Linear Regression Algorithm

In [1]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.datasets import fetch_openml
from sklearn.model_selection import cross_val_score
import pickle
```

In [3]:
```
from sklearn.datasets import fetch_california_housing
```

In [4]:
```
df = fetch_california_housing()
df
```

Out[4]:
```
{'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\n:Number of Instances: 20640
\n\n:Number of Attributes: 8 numeric, predictive attributes and the target\n\n:Att
ribute Information:\n    - MedInc        median income in block group\n    - House
Age      median house age in block group\n    - AveRooms      average number of ro
oms per household\n    - AveBedrms     average number of bedrooms per household\n
- Population    block group population\n    - AveOccup      average number of hous
ehold members\n    - Latitude      block group latitude\n    - Longitude     block
group longitude\n\n:Missing Attribute Values: None\n\nThis dataset was obtained fr
om the StatLib repository.\nhttps://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housin
g.html\n\nThe target variable is the median house value for California district
s,\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 publishe
s sample data (a block group typically has a population\nof 600 to 3,000 peopl
e).\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\n
columns may take surprisingly large values for block groups with few households\na
nd many empty houses, such as vacation resorts.\n\nIt can be downloaded/loaded usi
ng the\n:func:`sklearn.datasets.fetch_california_housing` function.\n\n.. rubric::
References\n\n- Pace, R. Kelley and Ronald Barry, Sparse Spatial Autoregression
s,\n  Statistics and Probability Letters, 33:291-297, 1997.\n'}
```

In [5]:
```
housing = fetch_california_housing(as_frame=True)
```

In [6]:
```
df = housing.frame
```

In [7]:
```
x = housing.data      #independent Variable
y = housing.target    #dependent variable
```

In [8]:
```
df.head()
```

Out[8]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedH
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 [9]:
```
df.shape
```

Out[9]:
```
(20640, 9)
```

In [10]:
```
x
```

Out[10]:

	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 [11]:
```
y
```

Out[11]:
```
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: MedHouseVal, Length: 20640, dtype: float64
```

In [13]:
```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.30, random_st
```

In [14]:
```
X_train
```

Out[14]:

	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 × 8 columns

In [15]:

```
X_test
```

Out[15]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
20046	1.6812	25.0	4.192201	1.022284	1392.0	3.877437	36.06	-119.01
3024	2.5313	30.0	5.039384	1.193493	1565.0	2.679795	35.14	-119.46
15663	3.4801	52.0	3.977155	1.185877	1310.0	1.360332	37.80	-122.44
20484	5.7376	17.0	6.163636	1.020202	1705.0	3.444444	34.28	-118.72
9814	3.7250	34.0	5.492991	1.028037	1063.0	2.483645	36.62	-121.93
...	...	...	...	...	...	...	...	...
17505	2.9545	47.0	4.195833	1.020833	581.0	2.420833	37.36	-121.90
13512	1.4891	41.0	4.551852	1.118519	994.0	3.681481	34.11	-117.32
10842	3.5120	16.0	3.762287	1.075614	5014.0	2.369565	33.67	-117.91
16559	3.6500	10.0	5.502092	1.060371	5935.0	3.547519	37.82	-121.28
5786	3.0520	17.0	3.355781	1.019695	4116.0	2.614994	34.15	-118.24

6192 rows × 8 columns

In [16]:

```
y_train
```

Out[16]:
```
7061     1.93800
14689    1.69700
17323    2.59800
10056    1.36100
15750    5.00001
          ...
11284    2.29200
11964    0.97800
5390     2.22100
860      2.83500
15795    3.25000
Name: MedHouseVal, Length: 14448, dtype: float64
```

In [17]:
```
y_test
```

Out[17]:
```
20046    0.47700
3024     0.45800
15663    5.00001
20484    2.18600
9814     2.78000
          ...
17505    2.37500
13512    0.67300
10842    2.18400
16559    1.19400
5786     2.09800
Name: MedHouseVal, Length: 6192, dtype: float64
```

In [18]:
```
#standardizing
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit_transform(X_train)
```

Out[18]:
```
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]])
```

In [19]:
```
X_train = scaler.fit_transform(X_train)
```

In [20]:
```
X_test = scaler.transform(X_test)
```

In [21]:
```
regression =  LinearRegression()
regression.fit(X_train,y_train)
```

Out[21]:
```
▾ LinearRegression
▶ Parameters
```

In [22]:
```
MSE = cross_val_score(regression,X_train,y_train,scoring='neg_mean_squared_error',
```

In [23]:
```
np.mean(MSE)
```

Out[23]:
```
-0.5268253746355749
```

In [24]:
```
np.median(MSE)
```

Out[24]:
```
-0.5204563897534458
```

In [25]:
```
np.var(MSE)
```

Out[25]:
```
0.0003516508699748547
```

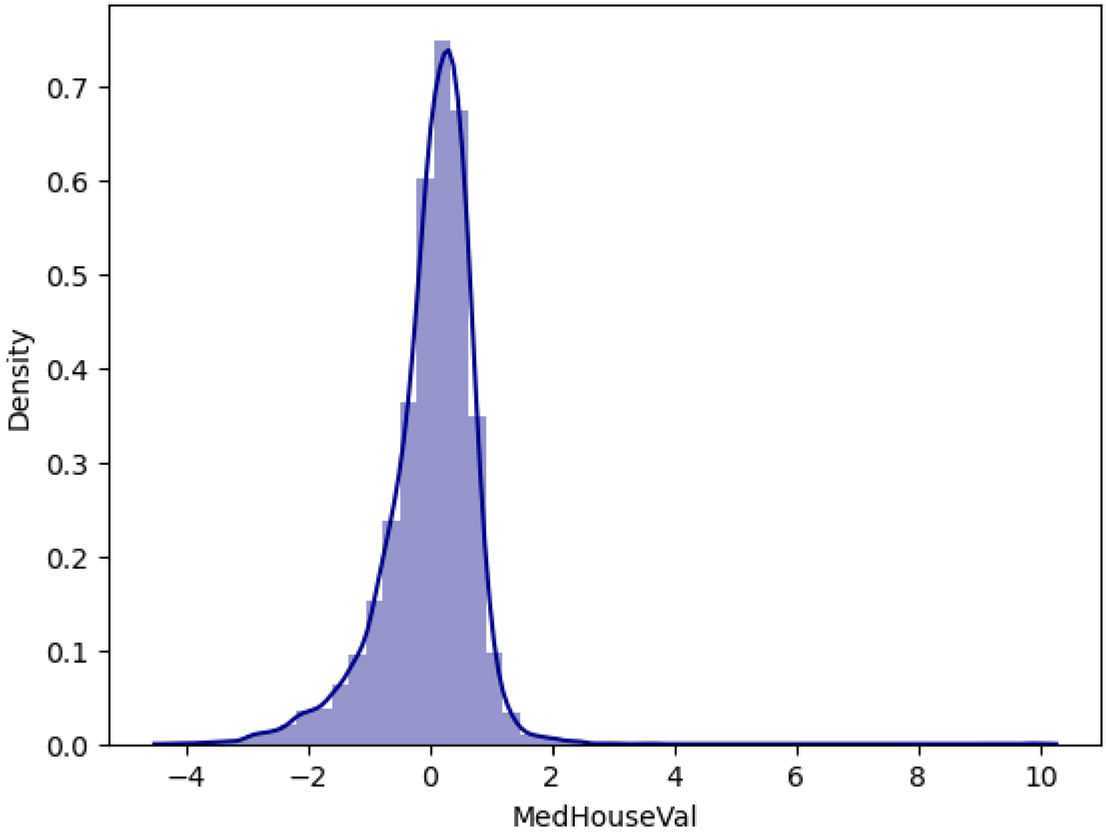
In [26]:
```
#Prediction
reg_pred=regression.predict(X_test)
```

In [27]:
```
reg_pred
```

Out[27]:
```
array([0.72604907, 1.76743383, 2.71092161, ..., 2.07465531, 1.57371395,
       1.82744133])
```

In [28]:
```
sns.distplot(reg_pred-y_test, color = 'darkblue')
```

Out[28]:
```
<Axes: xlabel='MedHouseVal', ylabel='Density'>
```



In [29]:
```
score = r2_score(reg_pred,y_test)
```

In [30]:
```
score
```

Out[30]:
```
0.3451339380943981
```

In [ ]:
```

```

In [ ]:
```

```

```
In [ ]:
```