lab-4

September 24, 2024

1.

PART 1

[4]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[5]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[6]:
```
file_path='/content/drive/MyDrive/USA_Housing.csv'
```

[7]:
```
df=pd.read_csv(file_path)
df.head()
```

[7]:
```
   Avg. Area Income  Avg. Area House Age  Avg. Area Number of Rooms  \
0      79545.458574             5.682861                   7.009188
1      79248.642455             6.002900                   6.730821
2      61287.067179             5.865890                   8.512727
3      63345.240046             7.188236                   5.586729
4      59982.197226             5.040555                   7.839388

   Avg. Area Number of Bedrooms  Area Population         Price  \
0                          4.09     23086.800503  1.059034e+06
1                          3.09     40173.072174  1.505891e+06
2                          5.13     36882.159400  1.058988e+06
3                          3.26     34310.242831  1.260617e+06
4                          4.23     26354.109472  6.309435e+05

                                             Address
0  208 Michael Ferry Apt. 674\nLaurabury, NE 3701…
1  188 Johnson Views Suite 079\nLake Kathleen, CA…
2  9127 Elizabeth Stravenue\nDanieltown, WI 06482…
3                          USS Barnett\nFPO AP 44820
4                         USNS Raymond\nFPO AE 09386
```

[8]:
```
df.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   Avg. Area Income              5000 non-null   float64
 1   Avg. Area House Age           5000 non-null   float64
 2   Avg. Area Number of Rooms     5000 non-null   float64
 3   Avg. Area Number of Bedrooms  5000 non-null   float64
 4   Area Population               5000 non-null   float64
 5   Price                         5000 non-null   float64
 6   Address                       5000 non-null   object
dtypes: float64(6), object(1)
memory usage: 273.6+ KB
```

[9]:
```
df.describe(percentiles=[0.1,0.25,0.5,0.75,0.9])
```

[9]:
```
       Avg. Area Income  Avg. Area House Age  Avg. Area Number of Rooms  \
count       5000.000000          5000.000000                5000.000000
mean       68583.108984             5.977222                   6.987792
std        10657.991214             0.991456                   1.005833
min        17796.631190             2.644304                   3.236194
10%        55047.633980             4.697755                   5.681951
25%        61480.562388             5.322283                   6.299250
50%        68804.286404             5.970429                   7.002902
75%        75783.338666             6.650808                   7.665871
90%        82081.188283             7.243978                   8.274222
max       107701.748378             9.519088                  10.759588

       Avg. Area Number of Bedrooms  Area Population         Price
count                   5000.000000      5000.000000  5.000000e+03
mean                       3.981330     36163.516039  1.232073e+06
std                        1.234137      9925.650114  3.531176e+05
min                        2.000000       172.610686  1.593866e+04
10%                        2.310000     23502.845262  7.720318e+05
25%                        3.140000     29403.928702  9.975771e+05
50%                        4.050000     36199.406689  1.232669e+06
75%                        4.490000     42861.290769  1.471210e+06
90%                        6.100000     48813.618633  1.684621e+06
max                        6.500000     69621.713378  2.469066e+06
```

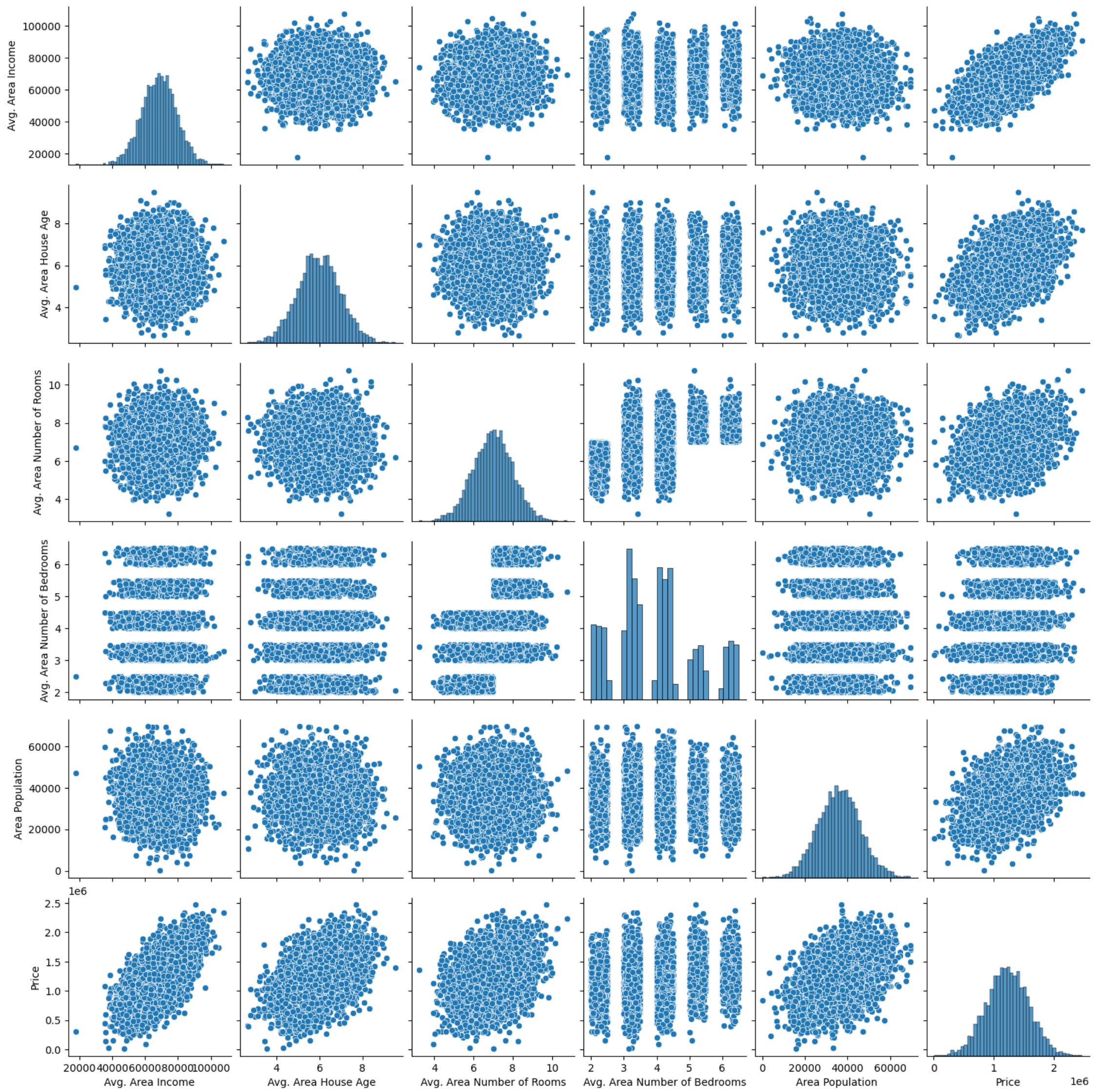
[10]:
```
df.columns
```

[10]:
```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Address'],
```

```
dtype='object')
```

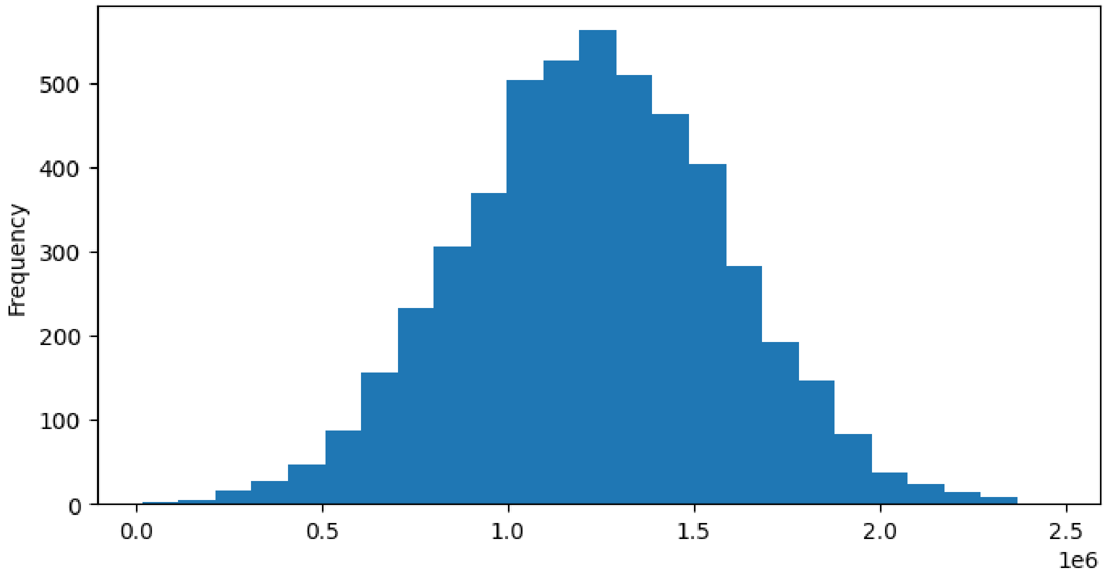
[11]:
```
sns.pairplot(df)
```

[11]: <seaborn.axisgrid.PairGrid at 0x78ec3ef33a30>



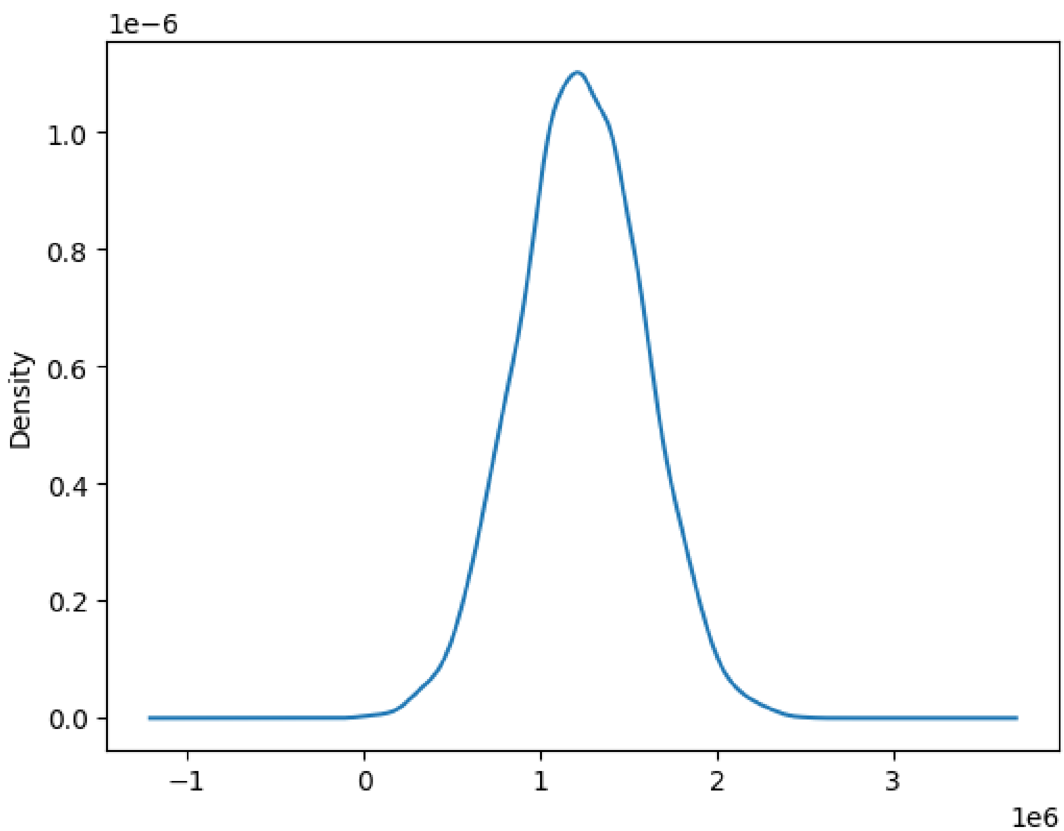
[12]:
```
df['Price'].plot.hist(bins=25,figsize=(8,4))
```

[12]: <Axes: ylabel='Frequency'>



[13]:
```
df['Price'].plot.density()
```

[13]: <Axes: ylabel='Density'>



[14]:
```python
df_cleaned=df.drop(columns=['Address'])
```

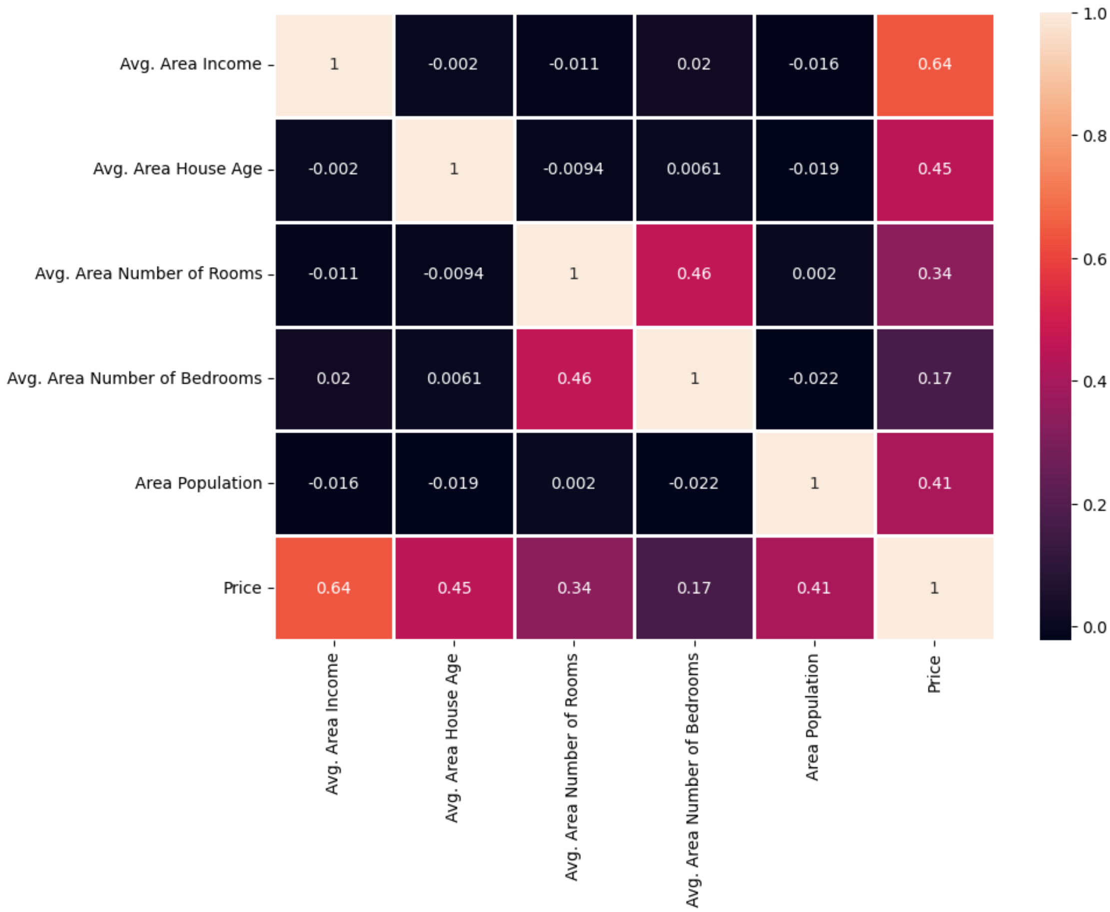
[15]:
```python
df_cleaned.corr()
```

[15]:

```
                              Avg. Area Income  Avg. Area House Age  \
Avg. Area Income                      1.000000            -0.002007
Avg. Area House Age                  -0.002007             1.000000
Avg. Area Number of Rooms            -0.011032            -0.009428
Avg. Area Number of Bedrooms          0.019788             0.006149
Area Population                      -0.016234            -0.018743
Price                                 0.639734             0.452543

                              Avg. Area Number of Rooms  \
Avg. Area Income                              -0.011032
Avg. Area House Age                           -0.009428
Avg. Area Number of Rooms                      1.000000
Avg. Area Number of Bedrooms                   0.462695
Area Population                                0.002040
Price                                          0.335664

                              Avg. Area Number of Bedrooms  Area Population  \
Avg. Area Income                                  0.019788        -0.016234
Avg. Area House Age                               0.006149        -0.018743
Avg. Area Number of Rooms                         0.462695         0.002040
Avg. Area Number of Bedrooms                      1.000000        -0.022168
Area Population                                  -0.022168         1.000000
Price                                             0.171071         0.408556

                                 Price
Avg. Area Income              0.639734
Avg. Area House Age           0.452543
Avg. Area Number of Rooms     0.335664
Avg. Area Number of Bedrooms  0.171071
Area Population               0.408556
Price                         1.000000
```

[16]:
```python
plt.figure(figsize=(10,7))
sns.heatmap(df_cleaned.corr(),annot=True,linewidths=2)
```

[16]: <Axes: >



[17]:
```
l_column=list(df.columns)
len_feature=len(l_column)
l_column
```

[17]:
```
['Avg. Area Income',
 'Avg. Area House Age',
 'Avg. Area Number of Rooms',
 'Avg. Area Number of Bedrooms',
 'Area Population',
 'Price',
 'Address']
```

[18]:
```
x=df[l_column[0:len_feature-2]]
y=df[l_column[len_feature-2]]
```

[19]:
```
print("feature set size:",x.shape)
print("Variable set size:",y.shape)
```

```
feature set size: (5000, 5)
```

```
Variable set size: (5000,)
```

[20]:
```python
x.head()
```

[20]:
```
   Avg. Area Income  Avg. Area House Age  Avg. Area Number of Rooms  \
0      79545.458574             5.682861                   7.009188
1      79248.642455             6.002900                   6.730821
2      61287.067179             5.865890                   8.512727
3      63345.240046             7.188236                   5.586729
4      59982.197226             5.040555                   7.839388

   Avg. Area Number of Bedrooms  Area Population
0                          4.09     23086.800503
1                          3.09     40173.072174
2                          5.13     36882.159400
3                          3.26     34310.242831
4                          4.23     26354.109472
```

PART 2

[21]:
```python
from sklearn.model_selection import train_test_split
```

[22]:
```python
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.
 ↪3,random_state=123)
```

[23]:
```python
print("Training feature set size:",x_train.shape)
print("Test feature set size:",x_test.shape)
print("Training variable set size:",y_train.shape)
print("Test variable set size:",y_test.shape)
```

```
Training feature set size: (3500, 5)
Test feature set size: (1500, 5)
Training variable set size: (3500,)
Test variable set size: (1500,)
```

[24]:
```python
from sklearn.linear_model import LinearRegression
from sklearn import metrics
```

[25]:
```python
lm=LinearRegression()
```

[26]:
```python
lm.fit(x_train,y_train)
```

[26]:
```
LinearRegression()
```

[27]:
```python
print("The intercept term of the linear model:",lm.intercept_)
```

```
The intercept term of the linear model: -2631028.9017454907
```

[28]:
```python
print("The coefficients of the linear model:",lm.coef_)
```

```
The coefficients of the linear model: [2.15976020e+01 1.65201105e+05
1.19061464e+05 3.21258561e+03
 1.52281212e+01]
```

[29]:
```python
cdf=pd.DataFrame(data=lm.coef_,index=x_train.columns,columns=['Coefficients'])
cdf
```

[29]:

	Coefficients
Avg. Area Income	21.597602
Avg. Area House Age	165201.104954
Avg. Area Number of Rooms	119061.463868
Avg. Area Number of Bedrooms	3212.585606
Area Population	15.228121

PART 3

[30]:
```python
n=x_train.shape[0]
k=x_train.shape[1]
dfN = n-k
train_pred=lm.predict(x_train)
train_error = np.square(train_pred-y_train)
sum_error=np.sum(train_error)
se=[0,0,0,0,0]
for i in range(k):
  r = (sum_error/dfN)
  r = r/np.sum(np.square(x_train[list(x_train.columns)[i]]-x_train[list(x_train.
↪columns)[i]].mean()))
  se[i]=np.sqrt(r)
cdf[ 'Standard Error']=se
cdf['t-statistic']=cdf[ 'Coefficients']/cdf['Standard Error']
cdf
```

[30]:

	Coefficients	Standard Error	t-statistic
Avg. Area Income	21.597602	0.160361	134.681505
Avg. Area House Age	165201.104954	1722.412068	95.912649
Avg. Area Number of Rooms	119061.463868	1696.546476	70.178722
Avg. Area Number of Bedrooms	3212.585606	1376.451759	2.333962
Area Population	15.228121	0.169882	89.639472

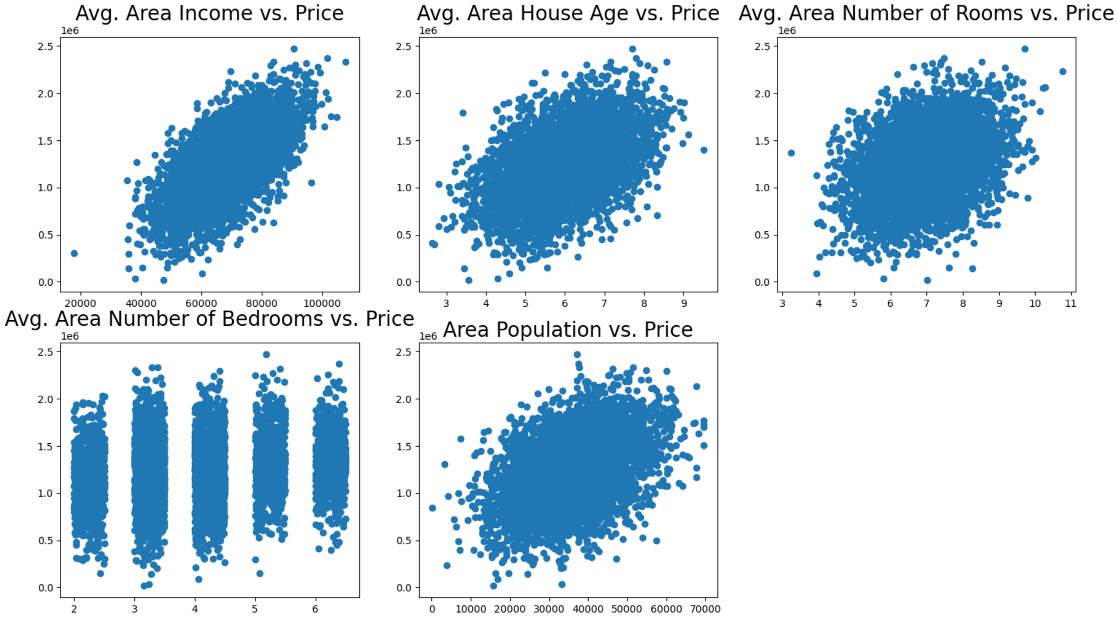
[31]:
```python
print("Therefore, features arranged in the order of importance for predicting␣
↪the house price",'-'*90,sep='')
l=list(cdf.sort_values('t-statistic',ascending=False).index)
print('>\n'.join(l))
```

```
Therefore, features arranged in the order of importance for predicting the house
price------------------------------------------------------------------------------------------
```

```
---------------
Avg. Area Income>
Avg. Area House Age>
Area Population>
Avg. Area Number of Rooms>
Avg. Area Number of Bedrooms
```

[32]:
```python
l=list(cdf.index)
from matplotlib import gridspec
fig = plt.figure(figsize=(18, 10))
gs = gridspec.GridSpec(2,3)
#f, ax = plt.subplots(nrows=1,ncols=len(l), sharey=True)
ax0 = plt.subplot(gs[0])
ax0.scatter(df[l[0]],df['Price'])
ax0.set_title(l[0]+" vs. Price", fontdict={'fontsize':20})

ax1 = plt.subplot(gs[1])
ax1.scatter(df[l[1]],df['Price'])
ax1.set_title(l[1]+" vs. Price",fontdict={'fontsize':20})

ax2 = plt.subplot(gs[2])
ax2.scatter(df[l[2]],df['Price'])
ax2.set_title(l[2]+" vs. Price",fontdict={'fontsize':20})

ax3 = plt.subplot(gs[3])
ax3.scatter(df[l[3]],df['Price'])
ax3.set_title(l[3]+" vs. Price",fontdict={'fontsize':20})

ax4 = plt.subplot(gs[4])
ax4.scatter(df[l[4]],df['Price'])
ax4.set_title(l[4]+" vs. Price",fontdict={'fontsize':20})
```

[32]:
```
Text(0.5, 1.0, 'Area Population vs. Price')
```



[33]:
```
print("R-squared value of this fit:",round(metrics.
 ↪r2_score(y_train,train_pred),3))
```

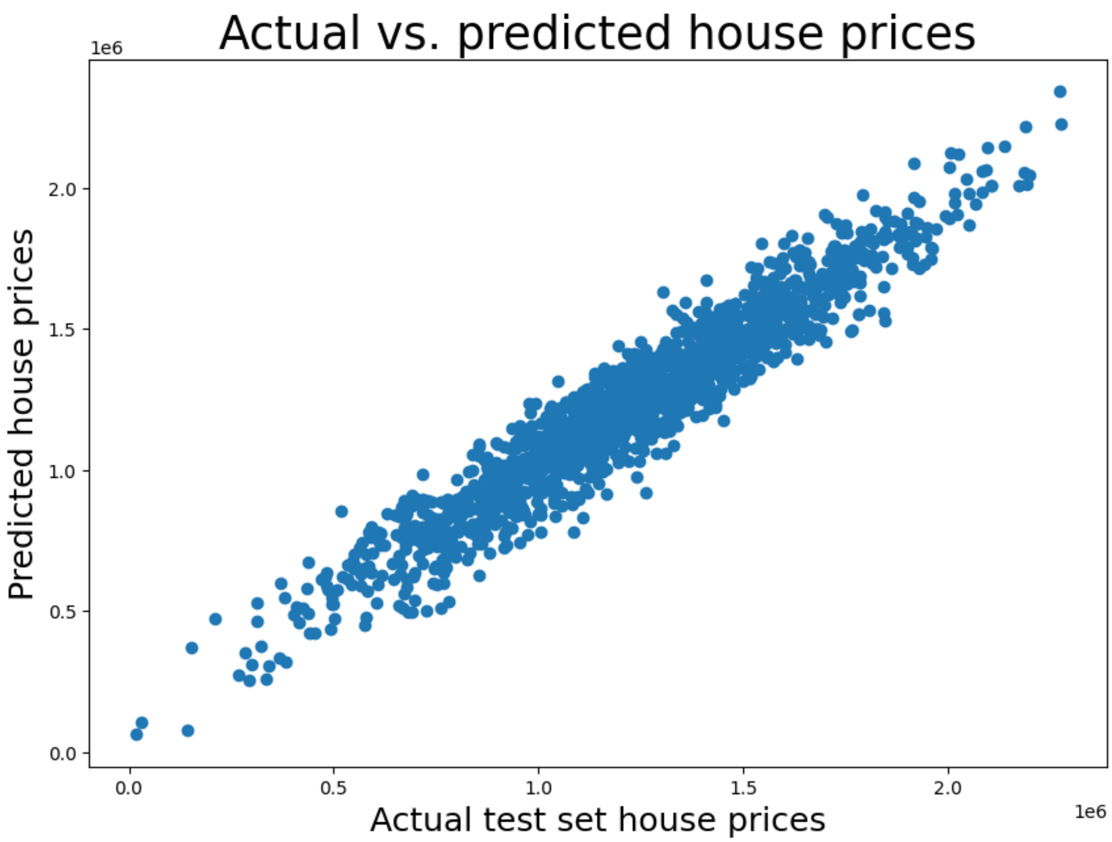
```
R-squared value of this fit: 0.917
```

[35]:
```
predictions = lm.predict(x_test)
print ("Type of the predicted object:", type(predictions))
print ("Size of the predicted object:", predictions.shape)
```

```
Type of the predicted object: <class 'numpy.ndarray'>
Size of the predicted object: (1500,)
```

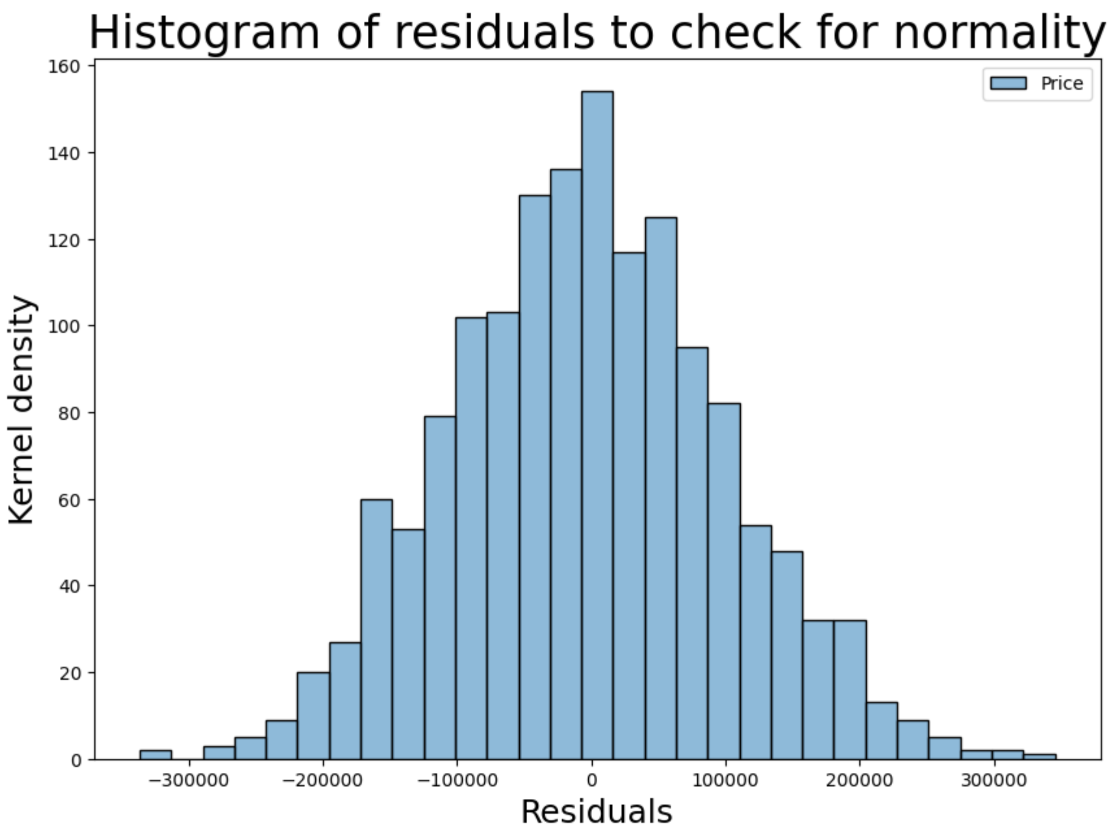
[36]:
```
plt.figure(figsize=(10,7))
plt.title("Actual vs. predicted house prices",fontsize=25)
plt.xlabel("Actual test set house prices",fontsize=18)
plt.ylabel("Predicted house prices", fontsize=18)
plt.scatter(x=y_test,y=predictions)
```

[36]:
```
<matplotlib.collections.PathCollection at 0x78ec30cec250>
```



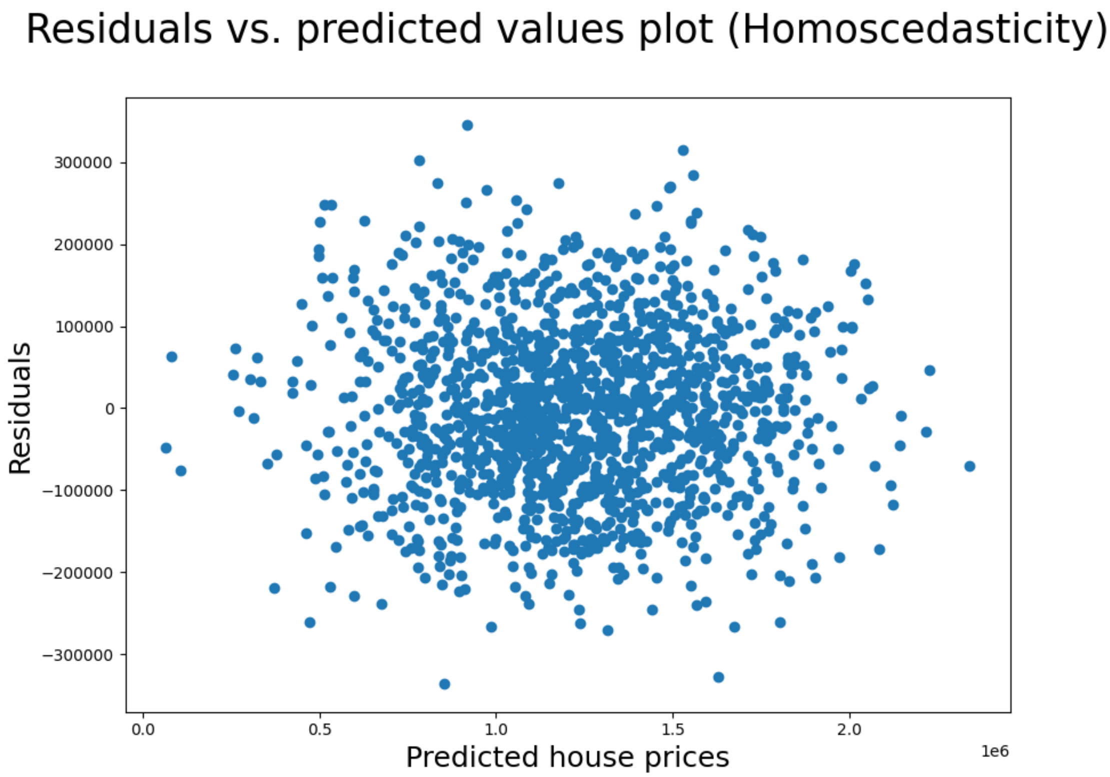
[37]:
```
plt.figure(figsize=(10,7))
plt.title("Histogram of residuals to check for normality",fontsize=25)
plt.xlabel("Residuals",fontsize=18)
plt.ylabel("Kernel density", fontsize=18)
sns.histplot([y_test-predictions])
```

[37]:
```
<Axes: title={'center': 'Histogram of residuals to check for normality'},
xlabel='Residuals', ylabel='Kernel density'>
```



[38]:
```
plt.figure(figsize=(10,7))
plt.title("Residuals vs. predicted values plot␣
  ↪(Homoscedasticity)\n",fontsize=25)
plt.xlabel("Predicted house prices",fontsize=18)
plt.ylabel("Residuals", fontsize=18)
plt.scatter(x=predictions,y=y_test-predictions)
```

[38]: <matplotlib.collections.PathCollection at 0x78ec30a23730>



[39]:
```
print("Mean absolute error (MAE):", metrics.
  ↪mean_absolute_error(y_test,predictions))
print("Mean square error (MSE):", metrics.
  ↪mean_squared_error(y_test,predictions))
print("Root mean square error (RMSE):", np.sqrt(metrics.
  ↪mean_squared_error(y_test,predictions)))
```

```
Mean absolute error (MAE): 81739.77482718184
Mean square error (MSE): 10489638335.804983
Root mean square error (RMSE): 102418.93543581179
```

[40]:
```
print("R-squared value of predictions:",round(metrics.
  ↪r2_score(y_test,predictions),3))
```

```
R-squared value of predictions: 0.919
```

2.

[41]:
```
#compute minmax value for observed price and expected price
import numpy as np
min=np.min(predictions/6000)
max=np.max(predictions/12000)
print(min, max)
```

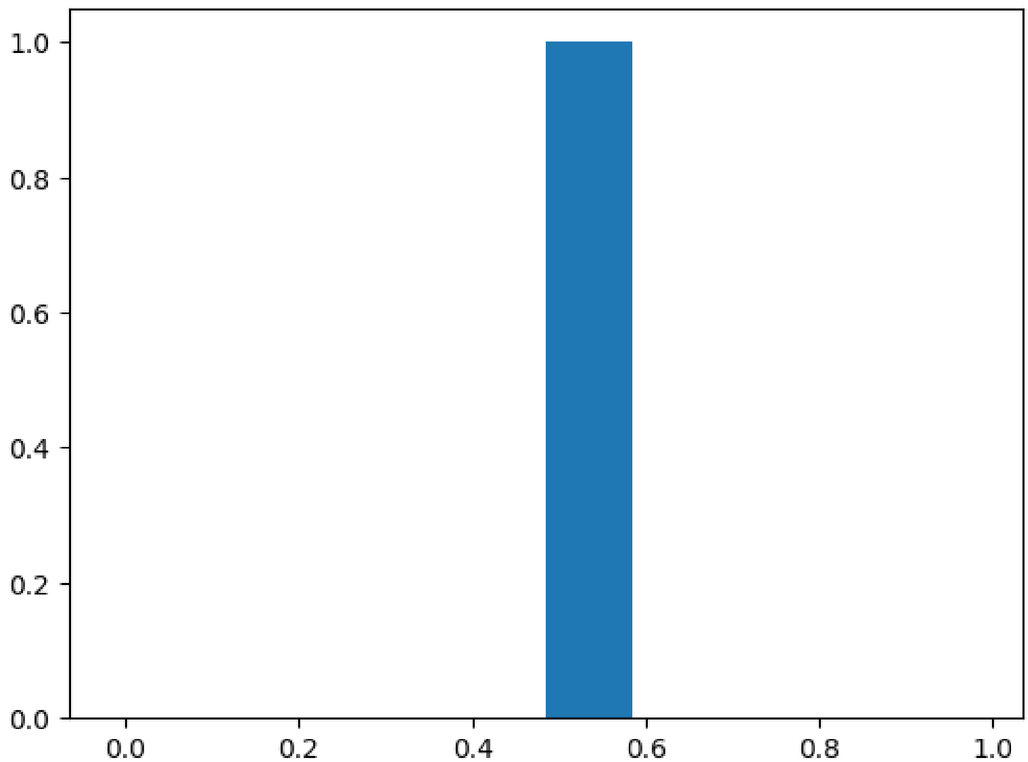
```
10.57339854753646 195.14363973516853
```

[42]:

```
#Compute MinMax value for Price=100
L = (100 - min)/(max - min)
L
plt.hist(L)
```

[42]:

```
(array([0., 0., 0., 0., 0., 1., 0., 0., 0., 0.]),
 array([-0.01548743,  0.08451257,  0.18451257,  0.28451257,  0.38451257,
         0.48451257,  0.58451257,  0.68451257,  0.78451257,  0.88451257,
         0.98451257]),
 <BarContainer object of 10 artists>)
```



[ ]: