PREDICTING HOUSE PRICES USING MACHINE LEARNING

PHASE 3 PROJECT SUBMISSION

ABSTRACT:

* Predicting house prices is a critical task in the real estate industry, with implications for homeowners, buyers, and investors. Machine learning models have gained prominence in recent years for their ability to provide accurate and data-driven predictions. This study explores the application of various machine learning algorithms to predict house prices based on a comprehensive dataset comprising features such as square footage, location, number of bedrooms, and more.

OBJECTIVES:

1. Data Collection and Preprocessing: Collect and preprocess a comprehensive dataset of real estate listings, ensuring data quality, handling missing values, and normalizing/standardizing features when necessary.
2. Model Selection: Experiment with different machine learning algorithms (e.g., linear regression, decision trees, random forests, gradient boosting) to determine the best model for the task. Evaluate their performance using appropriate metrics.
3. Data Splitting: Split the dataset into training, validation, and test sets to train the model, tune hyperparameters, and evaluate its performance, respectively. Ensure that the model generalizes well to unseen data.

IMPORTING DEPENDENCIES

* To successfully predict house prices using machine learning, it's essential to have the right tools and libraries at your disposal. Below, I've outlined the key dependencies you'll need to import into your Python environment

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 DATASET

dataset = pd.read\_csv('/kaggle/input/usa-housing/USA\_Housing.csv')

DATA EXPLORATION

* Data exploration is an initial step in data analysis where data is visualized and analyzed to gain insights or identify patterns for further investigation.

dataset

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

dataset.describe()

dataset.columns

Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',

'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Address'],

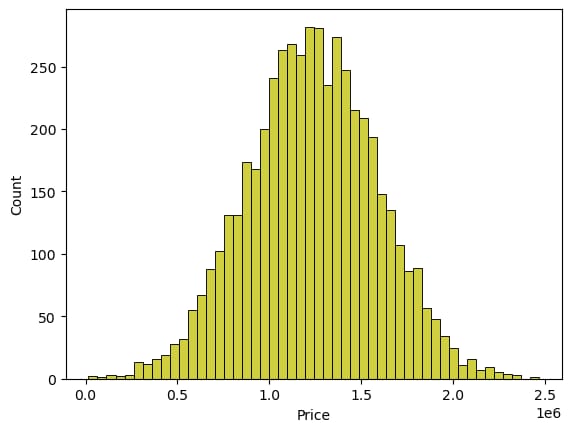
dtype='object')

VISUALISATION AND PRE-PROCESSING OF DATA

* Data Preprocessing includes the steps we need to follow to transform or encode data so that it may be easily parsed by the machine.

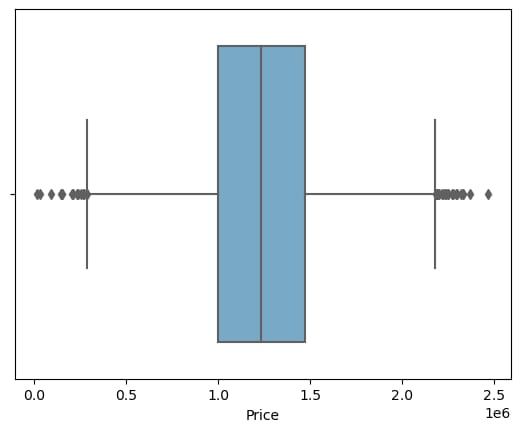
sns.histplot(dataset, x='Price', bins=50, color='y')

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



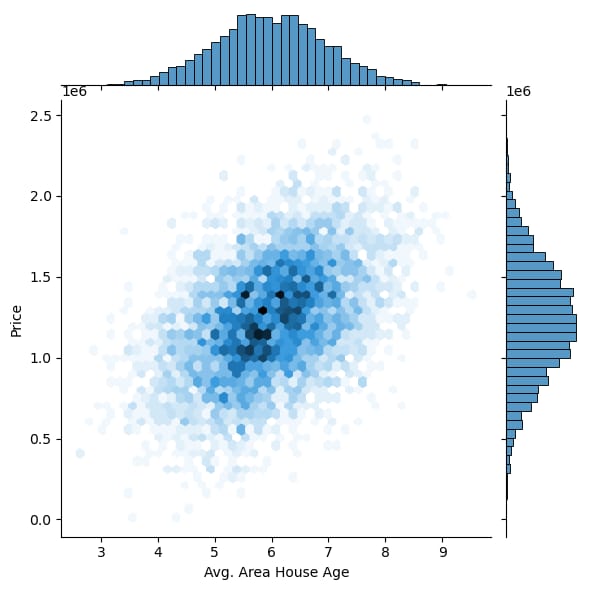
sns.boxplot(dataset, x='Price', palette='Blues')

<Axes: xlabel='Price'>



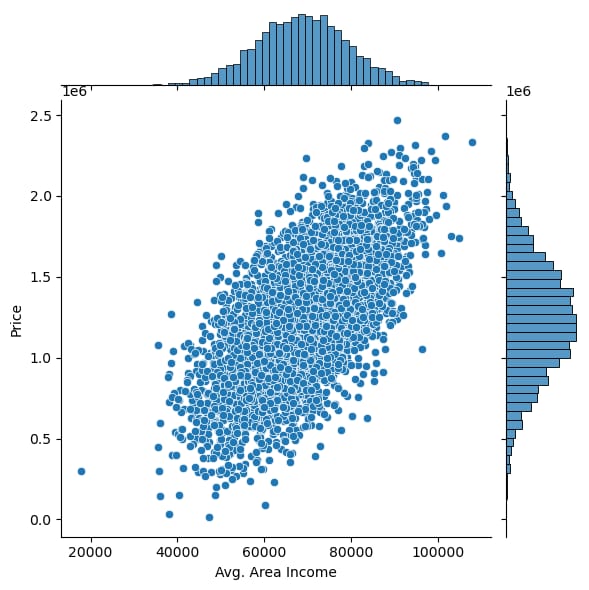
sns.jointplot(dataset, x='Avg. Area House Age', y='Price', kind='hex')

