**Predicting house prices**

**using machine learning**

**DEVELOPMENT PART 1**

**Introduction :**

Accurately estimating the value of real estate is an important problem for many stakeholders including house owners, house buyers, agents, creditors, and investors. It is also a difficult one. Though it is common knowledge that factors such as the size, number of rooms and location affect the price, there are many other things at play. Additionally, prices are sensitive to changes in market demand and the peculiarities of each situation, such as when a property needs to be urgently sold. The sales price of a property can be predicted in various ways, but is often based on regression techniques. All regression techniques essentially involve one or more predictor variables as input and a single target variable as output. In this paper, we compare different machine learning methods performance in predicting the selling price of houses based on a number of features such as the area, the number of bed- and bathrooms and the geographical position.

Machine learning algorithms :

In machine learning algorithms were compared against each other in order to investigate which one is more successful in predicting housing prices. As mentioned in the previous section, Baldominos et al. performed a similar study in which they compare four machine learning algorithms for housing prices. In their study they found that the Random forest regression algorithm predicted with the smallest error followed by k-Nearest neighbours regression. In the study performed by Oxenstierna k-Nearest neighbours regression and Artificial neural networks are suggested as methods for predicting house prices. Even though Oxenstierna finds the Artificial neural networks to perform better in many cases this study has excluded Artificial neural networks in order to limit the scope of the report and the time frame for the project. Since both reports study the performance of k-Nearest neighbours regression, the algorithm will be studied in this report as well. Since Baldominos et al. finds that Random forest regression gives the least error for predicting house prices and Oxenstierna mentions it as relevant for future work it will also be part of the study and compared against the k-Nearest neighbours regression algorithm.

**Training**:

Training data consists of 1,460 examples of houses with 79 features describing every aspect of the house. We are given sale prices (labels) for each house. The training data is what we will use to “teach” our models.

**Testing**:

The test data set consists of 1,459 examples with the same number of features as the training data. Our test data set excludes the sale price because this is what we are trying to predict. Once our models have been built we will run the best one the test data and submit it to the Kaggle leaderboard.

**Task**:

Machine learning tasks are usually split into three categories; supervised, unsupervised and reinforcement. For this competition, our task is supervised learning.

**Features :**

|  |  |  |
| --- | --- | --- |
| **1** | Id | To count the records. |
| **2** | MSSubClass | Identifies the type of dwelling involved in the sale. |
| **3** | MSZoning | Identifies the general zoning classification of the sale. |
| **4** | LotArea | Lot size in square feet. |
| **5** | LotConfig | Configuration of the lot |
| **6** | BldgType | Type of dwelling |
| **7** | OverallCond | Rates the overall condition of the house |
| **8** | YearBuilt | Original construction year |
| **9** | YearRemodAdd | Remodel date (same as construction date if no remodeling or additions). |
| **10** | Exterior1st | Exterior covering on house |
| **11** | BsmtFinSF2 | Type 2 finished square feet. |
| **12** | TotalBsmtSF | Total square feet of basement area |
| **13** | SalePrice | To be predicted |

### **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.

**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")

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>

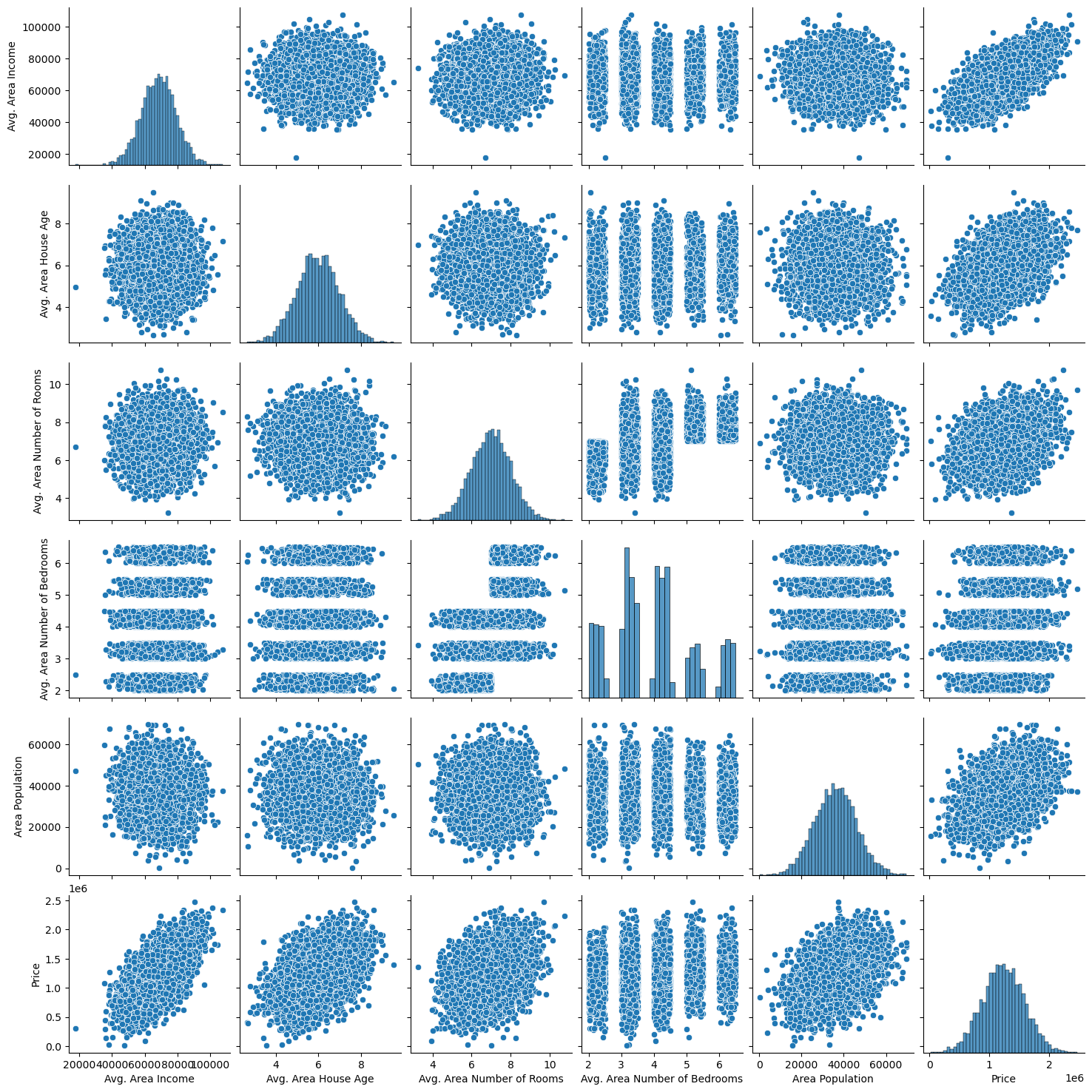


<seaborn.axisgrid.JointGrid at 0x7dbe1333c250>



<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'}>],

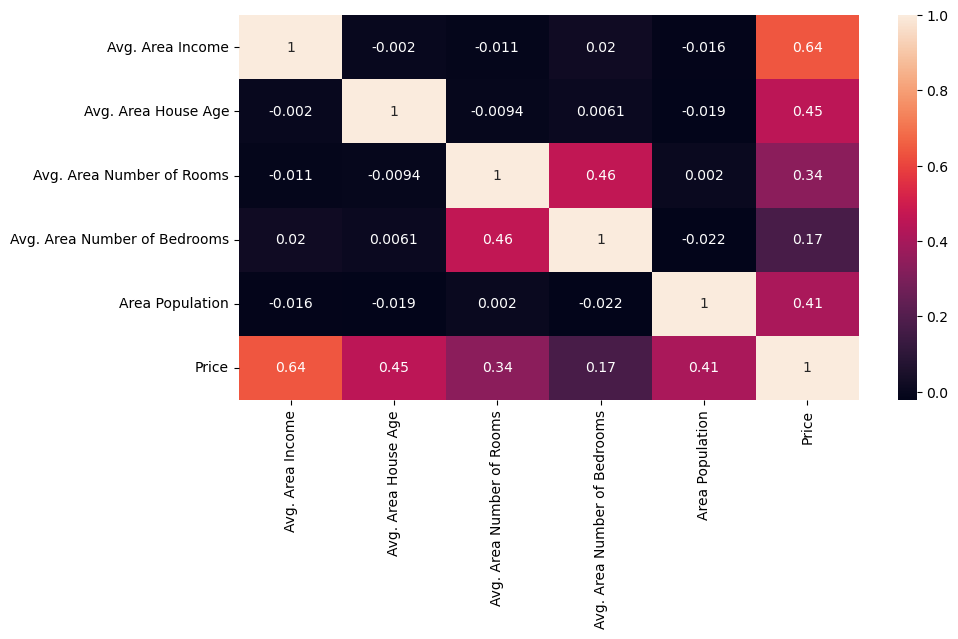
[<Axes: title={'center': 'Area Population'}>,

<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

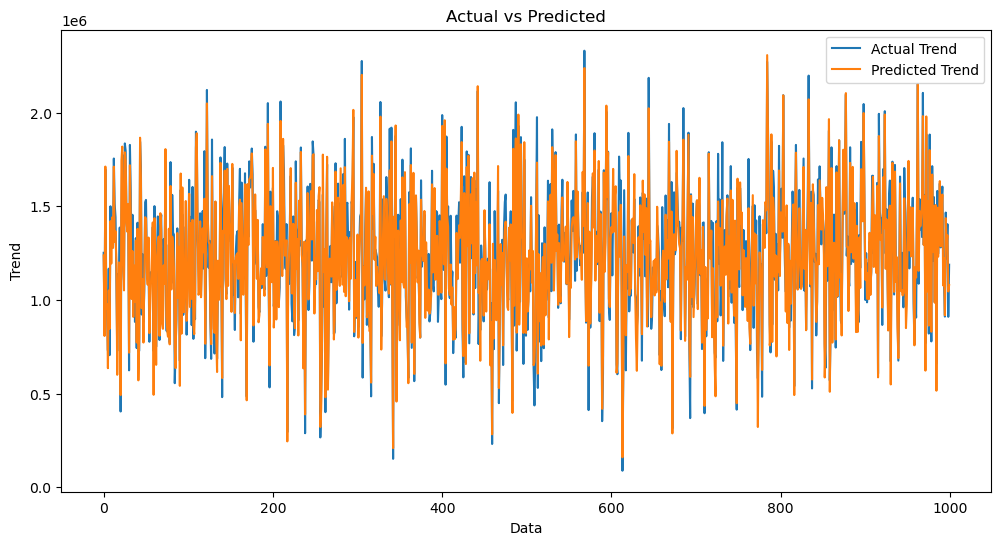
Name: Price, dtype: float64

(1000,)

LinearRegression

LinearRegression()

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



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



SVR

SVR()

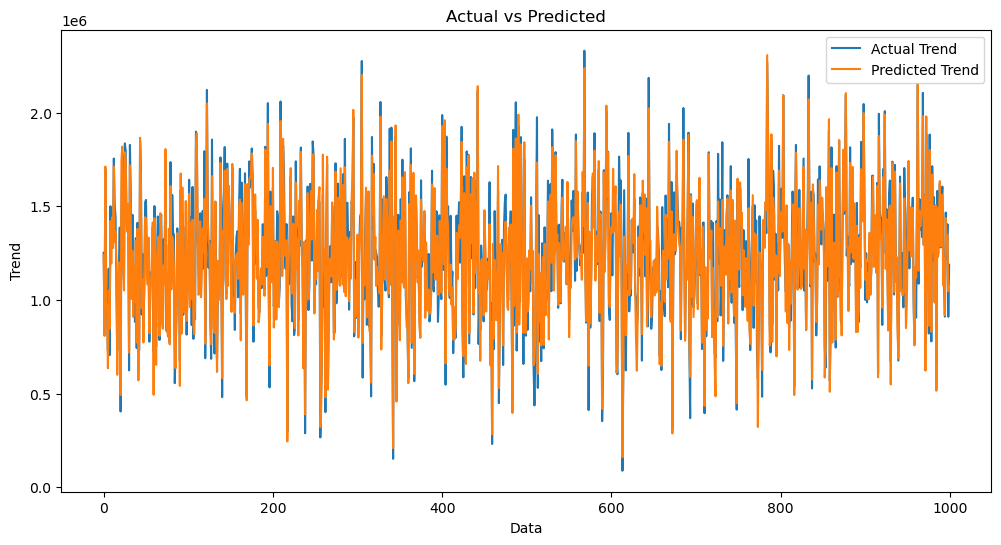
Text(0.5, 1.0, 'Actual vs Predicted')

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

Lasso

Lasso(alpha=1)

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



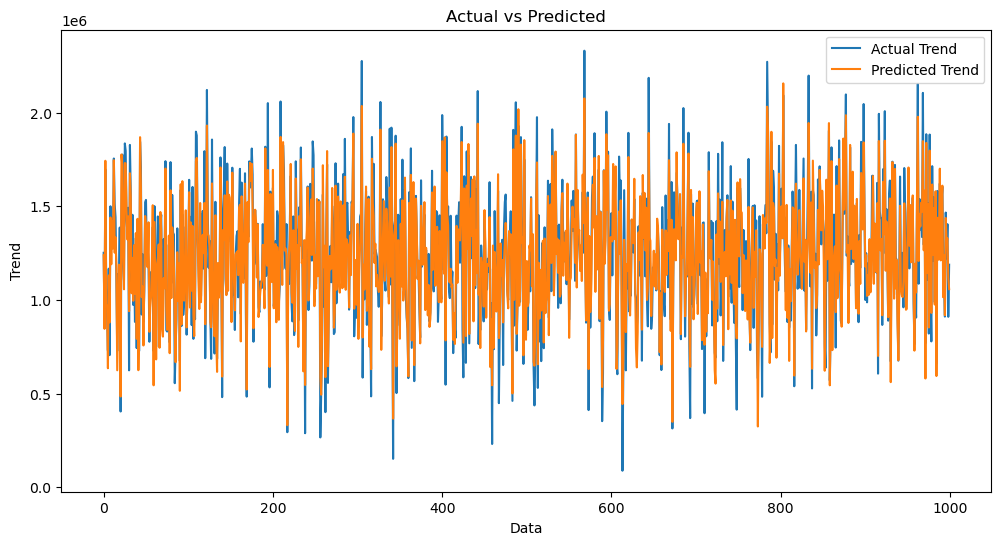
<Axes: xlabel='Price', ylabel='Count'>



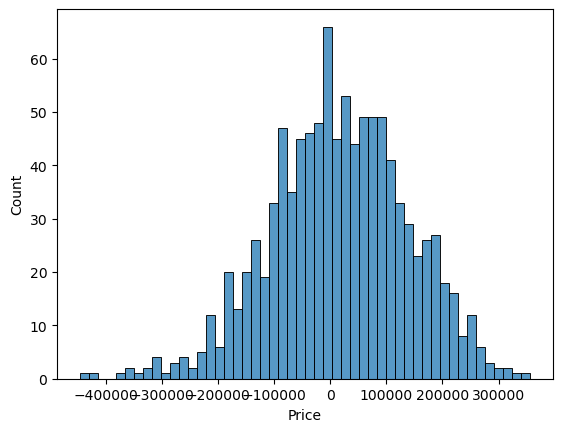
RandomForestRegressor

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,

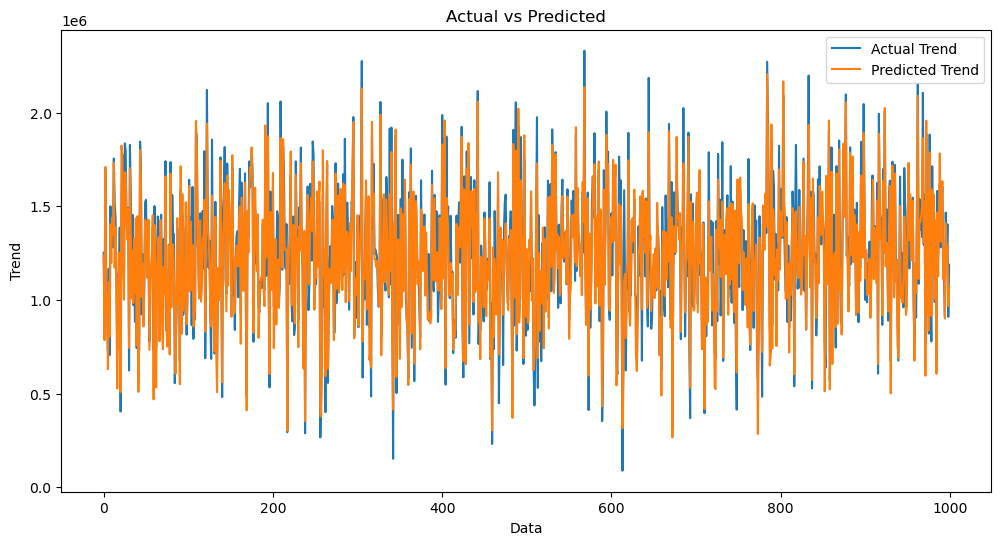
max\_delta\_step=None, max\_depth=None, max\_leaves=None,

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

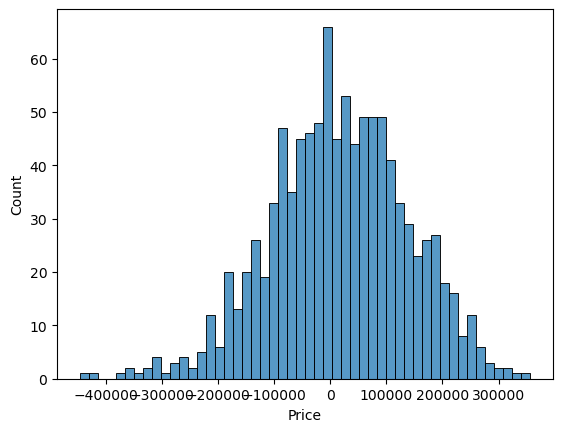
n\_estimators=100, n\_jobs=None, num\_parallel\_tree=None,

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

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



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



Conclusion :

 Thus, the machine learning model using linear regression algorithm is very helpful in predicting the house prices for real estate customers. Here we have used a supervised learning approach in machine learning field which will yield us a best possible result.