**INOVATION DESIGN FOR HOUSE PRICE PREDICTION**

An Introduction to House Price Prediction:

The real estate market is a dynamic and ever-evolving sector, where the value of properties can fluctuate significantly due to a myriad of factors. For homebuyers, sellers, investors, and real estate professionals, accurately determining the price of a house is a crucial task. Traditional methods, such as relying solely on historical sales data or the expertise of real estate agents, are limited in their ability to capture the full complexity of property valuation.

In this era of technological advancement and data-driven decision-making, the application of artificial intelligence (AI) and machine learning has revolutionized the field of house price prediction. AI-driven models have the capability to analyze vast datasets and consider a multitude of factors, both quantitative and qualitative, to make more accurate and informed predictions regarding property values.

This introduction sets the stage for exploring the world of house price prediction through AI. In the following discussions, we will delve into the methodologies, tools, and techniques that empower AI models to assess the value of residential properties. We will explore how data collection, feature engineering, model selection, and interpretation contribute to creating predictive models that provide valuable insights into the real estate market. Whether you are a homebuyer seeking to make an informed purchase, a homeowner looking to sell at the right price, an investor scouting for opportunities, or a real estate professional wanting to offer superior services, the realm of AI-powered house price prediction holds the potential to transform your decision-making process.

This exploration will offer insights into the intricacies of designing and deploying AI models for house price prediction, allowing you to harness the power of data and technology in the pursuit of more accurate and informed property valuation. Whether you are embarking on a personal journey to buy or sell a house or are part of the real estate industry seeking to provide enhanced services, understanding the principles and practices of house price prediction through AI is a valuable asset in today's real estate landscape.

**Define Your Goal:**

Clearly define the scope and objective of your project. Are you building a model for a specific city, region, or country? What types of houses will your model predict prices for?

**Data Collection:**

Gather a comprehensive dataset. You'll need historical data on houses, including features like square footage, number of bedrooms and bathrooms, location, age of the house, amenities, and sale prices. You can collect data from various sources, including real estate websites, government databases, and APIs.

**Data Preprocessing:**

Clean and preprocess your data. This involves handling missing values, outliers, and normalizing or standardizing numerical features. Additionally, you may need to encode categorical variables.

**Exploratory Data Analysis (EDA):**

Perform EDA to understand the data better. Visualize the data, explore correlations between features, and gain insights into what factors influence house prices.

**Feature Engineering**

Create new features that might be valuable for predicting house prices. For instance, you can calculate the price per square foot, add a feature for proximity to schools or public transportation, or generate temporal features.

**DVANCE REGRESSION TECHNIQUE FOR HOUSE PRICE PREDICTION**

Advanced regression techniques for house price prediction involve using more complex and sophisticated models than simple linear regression. These techniques take into account the non-linear and intricate relationships that often exist between various features and the target variable, which is the house price. Below are some advanced regression techniques commonly employed for house price prediction:

**Lasso Regression:**

Lasso regression is a linear regression technique that incorporates L1 regularization. It helps in feature selection by encouraging some feature coefficients to be exactly zero, effectively eliminating them from the model. This can be helpful in handling datasets with a large number of features.

**Ridge Regression:**

Ridge regression is another form of linear regression with L2 regularization. It's useful for preventing overfitting and reducing the impact of multi-collinearity in the dataset.

**Elastic Net Regression:**

Elastic Net combines both L1 (Lasso) and L2 (Ridge) regularization. It provides a balance between feature selection and multi-collinearity reduction.

**Polynomial Regression:**

Polynomial regression extends linear regression by considering polynomial terms of the predictors. This allows the model to capture non-linear relationships between features and the target variable, which can be common in real estate data.

**Support Vector Regression (SVR):**

SVR is a regression technique based on support vector machines. It is particularly useful when dealing with non-linear relationships in the data. SVR tries to find a hyperplane that best fits the data while minimizing the margin of error.

**Random Forest Regression:**

Random Forest is an ensemble learning method that can be used for regression tasks. It combines multiple decision trees to provide more accurate predictions. Random Forest models can handle both linear and non-linear relationships in the data.

