# House price prediction

**INTRODUCTION:**

**What is house price prediction?**

House price prediction can help the developer determine the selling price of a house and can help the customer to arrange the right time to purchase a house. There are three factors that influence the price of a house which include physical conditions, concept and location.

**What is the prediction of housing prices in India?**

Property prices in India are expected to increase 7.5% on a pan-India basis this year, the fastest growth in five years, according to a Reuters poll of property analysts. Average house prices were forecast to rise 6% next year and in 2024. The poll of 13 property analysts were held during May11-27.

**How to predict house price in machine learning?**

The machine learning model is given the test data but without the price of the properties in order to predict the price for them given the various features for the properties. The predicted price is then compared to the actual price in the test data.

**IMPORTANT DEPENDENCIES:**

## **Exploratory Data Analysis :**

[EDA](https://www.geeksforgeeks.org/what-is-exploratory-data-analysis/) refers to the deep analysis of data so as to discover different patterns and spot anomalies. Before making inferences from data it is essential to examine all your variables.

So here let’s make a [heatmap](https://www.geeksforgeeks.org/how-to-create-a-seaborn-correlation-heatmap-in-python/) using seaborn library.

## **Data Cleaning :**

[Data Cleaning](https://www.geeksforgeeks.org/data-preprocessing-in-data-mining/) is the way to improvise the data or remove incorrect, corrupted or irrelevant data.

As in our dataset, there are some columns that are not important and irrelevant for the model training. So, we can drop that column before training. There are 2 approaches to dealing with empty/null values

* We can easily delete the column/row (if the feature or record is not much important).
* Filling the empty slots with mean/mode/0/NA/etc. (depending on the dataset requirement).

## **Model and Accuracy :**

As we have to train the model to determine the continuous values, so we will be using these regression models.

* SVM-Support Vector Machine
* Random Forest Regressor
* Linear Regressor

### 

### **Competition Description :**



Ask a home buyer to describe their dream house, and they probably won't begin with the height of the basement ceiling or the proximity to an east-west railroad. But this playground competition's dataset proves that much more influences price negotiations than the number of bedrooms or a white-picket fence.

With 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa, this competition challenges you to predict the final price of each home.

### **Practice Skills :**

* Creative feature engineering
* Advanced regression techniques like random forest and gradient boosting

### **Goal :**

It is your job to predict the sales price for each house. For each Id in the test set, you must predict the value of the SalePrice variable.

**INPUT:**

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")

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

dataset

dataset.info()

dataset.describe()

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

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

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

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

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

sns.pairplot(dataset)

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

dataset.corr(numeric\_only=True)

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

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

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

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

Y\_train.head()

Y = dataset['Price']

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

Y\_train.head()

Y\_train.shape

Y\_test.head()

Y\_test.shape

sc = StandardScaler()

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

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

model\_lr=LinearRegression()

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

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

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')

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

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

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

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

model\_svr = SVR()

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

Prediction2 = model\_svr.predict(X\_test\_scal)

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

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

plt.plot(np.arange(len(Y\_test)), Prediction2, label='Predicted Trend')

plt.xlabel('Data')

plt.ylabel('Trend')

plt.legend()

plt.title('Actual vs Predicted')

sns.histplot((Y\_test-Prediction2), bins=50)

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

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

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

model\_lar = Lasso(alpha=1)

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

Prediction3 = model\_lar.predict(X\_test\_scal)

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

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

plt.plot(np.arange(len(Y\_test)), Prediction3, label='Predicted Trend')

plt.xlabel('Data')

plt.ylabel('Trend')

plt.legend()

plt.title('Actual vs Predicted')

sns.histplot((Y\_test-Prediction3), bins=50)

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

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

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

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

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

Prediction4 = model\_rf.predict(X\_test\_scal)

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')

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

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

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

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

model\_xg = xg.XGBRegressor()

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

Prediction5 = model\_xg.predict(X\_test\_scal)

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')

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

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

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

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

*### Linear Regression is giving us best Accuracy.*

Output:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Avg. Area  Income | Avg. Area  House Age | Avg. Area Number of Rooms | Avg. Area Number of Bedrooms | Area  Population | Price | Address |
| 0 | 79545.458574 | 5.682861 | 7.009188 | 4.09 | 23086.80003 | 1.059034e+06 | 208 Michael Ferry Apt. 674\nLaurabury, NE 3701... |
| 1 | 79248.642455 | 6.002900 | 6.730821 | 3.09 | 40173.072174 | 1.505891e+06 | 188 Johnson Views Suite 079\nLake Kathleen, CA... |
| 2 | 61287.067179 | 5.865890 | 8.512727 | 5.13 | 36882.159400 | 1.058988e+06 | 9127 Elizabeth Stravenue\nDanieltown, WI 06482... |
| 3 | 63345.240046 | 7.188236 | 5.586729 | 3.26 | 34310.242831 | 1.260617e+06 | USS Barnett\nFPO AP 44820 |
| 4 | 59982.197226 | 5.040555 | 7.839388 | 4.23 | 26354.109472 | 6.309435e+05 | USNS Raymond\nFPO AE 09386 |
| … | … | … | … | … | … | … | … |
| 4995 | 60567.944140 | 7.830362 | 6.137356 | 3.46 | 22837.361035 | 1.060194e+06 | USNS Williams\nFPO AP 30153-7653 |
| 4996 | 78491.275435 | 6.999135 | 6.576763 | 4.02 | 25616.115489 | 1.482618e+06 | PSC 9258, Box 8489\nAPO AA 42991- 3352 |
| 4997 | 63390.686886 | 7.250591 | 4.805081 | 2.13 | 33266.145490 | 1.030730e+06 | 4215 Tracy Garden Suite 076\nJoshualand, VA 01... |
| 4998 | 68001.331235 | 5.534388 | 7.130144 | 5.44 | 42625.620156 | 1.198657e+06 | USS Wallace\nFPO AE 73316 |
| 4999 | 65510.581804 | 5.992305 | 6.792336 | 4.07 | 46501.283803 | 1.298950e+0637778 | 37778 George Ridges Apt. 509\nEast Holly, NV 2... |

|  | **Avg. Area Income** | **Avg. Area House Age** | **Avg. Area Number of Rooms** | **Avg. Area Number of Bedrooms** | **Area Population** | **Price** |
| --- | --- | --- | --- | --- | --- | --- |
| **count** | 5000.000000 | 5000.000000 | 5000.000000 | 5000.000000 | 5000.000000 | 5.000000e+03 |
| **mean** | 68583.108984 | 5.977222 | 6.987792 | 3.981330 | 36163.516039 | 1.232073e+06 |
| **std** | 10657.991214 | 0.991456 | 1.005833 | 1.234137 | 9925.650114 | 3.531176e+05 |
| **min** | 17796.631190 | 2.644304 | 3.236194 | 2.000000 | 172.610686 | 1.593866e+04 |
| **25%** | 61480.562388 | 5.322283 | 6.299250 | 3.140000 | 29403.928702 | 9.975771e+05 |
| **50%** | 68804.286404 | 5.970429 | 7.002902 | 4.050000 | 36199.406689 | 1.232669e+06 |
| **75%** | 75783.338666 | 6.650808 | 7.665871 | 4.490000 | 42861.290769 | 1.471210e+06 |
| **max** | 107701.748378 | 9.519088 | 10.759588 | 6.500000 | 69621.713378 | 2.469066e+06 |

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

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

dtype='object')

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

****

<Axes: xlabel='Price'>

****

<seaborn.axisgrid.JointGrid at 0x7dbe246100a0>

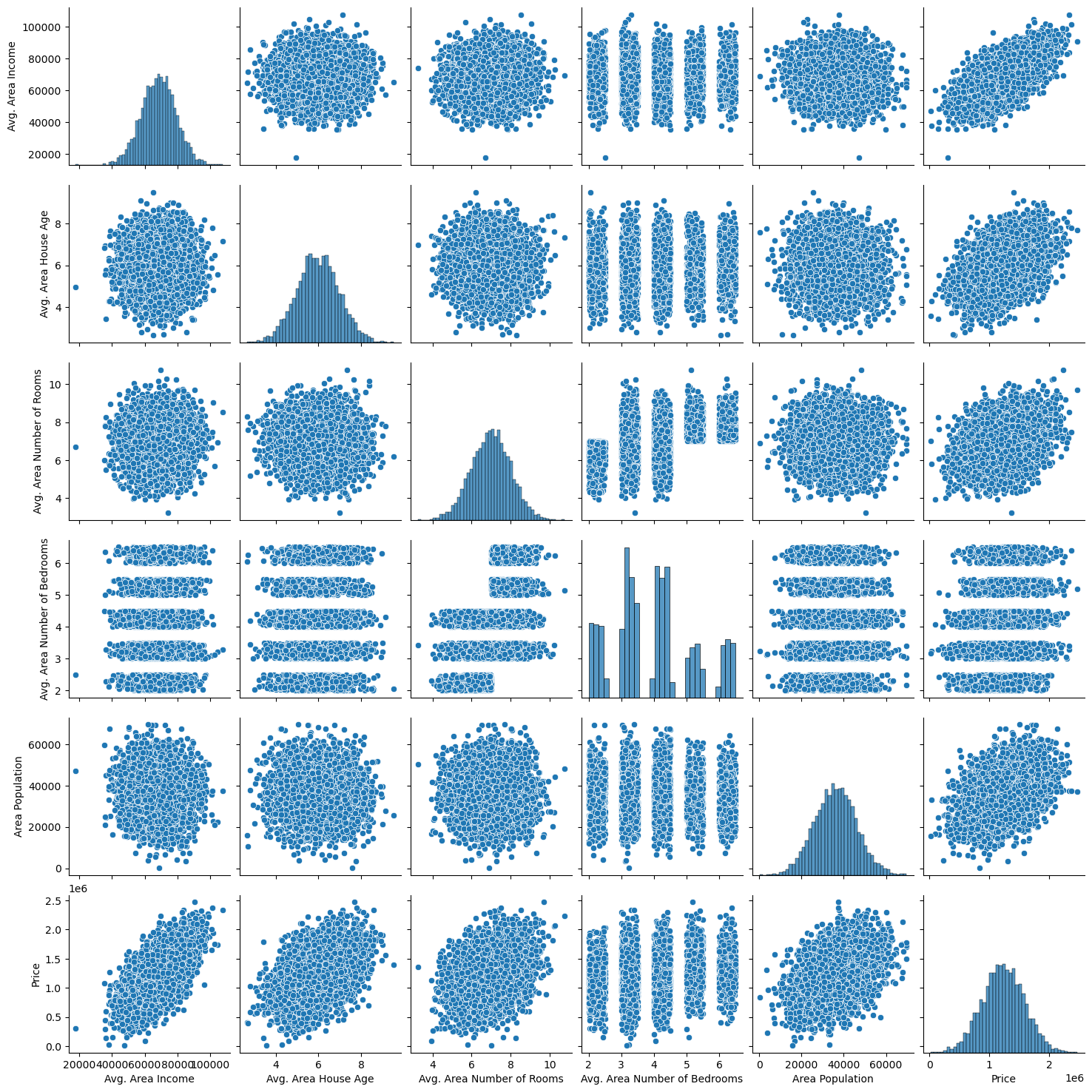


<seaborn.axisgrid.JointGrid at 0x7dbe1333c250>



<seaborn.axisgrid.PairGrid at 0x7dbe1333c340>

<Figure size 1200x800 with 0 Axes>

****

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'}>],

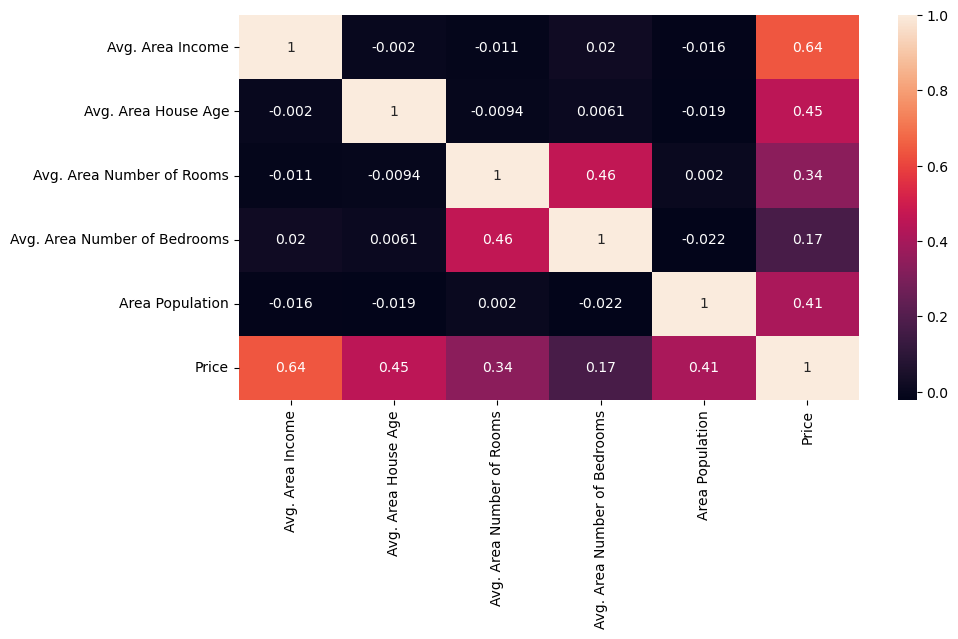
[<Axes: title={'center': 'Area Population'}>,

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



|  | 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 |

<Axes: >



3413 1.305210e+06

1610 1.400961e+06

3459 1.048640e+06

4293 1.231157e+06

1039 1.391233e+06

Name: Price, dtype: float64

(4000,)

1718 1.251689e+06

2511 8.730483e+05

345 1.696978e+06

2521 1.063964e+06

54 9.487883e+05

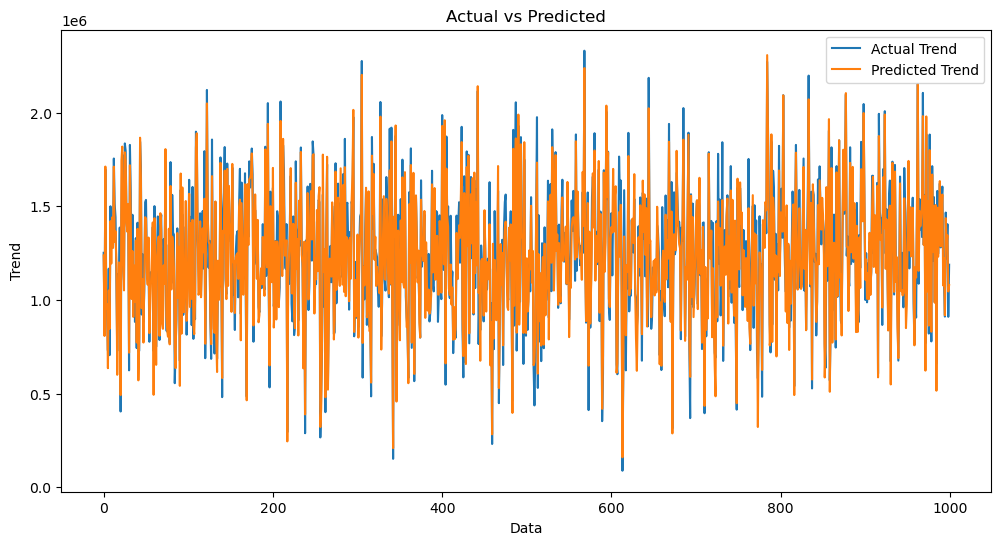
Name: Price, dtype: float64

(1000,)

LinearRegression

LinearRegression()

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



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



SVR

SVR()

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



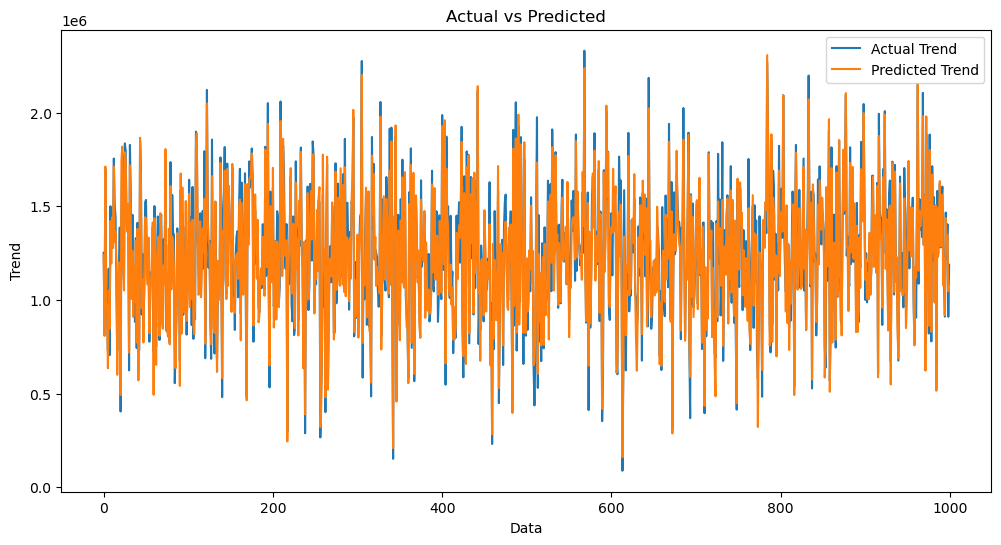
<Axes: xlabel='Price', ylabel='Count'>



Lasso

Lasso(alpha=1)

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



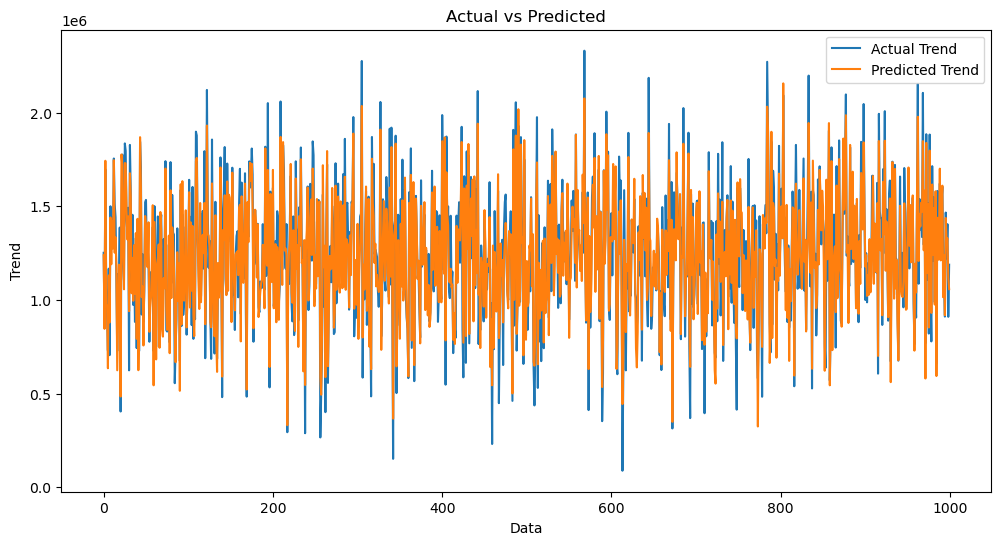
<Axes: xlabel='Price', ylabel='Count'>



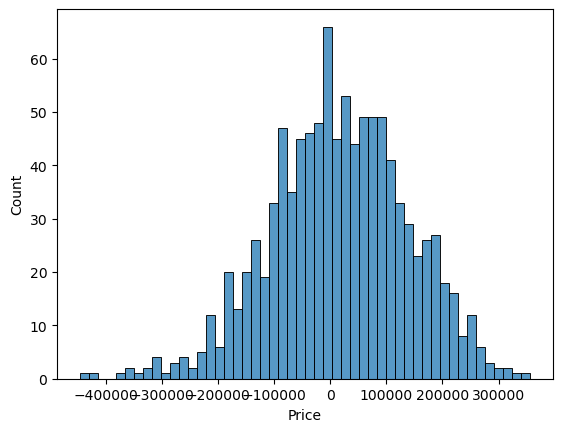
RandomForestRegressor

RandomForestRegressor(n\_estimators=50)

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



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

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,

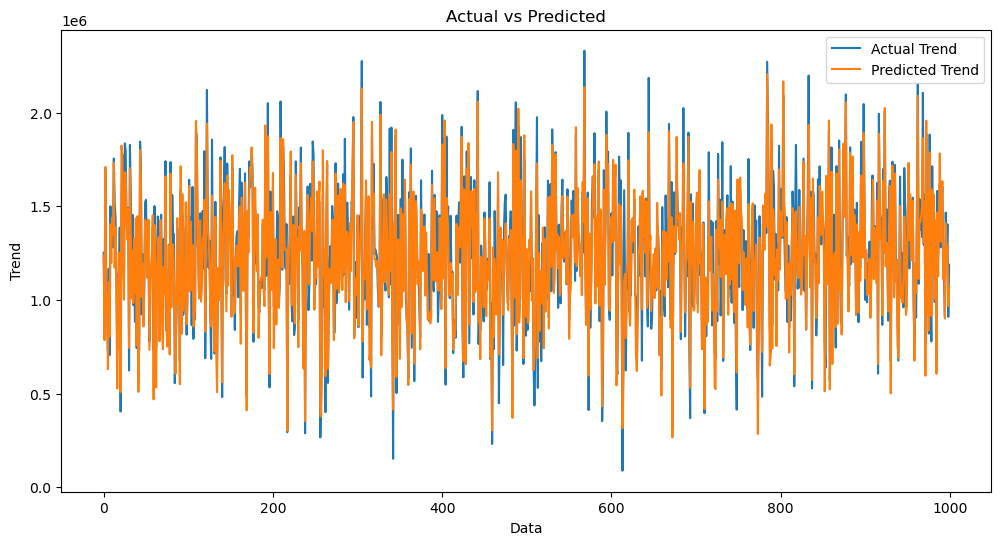
max\_delta\_step=None, max\_depth=None, max\_leaves=None,

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

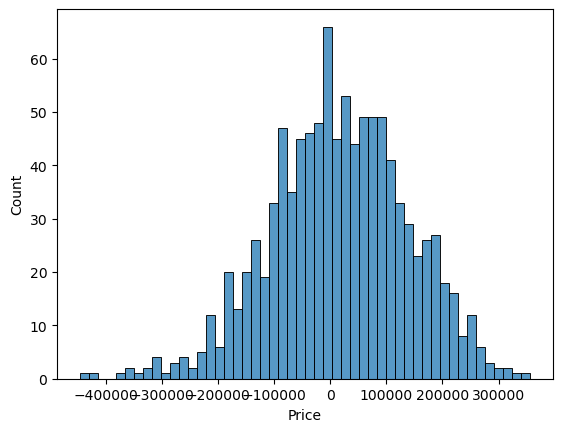
n\_estimators=100, n\_jobs=None, num\_parallel\_tree=None,

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

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



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



**CONCLUSION :**

To conclude, the application of machine learning in property research is still at an early stage. We hope this study has moved a small step ahead in providing some methodological and empirical contributions to property appraisal, and presenting an alternative approach to the valuation of housing prices. Future direction of research may consider incorporating additional property transaction data from a larger geographical location with more features, or analysing other property types beyond housing development.