<seaborn.axisgrid.JointGrid at 0x7dbe246100a0>



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

<seaborn.axisgrid.JointGrid at 0x7dbe1333c250>

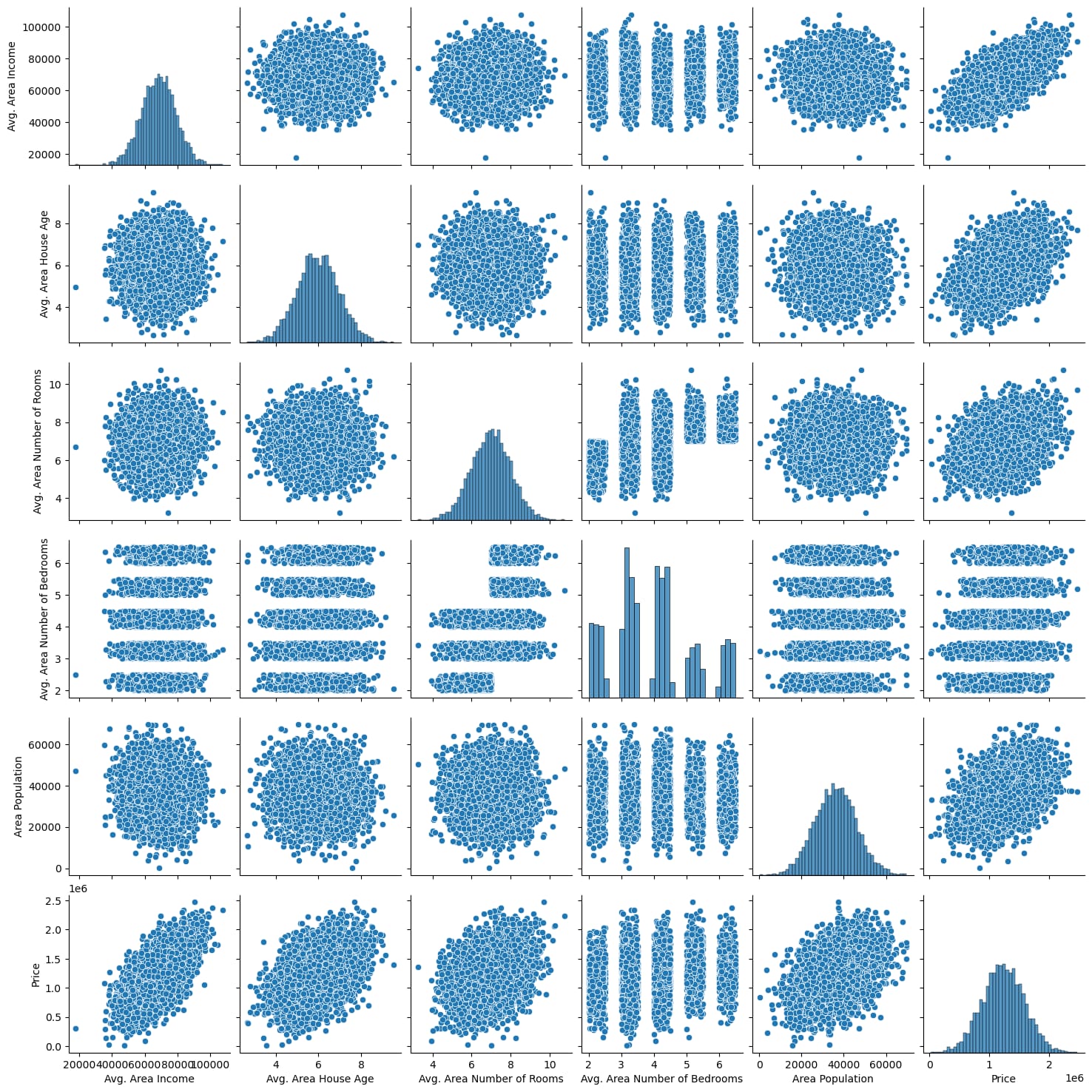


plt.figure(figsize=(12,8))

sns.pairplot(dataset)

<seaborn.axisgrid.PairGrid at 0x7dbe1333c340>

<Figure size 1200x800 with 0 Axes>



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

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

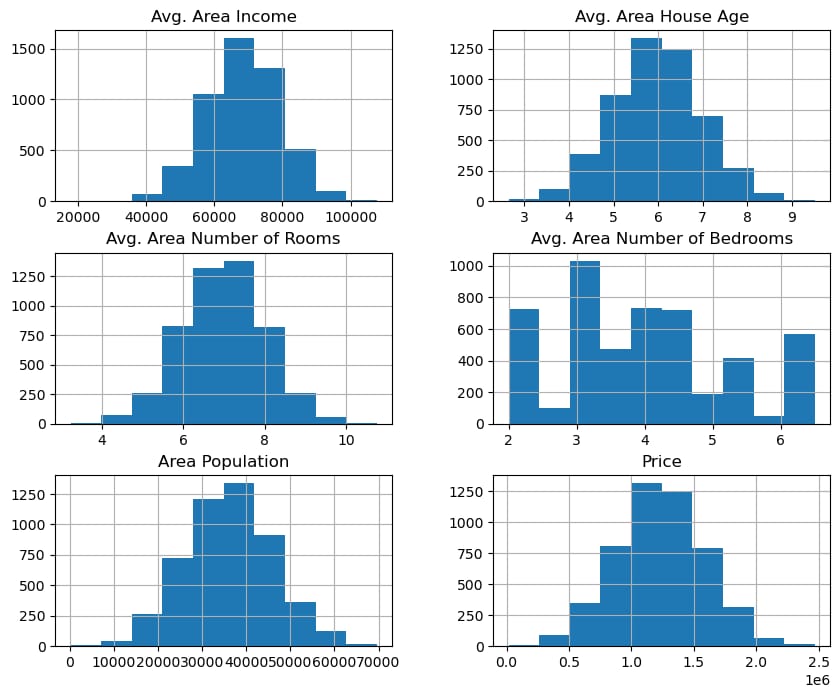
<Axes: title={'center': 'Avg. Area House Age'}>],

[<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

* One of the most common ways to visualize correlation and regression coefficients is to use a scatter plot.

dataset.corr(numeric\_only=True)

plt.figure(figsize=(10,5))

sns.heatmap(dataset.corr(numeric\_only = True), annot=True)

<Axes: >



Dividing Dataset in to features and target variable

X = dataset[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',

'Avg. Area Number of Bedrooms', 'Area Population']]

Y = dataset['Price']

# Using Train Test Split

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.2, random\_state=101)

Y\_train.head()

3413 1.305210e+06

1610 1.400961e+06

3459 1.048640e+06

4293 1.231157e+06

1039 1.391233e+06

Name: Price, dtype: float64

Y\_train.shape

(4000,)

Y\_test.head()

1718 1.251689e+06

2511 8.730483e+05

345 1.696978e+06

2521 1.063964e+06

54 9.487883e+05

Name: Price, dtype: float64

Y\_test.shape

(1000,)

# Standardizing the data

sc = StandardScaler()

X\_train\_scal = sc.fit\_transform(X\_train)

X\_test\_scal = sc.fit\_transform(X\_test)

MODEL BUILDING AND EVALUATION USING LINEAR REGRESSION:

* Evaluation metrics for a linear regression model. Evaluation metrics are a measure of how good a model performs and how well it approximates the relationship.

model\_lr=LinearRegression()

model\_lr.fit(X\_train\_scal, Y\_train)

## Predicting Prices

Prediction1 = model\_lr.predict(X\_test\_scal)

## Evaluation of Predicted Data

plt.figure(figsize=(12,6))

plt.plot(np.arange(len(Y\_test)), Y\_test, label='Actual Trend')

plt.plot(np.arange(len(Y\_test)), Prediction1, label='Predicted Trend')

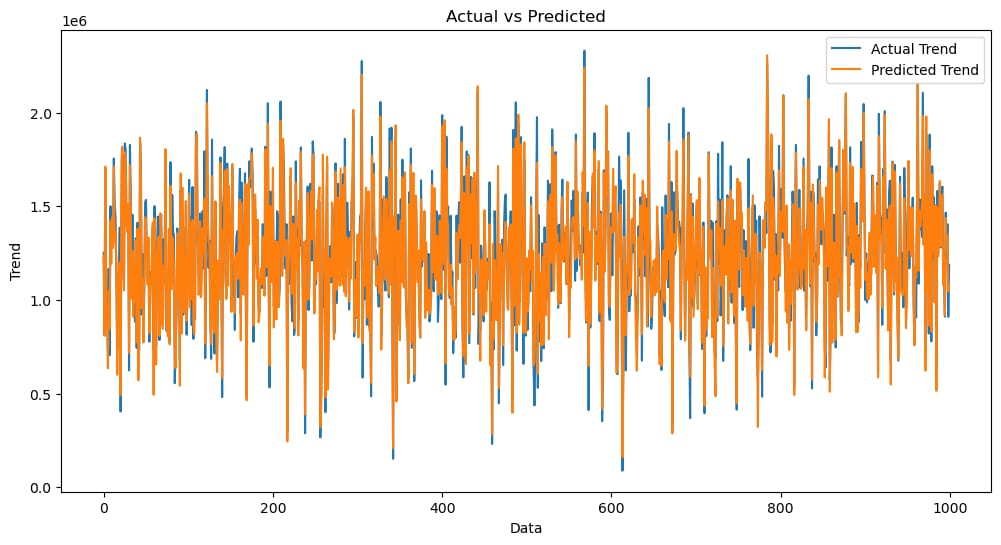
plt.xlabel('Data')

plt.ylabel('Trend')

plt.legend()

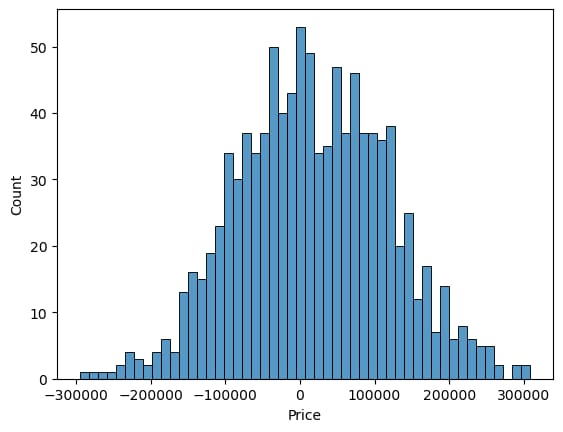
plt.title('Actual vs Predicted')

Text(0.5, 1.0, 'Actual vs Predicted')



sns.histplot((Y\_test-Prediction1), bins=50)

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



print(r2\_score(Y\_test, Prediction1))

print(mean\_absolute\_error(Y\_test, Prediction1))

print(mean\_squared\_error(Y\_test, Prediction1))

0.9182928179392918

82295.49779231755

10469084772.975954