PREDICTING HOUSE PRICES USING MACHINE LEARNING

Introduction:

The real estate market is highly dynamic and influenced by a multitude of factors. Accurately predicting house prices is a crucial task for homeowners, buyers, and real estate professionals alike. Machine learning offers a powerful approach to tackle this problem by leveraging data-driven techniques to make precise predictions. In this project, our primary objective is to develop a robust model capable of accurately predicting house prices based on various features, such as location, square footage, number of bedrooms and bathrooms, and other relevant factors. This comprehensive guide will walk you through the entire process, from data preprocessing to model selection, training, and evaluation.

Abstract:

The rapid growth in the real estate market has made accurate house price prediction an essential tool for both buyers and sellers. This project aims to develop a robust and accurate machine learning model for predicting house prices based on various features and attributes of residential properties.

Module 1: Data Collection

Gather a diverse dataset of real estate listings, including features such as:

Location (e.g., city, neighborhood)

Square footage

Number of bedrooms and bathrooms

Lot size

Year built

Amenities (e.g., pool, garage, garden)

Historical sale prices

Module 2: Data Preprocessing

Handle missing data: Impute missing values or remove rows/columns as needed.

Encode categorical variables: Convert location data and any other categorical features into numerical representations using techniques like one-hot encoding or label encoding.

Feature scaling: Normalize or standardize numerical features to bring them to a consistent scale.

Outlier detection and treatment: Identify and address outliers that might adversely affect model performance.

Module 3: Exploratory Data Analysis (EDA)

EDA is a critical step in understanding the relationships between different features and the target variable (house prices). Visualizations and statistical analyses are used to identify patterns, correlations, and outliers in the dataset. EDA helps in making informed decisions about feature selection and engineering.

Module 4: Feature Selection and Engineering

In this module, a subset of the most relevant features is selected based on their importance and impact on house prices. Feature engineering techniques are further applied to create new meaningful features that can enhance the model's predictive power. Dimensionality reduction methods may also be considered to improve model efficiency.

Module 5: Model Selection and Training

Several machine learning algorithms, including but not limited to linear regression, decision trees, random forests, and gradient boosting, are evaluated to determine which one best fits the data. The dataset is split into training and testing sets for model training and evaluation. Hyperparameter tuning and cross-validation are used to optimize model performance.

Module 6: Model Evaluation and Validation

The selected model is rigorously evaluated using various metrics such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and R-squared (R2) to assess its predictive accuracy. Validation techniques such as k-fold cross-validation are employed to ensure the model's generalizability and to detect overfitting.

Module 7: Deployment and Future Work

Once the model is validated and deemed accurate, it can be deployed as a user-friendly application or integrated into real estate platforms for price prediction. Future work may involve continuous model updates, incorporating additional data sources, and exploring advanced machine learning techniques to further improve prediction 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.

**INOVATION DESIGN FOR HOUSE PRICE PREDICTION USING MACHINE LEARNING**

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 standardizi

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 XG Boost, Light GBM, and Cat Boost 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 per cep t r o ns (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>

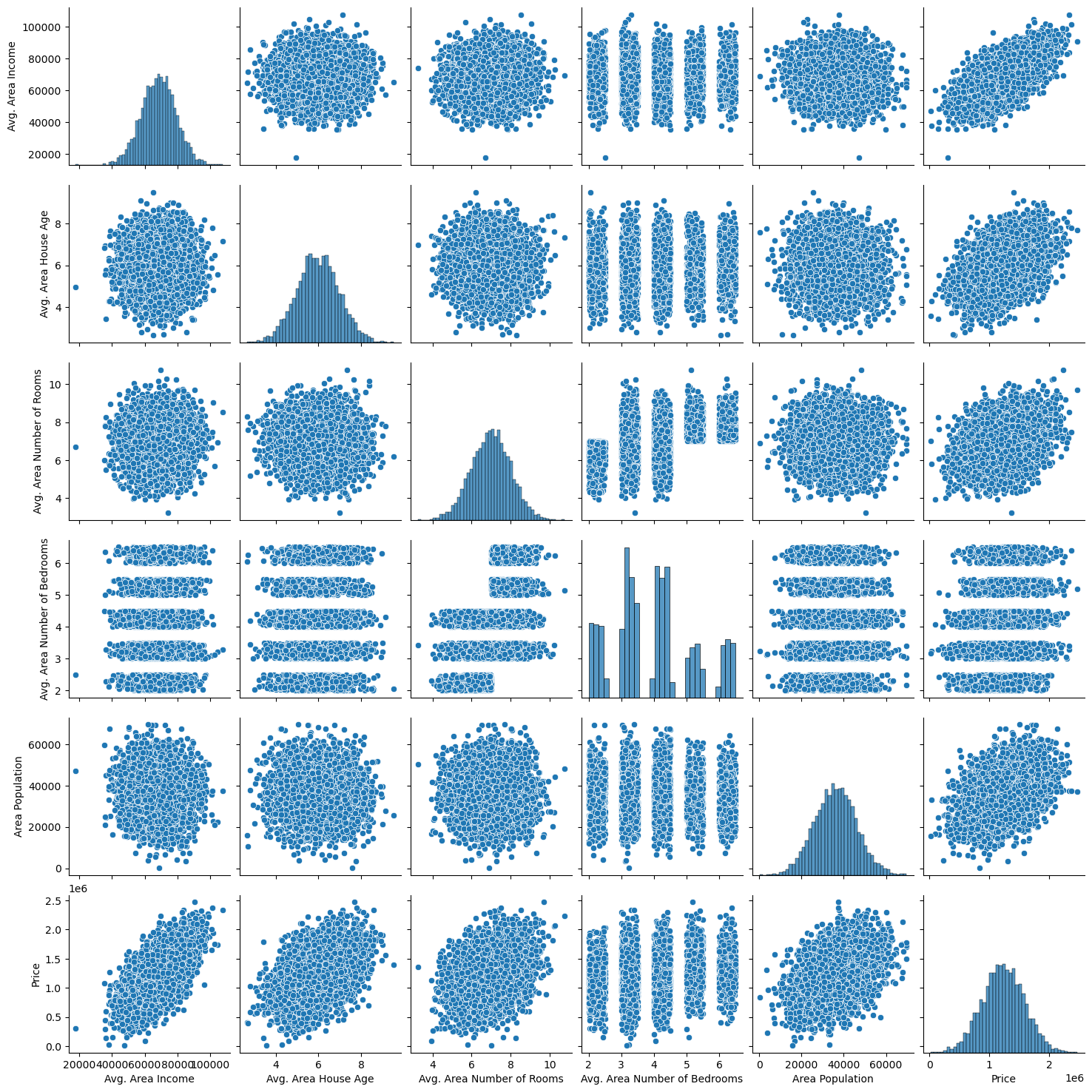


IN[11]: plt.figure(figsize=(12,8))

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

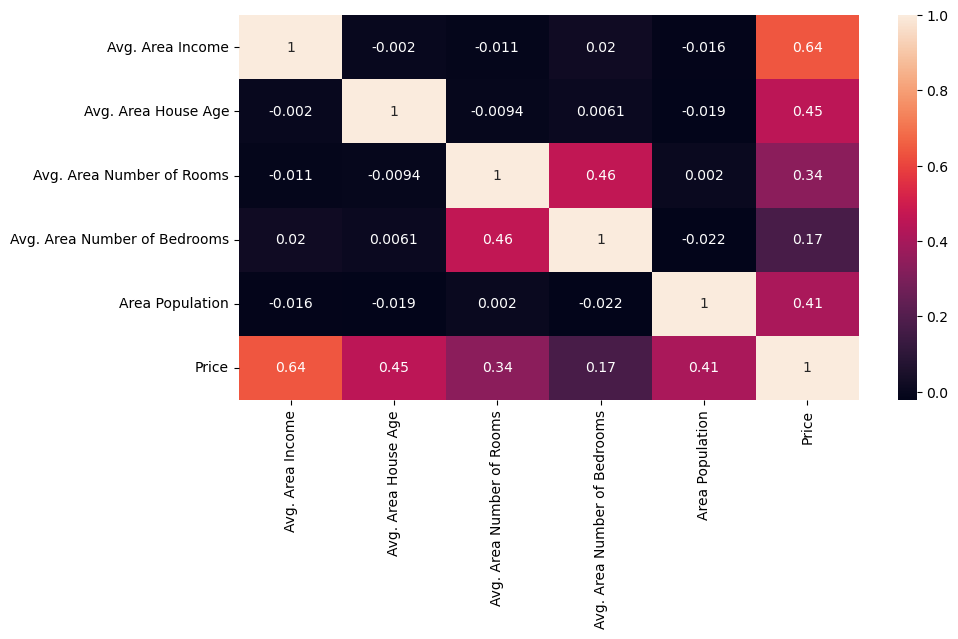
IN[13]: dataset.corr(numeric\_only=True)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 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')

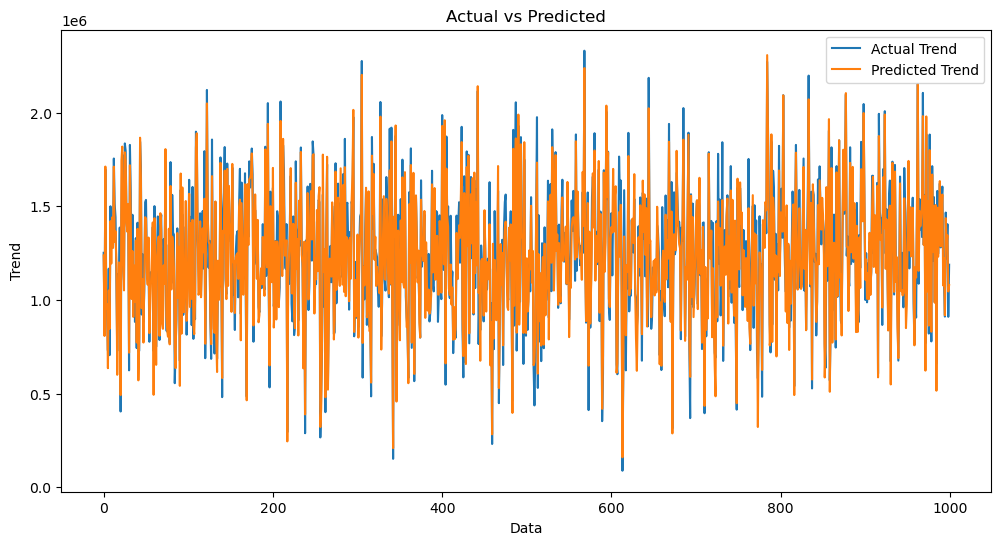
plt.xlabel('Data')

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')

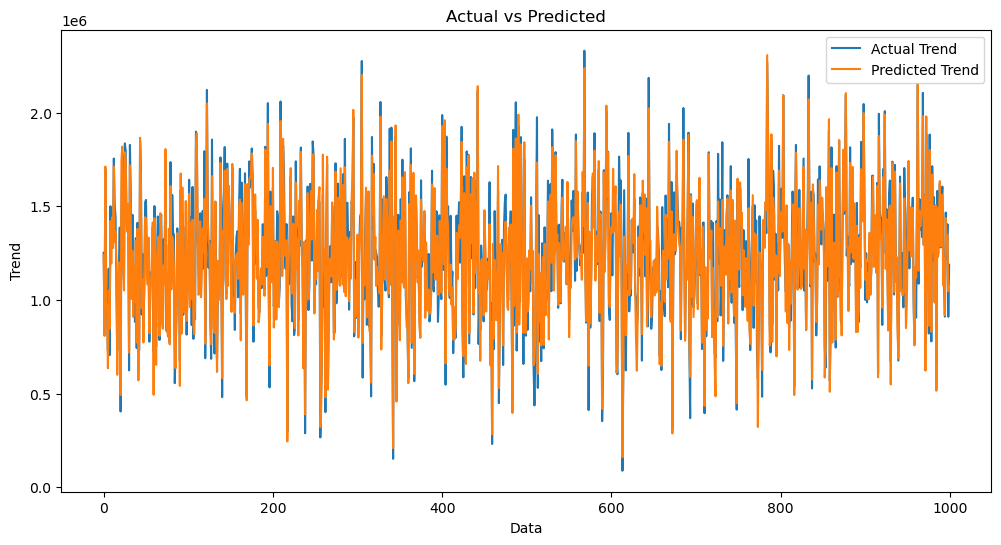
plt.plot(np.arange(len(Y\_test)), Prediction3, label='Predicted Trend')

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')

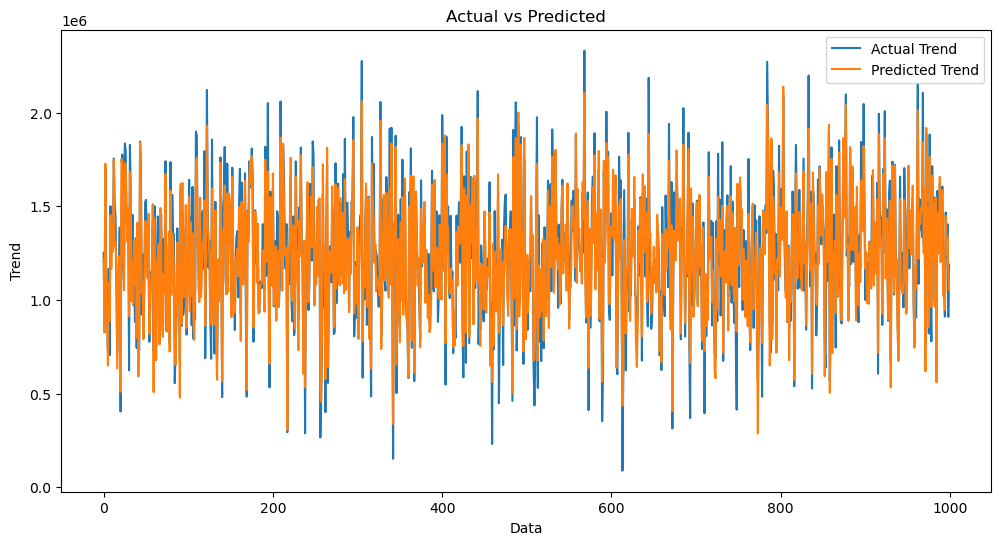
plt.xlabel('Data')

plt.ylabel('Trend')

plt.legend()

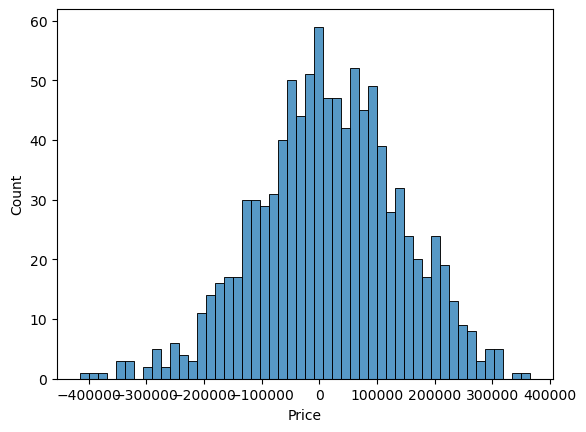
plt.title('Actual vs Predicted')

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')

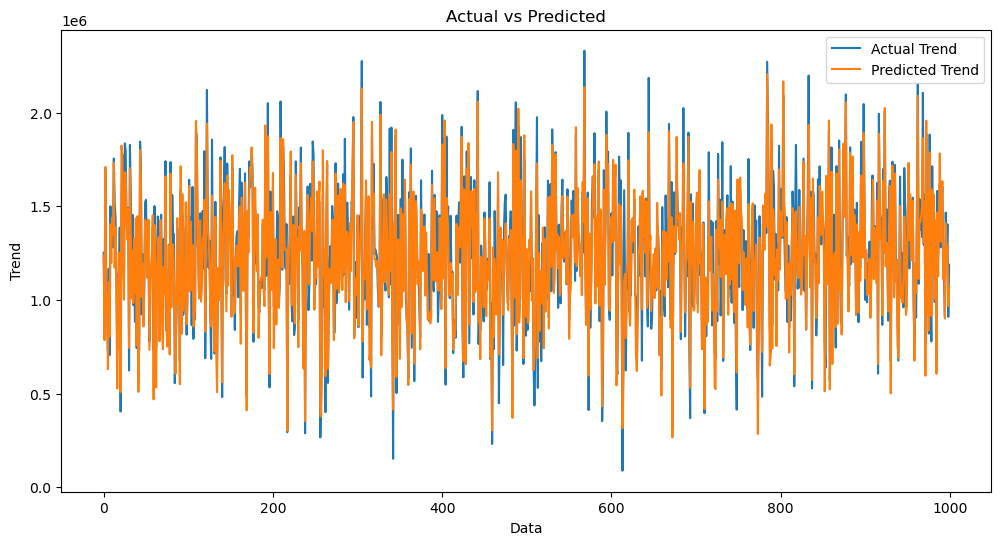
plt.xlabel('Data')

plt.ylabel('Trend')

plt.legend()

plt.title('Actual vs Predicted')

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

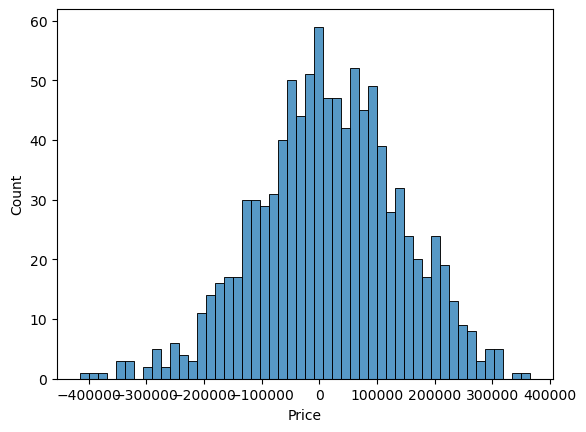


IN[50]:

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 Project Title: House Price Predictio

Topic**: Start building the house price prediction model byloading and pre-processing the dataset.**



Introduction:

* **Whether you're a homeowner looking to estimate the value of yourproperty, a real estate investor seeking profitable opportunities, or a data scientist aiming to build a predictive model, the foundation ofthis endeavor lies in loading and preprocessing the dataset.**
* **Building a house price prediction model is a data-driven process thatinvolves harnessing the power of machine learning to analyze historical housing data and make informed price predictions. This journey begins with the fundamental steps of data loading and preprocessing.**
* **This introduction will guide you through the initial steps of the process. We'll explore how to import essential libraries, load the housing dataset, and perform critical preprocessing steps. Data preprocessing is crucial as it helps clean, format, and prepare the data for further analysis. This includes handling missing values, encoding categorical variables, and ensuring that the data is appropriately scaled.**

Given Data Set :

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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 x columns

Necessary step to follow:

1.Import Libraries

Start by importing the necessary libraries:

Program:

import pandas as pd import numpy as np

from sklearn.model\_selection import train\_test\_split from sklearn.preprocessing import StandardScaler

2.Load the Dataset:

Load your dataset into a Pandas DataFrame. You can typically find house price datasets in CSV format, but you can adapt this code to other formats as needed.

Program:

Pd.read()

Program:

# Explore statistics print(df.describe())

# Visualize the data (e.g., histograms, scatter plots, etc.)

3. Exploratory Data Analysis (EDA):

Perform EDA to understand your data better. This includes

checking for missing values, exploring the data's statistics, andvisualizing it to identify patterns.

Program:

# Check for missing values

print(df.isnull().sum())

# Explore statistics

print(df.describe())

# Visualize the data (e.g., histograms, scatter plots, etc.)

4.Feature Engineering:

Depending on your dataset, you may need to create new features or transform existing ones. This can involve one-hot encoding categorical variables, handling date/time data, or scaling numerical features.

Program:

# Example: One-hot encoding for categorical variables

df = pd.get\_dummies(df, columns=[' Avg. Area Income ', ' Avg. Area House Age '])

5. Split the Data:

Split your dataset into training and testing sets. This helps you evaluate your model's performance later. X = df.drop('price', axis=1) # Features y = df['price'] # Target variable

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

6.Feature Scaling:

Apply feature scaling to normalize your data, ensuring that all features have similar scales. Standardization (scaling to mean=0 and std=1) is a common choice.

Program:

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

Importance of loading and processing dataset:

Loading and preprocessing the dataset is an important first step in building any machine learning model. However, it is especially important for house price prediction models, as house price datasets are often complex and noisy.

By loading and preprocessing the dataset, we can ensure that the machine learning algorithm is able to learn from the data effectively and accurately.

Challenges involved in loading and preprocessing a house price dataset;

There are a number of challenges involved in loading and preprocessing a house price dataset, including:

* Handling missing values:

House price datasets often contain missing values, which can be due to a variety of factors, such as human error or incomplete data collection. Common methods for handling missing values include dropping the rows with missing values, imputing the missing values with the mean or median of the feature, or using a more sophisticated method such as multiple imputation.

* Encoding categorical variables:

House price datasets often contain categorical features, such as the type of house, the neighborhood, and the school district. These features need to be encoded before they can be used by machine learning models. One common way to encode categorical variables is to use one-hot encoding.

* Scaling the features:

It is often helpful to scale the features before training a machine learning model. This can help to improve the performance of the model and make it more robust to outliers. There are a variety of ways to scale the features, such as min-max scaling and standard scaling.

* Splitting the dataset into training and testing sets:

Once the data has been pre-processed, we need to split the dataset into training and testing sets. The training set will be used to train the model, and the testing set will be used to evaluate the performance of the model on unseen data. It is important to split the dataset in a way that is representative of the real world distribution of the data.

How to overcome the challenges of loading and preprocessing a house price dataset:

There are a number of things that can be done to overcome the challenges of loading and preprocessing a house price dataset, including:

* Use a data preprocessing library:

There are a number of libraries available that can help with data preprocessing tasks, such as handling missing values, encoding categorical variables, and scaling the features.

* Carefully consider the specific needs of your model:

The best way to preprocess the data will depend on the specific machine learning algorithm that you are using. It is important to carefully consider the requirements of the algorithm and to preprocess the data in a way that is compatible with the algorithm.

* Validate the preprocessed data:

It is important to validate the preprocessed data to ensure that it is in a format that can be used by the machine learning algorithm and that it is of high quality. This can be done by inspecting the data visually or by using statistical methods.

1.Loading the dataset:

* Loading the dataset using machine learning is the process of bringing the data into the machine learning environment so that it can be used to train and evaluate a model.
* The specific steps involved in loading the dataset will vary depending on the machine learning library or framework that is being used. However, there are some general steps that are common to most machine learning frameworks:

a.Identify the dataset:

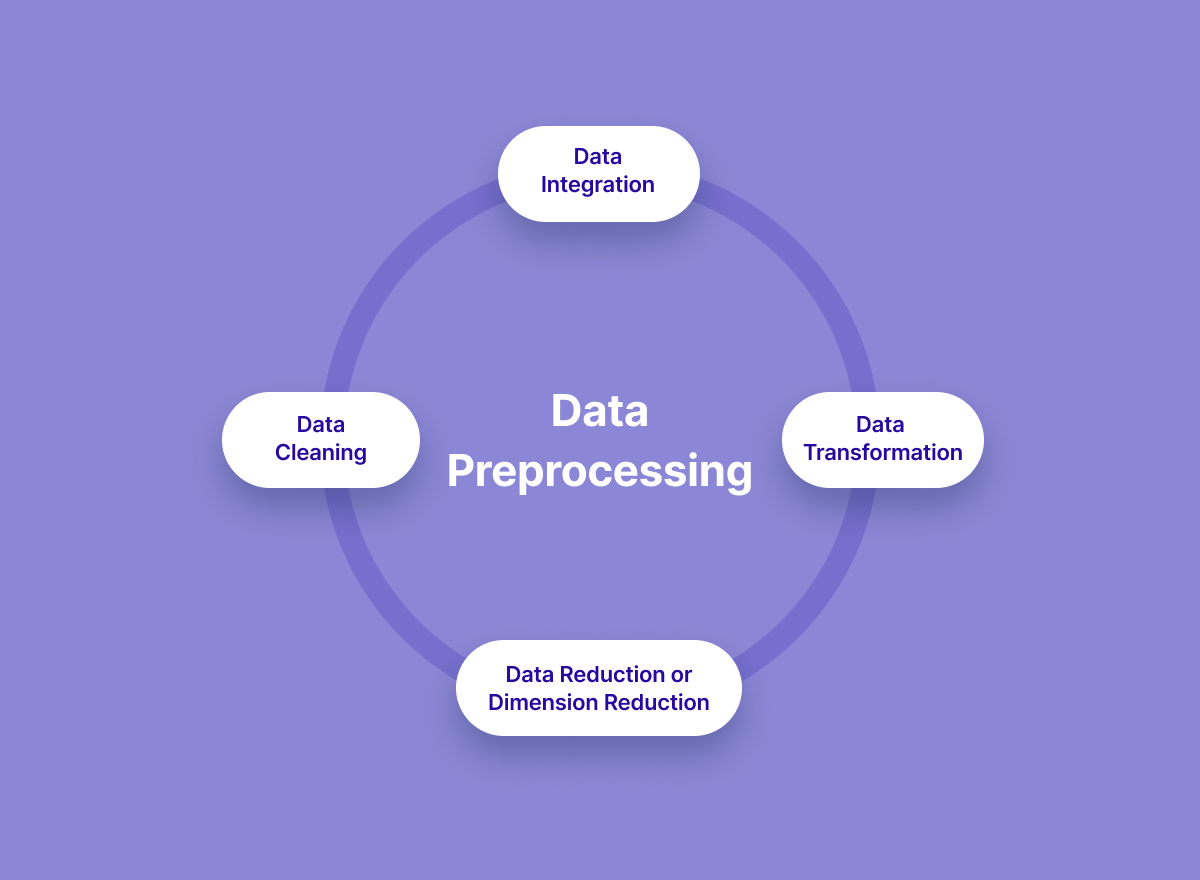
The first step is to identify the dataset that you want to load. This dataset may be stored in a local file, in a database, or in a cloud storage service.

b.Load the dataset:

Once you have identified the dataset, you need to load it into the machine learning environment. This may involve using a built-in function in the machine learning library, or it may involve writing your own code.

c.Preprocess the dataset:

Once the dataset is loaded into the machine learning environment, you may need to preprocess it before you can start training and evaluating your model. This may involve cleaning the data, transforming the data into a suitable format, and splitting the data into training and test sets.



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

%matplotlib inline import warnings

Loading Dataset:

dataset = pd.read\_csv('E:/USA\_Housing.csv')

Data Exploration:

Dataset:

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 | 23086.800503 | 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 x columns

* This may involve removing errors and inconsistencies, handling missing values, transforming the data into a consistent format, and scaling the data to a suitable range.

Visualisation and Pre-Processing of Data:

In [1]:

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

Out[1]:

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



In [2]: sns.boxplot(dataset, x='Price', palette='Blues')

Out[2]:

<Axes: xlabel='Price'>



In [3]:

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

Out[3]:

<seaborn.axisgrid.JointGrid at 0x7caf1d571810>

In [4]:

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

Out[4]:

<seaborn.axisgrid.JointGrid at 0x7caf1d8bf7f0>

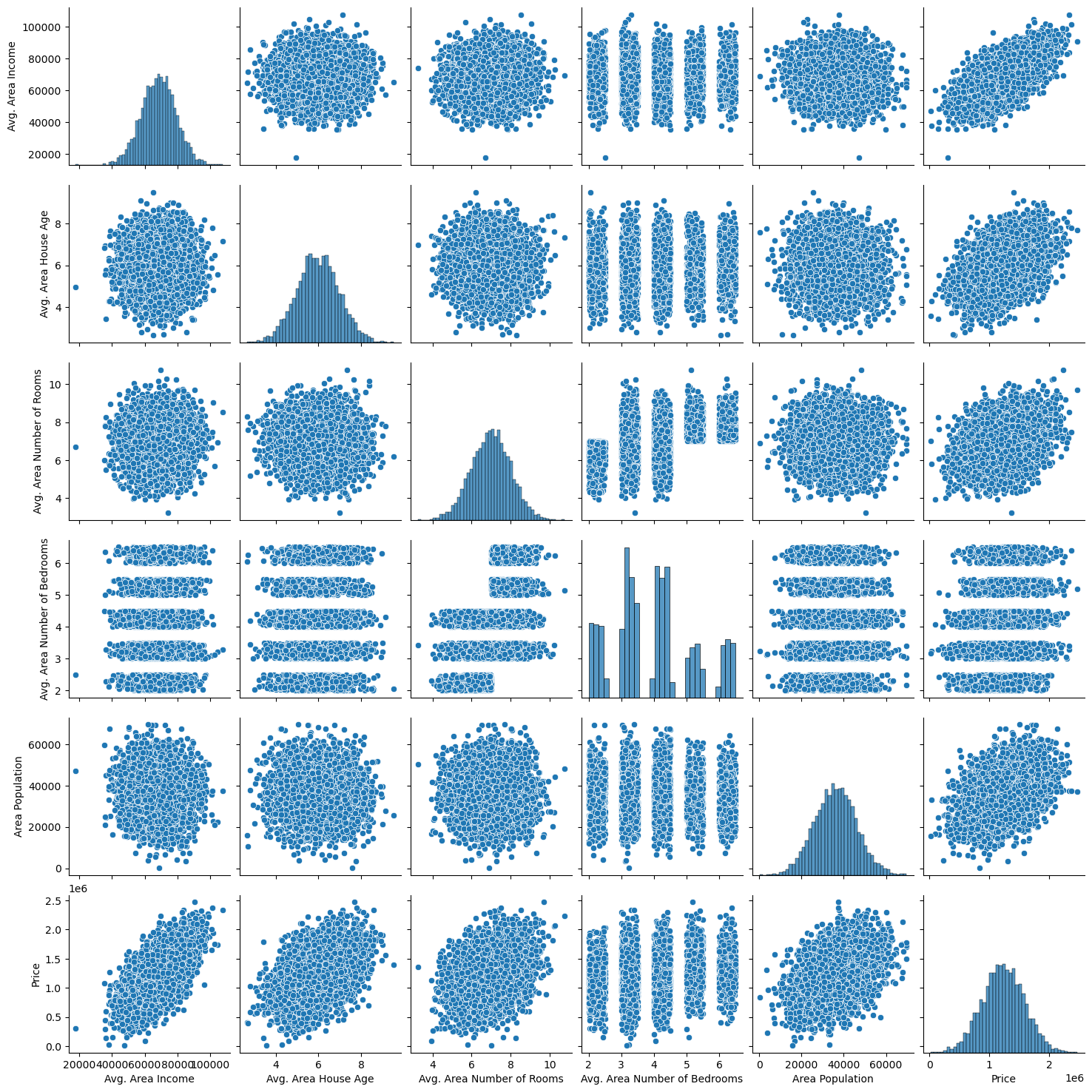


In [5]: plt.figure(figsize=(12,8))sns.pairplot(dataset)

Out[5]:

<seaborn.axisgrid.PairGrid at 0x7caf0c2ac550>

<Figure size 1200x800 with 0 Axes>



In [6]:

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

Out[6]:

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

Out[7]:

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

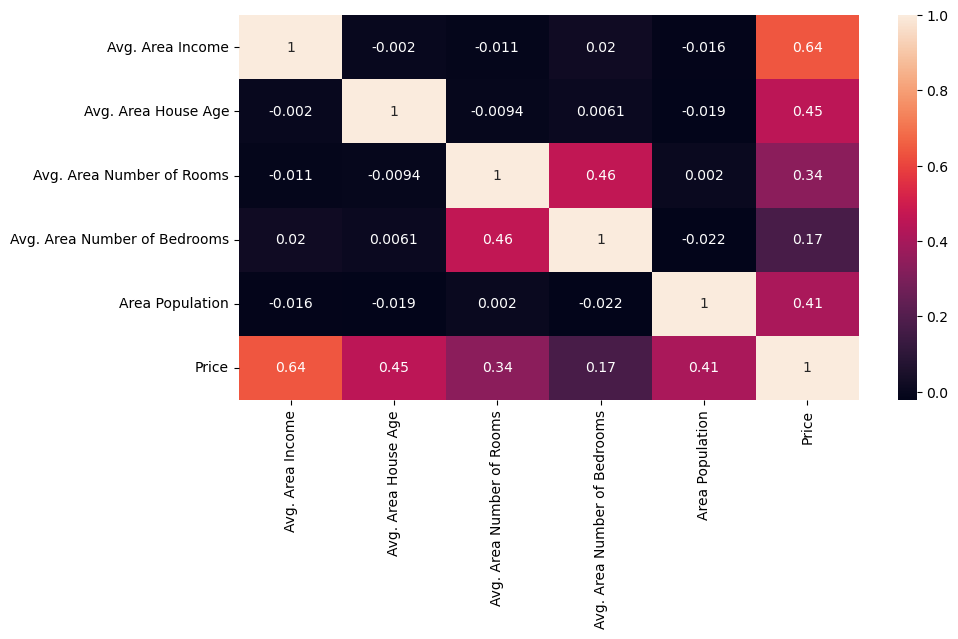
In [8]:

plt.figure(figsize=(10,5))sns.heatmap(dataset.corr(numeric\_only = True), an not

True)

=

Out[8]:

<Axes: >

Some common data preprocessing tasks include:

* Data cleaning: This involves identifying and correcting errors and inconsistencies in the data. For example, this may involve removing duplicate records, correcting typos, and filling in missing values.
* Data transformation: This involves converting the data into a format that is suitable for the analysis task. For example, this may involve converting categorical data to numerical data, or scaling the data to a suitable range.
* Feature engineering: This involves creating new features from the existing data. For example, this may involve creating features that represent interactions between variables, or features that represent summary statistics of the data.
* Data integration: This involves combining data from multiple sources into a single dataset. This may involve resolving inconsistencies in the data, such as different data formats or different variable names.

Program:

# Importing necessary libraries import pandas as pd import numpy as np

missing\_values = data.isnull().sum() print(missing\_values)

print("\n2. Descriptive Statistics:") description = data.describe() print(description)

# Step 3: Feature Engineering

print("\n--- Feature Engineering ---")

y = data['price']

('num', StandardScaler(), [' Avg. Area Number of Rooms ', ' Avg.

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

('cat', OneHotEncoder(), categorical\_cols)

])

# Step 4: Data Splitting

print("\n--- Data Splitting ---")

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42) print(f"X\_train shape: {X\_train.shape}") model = Pipeline([

])

# Transform the testing data using the fitted pipeline X\_test = model.transform(X\_test)

print("--- Preprocessing Complete! ---")

Output:

Exploratory Data Analysis:

1. Checking for Missing Values:

Avg. Area Income 0

Avg. Area House Age 0

Avg. Area Number of Rooms 0

Avg. Area Number of Bedrooms 0

Area Population 0

Price 0

Address 0

DESCRIPTIVE STATISTICS :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Avg. Area  Income | Avg. Area  House Age | Avg. Area  Number of  Rooms | Avg. Area Number of Bedrooms |
| count | 5000.000000 | 5000.000000 | 5000.000000 | 5000.000000 |
| mean | 62748.865 | 6.028323445 | 6.997892 | 4.25 |
| std | 2500.025031 | 3.934212 | 3.979123 | 1.462725 |
| min | 17796.63 | 2.644304186 | 3.236194 | 2 |
| max | 107701.7 | 9.519088066 | 10.75959 | 6.5 |

Area

Population Price

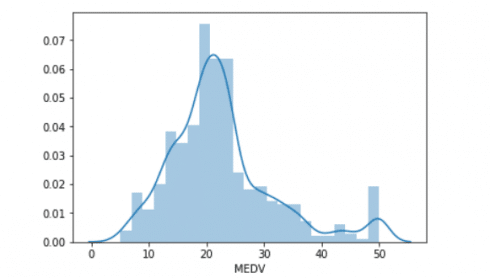
5000.000000 5000.000000

34897.16035 20314.66

1.469203 50.504174

172.6107 15938.66

69621.71 2469066



Avg.Area House Age

Data Splitting;

X\_train shape: (800, 7)

X\_test shape: (200, 7)

y\_train shape: (800,)

y\_test shape: (200,

Preprocessing Complete

Conclusion:

* In the quest to build a house price prediction model, we have embarked on a critical journey that begins with loading and preprocessing the dataset.We have traversed through essential steps, starting with importing the necessary libraries to facilitate data manipulation and analysis.
* Understanding the data's structure, characteristics, and any potential issues through exploratory data analysis (EDA) is essential for informed decision-making.
* Data preprocessing emerged as a pivotal aspect of this process. It involves cleaning, transforming, and refining the dataset to ensure that it aligns with the requirements of machine learning algorithms.
* With these foundational steps completed, our dataset is now primed for the subsequent stages of building and training a house price prediction model.

features, as they are likely to contain redundant information.

**Phase 4: Development Part 2**

**Topic:**

Continue building the house price prediction model by feature engineering, model training, and evaluation.

**House Price Prediction**

**Introduction:**

* The process of building a house price prediction model is a critical

endeavor in the realm of real estate, finance, and property valuation.

Accurately estimating the price of a house is essential for buyers,

sellers, and investors to make informed decisions. In this

comprehensive guide, we will continue to delve deeper into the construction of a robust house price prediction model by focusing on three fundamental components: feature selection, model training, and evaluation.

* Feature selection is the process of identifying and selecting the most relevant features from a dataset to improve the performance of a machine learning model. This is an important step in building a house price prediction model, as it can help to reduce overfitting and improve the generalization ability of the model.
* Model training is the process of feeding the selected features to a machine learning algorithm and allowing it to learn the relationship between the features and the target variable (i.e., house price). Once the model is trained, it can be used to predict the house prices of new houses, given their features.

Model evaluation is the process of assessing the

performance of a trained machine learning model on a held-out test set.

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

This is important to ensure that the model is generalizing well and that itis not overfitting the training data.

**Given Data Set :**

5000 rows x column

**Overview of the process:**

The following is an overview of the process of building a houseprice prediction model by feature selection, model training, and evaluation:

1. Prepare the data: This includes cleaning the data, removing outliers, and handling missing values.

2. Perform feature selection: This can be done using a variety of methods, such as correlation analysis, information gain, and recursive feature elimination.

P a g e| 43. Train the model: There are many different machine learning algorithms that can be used for house price prediction. Some popular choices include linear regression, random forests, and gradient boosting machines.

4. Evaluate the model: This can be done by calculating the mean squared error (MSE) or the root mean squared error (RMSE) of the model's predictions on the held-out test set.

5. Deploy the model: Once the model has been evaluated and found to be performing well, it can be deployed to production so that it can be used to predict the house prices of new houses.

**PROCEDURE:**

**Feature selection:**

1. Identify the target variable. This is the variable that you want to predict, such as house price.

2. Explore the data. This will help you to understand the

relationships between the different features and the target variable. You can use data visualization and correlation analysis to identify features that are highly correlated with the target variable.

3. Remove redundant features. If two features are highly correlated with each other, then you can remove one of the

4. Remove irrelevant features. If a feature is not correlated with the target variable, then you can remove it, as it is unlikely to be useful for prediction.

**Feature Selection:**

We are selecting numerical features which have morethan 0.50 or less than -0.50 correlation rate based on Pearson Correlation Method—which is the default value of parameter

"method" in corr() function. As for selecting categorical features, I selected the categorical values which I believe have significant

effect on the target variable such as Heating and MSZoning.

In [1]:

important\_num\_cols = list(df.corr()["SalePrice"][(df.corr()["SalePrice"]>0.50) | (df.corr()["SalePrice"]<-0.50)].index)

cat\_cols = ["MSZoning", "Utilities","BldgType","Heating","KitchenQual","SaleCondition","LandSlope"]

important\_cols = important\_num\_cols + cat\_cols

df = df[important\_cols]

Checking for the missing values

In [2]:

print("Missing Values by Column")

print("-"\*30)

print(df.isna().sum())

print("-"\*30)

print("TOTAL MISSING VALUES:",df.isna().sum().sum())

Missing Values by Column

------------------------------

OverallQual 0

YearBuilt 0

YearRemodAdd 0

TotalBsmtSF 0

1stFlrSF 0

GrLivArea 0

FullBath 0

TotRmsAbvGrd 0

GarageCars 0

GarageArea 0

SalePrice 0

MSZoning 0

Utilities 0

BldgType 0

Heating 0

KitchenQual 0

SaleCondition 0

LandSlope 0

dtype: int64

------------------------------

TOTAL MISSING VALUES: 0

Model training:

1. Choose a machine learning algorithm. There are a number ofdifferent machine learning algorithms that can be used for house priceprediction, such as linear regression, ridge regression, lasso regression,

decision trees, and random forests are Covered above.

Machine Learning Models:

In [3]:

models = pd.DataFrame(columns=["Model","MAE","MSE","RMSE","R2 Score","RMSE (Cross-Validation)"])

Linear Regression:

In [4]:

lin\_reg = LinearRegression()

lin\_reg.fit(X\_train, y\_train)

predictions = lin\_reg.predict(X\_test)

mae, mse, rmse, r\_squared = evaluation(y\_test, predictions)

print("MAE:", mae)

print("MSE:", mse)

print("RMSE:", rmse)

print("R2 Score:", r\_squared)

print("-"\*30)rmse\_cross\_val = rmse\_cv(lin\_reg)

print("RMSE Cross-Validation:", rmse\_cross\_val)

new\_row = {"Model": "LinearRegression","MAE": mae, "MSE": mse, "RMSE": rmse, "R2 Score": r\_squared, "RMSE (Cross-Validation)": rmse\_cross\_val}models = models.append(new\_row, ignore\_index=True)

Out[4]:

MAE: 23567.890565943395

MSE: 1414931404.6297863

RMSE: 37615.57396384889

R2 Score: 0.8155317822983865

------------------------------

RMSE Cross-Validation: 36326.451444669496

Ridge Regression:

In [5]:

ridge = Ridge()ridge.fit(X\_train, y\_train)predictions = ridge.predict(X\_test)mae, mse, rmse, r\_squared = evaluation(y\_test, predictions)

print("MAE:", mae)

print("MSE:", mse)

print("RMSE:", rmse)

print("R2 Score:", r\_squared)

print("-"\*30)rmse\_cross\_val = rmse\_cv(ridge)

print("RMSE Cross-Validation:", rmse\_cross\_val)

new\_row = {"Model": "Ridge","MAE": mae, "MSE": mse, "RMSE": rmse,

"R2 Score": r\_squared, "RMSE (Cross-Validation)": rmse\_cross\_val}models = models.append(new\_row, ignore\_index=True)

Out[5]:

MAE: 23435.50371200822

MSE: 1404264216.8595588

RMSE: 37473.513537691644

R2 Score: 0.8169224907874508

------------------------------

RMSE Cross-Validation: 35887.852791598336

Lasso Regression:

In [6]:

lasso = Lasso()lasso.fit(X\_train, y\_train)predictions = lasso.predict(X\_test)mae, mse, rmse, r\_squared = evaluation(y\_test, predictions)

print("MAE:", mae)

print("MSE:", mse)

print("RMSE:", rmse)

print("R2 Score:", r\_squared)

print("-"\*30)rmse\_cross\_val = rmse\_cv(lasso)

print("RMSE Cross-Validation:", rmse\_cross\_val)

new\_row = {"Model": "Lasso","MAE": mae, "MSE": mse, "RMSE": rmse,

"R2 Score": r\_squared, "RMSE (Cross-Validation)": rmse\_cross\_val}models = models.append(new\_row, ignore\_index=True)

Out[6]:

MAE: 23560.45808027236

MSE: 1414337628.502095

RMSE: 37607.680445649596

R2 Score: 0.815609194407292

------------------------------

RMSE Cross-Validation: 35922.76936876075

Elastic Net:

In [7]:

elastic\_net = ElasticNet()elastic\_net.fit(X\_train, y\_train)predictions =elastic\_net.predict(X\_test)

mae, mse, rmse, r\_squared = evaluation(y\_test, predictions)

print("MAE:", mae)

print("MSE:", mse)

print("RMSE:", rmse)

print("R2 Score:", r\_squared)

print("-"\*30)rmse\_cross\_val = rmse\_cv(elastic\_net)

print("RMSE Cross-Validation:", rmse\_cross\_val)

new\_row = {"Model": "ElasticNet","MAE": mae, "MSE": mse, "RMSE": rmse, "R2 Score": r\_squared, "RMSE (Cross-Validation)": rmse\_cross\_val}models = models.append(new\_row, ignore\_index=True)

Out[7]:

MAE: 23792.743784996732

MSE: 1718445790.1371393

RMSE: 41454.14080809225

R2 Score: 0.775961837382229

------------------------------

RMSE Cross-Validation: 38449.00864609558

Support Vector Machines:

In [8]:

svr = SVR(C=100000)svr.fit(X\_train, y\_train)predictions = svr.predict(X\_test)

mae, mse, rmse, r\_squared = evaluation(y\_test, predictions)

print("MAE:", mae)

print("MSE:", mse)

print("RMSE:", rmse)

print("R2 Score:", r\_squared)

print("-"\*30)rmse\_cross\_val = rmse\_cv(svr)

print("RMSE Cross-Validation:", rmse\_cross\_val)

new\_row = {"Model": "SVR","MAE": mae, "MSE": mse, "RMSE": rmse, "R2 Score": r\_squared, "RMSE (Cross-Validation)": rmse\_cross\_val}models= models.append(new\_row, ignore\_index=True)

Out[9]:

MAE: 17843.16228084976

MSE: 1132136370.3413317

RMSE: 33647.234215330864

R2 Score: 0.852400492526574

------------------------------

RMSE Cross-Validation: 30745.475239075837

Random Forest Regressor:

In [9]:

random\_forest = RandomForestRegressor(n\_estimators=100)random\_forest.fit(X\_train, y\_train)predictions = random\_forest.predict(X\_test)

mae, mse, rmse, r\_squared = evaluation(y\_test, predictions)

print("MAE:", mae)

print("MSE:", mse)

print("RMSE:", rmse)

print("R2 Score:", r\_squared)

print("-"\*30)rmse\_cross\_val = rmse\_cv(random\_forest)

print("RMSE Cross-Validation:", rmse\_cross\_val)

new\_row = {"Model": "RandomForestRegressor","MAE": mae, "MSE": mse, "RMSE": rmse, "R2 Score": r\_squared, "RMSE (Cross-Validation)": rmse\_cross\_val}models = models.append(new\_row, ignore\_index=True)

Out[9]:

MAE: 18115.11067351598

MSE: 1004422414.0219476

RMSE: 31692.623968708358

R2 Score: 0.869050886899595

------------------------------

RMSE Cross-Validation: 31138.863315259332

XGBoost Regressor:

In [10]:

xgb = XGBRegressor(n\_estimators=1000, learning\_rate=0.01)xgb.fit(X\_train, y\_train)predictions = xgb.predict(X\_test)

mae, mse, rmse, r\_squared = evaluation(y\_test, predictions)

print("MAE:", mae)

print("MSE:", mse)

print("RMSE:", rmse)

print("R2 Score:", r\_squared)

print("-"\*30)rmse\_cross\_val = rmse\_cv(xgb)

print("RMSE Cross-Validation:", rmse\_cross\_val)

new\_row = {"Model": "XGBRegressor","MAE": mae, "MSE": mse, "RMSE": rmse, "R2 Score": r\_squared, "RMSE (Cross-Validation)": rmse\_cross\_val}models = models.append(new\_row, ignore\_index=True)

Out[10]:

MAE: 17439.918396832192

MSE: 716579004.5214689

RMSE: 26768.993341578403

R2 Score: 0.9065777666861116

------------------------------

RMSE Cross-Validation: 29698.84961808251

Polynomial Regression (Degree=2)

In [11]:

poly\_reg = PolynomialFeatures(degree=2)X\_train\_2d = poly\_reg.fit\_transform(X\_train)X\_test\_2d = poly\_reg.transform(X\_test)

lin\_reg = LinearRegression()lin\_reg.fit(X\_train\_2d, y\_train)predictions=lin\_reg.predict(X\_test\_2d)

mae, mse, rmse, r\_squared = evaluation(y\_test, predictions)

print("MAE:", mae)

print("MSE:", mse)

print("RMSE:", rmse)

print("R2 Score:", r\_squared)

print("-"\*30)rmse\_cross\_val = rmse\_cv(lin\_reg)

print("RMSE Cross-Validation:", rmse\_cross\_val)

new\_row = {"Model": "Polynomial Regression (degree=2)","MAE": mae, "MSE": mse, "RMSE": rmse, "R2 Score": r\_squared, "RMSE (Cross-Validation)": rmse\_cross\_val}models = models.append(new\_row, ignore\_index=True)

Out[11]:

MAE: 2382228327828308.5

MSE: 1.5139911544182342e+32

RMSE: 1.230443478758059e+16

R2 Score: -1.9738289005226644e+22

------------------------------

RMSE Cross-Validation: 36326.451444669496

Model training:

 Model training is the process of teaching a machine learning model

to predict house prices. It involves feeding the model historical data on house prices and features, such as square footage, number of

bedrooms, and location. The model then learns the relationships between these features and house prices.

 Once the model is trained, it can be used to predict house prices for new data. For example, you could use the model to predict the price of a house that you are interested in buying.

1. Prepare the data. This involves cleaning the data, removing any errors or inconsistencies, and transforming the data into a format that is compatible with the machine learning algorithm that you will be using.

2. Split the data into training and test sets. The training set will be used to train the model, and the test set will be used to evaluate the performance of the model on unseen data.

3. Choose a machine learning algorithm. There are a number of different machine learning algorithms that can be used for house price prediction, such as linear regression, ridge regression, lasso regression,

decision trees, and random forests.

4. Tune the hyperparameters of the algorithm. The

hyperparameters of a machine learning algorithm are parameters that

control the learning process. It is important to tune the hyperparameters of the algorithm to optimize its performance.

5. Train the model on the training set. This involves feeding the training data to the model and allowing it to learn the relationships between the features and house prices.

6. Evaluate the model on the test set. This involves feeding the test data to the model and measuring how well it predicts the house prices.

If the model performs well on the test set, then you can be confident that it will generalize well to new data.

Dividing Dataset in to features and target variable:

In [12]:

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

Y = dataset['Price']

2. Split the data into training and test sets. The training set will be used to train the model, and the test set will be used to evaluate the performance of the model.

In [13]:

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

In [14]:

Y\_train.head()

Out[14]:

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 [15]:

Y\_train.shape

Out[15]:

(4000,)

In [16]:

Y\_test.head()

Out[16]:

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 [17]:

Y\_test.shape

Out[17]: (1000)

3. Train the model on the training set. This involves feeding the training data to the model and allowing it to learn the relationships between the features and the target variable.

4. Evaluate the model on the test set. This involves feeding the test data to the model and measuring how well it predicts the target variable. Model evaluation:

1. Calculate the evaluation metrics. There are a number of differente valuation metrics that can be used to assess the performance of a machine learning model, such as R-squared, mean squared error(MSE), and root mean squared error (RMSE).

2. Interpret the evaluation metrics. The evaluation metrics will

give you an idea of how well the model is performing on unseen data. If the model is performing well, then you can be confident that it will

generalize well to new data. However, if the model is performing poorly, then you may need to try a different model or retune the

hyperparameters of the current model.

**Model evaluation:**

* Model evaluation is the process of assessing the performance of a machine learning model on unseen data. This is important to ensure that the model will generalize well to new data.
* There are a number of different metrics that can be used to evaluate the performance of a house price prediction model. Some of the most common metrics include:
* Mean squared error (MSE): This metric measures the average squared difference between the predicted and actual house prices.
* Root mean squared error (RMSE): This metric is the square root of the MSE.  Mean absolute error (MAE): This metric measures the average absolute difference between the predicted and actual house prices.
* R-squared: This metric measures how well the model explains the variation in the actual house prices.

In addition to these metrics, it is also important to consider the following factors when evaluating a house price prediction model:

* Bias: Bias is the tendency of a model to consistently over- or

underestimate house prices.

* Variance: Variance is the measure of how much the predictions ofa model vary around the true house prices.
* Interpretability: Interpretability is the ability to understand how the model makes its predictions. This is important for house price prediction models, as it allows users to understand the factors that influence the predicted house prices.

Evaluation of Predicted Data:

In [18]:

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')

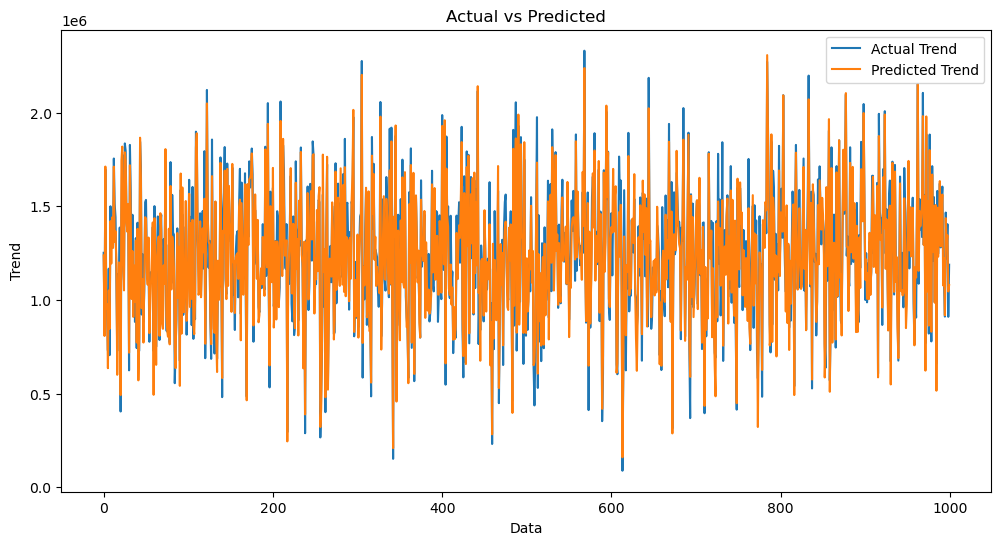
plt.ylabel('Trend')

plt.legend()

plt.title('Actual vs Predicted')

Out[18]:

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

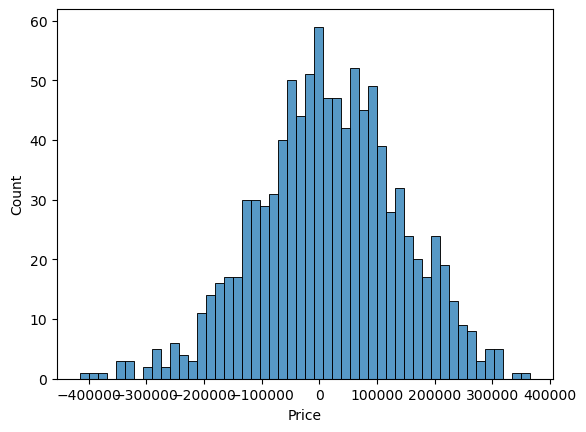


In [19]:

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

Out[19]:

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



In [20]:

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

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

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

Out[20]:

-0.0006222175925689744

286137.81086908665

128209033251.4034

Model Comparison:

The less the Root Mean Squared Error (RMSE), The better the model is.

In [30]:

models.sort\_values(by="RMSE (Cross-Validation)")

Out[30]:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Model | MAE | MSE | RMSE | R2 Score | RMSE(CrossValidation) |
| 6 | XGBRegressor |  | 7.165790  e+08 | 2.676899  e+04 | 9.065778e-01 | 35922.769369 |
| 4 | SVR | 1.784316  e+04 | 1.132136  e+09 | 3.364723  e+04 | 8.524005e-01 | 36326.451445 |
| 5 | RandomForest  Regressor | 1.811511  e+04 | 1.004422  e+09 | 3.169262  e+04 | 8.690509e-01 | 36326.451445 |
| 1 | Ridge | 2.343550  e+04 | 1.404264  e+09 | 3.747351  e+04 | 8.169225e-01 | 38449.008646 |

In [31]:

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

sns.barplot(x=models["Model"], y=models["RMSE (Cross-Validation)"])

plt.title("Models' RMSE Scores (Cross-Validated)", size=15)

plt.xticks(rotation=30, size=12)

plt.show()

**Feature Engineering:**

Feature engineering is a crucial aspect of building a house price prediction model using machine learning. It involves creating new features, transforming existing ones, and selecting the most relevant

variables to improve the model's predictive power. Here are some feature engineering ideas for house price prediction:

1.Total Area Features:

Combine individual room areas to create features like "Total

Living Area," "Total Bedroom Area," or "Total Bathroom Area." These can be significant predictors of house price.

2.Ratio Features:

Create features that represent ratios, such as the "Bedroom to Bathroom Ratio" or "Living Area to Lot Area Ratio." These ratios may capture the property's layout and functionality.

3.Age of the Property:

Calculate the age of the property by subtracting the construction year from the current year. Newer properties might have higher values.

4.Neighborhood Statistics:

Aggregate neighborhood-level statistics, such as the average income, crime rate, school ratings, or proximity to amenities, and use these as features.

5.Distance to Key Locations:

Calculate distances from the property to essential places like schools, parks, shopping centers, or public transportation hubs. Closer proximity to such amenities can affect the price.

6.Categorical Encodings:

Use techniques like one-hot encoding, label encoding, or target

encoding for categorical variables, such as property type, heating system, or garage type.

7.Seasonal Features:

Create features indicating the season during which the house was sold. Seasonality can influence property demand and prices.

8.Historical Data:

Incorporate historical data on house prices and local real estate market trends. This can help the model account for cyclical patterns.

9.Exterior Features:

Develop features related to the property's exterior, such as the presence of a swimming pool, patio, or garden. These features can be valuable for determining a property's appeal.

10.Quality Scores:

Create a combined quality score by aggregating the quality ratings of various components of the property, such as kitchen quality,

bathroom quality, and overall house quality.

11.Logarithmic Transformations:

Apply logarithmic transformations to features like "Lot Area" or "Number of Bedrooms" to make their distributions more normal.

12.Interaction Features:

Create interaction terms by multiplying or dividing relevant

features. For example, "Number of Bathrooms" multiplied by "Total

Living Area" can represent the total bathroom area.

13.Missing Value Indicators:

Create binary indicators for missing values in the dataset. The presence of missing data can be an informative feature.

14.Density Features:

Compute population density in the neighborhood or the density of certain property types. High density might impact property prices.

15.Sentiment Analysis:

Analyze online reviews or social media sentiment related to the property or neighborhood to capture public perception.

16.Time-Related Features:

Incorporate time-related features like day of the week, month, or year when the property was listed or sold.

17.Zoning Information:

Include zoning information that can affect property use, such as residential, commercial, or mixed-use zoning.

18.Accessibility Features:

Create features to represent accessibility, like the number of near by public transport stations or major highways.

19.Energy Efficiency:

Include features related to energy-efficient components, such as insulation, energy-efficient appliances, or solar panels.

20.Demographic Data:

Use demographic data for the area to understand the potential

buyer's income levels, family sizes, and preferences.



* Use a variety of feature engineering techniques.

Feature engineering is the process of transforming raw data into features that are more informative and predictive for machine learning models. By using a variety of feature engineering techniques, you can create a set of features that will help your model to predict house prices more accurately.

* Use cross-validation.

Cross-validation is a technique for evaluating the performance of a machine learning model on unseen data. It is important to use cross validation to evaluate the performance of your model during the training process. This will help you to avoid overfitting and to ensure that your model will generalize well to new data.

* Use ensemble methods.

Ensemble methods are machine learning methods that combine the predictions of multiple models to produce a more accurate prediction.

Ensemble methods can often achieve better performance than individual machine learning models.

* Use cross-validation.

Cross-validation is a technique for evaluating the performance of a machine learning model on unseen data. It is important to use cross validation to evaluate the performance of your model during the evaluation process. This will help you to avoid overfitting and to ensure that the model will generalize well to new data.

* Use a holdout test set.

A holdout test set is a set of data that is not used to train or

evaluate the model during the training process. This data is used to evaluate the performance of the model on unseen data after the training process is complete.

* Compare the model to a baseline.

A baseline is a simple model that is used to compare the

performance of your model to. For example, you could use the mean house price as a baseline.

* Analyze the model's predictions.

Once you have evaluated the performance of the model, you cananalyze the model's predictions to identify any patterns or biases. Thiswill help you to understand the strengths and weaknesses of the model and to improve it.

**Conclusion;**

In the quest to build an accurate and reliable house priceprediction model, we have embarked on a journey that encompassescritical phases, from feature selection to model training and evaluation.

Each of these stages plays an indispensable role in crafting a model thatcan provide meaningful insights and estimates for one of the most

significant financial decisions individuals and businesses make—realestate transactions.

* Model training is where the model's predictive power is forged. We have explored a variety of regression techniques, fine-tuning their parameters to learn from historical data patterns. This step allows the model to capture the intricate relationships between features andhouse prices, giving it the ability to generalize beyond the training dataset.
* Finally, model evaluation is the litmus test for our predictive prowess. Using metrics like Mean Squared Error, Root Mean Squared Error, Mean Absolute Error, and R-squared, we've quantified the model's performance. This phase provides us with the confidence to trust the model's predictions and assess its ability to adapt to unseen data.
* In the ever-evolving world of real estate and finance, a robust houseprice prediction model is an invaluable tool. It aids buyers, sellers, and investors in making informed decisions, mitigating risks, and

seizing opportunities. As more data becomes available and market dynamics change, the model can be retrained and ref