In [100…
```
from sklearn.datasets import fetch_california_housing
```

In [101…
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [102…
```
california=fetch_california_housing()
```

In [103…
```
california
```

Out[103]:
```
{'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: 2
0640\n\n    :Number of Attributes: 8 numeric, predictive attributes and the target
\n\n    :Attribute Information:\n        - MedInc        median income in block gr
oup\n        - HouseAge      median house age in block group\n        - AveRooms
average number of rooms per household\n        - AveBedrms     average number of b
edrooms per household\n        - Population    block group population\n        - A
veOccup      average number of household members\n        - Latitude      block gr
oup latitude\n        - Longitude     block group longitude\n\n    :Missing Attrib
ute Values: None\n\nThis dataset was obtained from the StatLib repository.\nhttp
s://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 thousan
ds of dollars ($100,000).\n\nThis dataset was derived from the 1990 U.S. census, u
sing one row per census\nblock group. A block group is the smallest geographical u
nit for which the U.S.\nCensus Bureau publishes sample data (a block group typical
ly has a population\nof 600 to 3,000 people).\n\nAn household is a group of people
residing within a home. Since the average\nnumber of rooms and bedrooms in this da
taset are provided per household, these\ncolumns may take surpinsingly large value
s for block groups with few households\nand many empty houses, such as vacation re
sorts.\n\nIt can be downloaded/loaded using the\n:func:`sklearn.datasets.fetch_cal
ifornia_housing` function.\n\n.. topic:: References\n\n    - Pace, R. Kelley and R
onald Barry, Sparse Spatial Autoregressions,\n      Statistics and Probability Let
ters, 33 (1997) 291-297\n'}
```

In [71]:
```
california.keys()
```

Out[71]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'feature_names', 'DESCR'])
```

In [8]:
```
print(california.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).

An 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 surpinsingly 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.

.. topic:: References

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

In [9]:
```
california.data.shape
```

Out[9]:
```
(20640, 8)
```

In [10]:
```
california.target_names
```

Out[10]:
```
['MedHouseVal']
```

In [11]:
```
california.feature_names
```

Out[11]:
```
['MedInc',
 'HouseAge',
 'AveRooms',
 'AveBedrms',
 'Population',
 'AveOccup',
 'Latitude',
 'Longitude']
```

In [12]:
```
california.target
```

Out[12]:
```
array([4.526, 3.585, 3.521, ..., 0.923, 0.847, 0.894])
```

In [14]:
```
#lets prepare dataset

df=pd.DataFrame(california.data,columns=california.feature_names)
```

In [16]:
```
df['Price']=california.target
```

In [17]:
```
df.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 [18]:
```
df.describe()
```

Out[18]:

	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 [19]:

```
df.isnull().sum()
```

Out[19]:

```
MedInc        0
HouseAge      0
AveRooms      0
AveBedrms     0
Population    0
AveOccup      0
Latitude      0
Longitude     0
Price         0
dtype: int64
```

In [20]:

```
df.corr()
```

Out[20]:

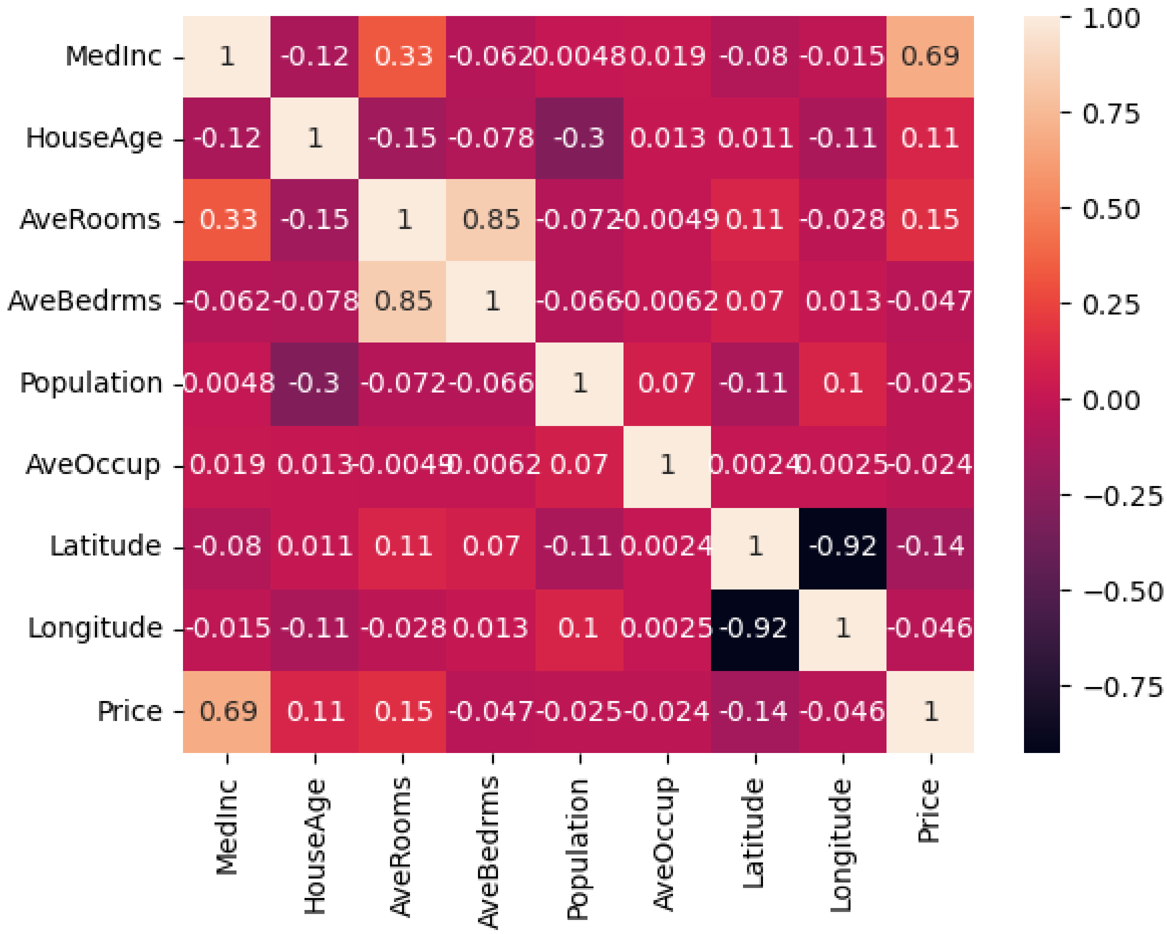
	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Long
MedInc	1.000000	-0.119034	0.326895	-0.062040	0.004834	0.018766	-0.079809	-0.0
HouseAge	-0.119034	1.000000	-0.153277	-0.077747	-0.296244	0.013191	0.011173	-0.1
AveRooms	0.326895	-0.153277	1.000000	0.847621	-0.072213	-0.004852	0.106389	-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 [21]:

```
sns.heatmap(df.corr(),annot=True)
```

Out[21]:

```
<AxesSubplot: >
```



In [22]:
```
df.head()
```

Out[22]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	Price
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422

In [23]:
```
#independent and dependent
x=df.iloc[:,:-1]
y=df.iloc[:,-1]
```

In [24]:
```
x.head()
```

Out[24]:

	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 [25]:
```
y
```

Out[25]:
```
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 [26]:
```
from sklearn.model_selection import train_test_split
```

In [27]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=10)
```

In [28]:
```
x_train.shape
```

Out[28]:
```
(13828, 8)
```

In [29]:
```
y_train.shape
```

Out[29]:
```
(13828,)
```

In [30]:
```
x_test.shape
```

Out[30]:
```
(6812, 8)
```

In [31]:
```
y_test.shape
```

Out[31]:
```
(6812,)
```

In [32]:
```
#make every ndependent feature become scaling
```

In [34]:
```
from sklearn.preprocessing import StandardScaler
```

In [35]:
```
scaler=StandardScaler()
```

In [36]:
```
x_train_scaled=scaler.fit_transform(x_train)
```

In [37]:
```
X_test_scaled=scaler.transform(x_test)
```

In [74]:
```
X_test_scaled
```

Out[74]:
```
array([[ 0.75154854, -1.31428337, -0.39376169, ...,  0.12606697,
        -0.68820027,  0.19491761],
       [ 0.05935857, -0.12595418, -0.33070668, ..., -0.12021013,
         0.89459042, -1.36503888],
       [ 0.34405687, -1.31428337, -0.41007104, ..., -0.15581759,
        -0.91698123,  0.89764561],
       ...,
       [ 0.36483158,  0.27015554,  0.04216837, ..., -0.08014641,
        -0.46875731, -0.43803598],
       [-0.90412152, -0.91817364,  0.66736933, ..., -0.10263685,
         2.51006411, -1.96808915],
       [-0.43377577,  1.22081889, -0.44835491, ...,  0.2807072 ,
        -0.74422826,  0.69330627]])
```

In [75]:
```
from sklearn.linear_model import LinearRegression
```

In [76]:
```
regression=LinearRegression()
```

In [77]:
```
regression
```

Out[77]:
```
▾ LinearRegression
LinearRegression()
```

In [78]:
```
regression.fit(x_train_scaled,y_train)
```

Out[78]:
```
▾ LinearRegression
LinearRegression()
```

In [79]:
```
regression.coef_
```

Out[79]:
```
array([ 0.82872299,  0.1231163 , -0.27068752,  0.32859106,  0.00213572,
       -0.02810091, -0.93017985, -0.89505497])
```

In [80]:
```
regression.intercept_
```

Out[80]:
```
2.0634768086491184
```

In [81]:
```
#prediction
y_pred_test=regression.predict(X_test_scaled)
```

In [84]:
```
y_pred_test
```

Out[84]:
```
array([3.00397485, 2.58011486, 2.3489077 , ..., 3.09003708, 0.79152007,
       2.04477012])
```

In [85]:
```
#performance metrics
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
```

In [86]:
```
mean_squared_error(y_test,y_pred_test)
```

Out[86]:
```
0.5522332399363619
```

In [87]:
```
mean_absolute_error(y_test,y_pred_test)
```

Out[87]:
```
0.537105694300796
```

In [88]:
```
np.sqrt(mean_squared_error(y_test,y_pred_test))
```

Out[88]:
```
0.7431239734636219
```

In [89]:
```
#rsquare and adj r square
from sklearn.metrics import r2_score
```

In [90]:
```
score=r2_score(y_test,y_pred_test)
```

In [91]:
```
score
```

Out[91]:
```
0.593595852643664
```

In [92]:
```
#
(1-(1-score)*(len(y_test)-1)/len(y_test)-x_test.shape[1]-1)
```

Out[92]:
```
-8.406344487322961
```

In [96]:
```
import pickle
```

In [97]:
```
pickle.dump(scaler,open('scaler.pkl','wb'))
```

In [98]:
```
pickle.dump(regression,open("regressor.pkl","wb"))
```

In [99]:
```
#load file
```

In [105...
```
model_regressor=pickle.load(open('regressor.pkl','rb'))
```

In [110...
```
model_regressor.predict(x_test)
```

```
/opt/conda/lib/python3.10/site-packages/sklearn/base.py:402: UserWarning: X has fe
ature names, but LinearRegression was fitted without feature names
  warnings.warn(
```

Out[110]:
```
array([82.68061719, 86.28203242, 84.56071577, ..., 85.87769366,
       77.99457178, 85.83207744])
```

In [111…
```
standard_scaler=pickle.load(open('scaler.pkl','rb'))
```

In [112…
```
model_regressor.predict(standard_scaler.transform(x_test))
```

Out[112]:
```
array([3.00397485, 2.58011486, 2.3489077 , ..., 3.09003708, 0.79152007,
       2.04477012])
```

In [ ]: