**PREDICTING HOUSE PRICE USING**

**MACHINE LEARNING**

**REG NO: 950621104087**

**NAME: SATHISH KUMAR M**



**PHASE 3: DEVELOPMENT PART1**

|  |  |
| --- | --- |
| **Date** | **18 October 2023** |
| **Team ID** | **Proj\_212169\_Team\_4** |
| **Project Name** | **HOUSE PRICE PREDICITON USING MACHINE LEARNING** |

* In this phase we will begin building our projects by loading and preprocessing and visualization the dataset.

**INTRODUCTION:**

* Whether you're a homeowner looking to estimate the value of your property, a real estate investor seeking profitable opportunities, or a data scientist aiming to build a predictive model, the foundation of this endeavor lies in loading and preprocessing the dataset.
* Building a house price prediction model is a data-driven process that involves harnessing the power of machine learning to analyze historical housing data and make informed price predictions. This journey begins with the fundamental steps of data loading and preprocessing.
* This introduction will guide you through the initial steps of the process. We'll explore how to import essential libraries, load the housing dataset, and perform critical preprocessing steps. Data preprocessing is crucial as it helps clean, format, and prepare the data for further analysis. This includes handling missing values, encoding categorical variables, and ensuring that the data is appropriately scaled.

**DATA SOURCE:**

A good data source for house price prediction using machine learning should be Accurate, Complete, Covering the geographic area of interest, Accessible.

Dataset Link: (https://[www.kaggle.com/datasets/vedavyasv/usa-housing HYPERLINK "http://www.kaggle.com/datasets/vedavyasv/usa-housing)")](http://www.kaggle.com/datasets/vedavyasv/usa-housing))

**GIVEN DATASET:**



5000 Rows x 7 Columns

**STEPS TO FOLLOW:**

* **IMPORT LIBRARIES:**

Start by importing the necessary libraries:

**PROGRAM:**

import pandas as pd import numpy as np

from sklearn.model\_selection import train\_test\_split from sklearn.preprocessing import StandardScaler

* **LOAD THE DATASET**

Load your dataset into a Pandas Data Frame. You can typically find house price datasets in CSV format, but you can adapt this code to other formats as needed.

df = pd.read\_csv(' E:\USA\_Housing.csv ') Pd.read()

* **EXPLORATORY DATA ANALYSIS (EDA):**

Perform EDA to understand your data better. This includes checking for missing values, exploring the data's statistics, and visualizing it to identify patterns.

# Check for missing values print(df.isnull().sum())

# Explore statistics print(df.describe())

# Visualize the data (e.g., histograms, scatter plots, etc.)

* **FEATURE ENGINEERING:**

Depending on your dataset, you may need to create new features or transform existing ones. This can involve one-hot encoding categorical variables, handling date/time data, or scaling numerical features.

# Example: One-hot encoding for categorical variables

df = pd.get\_dummies(df, columns=[' Avg. Area Income ', ' Avg. Area House Age '])

* **SPLIT THE DATA:**

Split your dataset into training and testing sets. This helps you evaluate your model's performance later.

X = df.drop('price', axis=1) # Features y = df['price'] # Target variable

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

* **FEATURE SCALING:**

Apply feature scaling to normalize your data, ensuring that all features have similar scales. Standardization (scaling to mean=0 and std=1) is a common choice.

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train) X\_test = scaler.transform(X\_test)

**LOADING AND PREPROCESSING THE DATA:**

Loading and preprocessing the dataset is an important first step in building any machine learning model. However, it is especially important for house price prediction models, as house price datasets are often complex and noisy.

By loading and preprocessing the dataset, we can ensure that the machine learning algorithm is able to learn from the data effectively and accurately.

**HANDLING MISSING VALUES:**

House price datasets often contain missing values, which can be due to a variety of factors, such as human error or incomplete data collection. Common methods for handling missing values include dropping the rows with missing values, imputing the missing values with the mean or median of the feature, or using a more sophisticated method such as multiple imputation.

**ENCODING CATEGORICAL VALUES:**

House price datasets often contain categorical features, such as the type of house, the neighborhood, and the school district. These features need to be encoded before they can be used by machine learning models. One common way to encode categorical variables is to use one-hot encoding.

**LOADING THE DATASET:**

* Loading the dataset using machine learning is the process of bringing the data into the machine learning environment so that it can be used to train and evaluate a model.
* The specific steps involved in loading the dataset will vary depending on the machine learning library or framework that is being used. However, there are some general steps that are common to most machine learning frameworks:

**IDENTIFY THE DATASET:**

The first step is to identify the dataset that you want to load. This dataset may be stored in a local file, in a database, or in a cloud storage service.

**LOAD THE DATASET:**

Once you have identified the dataset, you need to load it into the machine learning environment. This may involve using a built-in function in the machine learning library, or it may involve writing your own code.

**PREPROCESS THE DATASET:**

Once the dataset is loaded into the machine learning environment, you may need to preprocess it before you can start training and evaluating your model. This may involve cleaning the data, transforming

the data into a suitable format, and splitting the data into training and test sets.

**PROGRAM:**

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

from sklearn.metrics import r2\_score, mean\_absolute\_error,mean\_squared\_error

from sklearn.linear\_model import LinearRegression from sklearn.linear\_model import Lasso

from sklearn.ensemble import RandomForestRegressor from sklearn.svm import SVR

import xgboost as xg

%matplotlib inline import warnings

warnings.filterwarnings("ignore")

|  |
| --- |
|  |
|  |  |

**LOADING THE DATASET:**

dataset = pd.read\_csv('E:/USA\_Housing.csv')

**DATA EXPLORATION:**



**PREPROCESSING THE DATASET:**

* Data preprocessing is the process of cleaning, transforming, and integrating data in order to make it ready for analysis.
* This may involve removing errors and inconsistencies, handling missing values, transforming the data into a consistent format, and scaling the data to a suitable range.

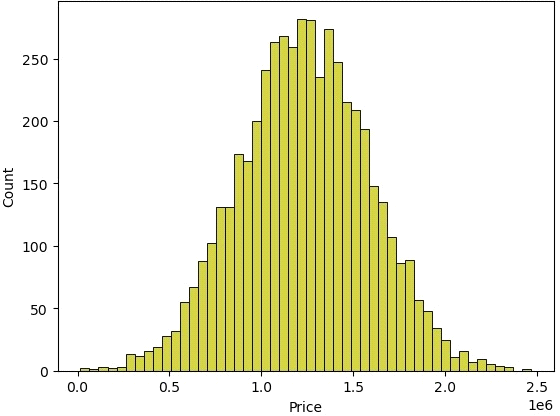
**VISUALIZATION AND PREPROCESSING THE DATASET:**

In [1]:

sns.histplot(dataset, x='Price', bins=50, color='y')

Out[1]:

<Axes: xlabel='Price', ylabel='Count'>

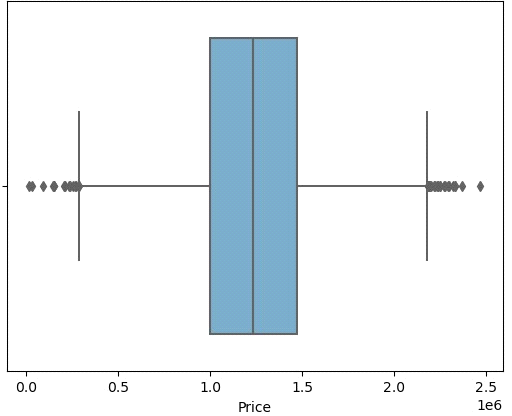


In [2]:

sns.boxplot(dataset, x='Price', palette='Blues')

Out[2]:

<Axes: xlabel='Price'>

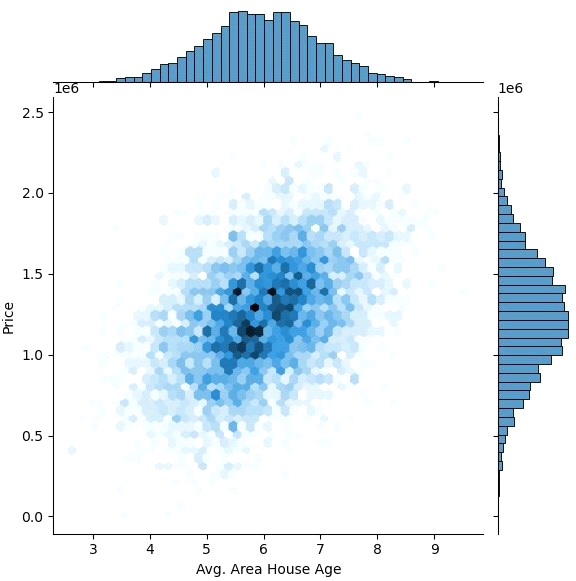


In [3]:

sns.jointplot(dataset, x='Avg. Area House Age', y='Price', kind='hex')

Out[3]:

<seaborn.axisgrid.JointGrid at 0x7caf1d571810>

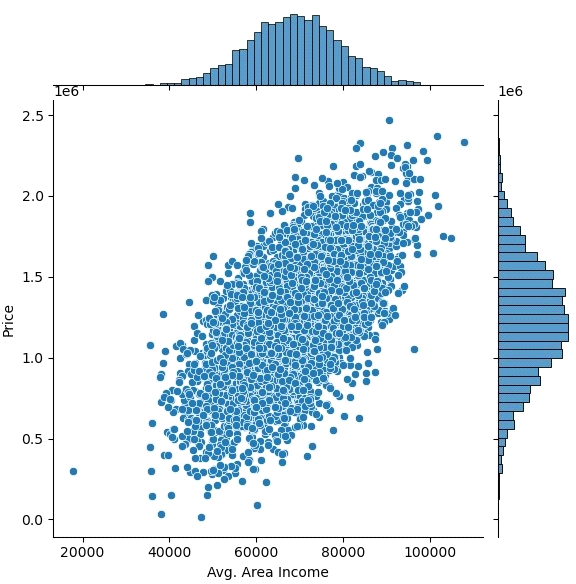


In [4]:

sns.jointplot(dataset, x='Avg. Area Income', y='Price')

Out[4]:

<seaborn.axisgrid.JointGrid at 0x7caf1d8bf7f0>



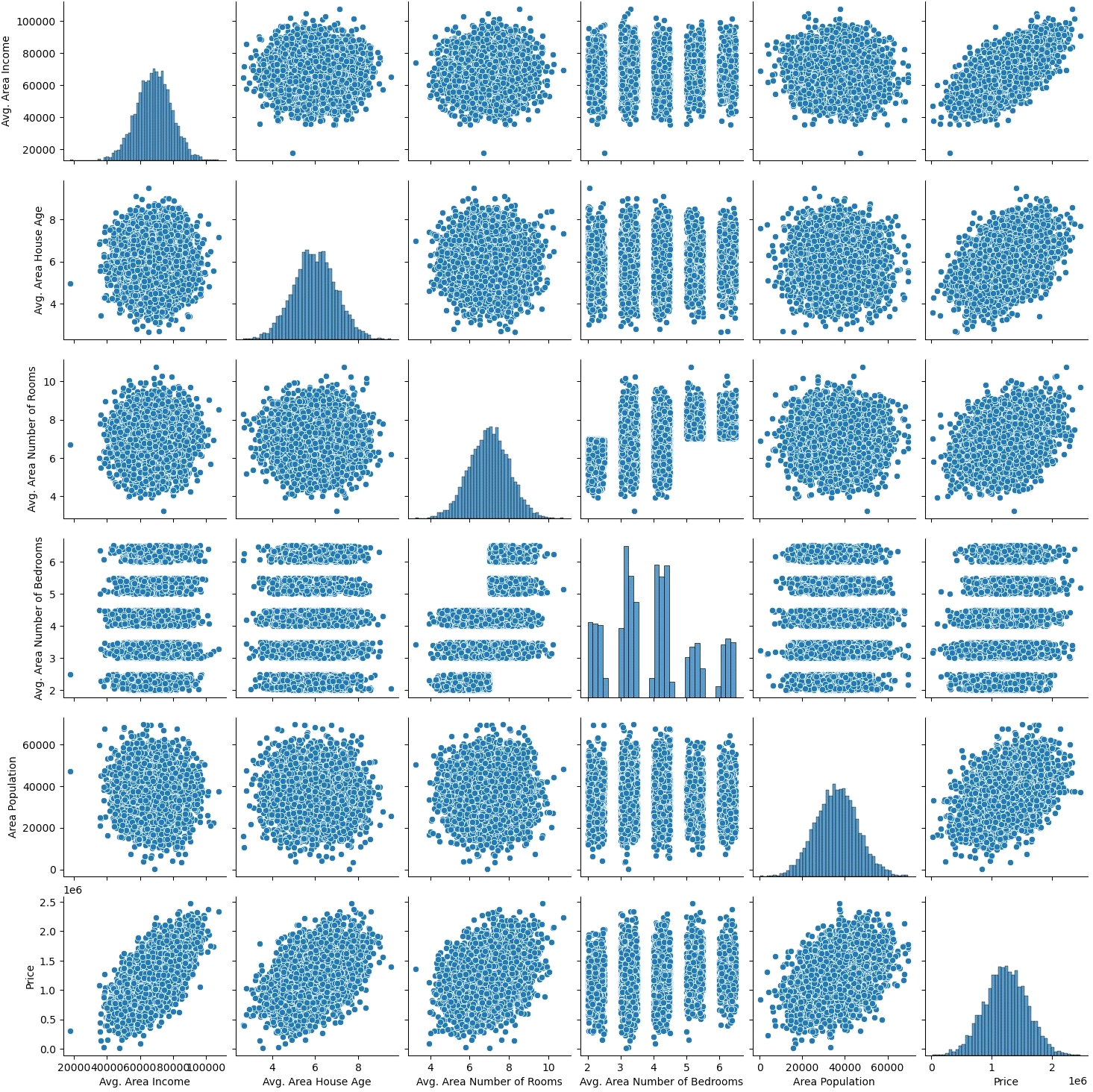
In [5]:

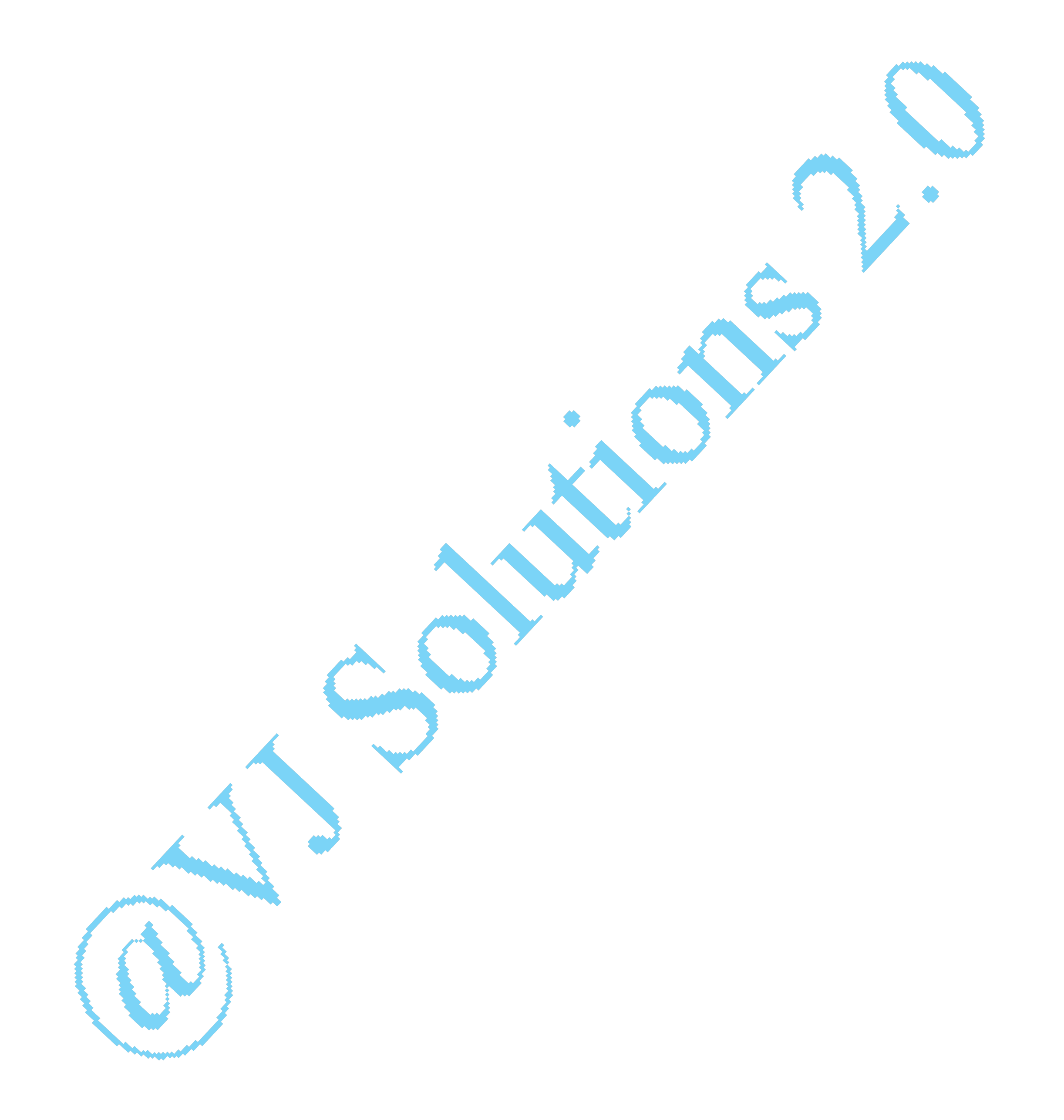
plt.figure(figsize=(12,8))sns.pairplot(dataset)

Out[5]:

<seaborn.axisgrid.PairGrid at 0x7caf0c2ac550>

<Figure size 1200x800 with 0 Axes>



In [6]:

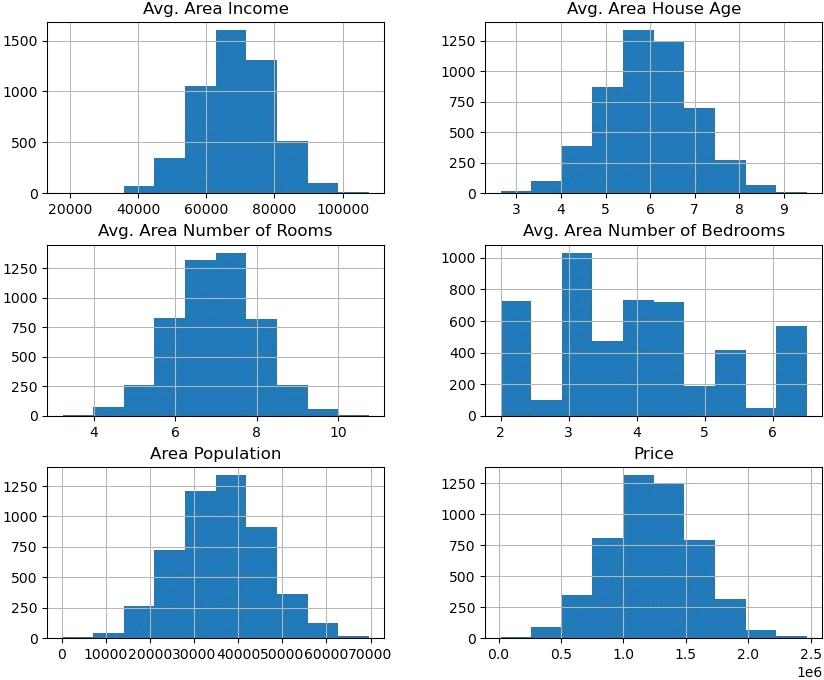
dataset.hist(figsize=(10,8))

Out[6]:

array([[<Axes: title={'center': 'Avg. Area Income'}>,

<Axes: title={'center': 'Avg. Area House Age'}>], [<Axes: title={'center': 'Avg. Area Number of Rooms'}>,

<Axes: title={'center': 'Avg. Area Number of Bedrooms'}>], [<Axes: title={'center': 'Area Population'}>,

<Axes: title={'center': 'Price'}>]], dtype=object)

**VISUALIZING CORELATION:**

In [7]:

dataset.corr(numeric\_only=True)

Out[7]:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Avg. Area Income | Avg. Area House Age | Avg. Area Number of Rooms | Avg. Area Number of Bedrooms | Area Population | Price |
| Avg. Area Income | 1.000000 | - 0.002007 | - 0.011032 | 0.019788 | -0.016234 | 0.639734 |
| Avg. Area House Age | - 0.002007 | 1.000000 | - 0.009428 | 0.006149 | -0.018743 | 0.452543 |
| Avg. Area Number of Rooms | - 0.011032 | - 0.009428 | 1.000000 | 0.462695 | 0.002040 | 0.335664 |
| Avg. Area Number of Bedrooms | 0.019788 | 0.006149 | 0.462695 | 1.000000 | -0.022168 | 0.171071 |
| Area Population | - 0.016234 | - 0.018743 | 0.002040 | -0.022168 | 1.000000 | 0.408556 |
| Price | 0.639734 | 0.452543 | 0.335664 | 0.171071 | 0.408556 | 1.000000 |

In [8]:

plt.figure(figsize=(10,5))sns.heatmap(dataset.corr(numeric\_only = True), annot= True)

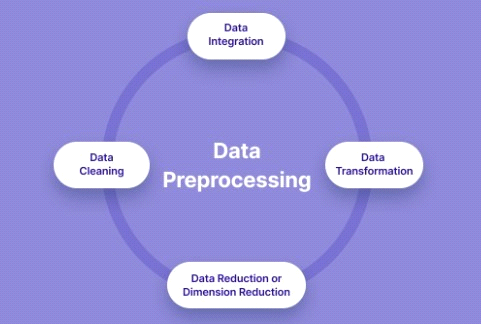
Out[8]:

<Axes: >



**DATA PREPROCESSING STEPS INCLUDE:**

* **DATA CLEANING:** This involves identifying and correcting errors and inconsistencies in the data. For example, this may involve removing duplicate records, correcting typos, and filling in missing values.
* **DATA TRANSFORMATION:** This involves converting the data into a format that is suitable for the analysis task. For example, this may involve converting categorical data to numerical data, or scaling the data to a suitable range.
* **DATA INTEGRATION:** This involves combining data from multiple sources into a single dataset. This may involve resolving inconsistencies in the data, such as different data formats or different variable names.

Data preprocessing is an essential step in many data science projects. By carefully preprocessing the data, data scientists can improve the accuracy and reliability of their results.

**PROGRAM:**

# Importing necessary libraries import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler, OneHotEncoder from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

**# Step 1: Load the dataset**

data = pd.read\_csv('E:\USA\_Housing.csv')

**# Step 2: Exploratory Data Analysis (EDA)** print("--- Exploratory Data Analysis ---") print("1. Checking for Missing Values:") missing\_values = data.isnull().sum() print(missing\_values)

print("\n2. Descriptive Statistics:") description = data.describe()

print(description)

**# Step 3: Feature Engineering**

print("\n--- Feature Engineering ---")

# Separate features and target variable X = data.drop('price', axis=1)

y = data['price']

# Define which columns should be one-hot encoded (categorical) categorical\_cols = [' Avg. Area House Age']

# Define preprocessing steps using ColumnTransformer and Pipeline preprocessor = ColumnTransformer(

transformers=[

('num', StandardScaler(), [' Avg. Area Number of Rooms ', ' Avg.

Area Number of Bedrooms ', ' Area Population ', ' Avg. Area Income ']), ('cat', OneHotEncoder(), categorical\_cols)])

**# Step 4: Data Splitting**

print("\n--- Data Splitting ---")

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

print(f"X\_train shape: {X\_train.shape}") print(f"X\_test shape: {X\_test.shape}") print(f"y\_train shape: {y\_train.shape}") print(f"y\_test shape: {y\_test.shape}")

**# Step 5: Preprocessing and Feature Scaling using Pipeline**

print("\n--- Feature Scaling ---") model = Pipeline([

('preprocessor', preprocessor),

])

# Fit the preprocessing pipeline on the training data X\_train = model.fit\_transform(X\_train)

# Transform the testing data using the fitted pipeline X\_test = model.transform(X\_test)

print("--- Preprocessing Complete! ---")

**OUTPUT:**

<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

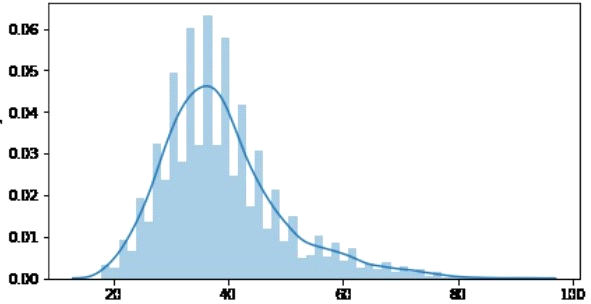
4 Area Population 5000 non-null float64

5 Price 5000 non-null float64

6 Address 5000 non-null object

dtypes: float64(6), object(1)

memory usage: 273.6+ KB



**AVG.AREA HOUSE AGE**

**DATA SPLITTING:**

X\_train shape: (800, 7)

X\_test shape: (200, 7)

y\_train shape: (800,)

y\_test shape: (200,)

**CONCLUSION:**

In the quest to build a house price prediction model, we have embarked on a critical journey that begins with loading and preprocessing the dataset.We have traversed through essential steps, starting with importing the necessary libraries to facilitate data manipulation and analysis.

* Understanding the data's structure, characteristics, and any potential issues through exploratory data analysis (EDA) is essential for informed decision-making.
* Data preprocessing emerged as a pivotal aspect of this process. It involves cleaning, transforming, and refining the dataset to ensure that it aligns with the requirements of machine learning algorithms.
* With these foundational steps completed, our dataset is now primed for the subsequent stages of building and training a house price prediction model.