**PREDICTION OF HOUSE PRICES USING MACHINE LEARNING**

**Introduction to Prediction of House Prices:**

* Predicting house prices using machine learning involves using algorithms and data to estimate the market value of a property based on its features, such as location, size, and amenities. By analyzing these variables, these models can identify patterns and relationships that are difficult to discern through traditional methods, leading to more precise and data-informed predictions.
* Key steps include data pre-processing, feature engineering, model selection, and rigorous evaluation using metrics like Mean Absolute Error (MAE) or Mean Squared Error (MSE). The process usually involves data collection, pre-processing, feature engineering, model selection, and training. Various algorithms, including linear regression, decision trees, random forests, and neural networks, are commonly employed to develop these predictive models. The models performance is then assessed using metrics Root Mean Squared Error (RMSE) and R-Squared to ensure accuracy and reliability.
* Ensuring model interpretability and addressing ethical considerations, such as bias, are essential. Deployment in a user-friendly interface and continuous monitoring for updates and market trends are crucial for maintaining accuracy in real-world applications.
* Predicting house prices with machine learning holds significant benefits, including improved pricing strategies for sellers, informed decisions for buyers, and enhanced market analysis for real estate professionals. It also plays a crucial role in the automation and optimization of property valuation, making it a promising field with practical applications in the housing market.

**GIVEN DATASETS:**

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

|  | 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 | 23086.800503 | 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+06 | 37778 George Ridges Apt. 509\nEast Holly, NV 2... |

**TOOLS USED:**

1. Python: Python is the primary programming language for machine learning in this domain due to its extensive libraries and data analysis capabilities.

2. Scikit- Learn: Scikit-Learn is a powerful Python library for machine learning that provides a wide range of tools for building and evaluating regression models, making it suitable for house price prediction.

3. Pandas: Pandas is used for data manipulation and analysis, helping you clean and preprocess housing data effectively.

4. NumPy: NumPy is crucial for numerical computations and working with arrays, which are fundamental in machine learning.

5. Matplotlib and Seaborn: These libraries are used for data visualization, allowing you to create insightful plots and graphs to better understand the data.

6. XGBoost, LightGBM, or CatBoost: These are gradient boosting frameworks that can be applied for regression tasks, offering highly accurate predictions.

7. TensorFlow and PyTorch: These deep learning frameworks are suitable for more complex models, such as neural networks, when dealing with large datasets and intricate patterns.

8. Jupyter Notebook: Jupyter Notebook is a popular tool for interactive data exploration and model development. It allows you to document and visualize your work step by step.

9. GitHub: For version control and collaboration, GitHub is invaluable when working on machine learning projects with a team.

10. SQL Databases: Storing and managing large datasets efficiently is essential, so databases like PostgreSQL or SQLite can be used to store and query housing data.

**DESIGN THINKING:**

np\_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

importSVR

import xgboost as xg

**BUILDING AND PREPROCESSING THE DATASET:**

**INPUT:**

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

**OUTPUT:**

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

**INPUT:**

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

**OUTPUT:**

<Axes: xlabel='Price'>



**INPUT:**

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

**OUTPUT:**

<seaborn.axisgrid.JointGrid at 0x7dbe246100a0>



**INPUT :**

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

**OUTPUT:**

<seaborn.axisgrid.JointGrid at 0x7dbe1333c250>



**INPUT:**

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

sns.pairplot(dataset)

**OUTPUT :**

<seaborn.axisgrid.PairGrid at 0x7dbe1333c340>

<Figure size 1200x800 with 0 Axes>

**INPUT :**

dataset.hist(figsize=(10,8)

**OUTPUT :**

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

**INPUT :**

dataset.corr(numeric\_only=True)

**OUTPUT :**

|  | **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 |

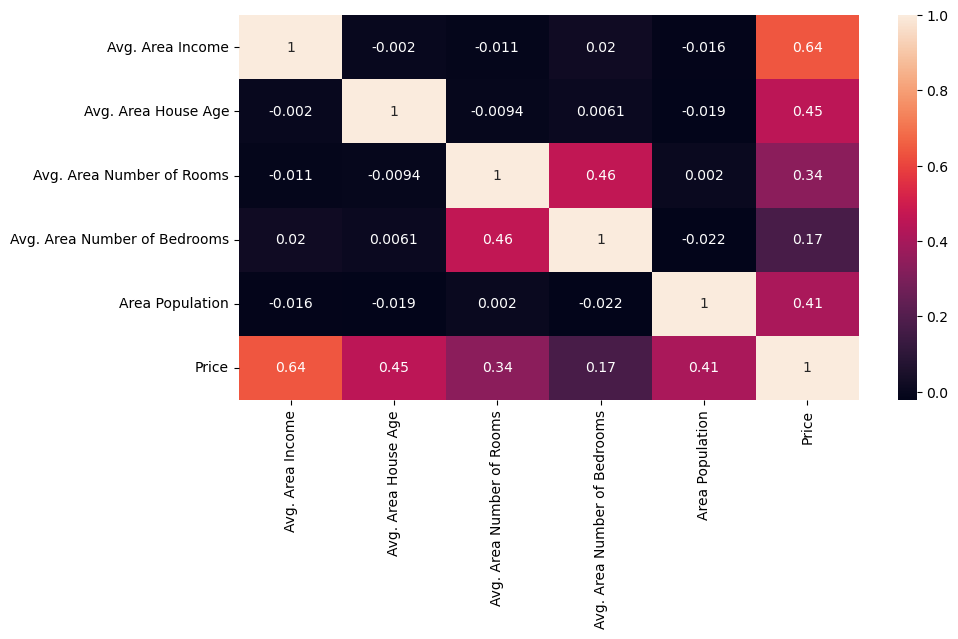
**INPUT :**

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

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

**OUTPUT :**

<Axes: >



# Dividing Dataset in to features and target variable

**INPUT:**

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

**INPUT:**

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

**INPUT:**

Y\_train.head()

**OUTPUT:**

3413 1.305210e+06

1610 1.400961e+06

3459 1.048640e+06

4293 1.231157e+06

1039 1.391233e+06

Name: Price, dtype: float64

**INPUT:**

Y\_train.shape

**OUTPUT:**

(4000)

**INPUT:**

Y\_test.head()

**OUTPUT:**

1718 1.251689e+06

2511 8.730483e+05

345 1.696978e+06

2521 1.063964e+06

54 9.487883e+05

Name: Price, dtype: float64

**INPUT:**

Y\_test.shape

**OUTPUT:**

(1000)

# Standardizing the data

**INPUT:**

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

**INPUT:**

model\_lr=LinearRegression()

**INPUT:**

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

**OUTPUT:**

LinearRegression

LinearRegression()

## Predicting Prices

**INPUT :**

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

## Evaluation of Predicted Data

**INPUT:**

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')

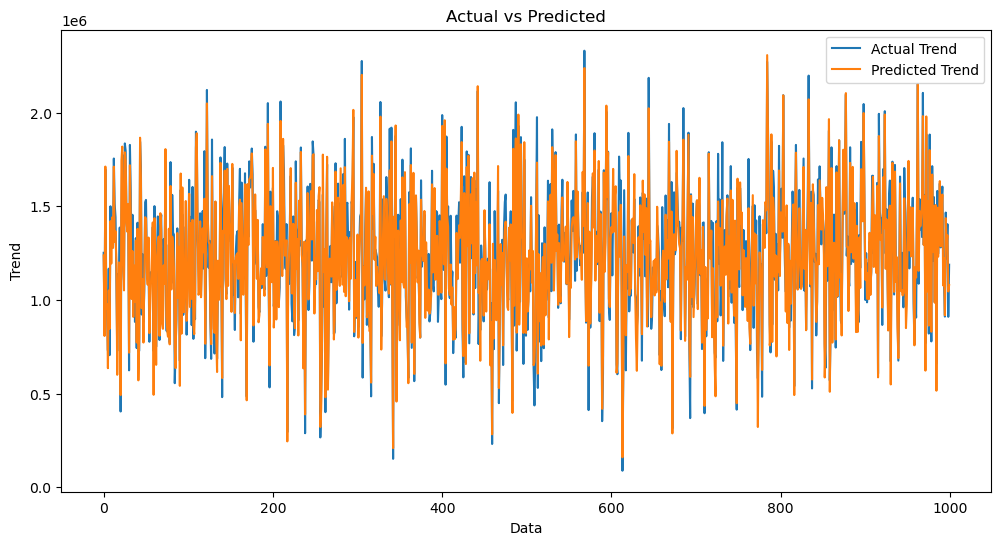
plt.ylabel('Trend')

plt.legend()

plt.title('Actual vs Predicted')

**OUTPUT:**

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



**INPUT:**

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

**OUTPUT:**

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



**INPUT:**

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 - Support Vector Regressor

**INPUT:**

model\_svr=SVR()

**INPUT:**

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

**OUTPUT:**

SVR

SVR()

## Predicting Prices

**INPUT:**

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

## Evaluation of Predicted Data

**INPUT:**

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')

**OUTPUT:**

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



**INPUT:**

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

**OUTPUT:**

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



**INPUT:**

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 - Lasso Regression

**INPUT:**

model\_lar=Lasso(alpha=1)

**INPUT:**

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

**OUTPUT:**

Lasso

Lasso(alpha=1)

## Predicting Prices

**INPUT:**

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

## Evaluation of Predicted Data

**INPUT:**

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')

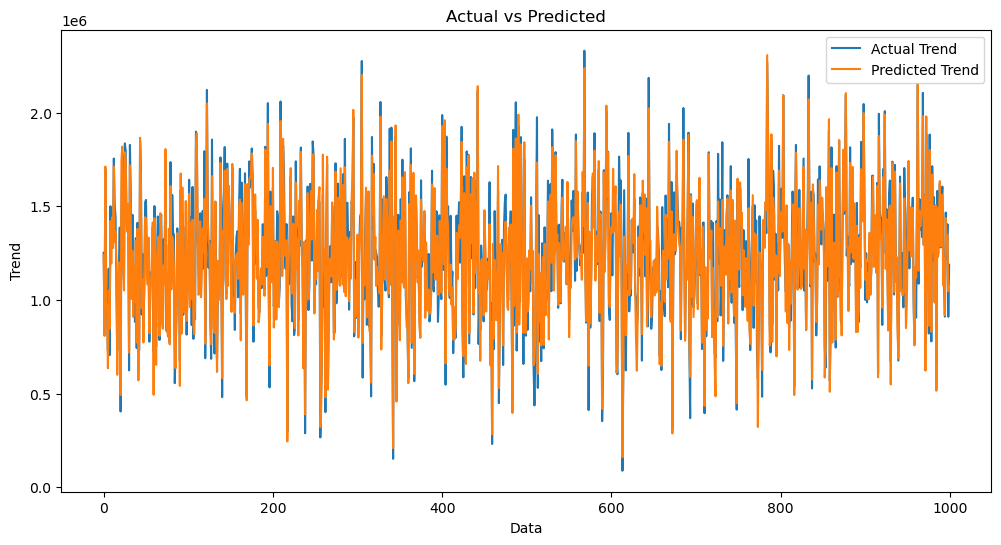
plt.ylabel('Trend')

plt.legend()

plt.title('Actual vs Predicted')

**OUTPUT:**

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



**INPUT:**

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

**OUTPUT:**

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



**INPUT:**

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 4 - Random Forest Regressor

**INPUT:**

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

**INPUT :**

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

**OUTPUT :**

RandomForestRegressor

RandomForestRegressor(n\_estimators=50)

## Predicting Prices

**INPUT :**

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

## Evaluation of Predicted Data

**INPUT:**

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')

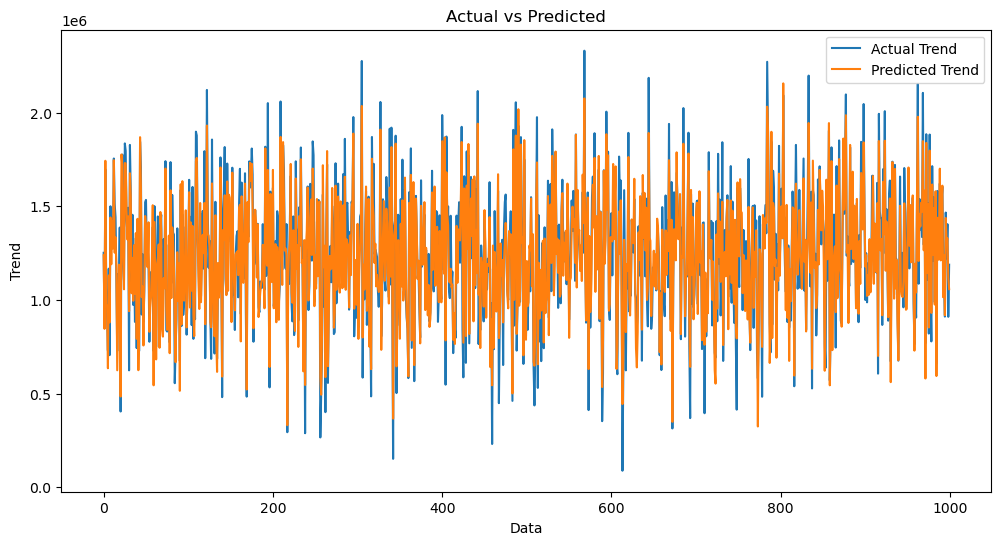
plt.ylabel('Trend')

plt.legend()

plt.title('Actual vs Predicted')

**OUTPUT:**

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

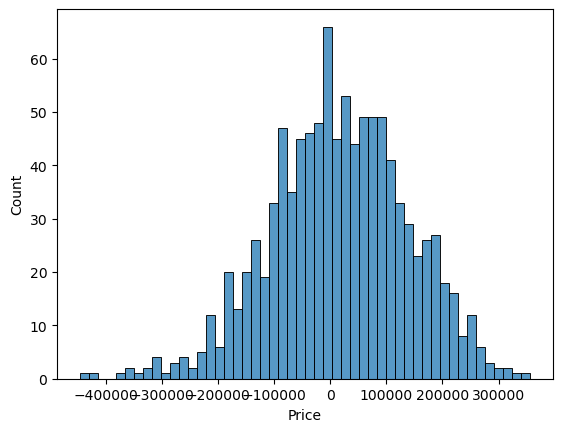


**INPUT:**

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

**OUTPUT:**

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



**INPUT :**

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 5 - XGboostRegressor

**INPUT:**

model\_xg=xg.XGBRegressor()

**INPUT:**

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

**OUTPUT :**

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

**INPUT:**

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

## Evaluation of Predicted Data

**INPUT:**

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')

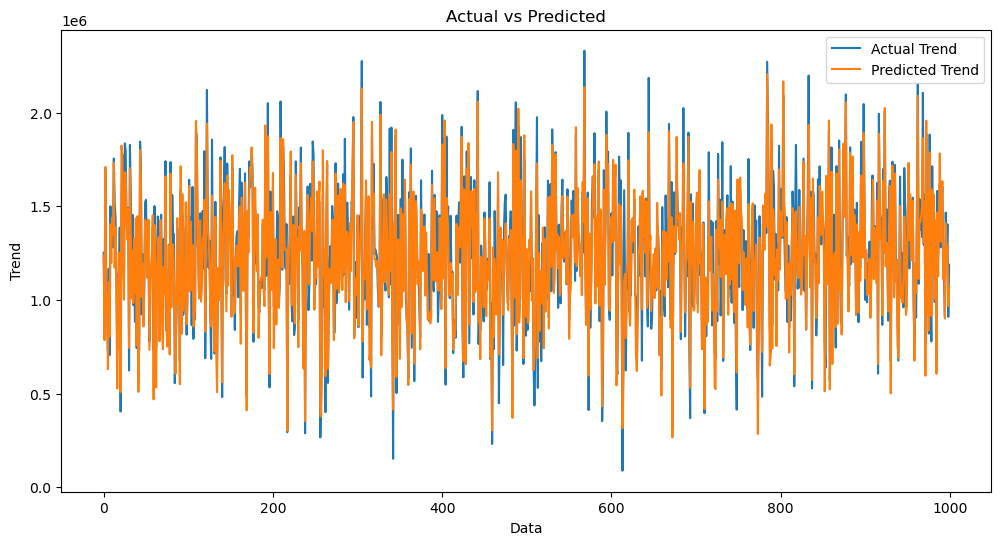
plt.ylabel('Trend')

plt.legend()

plt.title('Actual vs Predicted')

**OUTPUT:**

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

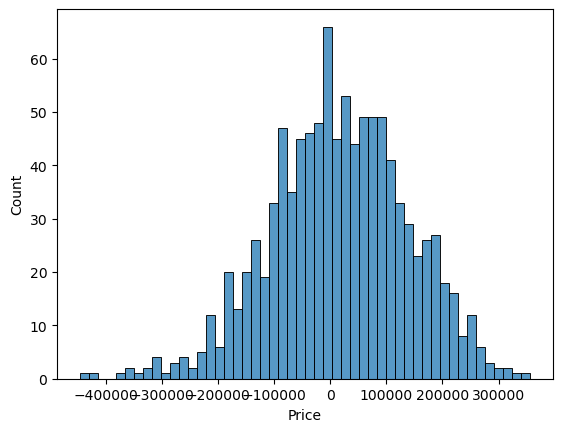


**INPUT :**

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

**OUTPUT :**

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



**ADVANTAGES:**

1. Accuracy: Machine learning models can provide more precise and data-driven estimates of house prices, reducing the potential for overpricing or underpricing properties.

2. Data-Driven Insights: These models leverage a wide range of data sources, allowing for a deeper understanding of the factors influencing property values, which can be valuable for market analysis and decision-making.

3. Automation: Machine learning automates the process of valuing properties, saving time and reducing the need for manual appraisal, especially in high-volume real estate markets.

4. Improved Decision-Making: Buyers and sellers can make more informed decisions about pricing, buying, or selling their homes, resulting in better outcomes for all parties involved.

5. Market Transparency: Predictive models contribute to greater transparency in the real estate market by providing standardized and data-based valuation methods.

6. Risk Mitigation: Lenders and financial institutions can use machine learning to assess the risk associated with mortgage lending, helping to reduce defaults and improve the stability of the housing market.

7. Competitive Advantage: Real estate professionals and agencies can gain a competitive edge by offering clients accurate and data-backed pricing advice.

8. Personalization: Machine learning can tailor house price predictions to specific markets and regions, taking into account local dynamics and trends.

9. Scalability: These models can handle large datasets and are adaptable to various property types, making them suitable for both individual homes and large real estate portfolios.

10. Continuous Learning: Machine learning models can continuously update and adapt to changing market conditions, ensuring that predictions remain relevant over time.

**DISADVANTAGES:**

1. Data Quality and Availability: Machine learning models heavily depend on the quality and quantity of data. Inaccurate, incomplete, or biased data can lead to incorrect predictions.

2. Model Complexity: More complex machine learning models may be difficult to interpret, making it challenging to understand the factors contributing to a particular prediction.

3. Overfitting: Complex models can overfit the training data, leading to poor generalization to new, unseen data. This can result in inaccurate predictions, especially when market conditions change.

4. Limited Historical Data: In some regions or markets, historical data may be limited, making it difficult for machine learning models to capture long-term trends accurately.

5. Data Privacy: Accessing and handling real estate data may raise privacy and security concerns, especially when dealing with sensitive information about property owners and buyers.

6. Influence of Outliers: Extreme outliers or anomalies in the data can disproportionately affect the model's predictions, leading to inaccuracies.

7. Market Dynamics: Real estate markets are influenced by various complex and often non-linear factors that can be challenging for machine learning models to capture.

8. Model Maintenance: Machine learning models require ongoing maintenance to adapt to changing market conditions and data sources. Failure to update models can result in outdated predictions.

9. Lack of Local Context: Machine learning models may not always consider local nuances and cultural factors that influence property values, leading to inaccuracies in specific regions.

10. Human Expertise: While machine learning can assist in pricing, human expertise is still essential for understanding local markets, unique property features, and customer preferences.

**CONCLUSION:**

* In conclusion, the prediction of house prices using machine learning represents a valuable and transformative approach in the real estate industry. While it offers significant advantages, including improved accuracy, data-driven insights, automation, and informed decision-making, it is not without its drawbacks and challenges.
* The success of machine learning in predicting house prices depends on the quality and quantity of data, the choice of appropriate models, and the understanding of local market dynamics. Additionally, the ethical and regulatory considerations, as well as the need for human expertise, remain essential components of this technology's deployment in real estate.
* Ultimately, machine learning can enhance the real estate market by providing more transparent, efficient, and data-informed pricing solutions. However, it should be used as a complementary tool alongside human judgment, local knowledge, and regulatory compliance. The responsible use of machine learning in house price prediction can benefit all stakeholders, from buyers and sellers to real estate professionals and financial institutions, while contributing to a fair and dynamic housing market.