In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
real = pd.read_csv("MCSReal_Estate.csv")
real.head()
```

Out[2]:

	ids	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
0	0	79545.458574	missing	?	NaN	23086.800503	$1059033.5578701235
1	1	79248.642455	6.0028998082752425	6.730821019094919	3.09	40173.072174	Rs112941818.61352125
2	2	61287.067179	5.865889840310001	8.512727430375099	5.13	36882.159400	$1058987.9878760849
3	3	63345.240046	7.1882360945186425	?	NaN	34310.242831	Rs94546260.4972085
4	4	59982.197226	5.040554523106283	7.839387785120487	4.23	26354.109472	$630943.4893385402

In [3]:
```
real.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 9 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   ids                          5000 non-null   int64  
 1   Avg. Area Income             5000 non-null   float64
 2   Avg. Area House Age          5000 non-null   object 
 3   Avg. Area Number of Rooms    5000 non-null   object 
 4   Avg. Area Number of Bedrooms  3333 non-null   float64
 5   Area Population              5000 non-null   float64
 6   Price                        5000 non-null   object 
 7   Address                      5000 non-null   object 
 8   Avg Area Comfort             200 non-null    float64
dtypes: float64(4), int64(1), object(4)
memory usage: 351.7+ KB
```

In [4]:
```
df1 = real[real["Price"].str.contains("Rs")]
```

In [5]:
```
df1.head()
```

Out[5]:

	ids	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
1	1	79248.642455	6.0028998082752425	6.730821019094919	3.09	40173.072174	Rs112941818.61352125
3	3	63345.240046	7.1882360945186425	?	NaN	34310.242831	Rs94546260.4972085
5	5	80175.754159	4.9884077575337145	6.104512439428879	4.04	26748.428425	Rs80110355.57951477
7	7	78394.339278	6.9897797477182815	6.620477995185026	2.42	36516.358972	Rs118045242.33582912
9	9	81885.927184	4.423671789897876	?	NaN	40149.965749	Rs115886610.94814718

In [6]:
```
df2 = real[~real["Price"].str.contains("Rs", na=False)]
```

In [7]:
```
df2.head()
```

Out[7]:

	ids	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
0	0	79545.458574	missing	?	NaN	23086.800503	$1059033.5578701235
2	2	61287.067179	5.865889840310001	8.512727430375099	5.13	36882.159400	$1058987.9878760849
4	4	59982.197226	5.040554523106283	7.839387785120487	4.23	26354.109472	$630943.4893385402
6	6	64698.463428	6.025335906887153	?	NaN	60828.249085	$1502055.8173744078
8	8	59927.660813	5.36212556960358	6.3931209805509015	2.30	29387.396003	$798869.5328331633

In [8]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2500 entries, 1 to 4999
Data columns (total 9 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   ids                           2500 non-null   int64  
 1   Avg. Area Income              2500 non-null   float64
 2   Avg. Area House Age           2500 non-null   object 
 3   Avg. Area Number of Rooms     2500 non-null   object 
 4   Avg. Area Number of Bedrooms  1667 non-null   float64
 5   Area Population               2500 non-null   float64
 6   Price                         2500 non-null   object 
 7   Address                       2500 non-null   object 
 8   Avg Area Comfort              100 non-null    float64
dtypes: float64(4), int64(1), object(4)
memory usage: 195.3+ KB
```

In [9]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2500 entries, 0 to 4998
Data columns (total 9 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   ids                           2500 non-null   int64  
 1   Avg. Area Income              2500 non-null   float64
 2   Avg. Area House Age           2500 non-null   object 
 3   Avg. Area Number of Rooms     2500 non-null   object 
 4   Avg. Area Number of Bedrooms  1666 non-null   float64
 5   Area Population               2500 non-null   float64
 6   Price                         2500 non-null   object 
 7   Address                       2500 non-null   object 
 8   Avg Area Comfort              100 non-null    float64
dtypes: float64(4), int64(1), object(4)
memory usage: 195.3+ KB
```

In [10]:
```
df1 = df1.replace({"Rs": ""}, regex=True)
```

In [11]:
```
df1.head()
```

Out[11]:

	ids	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	
1	1	79248.642455	6.0028998082752425	6.730821019094919	3.09	40173.072174	112941818.61352125	K
3	3	63345.240046	7.1882360945186425	?	NaN	34310.242831	94546260.4972085	
5	5	80175.754159	4.9884077575337145	6.104512439428879	4.04	26748.428425	80110355.57951477	0 44
7	7	78394.339278	6.9897797477182815	6.620477995185026	2.42	36516.358972	118045242.33582912	
9	9	81885.927184	4.423671789897876	?	NaN	40149.965749	115886610.94814718	0

In [12]:
```
la = lambda x:float(x[0:-1])
df1.Price = df1.Price.apply(la)
df1.Price.dtypes
```

Out[12]:
```
dtype('float64')
```

In [13]:
```
df1["Price"] = df1["Price"].round().astype(int)
```

In [14]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2500 entries, 1 to 4999
Data columns (total 9 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   ids                          2500 non-null   int64
 1   Avg. Area Income             2500 non-null   float64
 2   Avg. Area House Age          2500 non-null   object
 3   Avg. Area Number of Rooms    2500 non-null   object
 4   Avg. Area Number of Bedrooms  1667 non-null   float64
 5   Area Population              2500 non-null   float64
 6   Price                        2500 non-null   int32
 7   Address                      2500 non-null   object
 8   Avg Area Comfort             100 non-null    float64
dtypes: float64(4), int32(1), int64(1), object(3)
memory usage: 185.5+ KB
```

In [15]:
```
df1.head()
```

Out[15]:

	ids	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Addr
1	1	79248.642455	6.0028998082752425	6.730821019094919	3.09	40173.072174	112941819	188 Johns Views Su 079\nLa Kathleen, C
3	3	63345.240046	7.1882360945186425	?	NaN	34310.242831	94546260	U Barnett\nF AP 448
5	5	80175.754159	4.9884077575337145	6.104512439428879	4.04	26748.428425	80110356	06039 Jenn Islands A 443\nTracyp K
7	7	78394.339278	6.9897797477182815	6.620477995185026	2.42	36516.358972	118045242	972 Joy Viaduct\nLa William, 17778-64
9	9	81885.927184	4.423671789897876	?	NaN	40149.965749	115886611	Unit 9446 E 0958\nDPO 97(

In [16]:
```
df2.head()
```

Out[16]:

	ids	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
0	0	79545.458574	missing	?	NaN	23086.800503	$1059033.5578701235
2	2	61287.067179	5.865889840310001	8.512727430375099	5.13	36882.159400	$1058987.9878760849
4	4	59982.197226	5.040554523106283	7.839387785120487	4.23	26354.109472	$630943.4893385402
6	6	64698.463428	6.025335906887153	?	NaN	60828.249085	$1502055.8173744078
8	8	59927.660813	5.36212556960358	6.3931209805509015	2.30	29387.396003	$798869.5328331633

In [17]:
```
df2["Price"] = df2["Price"].str.replace("$", "")
```

In [18]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2500 entries, 0 to 4998
Data columns (total 9 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   ids                           2500 non-null   int64  
 1   Avg. Area Income              2500 non-null   float64
 2   Avg. Area House Age           2500 non-null   object 
 3   Avg. Area Number of Rooms     2500 non-null   object 
 4   Avg. Area Number of Bedrooms  1666 non-null   float64
 5   Area Population               2500 non-null   float64
 6   Price                         2500 non-null   object 
 7   Address                       2500 non-null   object 
 8   Avg Area Comfort              100 non-null    float64
dtypes: float64(4), int64(1), object(4)
memory usage: 195.3+ KB
```

In [19]:
```
la2 = lambda x:float(x[0:-1])
df2.Price = df2.Price.apply(la2)
df2.Price.dtypes
```

Out[19]:
```
dtype('float64')
```

In [20]:
```
df2["Price"] = df2["Price"].round().astype(int)
```

In [21]:
```
df2.head()
```

Out[21]:

	ids	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	
0	0	79545.458574	missing	?	NaN	23086.800503	1059034	208 Michael F 674\nLaura
2	2	61287.067179	5.865889840310001	8.512727430375099	5.13	36882.159400	1058988	9127 Stravenue\nDa W
4	4	59982.197226	5.040554523106283	7.839387785120487	4.23	26354.109472	630943	USNS Raymo A
6	6	64698.463428	6.025335906887153	?	NaN	60828.249085	1502056	4759 Dani 442\nNguyenb
8	8	59927.660813	5.36212556960358	6.3931209805509015	2.30	29387.396003	798870	USS Gilbert\r

In [22]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2500 entries, 0 to 4998
Data columns (total 9 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   ids                          2500 non-null   int64  
 1   Avg. Area Income             2500 non-null   float64
 2   Avg. Area House Age          2500 non-null   object 
 3   Avg. Area Number of Rooms    2500 non-null   object 
 4   Avg. Area Number of Bedrooms  1666 non-null   float64
 5   Area Population              2500 non-null   float64
 6   Price                        2500 non-null   int32  
 7   Address                      2500 non-null   object 
 8   Avg Area Comfort             100 non-null    float64
dtypes: float64(4), int32(1), int64(1), object(3)
memory usage: 185.5+ KB
```

In [23]:
```
df1["Price"] = df1["Price"]/75
```

In [24]:
```
df1.head()
```

Out[24]:

	ids	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Ad
1	1	79248.642455	6.0028998082752425	6.730821019094919	3.09	40173.072174	1.505891e+06	188 Jo Views 079 Kathleer
3	3	63345.240046	7.1882360945186425	?	NaN	34310.242831	1.260617e+06	Barnett AP
5	5	80175.754159	4.9884077575337145	6.104512439428879	4.04	26748.428425	1.068138e+06	06039 J Islan 443\nTra
7	7	78394.339278	6.9897797477182815	6.620477995185026	2.42	36516.358972	1.573937e+06	972 Viaduct Willia 17778
9	9	81885.927184	4.423671789897876	?	NaN	40149.965749	1.545155e+06	Unit 94 0958\nD

In [25]:
```
re = df2.append(df1)
```

In [26]:
```
re.head()
```

Out[26]:

	ids	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	
0	0	79545.458574	missing	?	NaN	23086.800503	1059034.0	208 Michae 674\nLau
2	2	61287.067179	5.865889840310001	8.512727430375099	5.13	36882.159400	1058988.0	912 Stravenue\nD \
4	4	59982.197226	5.040554523106283	7.839387785120487	4.23	26354.109472	630943.0	USNS Raym
6	6	64698.463428	6.025335906887153	?	NaN	60828.249085	1502056.0	4759 Da 442\nNguyer
8	8	59927.660813	5.36212556960358	6.3931209805509015	2.30	29387.396003	798870.0	USS Gilber

In [27]:
```
re.tail()
```

Out[27]:

	ids	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	
4991	4991	74102.191890	5.657841005858683	7.683993273008141	3.13	24041.270592	1.263721e+06	Lodg
4993	4993	69639.140896	5.007510102029201	7.778375216524826	6.05	54056.128430	1.381831e+06	
4995	4995	60567.944140	7.830362443635721	?	NaN	22837.361035	1.060194e+06	Wi
4997	4997	63390.686886	7.250590614779546	4.805080980291155	2.13	33266.145490	1.030730e+06	421 07
4999	4999	65510.581804	5.992305307333977	6.792336104424982	4.07	46501.283803	1.298950e+06	509

In [28]:
```
re.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5000 entries, 0 to 4999
Data columns (total 9 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   ids                          5000 non-null   int64  
 1   Avg. Area Income             5000 non-null   float64
 2   Avg. Area House Age          5000 non-null   object 
 3   Avg. Area Number of Rooms    5000 non-null   object 
 4   Avg. Area Number of Bedrooms  3333 non-null   float64
 5   Area Population              5000 non-null   float64
 6   Price                        5000 non-null   float64
 7   Address                      5000 non-null   object 
 8   Avg Area Comfort             200 non-null    float64
dtypes: float64(5), int64(1), object(3)
memory usage: 390.6+ KB
```

In [29]:
```
re["Price"] = re["Price"].round().astype(int)
```

In [30]:
```
re.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5000 entries, 0 to 4999
Data columns (total 9 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   ids                          5000 non-null   int64  
 1   Avg. Area Income             5000 non-null   float64
 2   Avg. Area House Age          5000 non-null   object 
 3   Avg. Area Number of Rooms    5000 non-null   object 
 4   Avg. Area Number of Bedrooms  3333 non-null   float64
 5   Area Population              5000 non-null   float64
 6   Price                        5000 non-null   int32  
 7   Address                      5000 non-null   object 
 8   Avg Area Comfort             200 non-null    float64
dtypes: float64(4), int32(1), int64(1), object(3)
memory usage: 371.1+ KB
```

In [31]:
```
re.tail()
```

Out[31]:

	ids	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	
4991	4991	74102.191890	5.657841005858683	7.683993273008141	3.13	24041.270592	1263721	2 Lodge\nAn GU 61
4993	4993	69639.140896	5.007510102029201	7.778375216524826	6.05	54056.128430	1381831	5 Caus 9 Al
4995	4995	60567.944140	7.830362443635721	?	NaN	22837.361035	1060194	Williams' 30
4997	4997	63390.686886	7.250590614779546	4.805080980291155	2.13	33266.145490	1030730	4215 Trac 076\nJo
4999	4999	65510.581804	5.992305307333977	6.792336104424982	4.07	46501.283803	1298950	3777 R 509\nEast

In [32]:
```
re
```

Out[32]:

	ids	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	
0	0	79545.458574	missing	?	NaN	23086.800503	1059034	208 Mic 674\n
2	2	61287.067179	5.865889840310001	8.512727430375099	5.13	36882.159400	1058988	S Stravenu
4	4	59982.197226	5.040554523106283	7.839387785120487	4.23	26354.109472	630943	USNS R
6	6	64698.463428	6.025335906887153	?	NaN	60828.249085	1502056	4759 442\nNgu
8	8	59927.660813	5.36212556960358	6.3931209805509015	2.30	29387.396003	798870	USS Gi
...	...	...	...	...	...	...	...	
4991	4991	74102.191890	5.657841005858683	7.683993273008141	3.13	24041.270592	1263721	Lodge G
4993	4993	69639.140896	5.007510102029201	7.778375216524826	6.05	54056.128430	1381831	5259 Da A
4995	4995	60567.944140	7.830362443635721	?	NaN	22837.361035	1060194	USNS A
4997	4997	63390.686886	7.250590614779546	4.805080980291155	2.13	33266.145490	1030730	4215 Suite 076
4999	4999	65510.581804	5.992305307333977	6.792336104424982	4.07	46501.283803	1298950	37778 (Apt. 5(

5000 rows × 9 columns

In [33]:
```
re["Address"][0]
```

Out[33]:
```
'208 Michael Ferry Apt. 674\nLaurabury, NE 37010-5101'
```

In [34]:
```
def getstate(add):
    return add.split()[-2]
```

In [35]:
```
getstate('208 Michael Ferry Apt. 674\nLaurabury, NE 37010-5101')
```

Out[35]:
```
'NE'
```

In [36]:
```
re["State"]=re["Address"].apply(getstate)
```

In [37]:
```
re.drop("Address",axis=1,inplace=True)
```

In [38]:
```
re.head()
```

Out[38]:

	ids	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Avg Area Comfort	Sta
0	0	79545.458574	missing	?	NaN	23086.800503	1059034	0.289937	N
2	2	61287.067179	5.865889840310001	8.512727430375099	5.13	36882.159400	1058988	NaN	'
4	4	59982.197226	5.040554523106283	7.839387785120487	4.23	26354.109472	630943	NaN	A
6	6	64698.463428	6.025335906887153	?	NaN	60828.249085	1502056	NaN	C
8	8	59927.660813	5.36212556960358	6.3931209805509015	2.30	29387.396003	798870	NaN	A

In [39]:
```
re.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5000 entries, 0 to 4999
Data columns (total 9 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   ids                          5000 non-null   int64  
 1   Avg. Area Income             5000 non-null   float64
 2   Avg. Area House Age          5000 non-null   object 
 3   Avg. Area Number of Rooms    5000 non-null   object 
 4   Avg. Area Number of Bedrooms  3333 non-null   float64
 5   Area Population              5000 non-null   float64
 6   Price                        5000 non-null   int32  
 7   Avg Area Comfort             200 non-null    float64
 8   State                        5000 non-null   object 
dtypes: float64(4), int32(1), int64(1), object(3)
memory usage: 500.1+ KB
```

In [40]:
```
# dropping Avg Area Comfort column as it contains more than 40% missing values

re.drop("Avg Area Comfort",axis=1,inplace=True)
```

In [41]:
```
re.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5000 entries, 0 to 4999
Data columns (total 8 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   ids                          5000 non-null   int64  
 1   Avg. Area Income             5000 non-null   float64
 2   Avg. Area House Age          5000 non-null   object 
 3   Avg. Area Number of Rooms    5000 non-null   object 
 4   Avg. Area Number of Bedrooms  3333 non-null   float64
 5   Area Population              5000 non-null   float64
 6   Price                        5000 non-null   int32  
 7   State                        5000 non-null   object 
dtypes: float64(3), int32(1), int64(1), object(3)
memory usage: 461.1+ KB
```

In [42]:
```
re["Avg. Area House Age"].value_counts()
```

Out[42]:
```
missing              5
5.664970500648301    1
5.56057858170534     1
6.877123173691898    1
6.072995728819622    1
                    ..
5.8927024261237495   1
5.640386033581861    1
7.721908810545508    1
6.444268504710308    1
5.992305307333977    1
Name: Avg. Area House Age, Length: 4996, dtype: int64
```

In [43]:
```
re["Avg. Area Number of Rooms"].value_counts()
```

Out[43]:
```
?                   1667
7.666779636732862      1
8.636107722808914      1
7.723659864394269      1
7.083621575990842      1
                     ...
5.816627673636299      1
8.536813721589319      1
6.165414013234222      1
7.297613242970803      1
6.792336104424982      1
Name: Avg. Area Number of Rooms, Length: 3334, dtype: int64
```

In [44]:
```
re["Avg. Area Number of Bedrooms"].value_counts()
```

Out[44]:
```
3.17    31
3.40    31
4.38    30
3.22    29
3.16    28
        ..
6.47     3
6.44     3
5.42     2
5.30     2
6.02     2
Name: Avg. Area Number of Bedrooms, Length: 255, dtype: int64
```

In [45]:
```
re.isna().sum()
```

Out[45]:
```
ids                             0
Avg. Area Income                0
Avg. Area House Age             0
Avg. Area Number of Rooms       0
Avg. Area Number of Bedrooms    1667
Area Population                 0
Price                           0
State                           0
dtype: int64
```

In [46]:
```
# Replace "Missing", "?", "nan" values
```

In [47]:
```
re["Avg. Area House Age"].replace("missing", np.nan, inplace=True)
re["Avg. Area House Age"] = re["Avg. Area House Age"].astype("float64")
```

In [48]:
```
re.dropna(how="all", subset=["Avg. Area House Age"], inplace=True)
```

In [49]:
```
re.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4995 entries, 2 to 4999
Data columns (total 8 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   ids                          4995 non-null   int64  
 1   Avg. Area Income             4995 non-null   float64
 2   Avg. Area House Age          4995 non-null   float64
 3   Avg. Area Number of Rooms    4995 non-null   object 
 4   Avg. Area Number of Bedrooms  3330 non-null   float64
 5   Area Population              4995 non-null   float64
 6   Price                        4995 non-null   int32  
 7   State                        4995 non-null   object 
dtypes: float64(4), int32(1), int64(1), object(2)
memory usage: 331.7+ KB
```

In [50]:
```
re["Avg. Area Number of Rooms"].replace("?", np.nan, inplace=True)
re["Avg. Area Number of Rooms"] = re["Avg. Area Number of Rooms"].astype("float64")
nr = re["Avg. Area Number of Rooms"].mean()
re["Avg. Area Number of Rooms"].fillna(nr, inplace=True)
```

In [51]:
```
re.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4995 entries, 2 to 4999
Data columns (total 8 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   ids                          4995 non-null   int64  
 1   Avg. Area Income             4995 non-null   float64
 2   Avg. Area House Age          4995 non-null   float64
 3   Avg. Area Number of Rooms    4995 non-null   float64
 4   Avg. Area Number of Bedrooms  3330 non-null   float64
 5   Area Population              4995 non-null   float64
 6   Price                        4995 non-null   int32  
 7   State                        4995 non-null   object 
dtypes: float64(5), int32(1), int64(1), object(1)
memory usage: 331.7+ KB
```

In [52]:
```
nb = re["Avg. Area Number of Bedrooms"].mean()
re["Avg. Area Number of Bedrooms"].fillna(nb, inplace=True)
```

In [53]:
```
re.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4995 entries, 2 to 4999
Data columns (total 8 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   ids                          4995 non-null   int64  
 1   Avg. Area Income             4995 non-null   float64
 2   Avg. Area House Age          4995 non-null   float64
 3   Avg. Area Number of Rooms    4995 non-null   float64
 4   Avg. Area Number of Bedrooms  4995 non-null   float64
 5   Area Population              4995 non-null   float64
 6   Price                        4995 non-null   int32  
 7   State                        4995 non-null   object 
dtypes: float64(5), int32(1), int64(1), object(1)
memory usage: 331.7+ KB
```

In [54]:
```
re.head()
```

Out[54]:

	ids	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	State
2	2	61287.067179	5.865890	8.512727	5.130000	36882.159400	1058988	WI
4	4	59982.197226	5.040555	7.839388	4.230000	26354.109472	630943	AE
6	6	64698.463428	6.025336	6.994149	3.998231	60828.249085	1502056	CO
8	8	59927.660813	5.362126	6.393121	2.300000	29387.396003	798870	AA
10	10	80527.472083	8.093513	5.042747	4.100000	47224.359840	1707046	NM

In [55]:
```
re.describe()
```

Out[55]:

	ids	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
count	4995.000000	4995.000000	4995.000000	4995.000000	4995.000000	4995.000000	4.995000e+03
mean	2500.000000	68582.262156	5.976820	6.994149	3.998231	36168.780699	1.232090e+06
std	1443.462435	10657.097347	0.990794	0.829799	1.011379	9924.601840	3.532437e+05
min	1.000000	17796.631190	2.644304	3.236194	2.000000	172.610686	1.593900e+04
25%	1250.500000	61481.724998	5.322350	6.687183	3.360000	29409.079129	9.974510e+05
50%	2500.000000	68803.552077	5.970953	6.994149	3.998231	36200.372388	1.232872e+06
75%	3749.500000	75781.478131	6.650499	7.341102	4.280000	42865.210579	1.471389e+06
max	4999.000000	107701.748378	9.519088	10.219902	6.500000	69621.713378	2.469066e+06

In [56]:
```
# analysis
```

In [57]:
```
re.head()
```

Out[57]:

	ids	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	State
2	2	61287.067179	5.865890	8.512727	5.130000	36882.159400	1058988	WI
4	4	59982.197226	5.040555	7.839388	4.230000	26354.109472	630943	AE
6	6	64698.463428	6.025336	6.994149	3.998231	60828.249085	1502056	CO
8	8	59927.660813	5.362126	6.393121	2.300000	29387.396003	798870	AA
10	10	80527.472083	8.093513	5.042747	4.100000	47224.359840	1707046	NM

In [58]:
```
from sklearn.preprocessing import OrdinalEncoder
oe = OrdinalEncoder()
```

In [59]:
```
colname = re.select_dtypes(object).columns
```

In [60]:
```
re[colname] = oe.fit_transform(re[colname])
```

In [61]:
```
re.head()
```

Out[61]:

	ids	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	State
2	2	61287.067179	5.865890	8.512727	5.130000	36882.159400	1058988	59.0
4	4	59982.197226	5.040555	7.839388	4.230000	26354.109472	630943	1.0
6	6	64698.463428	6.025336	6.994149	3.998231	60828.249085	1502056	9.0
8	8	59927.660813	5.362126	6.393121	2.300000	29387.396003	798870	0.0
10	10	80527.472083	8.093513	5.042747	4.100000	47224.359840	1707046	40.0

In [62]:
```
from sklearn.preprocessing import StandardScaler
```

In [63]:
```
sc = StandardScaler()
```

In [64]:
```
for col in re:
    re[[col]] = sc.fit_transform(re[[col]])
```

In [65]:
```
re.head()
```

Out[65]:

	ids	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	State
2	-1.730734	-0.684607	-0.111972	1.830239e+00	1.119148e+00	0.071887	-0.490084	1.624245
4	-1.729349	-0.807061	-0.945059	1.018709e+00	2.291842e-01	-0.989022	-1.701961	-1.496241
6	-1.727963	-0.364470	0.048972	-2.033877e-14	1.756548e-15	2.484930	0.764326	-1.065829
8	-1.726577	-0.812179	-0.620468	-7.243788e-01	-1.679293e+00	-0.683359	-1.226528	-1.550043
10	-1.725192	1.120981	2.136573	-2.351893e+00	1.006339e-01	1.114068	1.344691	0.602016

In [66]:
```
x = re.iloc[:, [1,2,3,4,5,7]]
x
```

Out[66]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	State
2	-0.684607	-0.111972	1.830239e+00	1.119148e+00	0.071887	1.624245
4	-0.807061	-0.945059	1.018709e+00	2.291842e-01	-0.989022	-1.496241
6	-0.364470	0.048972	-2.033877e-14	1.756548e-15	2.484930	-1.065829
8	-0.812179	-0.620468	-7.243788e-01	-1.679293e+00	-0.683359	-1.550043
10	1.120981	2.136573	-2.351893e+00	1.006339e-01	1.114068	0.602016
...	...	...	...	...	...	...
4991	0.518010	-0.321975	8.314222e-01	-8.585491e-01	-1.222087	-0.689219
4993	0.099181	-0.978414	9.451744e-01	2.028888e+00	1.802504	-0.474013
4995	-0.752092	1.870951	-2.033877e-14	1.756548e-15	-1.343404	-1.334837
4997	-0.487196	1.285734	-2.638336e+00	-1.847398e+00	-0.292498	1.409039
4999	-0.288257	0.015631	-2.432319e-01	7.096843e-02	1.041204	0.655818

4995 rows × 6 columns

In [67]:
```
y = re.iloc[:, -2]
y
```

Out[67]:
```
2      -0.490084
4      -1.701961
6       0.764326
8      -1.226528
10      1.344691
           ...
4991    0.089554
4993    0.423946
4995   -0.486670
4997   -0.570088
4999    0.189294
Name: Price, Length: 4995, dtype: float64
```

In [68]:
```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest = train_test_split(x,y, test_size=0.3, random_state=1)
```

In [69]:
```
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
linreg.fit(xtrain,ytrain)
ypred = linreg.predict(xtest)
```

In [70]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
mae = mean_absolute_error(ytest,ypred)
mse = mean_squared_error(ytest,ypred)
rmse = np.sqrt(mse)
r2 = r2_score(ytest,ypred)

print(f"MAE: {mae}\nMSE: {mse}\nRMSE: {rmse}\nAccuracy: {r2}")
```

```
MAE: 0.2723569454504594
MSE: 0.12267016549506408
RMSE: 0.35024300920227386
Accuracy: 0.8785191244557695
```

In [71]:
```
linreg.score(xtrain,ytrain)
```

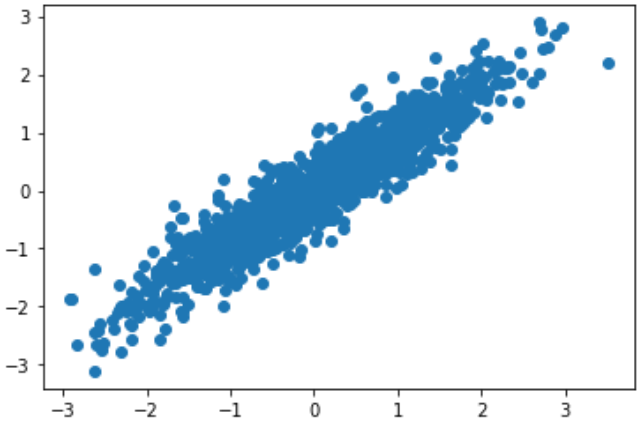
Out[71]:
```
0.8809550998471474
```

In [72]:
```
linreg.score(xtest,ytest)
```

Out[72]:
```
0.8785191244557695
```

In [73]:
```
plt.scatter(ytest, ypred)
```

Out[73]:
```
<matplotlib.collections.PathCollection at 0x264bd0c6370>
```



In [74]:
```
from sklearn.svm import SVR
svm = SVR()
svm.fit(xtrain,ytrain)
ypred = svm.predict(xtest)
```

In [75]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
mae = mean_absolute_error(ytest,ypred)
mse = mean_squared_error(ytest,ypred)
rmse = np.sqrt(mse)
r2 = r2_score(ytest,ypred)

print(f"MAE: {mae}\nMSE: {mse}\nRMSE: {rmse}\nAccuracy: {r2}")
```

```
MAE: 0.2849301405137077
MSE: 0.13520152199824298
RMSE: 0.3676975958559465
Accuracy: 0.8661092597293352
```

In [76]:
```
from sklearn.model_selection import GridSearchCV
param_grid ={'C':[0.1,1,10,100,1000],
             'gamma':[1,0.1,0.01,0.001,0.0001],
             'kernel':['rbf']}

grid = GridSearchCV(SVR(),param_grid,refit=True,verbose=3)
grid.fit(xtrain,ytrain)
```

```
[CV 4/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.573 total time=   0.7s
[CV 5/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.574 total time=   0.7s
[CV 1/5] END .........C=10, gamma=1, kernel=rbf;, score=0.691 total time=   1.4s
[CV 2/5] END .........C=10, gamma=1, kernel=rbf;, score=0.673 total time=   1.4s
[CV 3/5] END .........C=10, gamma=1, kernel=rbf;, score=0.660 total time=   1.5s
[CV 4/5] END .........C=10, gamma=1, kernel=rbf;, score=0.677 total time=   1.5s
[CV 5/5] END .........C=10, gamma=1, kernel=rbf;, score=0.643 total time=   1.4s
[CV 1/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.862 total time=   1.1s
[CV 2/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.874 total time=   1.1s
[CV 3/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.853 total time=   1.2s
[CV 4/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.876 total time=   1.2s
[CV 5/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.860 total time=   1.1s
[CV 1/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.870 total time=   0.6s
[CV 2/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.891 total time=   0.6s
[CV 3/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.871 total time=   0.6s
[CV 4/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.885 total time=   0.6s
[CV 5/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.875 total time=   0.6s
[CV 1/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.874 total time=   0.6s
[CV 2/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.892 total time=   0.6s
```

In [77]:
```
grid.best_score_
```

Out[77]:
```
0.8802060762888771
```

In [78]:
```
grid.best_estimator_
```

Out[78]:
```
SVR(C=100, gamma=0.0001)
```

In [79]:
```
grid.best_params_
```

Out[79]:
```
{'C': 100, 'gamma': 0.0001, 'kernel': 'rbf'}
```

In [80]:
```
lr = grid.best_estimator_
lr.fit(xtrain,ytrain)
ypred = lr.predict(xtest)
print(r2_score(ytest,ypred))
```

```
0.8785286041938025
```

In [81]:
```
from sklearn.svm import SVR
svm = SVR(C= 100, gamma= 0.0001, kernel= 'rbf')
svm.fit(xtrain,ytrain)
ypred = svm.predict(xtest)
```

In [82]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
mae = mean_absolute_error(ytest,ypred)
mse = mean_squared_error(ytest,ypred)
rmse = np.sqrt(mse)
r2 = r2_score(ytest,ypred)

print(f"MAE: {mae}\nMSE: {mse}\nRMSE: {rmse}\nAccuracy: {r2}")
```

```
MAE: 0.27246407015152097
MSE: 0.12266059295100604
RMSE: 0.35022934336089834
Accuracy: 0.8785286041938025
```

In [ ]:
```

```