## **Title**: **Predicting House Prices using Machine Learning**

**Phase 5: Project Documentation**

**Introduction:**

House Price prediction are very stressful work as we have to consider different things while buying a house like the structure and the rooms. This project aims to use machine learning techniques to predict house prices based on various features such as location, square footage, number of bedrooms and bathrooms, and other relevant factors. Which influence the house price. But by using the Machine learning we can easily find the house which is to be prefect for us and helps to predict the price accurately

**Problem Statement:** 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. This project aims to use machine learning techniques to predict house prices based on various features such as location, square footage, number of bedrooms and bathrooms, and other relevant factors

**Problem Definition:** The problem is to predict house prices using machine learning techniques. The objective is to develop a model that Accurately predicts the prices of houses based on a set of features such as location, square footage number of bedrooms and Bathrooms, and other relevant factors. This project involves data pre-processing, feature engineering, model selection, training, and Evaluation

**Design Thinking:**

**Data source:** understanding the Client and their Problem

A benefit to this study is that we can have two clients at the same time! (Think of being a divorce lawyer for both interested parties) However, in this case, we can have both clients with no conflict of interest!

**Client House buyer:** This client wants to find their next dream home with a reasonable price tag. They have their locations of interest ready. Now, they want to know if the house price matches the house value. With this study, they can understand which features (ex. Number of bathrooms, location, etc.) influence the final price of the house. If all matches, they can ensure that they are getting a fair price.

**Client House seller:** Think of the average house-flipper. This client wants to take advantage of the features that influence a house price the most. They typically want to buy a house at a low price and invest on the features that will give the highest return. For example, buying a house at a good location but small square footage. The client will invest on making rooms at a small cost to get a large return.

**Dataset Link:** <https://www.kaggle.com/datasets/vedavyasv/usa.housing>

**Importing required libraries.**

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")

**Data processing:** typically involves several steps of data processing. Here’s an overview of the process:

1.Data collection: Gather a dataset that includes information about houses, such as square footage, number of bedrooms, location, and sale prices. You can obtain data from sources like real estate websites, government databases, or through web scraping.

2.Data Cleaning:

- Handle missing values: Determine how to handle missing data, either by imputing values or removing rows with missing values.

- Outlier detection: Identify and deal with outliers that could skew the model’s predictions.

3. Feature Selection :

- Select relevant features: Choose which features (attributes) are most important for predicting house prices. Common features include square footage, number of bedrooms and bathrooms, location, and more.

- Engineer new features: Create new features that might improve prediction accuracy, such as the price per square foot or a distance metric to important amenities.

4.Data Transformation:

- Encode categorical variables: Convert categorical variables (like location or type of house) into numerical representations, such as one-hot encoding.

- Scale numerical features: Normalize or standardize numerical features to bring them to a similar scale.

5.Split Data:

- Divide the dataset into a training set and a testing/validation set. This allows you to train the model on one portion of the data and evaluate its performance on unseen data.

6. Model Selection:

- Choose a machine learning algorithm suitable for regression tasks, such as linear regression, decision trees, random forests, or neural networks.

- Train multiple models and compare their performance to select the best one.

7. Model Training:

- Train the selected model on the training data using appropriate hyperparameters.

- Regularize the model to prevent overfitting if needed.

8. Model Evaluation:

- Evaluate the model’s performance on the testing/validation dataset using metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE).

9. Hyperparameter Tuning:

- Fine-tune the model’s Hyperparameter to optimize its performance.

10. Deployment:

- Once satisfied with the model’s performance, deploy it for making predictions on new, unseen data.

11. Monitoring and Maintenance:

- Continuously monitor the model’s performance in a production environment and retrain or update it as needed to account for changing real estate market trends.

Remember that predicting house prices is a complex task, and the success of your model depends on the quality of your data, feature engineering, and the choice of the appropriate machine learning algorithm. Additionally, ethical considerations, such as bias in data and model, should be taken into account when using such models for real-world applications.

**Data Exploration**

Dataset: <https://www.kaggle.com/datasets/vedavyasv/usa.housing>

**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

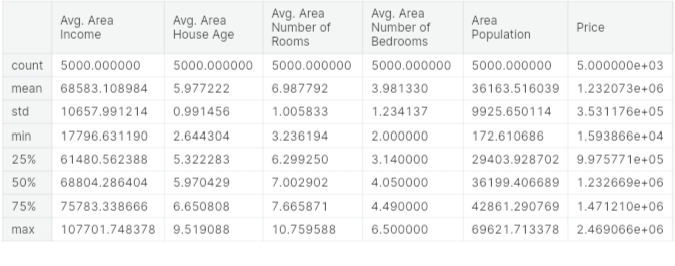
5 Price 5000 non-null float64

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:**

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 0x7caf1d571810>

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

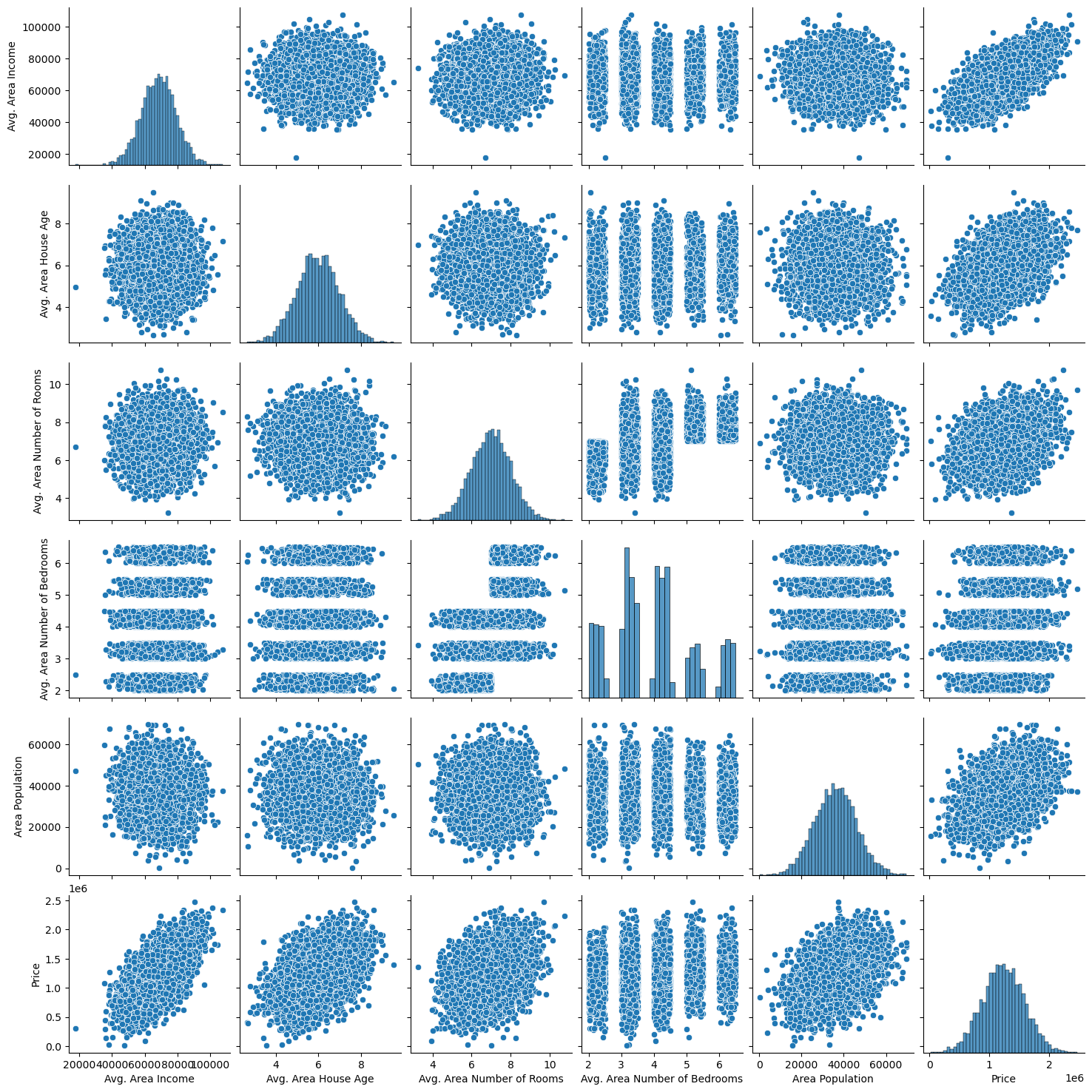
<seaborn.axisgrid.JointGrid at 0x7caf1d8bf7f0>

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

sns.pairplot(dataset)

<seaborn.axisgrid.PairGrid at 0x7caf0c2ac550>

<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’}>,

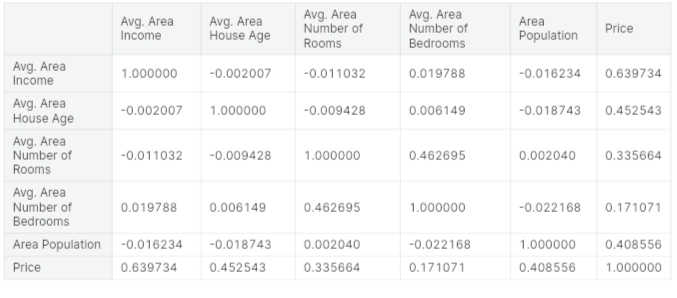
<Axes: title={‘center’: ‘Avg. Area Number of Bedrooms’}>],

[<Axes: title={‘center’: ‘Area Population’}>

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



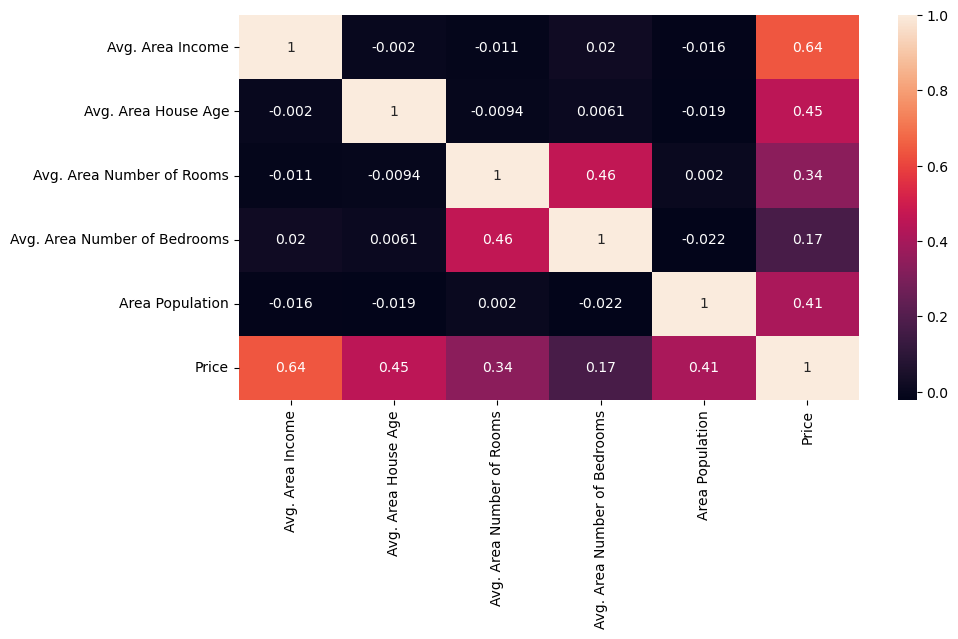
**Visualising Correlation:**

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**

**Model 1 - Linear Regression**

To predict house prices using a linear regression algorithm, you can follow these steps:

Prediction: Once you are satisfied with the model’s performance, you can use it to make predictions on new, unseen data to estimate house prices.

Remember that linear regression assumes a linear relationship between features and the target variable. If the relationship Is more complex you might consider other regression techniques or ensemble methods for better accuracy.

\_lr=LinearRegression()

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

LinearRegression

LinearRegression()

**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')

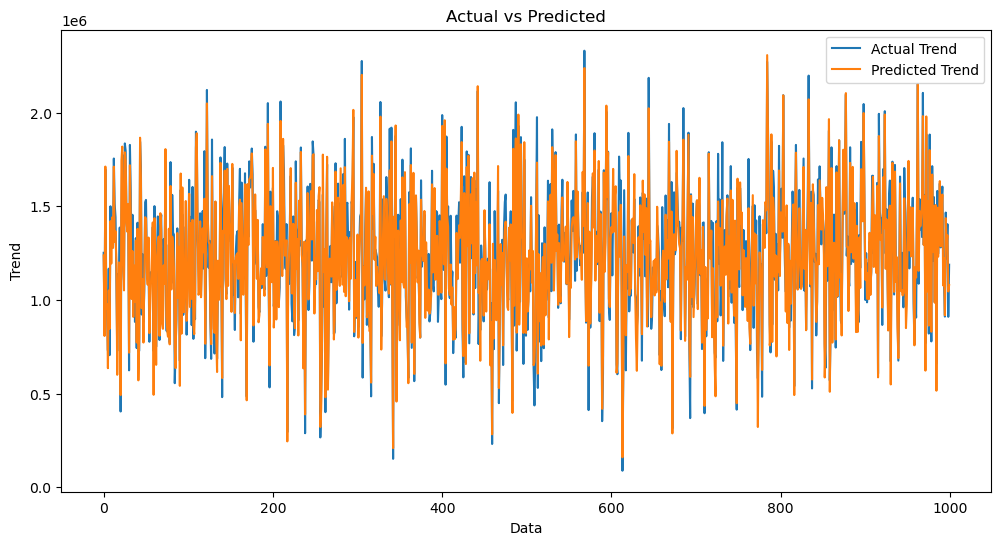
plt.xlabel('Data')

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

**Model 2- Random Forest Regressor**

Random Forest Regression is a powerful algorithm for predicting house prices using machine learning. Here are the steps to use Random Forest Regression for this task:

Prediction: Once you are satisfied with the model’s performance, you can use it to make predictions on new, unseen data to estimate house prices.

Random Forest Regression is particularly robust and less prone to overfitting compared to a single decision tree. It can capture both linear and non-linear relationships between features and house prices, making it a popular choice for regression tasks like predicting house prices.

model\_rf = RandomForestRegressor(n\_estimators=50)

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

RandomForestRegressor

RandomForestRegressor(n\_estimators=50)

**Predicting Prices**

Prediction4 = model\_rf.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)), Prediction4, label='Predicted Trend')

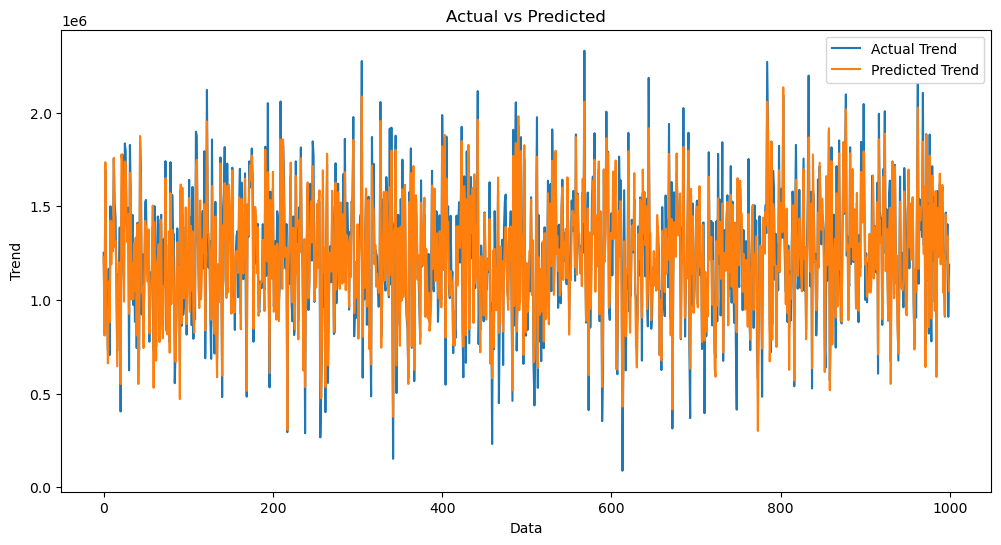
plt.xlabel('Data')

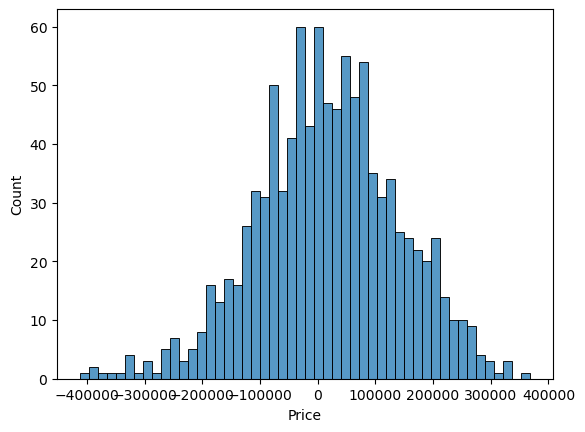
plt.ylabel('Trend')

plt.legend()

plt.title('Actual vs Predicted')

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

sns.histplot((Y\_test-Prediction4), bins=50)

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

Print(r2\_score(Y\_test, Prediction2))

Print(mean\_absolute\_error(Y\_test, Prediction2))

Print(mean\_squared\_error(Y\_test, Prediction2))

-0.0006222175925689744

286137.81086908665

128209033251.4034

**Model 3:**

**XGboost Regressor**

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way. The same code runs on major distributed environment (Kubernetes, Hadoop, SGE, disk, Spark, PySpark) and can solve problems beyond billions of examples.

model\_xg = xg.XGBRegressor()

model\_xg.fit(X\_train\_scal, Y\_trai

**XGBRegressor**

XGBRegressor(base\_score=None, booster=None, callbacks=None,

colsample\_bylevel=None, colsample\_bynode=None,

colsample\_bytree=None, early\_stopping\_rounds=None,

enable\_categorical=False, eval\_metric=None, feature\_types=None,

gamma=None, gpu\_id=None, grow\_policy=None, importance\_type=None,

interaction\_constraints=None, learning\_rate=None, max\_bin=None,

max\_cat\_threshold=None, max\_cat\_to\_onehot=None,

max\_delta\_step=None, max\_depth=None, max\_leaves=None,

min\_child\_weight=None, missing=nan, monotone\_constraints=None,

n\_estimators=100, n\_jobs=None, num\_parallel\_tree=None,

predictor=None, random\_state=None, ...)

**Predicting Prices**

Prediction5 = model\_xg.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)), Prediction5, label='Predicted Trend')

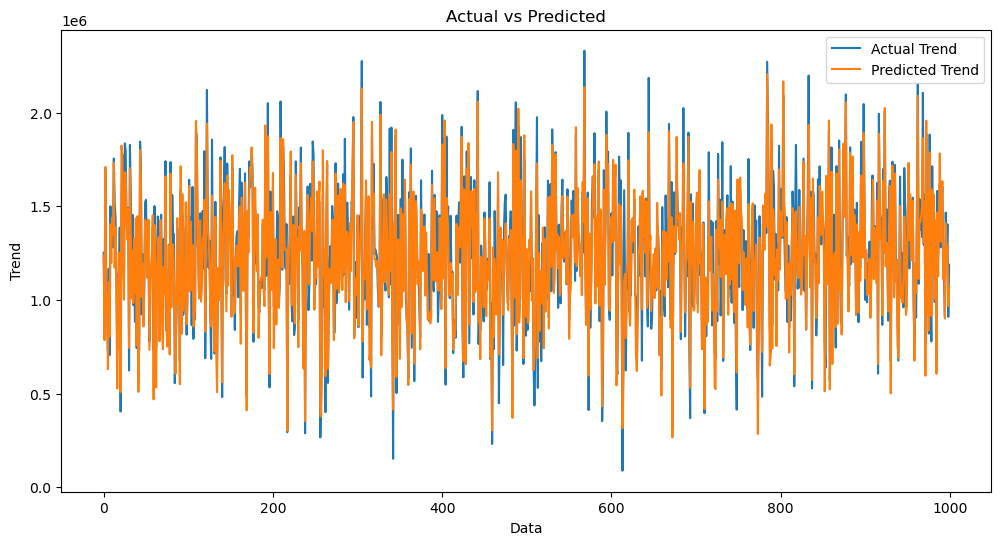
plt.xlabel('Data')

plt.ylabel('Trend')

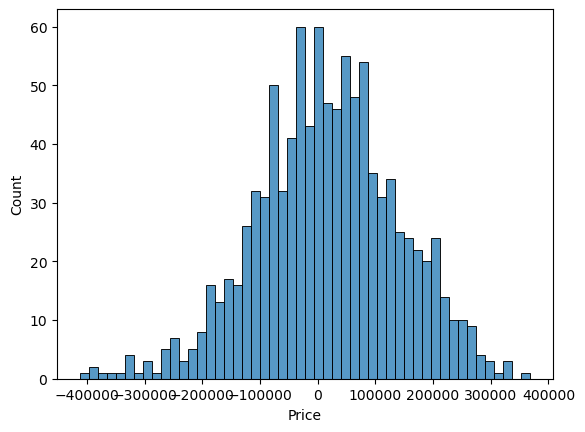
plt.legend()

plt.title('Actual vs Predicted')

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

sns.histplot((Y\_test-Prediction4), bins=50)

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

**Conclusion:**

The research presented in this paper demonstrates the potential of machine learning algorithms for accurately predicting house prices. With the proper data and features, a well-trained model based on Linear Regression can be used to accurately predict the price of a house. However the accuracy levels can vary based on the datasets used. While the results of this study are promising, there are many opportunities for future research. For instance, exploring different model architectures, such as deep learning and transfer learning, can improve model performance. Additionally, further research could be done to identify the most important features for house price prediction, as well as to explore the impact of different types of data, such as location and neighbourhood characteristics, on model performance. Finally, developing more efficient methods for training and deploying models could enable the use of machine learning algorithms in a wide range of applications.