Importing Libraries

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
```

In [2]:

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

In [3]:

```
# Load the dataset
dataset=fetch_california_housing()
print(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, ..., 0.92
3, 0.847, 0.894]), 'frame': None, 'target_names': ['MedHouseVal'], 'feature_names': ['Med
Inc', 'HouseAge', 'AveRooms', 'AveBedrms', 'Population', 'AveOccup', 'Latitude', 'Longitu
de'], '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    :Attribute
Information:\n        - MedInc        median income in block group\n        - HouseAge
median house age in block group\n        - AveRooms      average number of rooms per hous
ehold\n        - AveBedrms     average number of bedrooms per household\n        - Popul
ation    block group population\n        - AveOccup      average number of household memb
ers\n        - Latitude      block group latitude\n        - Longitude     block group l
ongitude\n\n    :Missing Attribute Values: None\n\nThis dataset was obtained from the Sta
tLib repository.\nhttps://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html\n\nThe tar
get variable is the median house value for California districts,\nexpressed in hundreds o
f 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 fo
r which the U.S.\nCensus Bureau publishes sample data (a block group typically has a popu
lation\nof 600 to 3,000 people).\n\nAn household is a group of people residing within a h
ome. Since the average\nnumber of rooms and bedrooms in this dataset are provided per hou
sehold, these\ncolumns may take surpinsingly large values for block groups with few house
holds\nand many empty houses, such as vacation resorts.\n\nIt can be downloaded/loaded us
ing the\n:func:`sklearn.datasets.fetch_california_housing` function.\n\n.. topic:: Refere
nces\n\n    - Pace, R. Kelley and Ronald Barry, Sparse Spatial Autoregressions,\n      St
atistics and Probability Letters, 33 (1997) 291-297\n'}
```

In [4]:

```
print(dataset.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 [5]:

```
print(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, ..., 0.92
3, 0.847, 0.894]), 'frame': None, 'target_names': ['MedHouseVal'], 'feature_names': ['Med
Inc', 'HouseAge', 'AveRooms', 'AveBedrms', 'Population', 'AveOccup', 'Latitude', 'Longitu
de'], '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    :Attribute
Information:\n        - MedInc        median income in block group\n        - HouseAge
median house age in block group\n        - AveRooms      average number of rooms per hous
ehold\n        - AveBedrms     average number of bedrooms per household\n        - Popul
ation    block group population\n        - AveOccup      average number of household memb
ers\n        - Latitude      block group latitude\n        - Longitude     block group l
ongitude\n\n    :Missing Attribute Values: None\n\nThis dataset was obtained from the Sta
tLib repository.\nhttps://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html\n\nThe tar
get variable is the median house value for California districts,\nexpressed in hundreds o
f 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 fo
```

```
r which the U.S.\nCensus Bureau publishes sample data (a block group typically has a popu
lation\nof 600 to 3,000 people).\n\nAn household is a group of people residing within a h
ome. Since the average\nnumber of rooms and bedrooms in this dataset are provided per hou
sehold, these\ncolumns may take surpinsingly large values for block groups with few house
holds\nand many empty houses, such as vacation resorts.\n\nIt can be downloaded/loaded us
ing the\n:func:`sklearn.datasets.fetch_california_housing` function.\n\n.. topic:: Refere
nces\n\n    - Pace, R. Kelley and Ronald Barry, Sparse Spatial Autoregressions,\n      St
atistics and Probability Letters, 33 (1997) 291-297\n'}
```

In [6]:

```
print(dataset.feature_names)
```

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

In [7]:

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

Out[7]:

	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 [8]:

```
# Price is a target column
df['Price']=dataset.target
```

In [9]:

```
# Top 5 rows
df.head()
```

Out[9]:

	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 [10]:

```
#Last 5 rows
df.tail(5)
```

Out[10]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	Price
20635	1.5603	25.0	5.045455	1.133333	845.0	2.560606	39.48	-121.09	0.781
20636	2.5568	18.0	6.114035	1.315789	356.0	3.122807	39.49	-121.21	0.771
20637	1.7000	17.0	5.205543	1.120092	1007.0	2.325635	39.43	-121.22	0.923
20638	1.8672	18.0	5.329513	1.171920	741.0	2.123209	39.43	-121.32	0.847

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	Price
20639	2.3886	16.0	5.254717	1.162264	1387.0	2.616981	39.37	-121.24	0.894

In [11]:

```
df.describe()
```

Out[11]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	
count	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	206
mean	3.870671	28.639486	5.429000	1.096675	1425.476744	3.070655	35.631861	-119.569704	
std	1.899822	12.585558	2.474173	0.473911	1132.462122	10.386050	2.135952	2.003532	
min	0.499900	1.000000	0.846154	0.333333	3.000000	0.692308	32.540000	-124.350000	
25%	2.563400	18.000000	4.440716	1.006079	787.000000	2.429741	33.930000	-121.800000	
50%	3.534800	29.000000	5.229129	1.048780	1166.000000	2.818116	34.260000	-118.490000	
75%	4.743250	37.000000	6.052381	1.099526	1725.000000	3.282261	37.710000	-118.010000	
max	15.000100	52.000000	141.909091	34.066667	35682.000000	1243.333333	41.950000	-114.310000	

In [12]:

```
# Check null values present or not
df.isnull().sum()
```

Out[12]:

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

In [18]:

```
#Check column names
df.columns
```

Out[18]:

```
Index(['MedInc', 'HouseAge', 'AveRooms', 'AveBedrms', 'Population', 'AveOccup',
       'Latitude', 'Longitude', 'Price'],
      dtype='object')
```

In [14]:

```
df_copy=df.sample(frac=0.25)
```

In [15]:

```
df_copy.shape
```

Out[15]:

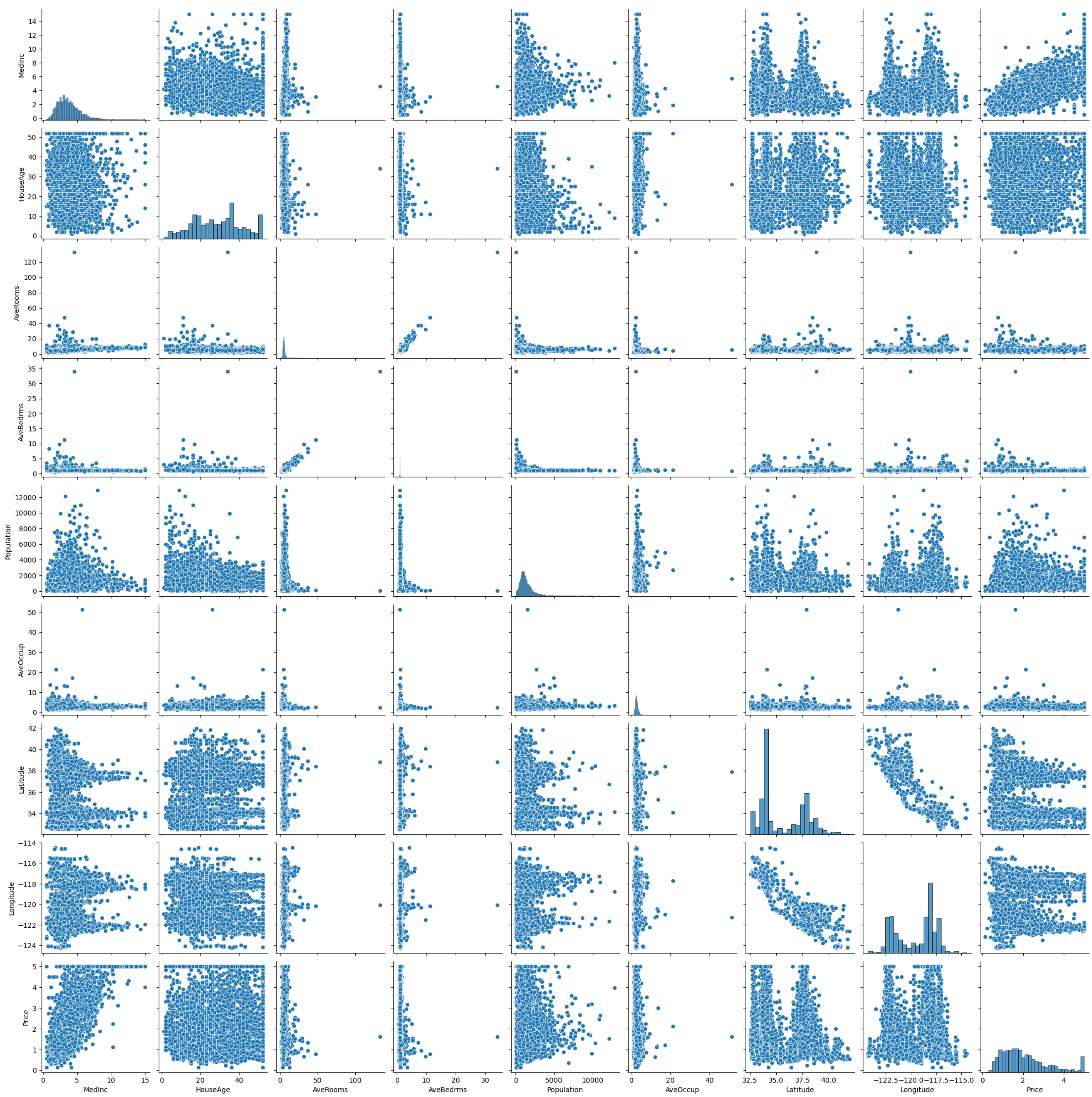
```
(5160, 9)
```

In [16]:

```
sns.pairplot(df_copy)
```

Out[16]:

```
<seaborn.axisgrid.PairGrid at 0x235a2c8dc70>
```



In [17]:

```
df.head()
```

Out[17]:

	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

Divide the dataset into independent and dependent variable

In [19]:

```
## Divide  the dataset into independent and dependent
```

```
X=df.iloc[:,:-1]
y=df.iloc[:,-1]
```

Divide the dataset into train and test

In [20]:

```
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=36)
```

In [21]:

```
X.shape
```

Out[21]:

```
(20640, 8)
```

In [22]:

```
X_train.shape,X_test.shape
```

Out[22]:

```
((13828, 8), (6812, 8))
```

Feature scaling-Standardization

In [23]:

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
```

In [24]:

```
scaler.fit(X_train)
```

Out[24]:

```
StandardScaler()
```

In [25]:

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

In [26]:

```
X_train
```

Out[26]:

```
array([[ 2.94615603,  1.69257444,  0.65840852, ...,  0.00631568,
        -0.67676958,  0.7291751 ],
       [-0.53712836,  0.26178584, -0.61551141, ...,  0.10882511,
        -0.78878315,  0.59982719],
       [ 0.02597188,  0.50025061, -0.14510172, ..., -0.11288116,
         0.79807576, -1.19114398],
       ...,
       [-0.46390242,  0.57973886, -0.72060079, ...,  0.24671234,
        -0.77478146,  0.69932558],
       [ 0.03045085, -0.85104973,  0.29681501, ..., -0.04515438,
         1.27880067, -1.66873629],
       [ 0.48355594, -0.69207322, -0.02275645, ..., -0.12518059,
         0.95676166, -1.25084302]])
```

In [27]:

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

Linear Regression

In [28]:

```
from sklearn.linear_model import LinearRegression
```

In [29]:

```
regression=LinearRegression()
```

In [30]:

```
regression.fit(X_train,y_train)
```

Out[30]:

```
LinearRegression()
```

In [31]:

```
regression.coef_
```

Out[31]:

```
array([ 0.83700024,  0.12271899, -0.26347102,  0.30713139, -0.0081633 ,
       -0.02764702, -0.90609856, -0.87576409])
```

In [32]:

```
regression.intercept_
```

Out[32]:

```
2.0708259184263804
```

In [33]:

```
## prediction
y_pred=regression.predict(X_test)
```

In [34]:

```
from sklearn.metrics import mean_squared_error,mean_absolute_error
```

In [35]:

```
import numpy as np
mse=mean_squared_error(y_test,y_pred)
print(mse)
mae=mean_absolute_error(y_test,y_pred)
print(mae)
print(np.sqrt(mse))
```

```
0.533502915515714
0.540898948179417
0.7304128390956132
```

In [36]:

```
## Accuracy r2 and adjusted r square
from sklearn.metrics import r2_score
```

In [37]:

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

In [38]:

```
score
```

Out[38]:

```
0.5875394343499214
```

In [39]:

```
#display adjusted R-squared
1 - (1-score)*(len(y)-1)/(len(y)-X.shape[1]-1)
```

Out[39]:

```
0.5873794961731389
```

In [40]:

```
from sklearn.linear_model import Ridge
ridge=Ridge(alpha=20.0)
ridge.fit(X_train,y_train)
```

Out[40]:

```
Ridge(alpha=20.0)
```

In [41]:

```
y_pred=ridge.predict(X_test)
```

In [42]:

```
mse=mean_squared_error(y_test,y_pred)
print(mse)
mae=mean_absolute_error(y_test,y_pred)
print(mae)
print(np.sqrt(mse))
```

```
0.5335706984910803
0.5408065397594833
0.7304592380763489
```

Lasso Regression

In [43]:

```
from sklearn.linear_model import Lasso
```

In [44]:

```
lasso=Lasso(alpha=20.0)
lasso.fit(X_train,y_train)
```

Out[44]:

```
Lasso(alpha=20.0)
```

In [45]:

```
y_pred=lasso.predict(X_test)
mse=mean_squared_error(y_test,y_pred)
print(mse)
mae=mean_absolute_error(y_test,y_pred)
print(mae)
print(np.sqrt(mse))
```

```
1.2935112672240048
0.9000629779192262
1.137263679454568
```

ElasticNet Regression

In [46]:

```
from sklearn.linear_model import ElasticNet
```

In [47]:

```
elasticnet=ElasticNet(alpha=20.0)
elasticnet.fit(X_train,y_train)
```

Out[47]:

```
ElasticNet(alpha=20.0)
```

In [48]:

```
y_pred=elasticnet.predict(X_test)
mse=mean_squared_error(y_test,y_pred)
print(mse)
mae=mean_absolute_error(y_test,y_pred)
print(mae)
print(np.sqrt(mse))
```

```
1.2935112672240048
0.9000629779192262
1.1373263679454568
```

In [49]:

```
df_copy.corr()
```

Out[49]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	Price
MedInc	1.000000	-0.115862	0.304895	-0.052281	0.013438	-0.037142	-0.089293	-0.005679	0.683772
HouseAge	-0.115862	1.000000	-0.145495	-0.063038	-0.317230	-0.010246	0.009996	-0.107215	0.112032
AveRooms	0.304895	-0.145495	1.000000	0.875202	-0.063717	-0.040469	0.099176	-0.026189	0.121449
AveBedrms	-0.052281	-0.063038	0.875202	1.000000	-0.061585	-0.045989	0.062055	0.007424	-0.056561
Population	0.013438	-0.317230	-0.063717	-0.061585	1.000000	0.153970	-0.105808	0.097867	-0.028001
AveOccup	-0.037142	-0.010246	-0.040469	-0.045989	0.153970	1.000000	-0.097197	0.099202	-0.181162
Latitude	-0.089293	0.009996	0.099176	0.062055	-0.105808	-0.097197	1.000000	-0.923958	-0.145424
Longitude	-0.005679	-0.107215	-0.026189	0.007424	0.097867	0.099202	-0.923958	1.000000	-0.043077
Price	0.683772	0.112032	0.121449	-0.056561	-0.028001	-0.181162	-0.145424	-0.043077	1.000000