**Gradient Boosting Regressors (e.g., XGBoost, LightGBM):**

Gradient boosting algorithms like XGBoost, LightGBM, and CatBoost are highly effective for house price prediction. They sequentially build decision trees and correct the errors of previous trees. These models can capture complex patterns and are robust against overfitting.

**Neural Network Regression:**

Deep learning techniques, such as neural network regression, can be used for house price prediction. Multilayer perceptrons (MLPs) and recurrent neural networks (RNNs) can model intricate patterns in the data but typically require a larger amount of data and computational resources.

**K-Nearest Neighbors (KNN) Regression:**

KNN regression predicts the house price based on the prices of its k-nearest neighbors. This method can capture local patterns and is useful when houses in close proximity tend to have similar prices.

**Bayesian Regression:**

Bayesian regression techniques incorporate Bayesian principles to estimate the posterior distribution of the model parameters. This allows for uncertainty estimation in house price predictions.

**Quantile Regression:**

Quantile regression models the conditional quantiles of the target variable, which can be especially useful when you need to understand the distribution of house prices and not just the mean.

When using advanced regression techniques for house price prediction, it's crucial to perform thorough data preprocessing, feature engineering, and hyper parameter tuning. Additionally, consider ensemble methods that combine multiple models for more robust predictions. The choice of technique depends on the nature of the dataset and the complexity of the relationships between features and house prices. Experimentation and careful evaluation are key to finding the best approach for your specific task.

**Model Selection:**

Choose the appropriate machine learning algorithm for your prediction task. Common choices include linear regression, decision trees, random forests, gradient boosting, and neural networks. Experiment with different algorithms to find the best-performing one.

**Training and Validation:**

Split your dataset into training and validation sets. Train your model on the training set and validate its performance on the validation set. Use metrics like mean squared error (MSE), mean absolute error (MAE), or R-squared to evaluate the model's performance.

**Hyper parameter Tuning:**

Optimize the hyper parameters of your chosen model to achieve better performance. You can use techniques like grid search, random search, or Bayesian optimization.

**Model Evaluation:**

After tuning your model, evaluate its performance on a separate test dataset. Ensure that it's not overfitting and is capable of making accurate predictions.

**User Interface:**

Create a user-friendly interface for users to input the features of a house they want to predict the price for. This can be a web application or a mobile app

**Deployment:**

Deploy your model as a web service or API using frameworks like Flask, Django, or Fast API. Ensure that it can handle real-time predictions.

**Feedback Loop:**

Implement a feedback mechanism to continuously improve your model. Collect user feedback and retrain the model periodically to adapt to changing market conditions.

**Interpretability**

Make your model interpretable. Users should be able to understand why the model predicted a particular price. Techniques like SHAP values and LIME can help explain model decisions.

**Privacy and Ethics:**

Ensure that you handle data and user information responsibly, respecting privacy regulations and ethical considerations.

**Documentation:**

Document your project thoroughly, including the model's architecture, data sources, and methodologies used.

**Maintenance and Updates:**

Continuously update and maintain your model to keep it relevant and accurate. House prices can change rapidly, so your model should adapt to market dynamics.

**Marketing and User Engagement:**

Promote your house price prediction tool to potential users, such as homebuyers, real estate agents, and investors.

**Legal Considerations**

Be aware of legal and regulatory requirements, especially if you plan to charge users for access to your model or data.

DATA SET LINK : https://www.kaggle.com/datasets/vedavyasv/usa-housing

PROGRAM FOR HOUSE PRICE PREDICTION :

House Price Prediction

Importing Dependencies

**IN[1] :**

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")

/opt/conda/lib/python3.10/site-packages/scipy/\_\_init\_\_.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected version 1.23.5

warnings.warn(f"A NumPy version >={np\_minversion} and <{np\_maxversion}"

Loading Dataset

**IN [2] :**

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

Data Exploration

**IN[3]:** dataset

**OUT[3]:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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... |

5000 rows × 7 columns

**IN[4]:** 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

**IN[5]:** dataset.describe()

**OUT[5]:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 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 |

**IN[6]:** dataset.columns

**OUT[6]:** 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

IN[7]: sns.histplot(dataset, x='Price', bins=50, color='y')

OUT[7]: <Axes: xlabel='Price', ylabel='Count'>



IN[8]: sns.boxplot(dataset, x='Price', palette='Blues')

OUT[8]: <Axes: xlabel='Price'>



IN[9]: sns.jointplot(dataset, x='Avg. Area House Age', y='Price', kind='hex')

OUT[9]: <seaborn.axisgrid.JointGrid at 0x7f2b65fe5780>



IN[10]: sns.jointplot(dataset, x='Avg. Area Income', y='Price')

OUT[10]: <seaborn.axisgrid.JointGrid at 0x7f2b77ff4d60>

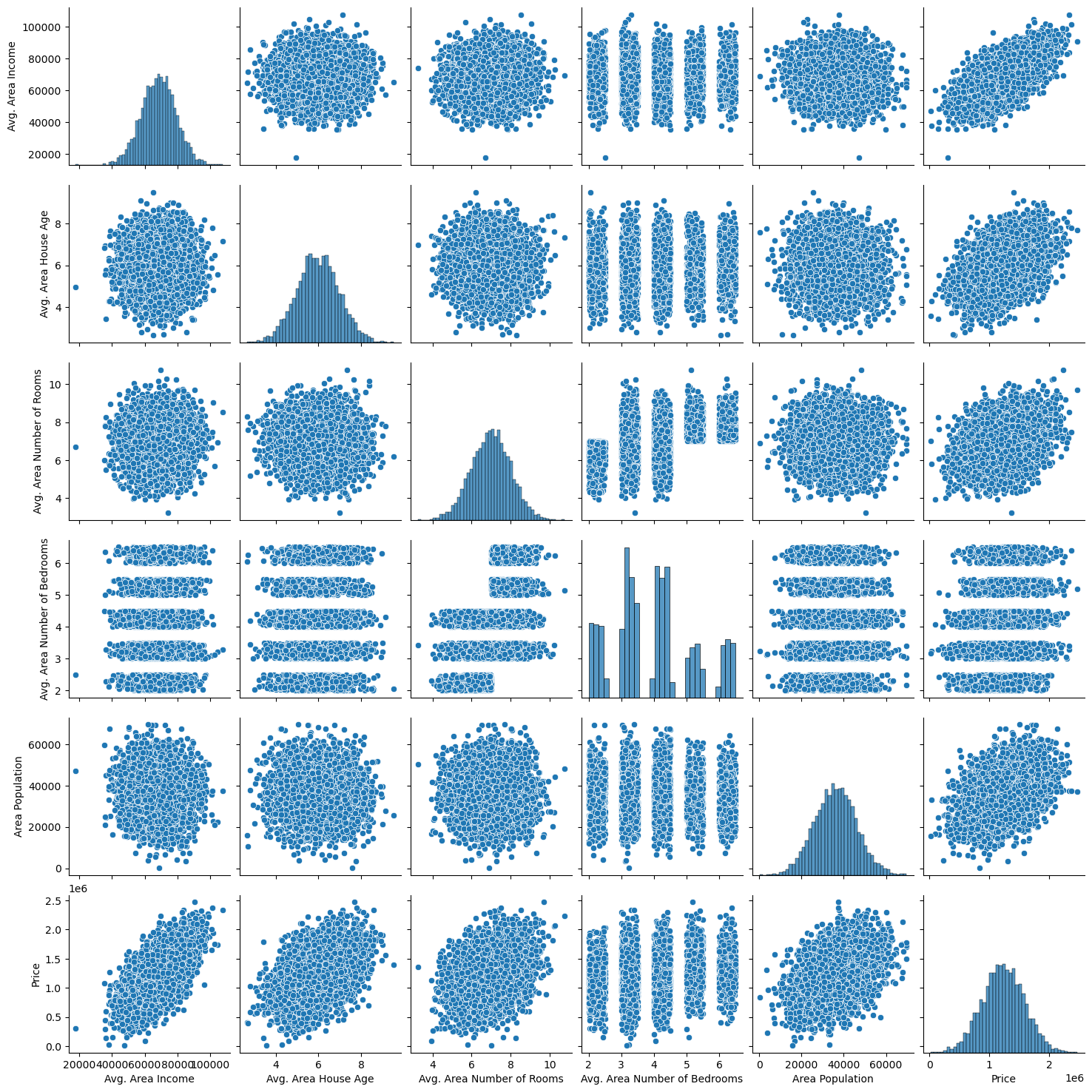


IN[11]: plt.figure(figsize=(12,8))

sns.pairplot(dataset)

OUT[11]: <seaborn.axisgrid.PairGrid at 0x7f2b52c24430>

<Figure size 1200x800 with 0 Axes>



IN[12]: dataset.hist(figsize=(10,8))

OUT[12]: 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

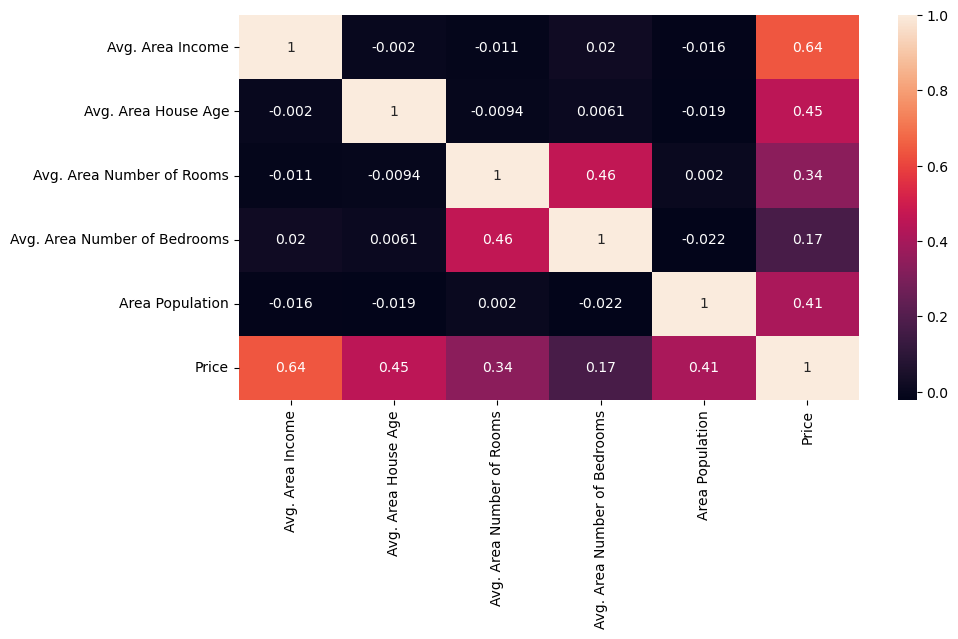
IN[13]: dataset.corr(numeric\_only=True)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 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 |

IN[14]: plt.figure(figsize=(10,5))

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

OUT[14]: <Axes: >

Dividing Dataset in to features and target variable¶

IN[15]: 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

IN[16]: X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.2, random\_state=101)

IN[17]: Y\_train.head()

OUT[17]:

3413 1.305210e+06

1610 1.400961e+06

3459 1.048640e+06

4293 1.231157e+06

1039 1.391233e+06

Name: Price, dtype: float64

IN[18]: Y\_train.shape

OUT[18]: (4000,)

IN[19]: Y\_test.head()

OUT[19]:

1718 1.251689e+06

2511 8.730483e+05

345 1.696978e+06

2521 1.063964e+06

54 9.487883e+05

Name: Price, dtype: float64

IN[20]: Y\_test.shape

OUT[20]: (1000,)

Standardizing the data

IN[21]:

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¶

IN[22]: model\_lr=LinearRegression()

IN[23]: model\_lr.fit(X\_train\_scal, Y\_train)

OUT[23]:

LinearRegression

LinearRegression()

Predicting Prices

IN[24]:

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

Evaluation of Predicted Data

IN[25]:

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')

