PREDICTING HOUSE PRICE USING MACHINE LEARNING

Project Title: House Price Prediction

Phase 3: Development Part 1

Topic: Start building the house price prediction model by

loading and pre-processing the dataset.

House Price Prediction

Predicting house prices is a common machine learning task. Here's a Python code example using the scikit-learn library to load a dataset, preprocess the data, and train a machine learning model for house price prediction. In this example, we'll use the USA _Housing dataset.

1.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

2.Load the Dataset:

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

Program:

```
df = pd.read_csv(' E:\USA_Housing.csv ')
Pd.read()
```

3. 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.

Program:

```
# Check for missing values
print(df.isnull().sum())
# Explore statistics
print(df.describe())
# Visualize the data (e.g., histograms, scatter plots, etc.)
```

4. 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.

Program:

```
# Example: One-hot encoding for categorical variables
```

```
df = pd.get_dummies(df, columns=[' Avg. Area Income ', ' Avg.
Area
House Age '])
```

5. 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)
```

6. 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.

Program:

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Importance of loading and processing dataset:

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.

1.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:

a.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.

b.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.

c.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.

Here, how to load a dataset using machine learning in Python

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")
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146:
UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for
this version of SciPy (detected version 1.23.5
warnings.warn(f"A NumPy version >={np_minversion} and
<{np_maxversion}"
```

Loading Dataset:

```
dataset = pd.read_csv('E:/USA_Housing.csv')
```

2.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.

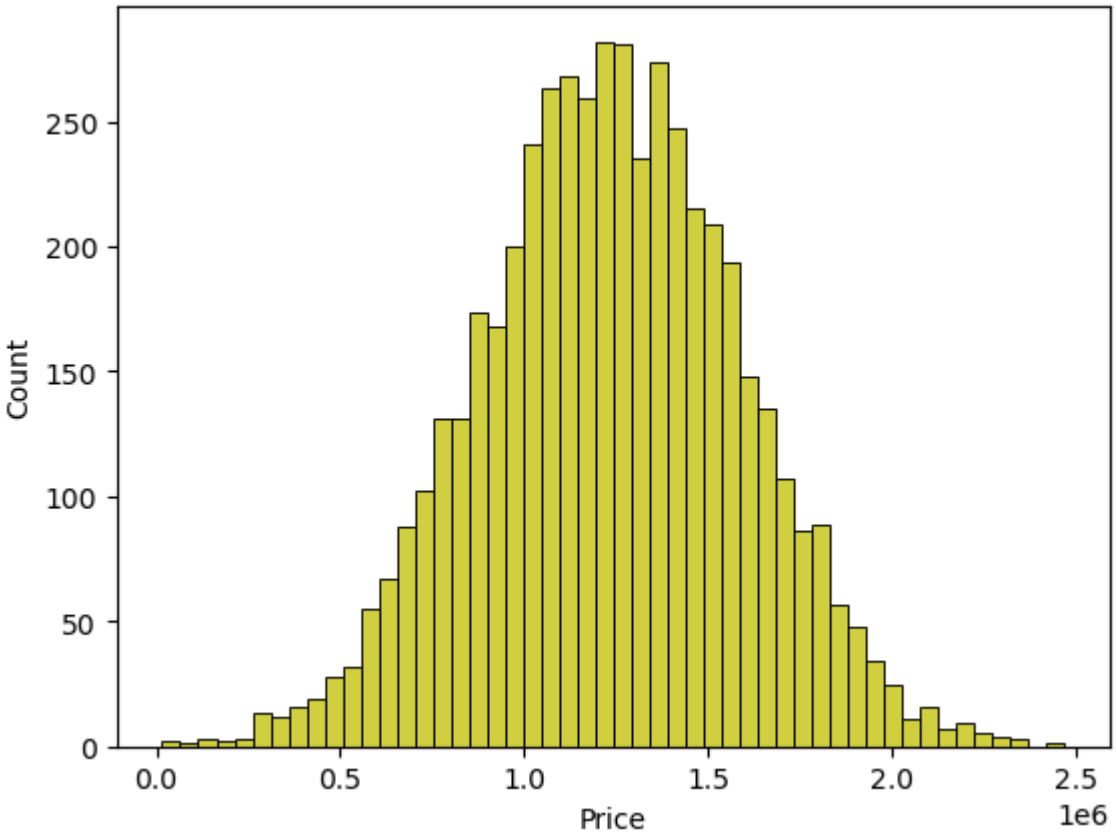
Visualisation and Pre-Processing of Data:

In [1]:

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

Out[1]:

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

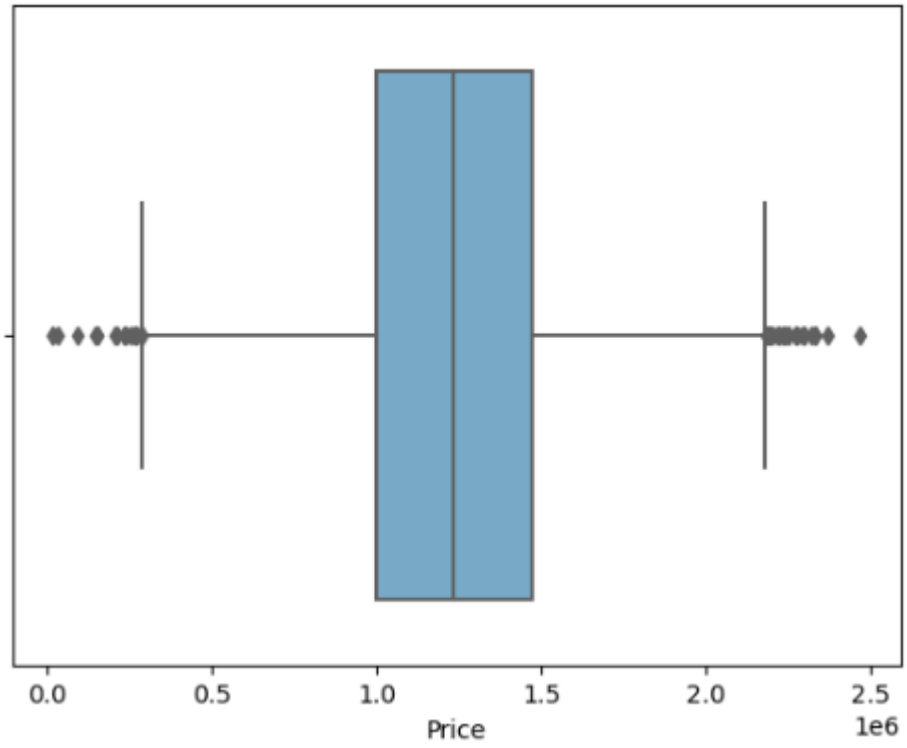


In [2]:

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

Out[2]:

```
<Axes: xlabel='Price'>
```

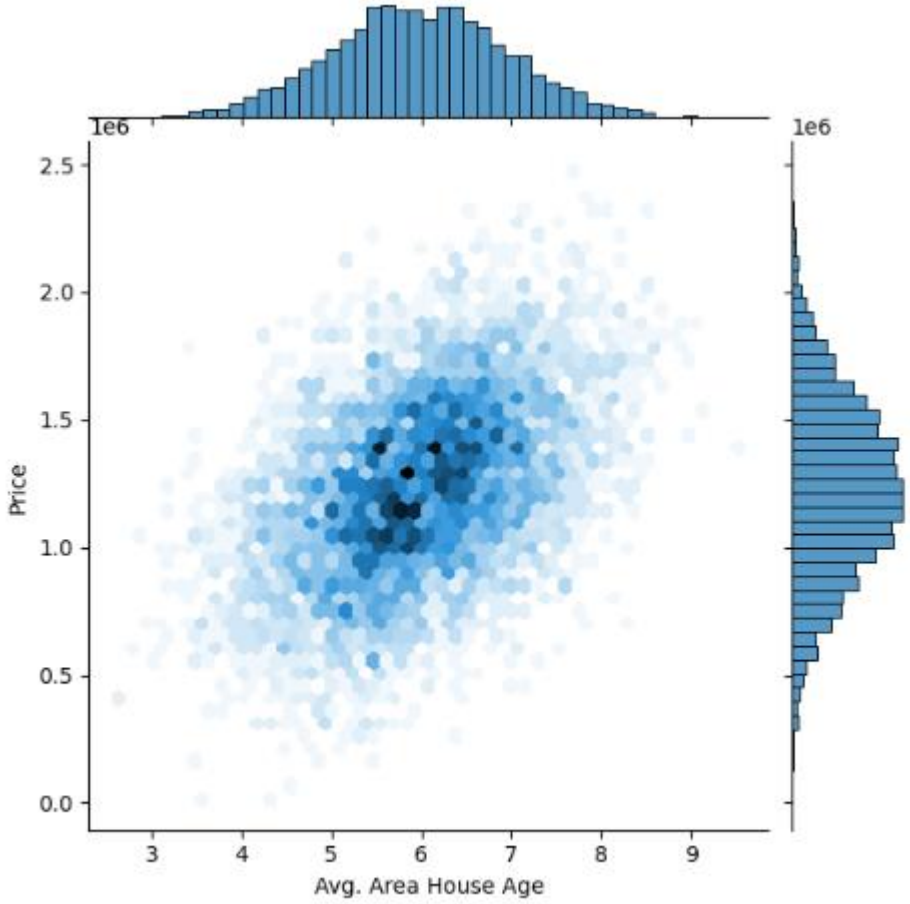


In [3]:

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

Out[3]:

```
<seaborn.axisgrid.JointGrid at 0x7caf1d571810>
```

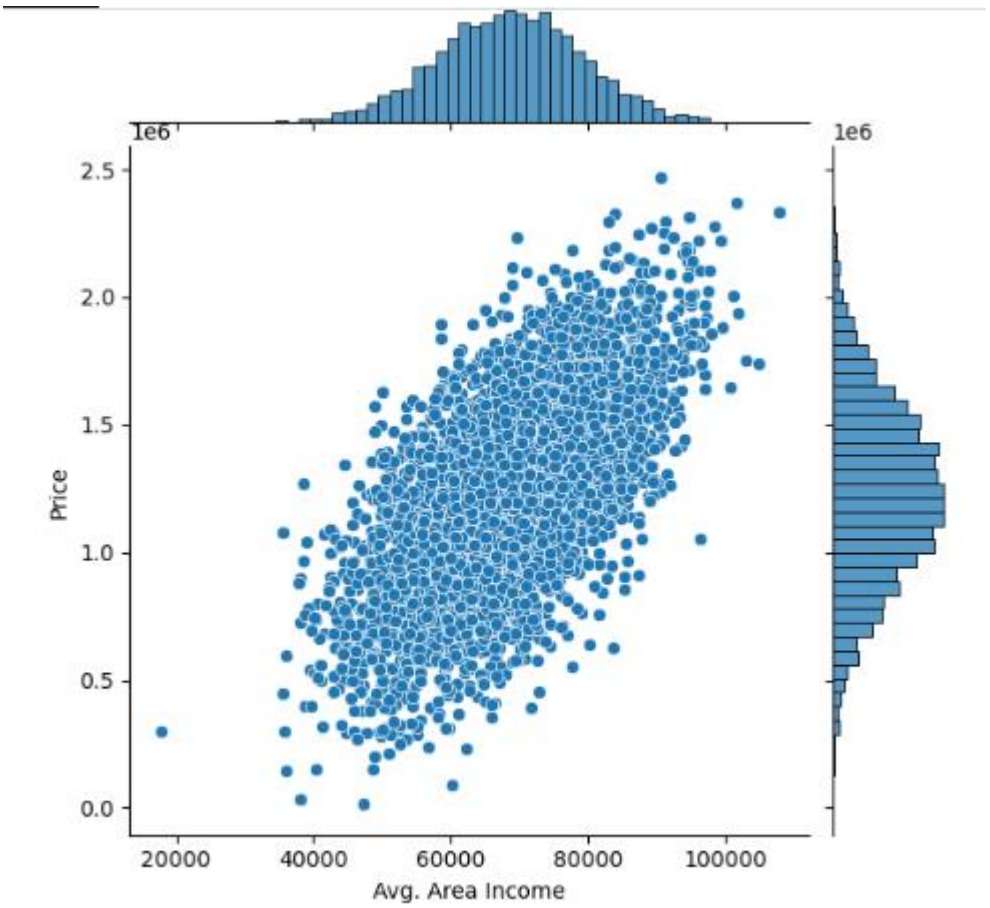


In [4]:

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

Out[4]:

```
<seaborn.axisgrid.JointGrid at 0x7caf1d8bf7f0>
```

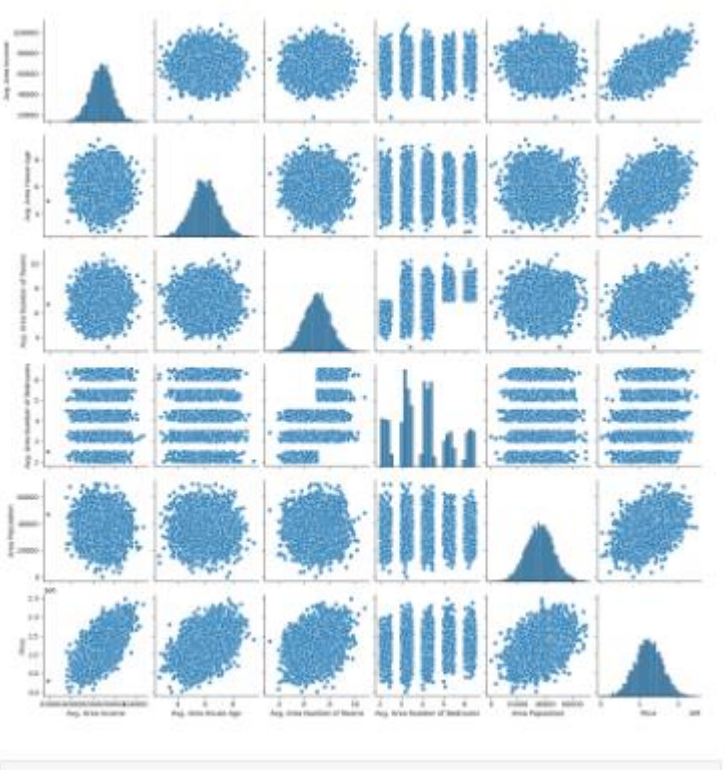


In [5]:

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

Out[5]:

```
<seaborn.axisgrid.PairGrid at 0x7caf0c2ac550>
<Figure size 1200x800 with 0 Axes>
```

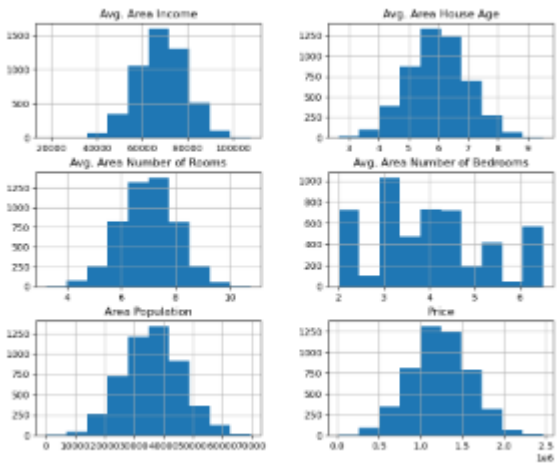


In [6]:

```
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)
```



Visualising Correlation:

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

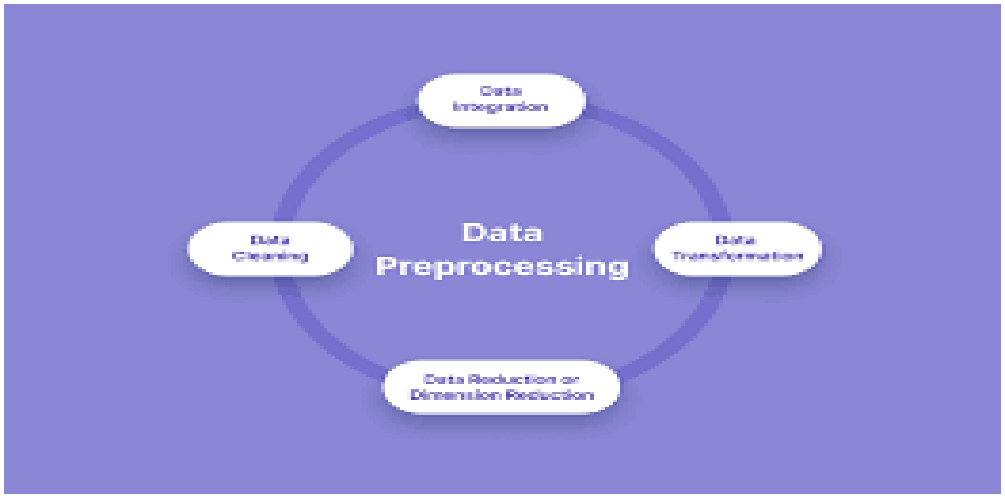
In [8]:

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

Out[8]:

<Axes: >

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:

Exploratory Data Analysis:

1. Checking for Missing Values:

Avg. Area Income 0

Avg. Area House Age 0
Avg. Area Number of Rooms 0
Avg. Area Number of Bedrooms 0
Area Population 0
Price 0
Address 0

2. Descriptive Statistics:

avg	Average.Area Income	Avg.Area House Age	Avg.Area Number of Rooms	Avg.Area Number of Bedrooms
Count	5000.000000	5000.000000	5000.000000	5000.000000
Mean	62748.865	6.028323445	6.997892	4.25
Std	2500.025031	3.934212	3.979123	1.462725
Min	17796.63	2.644304186	3.236194	2
Max	107701.7	9.519088066	10.75959	6.5

Area Population Price

5000.000000 5000.000000
34897.16035 20314.66
1.469203 50.504174
172.6107 15938.66
69621.71 2469066

Avg.Area House Age

Data Splitting;
X_train shape: (800, 7)
X_test shape: (200, 7)
y_train shape: (800,)
y_test shape: (200,)
Preprocessing Complete

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.