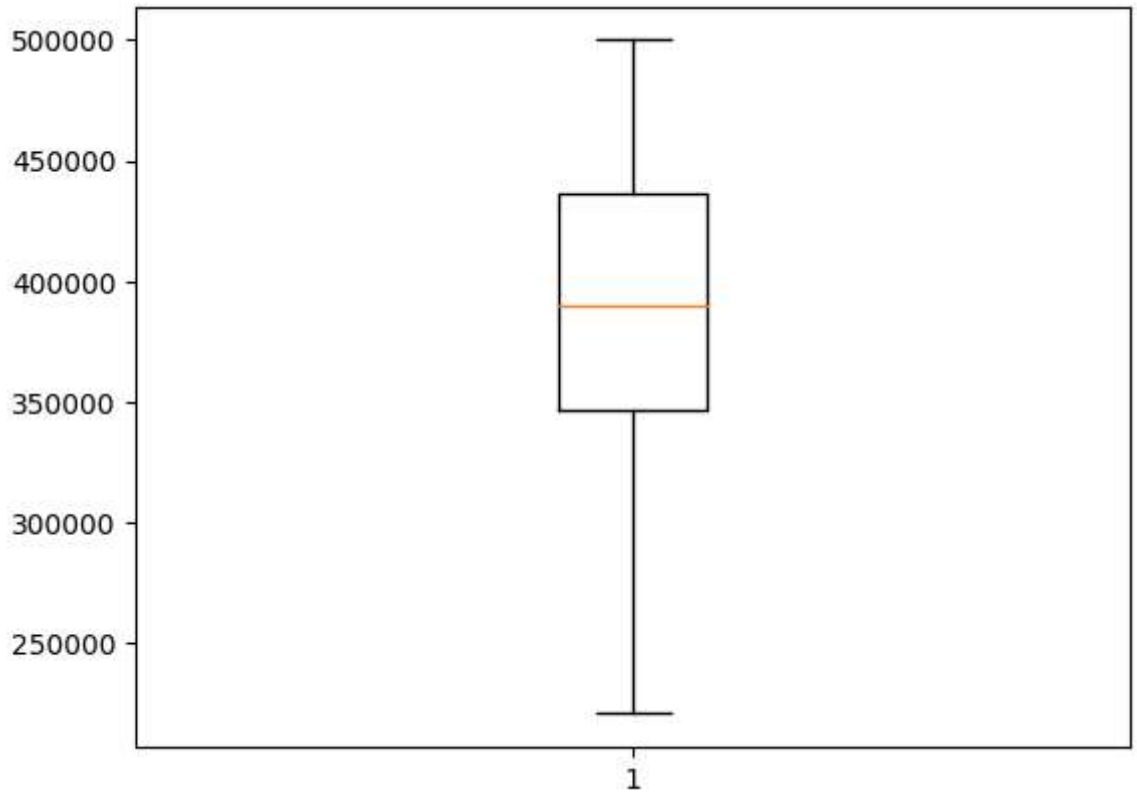
```
29 {'whiskers': [<matplotlib.lines.Line2D at 0x2f6b8f4a8b0>,
     <matplotlib.lines.Line2D at 0x2f6b8f4ab50>],
    'caps': [<matplotlib.lines.Line2D at 0x2f6b8f4adf0>,
     <matplotlib.lines.Line2D at 0x2f6ba0700d0>],
    'boxes': [<matplotlib.lines.Line2D at 0x2f6b8f4a610>],
    'medians': [<matplotlib.lines.Line2D at 0x2f6ba070370>],
    'fliers': [<matplotlib.lines.Line2D at 0x2f6ba070610>],
    'means': []}
```




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

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

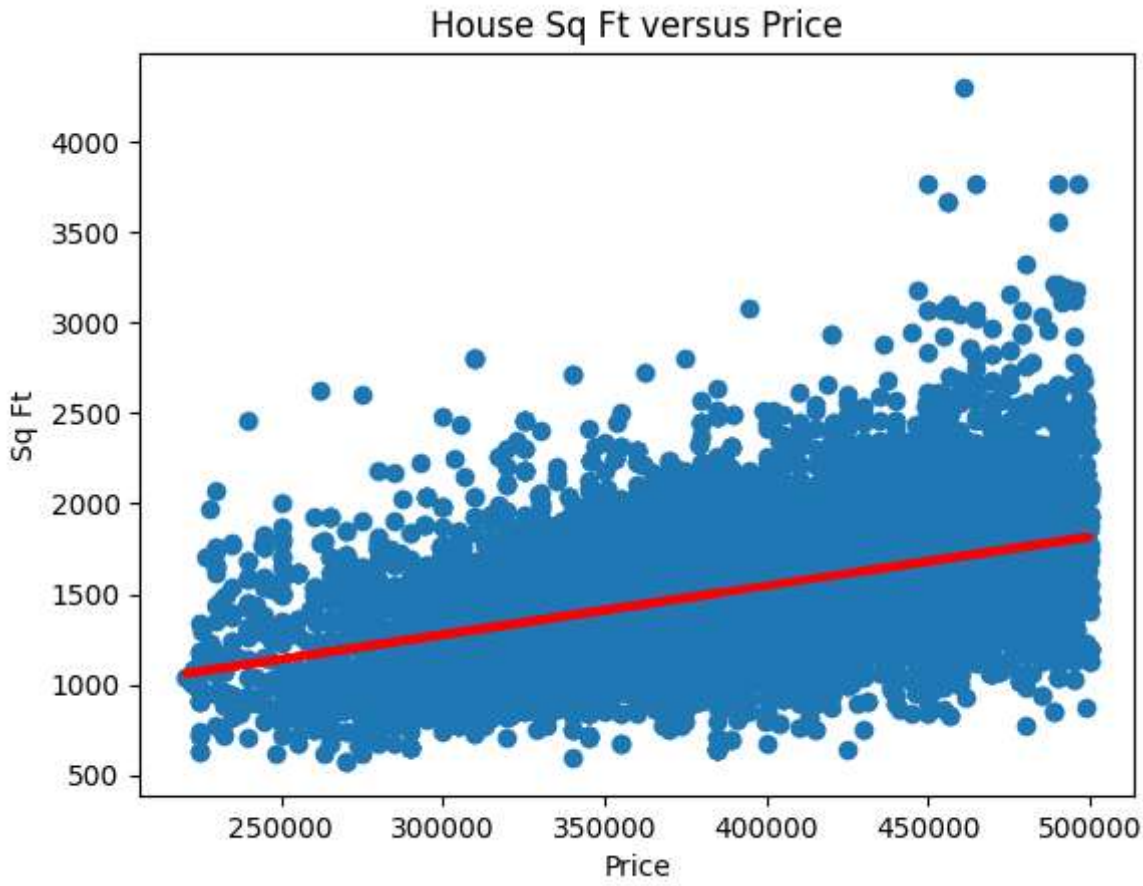
```
31  {'whiskers': [<matplotlib.lines.Line2D at 0x2f6ba4fc9d0>,
      <matplotlib.lines.Line2D at 0x2f6ba4fcc70>],
     'caps': [<matplotlib.lines.Line2D at 0x2f6ba4fcf10>,
      <matplotlib.lines.Line2D at 0x2f6ba5091f0>],
     'boxes': [<matplotlib.lines.Line2D at 0x2f6ba4fc730>],
     'medians': [<matplotlib.lines.Line2D at 0x2f6ba509490>],
     'fliers': [<matplotlib.lines.Line2D at 0x2f6ba509730>],
     'means': []}
```



Compare house size to price

```
32  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)
```

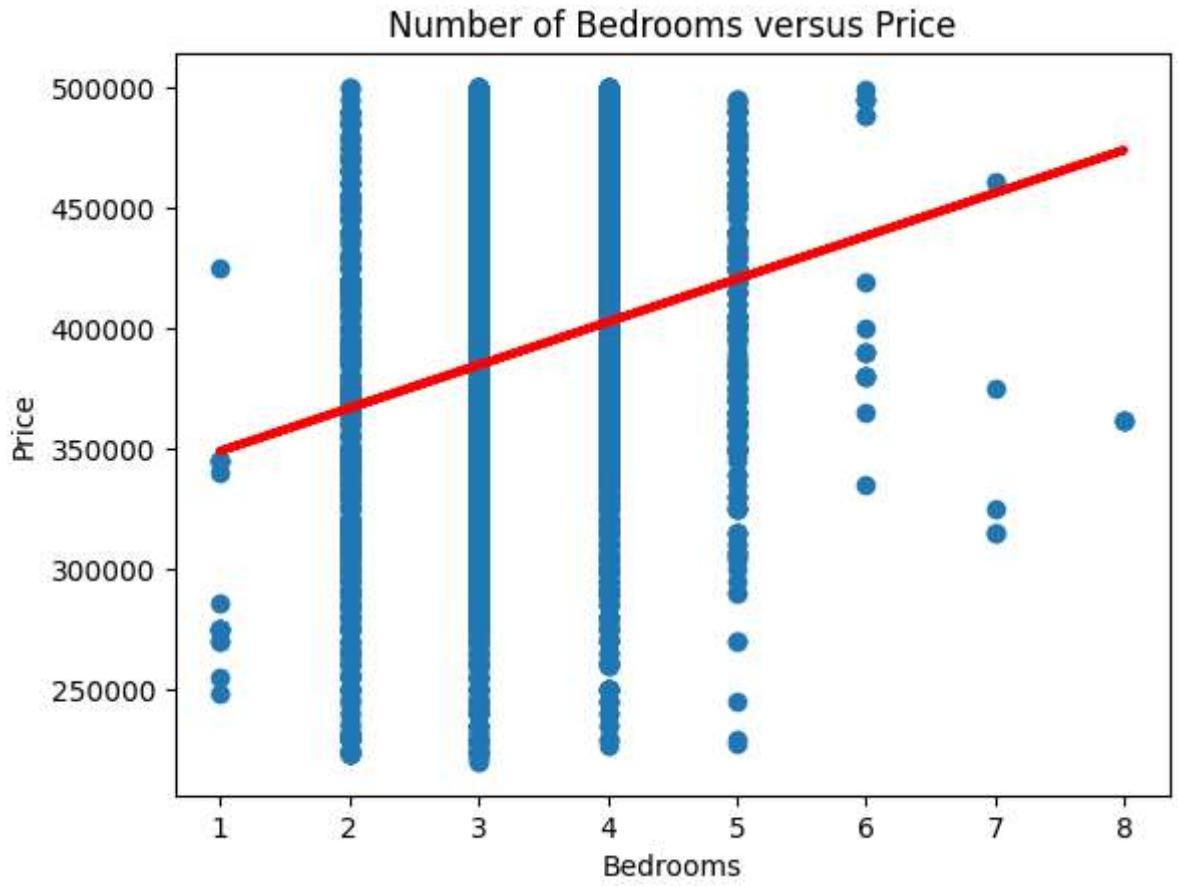
```
32  [<matplotlib.lines.Line2D at 0x2f6ba56e280>]
```



Compare number of bedrooms to price

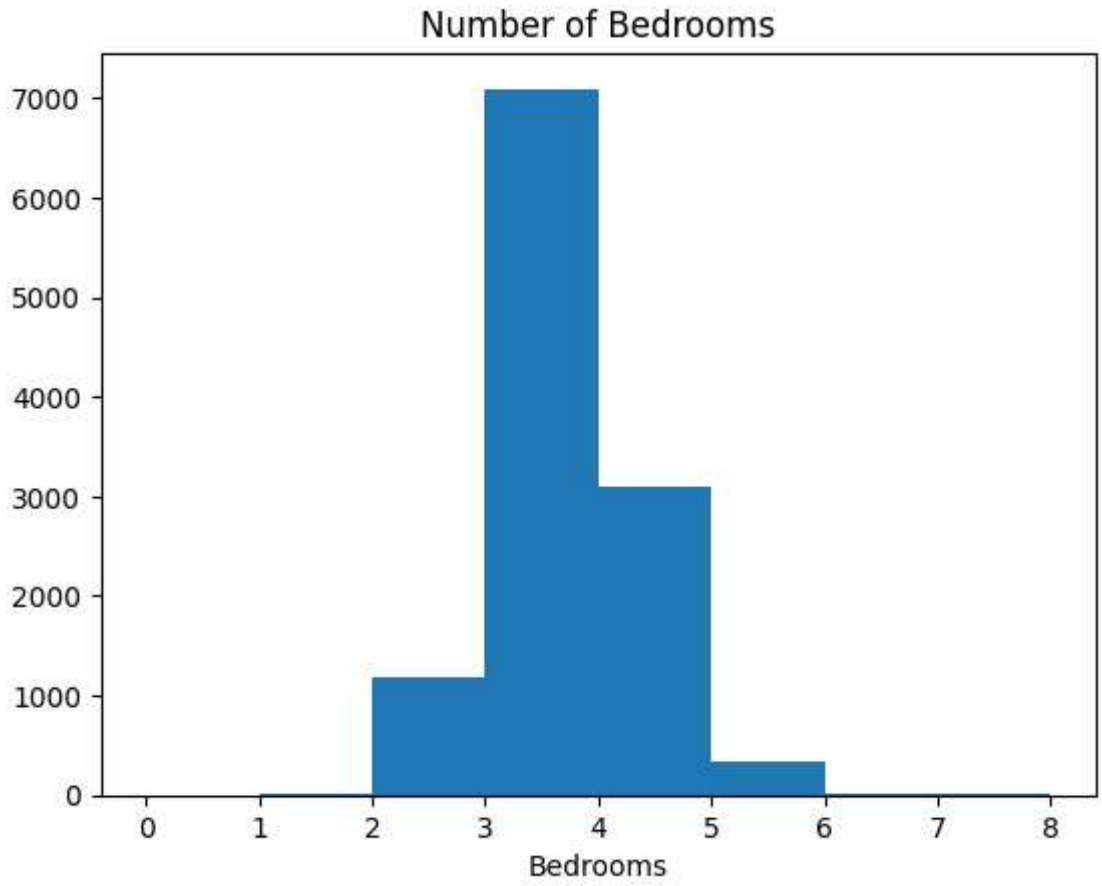
```
33  y=data['PRICE']
    x=data['BEDS']
    plt.scatter(x,y)
    plt.title('Number of Bedrooms versus Price')
    plt.xlabel('Bedrooms')
    plt.ylabel('Price')

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

```
33  [<matplotlib.lines.Line2D at 0x2f6ba600700>]
```



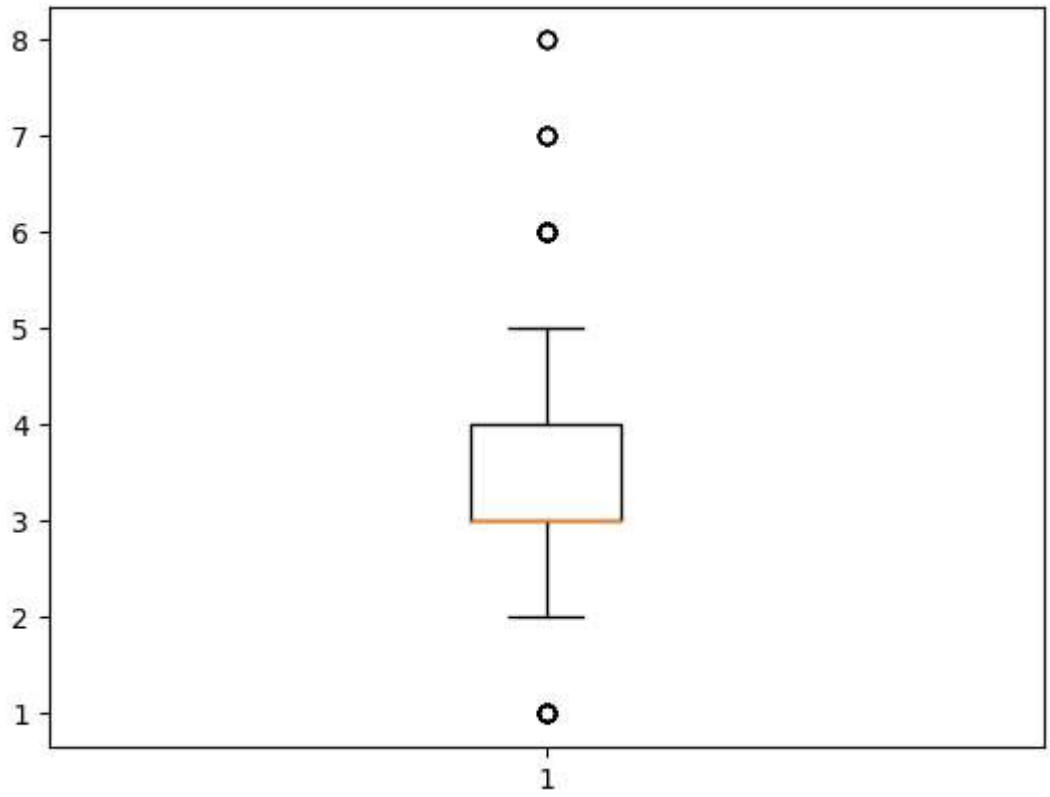
Histogram of number of Bedrooms

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



35
```
#View the distribution of the number of bedrooms to identify outliers
plt.boxplot(data['BEDS'])
```

35
```
{'whiskers': [<matplotlib.lines.Line2D at 0x2f6bc881f10>,
  <matplotlib.lines.Line2D at 0x2f6bc8921f0>],
 'caps': [<matplotlib.lines.Line2D at 0x2f6bc892490>,
  <matplotlib.lines.Line2D at 0x2f6bc892730>],
 'boxes': [<matplotlib.lines.Line2D at 0x2f6bc881c70>],
 'medians': [<matplotlib.lines.Line2D at 0x2f6bc8929d0>],
 'fliers': [<matplotlib.lines.Line2D at 0x2f6bc892c70>],
 'means': []}
```

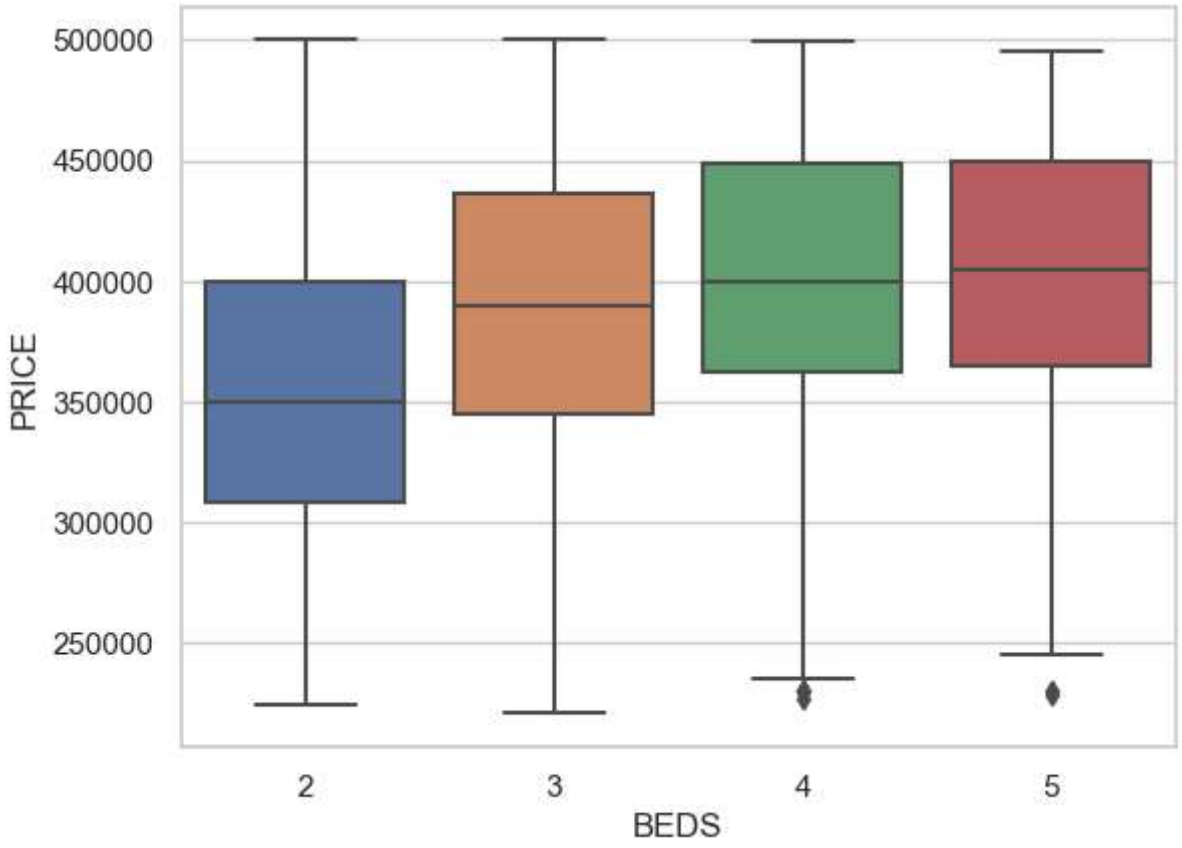



```
36  #remove the 1 bedroom and more than 5 bedrooms
    data = data[data.BEDS <6]
```

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

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

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



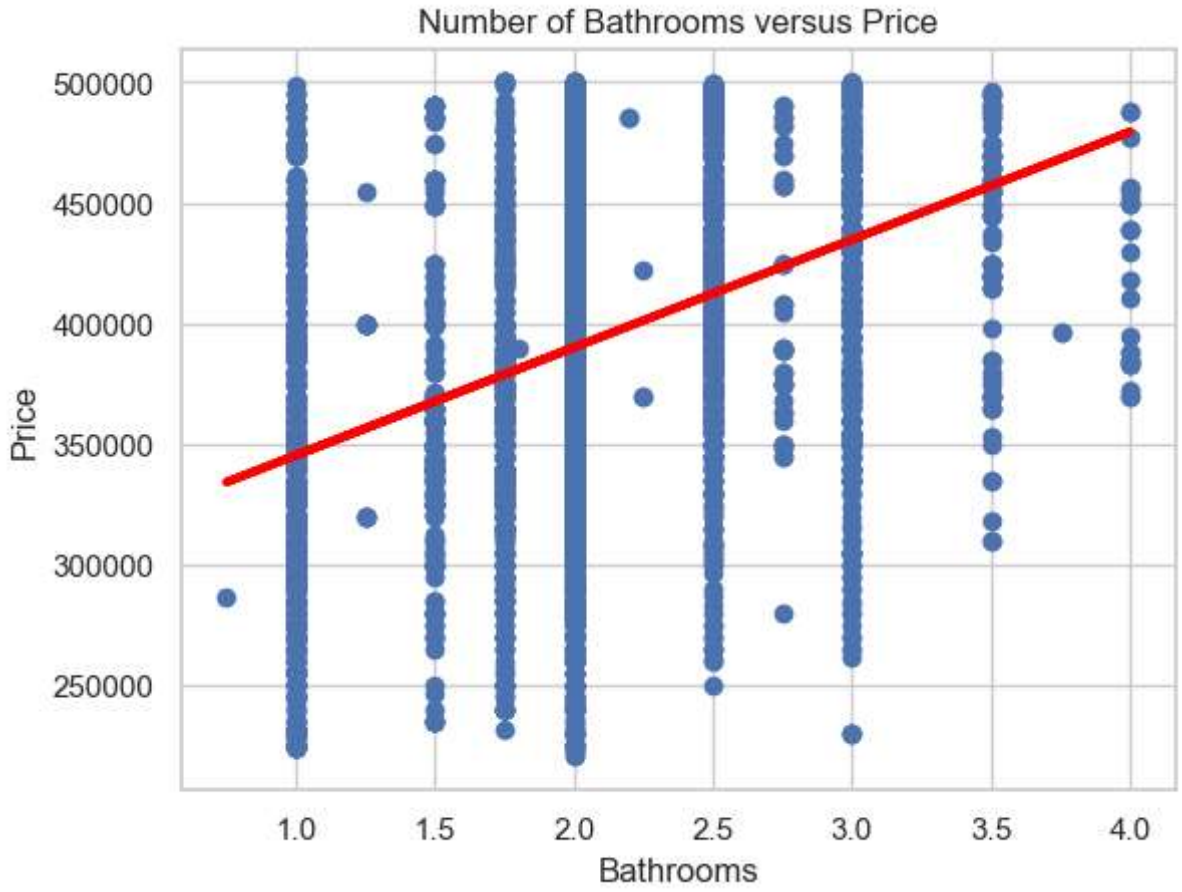
39
```
#Remove houses with 4+ bedrooms and sold for less than 250,000
dataremove = data[(data['BEDS']>=4) & (data['PRICE']< 250000)].index
data.drop(dataremove,inplace=True)
```

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

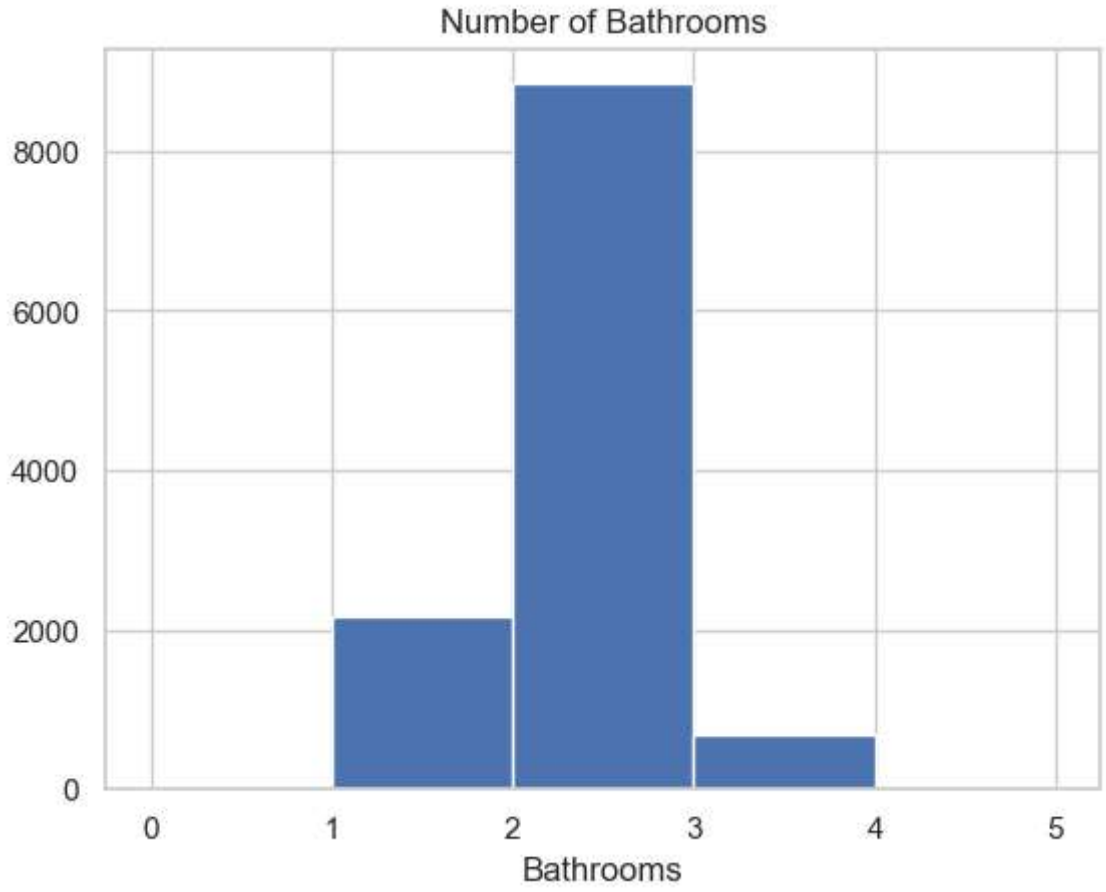
Compare number of bathrooms to price

40
```
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)
```

40
```
[<matplotlib.lines.Line2D at 0x2f6bcc37e20>]
```

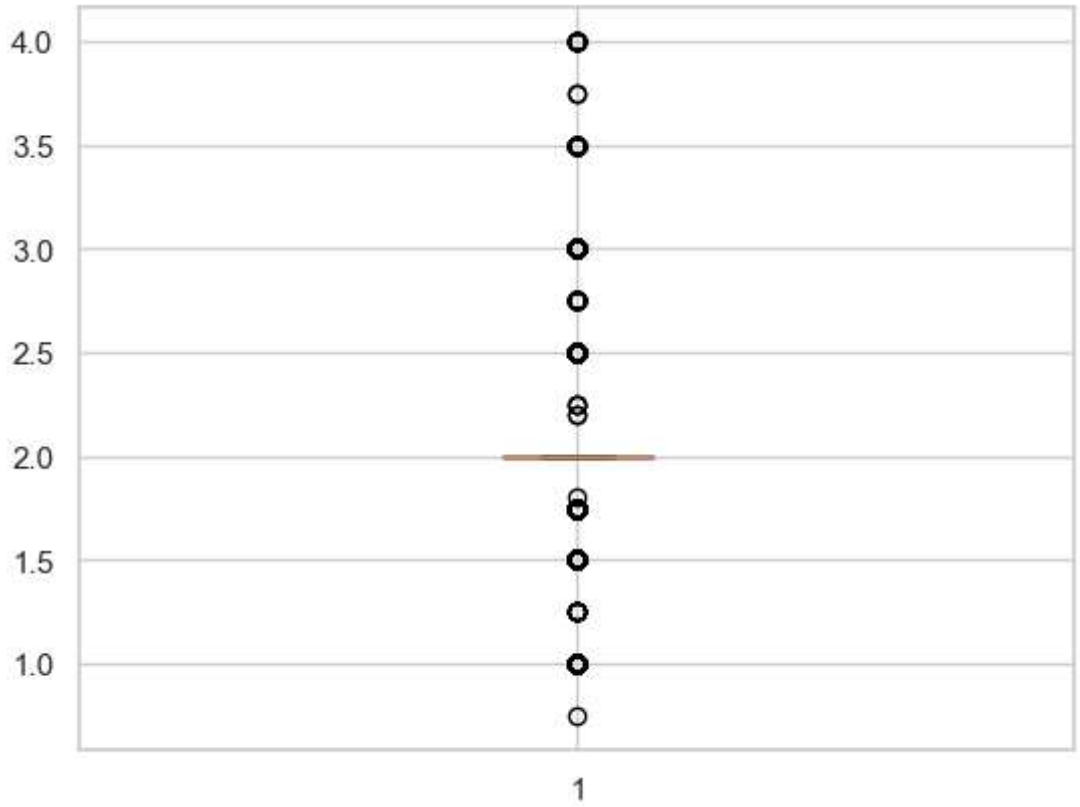


41
```
# view the distribution of bathrooms
plt.hist(data['BATHS'], bins= [0,1,2,3,4,5])
plt.title('Number of Bathrooms')
plt.xlabel('Bathrooms')
plt.show()
```



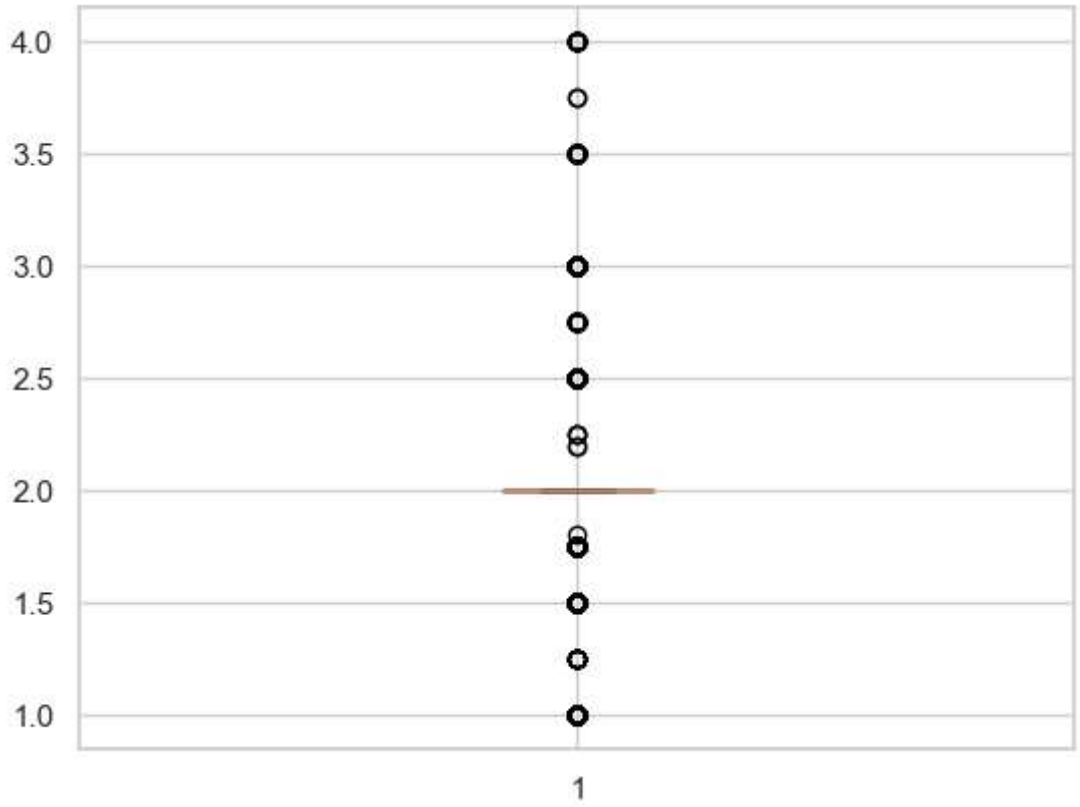
42
```
#identify outliers in the number of bathrooms
plt.boxplot(data['BATHS'])
```

42
```
{'whiskers': [<matplotlib.lines.Line2D at 0x2f6bcd2ae80>,
  <matplotlib.lines.Line2D at 0x2f6bcd3b160>],
 'caps': [<matplotlib.lines.Line2D at 0x2f6bcd3b2e0>,
  <matplotlib.lines.Line2D at 0x2f6bcd3b580>],
 'boxes': [<matplotlib.lines.Line2D at 0x2f6bcd2abe0>],
 'medians': [<matplotlib.lines.Line2D at 0x2f6bcd3b820>],
 'fliers': [<matplotlib.lines.Line2D at 0x2f6bcd3bac0>],
 'means': []}
```



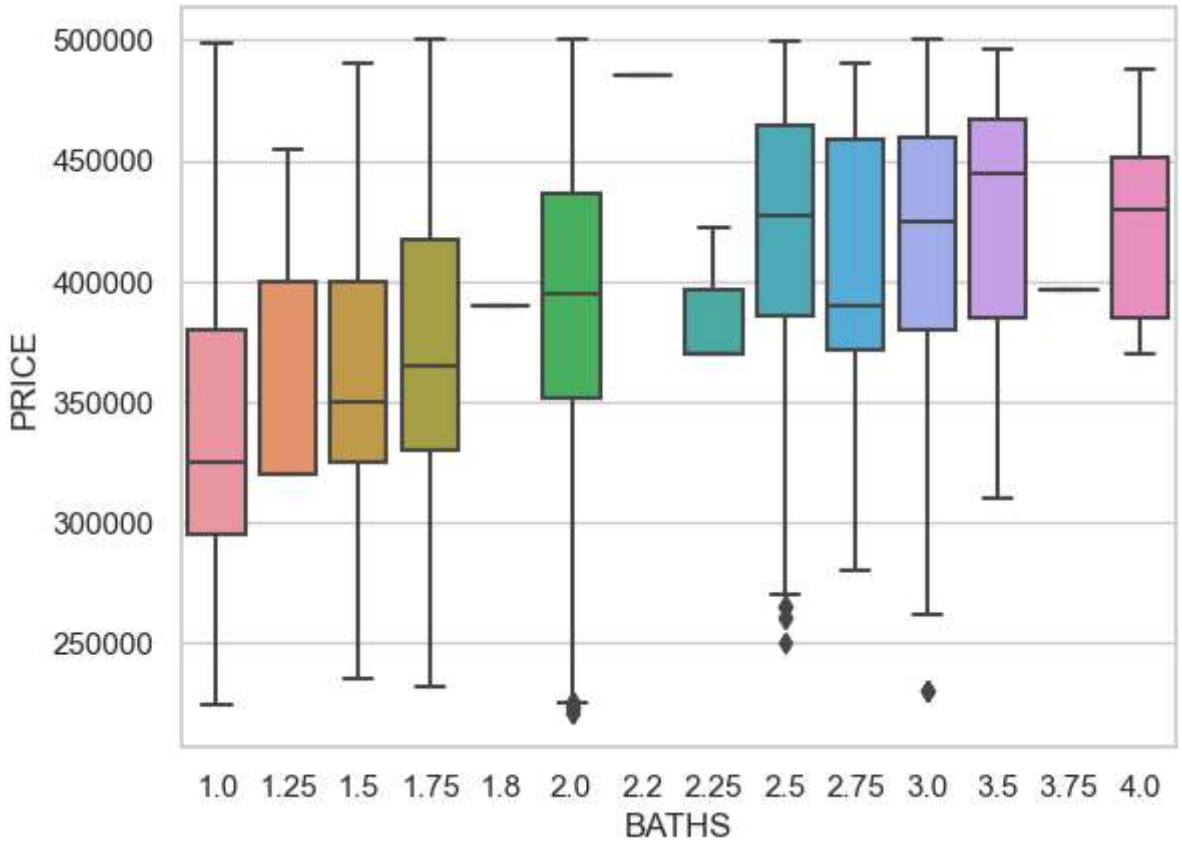

```
43  #remove less than 1 bathroom
    data = data[data.BATHS >= 1]
    plt.boxplot(data['BATHS'])
```

```
43  {'whiskers': [<matplotlib.lines.Line2D at 0x2f6bcc56cd0>,
      <matplotlib.lines.Line2D at 0x2f6bcc488e0>],
     'caps': [<matplotlib.lines.Line2D at 0x2f6bcc48c10>,
      <matplotlib.lines.Line2D at 0x2f6ba5b0130>],
     'boxes': [<matplotlib.lines.Line2D at 0x2f6bcc378e0>],
     'medians': [<matplotlib.lines.Line2D at 0x2f6ba5325b0>],
     'fliers': [<matplotlib.lines.Line2D at 0x2f6ba618b80>],
     'means': []}
```

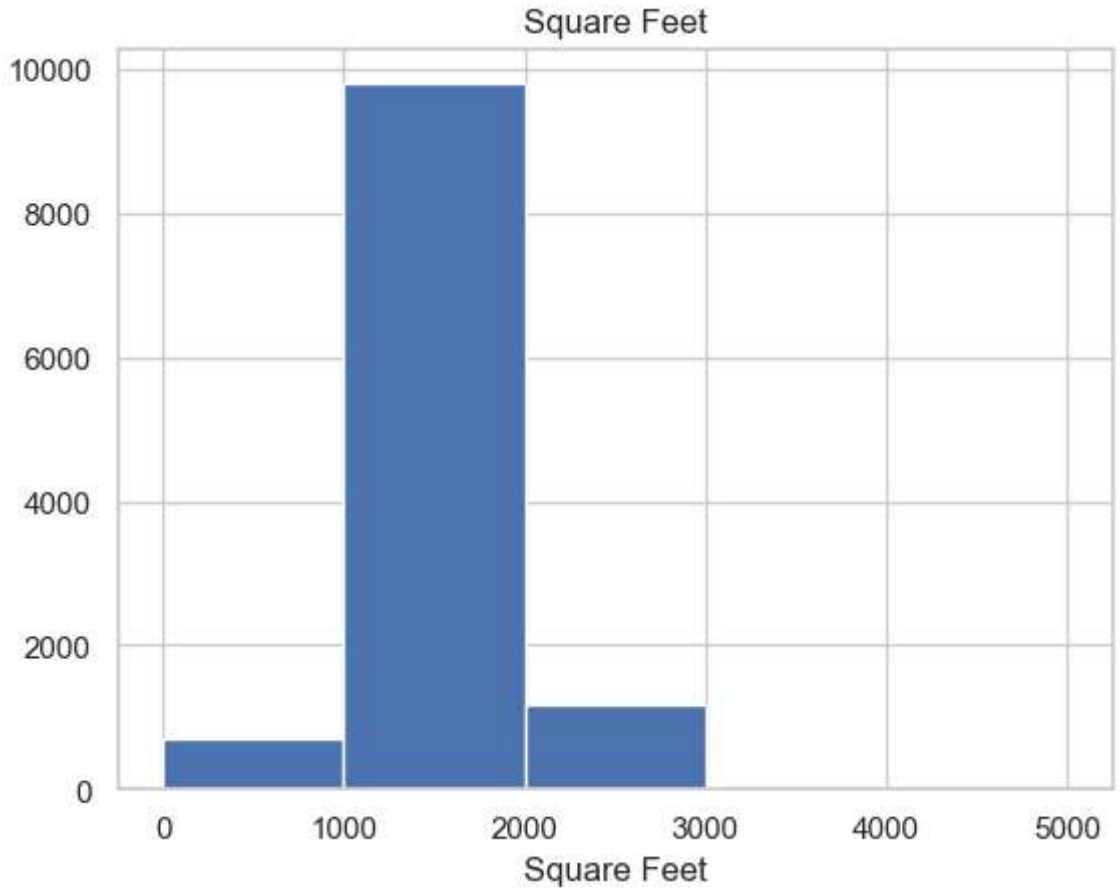



```
44  #view the relationship between bathrooms and price after removing outliers
    sn.set(style = 'whitegrid')
    sn.boxplot(x='BATHS',
               y='PRICE',
               data = data)
```

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

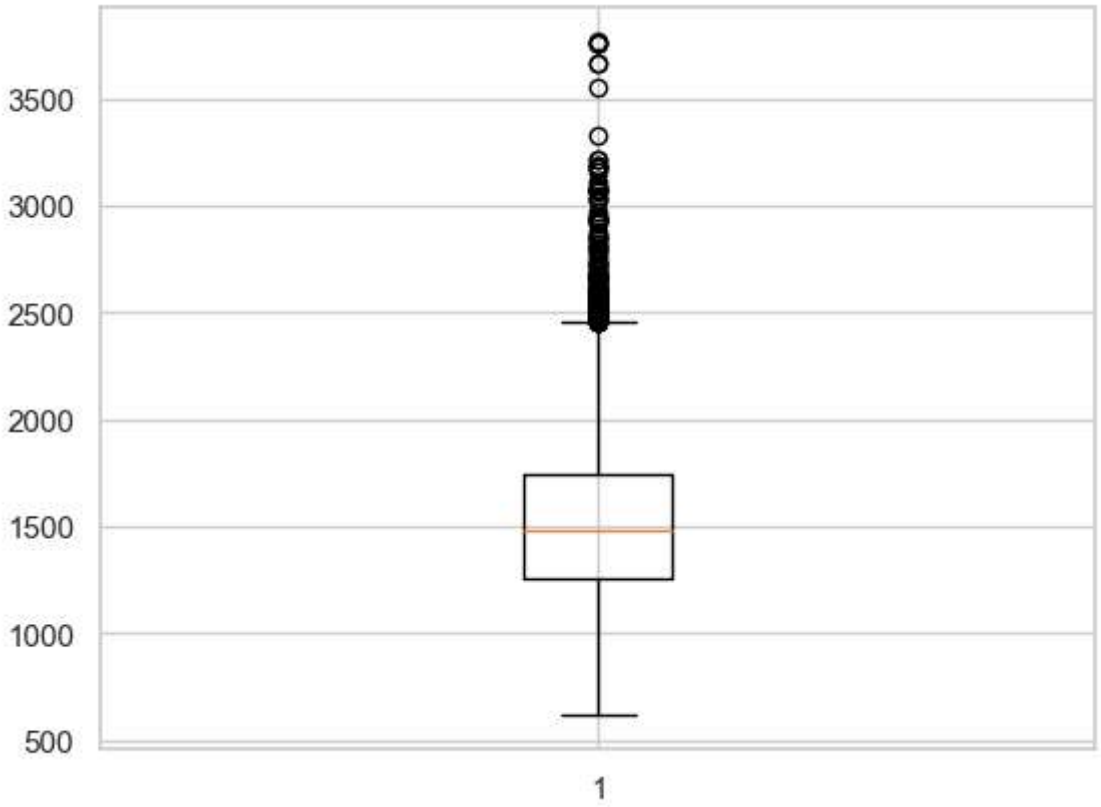



```
45  #view distribution of house square footage
    plt.hist(data['SQFT'], bins= [0,1000,2000,3000,4000,5000])
    plt.title('Square Feet')
    plt.xlabel('Square Feet')
    plt.show()
```



46
```
#view square footage as a boxplot
plt.boxplot(data['SQFT'])
```

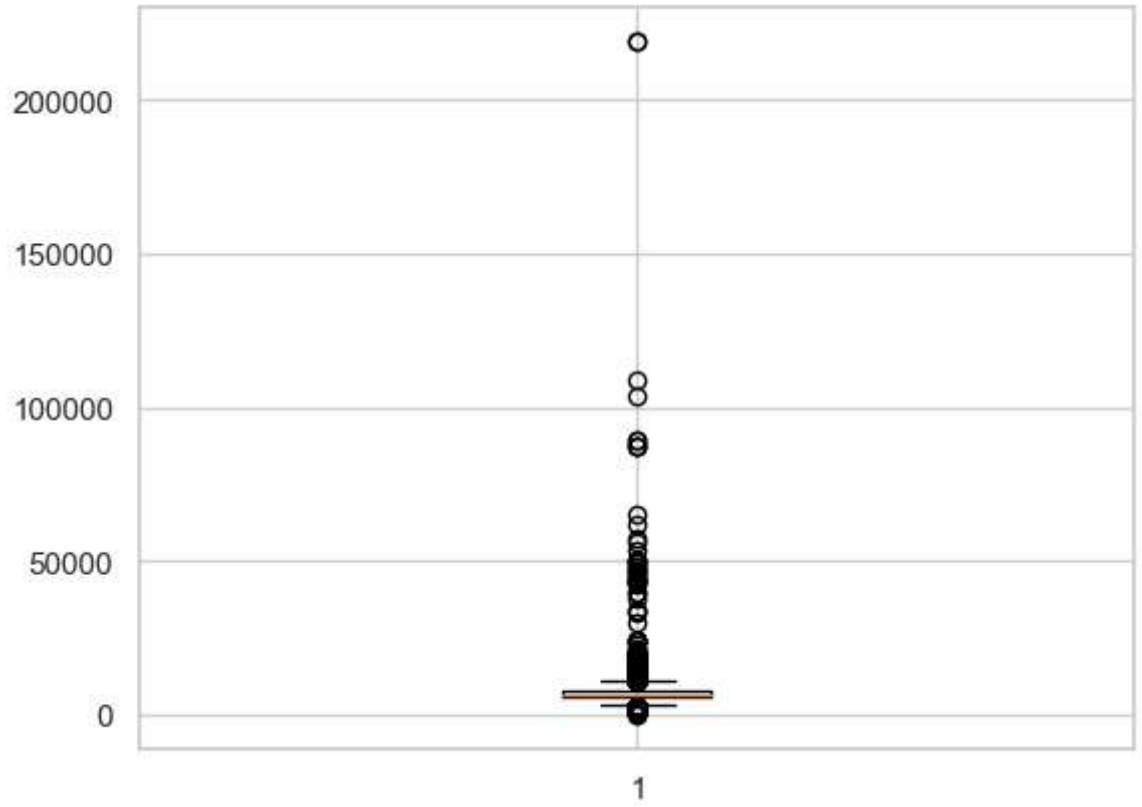
46
```
{'whiskers': [<matplotlib.lines.Line2D at 0x2f6bd3ad280>,
  <matplotlib.lines.Line2D at 0x2f6bd3ad520>],
 'caps': [<matplotlib.lines.Line2D at 0x2f6bd3ad7c0>,
  <matplotlib.lines.Line2D at 0x2f6bd3ada60>],
 'boxes': [<matplotlib.lines.Line2D at 0x2f6bd39dfa0>],
 'medians': [<matplotlib.lines.Line2D at 0x2f6bd3add00>],
 'fliers': [<matplotlib.lines.Line2D at 0x2f6bd3adfa0>],
 'means': []}
```



Explore Lot Size

47
```
#view lot size in a boxplot to identify outliers
plt.boxplot(data['LOTSIZE'])
```

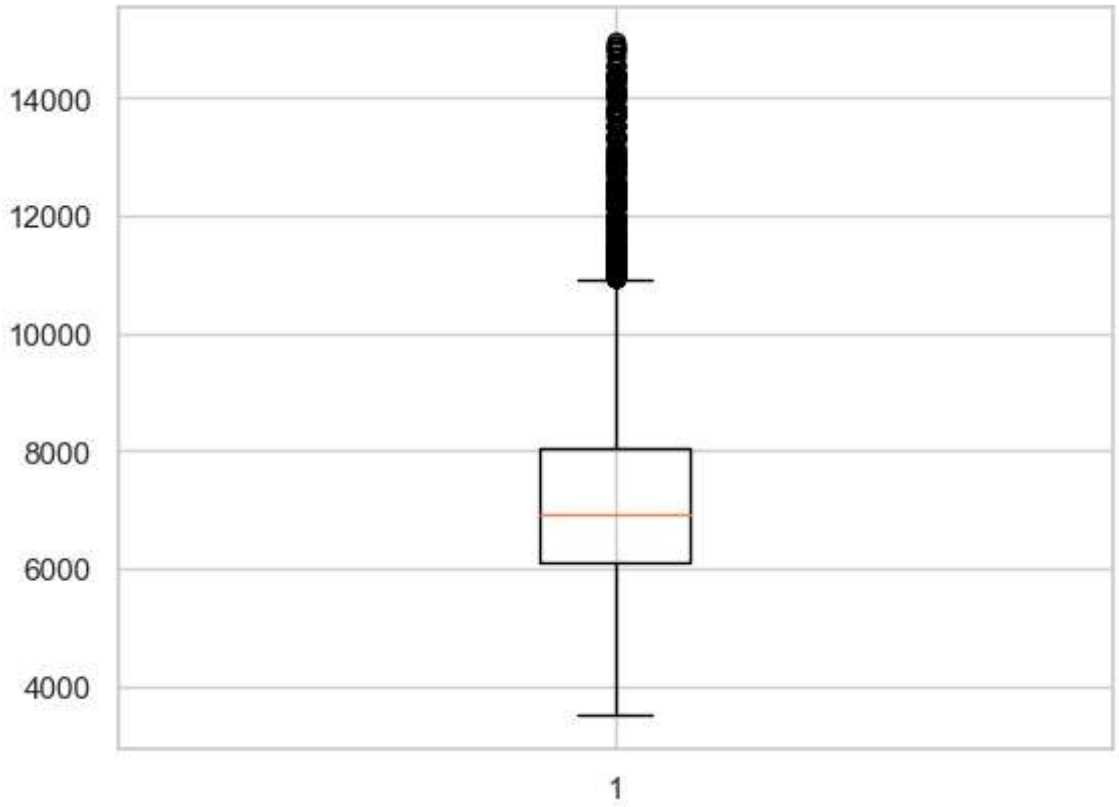
47
```
{'whiskers': [<matplotlib.lines.Line2D at 0x2f6bcf61310>,
  <matplotlib.lines.Line2D at 0x2f6bcf61a30>],
 'caps': [<matplotlib.lines.Line2D at 0x2f6bcf61d90>,
  <matplotlib.lines.Line2D at 0x2f6bcf61af0>],
 'boxes': [<matplotlib.lines.Line2D at 0x2f6bcf61700>],
 'medians': [<matplotlib.lines.Line2D at 0x2f6bcf43970>],
 'fliers': [<matplotlib.lines.Line2D at 0x2f6bcf43790>],
 'means': []}
```



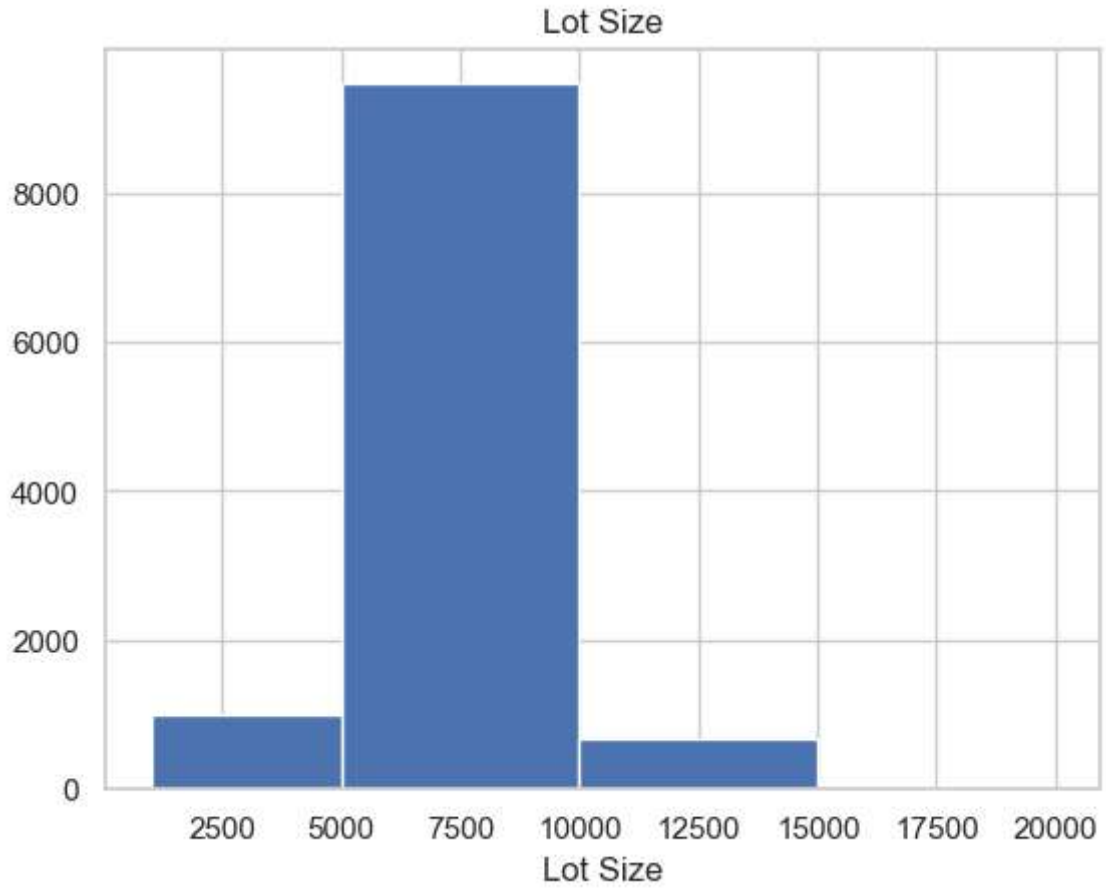
48
```
#Remove greater than 15,000sq ft and less than 3500 as unrealistic for price range
data = data[data.LOTSIZE < 15000]
data = data[data.LOTSIZE > 3500]
```

49
```
#view lotsize as a boxplot after removing outliers identified above
plt.boxplot(data['LOTSIZE'])
```

49
```
{'whiskers': [<matplotlib.lines.Line2D at 0x2f6bd1717f0>,
  <matplotlib.lines.Line2D at 0x2f6bd171a90>],
 'caps': [<matplotlib.lines.Line2D at 0x2f6bd171d30>,
  <matplotlib.lines.Line2D at 0x2f6bd171fd0>],
 'boxes': [<matplotlib.lines.Line2D at 0x2f6bd171550>],
 'medians': [<matplotlib.lines.Line2D at 0x2f6bd1802b0>],
 'fliers': [<matplotlib.lines.Line2D at 0x2f6bd180550>],
 'means': []}
```

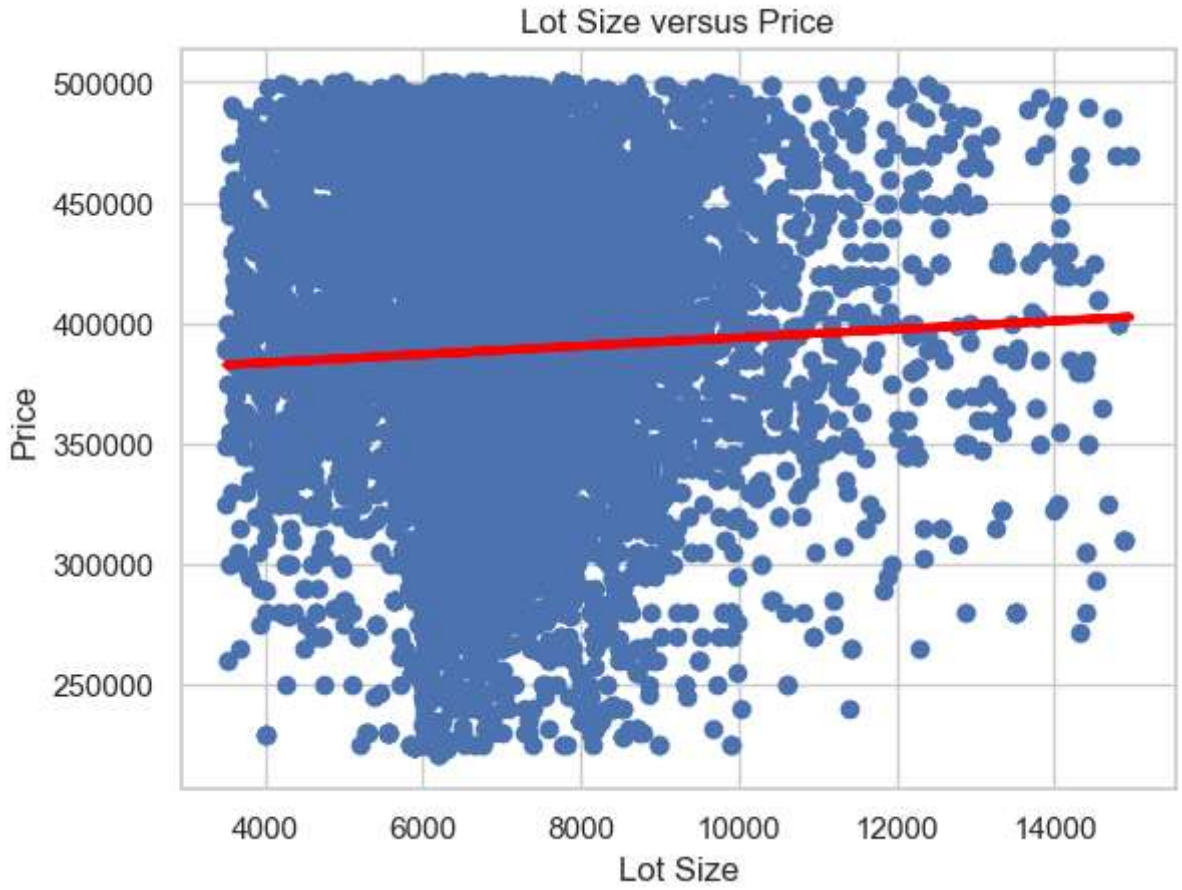



