In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
from sklearn.datasets import fetch_california_housing
housing = fetch_california_housing()
```

In [3]:
```
housing.keys()
```

Out[3]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'feature_names', 'DESCR'])
```

In [4]:
```
dataset=pd.DataFrame(housing.data,columns=housing.feature_names)
```

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

Out[5]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
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

In [6]:
```
dataset['price']=housing.target
```

In [7]:
```
print(dataset['price'])
```

```
0        4.526
1        3.585
2        3.521
3        3.413
4        3.422
         ...
20635    0.781
20636    0.771
20637    0.923
20638    0.847
20639    0.894
Name: price, Length: 20640, dtype: float64
```

In [8]:
```
dataset.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
```

In [9]:
```
dataset.isnull().sum()
```

Out[9]:
```
MedInc        0
HouseAge      0
AveRooms      0
AveBedrms     0
Population    0
AveOccup      0
Latitude      0
Longitude     0
price         0
dtype: int64
```

In [10]:
```
dataset.describe()
```

Out[10]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	L
count	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640
mean	3.870671	28.639486	5.429000	1.096675	1425.476744	3.070655	35
std	1.899822	12.585558	2.474173	0.473911	1132.462122	10.386050	2
min	0.499900	1.000000	0.846154	0.333333	3.000000	0.692308	32
25%	2.563400	18.000000	4.440716	1.006079	787.000000	2.429741	33
50%	3.534800	29.000000	5.229129	1.048780	1166.000000	2.818116	34
75%	4.743250	37.000000	6.052381	1.099526	1725.000000	3.282261	37
max	15.000100	52.000000	141.909091	34.066667	35682.000000	1243.333333	41

In [11]:
```
dataset.corr()
```

Out[11]:

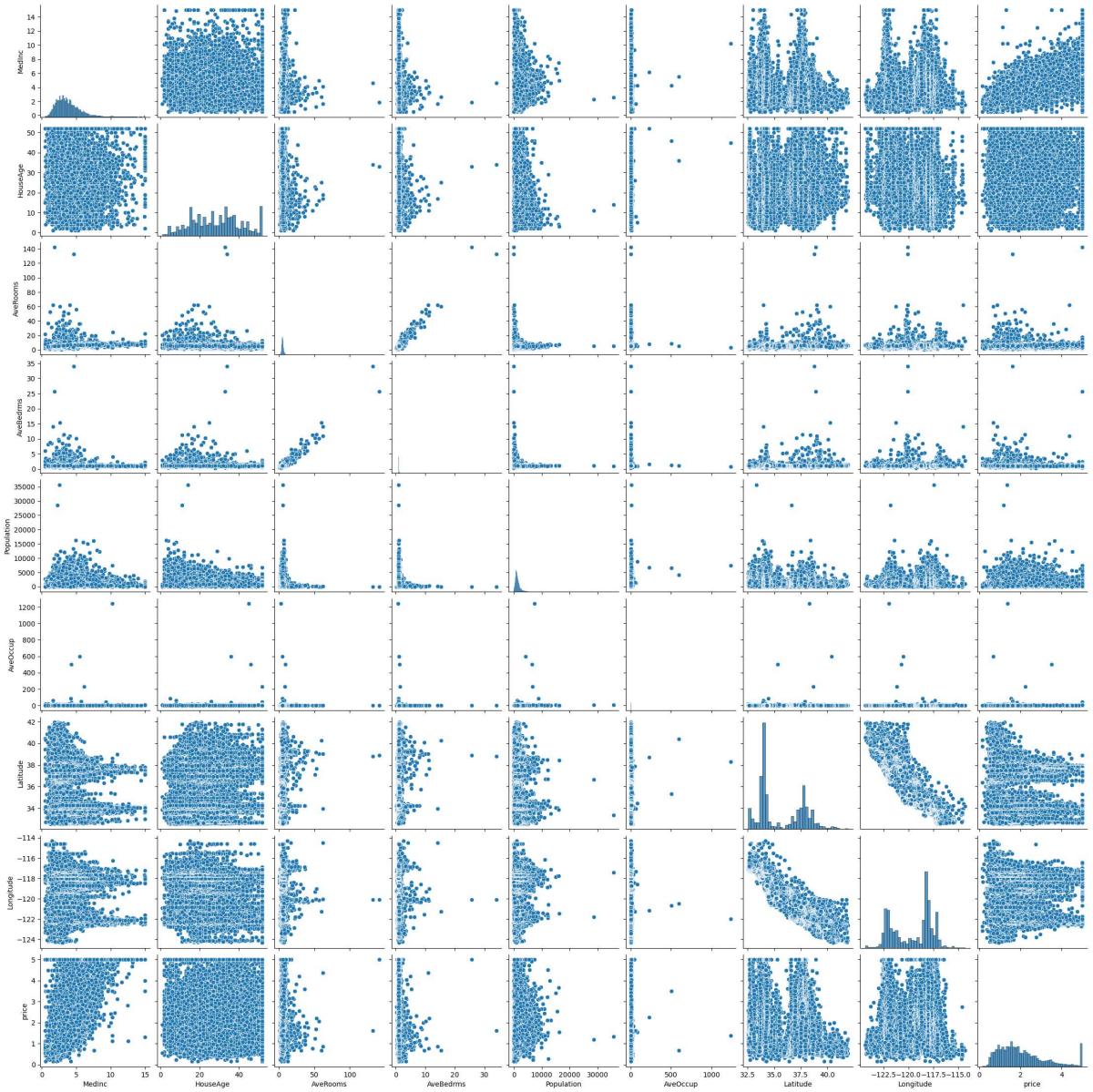
	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Long
MedInc	1.000000	-0.119034	0.326895	-0.062040	0.004834	0.018766	-0.079809	-0.0
HouseAge	-0.119034	1.000000	-0.153277	-0.077747	-0.296244	0.013191	0.011173	-0.1
AveRooms	0.326895	-0.153277	1.000000	0.847621	-0.072213	-0.004852	0.106389	-0.0
AveBedrms	-0.062040	-0.077747	0.847621	1.000000	-0.066197	-0.006181	0.069721	0.0
Population	0.004834	-0.296244	-0.072213	-0.066197	1.000000	0.069863	-0.108785	0.0
AveOccup	0.018766	0.013191	-0.004852	-0.006181	0.069863	1.000000	0.002366	0.0
Latitude	-0.079809	0.011173	0.106389	0.069721	-0.108785	0.002366	1.000000	-0.9
Longitude	-0.015176	-0.108197	-0.027540	0.013344	0.099773	0.002476	-0.924664	1.0
price	0.688075	0.105623	0.151948	-0.046701	-0.024650	-0.023737	-0.144160	-0.0

In [12]:
```
dataset.columns
```

Out[12]:
```
Index(['MedInc', 'HouseAge', 'AveRooms', 'AveBedrms', 'Population', 'AveOccup',
       'Latitude', 'Longitude', 'price'],
      dtype='object')
```

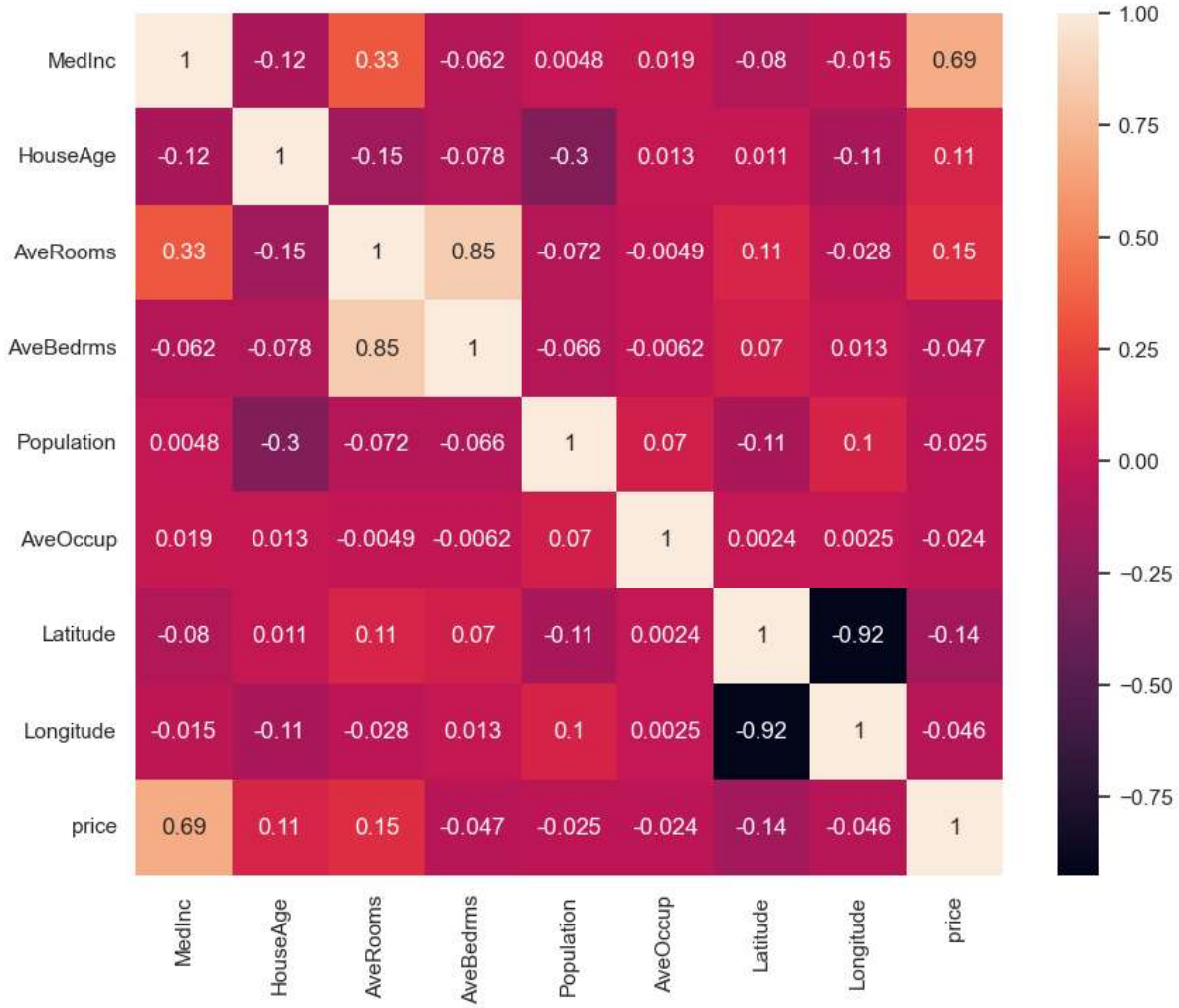
In [13]:
```
sns.pairplot(dataset)
```

Out[13]:
```
<seaborn.axisgrid.PairGrid at 0x244d14de3b0>
```



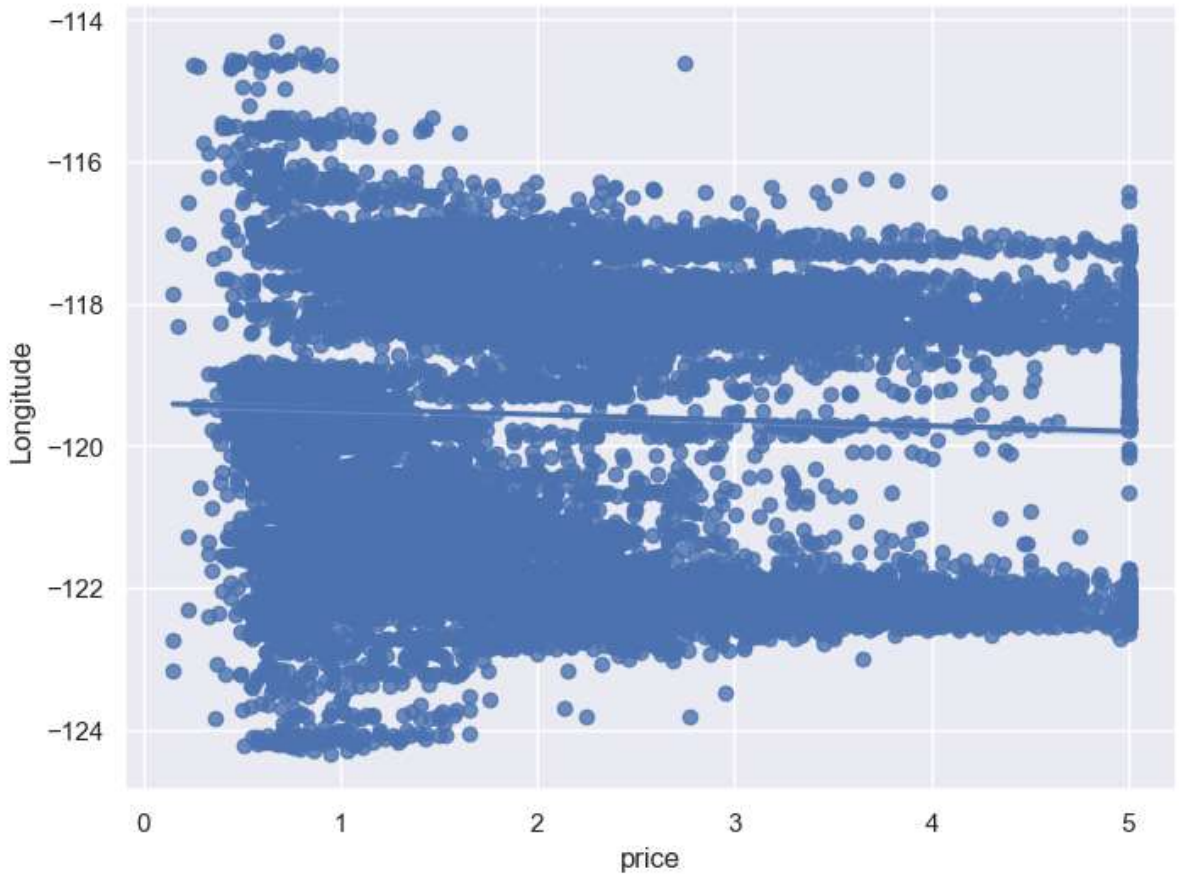
In [14]:
```
sns.set(rc={'figure.figsize':(10,8)})
sns.heatmap(dataset.corr(),annot=True )
```

Out[14]: <AxesSubplot: >



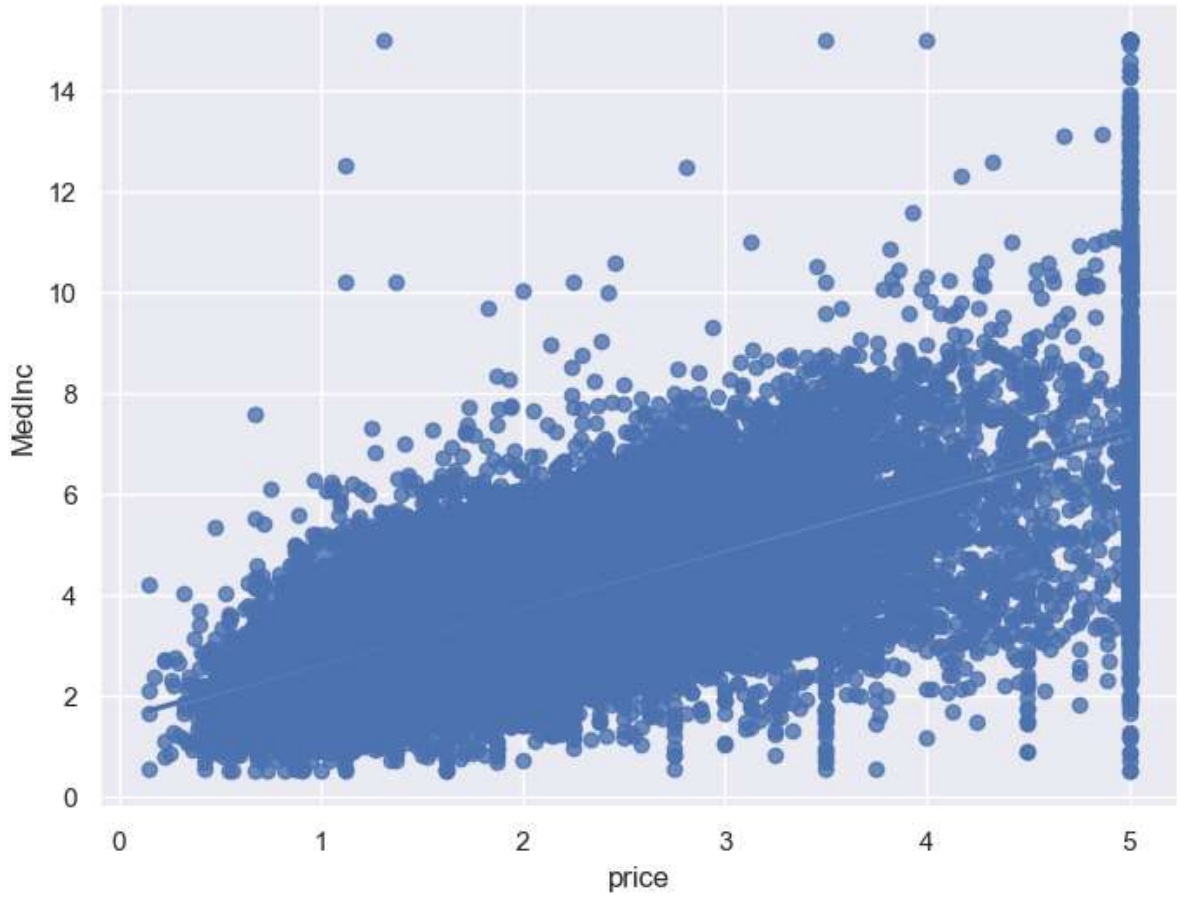
In [15]:
```
sns.set(rc={'figure.figsize':(8,6)})
sns.regplot(x='price',y='Longitude',data=dataset )
```

Out[15]:
```
<AxesSubplot: xlabel='price', ylabel='Longitude'>
```



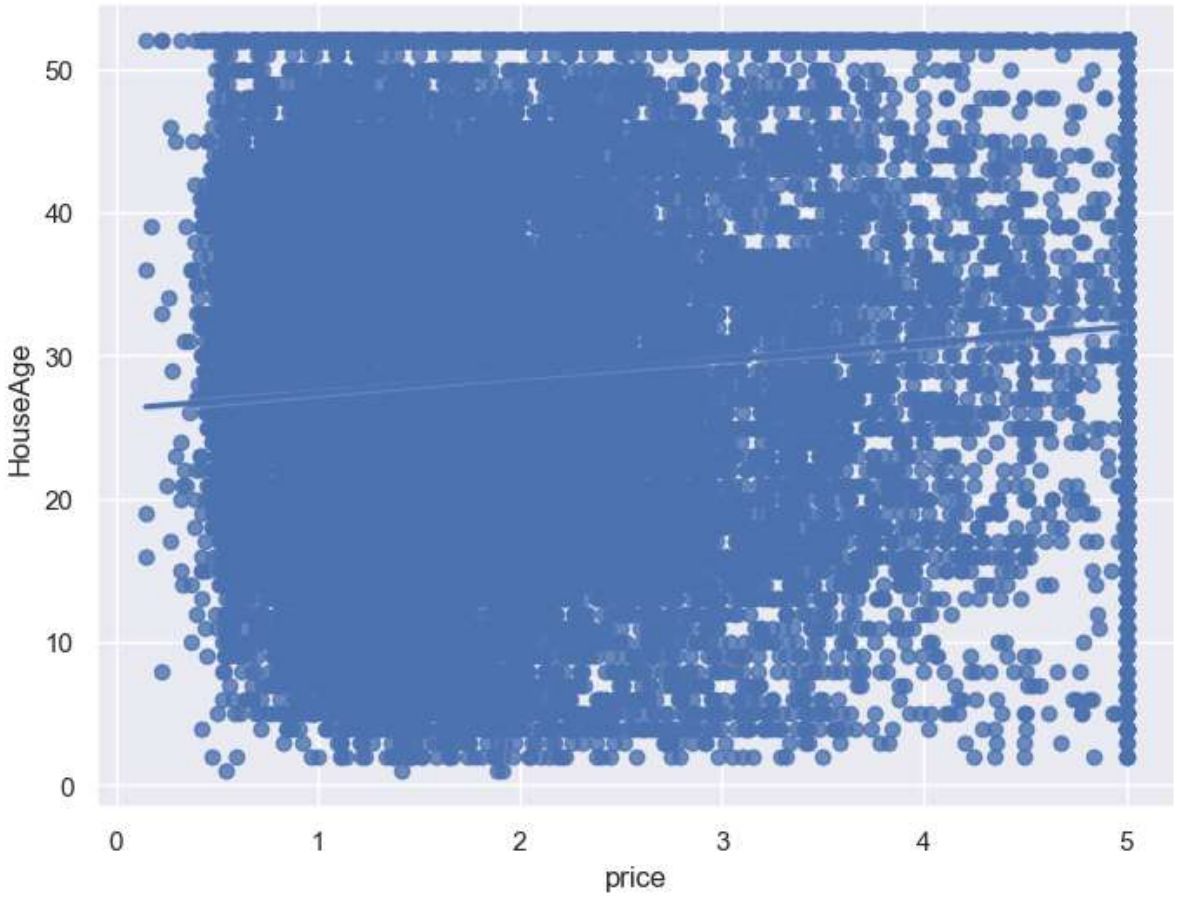
In [16]:
```
sns.set(rc={'figure.figsize':(8,6)})
sns.regplot(x='price',y= 'MedInc',data=dataset )
```

Out[16]:
```
<AxesSubplot: xlabel='price', ylabel='MedInc'>
```



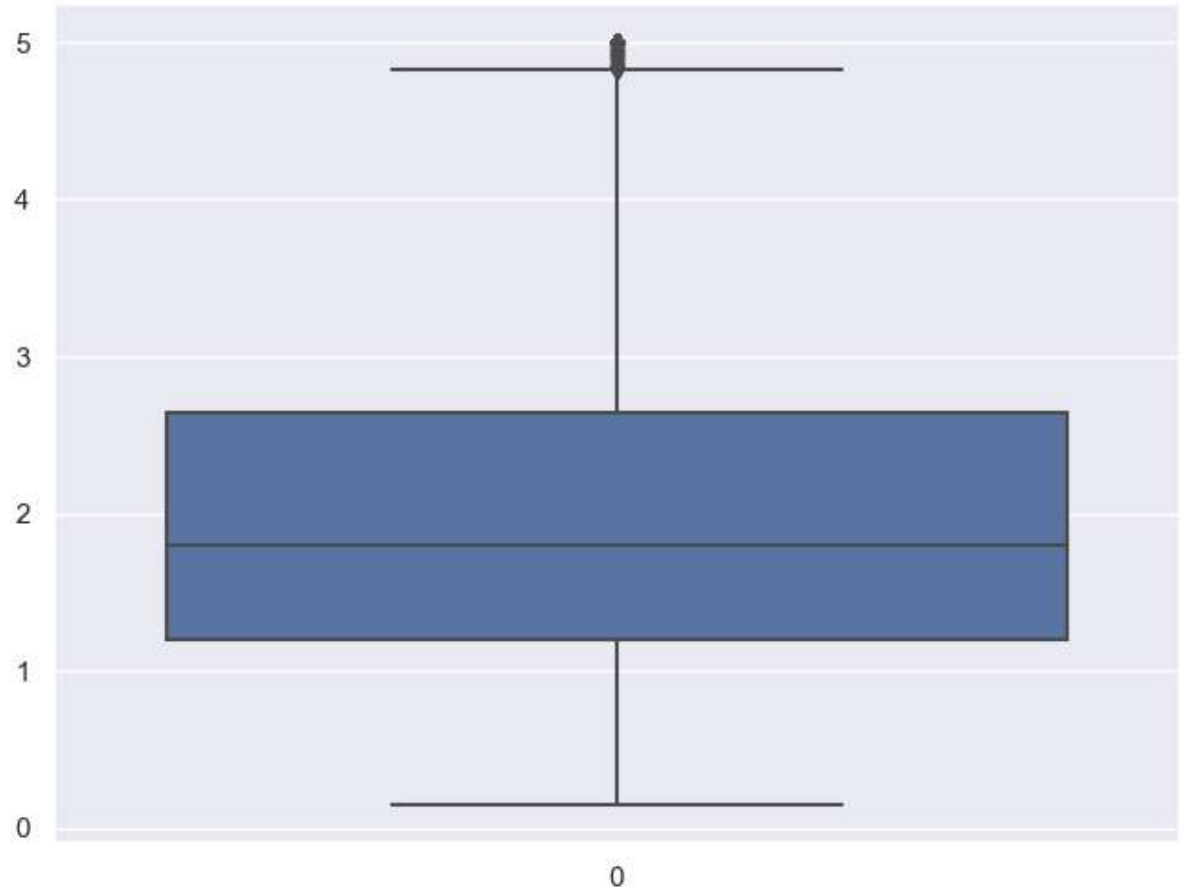
In [17]:
```
sns.set(rc={'figure.figsize':(8,6)})
sns.regplot(x='price',y= 'HouseAge',data=dataset )
```

Out[17]:
```
<AxesSubplot: xlabel='price', ylabel='HouseAge'>
```



In [18]:
```
sns.boxplot(dataset['price'])
```

Out[18]: <AxesSubplot: >



In [19]:
```
#independent and dependent feature

x=dataset.iloc[:,:-1]
y=dataset.iloc[:,-1]
```

In [20]:
```
x
```

Out[20]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
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
...	...	...	...	...	...	...	...	...
20635	1.5603	25.0	5.045455	1.133333	845.0	2.560606	39.48	-121.09
20636	2.5568	18.0	6.114035	1.315789	356.0	3.122807	39.49	-121.21
20637	1.7000	17.0	5.205543	1.120092	1007.0	2.325635	39.43	-121.22
20638	1.8672	18.0	5.329513	1.171920	741.0	2.123209	39.43	-121.32
20639	2.3886	16.0	5.254717	1.162264	1387.0	2.616981	39.37	-121.24

20640 rows × 8 columns

In [21]:
```
y
```

Out[21]:
```
0        4.526
1        3.585
2        3.521
3        3.413
4        3.422
         ...
20635    0.781
20636    0.771
20637    0.923
20638    0.847
20639    0.894
Name: price, Length: 20640, dtype: float64
```

In [22]:
```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=12)
```

In [23]:
```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

Out[23]:
```
((13828, 8), (6812, 8), (13828,), (6812,))
```

In [24]:
```
6812/(13828+6812)*100 #33% data is in testing and rest is in training
```

Out[24]:
```
33.00387596899225
```

In [25]:
```
from sklearn.preprocessing import StandardScaler
scalar=StandardScaler()
```

In [26]:
```
scalar
```

Out[26]:
```
▾ StandardScaler
StandardScaler()
```

In [27]:
```
x_train=scalar.fit_transform(x_train)
```

In [28]:
```
x_test=scalar.fit_transform(x_test)
```

In [29]:
```
x_train
```

Out[29]:
```
array([[-1.24904603,  0.42633018, -0.61320277, ..., -0.04338037,
        -0.75605028,  0.611307  ],
       [-0.14406409,  0.50577515,  0.18856354, ..., -0.05536191,
        -0.11982222,  0.31137802],
       [ 0.9692072 ,  0.3468852 ,  0.63641081, ...,  0.00476754,
        -1.39227835,  1.25615429],
       ...,
       [-0.93899219, -1.24201426, -0.09375134, ..., -0.07052394,
         1.65787385, -1.02830472],
       [-0.67617706, -0.76534442, -0.67423435, ..., -0.05918003,
        -0.69523436,  0.66629397],
       [ 0.25929815,  0.74411007,  0.26440744, ..., -0.05489583,
        -1.34081873,  1.25115547]])
```

In [30]:
```
x_test
```

Out[30]:
```
array([[ 0.24916919,  1.06101413, -0.08969363, ...,  0.26576233,
        -0.74243808,  0.71319515],
       [-0.54873699, -0.13125719, -0.62158373, ..., -0.56352247,
         0.95047089, -1.25738263],
       [ 0.36609347, -1.00558949,  0.04177291, ...,  0.01951513,
        -0.93470751,  0.92219582],
       ...,
       [-0.13653111,  0.50462084,  0.06073846, ...,  0.02910616,
         0.9176444 , -0.59554717],
       [-0.69692031, -1.64146753,  0.31454275, ..., -0.88159557,
        -0.89250203,  1.17598235],
       [ 0.20612989,  1.85586167,  0.00917042, ..., -0.75619168,
         1.05832936, -1.34695435]])
```

In [49]:
```
from sklearn.linear_model import LinearRegression
```

In [50]:
```
regression=LinearRegression()
```

In [51]:
```
regression.fit(x_train,y_train)
```

Out[51]:
```
▾ LinearRegression
LinearRegression()
```

In [52]:
```
regression.coef_
```

Out[52]:
```
array([ 0.8160491 ,  0.11835219, -0.25538202,  0.3034737 , -0.00345005,
       -0.04064189, -0.9039433 , -0.87297273])
```

In [53]:
```
regression.intercept_
```

Out[53]:
```
2.06527526974255
```

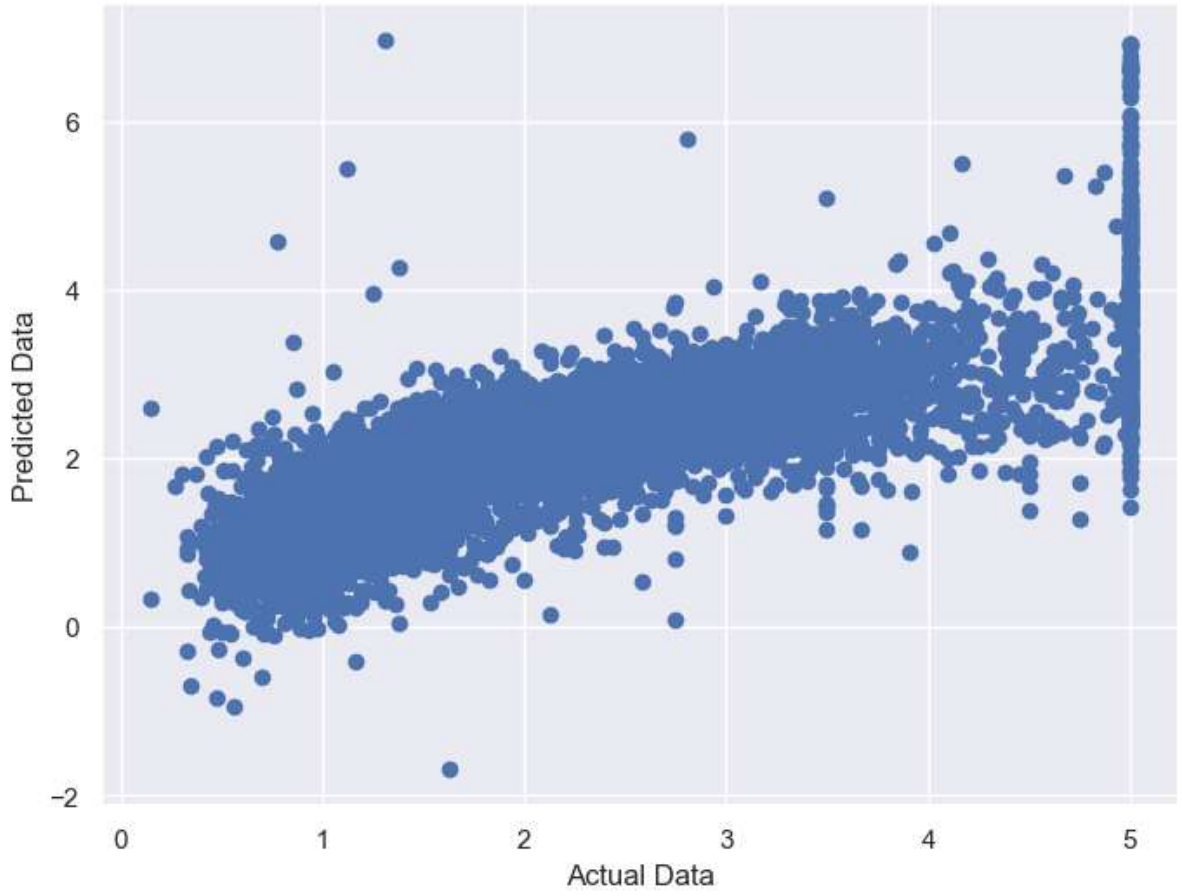
In [55]:
```
reg_pred=regression.predict(x_test)
```

In [57]:
```
print(reg_pred)
```

