HOUSE PRICE PREDICTION ANALYSIS

USING MACHINE LEARNING

PHASE 3:DEVELOPMENT PART 1

NOTEBOOK LINK:

https://colab.research.google.com/drive/1hv4Br_go8VEC9zGaZ-av9bhd8XFzZJKs?authuser=1#scrollTo=HCAzZMl7fueV

LOADING THE DATASET:

C0DE:

```
import pandas as pd

data = pd.read_csv('/content/dataset.csv.csv')

# Display the first few rows of the dataset to inspect the data

print(data.head())
```

EXPLANATION:

1.Importing Libraries:

In this part, we first import the necessary libraries. We import pandas, a powerful library for data manipulation in Python.

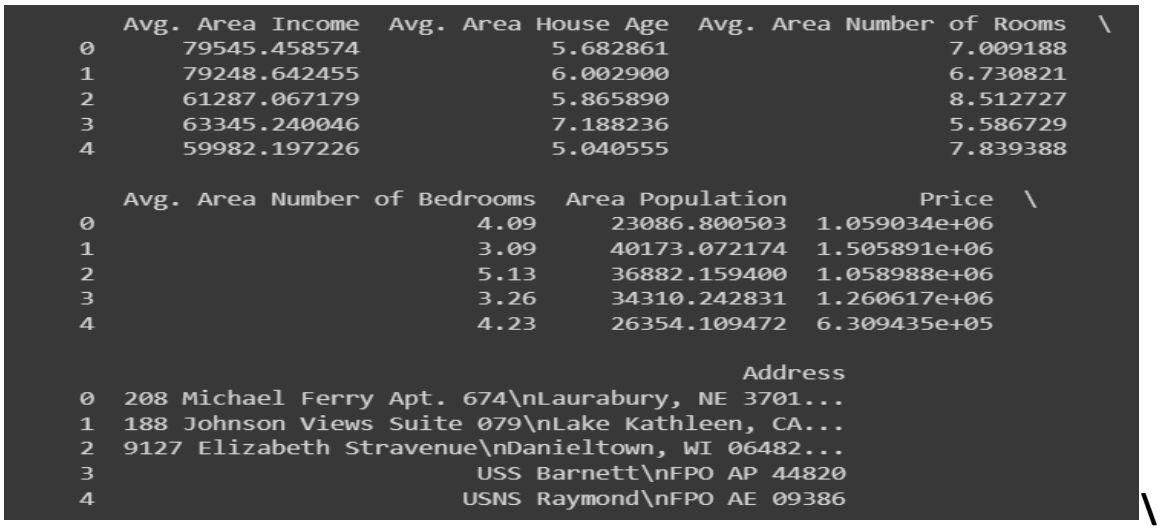
2. Load the Dataset:

We use the pd.read_csv() function to load the dataset from a CSV file. The loaded data is stored in a Pandas DataFrame, which is a two-dimensional, size-mutable, and heterogeneous tabular data structure.

3. Display the Data:

After loading the dataset, checking the first few rows to inspect the data. We do this by using the head() method, which displays the first few rows of the DataFrame.

OUTPUT:



```
   Avg. Area Income  Avg. Area House Age  Avg. Area Number of Rooms  \
0      79545.458574             5.682861                   7.009188
1      79248.642455             6.002900                   6.730821
2      61287.067179             5.865890                   8.512727
3      63345.240046             7.188236                   5.586729
4      59982.197226             5.040555                   7.839388

   Avg. Area Number of Bedrooms  Area Population         Price  \
0                          4.09     23086.800503  1.059034e+06
1                          3.09     40173.072174  1.505891e+06
2                          5.13     36882.159400  1.058988e+06
3                          3.26     34310.242831  1.260617e+06
4                          4.23     26354.109472  6.309435e+05

                                        Address
0  208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1  188 Johnson Views Suite 079\nLake Kathleen, CA...
2  9127 Elizabeth Stravenue\nDanieltown, WI 06482...
3                    USS Barnett\nFPO AP 44820
4                   USNS Raymond\nFPO AE 09386
```

\

PREPROCESSING THE DATASET:

1.importing libraries:

CODE:

```
from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.impute import SimpleImputer

from sklearn.preprocessing import OneHotEncoder
```

EXPLANATION: In this part, we import additional libraries required for data preprocessing and splitting the dataset. We import train_test_split from sklearn.model_selection to split the dataset and StandardScaler from sklearn.preprocessing to standardize the feature values.

2.handling missing values:

CODE:

```
imputer = SimpleImputer(strategy='mean')

data[['Avg. Area Number of Bedrooms']] = imputer.fit_transform(data[['Avg. Area Number of Bedrooms']])
```

EXPLANATION: We use the SimpleImputer from scikit-learn to fill missing values in the 'Avg. Area Number of Bedrooms' column with the mean of that column. This helps handle missing data in a numerical feature.

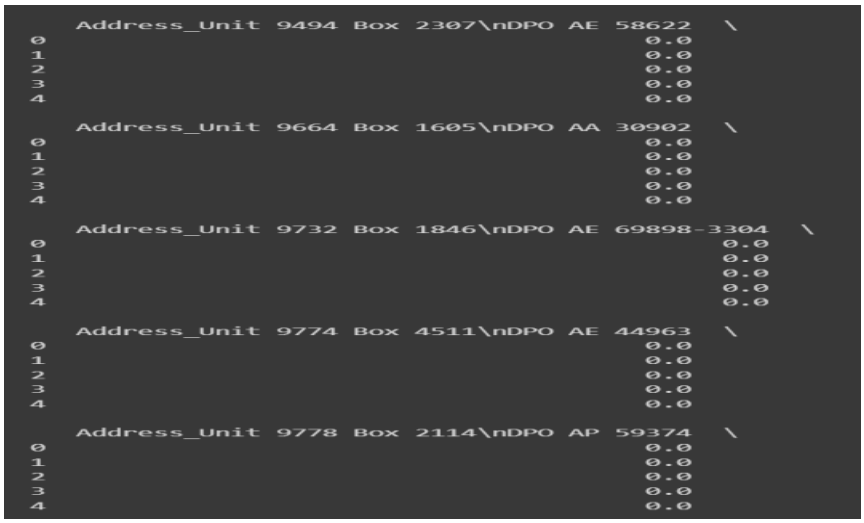
3.encoding categorical variables:

CODE:

```
encoder = OneHotEncoder(sparse=False, drop='first')

encoded_address = encoder.fit_transform(data[['Address']])

encoded_df = pd.DataFrame(encoded_address,
columns=encoder.get_feature_names_out(['Address']))

data = pd.concat([data, encoded_df], axis=1)

data.drop(['Address'], axis=1, inplace=True)

print(data.head())
```

EXPLANATION: If 'Address' is assumed to be a categorical variable, we use the OneHotEncoder to convert it into numerical values. It creates binary columns for each category, and we concatenate these columns to the original dataset while dropping the first (avoiding multicollinearity). The original 'Address' column is then dropped from the dataset.

OUTPUT:



```
   Address_Unit 9494 Box 2307\nDPO AE 58622  \
0                                         0.0
1                                         0.0
2                                         0.0
3                                         0.0
4                                         0.0

   Address_Unit 9664 Box 1605\nDPO AA 30902  \
0                                         0.0
1                                         0.0
2                                         0.0
3                                         0.0
4                                         0.0

   Address_Unit 9732 Box 1846\nDPO AE 69898-3304  \
0                                              0.0
1                                              0.0
2                                              0.0
3                                              0.0
4                                              0.0

   Address_Unit 9774 Box 4511\nDPO AE 44963  \
0                                         0.0
1                                         0.0
2                                         0.0
3                                         0.0
4                                         0.0

   Address_Unit 9778 Box 2114\nDPO AP 59374  \
0                                         0.0
1                                         0.0
2                                         0.0
3                                         0.0
4                                         0.0
```

4. Separate Features and Target Variable:

CODE:

```
X = data[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms', 'Avg. Area Number of Bedrooms', 'Area Population']]

y = data[['Price']]
```

EXPLANATION: We split the DataFrame into features (X) and the target variable (y). In this case, features include 'Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms', 'Avg. Area Number of Bedrooms', and 'Area Population'. The target variable is 'Price'.

5. Split the Dataset:

CODE:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

EXPLANATION: We use the train_test_split() function to split the data into training and testing sets. The test_size parameter specifies the proportion of the data to be used for testing (in this case, 20%), and random_state is set for reproducibility

6.standardize feature value:

CODE:

```
scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)
```

EXPLANATION: Standardization is an important preprocessing step. We create a StandardScaler object and use it to standardize the feature values. Standardization scales the data to have a mean of 0 and a standard deviation of 1. This can help improve the performance of many machine learning algorithms

DATA ANALYSIS:

1.Basic Statistics:

CODE:

```
print(data.describe())
```

EXPLANATION: This part of the code uses data.describe() to generate basic statistics for numerical columns in the dataset, including count, mean, standard deviation, minimum, maximum, and quartiles. It provides a summary of the central tendency and distribution of the data.

OUTPUT:

```
std                                       0.014142
min                                       0.000000
25%                                       0.000000
50%                                       0.000000
75%                                       0.000000
max                                       1.000000

       Address_Unit 9774 Box 4511\nDPO AE 44963  \
count                                5000.000000
mean                                    0.000200
std                                     0.014142
min                                     0.000000
25%                                     0.000000
50%                                     0.000000
75%                                     0.000000
max                                     1.000000

       Address_Unit 9778 Box 2114\nDPO AP 59374  \
count                                5000.000000
mean                                    0.000200
std                                     0.014142
min                                     0.000000
25%                                     0.000000
50%                                     0.000000
75%                                     0.000000
max                                     1.000000

       Address_Unit 9785 Box 0790\nDPO AP 60371-0797  \
count                                     5000.000000
mean                                         0.000200
std                                          0.014142
min                                          0.000000
25%                                          0.000000
50%                                          0.000000
75%                                          0.000000
max                                          1.000000
```

2. Correlation Matrix and Heatmap:

CODE:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
data = pd.read_csv('/content/dataset.csv.csv')


# Randomly sample a subset of the data
sampled_data = data.sample(frac=0.1, random_state=42)  # You can adjust the fraction as needed

# Calculate the correlation matrix for the sampled data
corr_matrix = sampled_data.corr()

# Create a heatmap for the correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix Heatmap (Sampled Data)')
plt.show()
```

EXPLANATION:

1.Randomly Sample a Subset of the Data:

sampled_data = data.sample(frac=0.1, random_state=42):

This line selects a random subset of the dataset using the sample method. The frac parameter specifies the fraction of data to sample, and in this case, 10% of the data is sampled. The random_state parameter ensures that the sampling is reproducible by setting a random seed (42).

2.Calculate the Correlation Matrix :

corr_matrix = sampled_data.corr():

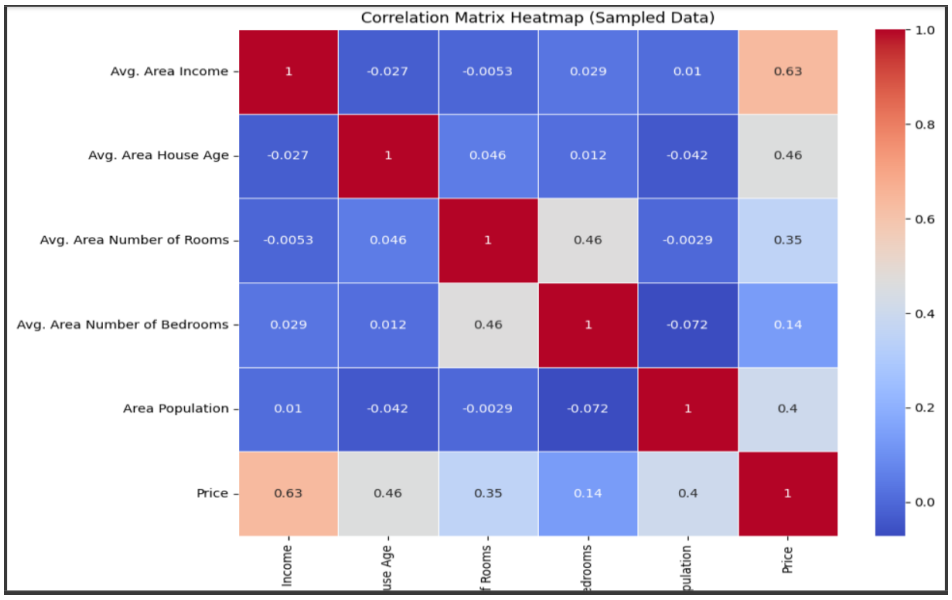
This line calculates the correlation matrix for the sampled data. The correlation matrix shows how each numerical feature in the dataset is correlated with every other feature. It provides insights into relationships between variables.

3.Create a Heatmap for the Correlation Matrix:

plt.figure(figsize=(10, 8)):

This line creates a Matplotlib figure with a specified figure size to control the dimensions of the heatmap plot.sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5): This line generates a heatmap using Seaborn (sns.heatmap). It visualizes the correlation matrix with color-coded cells. The annot=True parameter displays the correlation values within each cell, and the cmap parameter specifies the color map (coolwarm). The linewidths parameter controls the width of lines separating the cells.plt.title('Correlation Matrix Heatmap (Sampled Data)'): This line adds a title to the heatmap plot.plt.show(): This line displays the heatmap plot to visualize the correlations between features in the sampled data

OUTPUT:



3. Pairplot:

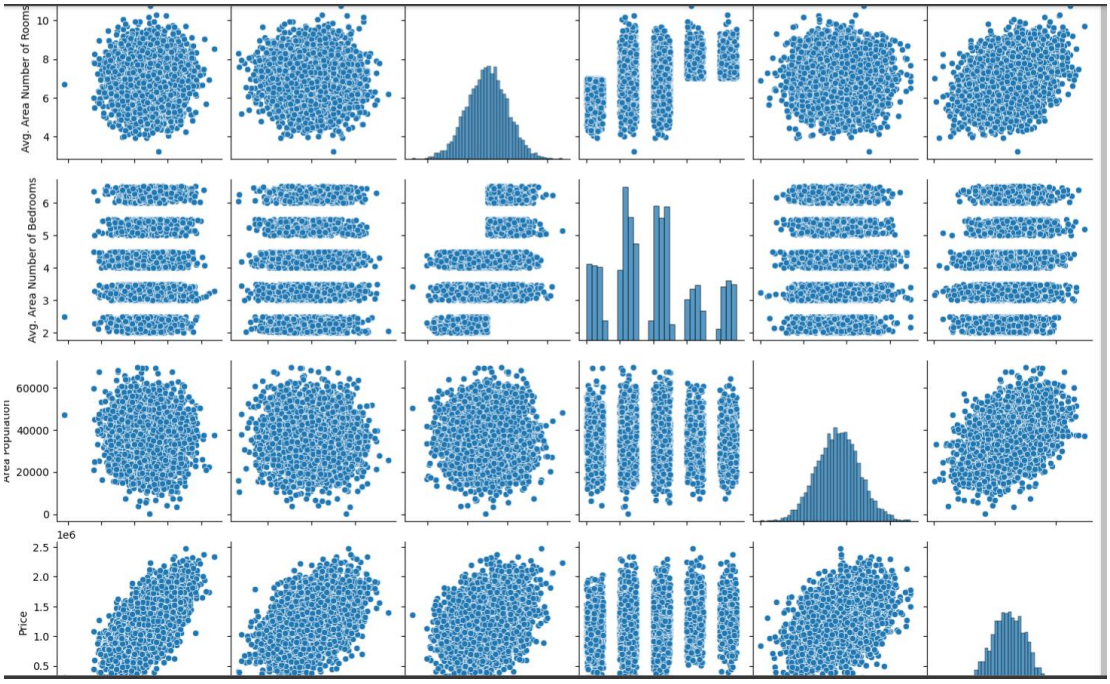
CODE:

sns.pairplot(data)

EXPLANATION: This part creates a grid of scatterplots for all pairs of numerical features in the dataset. Each scatterplot shows the relationship between two

features, which can help identify potential patterns, trends, or outliers in the data

OUTPUT:



DATA VISUALIZATION:

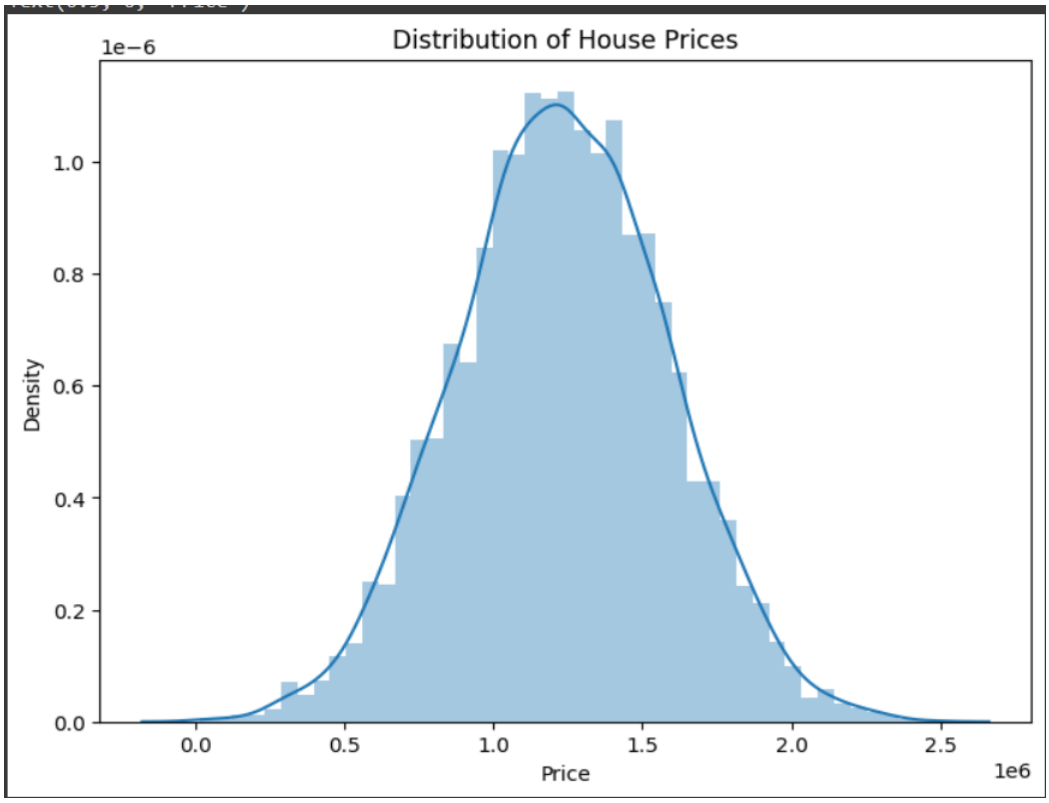
1. Distribution of the Target Variable:

CODE:

```
plt.figure(figsize=(8, 6))

sns.distplot(data['Price'])

plt.title('Distribution of House Prices')

plt.xlabel('Price')
```

EXPLANATION: This section uses sns.distplot() to create a histogram of the target variable ('Price'). It visualizes the distribution of house prices, helping you understand the spread and central tendency of this variable.

OUTPUT:



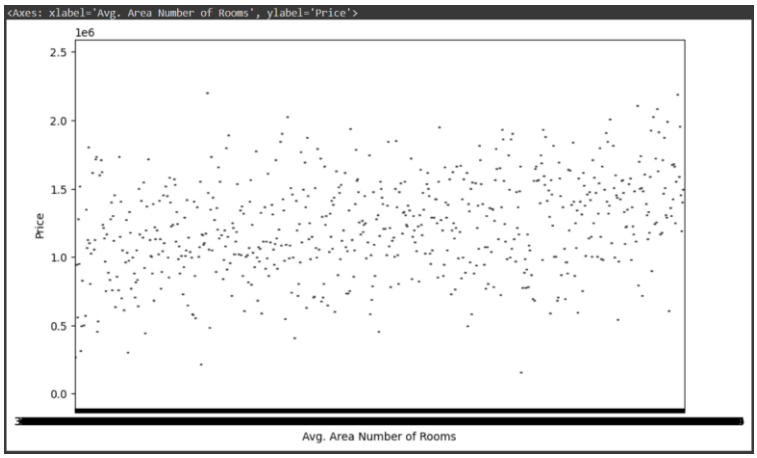
2. Box Plot:

CODE:

```
plt.figure(figsize=(10, 6))

sns.boxplot(x='Avg. Area Number of Rooms', y='Price', data=data)
```

EXPLANATION: Here, a box plot is generated using sns.boxplot() to visualize the relationship between 'Avg. Area Number of Rooms' and 'Price'. Box plots show the distribution of a numeric variable for different categories, allowing you to observe variations in house prices with respect to the number of rooms.different categories, allowing you to observe variations in house prices with respect to the number of rooms.

OUTPUT:



3. Scatter Plot:

CODE:

```
plt.figure(figsize=(8, 6))

sns.scatterplot(x='Avg. Area Income', y='Price', data=data)
```

EXPLANATION: This part creates a scatter plot using sns.scatterplot() to depict the relationship between 'Avg. Area Income' and 'Price'. Scatter plots can help you visualize the correlation or lack thereof between two variables.

OUTPUT: