PROJECT 9: PREDICTING HOUSE PRICE USING MACHINE LEARNING

PHASE 3: DEVELOPMENT PART

INTRODUCTION:

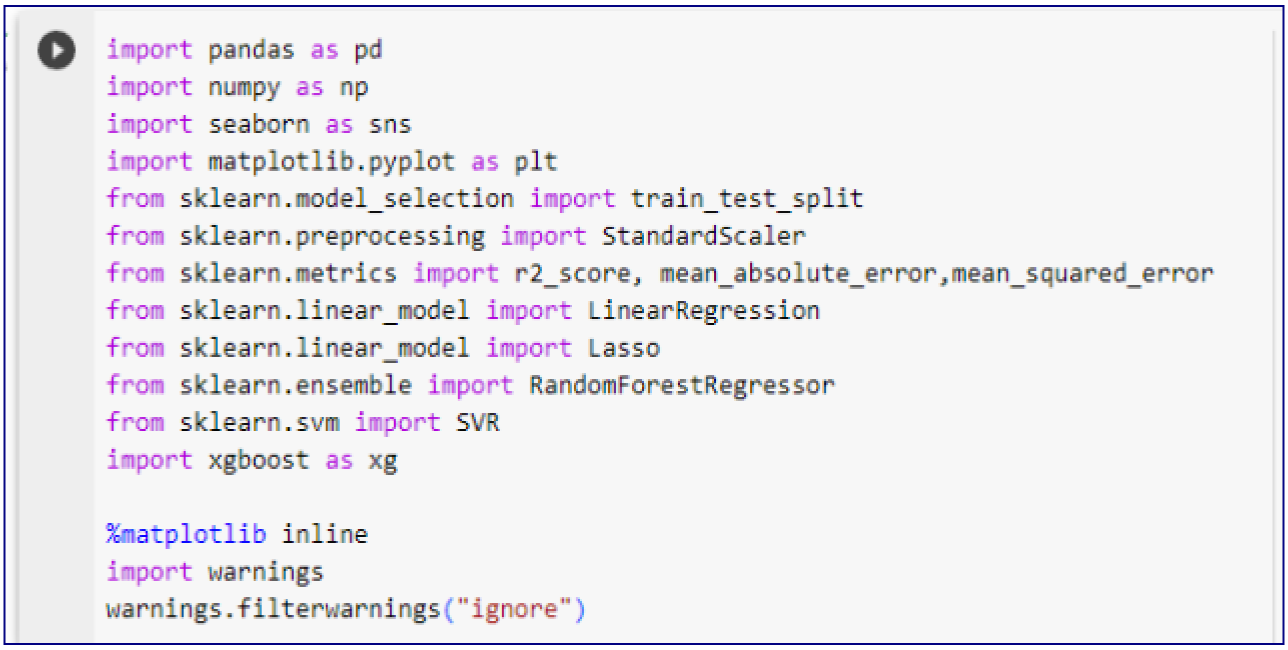
A house price prediction project in machine learning is a common and practical application of regression analysis. The goal is to develop a model that can predict the selling price of houses based on various features or attributes such as square footage, number of bedrooms, location, and more. Here are the general steps involved in such a project.

IMPORTING DATASET:

A house price dataset that suits your project's requirements,Once you've identified a suitable dataset, download it to your local machine. House price datasets are typically available in formats like CSV, Excel, or database files. Make sure you know where the dataset is saved on your computer.

IMPORTING LIBRARIES:

- Pandas is a powerful data manipulation and analysis library.
- NumPy is used for numerical and mathematical operationsPython,especially for working with arrays and matrices.
- Seaborn is a data visualization library built on top of Matplotlib. It provides a high-level interface for creating informative and attractive statistical graphics.
- Matplotlib is a widely used library for creating static, animated, and interactive visualizations in Python.
- The train_test_split function is used to split a dataset into training and testing subsets, which is crucial for model training and evaluation in machine learning.
- This preprocessing step is often necessary to make sure features are on the same scale, which can improve the performance of many machine learning algorithms.

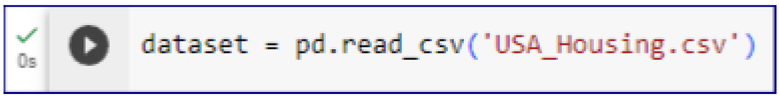


```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score, mean_absolute_error,mean_squared_error
from sklearn.linear_model 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")
```

LOAD DATASET:

- pd is an alias for the Pandas library, which you imported earlier.
- read_csv() is a Pandas function that is used to read CSV files and load their data into a DataFrame.

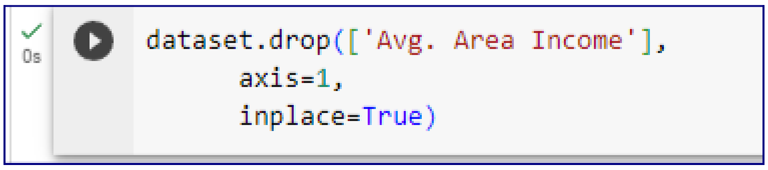


```
dataset = pd.read_csv('USA_Housing.csv')
```

DATA CLEANING:

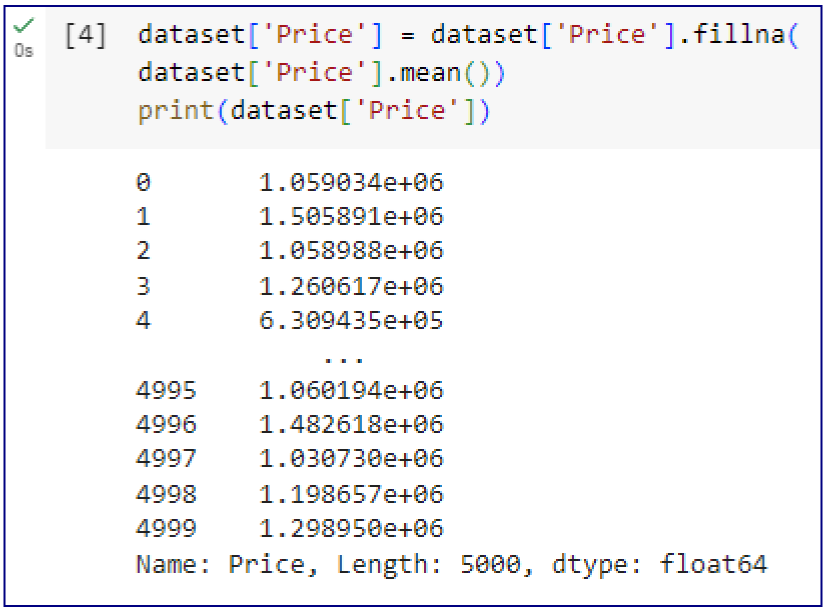
Data cleaning is a crucial step in the data preparation process. It involves identifying and correcting errors or inconsistencies in your data to ensure that it is accurate, reliable, and suitable for analysis.

- Data Collection
- Handling Missing Values
- Handling Duplicate Data
- Data Transformation
- Data Encoding
- Data Validation



```
dataset.drop(['Avg. Area Income'],
        axis=1,
        inplace=True)
```

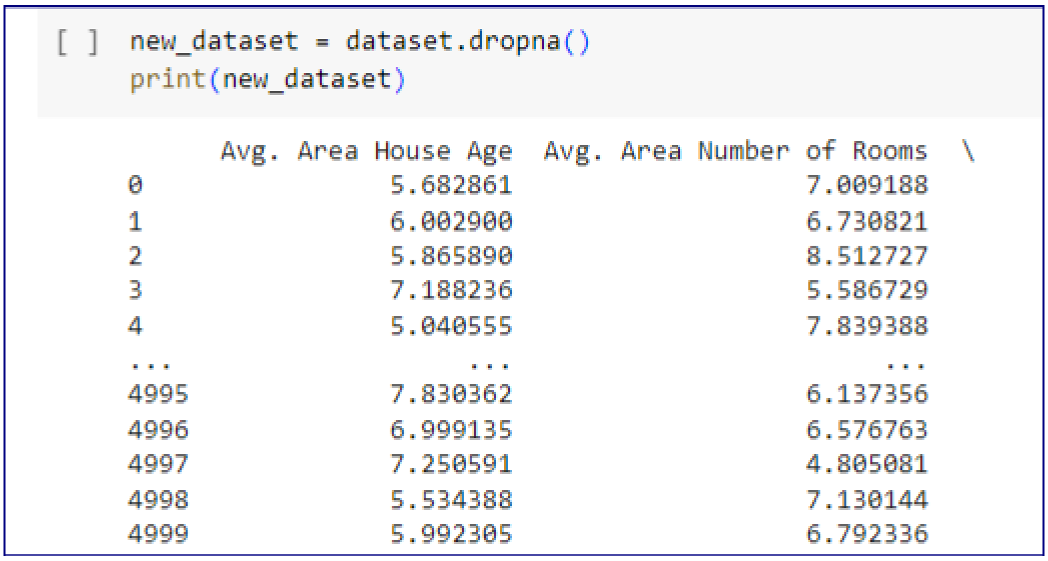
This line of code drops the 'Avg. Area Income' column from the DataFrame named 'dataset'.



```
[4]  dataset['Price'] = dataset['Price'].fillna(
     dataset['Price'].mean())
     print(dataset['Price'])
```

```
0       1.059034e+06
1       1.505891e+06
2       1.058988e+06
3       1.260617e+06
4       6.309435e+05
            ...
4995    1.060194e+06
4996    1.482618e+06
4997    1.030730e+06
4998    1.198657e+06
4999    1.298950e+06
Name: Price, Length: 5000, dtype: float64
```

This code fills missing values in the 'Price' column of the 'dataset' DataFrame with the mean value of the 'Price' column and then prints the updated 'Price' column.



```
[ ]  new_dataset = dataset.dropna()
     print(new_dataset)
```

```
      Avg. Area House Age  Avg. Area Number of Rooms  \
0                5.682861                   7.009188
1                6.002900                   6.730821
2                5.865890                   8.512727
3                7.188236                   5.586729
4                5.040555                   7.839388
...                   ...                        ...
4995             7.830362                   6.137356
4996             6.999135                   6.576763
4997             7.250591                   4.805081
4998             5.534388                   7.130144
4999             5.992305                   6.792336
```

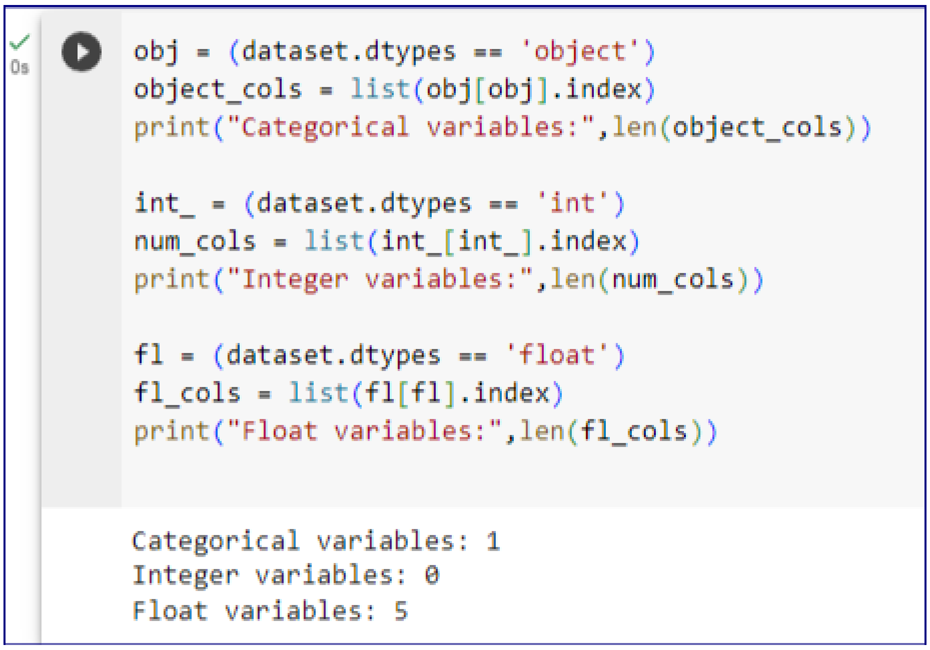
It creates a new DataFrame called 'new_dataset' by removing rows with missing values (NaN) from the original 'dataset' and then prints the 'new_dataset'.

DATA PREPROCESSING:

It is defined as Collection, manipulation, and processing of collected data for the required use. It is a task of converting data from a given form to a much more usable and desired form i.e. making it more meaningful and informative. Using Machine Learning algorithms, mathematical modelling and statistical knowledge, this entire process can be automated.

- Data Transformation
- Feature Selection
- Handling Outliers

- Normalization and Scaling
- Data Splitting
- Data Imputation



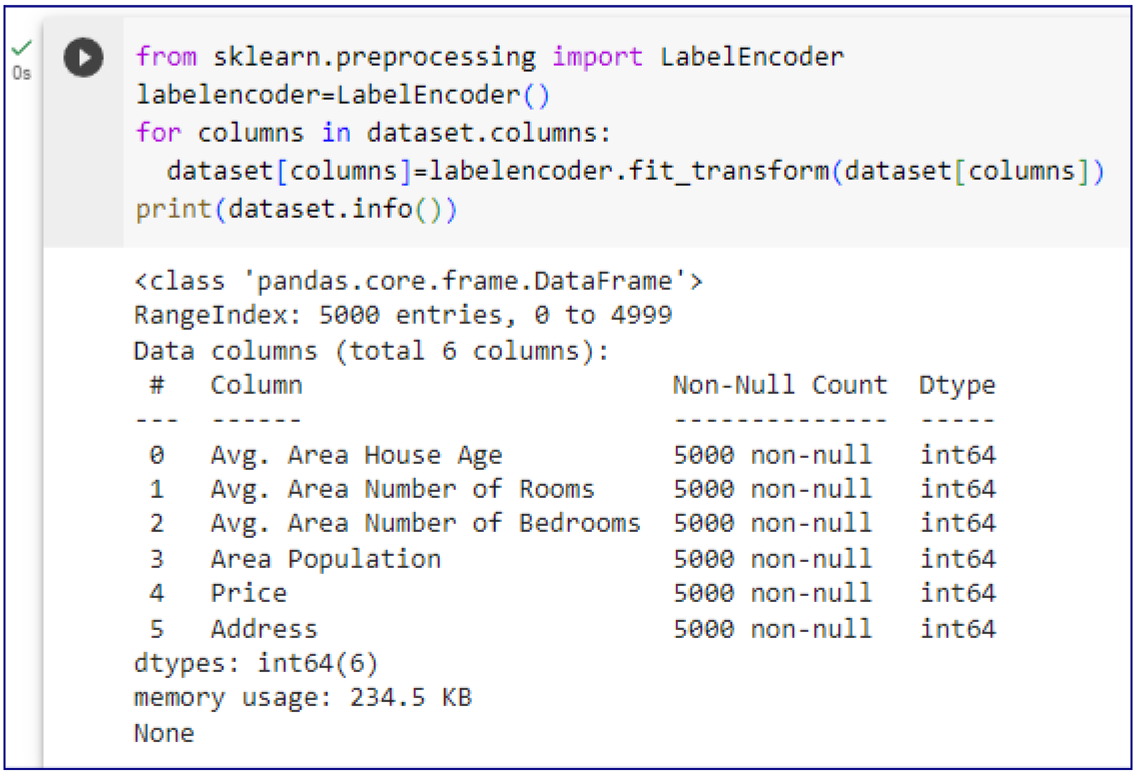
```
obj = (dataset.dtypes == 'object')
object_cols = list(obj[obj].index)
print("Categorical variables:",len(object_cols))

int_ = (dataset.dtypes == 'int')
num_cols = list(int_[int_].index)
print("Integer variables:",len(num_cols))

fl = (dataset.dtypes == 'float')
fl_cols = list(fl[fl].index)
print("Float variables:",len(fl_cols))
```

```
Categorical variables: 1
Integer variables: 0
Float variables: 5
```

This code is used to identify and categorize the columns in your dataset into three types: categorical, integer, and float variables.



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

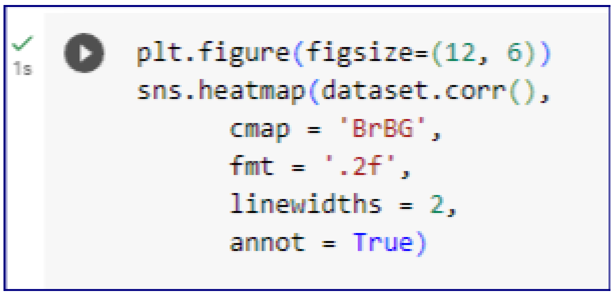
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 6 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   Avg. Area House Age           5000 non-null   int64
 1   Avg. Area Number of Rooms     5000 non-null   int64
 2   Avg. Area Number of Bedrooms  5000 non-null   int64
 3   Area Population               5000 non-null   int64
 4   Price                         5000 non-null   int64
 5   Address                       5000 non-null   int64
dtypes: int64(6)
memory usage: 234.5 KB
None
```

The code you've provided is using scikit-learn's LabelEncoder to transform the categorical (object) columns in the dataset into numerical labels. This is a common preprocessing step when working with machine learning models that require numerical input data.

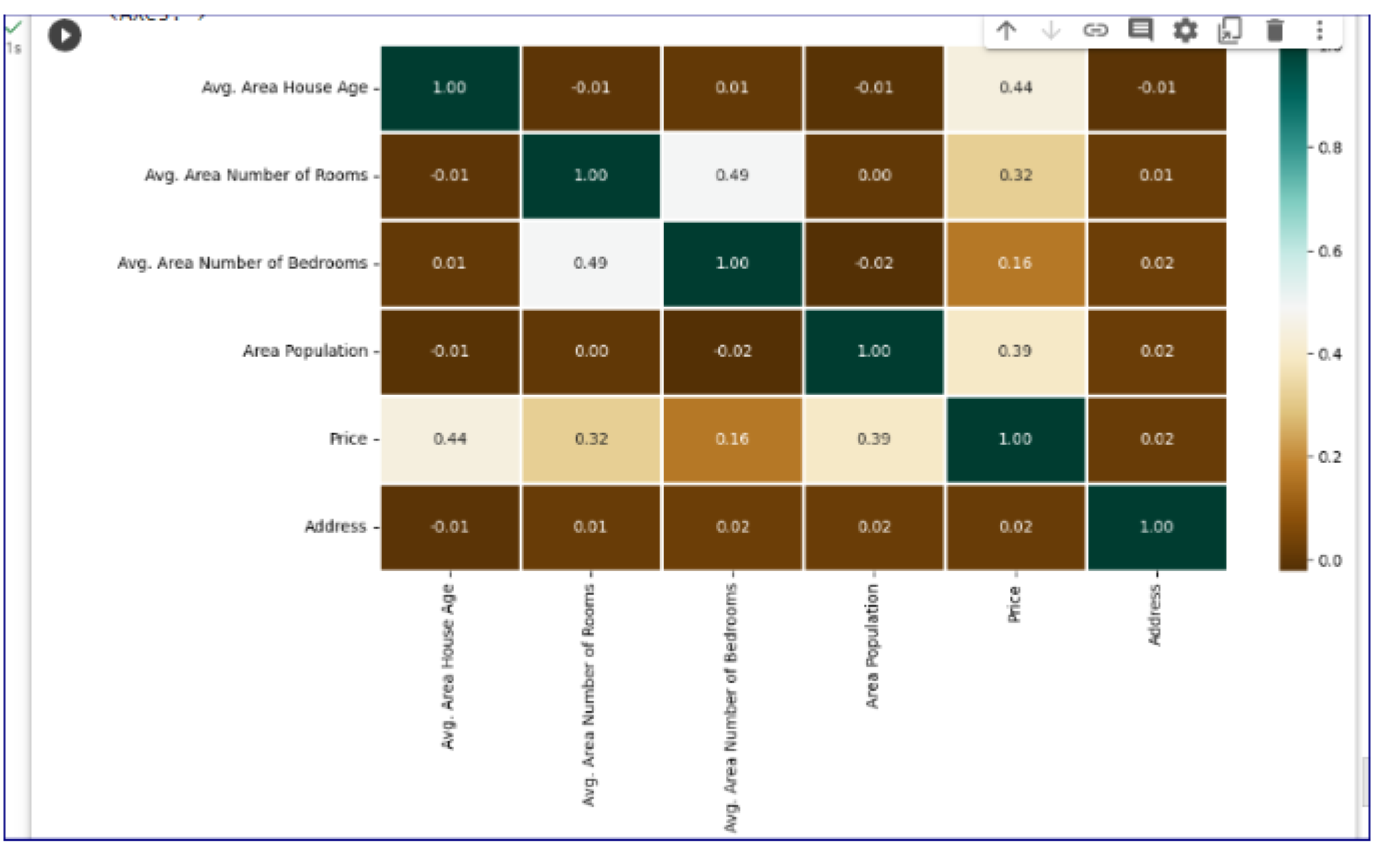
Data preprocessing plays a significant role in the quality and reliability of data-driven projects. It sets the stage for accurate analysis, meaningful insights, and the development of predictive models in various domains, from finance and healthcare to marketing and more. Proper data preprocessing can make the difference between a successful and unsuccessful data-driven project.

DATA ANALYSIS:

Data analysis is the process of inspecting, cleaning, transforming, and modeling data to discover useful information, draw meaningful conclusions, and support decision-making. It is a critical step in various fields, from business and science to healthcare and social sciences, where data-driven insights are essential for making informed choices.

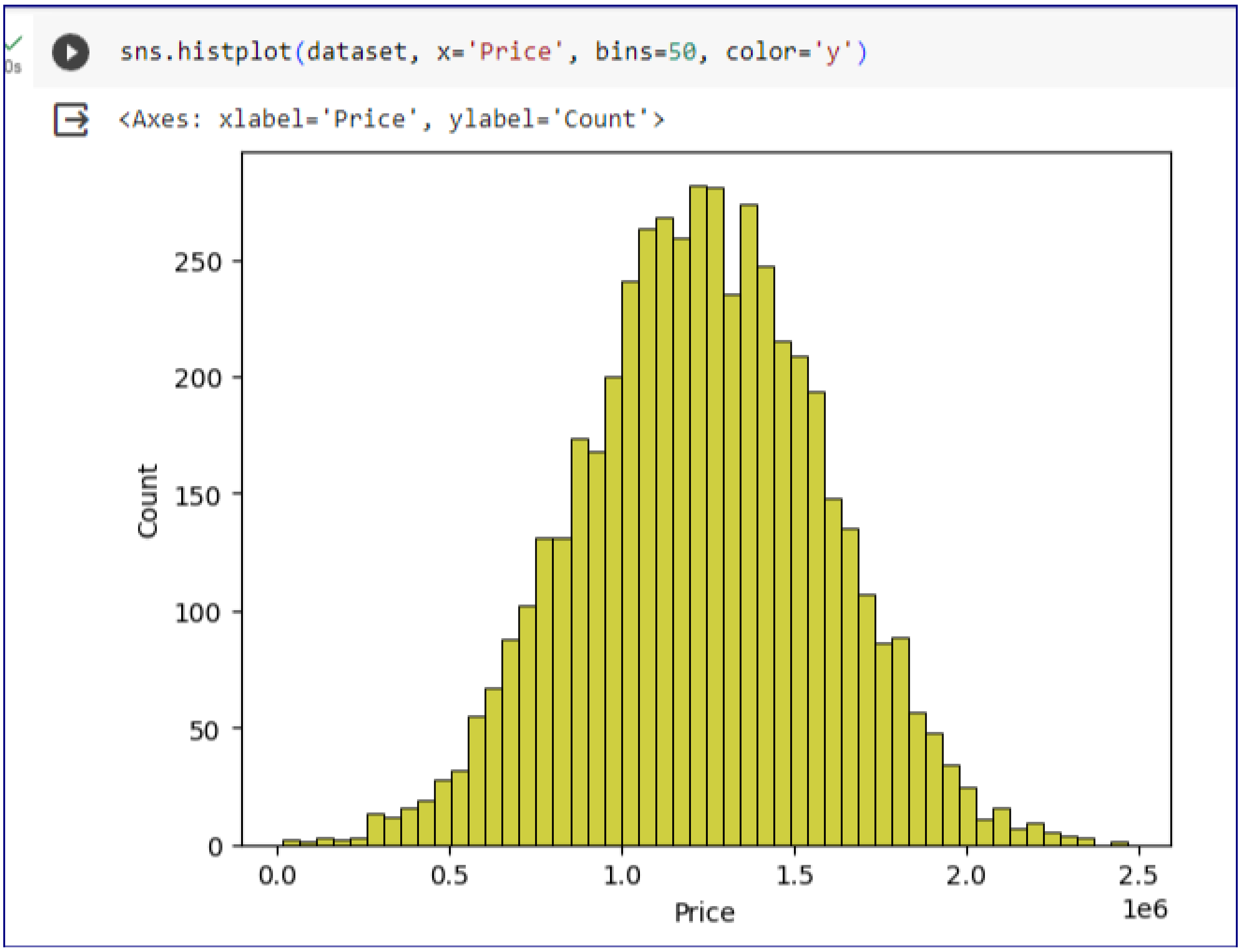


```
plt.figure(figsize=(12, 6))
sns.heatmap(dataset.corr(),
      cmap = 'BrBG',
      fmt = '.2f',
      linewidths = 2,
      annot = True)
```



This code uses Matplotlib and Seaborn to create a heatmap of the correlation matrix of your dataset.

The code generates a heatmap that visually represents the correlation between pairs of variables in your dataset. Positive correlations are shown in one color, while negative correlations are shown in another color.
The degree of correlation is indicated by the color intensity.Heatmaps are a useful tool for exploring relationships between variables and identifying potential multicollinearity (high correlations) in the data, which can be important when building predictive models.



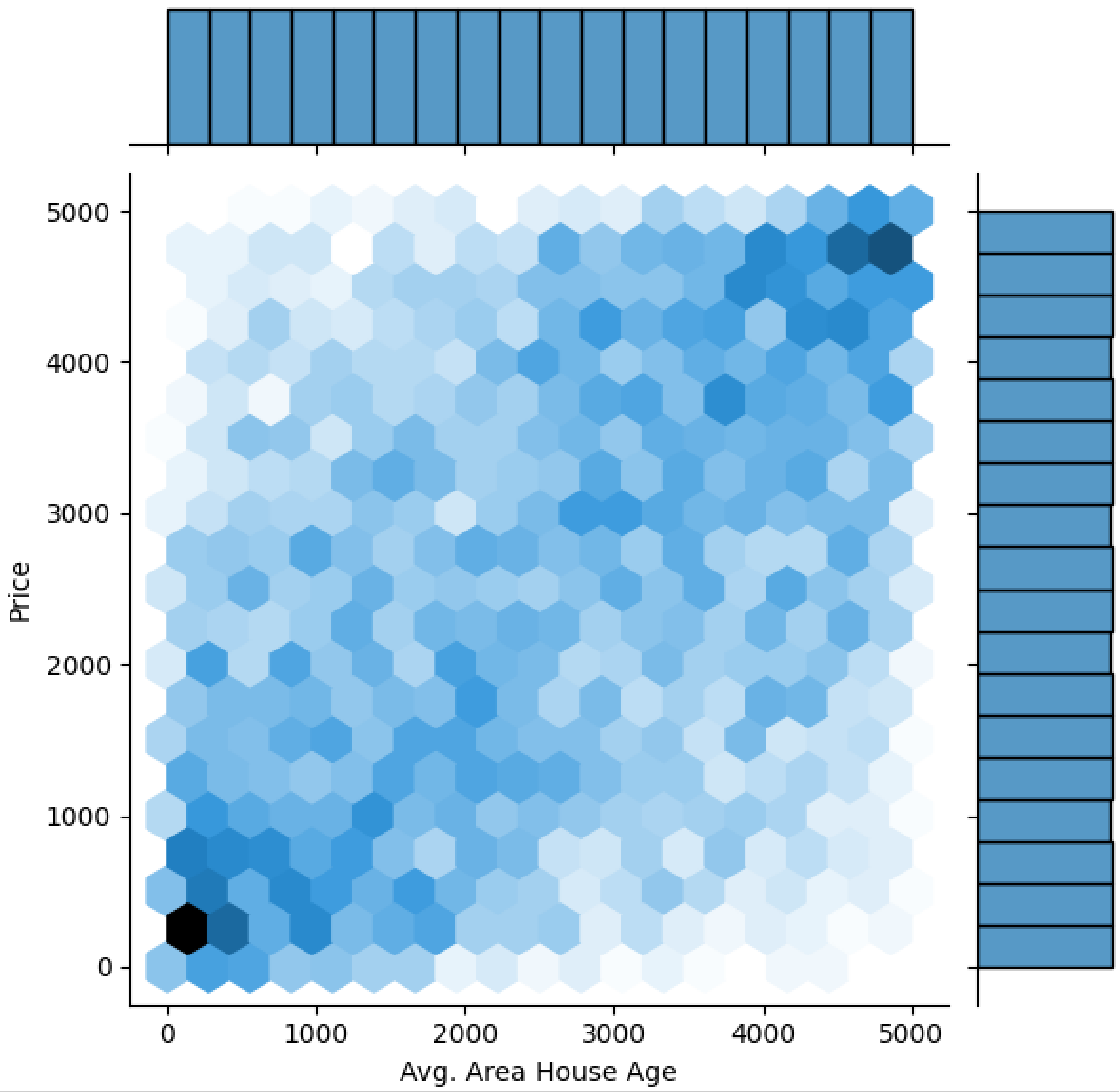
```
sns.histplot(dataset, x='Price', bins=50, color='y')
```

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

The code you've provided uses Seaborn to create a histogram plot of the 'Price' column in your dataset.
The code will generate a histogram of the 'Price' column, showing the distribution of house prices. The histogram will have 50 bins, and each bin represents a range of house prices. The height of the bars in each bin represents the frequency of houses falling within that price range.Histograms are useful for visualizing the distribution of a numerical variable and understanding its central tendency and spread. They can provide insights into the typical price range for houses in your dataset.

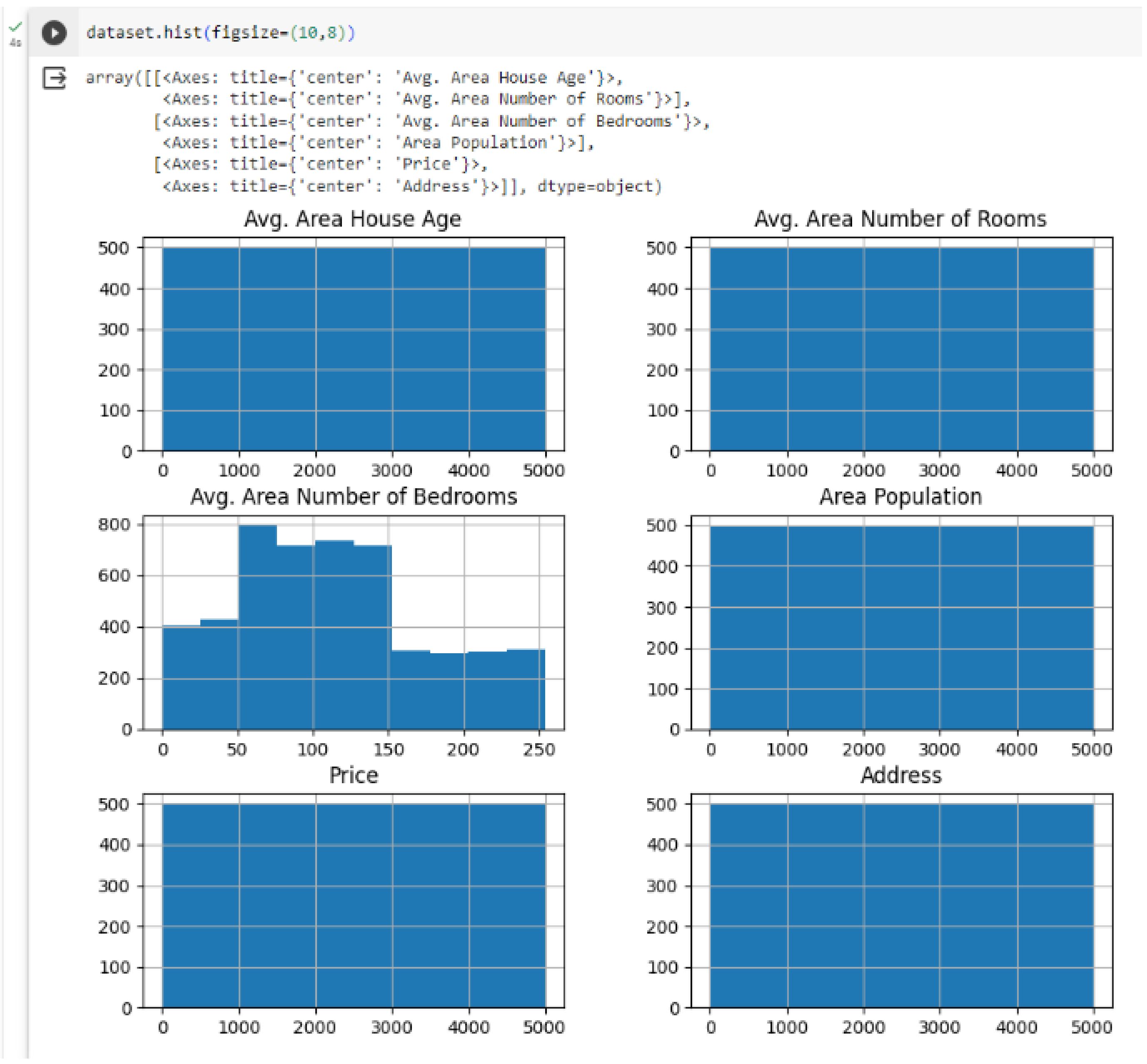
```
sns.jointplot(dataset, x='Avg. Area House Age', y='Price', kind='hex')
```

```
<seaborn.axisgrid.JointGrid at 0x7ab322e87580>
```



The code you've provided uses Seaborn to create a joint plot between the 'Avg. Area House Age' and 'Price' columns in your dataset. Specifically, it uses a hexbin plot for visualization.

The code will generate a joint plot that shows the relationship between the average area house age and house prices. The hexbin plot will represent the density of data points, with darker hexagons indicating areas where more data points are concentrated. This type of plot helps visualize the distribution and relationship between two numerical variables.



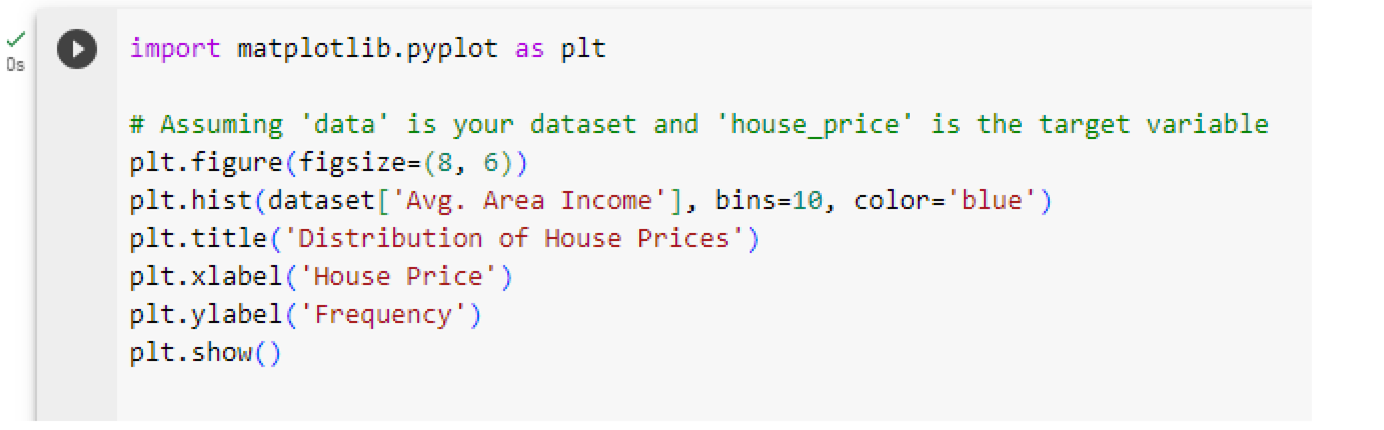
```
dataset.hist(figsize=(10,8))
```

```
array([[<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'}>,
        <Axes: title={'center': 'Address'}>]], dtype=object)
```

The code you've provided uses Seaborn to create a joint plot between the 'Avg. Area House Age' and 'Price' columns in your dataset. Specifically, it uses a hexbin plot for visualization.

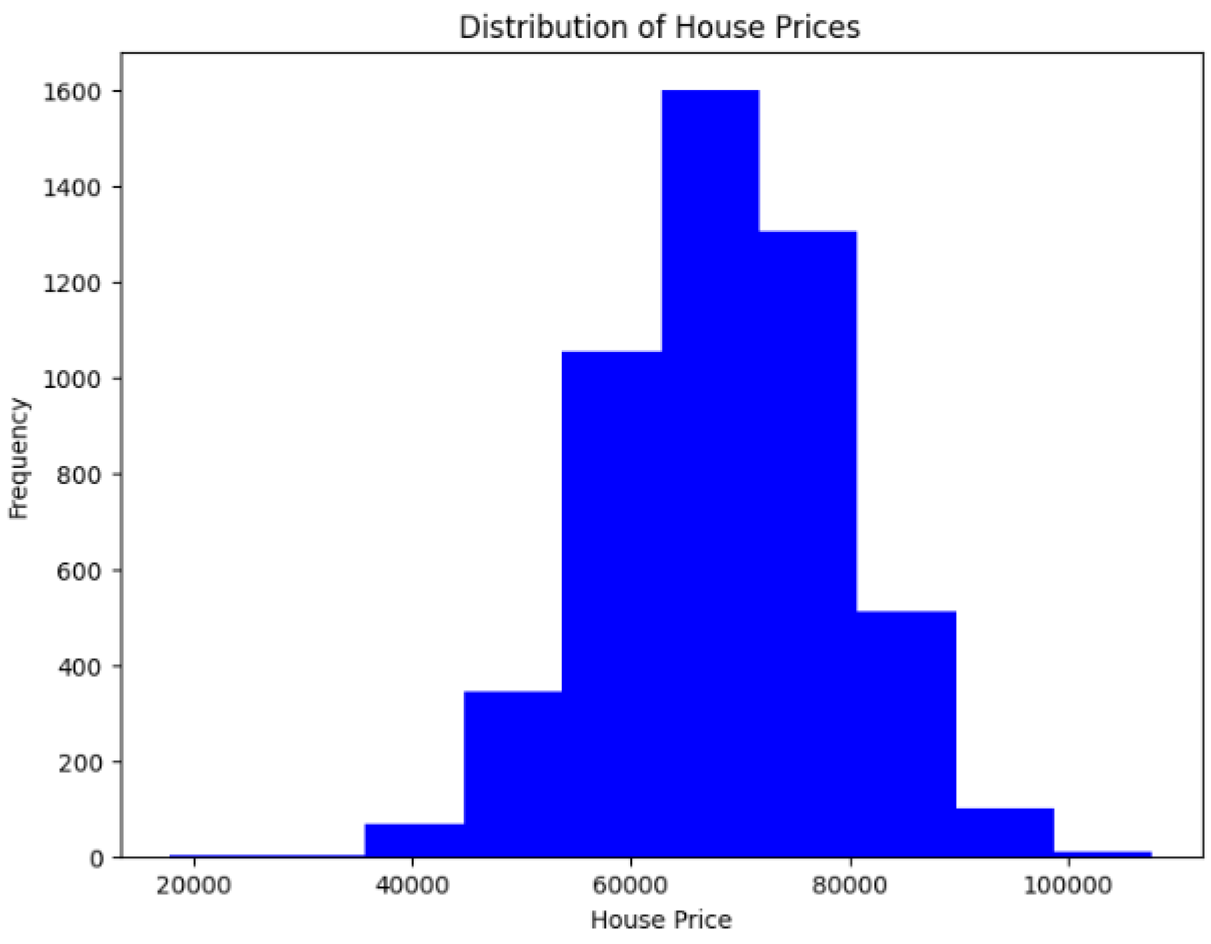
The code will generate a grid of histograms, with each histogram representing the distribution of a numerical column in your dataset. This provides a quick visual overview of the data's distribution, which can be useful for understanding the range, spread, and central tendency of each numerical variable.

Each histogram in the grid will show the frequency (or count) of data points within different intervals (bins) of the variable. This visualization is a helpful initial step in exploring your data and identifying potential patterns or outliers in the numerical features.



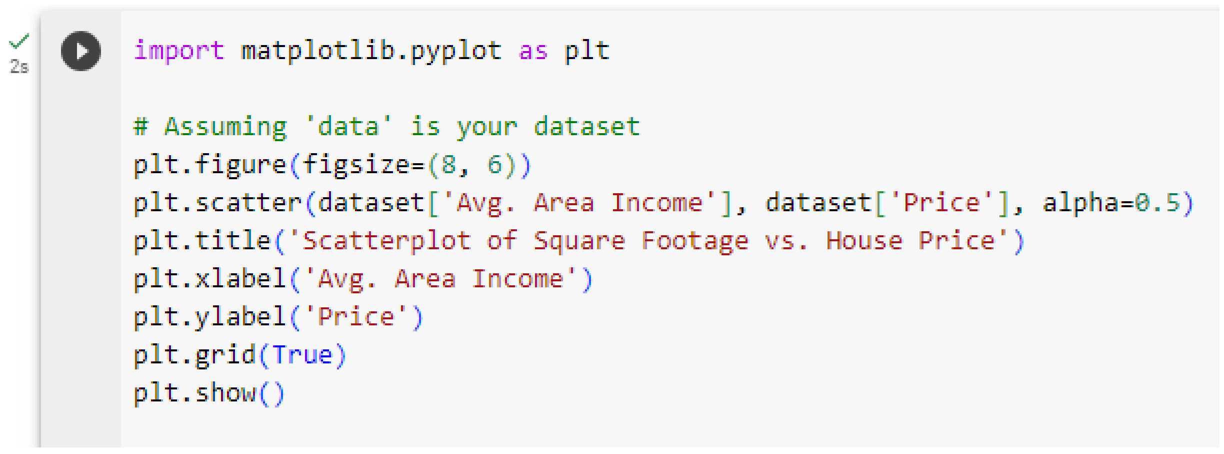
```
import matplotlib.pyplot as plt

# Assuming 'data' is your dataset and 'house_price' is the target variable
plt.figure(figsize=(8, 6))
plt.hist(dataset['Avg. Area Income'], bins=10, color='blue')
plt.title('Distribution of House Prices')
plt.xlabel('House Price')
plt.ylabel('Frequency')
plt.show()
```



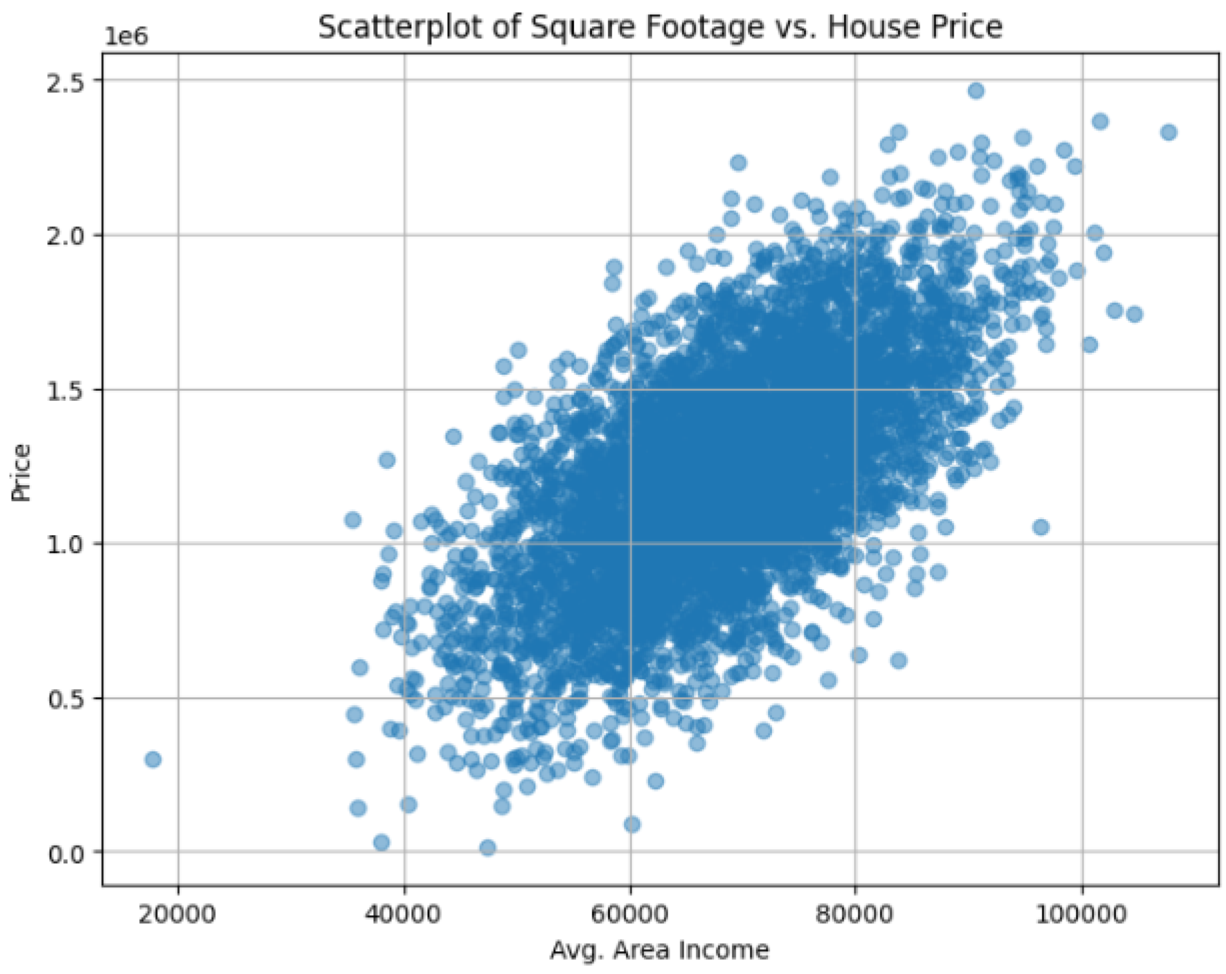
The code you provided uses the matplotlib library in Python to create a histogram of a dataset, specifically focusing on the 'Avg. Area Income' feature.

The code creates a histogram of the 'Avg. Area Income' feature from your dataset using the matplotlib library. It sets up the figure size, labels, and the number of bins for the histogram and then displays the plot. However, it might be a good idea to update the title and x-axis label to accurately reflect the data you're visualizing



```
import matplotlib.pyplot as plt

# Assuming 'data' is your dataset
plt.figure(figsize=(8, 6))
plt.scatter(dataset['Avg. Area Income'], dataset['Price'], alpha=0.5)
plt.title('Scatterplot of Square Footage vs. House Price')
plt.xlabel('Avg. Area Income')
plt.ylabel('Price')
plt.grid(True)
plt.show()
```



The code you provided uses the matplotlib library in Python to create a scatter plot of two variables from your dataset, 'Avg. Area Income' on the x-axis and 'Price' on the y-axis.

In summary, the code creates a scatter plot of 'Avg. Area Income' against 'Price' from your dataset using matplotlib. It sets up the figure size, labels for both axes, a title, and a grid for better visualization of the data points. Just ensure that the title accurately describes the data being visualized, as it currently mentions "Square Footage" instead of "Avg. Area Income."

CONCLUSION:

The process of house price prediction is greatly enhanced by a robust combination of data analysis, data preprocessing, and data cleaning. These steps are fundamental in ensuring that predictive models are accurate, reliable, and capable of providing meaningful insights to homeowners, investors, and real estate professionals.