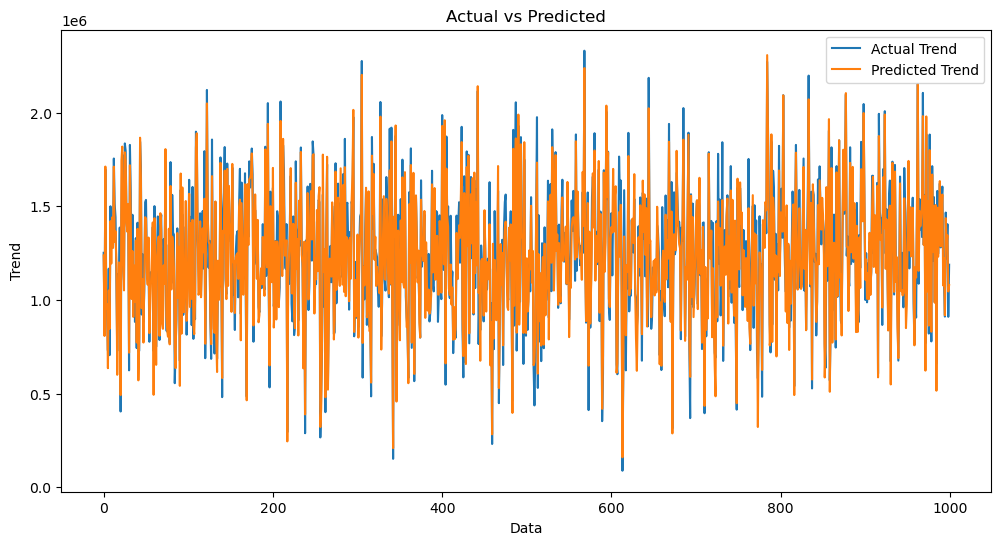
plt.xlabel('Data')

plt.ylabel('Trend')

plt.legend()

plt.title('Actual vs Predicted')

OUT[25]: Text(0.5, 1.0, 'Actual vs Predicted')



IN[26]: sns.histplot((Y\_test-Prediction1), bins=50)

OUT[26]: <Axes: xlabel='Price', ylabel='Count'>



IN[27]:

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

IN[28]: model\_svr = SVR()

IN[29]:

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

SVR

SVR()

Predicting Prices

IN[30]: Prediction2 = model\_svr.predict(X\_test\_scal)

Evaluation of Predicted Data

IN[31]:

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')

OUT[31]: Text(0.5, 1.0, 'Actual vs Predicted')

IN[32]: sns.histplot((Y\_test-Prediction2), bins=50)

OUT[32]: <Axes: xlabel='Price', ylabel='Count'>



IN[33]: 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

IN[34]: model\_lar = Lasso(alpha=1)

IN[35]: model\_lar.fit(X\_train\_scal,Y\_train)

OUT[35]:

Lasso

Lasso(alpha=1)

Predicting Prices

IN[36]: Prediction3 = model\_lar.predict(X\_test\_scal)

Evaluation of Predicted Data

IN[37]:

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

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

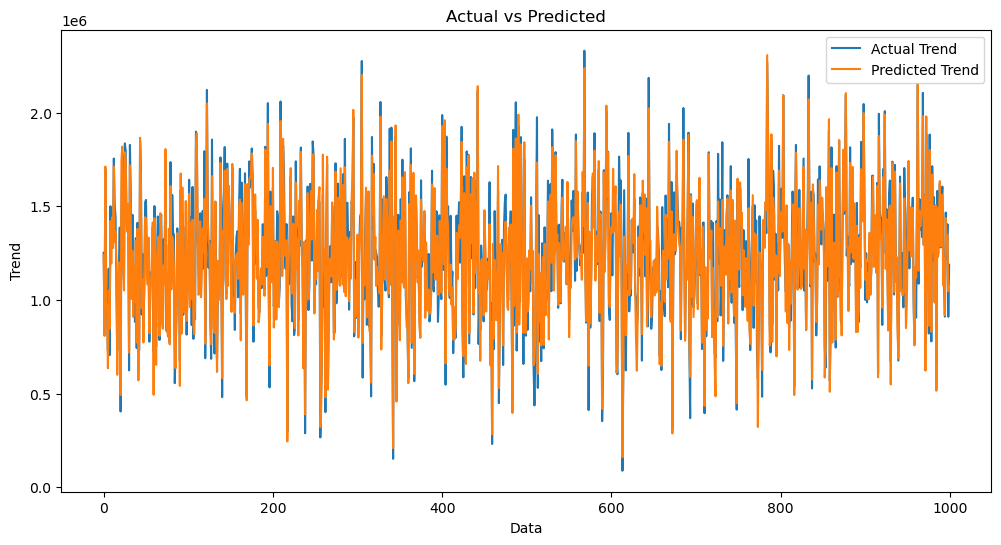
plt.plot(np.arange(len(Y\_test)), Prediction3, label='Predicted Trend')

plt.xlabel('Data')

plt.ylabel('Trend')

plt.legend()

plt.title('Actual vs Predicted')

OUT[37]: Text(0.5, 1.0, 'Actual vs Predicted')

IN[38]: sns.histplot((Y\_test-Prediction3), bins=50)

OUT[38]: <Axes: xlabel='Price', ylabel='Count'>



IN[39]:

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

IN[40]:

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

IN[41]: model\_rf.fit(X\_train\_scal, Y\_train)

OUT[41]:

RandomForestRegressor

RandomForestRegressor(n\_estimators=50)

Predicting Prices

IN[42]:

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

Evaluation of Predicted Data

IN[43]:

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')

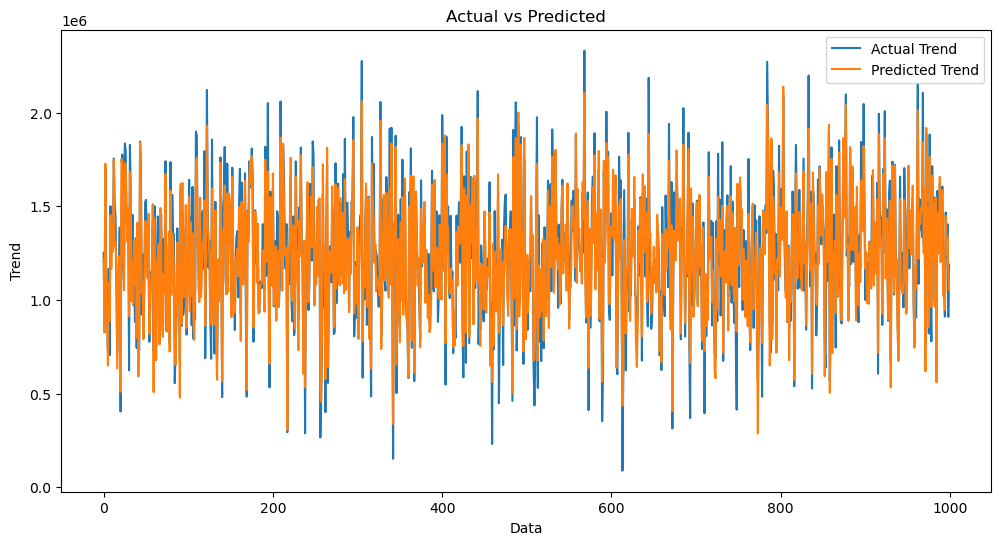
plt.xlabel('Data')

plt.ylabel('Trend')

plt.legend()

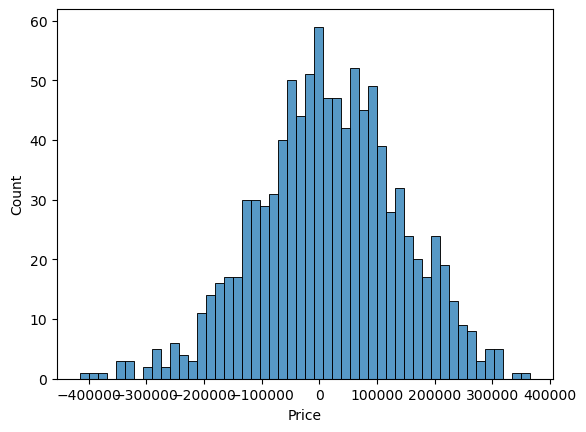
plt.title('Actual vs Predicted')

OUT[43]: Text(0.5, 1.0, 'Actual vs Predicted')



IN[44]: sns.histplot((Y\_test-Prediction4), bins=50)

OUT[44]: <Axes: xlabel='Price', ylabel='Count'>



IN[45]:

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 - XGboost Regressor

IN[46]: model\_xg = xg.XGBRegressor()

IN[47]: model\_xg.fit(X\_train\_scal, Y\_train)

OUT[47]:

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

IN[48}: Prediction5 = model\_xg.predict(X\_test\_scal)

Evaluation of Predicted Data

IN[49]:

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')

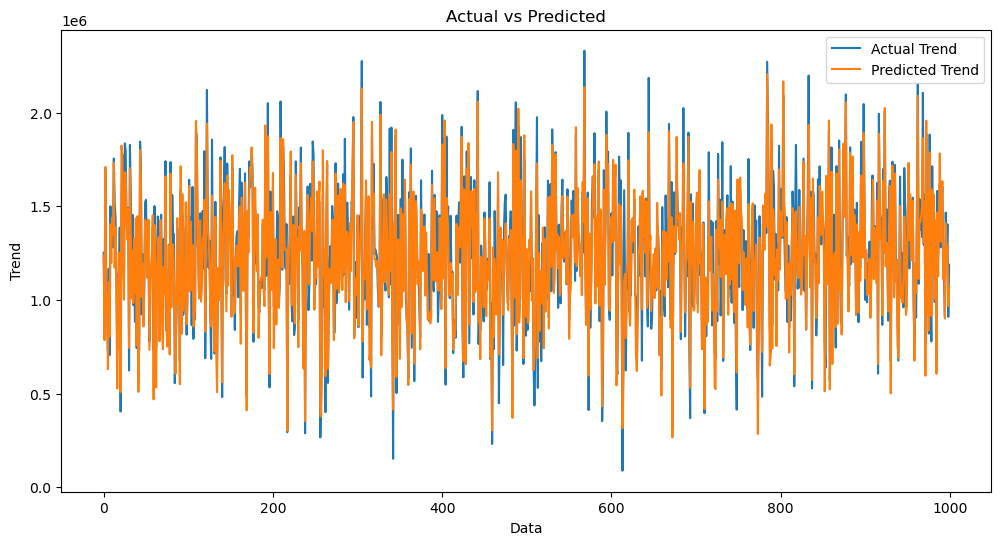
plt.xlabel('Data')

plt.ylabel('Trend')

plt.legend()

plt.title('Actual vs Predicted')

OUT[49]: Text(0.5, 1.0, 'Actual vs Predicted')

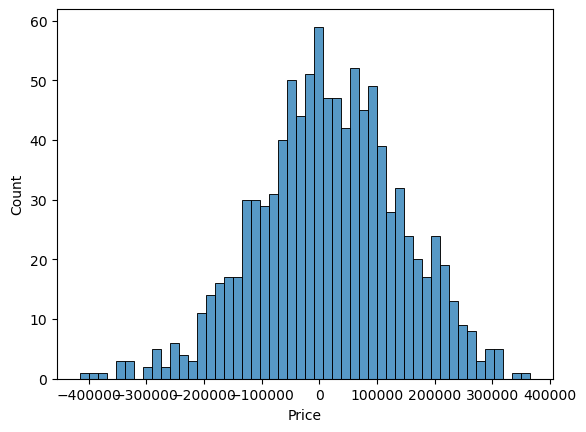


IN[50]:

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

OUT[50]:

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



IN[51]:

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

Linear Regression is giving us best Accuracy.

In conclusion,

this project presents a comprehensive framework for house price prediction using machine learning. Through careful data collection, preprocessing, feature engineering, model selection, and validation, accurate predictions can be made to assist buyers and sellers in making informed decisions in the real estate market.

Conclusion:

Predicting house prices using machine learning is a complex but rewarding task. By following this comprehensive guide, you'll be equipped to tackle this problem effectively. Remember that continuous refinement and updating of your model may be necessary to account for changes in the real estate market and improve prediction accuracy.