Double-click (or enter) to edit

Double-click (or enter) to edit

Implement linear regression model using US housing model

```
import numpy as np
import pandas as dp
import matplotlib.pyplot as plt
housingdata=dp.read_csv("/cars.csv")
housingdata.head()
housingdata.tail()
```

	model	year	price	transmission	mileage	fuelType	tax	mpg	engineSize
17960	Fiesta	2016	7999	Manual	31348	Petrol	125	54.3	1.2
17961	B-MAX	2017	8999	Manual	16700	Petrol	150	47.1	1.4
17962	B-MAX	2014	7499	Manual	40700	Petrol	30	57.7	1.0
17963	Focus	2015	9999	Manual	7010	Diesel	20	67.3	1.6
17964	KA	2018	8299	Manual	5007	Petrol	145	57.7	1.2

```
housingdata.mean()
```

```
<ipython-input-14-e898f1e406af>:1: FutureWarning: The default value of numeric_only in DataFrame.mean is deprecated. In a future version
  housingdata.mean()
year           2016.866574
price         12279.756415
mileage       23363.630504
tax             113.334539
mpg              57.906991
engineSize        1.350827
dtype: float64
```

```
housingdata.mean(axis=1)
```

```
<ipython-input-15-291e13d1e3e1>:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is dep
  housingdata.mean(axis=1)
0        5028.283333
1        4218.283333
2        4613.616667
3        5027.633333
4        3365.950000
            ...
17960    6923.916667
17961    4652.416667
17962    8383.616667
17963    3185.483333
17964    2587.983333
Length: 17965, dtype: float64
```

Double-click (or enter) to edit

```
housingdata.median()
```

```
<ipython-input-16-e56fae5ee9b1>:1: FutureWarning: The default value of numeric_only in DataFrame.median is deprecated. In a future versi
  housingdata.median()
year           2017.0
price         11291.0
mileage       18243.0
tax             145.0
mpg              58.9
engineSize        1.2
dtype: float64
```

```
housingdata.mode()
```


	model	year	price	transmission	mileage	fuelType	tax	mpg	engineSize
0	Fiesta	2017	10000	Manual	10	Petrol	145	65.7	1.0

```
housingdata.max()
```

```
model            Transit Tourneo
year                        2060
price                      54995
transmission          Semi-Auto
mileage                   177644
fuelType                  Petrol
tax                          580
mpg                        201.8
engineSize                   5.0
dtype: object
```

```
housingdata.min()
```

```
model              B-MAX
year                1996
price                495
transmission    Automatic
mileage                 1
fuelType           Diesel
tax                     0
mpg                  20.8
engineSize            0.0
dtype: object
```

```
max(housingdata["price"])
```

```
54995
```

```
max(housingdata["price"]) - min(housingdata["tax"])
```

```
54995
```

```
fuelType          Diesel
tax                    0
mpg                 20.8
engineSize           0.0
dtype: object
```

```
max(housingdata["price"])
```

```
54995
```

```
max(housingdata["price"]) - min(housingdata["tax"])
```

```
54995
```

```
five_num=[housingdata["price"].quantile(0),
          housingdata["price"].quantile(0.25),
          housingdata["price"].quantile(0.50),
          housingdata["price"].quantile(0.75),
          housingdata["price"].quantile(0.9)]
five_num
```

```
[495.0, 8999.0, 11291.0, 15299.0, 18398.6]
```

```
import statistics
statistics.stdev(housingdata["year"])
```

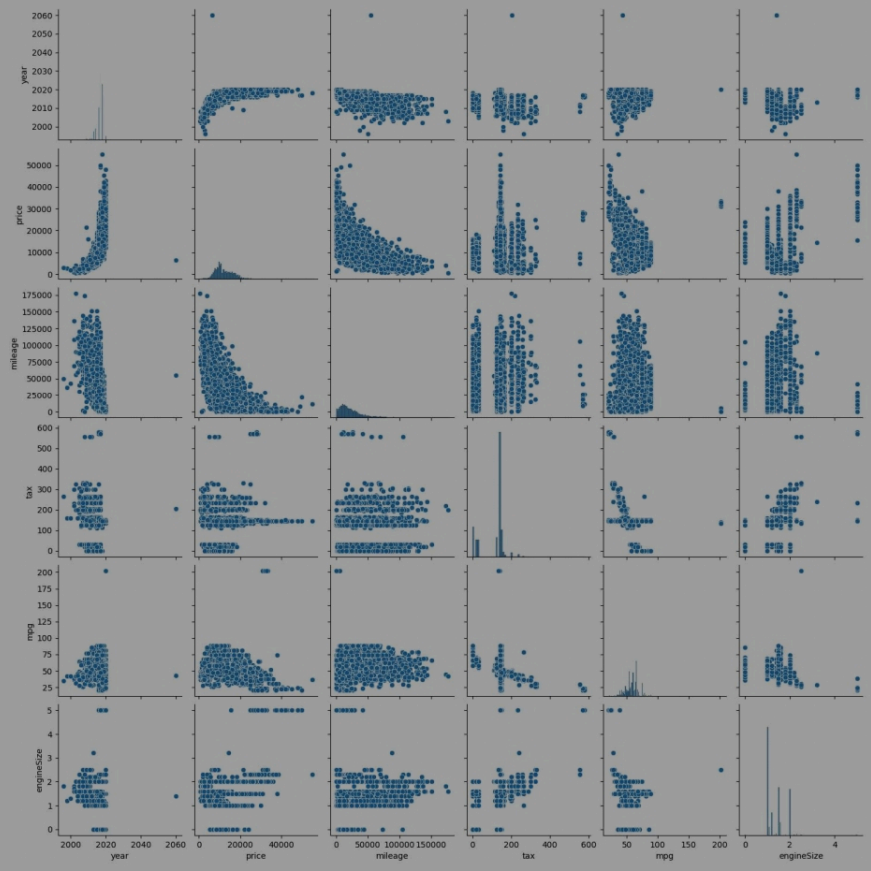
```
2.0503457521500708
```

```
import seaborn as sns

import matplotlib.pyplot as plt

car = pd.read_csv("/cars.csv")

sns.pairplot(car)
plt.show()
```

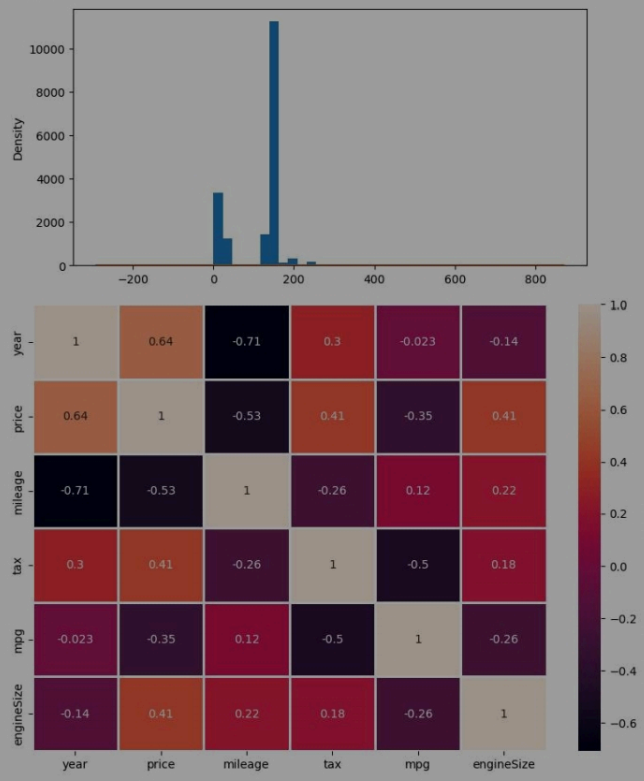



Split the data generated from list created as X, Y is distributed using train test split function as X train, Y train, X test, Y test

Split the data generated from list created as X, Y is distributed using train test split function as X train, Y train, X test, Y test


```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
housingdata = pd.read_csv("/content/cars.csv")
housingdata['tax'].plot.hist(bins=25,figsize=(8,4))
housingdata['tax'].plot.density()
housingdata.corr()
plt.figure(figsize=(10,7))
sns.heatmap(housingdata.corr(),annot=True,linewidths=2)
```

```
<ipython-input-17-6fbae54e4dc6>:8: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version
  housingdata.corr()
<ipython-input-17-6fbae54e4dc6>:10: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future versic
  sns.heatmap(housingdata.corr(),annot=True,linewidths=2)
<Axes: >
```



Apply the linear regression model of sklearn package


```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
df = pd.read_csv("/content/cars.csv")
df.head()
housingdata_column = list(df.columns) # Making a list out of column names
len_feature = len(housingdata_column) # Length of column vector list
housingdata_column
```

```
['model',
 'year',
 'price',
 'transmission',
 'mileage',
 'fuelType',
 'tax',
 'mpg',
 'engineSize']
```

Fit the data to the Linear Model using fit

```
from sklearn.linear_model import LinearRegression
from sklearn import metrics
lm = LinearRegression()
lm.fit(X_train,y_train)
print("The intercept term of the linear model:", lm.intercept_)
print("The coefficients of the linear model:", lm.coef_)
```

```
Training feature set size: (12575, 7)
```

50%	68804.286404	5.970429	7.002902	4.050000	36199.406689	1.232669e+06
75%	75783.338666	6.650808	7.665871	4.490000	42861.290769	1.471210e+06
90%	82081.188283	7.243978	8.274222	6.100000	48813.618633	1.684621e+06
max	107701.748378	9.519088	10.759588	6.500000	69621.713378	2.469066e+06

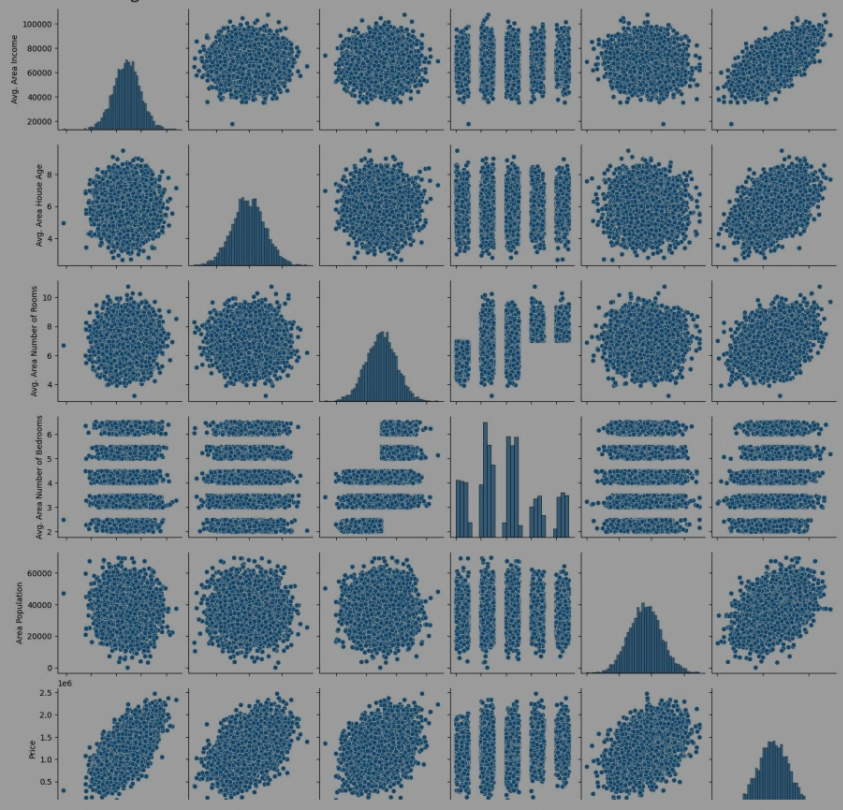
```
df.columns
```

```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Address'],
      dtype='object')
```

```
sns.pairplot(df)
```

2

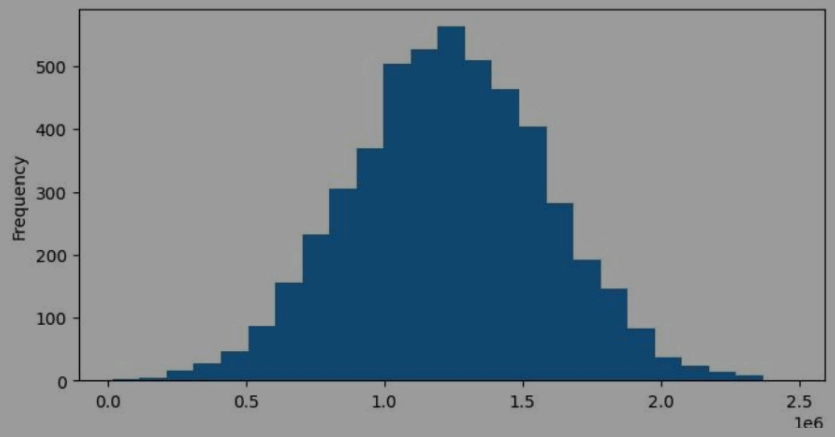

```
<seaborn.axisgrid.PairGrid at 0x7fcbb5477550>
```




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

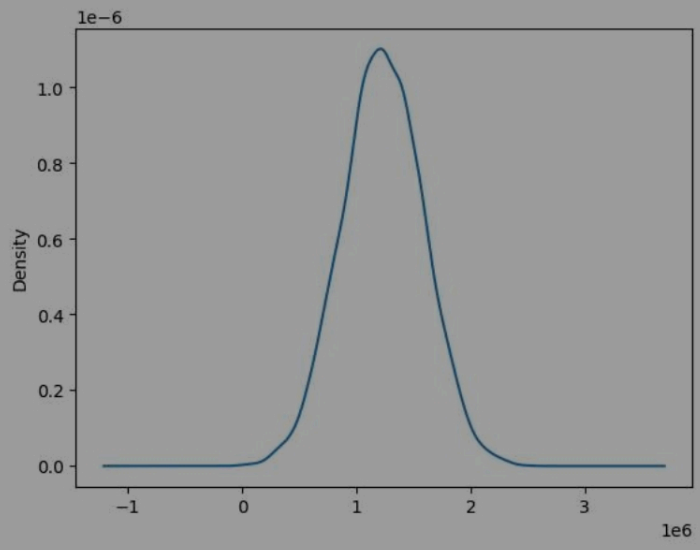
3

```
<Axes: ylabel='Frequency'>
```




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

```
<Axes: ylabel='Density'>
```




```
df.corr()
```



```
<ipython-input-9-2f6f6606aa2c>:1: FutureWarning: The default value of numeric_only in Da
  df.corr()
```

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
Avg. Area Income	1.000000	-0.002007	-0.011032	0.019788	-0.016234	0.639734
Avg. Area House Age	-0.002007	1.000000	-0.009428	0.006149	-0.018743	0.452543
Avg. Area Number of Rooms	-0.011032	-0.009428	1.000000	0.462695	0.002040	0.335664
Avg. Area Number of Bedrooms	0.019788	0.006149	0.462695	1.000000	-0.022168	0.171071
Area Population	-0.016234	-0.018743	0.002040	-0.022168	1.000000	0.408556
Price	0.639734	0.452543	0.335664	0.171071	0.408556	1.000000

```
df.corr()
```



```
<ipython-input-9-2f6f6606aa2c>:1: FutureWarning: The default value of numeric_only in Da
  df.corr()
```

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
Avg. Area Income	1.000000	-0.002007	-0.011032	0.019788	-0.016234	0.639734
Avg. Area House Age	-0.002007	1.000000	-0.009428	0.006149	-0.018743	0.452543
Avg. Area Number of Rooms	-0.011032	-0.009428	1.000000	0.462695	0.002040	0.335664
Avg. Area Number of Bedrooms	0.019788	0.006149	0.462695	1.000000	-0.022168	0.171071
Area Population	-0.016234	-0.018743	0.002040	-0.022168	1.000000	0.408556
Price	0.639734	0.452543	0.335664	0.171071	0.408556	1.000000

```
plt.figure(figsize=(10,7))
sns.heatmap(df.corr(),annot=True,linewidths=2)
```



```
<ipython-input-10-73d88c5a3f1a>:2: FutureWarning: The default value of numeric_only in D
  sns.heatmap(df.corr(),annot=True,linewidths=2)
<Axes: >
```

```
l_column = list(df.columns) # Making a list out of column names
len_feature = len(l_column) # Length of column vector list
l_column
```

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

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



```
print("Feature set size:",X.shape)
print("Variable set size:",y.shape)
```

```
Feature set size: (5000, 5)
Variable set size: (5000,)
```

```
X.head()
```

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population
0	79545.458574	5.682861	7.009188	4.09	23086.800503
1	79248.642455	6.002900	6.730821	3.09	40173.072174
2	61287.067179	5.865890	8.512727	5.13	36882.159400
3	63345.240046	7.188236	5.586729	3.26	34310.242831
4	59982.197226	5.040555	7.839388	4.23	26354.109472

Next steps: View recommended plots

```
y.head()
```

```
0    1.059034e+06
1    1.505891e+06
2    1.058988e+06
3    1.260617e+06
4    6.309435e+05
Name: Price, dtype: float64
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                    test_size=0.3, random_state=123)

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,)
```



```
from sklearn.linear_model import LinearRegression
from sklearn import metrics

lm = LinearRegression() # Creating a Linear Regression object 'lm'

lm.fit(X_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
The coefficients of the linear model: [2.15976020e+01 1.65201105e+05 1.19061464e+05 3.21
 1.52281212e+01]
```

```
cdf = pd.DataFrame(data=lm.coef_, index=X_train.columns, columns=["Coefficients"])
#cdf=pd.concat([idf,cdf], axis=0)
cdf
```

	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


```
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
```

	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

Next steps: View recommended plots

```
print("Therefore, features arranged in the order of importance for predicting the house pric
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
```

```
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])
```

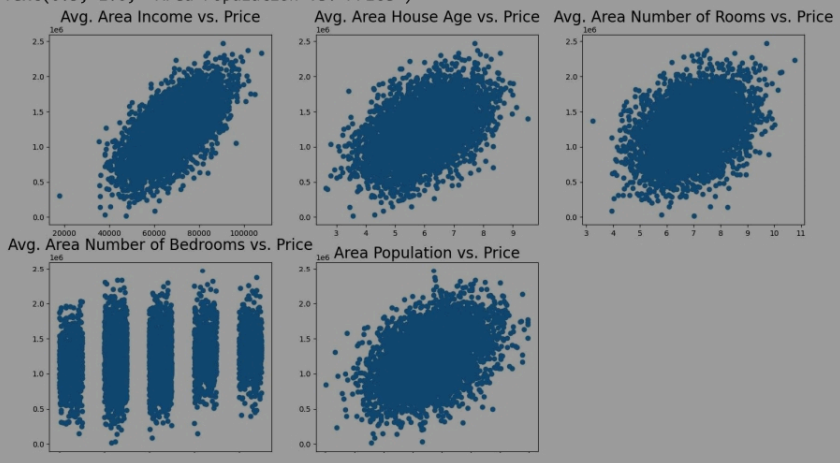
```
ax1 = plt.subplot(gs[1])
ax1.scatter(df[l[1]],df['Price'])
ax1.set_title(l[1]+" vs. Price",fontdict={'fontsize':20})

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})
```

```
Text(0.5, 1.0, 'Area Population vs. Price')
```




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

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

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

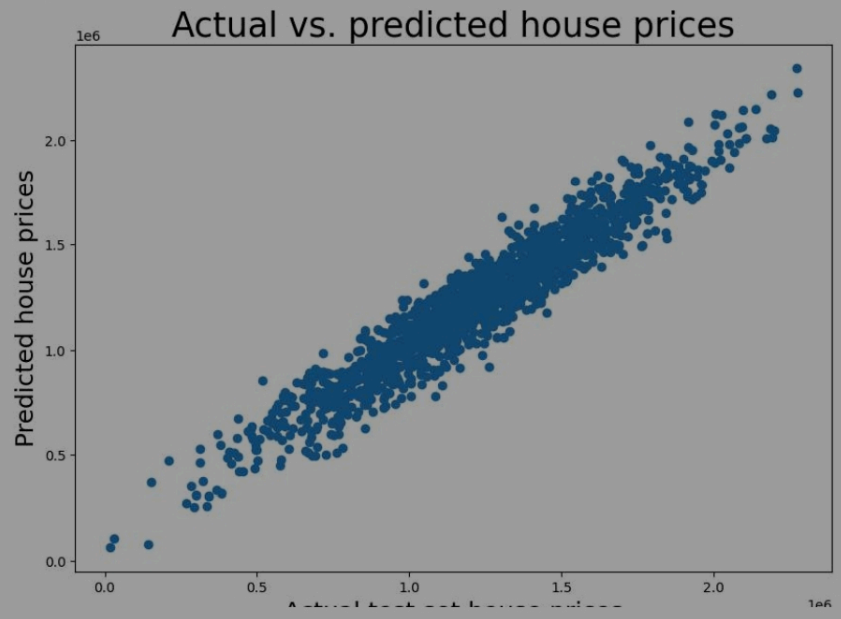
```
Type of the predicted object: <class 'numpy.ndarray'>
Size of the predicted object: (1500,)
```

```
plt.figure(figsize=(10,7))
```



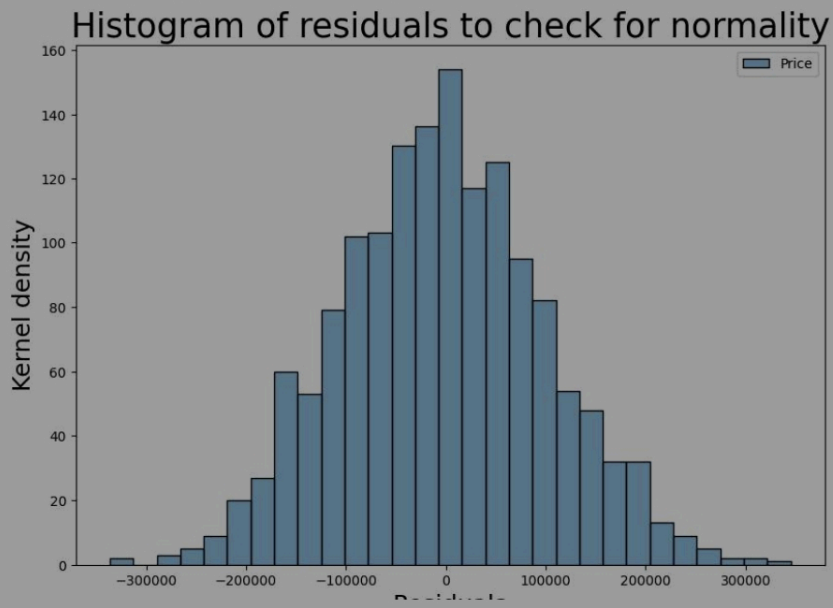
```
plt.title("Actual vs. predicted house prices",fontsize=25)
plt.xlabel("Actual test set house prices",fontsize=18)
plt.ylabel("Predicted house prices", fontsize=18)
plt.scatter(x=y_test,y=predictions)
```

```
<matplotlib.collections.PathCollection at 0x7fcbaba49f00>
```




```
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)
```

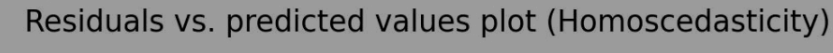
```
<Axes: title={'center': 'Histogram of residuals to check for normality'},
xlabel='Residuals', ylabel='Kernel density'>
```




```
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)
```



```
<matplotlib.collections.PathCollection at 0x7fcbab986020>
```



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

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

```
[ ] import numpy as np
min=np.min(predictions/6000)
max=np.max(predictions/12000)
print(min, max)
```

```
10.573398580303339 195.14363972586435
```

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

```
L = (100 - min)/(max - min)
L
plt.hist(L)
```

```
(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>)
```