```
50  #view the distribution of houses by lotsize binning at 5000 sq ft intervals
    plt.hist(data['LOTSIZE'], bins= [1000,5000,10000,15000,20000])
    plt.title('Lot Size')
    plt.xlabel('Lot Size')
    plt.show()
```



51
```
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)
```

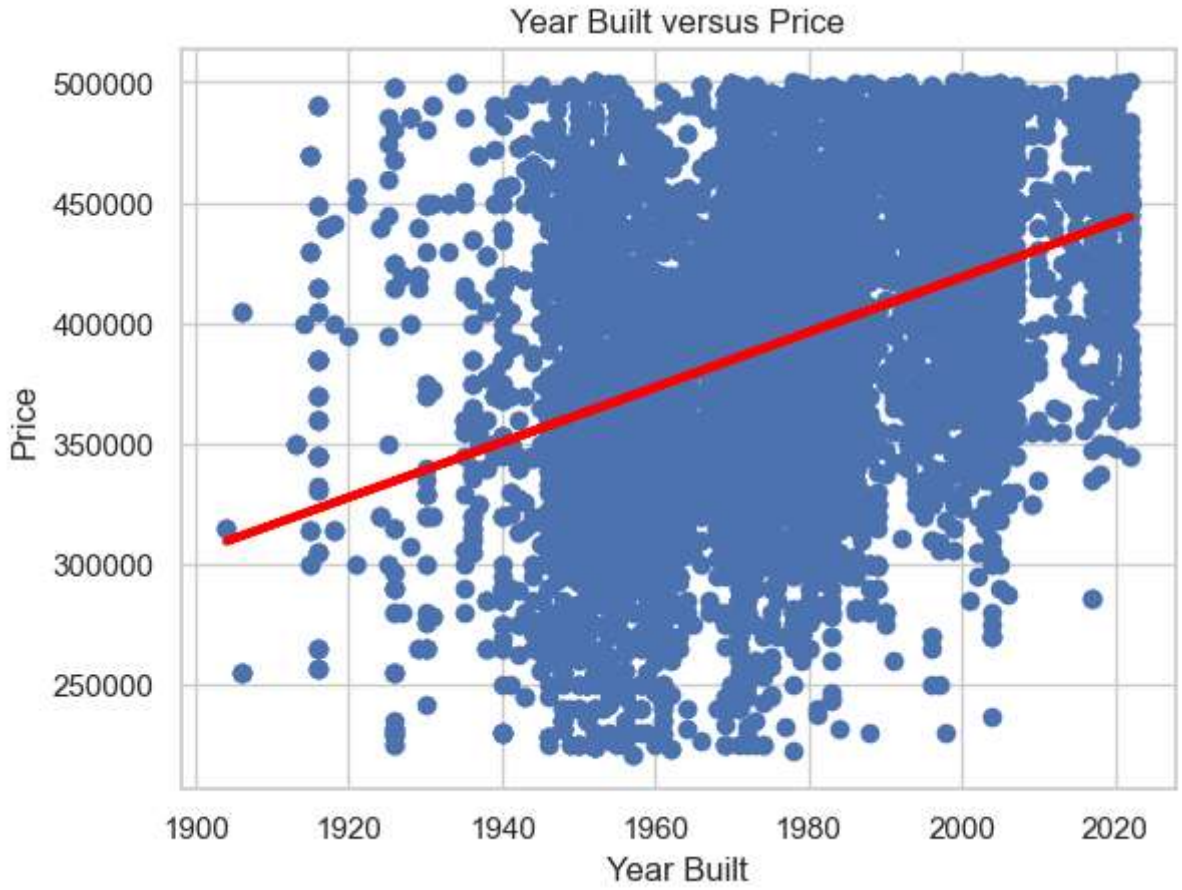
51
```
[<matplotlib.lines.Line2D at 0x2f6bd3c1790>]
```



Explore Year Built

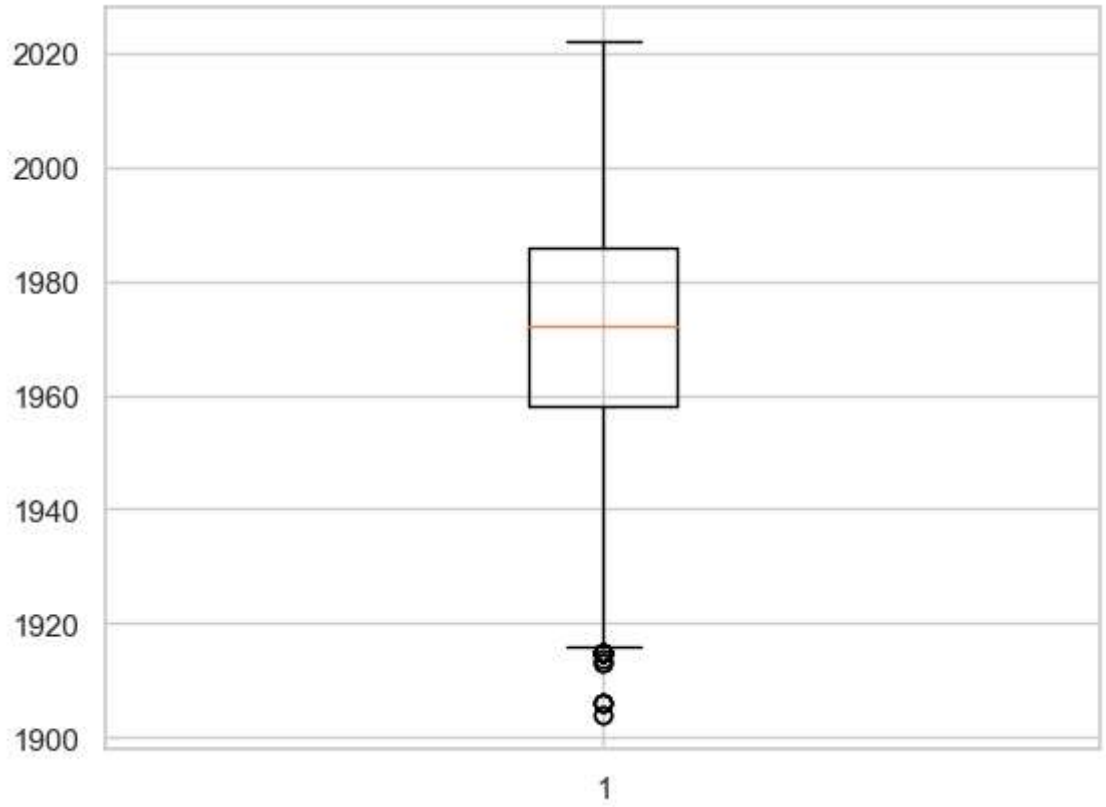
52
```
y=data['PRICE']
x=data['YEARBUILT']
plt.scatter(x,y)
plt.title('Year Built versus Price')
plt.xlabel('Year Built')
plt.ylabel('Price')

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

52
```
[<matplotlib.lines.Line2D at 0x2f6bd3d38e0>]
```



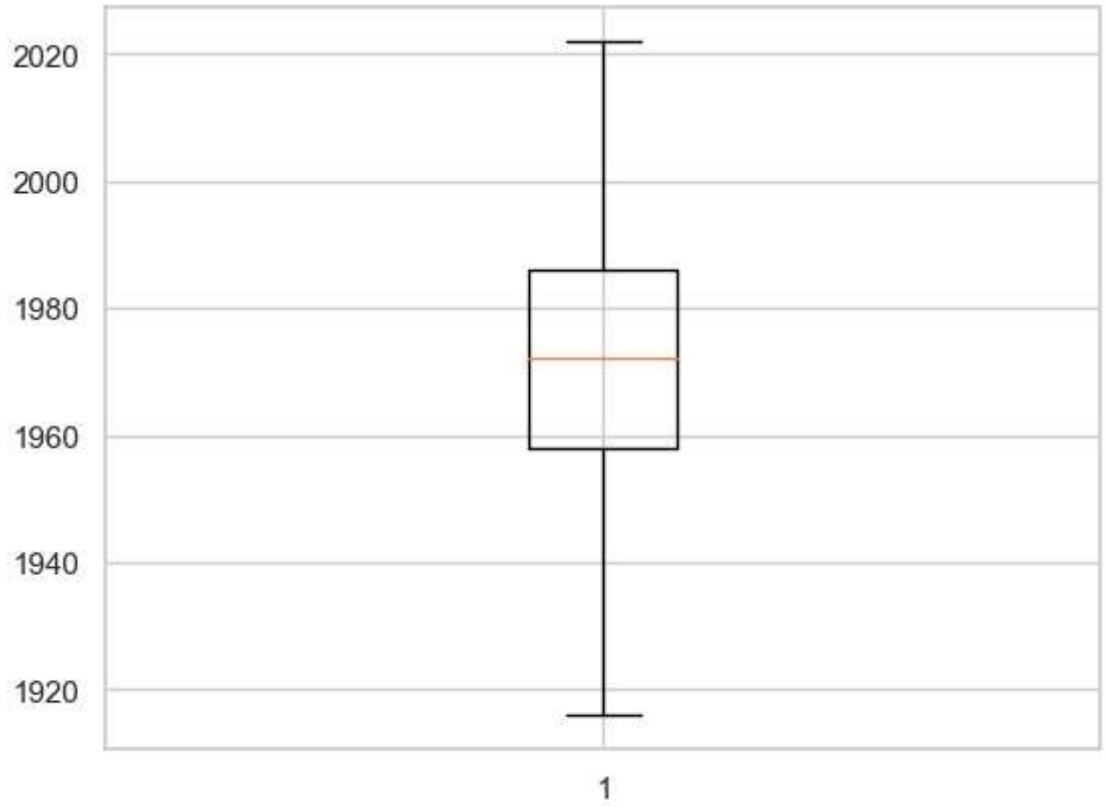
53
```
#view houses by year built to identify outliers
plt.boxplot(data['YEARBUILT'])
```

53
```
{'whiskers': [<matplotlib.lines.Line2D at 0x2f6be520100>,
  <matplotlib.lines.Line2D at 0x2f6be5203a0>],
 'caps': [<matplotlib.lines.Line2D at 0x2f6be520640>,
  <matplotlib.lines.Line2D at 0x2f6be5208e0>],
 'boxes': [<matplotlib.lines.Line2D at 0x2f6be50fe20>],
 'medians': [<matplotlib.lines.Line2D at 0x2f6be520b80>],
 'fliers': [<matplotlib.lines.Line2D at 0x2f6be520e20>],
 'means': []}
```



54
```
#Remove anything built before 1915
data = data[data.YEARBUILT > 1915]
plt.boxplot(data['YEARBUILT'])
```

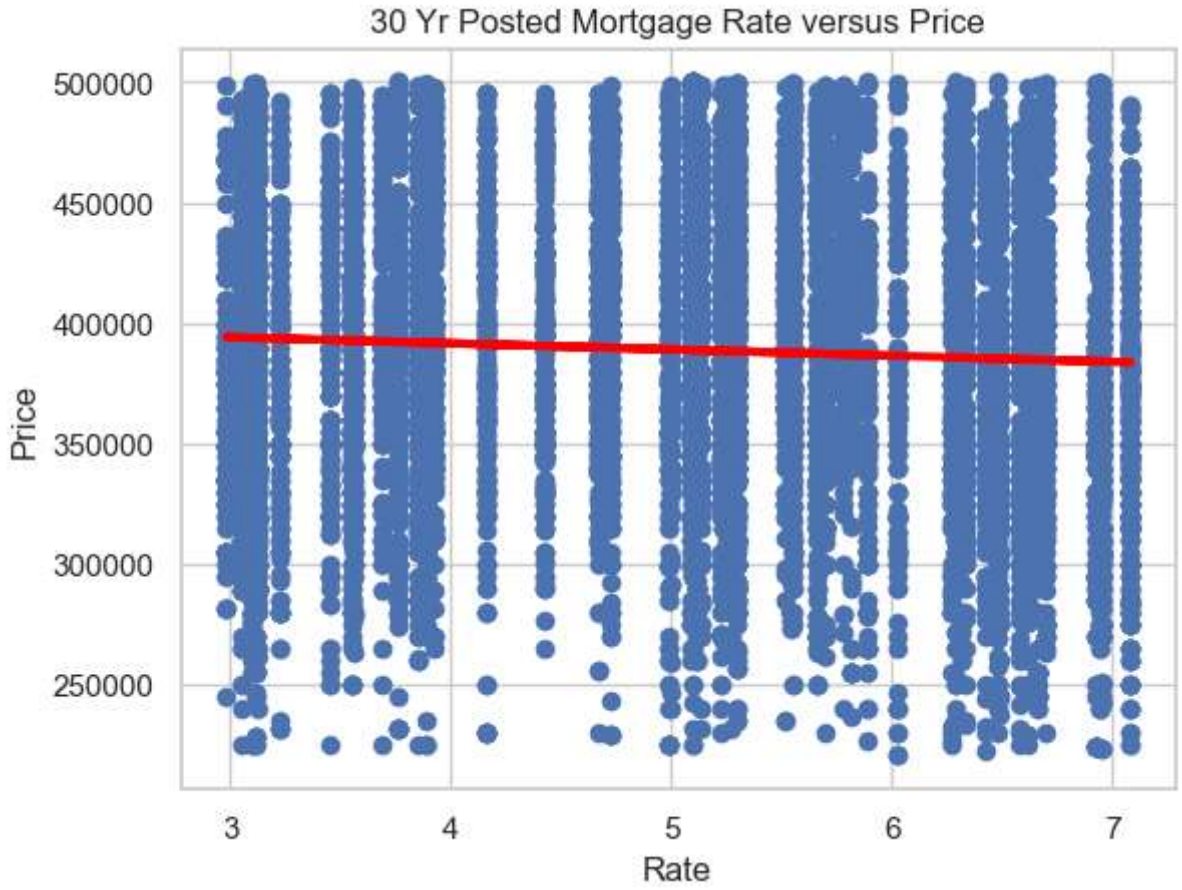
54
```
{'whiskers': [<matplotlib.lines.Line2D at 0x2f6be587bb0>,
  <matplotlib.lines.Line2D at 0x2f6be587e50>],
 'caps': [<matplotlib.lines.Line2D at 0x2f6be598130>,
  <matplotlib.lines.Line2D at 0x2f6be5983d0>],
 'boxes': [<matplotlib.lines.Line2D at 0x2f6be587a30>],
 'medians': [<matplotlib.lines.Line2D at 0x2f6be598670>],
 'fliers': [<matplotlib.lines.Line2D at 0x2f6be598910>],
 'means': []}
```



Explore Mortgage Rates

55
```
y=data['PRICE']
x=data['RATE']
plt.scatter(x,y)
plt.title('30 Yr Posted Mortgage Rate versus Price')
plt.xlabel('Rate')
plt.ylabel('Price')

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

55
```
[<matplotlib.lines.Line2D at 0x2f6be9a6f40>]
```



56
```
#Remove Sold Date, Address, City, State or Provice, MLS#, Latitude and Longitude

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

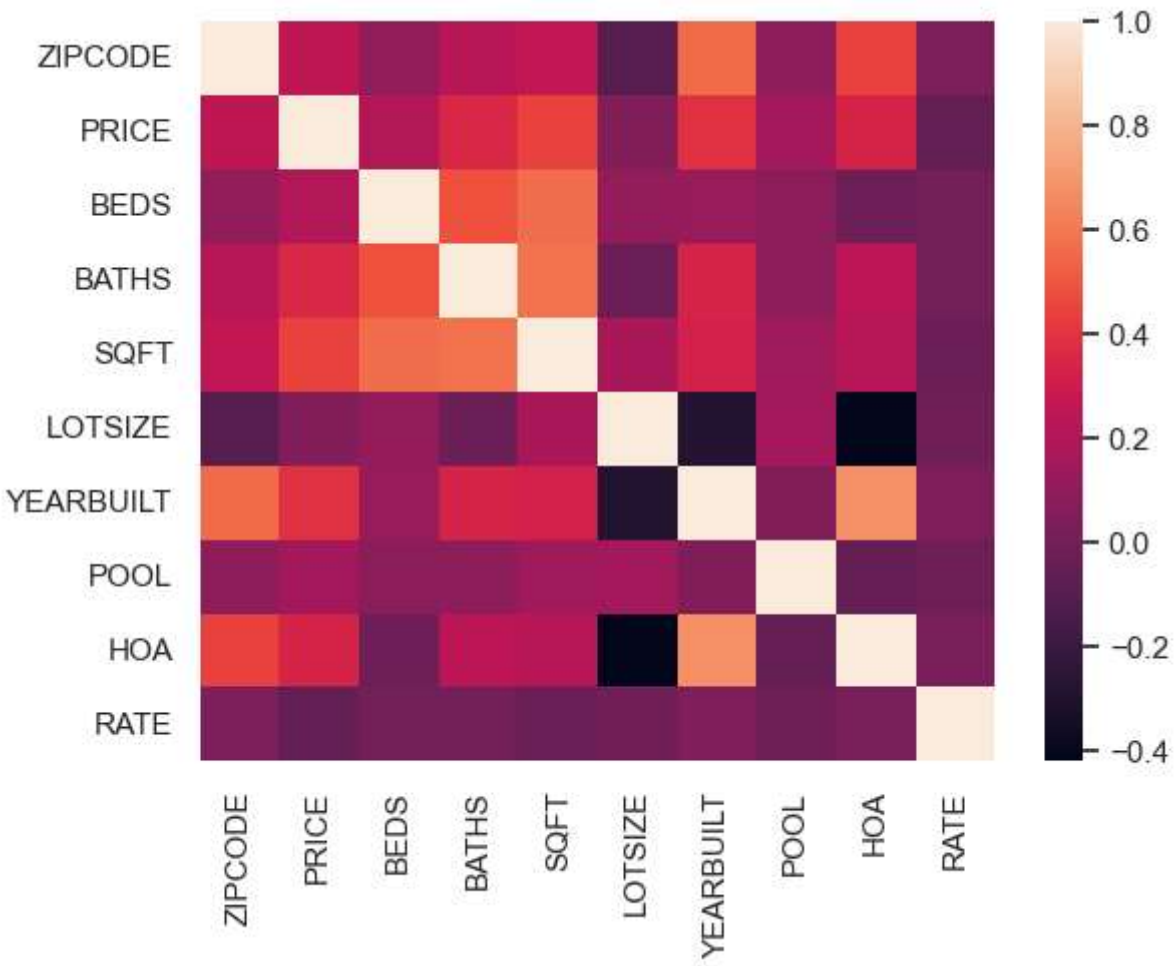
57
```
#view the first five rows of the model data and view the statistical data
model_data.head()
model_data.describe()
```

57

	ZIPCODE	PRICE	BEDS	BATHS	SQFT	LOTSIZE
count	11138.000000	11138.000000	11138.000000	11138.000000	11138.000000	11138.000000
mean	85031.244658	389081.565362	3.232896	1.965811	1513.171575	7156.721314
std	16.274615	60902.564563	0.655809	0.453167	373.280027	1732.882138
min	85003.000000	220500.000000	2.000000	1.000000	618.000000	3508.000000
25%	85019.000000	346000.000000	3.000000	2.000000	1256.000000	6103.000000
50%	85032.000000	390000.000000	3.000000	2.000000	1476.000000	6930.000000
75%	85041.000000	438000.000000	4.000000	2.000000	1740.000000	8032.000000
max	85087.000000	499999.000000	5.000000	4.000000	3770.000000	14963.000000

Heatmap of numerical values

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




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

```
59  PRICE        1.000000
    SQFT         0.443568
    YEARBUILT    0.386119
    BATHS        0.353641
    HOA          0.335034
    ZIPCODE      0.250123
    BEDS         0.205814
    POOL         0.154119
    LOTSIZE      0.050606
    RATE        -0.053843
    Name: PRICE, dtype: float64
```

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

```
60  ZIPCODE        int64
    PRICE        float64
    BEDS           int32
    BATHS        float64
    SQFT           int64
```

```
LOTSIZE        int64
YEARBUILT      int64
POOL           int32
HOA            int32
RATE         float64
dtype: object
```

Build Models

```
#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']
```

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

```
array([[-1.55123738, -1.88004598, -2.13134125, ...,  2.29058401,
        -0.54185983,  0.81079492],
       [-1.48978923,  1.16975771,  0.07544851, ...,  2.29058401,
        -0.54185983,  0.53018504],
       [-1.48978923, -1.88004598, -2.13134125, ...,  2.29058401,
        -0.54185983,  1.52790904],
       ...,
       [ 3.36461415, -0.35514414,  1.17884339, ..., -0.43656989,
         1.84549571,  1.53570376],
       [ 3.36461415, -1.88004598,  0.07544851, ..., -0.43656989,
         1.84549571,  1.03684176],
       [ 3.36461415, -0.35514414,  0.07544851, ..., -0.43656989,
         1.84549571,  1.00566288]])
```

```
#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.

```
#Using GridSearchCV found the best parameters for the random forest model
RandFor = RandomForestRegressor(bootstrap=False, max_features=5, n_estimators=375, random_state=10)
RandFor.fit(X_train,y_train)

y_pred = RandFor.predict(X_test)
```

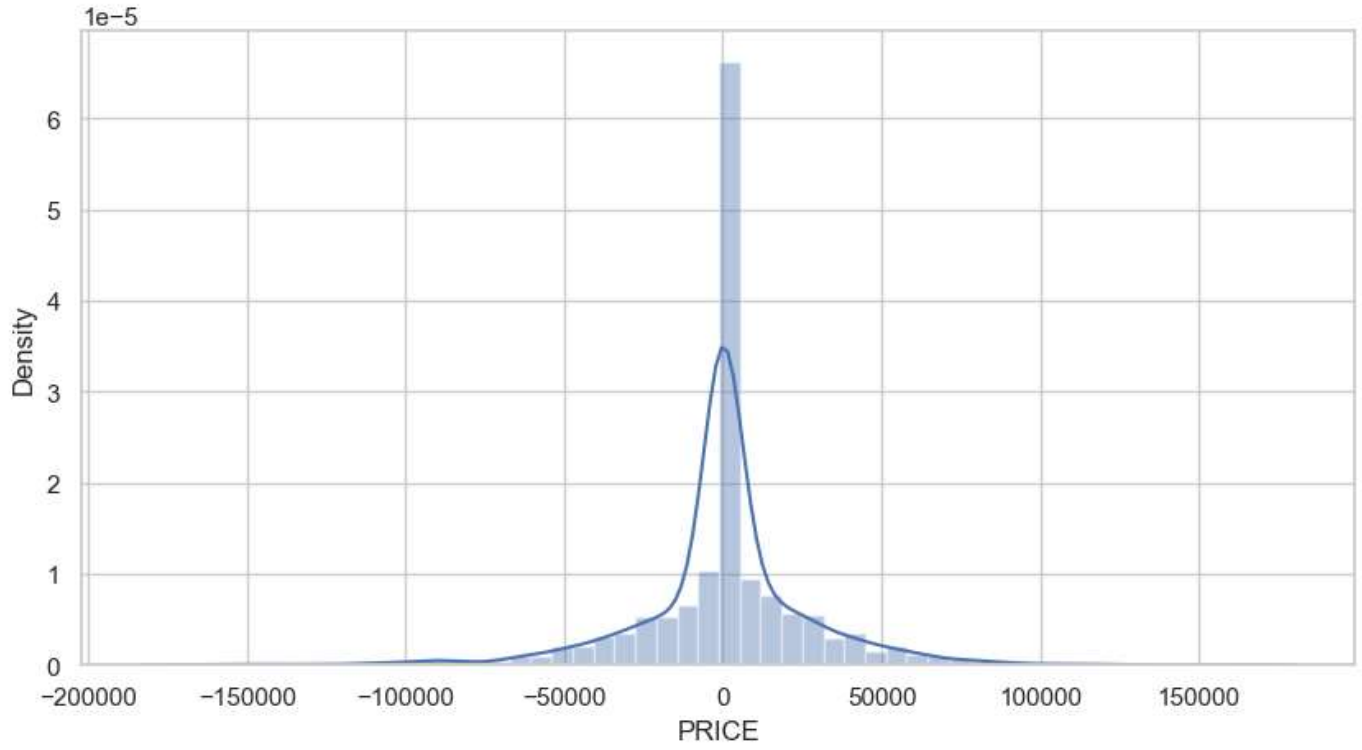
```
print('MAE: ', mean_absolute_error(y_test,y_pred))
print('MSE: ', mean_squared_error(y_test,y_pred))
print('R2: ', r2_score(y_test,y_pred))
print('VarScore: ',explained_variance_score(y_test,y_pred) )
print('RMSE: ', np.sqrt(mean_squared_error(y_test,y_pred)))

MAE:  15931.627524835427
MSE:  755869659.0647352
R2:  0.8001049329738827
```

```
VarScore:  0.8001485884840136
RMSE:  27493.083840572253
```

66
```
fig=plt.figure(figsize=(10,5))
residuals = (y_test - y_pred)
sn.distplot(residuals)
```

```
C:\Users\belbi\AppData\Local\Temp\ipykernel_123424\1542007201.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)
```

66
```
<AxesSubplot: xlabel='PRICE', ylabel='Density'>
```



67
```
#change working directory
os.chdir(r"C:\Users\belbi\PycharmProject\Capstone")
```

68
```
## to test the model

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

69
```
datatest = scaler.transform(testdata)
prediction = RandFor.predict(datatest)
prediction
```

69
```
array([404893.57866667, 409609.11733333, 459530.59466667])
```

Save trained model

```
70  # 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)



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




72  # 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)


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



74
    # 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)



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


76  # 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)


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