

# HOUSE PRICE PREDICTION ANALYSIS

## USING MACHINE LEARNING

### PHASE 3: DEVELOPMENT PART 1

Creating a house price prediction model for a dataset of USA housing prices is a valuable task for both real estate professionals and individuals looking to buy or sell homes. Such a model can provide insights into the factors that influence home prices and assist in making informed decisions. Here's an introduction to building a house price prediction model for a USA housing dataset:

**Project Title:** Predicting House Prices: A Machine Learning Approach

#### INTRODUCTION:

The housing market is an important and complex sector that impacts people's lives in many ways. For many individuals and families, buying a house is one of the biggest investments they will make in their lifetime. Therefore, it is essential to accurately predict the prices of houses so that buyers and sellers can make informed decisions. In this project, we aim to develop a robust machine learning model that predicts house prices in the USA based on various attributes.

In this phase we are going to see how to load and preprocess the given dataset.

#### PROGRAM:

##### Loading the dataset:

Loading data is a crucial step in any data analysis or machine learning task. It involves bringing external datasets into your programming environment so that you can manipulate, analyze, and draw insights from the data.

##### ❖ Importing Libraries:

- **Pandas:** Pandas, the abbreviation for **Panel-Data**, is a library for representing data on a data-frame.
- **Numpy:** Numpy is a math library that supports many operations on arrays, from simple to complex.

- **Seaborn:** Seaborn is a python library for data visualization. It is built on top of matplotlib.
- **Matplotlib:** Matplotlib is a comprehensive library for visualization in Python. It supports most of the basic plots that we need when starting with data science.
- **Scikit learn:** It is a simple and very fast tool for predictive data analysis and statistically modeling.

```
[ ] import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```


#### ❖ Loading the dataset:

Use Pandas to load the dataset into a DataFrame.

```
[ ] dataset = pd.read_csv('USA_Housing.csv')
```

#### Data Exploration:

Check out the first few rows using dataset.head().

 dataset.head()

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.45857	5.682861	7.009188	4.09	23086.80050	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.64245	6.002900	6.730821	3.09	40173.07217	1.505891e+06	188 Johnson Views Suite 079\nLake Kathleen, CA...
2	61287.06718	5.865890	8.512727	5.13	36882.15940	1.058988e+06	9127 Elizabeth Stravenue\nDanielstown, WI 06482...
3	63345.24005	7.188236	5.586729	3.26	34310.24283	1.260617e+06	USS Barnett\nFPO AP 44820
4	59982.19723	5.040555	7.839388	4.23	26354.10947	6.309435e+05	USNS Raymond\nFPO AE 09386

#### Data Preprocessing techniques:

Data preprocessing is a crucial step in data analysis and machine learning. It involves cleaning, transforming, and organizing raw data into a format that can be effectively used for modeling and analysis. Here are some common data preprocessing techniques:

##### Data cleaning

#### ❖ Data cleaning:

- **Handling missing data:** You can remove or impute missing values using techniques like mean, median, mode imputation, or more advanced methods like regression imputation.

```
round((dataset.isnull().sum()/dataset.shape[0])*100,2)
```

```

Avg. Area Income      0.0
Avg. Area House Age   0.0
Avg. Area Number of Rooms  0.0
Avg. Area Number of Bedrooms  0.0
Area Population        0.0
Price                 0.0
Address               0.0
dtype: float64

```

For our dataset, since there is no missing values ,no need to use the data handling techniques like mean, median, mode.

- **Outlier detection and treatment:** To identify the outliers, we use techniques like describe(), info(), duplicated() and to handle the outliers, we use techniques like percentiles, z-score using statistical methods or domain knowledge.

To find if there is any duplicates in the dataset, we use duplicated().

```
dataset.duplicated()
```

```

0      False
1      False
2      False
3      False
4      False
...
4995   False
4996   False
4997   False
4998   False
4999   False
Length: 5000, dtype: bool

```

To get an overview of data types and missing values, we use info().

```
dataset.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
 #   Column                                  Non-Null Count  Dtype
---  -
 0   Avg. Area Income                       5000 non-null   float64
 1   Avg. Area House Age                    5000 non-null   float64
 2   Avg. Area Number of Rooms               5000 non-null   float64
 3   Avg. Area Number of Bedrooms            5000 non-null   float64
 4   Area Population                         5000 non-null   float64
 5   Price                                  5000 non-null   float64
 6   Address                                5000 non-null   object
dtypes: float64(6), object(1)
memory usage: 273.6+ KB

```

We use describe(), to get statistical summaries.

```
dataset.describe()
```

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5.000000e+03
mean	68583.108984	5.977222	6.987792	3.981330	36163.516039	1.232073e+06
std	10657.991214	0.991456	1.005833	1.234137	9925.650114	3.531176e+05
min	17796.631190	2.644304	3.236194	2.000000	172.610686	1.593866e+04
25%	61480.562390	5.322283	6.299250	3.140000	29403.928700	9.975771e+05
50%	68804.286405	5.970429	7.002902	4.050000	36199.406690	1.232669e+06
75%	75783.338665	6.650808	7.665871	4.490000	42861.290770	1.471210e+06
max	107701.748400	9.519088	10.759588	6.500000	69621.713380	2.469066e+06

```
# Categorical columns
cat_col = [col for col in dataset.columns if dataset[col].dtype == 'object']
print('Categorical columns :',cat_col)
# Numerical columns
num_col = [col for col in dataset.columns if dataset[col].dtype != 'object']
print('Numerical columns :',num_col)
```

```
Categorical columns : ['Address']
Numerical columns : ['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms', 'Avg. Area Number of Bedrooms', 'Area Population', 'Price']
```

This nunique() function in pandas is used to count the number of unique values in a dataframe.

```
[ ] dataset[cat_col].nunique()

Address      5000
dtype: int64
```

Some statistical techniques to handle the missing values and the outliers.

- Mean
- Median
- Mode
- Standard deviation
- Variance
- Percentile

The **mean** is a measure of central tendency that represents the average value of a set of numbers. In the context of a dataset, it is often used to describe the central or typical value of a numerical variable.

```
[ ] def mean(df):
    return sum(dataset.Price)/len(dataset)
print(mean(dataset))
```

```
1232072.6541452995
```

The **median** is a measure of central tendency in a dataset, representing the middle value when the data is sorted in ascending or descending order. Unlike the mean (average), which is sensitive to extreme values, the median is resistant to outliers.

```
median_value = np.median(dataset['Price'])
print(median_value)
```

1232669.378

In the context of a dataset, the "**mode**" refers to the value that appears most frequently in a particular column or variable. It's a measure of central tendency, similar to mean (average) and median. It's particularly handy when dealing with categorical variables or when you want to identify the most common value in a dataset

```
import statistics
mode_result = statistics.mode(dataset['Price'])

print(f'Mode: {mode_result}')
```

Mode: 1059033.558

**Standard deviation** is a measure of the amount of variation or dispersion in a set of values. In the context of a dataset, it provides a quantifiable measure of how much individual data points differ from the mean (average) of the dataset.

```
std_deviation = np.std(dataset['Price'])
print(f"Standard Deviation: {std_deviation}")
```

Standard Deviation: 353082.3130552725

**Percentiles** are a statistical concept used to describe the relative standing of a particular value within a dataset.

### Median (50th percentile):

- The value below which 50% of the data falls.
- It divides the dataset into two equal halves.

### Quartiles:

- Q1 (25th percentile): The value below which 25% of the data falls.
- Q2 (50th percentile, median): The value below which 50% of the data falls.
- Q3 (75th percentile): The value below which 75% of the data falls.

### Percentile Ranks:

- The nth percentile is the value below which n% of the data falls.

- For example, the 90th percentile is the value below which 90% of the data falls.

```
percentiles = np.percentile(dataset['Price'], [25, 50, 75])
print(f"25th Percentile (Q1): {percentiles[0]}")
print(f"50th Percentile (Q2 or Median): {percentiles[1]}")
print(f"75th Percentile (Q3): {percentiles[2]}")
```

```
25th Percentile (Q1): 997577.135075
50th Percentile (Q2 or Median): 1232669.378
75th Percentile (Q3): 1471210.2045
```

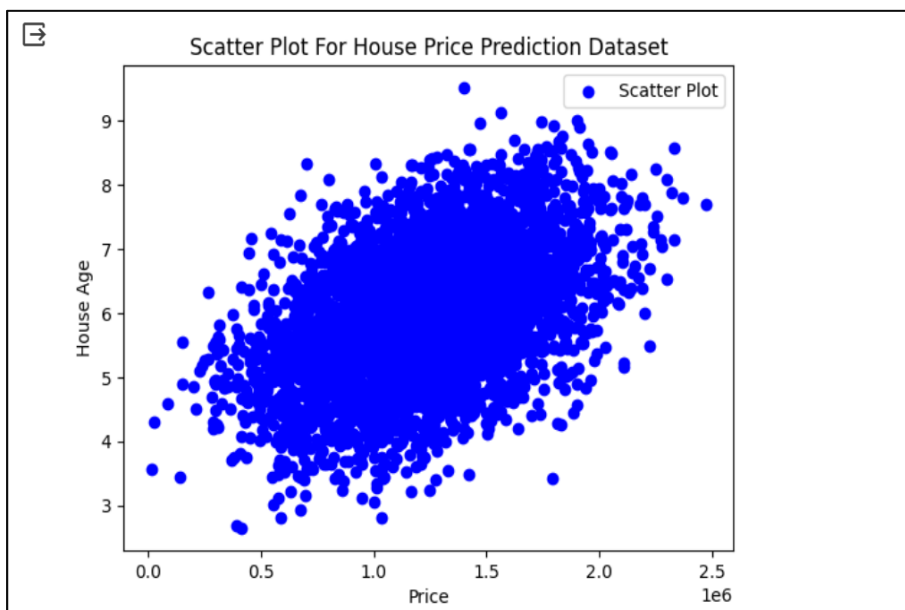
## Data Visualization Techniques:

Data visualization is a powerful tool in house price prediction analysis. It helps you understand the data, identify trends, outliers, and relationships between different variables. Here are some data visualization techniques commonly used in house price prediction analysis:

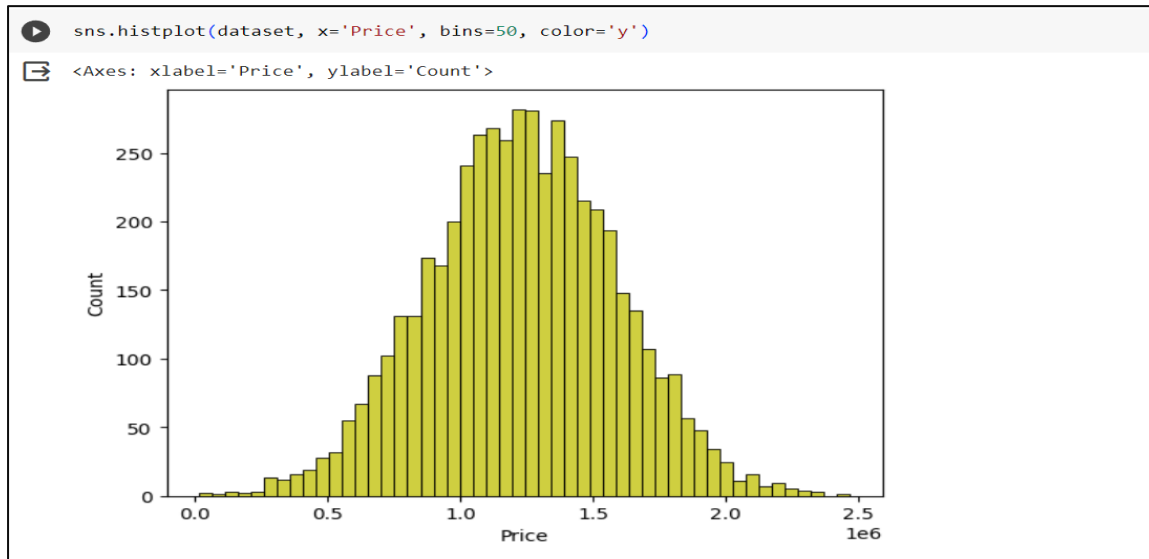
- **Scatter Plot:** Create scatter plots to explore the relationship between variables like square footage, number of bedrooms, or location and house prices. This can help identify patterns and outliers.

```
x_values = dataset['Price']
y_values = dataset['Avg. Area House Age']

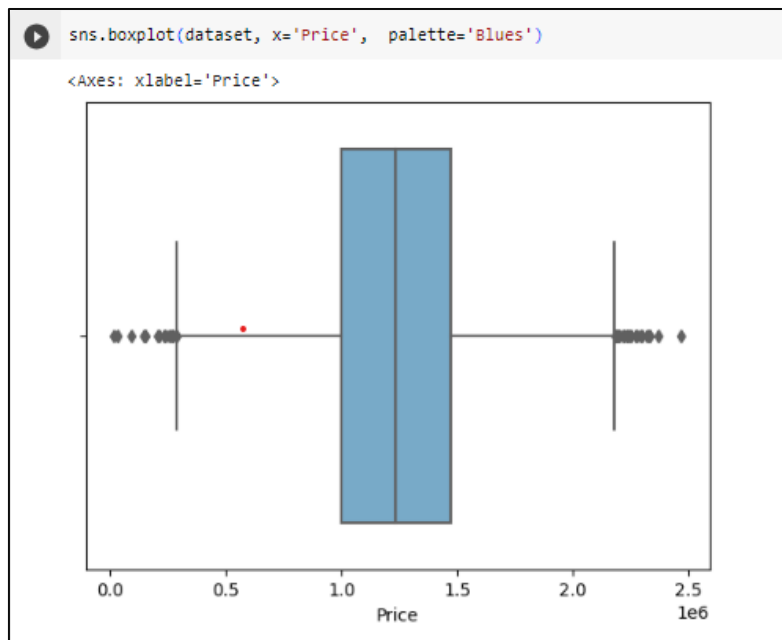
plt.scatter(x_values, y_values, c='blue', label='Scatter Plot')
plt.title('Scatter Plot For House Price Prediction Dataset')
plt.xlabel('Price')
plt.ylabel('House Age')
plt.legend()
plt.show()
```



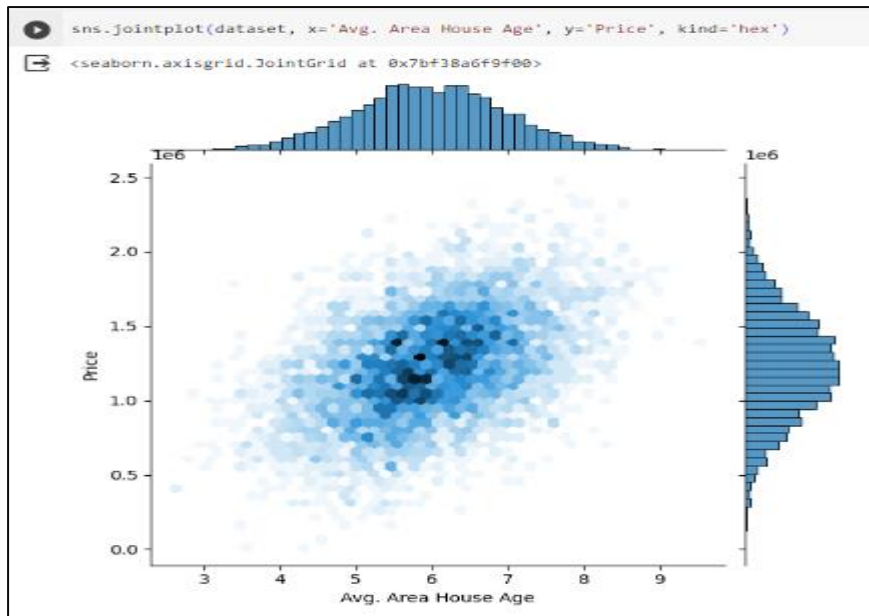
- **Histograms:** Use histograms to visualize the distribution of house prices. This helps you understand the central tendency and spread of the prices.



- **Box Plot:** Box plots are useful for visualizing the distribution of house prices by different categories, such as the number of bedrooms or the neighbourhood.



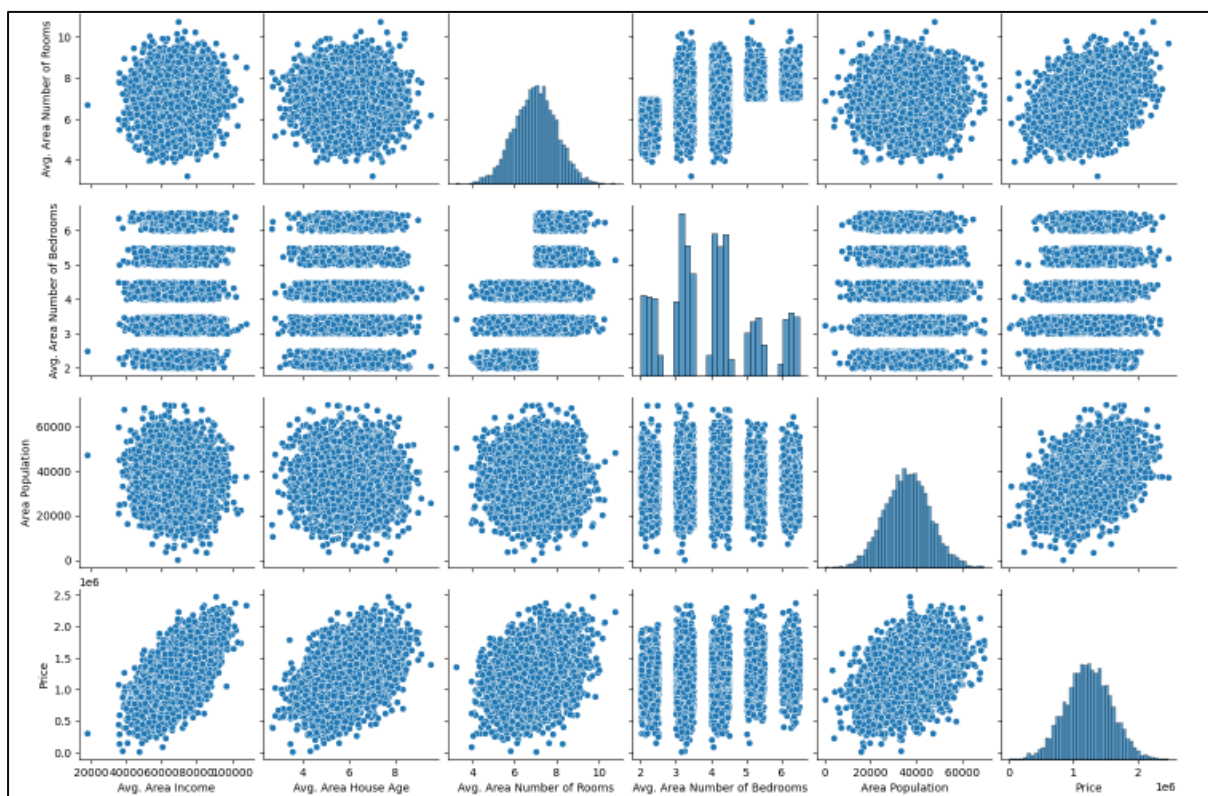
- **Joint plot:** It helps you to explore relationships between various features and the target variable (house prices).



- **Pairplot:** A pair plot displays scatter plots for multiple variables in your dataset. It's useful for identifying relationships and correlations among multiple features.

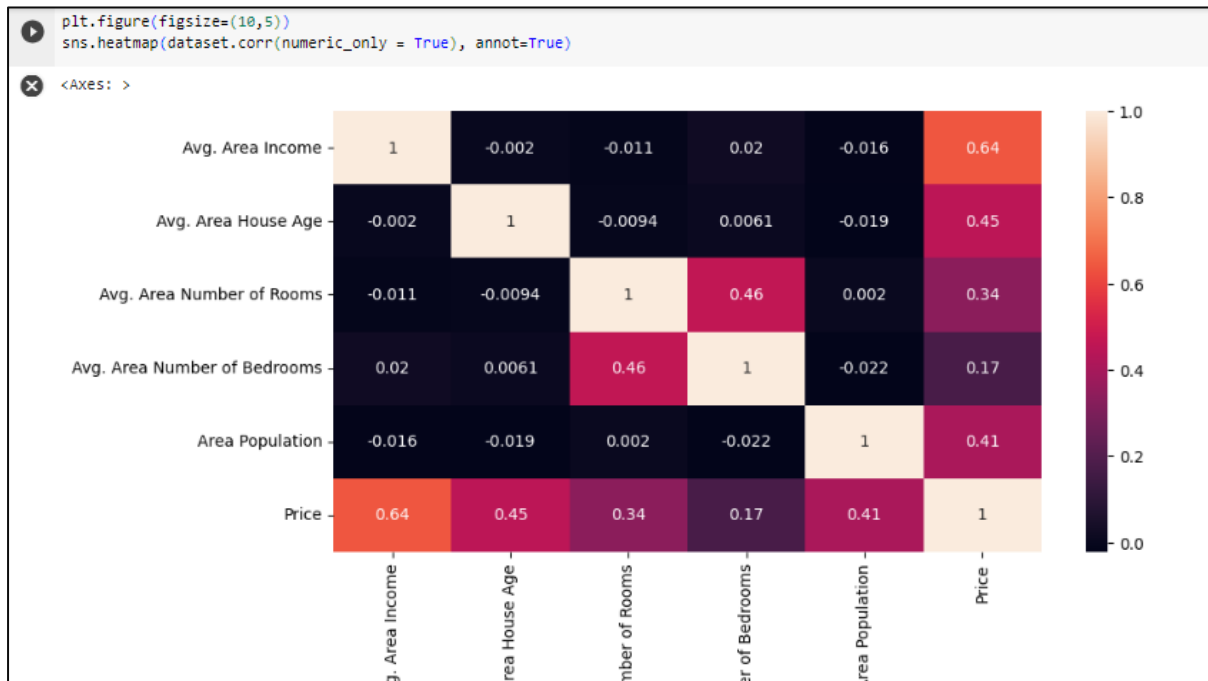
```
plt.figure(figsize=(12,8))
sns.pairplot(dataset)
```

<seaborn.axisgrid.PairGrid at 0x7b3882696c0>  
<Figure size 1280x800 with 0 Axes>





- **Heatmap:** Heatmaps help to visualize the relationships and patterns in the data, which can be useful for understanding the factors that influence house prices.



## Using LabelEncoder:

Label encoder is often used to convert the categorical values like address to the respective numerical values. These label encoder is available on the sklearn.preprocessing package.

```
from sklearn.preprocessing import LabelEncoder
labelencoder=LabelEncoder()
for column in dataset.columns:
    dataset[column] = labelencoder.fit_transform(dataset[column])
```

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
 #   Column              Non-Null Count  Dtype
---  -
 0   Avg. Area Income    5000 non-null   int64
 1   Avg. Area House Age 5000 non-null   int64
 2   Avg. Area Number of Rooms 5000 non-null   int64
 3   Avg. Area Number of Bedrooms 5000 non-null   int64
 4   Area Population      5000 non-null   int64
 5   Price                5000 non-null   int64
 6   Address              5000 non-null   int64
dtypes: int64(7)
memory usage: 273.6 KB
```

## **CONCLUSION:**

These steps provide a simplified overview of concepts like data cleaning, data preprocessing, data analysis for the house price prediction. The specific implementation details and choice of algorithms may vary depending on the dataset and the goals of the project.