siddhu4-1

April 15, 2025

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import fetch_california_housing
```

[2]:
```
housing = fetch_california_housing()
```

[3]:
```
%matplotlib inline
```

[4]:
```
housing=fetch_california_housing()
```

[5]:
```
print(housing.DESCR)
```

```
.. _california_housing_dataset:

California Housing dataset
--------------------------

**Data Set Characteristics:**

:Number of Instances: 20640

:Number of Attributes: 8 numeric, predictive attributes and the target

:Attribute Information:
    - MedInc        median income in block group
    - HouseAge      median house age in block group
    - AveRooms      average number of rooms per household
    - AveBedrms     average number of bedrooms per household
    - Population    block group population
    - AveOccup      average number of household members
    - Latitude      block group latitude
    - Longitude     block group longitude

:Missing Attribute Values: None

This dataset was obtained from the StatLib repository.
```

```
https://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html

The target variable is the median house value for California districts,
expressed in hundreds of thousands of dollars ($100,000).

This dataset was derived from the 1990 U.S. census, using one row per census
block group. A block group is the smallest geographical unit for which the U.S.
Census Bureau publishes sample data (a block group typically has a population
of 600 to 3,000 people).

A household is a group of people residing within a home. Since the average
number of rooms and bedrooms in this dataset are provided per household, these
columns may take surprisingly large values for block groups with few households
and many empty houses, such as vacation resorts.

It can be downloaded/loaded using the
:func:`sklearn.datasets.fetch_california_housing` function.

.. rubric:: References

- Pace, R. Kelley and Ronald Barry, Sparse Spatial Autoregressions,
  Statistics and Probability Letters, 33 (1997) 291-297
```

[6]:
```
housing.keys()
```

[6]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'feature_names', 'DESCR'])
```

[7]:
```
print(housing.feature_names)
```

```
['MedInc', 'HouseAge', 'AveRooms', 'AveBedrms', 'Population', 'AveOccup',
'Latitude', 'Longitude']
```

[9]:
```
SD= pd.DataFrame(housing.data)
```

[10]:
```
SD.head()
```

[10]:

	0	1	2	3	4	5	6	7
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25

[11]:
```
SD.columns=housing.feature_names
```

[12]:
```
SD.head()
```

[12]:
```
   MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  Latitude  \
0  8.3252      41.0  6.984127   1.023810       322.0  2.555556     37.88
1  8.3014      21.0  6.238137   0.971880      2401.0  2.109842     37.86
2  7.2574      52.0  8.288136   1.073446       496.0  2.802260     37.85
3  5.6431      52.0  5.817352   1.073059       558.0  2.547945     37.85
4  3.8462      52.0  6.281853   1.081081       565.0  2.181467     37.85

   Longitude
0    -122.23
1    -122.22
2    -122.24
3    -122.25
4    -122.25
```

[13]:
```
housing.target.shape
```

[13]:
```
(20640,)
```

[14]:
```
SD['PRICE'] = housing.target
```

[15]:
```
SD.isnull().sum()
```

[15]:
```
MedInc        0
HouseAge      0
AveRooms      0
AveBedrms     0
Population    0
AveOccup      0
Latitude      0
Longitude     0
PRICE         0
dtype: int64
```

[16]:
```
SD.head()
```

[16]:
```
   MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  Latitude  \
0  8.3252      41.0  6.984127   1.023810       322.0  2.555556     37.88
1  8.3014      21.0  6.238137   0.971880      2401.0  2.109842     37.86
2  7.2574      52.0  8.288136   1.073446       496.0  2.802260     37.85
3  5.6431      52.0  5.817352   1.073059       558.0  2.547945     37.85
4  3.8462      52.0  6.281853   1.081081       565.0  2.181467     37.85

   Longitude  PRICE
0    -122.23  4.526
1    -122.22  3.585
2    -122.24  3.521
3    -122.25  3.413
```

```
4    -122.25  3.422
```

[17]:
```
SD.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   MedInc      20640 non-null  float64
 1   HouseAge    20640 non-null  float64
 2   AveRooms    20640 non-null  float64
 3   AveBedrms   20640 non-null  float64
 4   Population  20640 non-null  float64
 5   AveOccup    20640 non-null  float64
 6   Latitude    20640 non-null  float64
 7   Longitude   20640 non-null  float64
 8   PRICE       20640 non-null  float64
dtypes: float64(9)
memory usage: 1.4 MB
```

[18]:
```
SD.describe()
```

[18]:

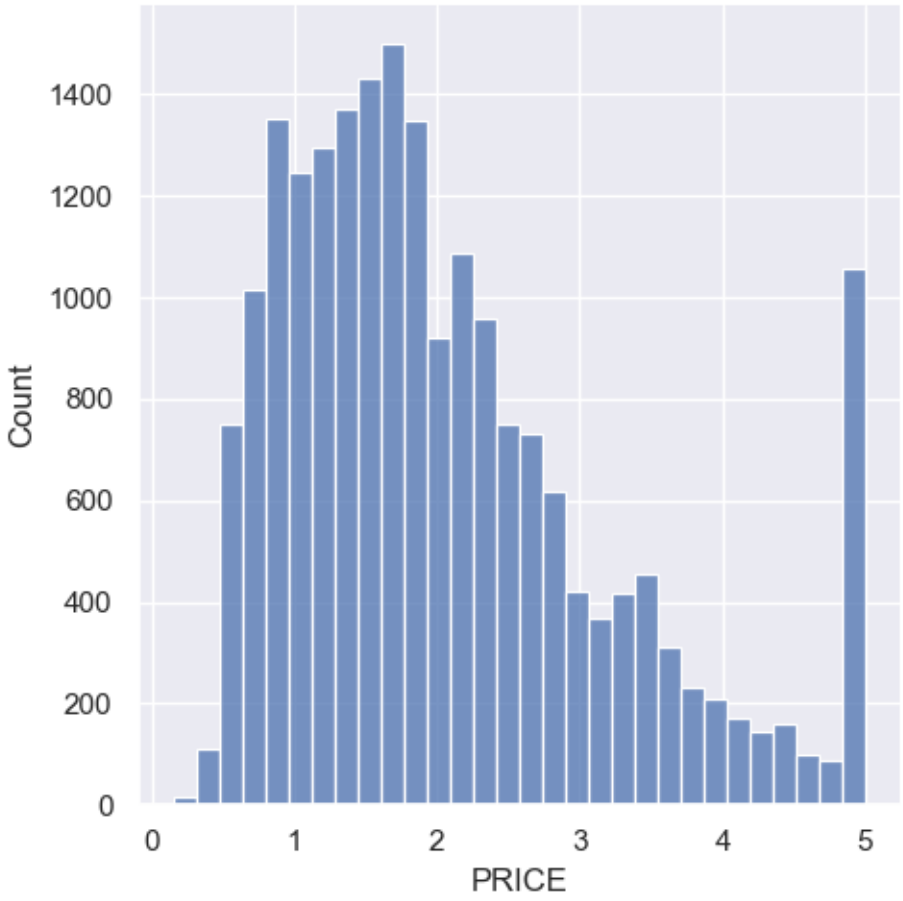
	MedInc	HouseAge	AveRooms	AveBedrms	Population	\
count	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	
mean	3.870671	28.639486	5.429000	1.096675	1425.476744	
std	1.899822	12.585558	2.474173	0.473911	1132.462122	
min	0.499900	1.000000	0.846154	0.333333	3.000000	
25%	2.563400	18.000000	4.440716	1.006079	787.000000	
50%	3.534800	29.000000	5.229129	1.048780	1166.000000	
75%	4.743250	37.000000	6.052381	1.099526	1725.000000	
max	15.000100	52.000000	141.909091	34.066667	35682.000000	

	AveOccup	Latitude	Longitude	PRICE
count	20640.000000	20640.000000	20640.000000	20640.000000
mean	3.070655	35.631861	-119.569704	2.068558
std	10.386050	2.135952	2.003532	1.153956
min	0.692308	32.540000	-124.350000	0.149990
25%	2.429741	33.930000	-121.800000	1.196000
50%	2.818116	34.260000	-118.490000	1.797000
75%	3.282261	37.710000	-118.010000	2.647250
max	1243.333333	41.950000	-114.310000	5.000010

[19]:
```
sns.set(rc={'figure.figsize':(7.7,4.27)})
```

[28]:
```
sns.displot(SD['PRICE'], bins=30)
 plt.show()
```

```
C:\Users\WINDOWS 10\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



[29]:
```
SD.corr()
```

[29]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	\
MedInc	1.000000	-0.119034	0.326895	-0.062040	0.004834	0.018766	
HouseAge	-0.119034	1.000000	-0.153277	-0.077747	-0.296244	0.013191	
AveRooms	0.326895	-0.153277	1.000000	0.847621	-0.072213	-0.004852	
AveBedrms	-0.062040	-0.077747	0.847621	1.000000	-0.066197	-0.006181	
Population	0.004834	-0.296244	-0.072213	-0.066197	1.000000	0.069863	
AveOccup	0.018766	0.013191	-0.004852	-0.006181	0.069863	1.000000	
Latitude	-0.079809	0.011173	0.106389	0.069721	-0.108785	0.002366	
Longitude	-0.015176	-0.108197	-0.027540	0.013344	0.099773	0.002476	
PRICE	0.688075	0.105623	0.151948	-0.046701	-0.024650	-0.023737	

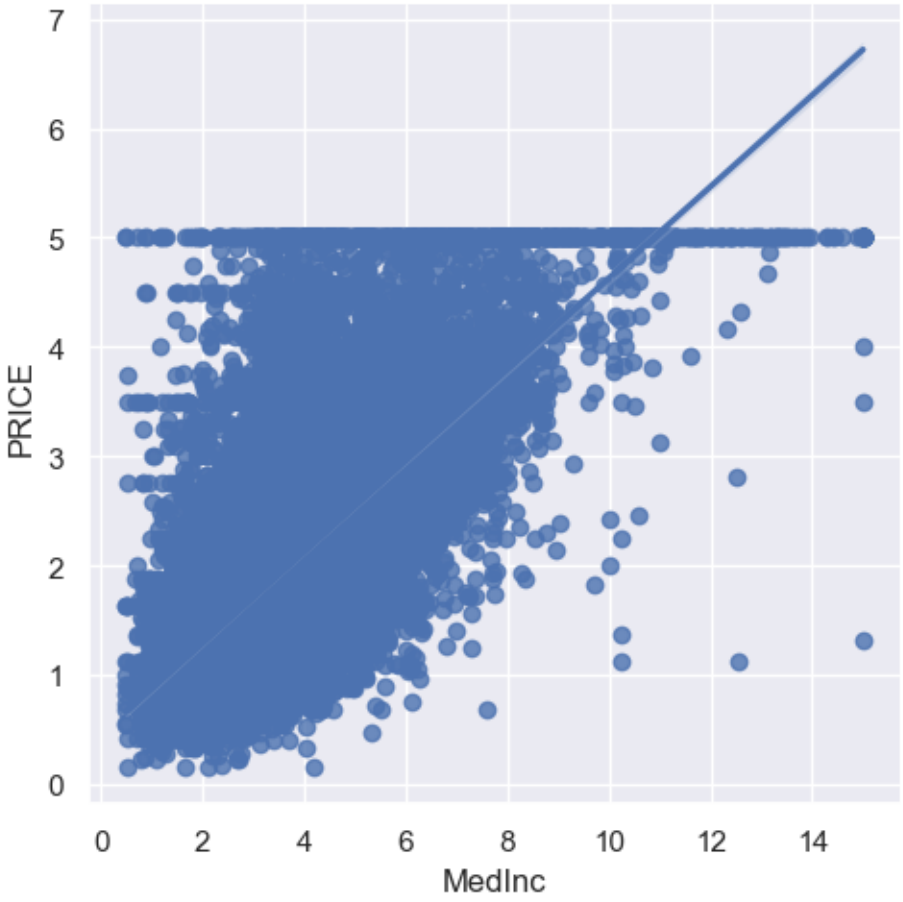
	Latitude	Longitude	PRICE

```
MedInc      -0.079809  -0.015176  0.688075
HouseAge     0.011173  -0.108197  0.105623
AveRooms     0.106389  -0.027540  0.151948
AveBedrms    0.069721   0.013344 -0.046701
Population  -0.108785   0.099773 -0.024650
AveOccup     0.002366   0.002476 -0.023737
Latitude     1.000000  -0.924664 -0.144160
Longitude   -0.924664   1.000000 -0.045967
PRICE       -0.144160  -0.045967  1.000000
```

[31]:
```
plt.figure(figsize=(1, 1))
 correlation_matrix=SD.corr()
 plt.show()
```

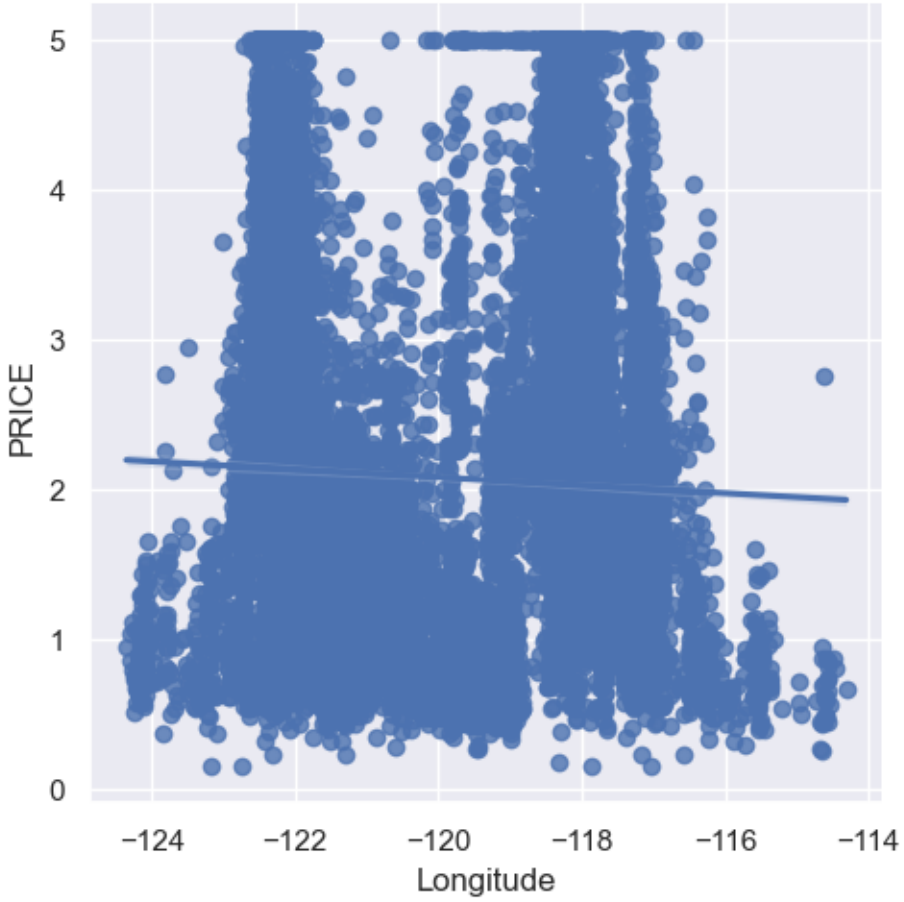
```
<Figure size 100x100 with 0 Axes>
```

[33]:
```
sns.lmplot(x='MedInc',y='PRICE' ,data=SD);
 plt.show()
```

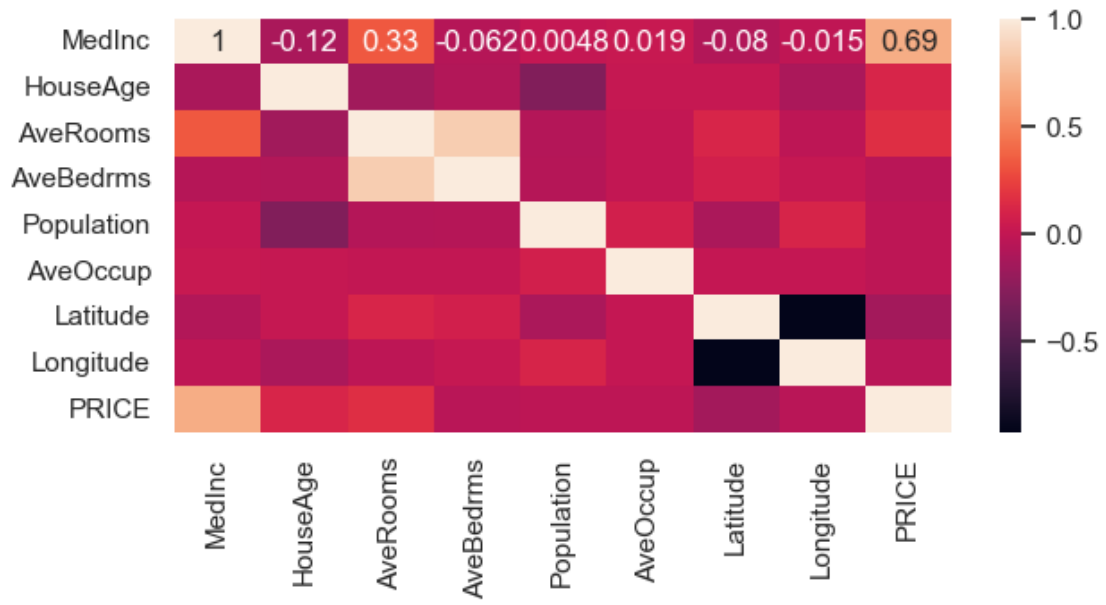


[82]:
```
plt.figure(figsize=(3, 3))
sns.lmplot(x='Longitude',y='PRICE' ,data=bs);
plt.show()
```

```
<Figure size 300x300 with 0 Axes>
```



[34]:
```
plt.figure(figsize=(7, 3))
 sns.heatmap(data=correlation_matrix,annot=True)
 plt.show()
```



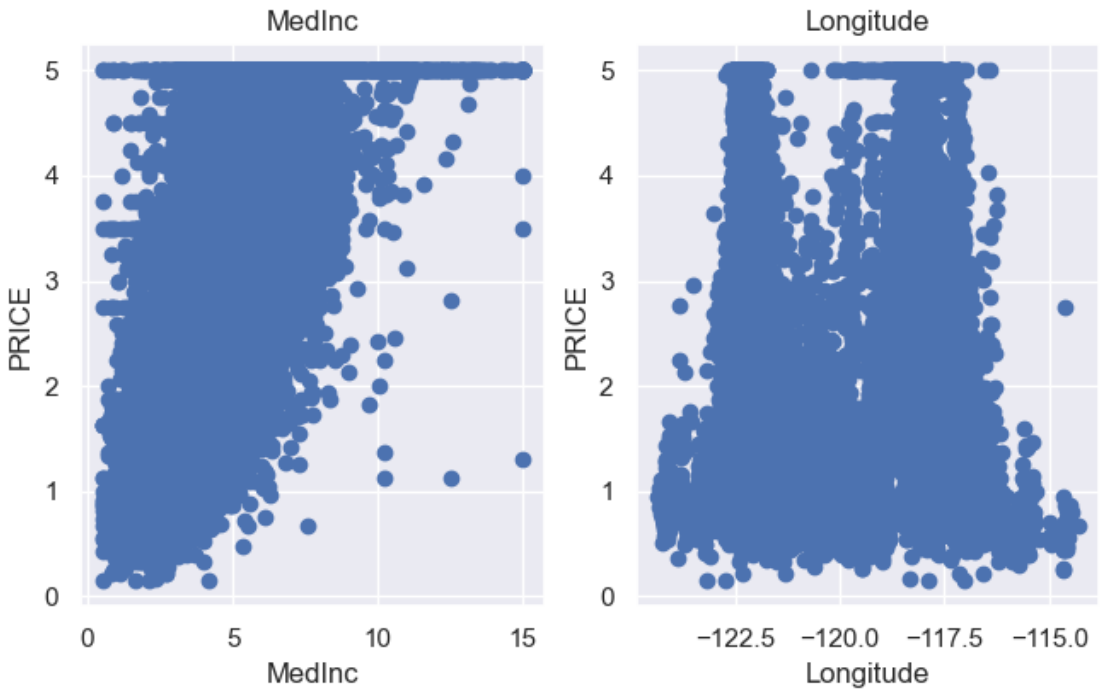
[35]:
```
plt.figure(figsize=(7, 3))
```

[35]:
```
<Figure size 700x300 with 0 Axes>
```

```
<Figure size 700x300 with 0 Axes>
```

[37]:
```
features = ['MedInc', 'Longitude']
target = SD['PRICE']
```

[38]:
```
for i, col in enumerate(features):
 plt.subplot(1, len(features) , i+1)
 x = SD[col]
 y = target
 plt.scatter(x, y, marker='o')
 plt.title(col)
 plt.xlabel(col)
 plt.ylabel('PRICE')
```



[40]:
```
plt.show()
```

[41]:
```
X = SD.drop('Longitude' , axis = 1)
 Y = SD['PRICE']
```

[42]:
```
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 =42)
```

[43]:
```
from sklearn.linear_model import LinearRegression
```

[44]:
```
lm = LinearRegression()
```

[45]:
```
lm.fit(X_train, Y_train)
```

[45]:
```
LinearRegression()
```

[47]:
```
# print the inercept
 print(lm.intercept_)
```

```
2.886579864025407e-14
```

[56]:
```
coeff_df = pd.DataFrame(lm.coef_,X.columns,columns=['Coefficient'])
 coeff_df
```

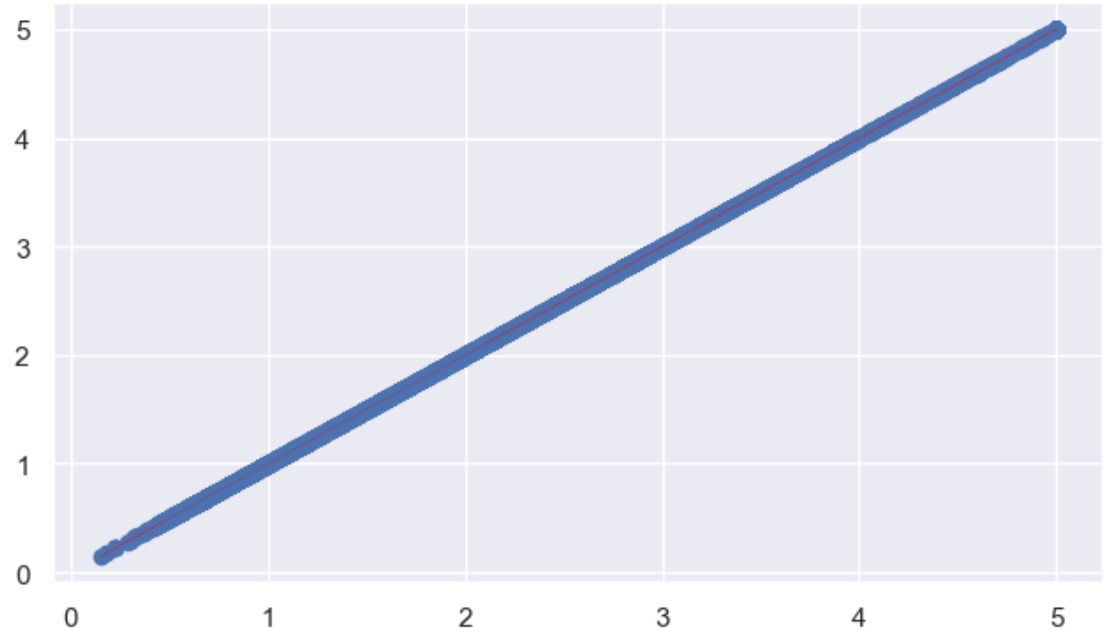
[56]:

	Coefficient
MedInc	4.599519e-15
HouseAge	3.913536e-15
AveRooms	-1.179612e-15
AveBedrms	3.044440e-16
Population	-1.110223e-16
AveOccup	-4.683753e-17
Latitude	2.359224e-16
PRICE	1.000000e+00

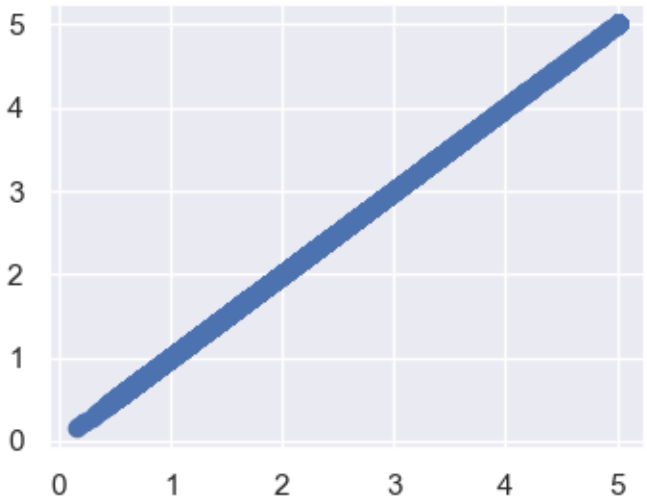
[48]:
```
Y_pred=lm.predict(X_test)
```

[58]:
```
plt.scatter(Y_test,Y_pred)
 m,b=np.polyfit(Y_test,Y_pred,1)
 plt.plot(Y_test,m*Y_test+b,color="red",linewidth=0.2,linestyle="--")
 print(f"Equation of line: Y={m}*X+{b}")
 print(f"Slope :{m}")
 print(f"Y=intercept:{b}")
```

```
Equation of line: Y=1.00000000000001*X+-2.112956651590836e-14
Slope :1.00000000000001
Y=intercept:-2.112956651590836e-14
```



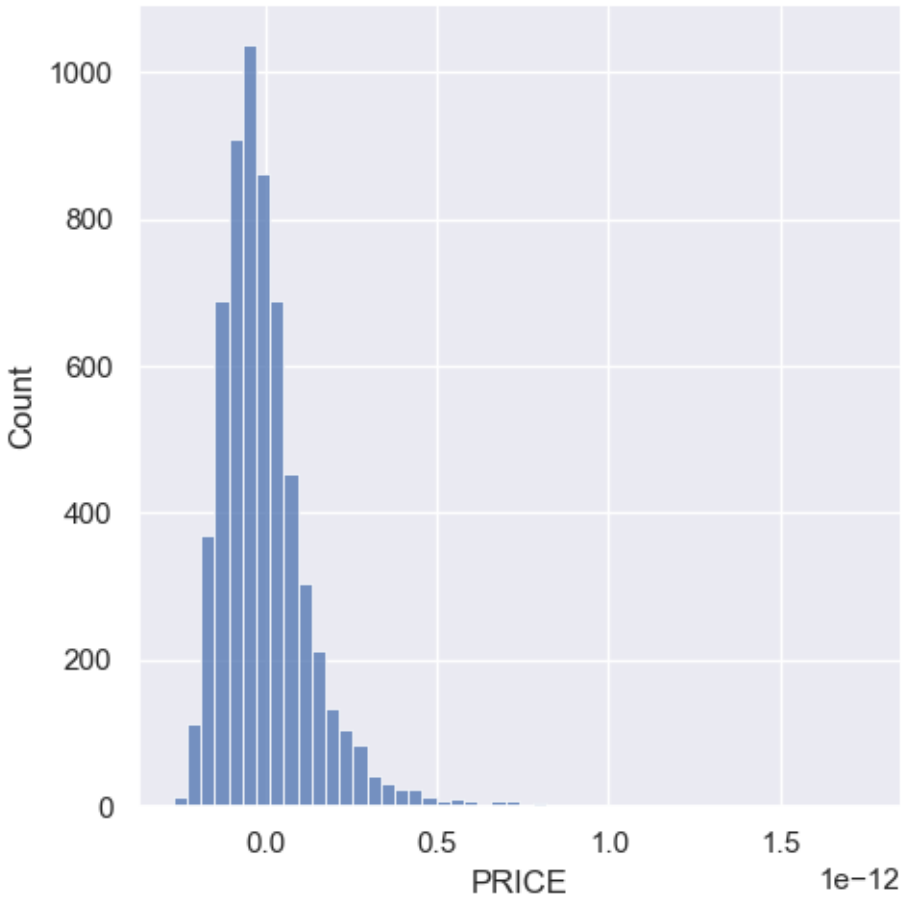
[49]:
```
plt.figure(figsize=(4, 3))
 plt.scatter(Y_test,Y_pred)
 plt.show()
```



[61]:
```
plt.figure(figsize=(7, 3))
 sns.displot((Y_test-Y_pred),bins=50);
 plt.show()
```

```
C:\Users\WINDOWS 10\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

```
<Figure size 700x300 with 0 Axes>
```



[62]:
```
from sklearn import metrics
```

[ ]:
```

```