

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
from sklearn.datasets import fetch_california_housing
```

```
dataset=fetch_california_housing()
dataset
```

```
{'data': array([[ 8.3252, 41., 6.98412698, ..., 2.55555556,
 37.88, -122.23, ],
 [ 8.3014, 21., 6.23813708, ..., 2.10984183,
 37.86, -122.22, ],
 [ 7.2574, 52., 8.28813559, ..., 2.80225989,
 37.85, -122.24, ],
 ...,
 [ 1.7, 17., 5.20554273, ..., 2.3256351,
 39.43, -121.22, ],
 [ 1.8672, 18., 5.32951289, ..., 2.12320917,
 39.43, -121.32, ],
 [ 2.3886, 16., 5.25471698, ..., 2.61698113,
 39.37, -121.24, ]]),
'target': array([4.526, 3.585, 3.521, ..., 0.923, 0.847, 0.894]),
'frame': None,
'target_names': ['MedHouseVal'],
'feature_names': ['MedInc',
'HouseAge',
'AveRooms',
'AveBedrms',
'Population',
'AveOccup',
'Latitude',
'Longitude'],
'DESCR': '.. _california_housing_dataset:\n\nCalifornia Housing dataset\n-----\n\n**Data Set Characteristics:**\n\nNumber of Instances: 20640\n\nNumber of Attributes: 8 numeric, predictive attributes and the target\n\nAttribute Information:\n - MedInc median income in block group\n - HouseAge median house age in block group\n - AveRooms average number of rooms per household\n - AveBedrms average number of bedrooms per household\n - Population block group population\n - AveOccup average number of household members\n - Latitude block group latitude\n - Longitude block group longitude\n\nMissing Attribute Values: None\n\nThis dataset was obtained from the Statlib repository.\nhttps://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html\n\nThe target variable is the median house value for California districts,\nexpressed in hundreds of thousands of dollars ($100,000).\n\nThis dataset was derived from the 1990 U.S. census, using one row per census\nblock group. A block group is the smallest geographical unit for which the U.S.\nCensus Bureau publishes sample data (a block group typically has a population\nof 600 to 3,000 people).\n\nA household is a group of people residing within a home. Since the average\nnumber of rooms and bedrooms in this dataset are provided per household, these\ncolumns may take surprisingly large values for block groups with few households\nand many empty houses, such as vacation resorts.\n\nIt can be downloaded/loaded using the\nfunc:`sklearn.datasets.fetch_california_housing` function.\n\n.. topic:: References\n\n - Pace, R. Kelley and Ronald Barry, Sparse Spatial Autoregressions,\nStatistics and Probability Letters, 33 (1997) 291-297\n'}
```

```
##make the dataframe of the data
df=pd.DataFrame(dataset.data)
df
```

	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
...	...	...	...	...	...	...	...	...
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 x 8 columns

```
##set column name
df.columns=dataset.feature_names
df.head(4)
```



	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

```
## dependent and independent features
```

```
X=df
```

```
y=dataset.target
```

```
X
```



	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 x 8 columns

```
y
```



```
array([4.526, 3.585, 3.521, ..., 0.923, 0.847, 0.894])
```

```
## train_test_split
```

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test,y_train,y_test=train_test_split(X,y,test_size=0.30,random_state=42)
```

```
X_train
```



	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
7061	4.1312	35.0	5.882353	0.975490	1218.0	2.985294	33.93	-118.02
14689	2.8631	20.0	4.401210	1.076613	999.0	2.014113	32.79	-117.09
17323	4.2026	24.0	5.617544	0.989474	731.0	2.564912	34.59	-120.14
10056	3.1094	14.0	5.869565	1.094203	302.0	2.188406	39.26	-121.00
15750	3.3068	52.0	4.801205	1.066265	1526.0	2.298193	37.77	-122.45
...	...	...	...	...	...	...	...	...
11284	6.3700	35.0	6.129032	0.926267	658.0	3.032258	33.78	-117.96
11964	3.0500	33.0	6.868597	1.269488	1753.0	3.904232	34.02	-117.43
5390	2.9344	36.0	3.986717	1.079696	1756.0	3.332068	34.03	-118.38
860	5.7192	15.0	6.395349	1.067979	1777.0	3.178891	37.58	-121.96
15795	2.5755	52.0	3.402576	1.058776	2619.0	2.108696	37.77	-122.42

14448 rows x 8 columns

```
##standardisation
```

```
from sklearn.preprocessing import StandardScaler
```

```
scaler=StandardScaler()
```

```
X_train=scaler.fit_transform(X_train)
```

X\_train

```
array([[ 0.13350629,  0.50935748,  0.18106017, ..., -0.01082519,
        -0.80568191,  0.78093406],
       [-0.53221805, -0.67987313, -0.42262953, ..., -0.08931585,
        -1.33947268,  1.24526986],
       [ 0.1709897 , -0.36274497,  0.07312833, ..., -0.04480037,
        -0.49664515, -0.27755183],
       ...,
       [-0.49478713,  0.58863952, -0.59156984, ...,  0.01720102,
        -0.75885816,  0.60119118],
       [ 0.96717102, -1.07628333,  0.39014889, ...,  0.00482125,
        0.90338501, -1.18625198],
       [-0.68320166,  1.85715216, -0.82965604, ..., -0.0816717 ,
        0.99235014, -1.41592345]])
```

```
X_test=scaler.transform(X_test)
```

```
## apply linesr regression algo
from sklearn.linear_model import LinearRegression
## cross validation
from sklearn.model_selection import cross_val_score
```

```
##regression object
regression=LinearRegression()
regression.fit(X_train,y_train)
```

```
LinearRegression
LinearRegression()
```

```
mse=cross_val_score(regression,X_train,y_train,scoring='neg_mean_squared_error',cv=10)
mse
```

```
array([-0.56799593, -0.52918854, -0.48456478, -0.51627013, -0.54833859,
       -0.47884524, -0.47828648, -0.55465187, -0.55647389, -0.54248888])
```

```
np.mean(mse)
```

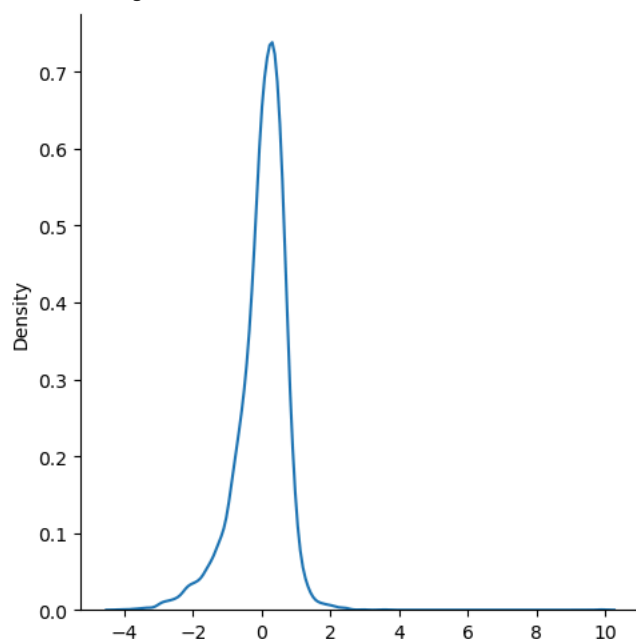
```
-0.5257104326776999
```

```
##prediction of the test data
reg_pred=regression.predict(X_test)
reg_pred
```

```
array([0.72604907, 1.76743383, 2.71092161, ..., 2.07465531, 1.57371395,
       1.82744133])
```

```
import seaborn as sns
sns.displot(reg_pred-y_test,kind='kde')
```

```
<seaborn.axisgrid.FacetGrid at 0x19a1f1b3680>
```



```
from sklearn.metrics import r2_score
```

```
score=r2_score(reg_pred,y_test)
```

```
score
```

```
0.3451339380943963
```

Start coding or [generate](#) with AI.