```
[2.43556627 2.06941919 2.28208702 ... 1.64898575 1.11214087 2.66106521]
```

In [65]:
```
plt.scatter(y_test,reg_pred)
plt.xlabel('Actual Data')
plt.ylabel('Predicted Data')
```

Out[65]:
```
Text(0, 0.5, 'Predicted Data')
```



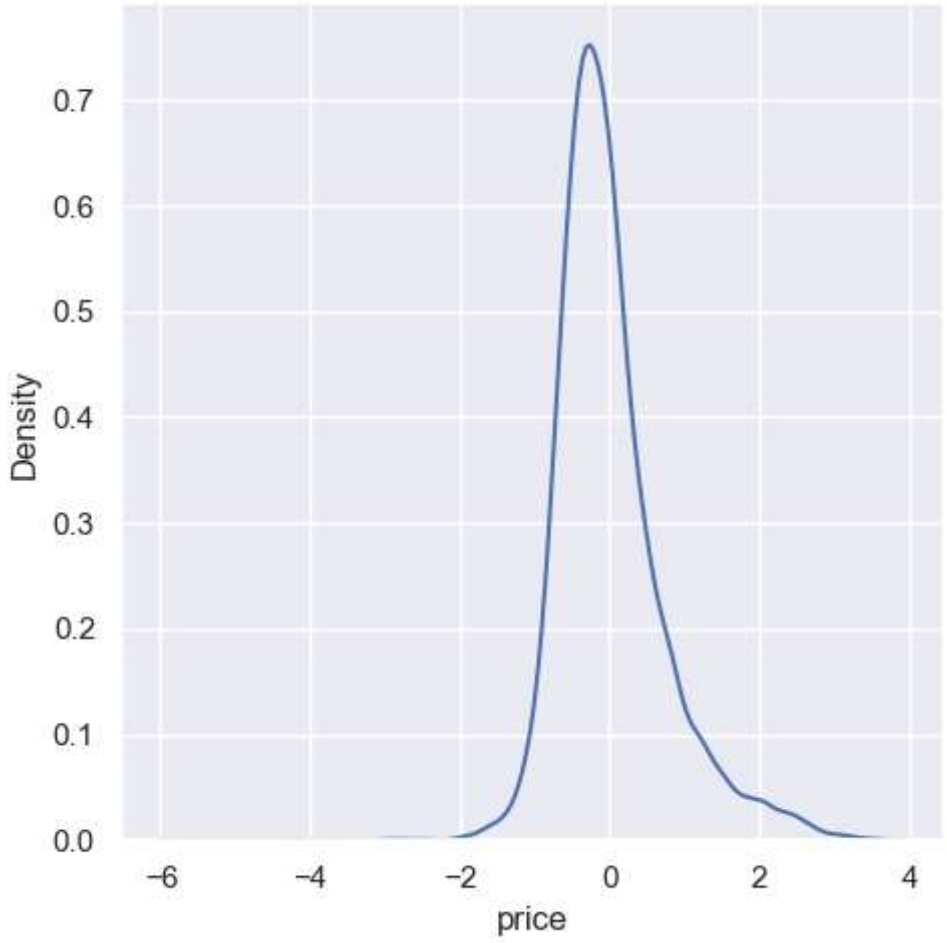
In [67]:
```
residual=y_test-reg_pred
```

In [68]:
```
residual
```

Out[68]:
```
6906    -0.321566
767     -0.117419
10555    0.135913
17456   -1.391623
20617   -0.379597
           ...
861     -0.096177
10326   -0.022179
19588    0.137014
12146   -0.011141
425      0.187935
Name: price, Length: 6812, dtype: float64
```

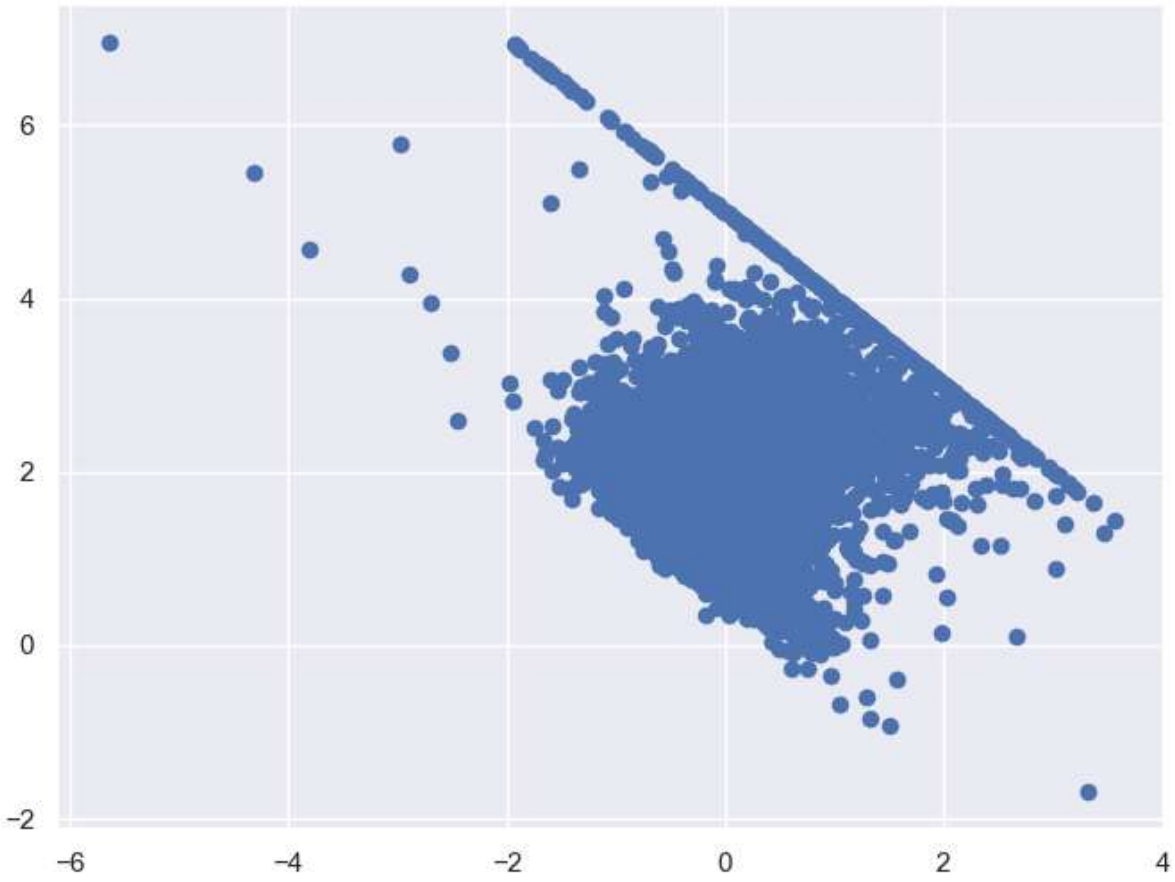
In [69]:
```
sns.displot(residual,kind='kde')
```

Out[69]:
```
<seaborn.axisgrid.FacetGrid at 0x244f44bf010>
```



In [71]:
```
plt.scatter(residual,reg_pred)
```

Out[71]:
```
<matplotlib.collections.PathCollection at 0x244f4779f00>
```



Performance Matrics

In [77]:
```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
print(mean_squared_error(y_test,reg_pred))
print(mean_absolute_error(y_test,reg_pred))
print(np.sqrt(mean_squared_error(y_test,reg_pred)))
```

```
0.5204783529558654
0.5257562312164743
0.7214418569475058
```

R Square Value and Adjusted R Square

In [78]:
```
from sklearn.metrics import r2_score
score=r2_score(y_test,reg_pred)
```

In [79]:
```
score
```

Out[79]:
```
0.6125609410014536
```

In [80]:
```
# Adjusted R Square

1-(1-score)*(len(y_test)-1)/(len(y_test)-x_test.shape[1]-1)
```

Out[80]: 0.6121053313480671

In [127…
```
df1=pd.DataFrame({'Actual':y_test,'Predict':reg_pred})
df1
```

Out[127]:

	Actual	Predict
6906	2.114	2.435566
767	1.952	2.069419
10555	2.418	2.282087
17456	1.283	2.674623
20617	0.708	1.087597
...	...	...
861	2.169	2.265177
10326	2.832	2.854179
19588	1.786	1.648986
12146	1.101	1.112141
425	2.849	2.661065

6812 rows × 2 columns

Lasso Regression Model

In [82]:
```
from sklearn.linear_model import Lasso
lasso = Lasso()
lasso
```

Out[82]:
```
Lasso()
```

In [84]:
```
lasso.fit(x_train, y_train)
```

Out[84]:
```
Lasso()
```

In [86]:
```
print(lasso.coef_)
```

```
[ 0.  0.  0. -0. -0. -0. -0. -0.]
```

In [92]:
```
print(lasso.intercept_)
```

```
2.065275269742551
```

In [93]:
```
lasso_predict=lasso.predict(x_test)
```

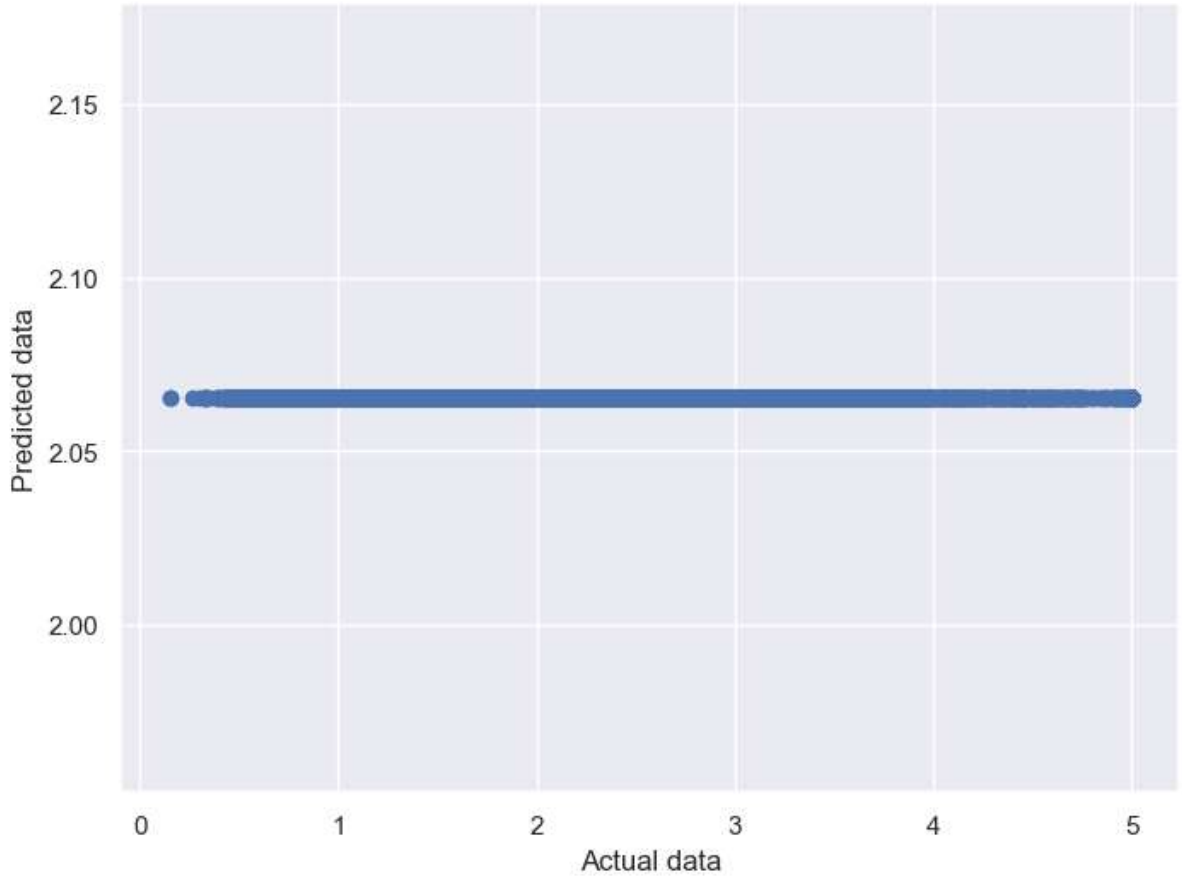
In [94]:
```
lasso_predict
```

Out[94]:
```
array([2.06527527, 2.06527527, 2.06527527, ..., 2.06527527, 2.06527527,
       2.06527527])
```

Assumption of lasso

In [97]:
```
plt.scatter(y_test,lasso_predict)
plt.xlabel('Actual data')
plt.ylabel('Predicted data')
```

Out[97]:
```
Text(0, 0.5, 'Predicted data')
```



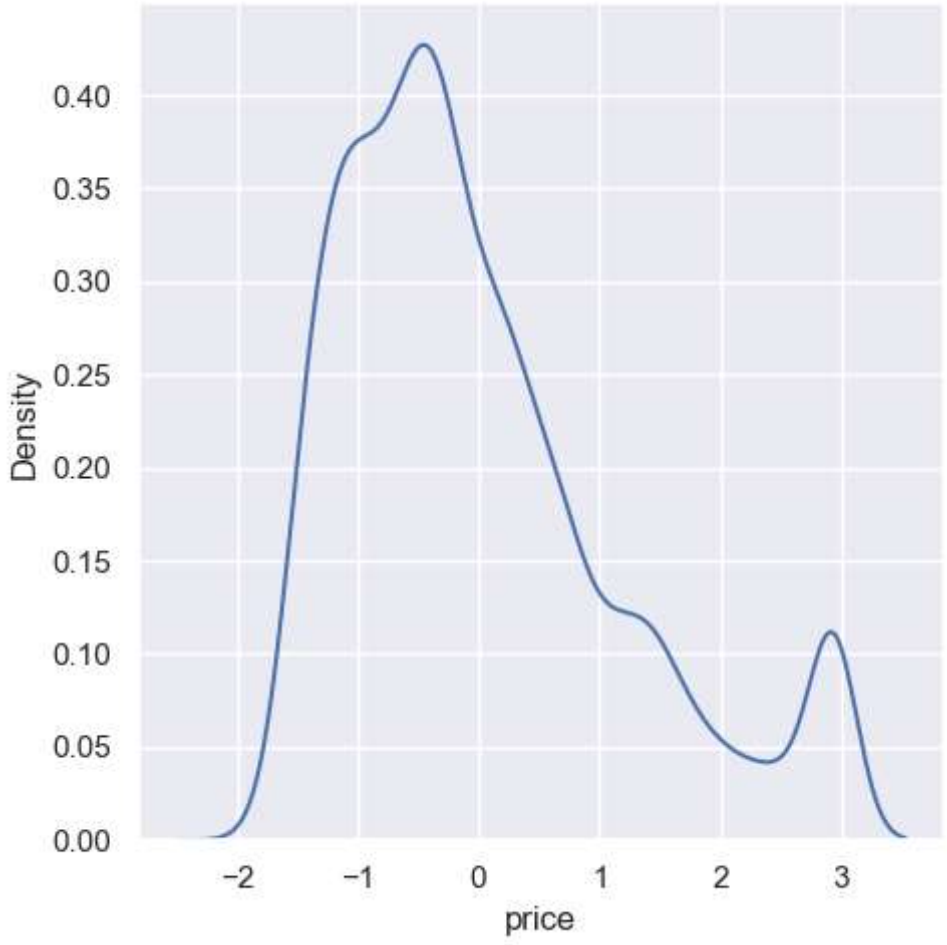
In [98]:
```
residual = y_test-lasso_predict
```

In [99]:
```
residual
```

Out[99]:
```
6906     0.048725
767     -0.113275
10555    0.352725
17456   -0.782275
20617   -1.357275
           ...
861      0.103725
10326    0.766725
19588   -0.279275
12146   -0.964275
425      0.783725
Name: price, Length: 6812, dtype: float64
```

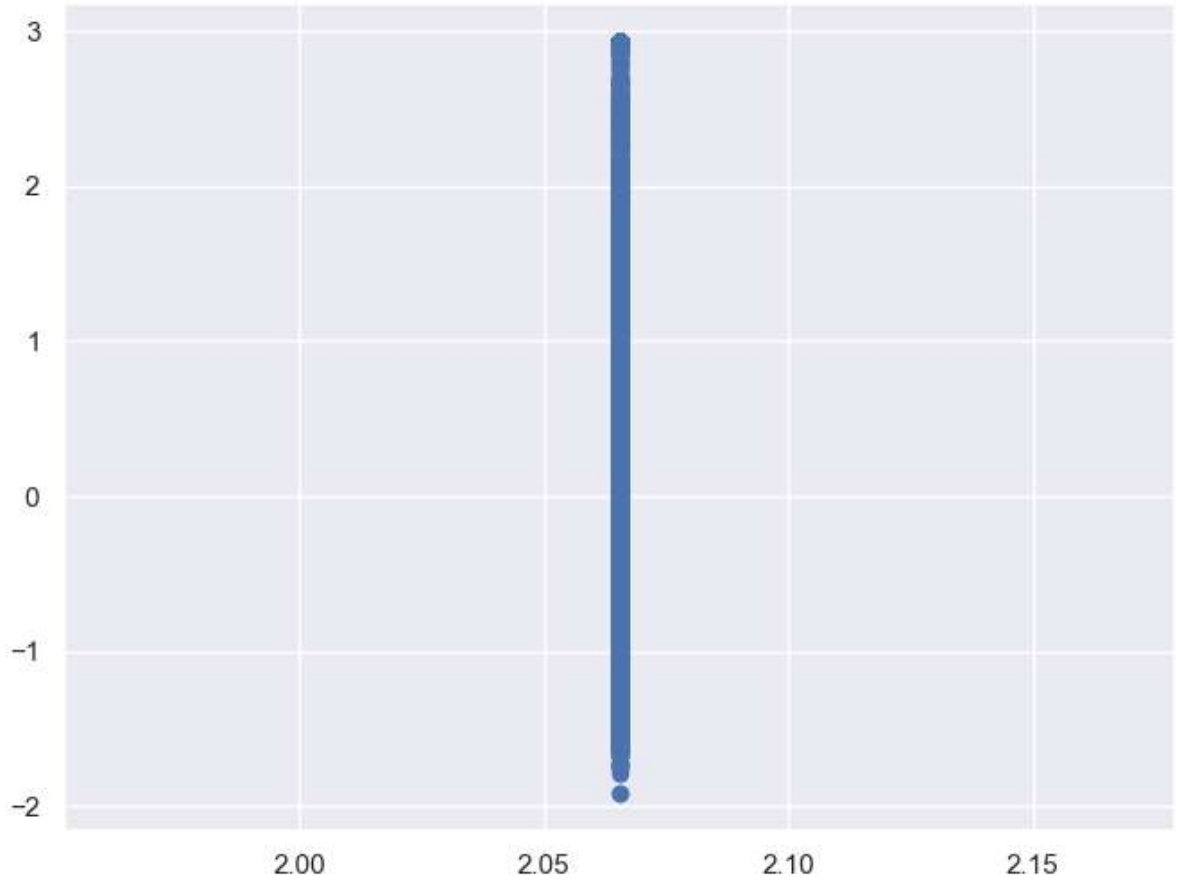
In [100…
```
sns.displot(residual,kind='kde')
```

Out[100]:
```
<seaborn.axisgrid.FacetGrid at 0x244f40e8520>
```



In [101…
```
plt.scatter(lasso_predict,residual)
```

Out[101]:
```
<matplotlib.collections.PathCollection at 0x244f4dbc550>
```



Performance Matrics

In [102…
```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
print(mean_squared_error(y_test,lasso_predict))
print(mean_absolute_error(y_test,lasso_predict))
print(np.sqrt(mean_squared_error(y_test,lasso_predict)))
```

```
1.3434801558184395
0.9136594481730159
1.1590859139073513
```

R Square Value and Adjusted R Square

In [103…
```
from sklearn.metrics import r2_score
score=r2_score(y_test,lasso_predict)
score
```

Out[103]:
```
-7.365224208366605e-05
```

In [104…
```
# Adjusted R Square

1-(1-score)*(len(y_test)-1)/(len(y_test)-x_test.shape[1]-1)
```

Out[104]: -0.0012496906395460528

In [126…
```
df1=pd.DataFrame({'Actual':y_test,'Predict':lasso_predict})
df1
```

Out[126]:

	Actual	Predict
6906	2.114	2.065275
767	1.952	2.065275
10555	2.418	2.065275
17456	1.283	2.065275
20617	0.708	2.065275
...	...	...
861	2.169	2.065275
10326	2.832	2.065275
19588	1.786	2.065275
12146	1.101	2.065275
425	2.849	2.065275

6812 rows × 2 columns

Elasticnet Regression Model

In [106…
```
from sklearn.linear_model import ElasticNet

elastic=ElasticNet()
elastic
```

Out[106]: ▾ ElasticNet

ElasticNet()

In [107…
```
elastic.fit(x_train,y_train)
```

Out[107]: ▾ ElasticNet

ElasticNet()

In [108…
```
print(elastic.coef_)
```

```
[ 0.19553816  0.          0.         -0.         -0.         -0.
 -0.         -0.        ]
```

In [109…
```
print(elastic.intercept_)
```

```
2.065275269742551
```

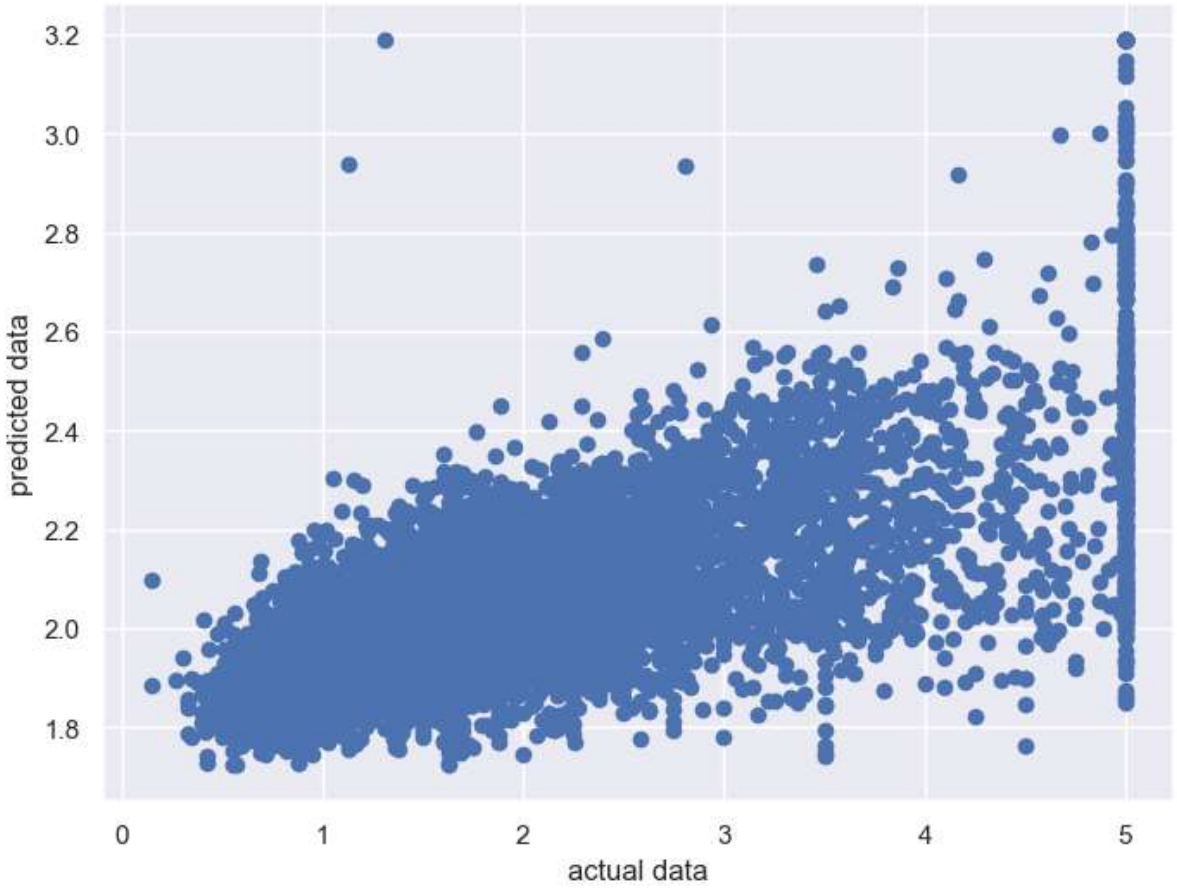
In [112…
```
elastic_pred=elastic.predict(x_test)
elastic_pred
```

Out[112]:
```
array([2.11399735, 1.95797625, 2.13686051, ..., 2.03857823, 1.92900076,
       2.10558153])
```

Assumption of Elasticnet Regression model

In [116…
```
plt.scatter(y_test,elastic_pred)
plt.xlabel('actual data')
plt.ylabel('predicted data')
```

Out[116]:
```
Text(0, 0.5, 'predicted data')
```



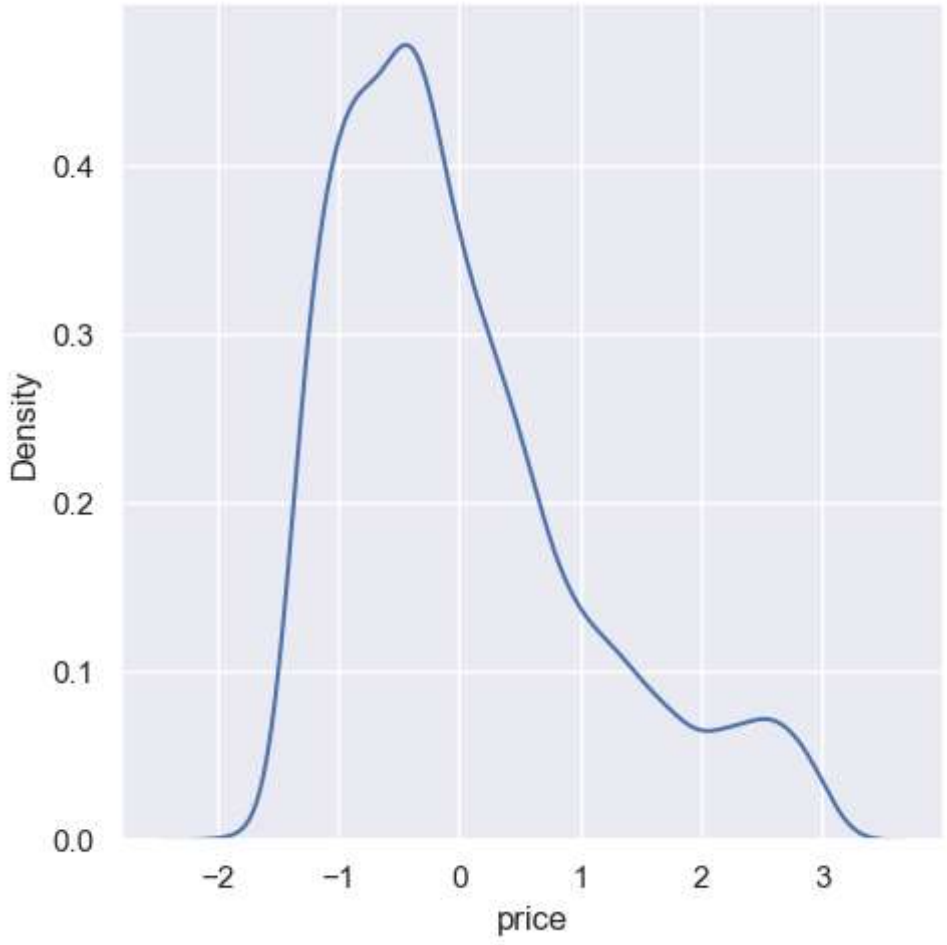
In [117…
```
residual = y_test-elastic_pred
```

In [118…
```
residual
```

Out[118]:
```
6906     0.000003
767     -0.005976
10555    0.281139
17456   -0.753982
20617   -1.298653
           ...
861      0.123845
10326    0.485087
19588   -0.252578
12146   -0.828001
425      0.743418
Name: price, Length: 6812, dtype: float64
```

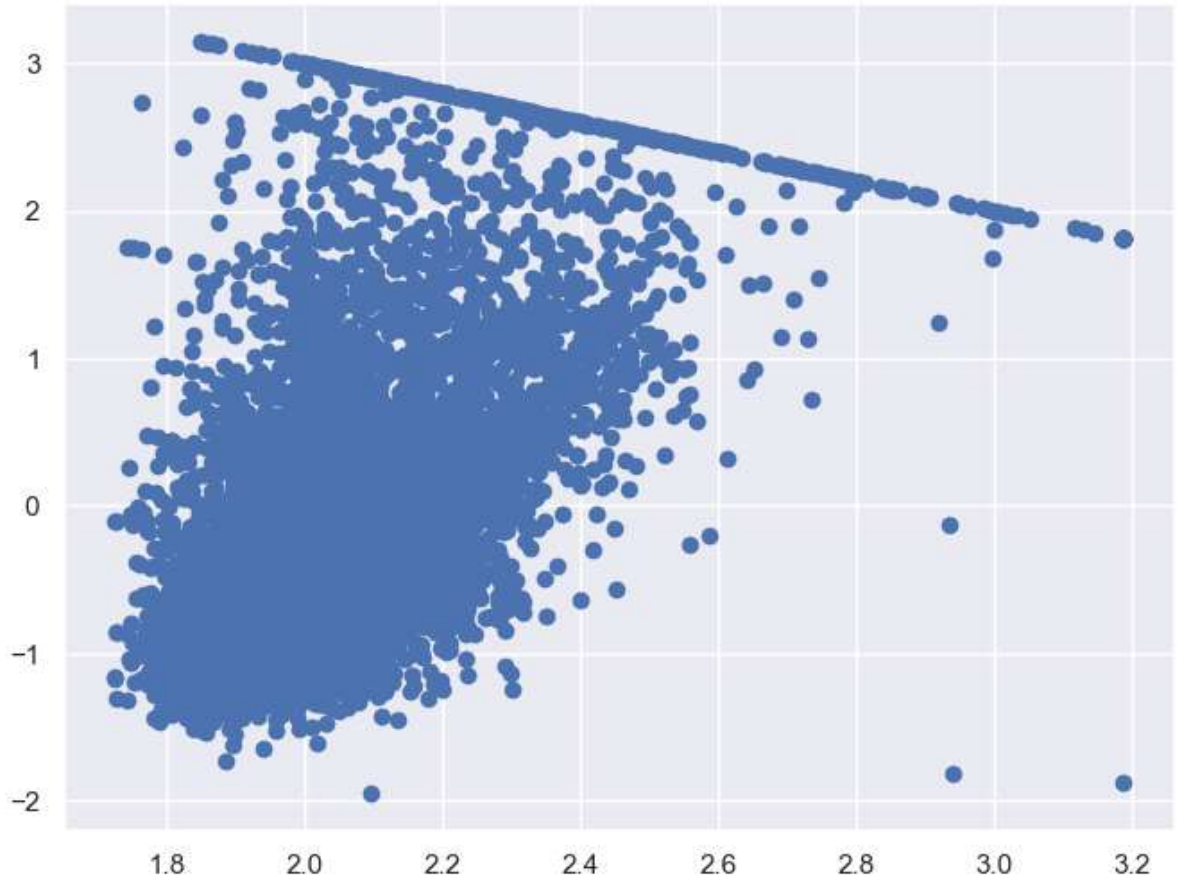
In [119…
```
sns.displot(residual,kind='kde')
```

Out[119]:
```
<seaborn.axisgrid.FacetGrid at 0x244f4e4c250>
```



In [120…
```
plt.scatter(elastic_pred,residual )
```

Out[120]:
```
<matplotlib.collections.PathCollection at 0x244f4dbcb50>
```



Perfomance Metrics

In [121…
```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
print(mean_squared_error(y_test,elastic_pred))
print(mean_absolute_error(y_test,elastic_pred))
print(np.sqrt(mean_squared_error(y_test,elastic_pred)))
```

```
1.0706121914761009
0.8144919283598954
1.0347039148839154
```

R Square and Adjusted R Square

In [123…
```
from sklearn.metrics import r2_score
score=r2_score(y_test,elastic_pred)
score
```

Out[123]:
```
0.2030466249706926
```

In [124…
```
# Adjusted R Square

1-(1-score)*(len(y_test)-1)/(len(y_test)-x_test.shape[1]-1)
```

Out[124]: 0.2021094462259866

In [125…
```
df1=pd.DataFrame({'Actual':y_test,'Predict':elastic_pred})
df1
```

Out[125]:

	Actual	Predict
6906	2.114	2.113997
767	1.952	1.957976
10555	2.418	2.136861
17456	1.283	2.036982
20617	0.708	2.006653
...	...	...
861	2.169	2.045155
10326	2.832	2.346913
19588	1.786	2.038578
12146	1.101	1.929001
425	2.849	2.105582

6812 rows × 2 columns

In [ ]: