HOUSE PRICE PREDICTION

Phase:4. Development Part 2

In this part you will continue building your project. Continue building the house price prediction model by Feature selection Model training Evaluation.

In [1]:

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

In [2]:

```
df=pd.read_csv("/kaggle/input/usa-housing/USA_Housing.csv")
veri = df.copy()
veri.head()
```

Out[2]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Views Suite 079\nLake Kathleen, CA...
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabeth Stravenue\nDanieltown, WI 06482...
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO AP 44820
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFPO AE 09386

For feature selection, we can employ techniques like correlation analysis, backward elimination, or Lasso regularization to identify the most influential features impacting house prices. By selecting the most significant attributes, we can enhance the model's predictive capabilities and streamline its performance.

Model Training:**

We will use various regression techniques such as linear regression, decision trees, random forests, or gradient boosting to train the house price prediction model. By feeding the model with the selected features and the corresponding target variable, we can enable it to learn the complex patterns within the data and make accurate predictions.

In [5]:
```
df.describe().T
```

Out[5]:

	count	mean	std	min	25%	50%	75%
Avg. Area Income	5000.0	6.858311e+04	10657.991214	17796.631190	61480.562388	6.880429e+04	7.5783
Avg. Area House Age	5000.0	5.977222e+00	0.991456	2.644304	5.322283	5.970429e+00	6.6508
Avg. Area Number of Rooms	5000.0	6.987792e+00	1.005833	3.236194	6.299250	7.002902e+00	7.6658
Avg. Area Number of Bedrooms	5000.0	3.981330e+00	1.234137	2.000000	3.140000	4.050000e+00	4.4900
Area Population	5000.0	3.616352e+04	9925.650114	172.610686	29403.928702	3.619941e+04	4.2861
Price	5000.0	1.232073e+06	353117.626581	15938.657923	997577.135049	1.232669e+06	1.4712

Evaluation:

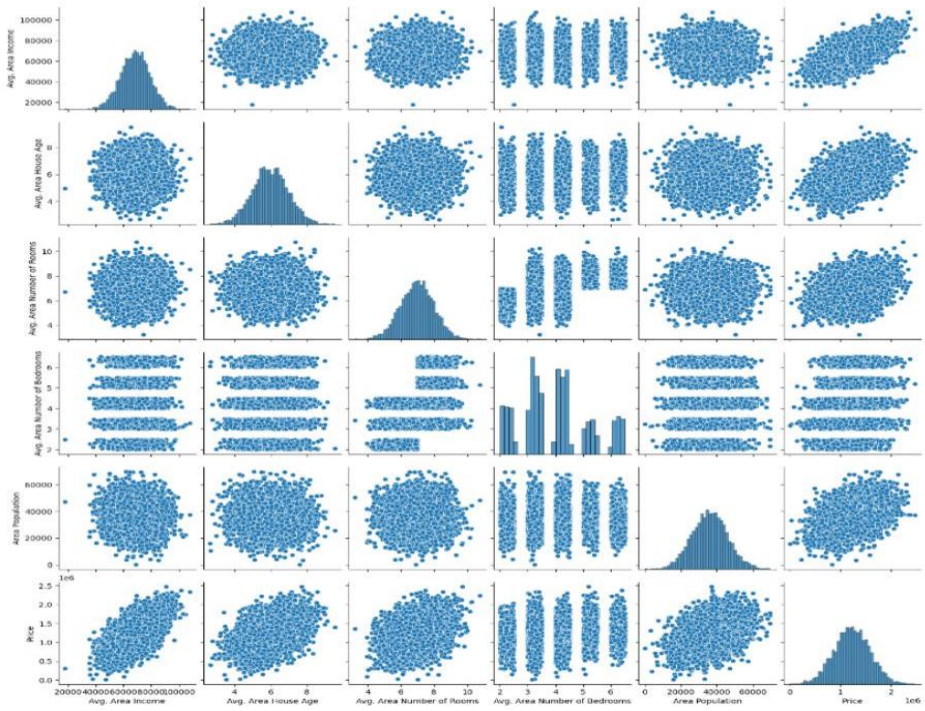
To evaluate the model's performance, we will utilize metrics such as mean squared error, mean absolute error, and R-squared to assess how well the model predicts house prices. Additionally, we'll employ techniques like cross-validation to ensure the model's robustness and prevent overfitting, thereby guaranteeing its effectiveness in real-world scenarios.

In [6]:
```
sns.pairplot(veri)
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/axisgrid.py:118: UserWarning: The f
igure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[6]:

```
<seaborn.axisgrid.PairGrid at 0x7819c8fe0d90>
```



In [7]:
```
sns.heatmap(veri.corr(),annot=True)
```

Out[7]:
```
<Axes: >
```

In [10]:
```
from sklearn.preprocessing import StandardScaler

ss = StandardScaler()

X_train = ss.fit_transform(X_train)
X_test = ss.transform(X_test)
```

In [11]:
```
import sklearn.metrics as mt

def cross_val(model):
    vali=cross_val_score(model,X,y,cv=10)
    return vali.mean()
def success(true_,pred):
    rmse=mt.mean_absolute_error(true_,pred)
    r2=mt.r2_score(true_,pred)
    return[rmse,r2]
```

In [12]:

```
from sklearn.linear_model import LinearRegression,Ridge,Lasso,Elas
ticNet
li_model=LinearRegression()
li_model.fit(X_train,y_train)
li_pred = li_model.predict(X_test)

ridge_model=Ridge(alpha=0.1)
ridge_model.fit(X_train,y_train)
ridge_pred = ridge_model.predict(X_test)

lasso_model=Lasso(alpha=0.1)
lasso_model.fit(X_train,y_train)
lasso_pred = lasso_model.predict(X_test)


elas_model=ElasticNet(alpha=0.1)
elas_model.fit(X_train,y_train)
elas_pred = elas_model.predict(X_test)
```

In [13]:

```
result=[["Linear model",success(y_test,li_pred)[0],success(y_test,
li_pred)[1],cross_val(li_model)],
        ["Ridge model",success(y_test,ridge_pred)[0],success(y_tes
t,ridge_pred)[1],cross_val(ridge_model)],
        ["Lasso model",success(y_test,lasso_pred)[0],success(y_tes
t,lasso_pred)[1],cross_val(lasso_model)],
        ["ElasticNet model",success(y_test,elas_pred)[0],success(y
_test,elas_pred)[1],cross_val(elas_model)]
        ]
pd.options.display.float_format="{:.4f}".format
result=pd.DataFrame(result,columns=["Model","RMSE","R2","Verificat
ion"])
result
```

Out[13]:

	Model	RMSE	R2	Verification
0	Linear model	80879.0972	0.9180	0.9174
1	Ridge model	80878.9638	0.9180	0.9174
2	Lasso model	80879.0910	0.9180	0.9174
3	ElasticNet model	81617.9048	0.9157	0.9165

In [ ]: