**PREDICTING HOUSE PRICES USING MACHINE LEARNING**

**Introduction:**

The real estate market is a dynamic and complex area, where Property values can fluctuate significantly due to a multitude of factors. For both homebuyers and sellers, accurately determining the fair market value of a property is of paramount importance.

 In this era of technological advancement, machine learning has emerged as a game-changing tool in the realm of real estate. One of its most compelling applications is predicting house prices with remarkable accuracy.

 Traditional methods of property valuation, relying on factors such as location, square footage, and recent sales data, are undoubtedly useful. However, they often fall short in capturing the intricacies and nuances that drive real estate market dynamics.

 Machine learning, on the other hand, has the capability to process vast volumes of data and identify patterns that human appraisers might overlook. This technology has the potential to revolutionize the way we value real estate, offering more precise and data-driven predictions.

 In this exploration, we delve into the exciting world of predicting house prices using machine learning. We will uncover how this cutting-edge technology harnesses the power of algorithms and data to create predictive models that consider an array of variables, such as neighborhood characteristics, property features, economic indicators, and even social trends.

 By doing so, machine learning enables us to make informed, data-backed predictions about the future value of a property

 This transformation of the real estate industry is not only beneficial for buyers and sellers but also for investors, developers, and policymakers. Accurate house price predictions can inform investment decisions, urban planning, and housing policy development, leading to a more efficient and equitable real estate market.

 As we embark on this journey into the realm of machine learning for house price prediction, we will explore the various techniques, data sources, and challenges involved.

Dataset Link: (<https://www.kaggle.com/datasets/vedavyasv/usa-housing> )

**Given data set:**

**Here’s a list of tools and software commonly used in the Process:**

**1.Programming Language:**

-Python is the most popular language for machine learning due to its extensive libraries and frameworks. You can use libraries like NumPy, Pandas, scikit-learn, and more.

**2.Integrated Development Environment (IDE):**

-Choose an IDE for coding and running machine learning experiments. Some popular options include Jupyter Notebook, Google colab, or traditional IDEs like PyCharm.

**3.Machine Learning Libraries:**

You’ll need various machine learning libraries, including:

-Scikit-learn for building and evaluating machine learning models.

-TensorFlow or PyTorch for deep learning, if needed.

-XGBoost, LightGBM, or CatBoost for gradient boosting models.

**4.Data Visualization Tools:**

-Tools like Matplotlib, Seaborn, or Plotly are essential for data exploration and visualization.

**5. Data Preprocessing Tools:**

- Libraries like pandas help with data cleaning, manipulation, and Preprocessing.

**6. Data Collection and Storage:**

-Depending on your data source, you might need web tools (e.g., BeautifulSoup or Scrapy) or databases (e.g., SQLite, PostgreSQL) for data storage.

**7. Version Control:**

**-**Version control systems like Git are valuable for trackingchanges in your code and collaborating with others.

**8. Notebooks and Documentation:**

-Tools for documenting your work, such as Jupyter Notebooks or Markdown for creating README files and documentation.

**9. Hyperparameter Tuning:**

-Tools like GridSearchCV or RandomizedSearchCV from Scikit-learn can help with hyperparameter tuning.

**10. Web Development Tools (for Deployment):**

-If you plan to create a web application for model deployment, Knowledge of web development tools like Flask or Django for backend Development, and HTML, CSS, and JavaScript for the front-end can be useful.

**11. Cloud Services (for Scalability):**

-For large-scale applications, cloud platforms like AWS, Google Cloud, or Azure can provide scalable computing and storage resources.

**12. External Data Sources (if applicable):**

Depending on your project’s scope, you might require tools to access external data sources, such as APIs or data scraping tools.

**13. Data Annotation and Labeling Tools (if applicable):**

-For specialized projects, tools for data annotation and beling may be necessary, such as Labelbox or Supervisely.

**14. Geospatial Tools (for location-based features):**

-If your dataset includes geospatial data, geospatial libraries Like GeoPandas can be helpful.

**1.DESIGN THINKING AND PRESENT IN FORM OF DOCUMENT**

**1.Empathize:**

 Understand the needs and challenges of all stakeholders involved in the house price prediction process, including homebuyers, sellers, real estate professionals, appraisers, and investors.

 Conduct interviews and surveys to gather insights on what users Value in property valuation and what information is most critical for their decision-making.

**2.Define:**

 Clearly articulate the problem statement, such as “How might we predict house prices more accurately and transparently using machine Learning?”

 Identify the key goals and success criteria for the project, such as increasing prediction accuracy, reducing bias, or improving user trust in the valuation process.

**3.Ideate:**

 Brainstorm creative solutions and data sources that can enhance the accuracy and transparency of house price predictions.

 Encourage interdisciplinary collaboration to generate a wide range of ideas, including the use of alternative data, new algorithms, or improved visualization techniques.

**4.Prototype:**

 Create prototype machine learning models based on the technique generated during the ideation phase.

 Test and iterate on these prototypes to determine which approaches are most promising in terms of accuracy and usability.

**5.Test:**

 Gather feedback from users and stakeholders by testing the machine learning models with real-world data and scenarios.

 Assess how well the models meet the defined goals and success Criteria, and make adjustments based on user feedback.

**6.Implement:**

 Develop a production-ready machine learning solution for predicting house prices, integrating the best-performing algorithms and data Sources.

 Implement transparency measures, such as model interpretability tools, to ensure users understand how predictions are generated.

**7.Evaluate:**

 Continuously monitor the performance of the machine learning model after implementation to ensure it remains accurate and Relevant in a changing real estate market.

 Gather feedback and insights from users to identify areas for improvement.

**8.Iterate**:

 Apply an iterative approach to refine the machine learning model Based on ongoing feedback and changing user needs.

 Continuously seek ways to enhance prediction accuracy, transparency, and user satisfaction.

**9.Scale and Deploy:**

 Once the machine learning model has been optimized and validated, Deploy it at scale to serve a broader audience, such as real estate Professionals, investors, and homeowners.

 Ensure the model is accessible through user-friendly interfaces and Integrates seamlessly into real estate workflows.

**10.Educate and Train:**

 Provide training and educational resources to help users understand How the machine learning model works, what factors it considers, and its limitations.

 Foster a culture of data literacy among stakeholders to enhance trust In the technology.

**2.DESIGN INTO INNOVATION**

**1.Data Collection:**

Gather a comprehensive dataset that includes features such as loocation, size, age, amenities, nearby schools, crime rates, and other relevant variables.

**2.Data Preprocessing:**

Clean the data by handling missing values, outliers, and Encoding categorical variables. Standardize or normalize numerical features as necessary.

**PYHON PROGRAM:**

**# Import necessary libraries**

import pandas as pd

from sklearn.preprocessing import LabelEncoder

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

from sklearn.impute import SimpleImputer

from sklearn.preprocessing import StandardScaler

**# Load the dataset (replace ‘house\_data.csv’ with your dataset file)**

data = pd.read\_csv(‘E:/USA\_Housing.csv’)

**# Display the first few rows of the dataset to get an overview**

Print(“Dataset Preview:”)

Print(data.head())

# **Data Pre-processing**

# Handle Missing Values

# Let’s fill missing values in numeric columns with the mean and in Categorical columns with the most frequent value.

numeric\_cols =data.select\_dtypes(include=’number’).columns

categorical\_cols = data.select\_dtypes(exclude=’number’).columns

imputer\_numeric = SimpleImputer(strategy=’mean’)

imputer\_categorical=SimpleImputer(strategy=’most\_frequent’)

data[numeric\_cols] =

imputer\_numeric.fit\_transform(data[numeric\_cols])

data[categorical\_cols]=

imputer\_categorical.fit\_transform(data[categorical\_cols])

# Convert Categorical Features to Numerical

# We’ll use Label Encoding for simplicity here. You can also use one- Hot encoding for nominal categorical features.

label\_encoder = LabelEncoder()

for col in categorical\_cols:

data[col] = label\_encoder.fit\_transform(data[col])

# Split Data into Features (X) and Target (y)

X = data.drop(columns=[‘Price’]) # Features

y= data[‘Price’] # Target

# Normalize the Data

Scaler = StandardScaler()

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

# Split data into training and testing sets (adjust test\_size as needed)

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

# Display the preprocessed data

print(“\nPreprocessed Data:”)

print(X\_train[:5]) # Display first 5 rows of preprocessed features

print(y\_train[:5]) # Display first 5 rows of target values

**OUTPUT:**

**Dataset Preview:**



**Preprocessed Data:**

[[-0.19105816 -0.13226994 -0.13969293 0.12047677 -0.83757985 -1.0

0562872]

[-1.39450169 0.42786736 0.79541275 -0.55212509 1.15729018 1.61

946754]

[-0.35137865 0.46394489 1.70199509 0.03133676 -0.32671213 1.63

886651]

[-0.13944143 0.1104872 0.22289331 -0.75471601 -0.90401197 -1.54

810704]

[ 0.62516685 2.20969666 0.42984356 -0.45488144 0.12566216 0.98

830821]]

4227 1.094880e+06

4676 1.300389e+06

800 1.382172e+06

3671 1.027428e+06

4193 1.562887e+06

Name: Price, dtype: float64

**3.Feature Engineering:**

Create new features or transform existing ones to extract more vauable information. For example, you can calculate the distance to the nearest public transportation, or create a feature for the overall condition of the house.

**4.Model Selection:**

Choose the appropriate machine learning model for the task.Common models for regression problems like house price prediction Include Linear Regression, Decision Trees, Random Forest, Gradient Boosting, and Neural Networks.

**5. Training:**

Split the dataset into training and testing sets to evaluate the model’s performance. Consider techniques like cross-validation to prevent overfitting.

**6. Hyperparameter Tuning:**

Optimize the model’s hyperparameters to improve its predictive accuracy. Techniques like grid search or random search can help with this.

**7.Evaluation Metrics:**

Select appropriate evaluation metrics for regression tasks, such as Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE). Choose the metric that aligns with the specific objectives of your project.

**8.Regularization:**

Apply regularization techniques like L1 (Lasso) or L2 (Ridge) Regularization to prevent overfitting.

**9.Feature Selection:**

Use techniques like feature importance scores or recursive feature elimination to identify the most relevant features for the prediction.

**10. Interpretability:**

Ensure that the model’s predictions are interpretable and explainable. This is especially important for real estate applications where stakeholders want to understand the factors affecting predictions.

**11. Deployment:**

Develop a user-friendly interface or API for end-users to input property details and receive price predictions.

**12. Continuous Improvement:**

Implement a feedback loop for continuous model improvement based on user feedback and new data.

**13. Ethical Considerations:**

Be mindful of potential biases in the data and model. Ensure fairness and transparency in your predictions.

**14. Monitoring and Maintenance:**

Regularly monitor the model’s performance in the real world and update it as needed.

**15. Innovation:**

Consider innovative approaches such as using satellite imagery or IoT data for real-time property condition monitoring, or integrating natural language processing for textual property descriptions.

**3.BUILD LOADING AND PREPROCESSING THE DATASET**

**1.Data Collection:**

Obtain a dataset that contains information about houses and their corresponding prices. This dataset can be obtained from sources like real estate websites, government records, or other reliable data providers.

**2.Load the Dataset:**

 Import relevant libraries, such as pandas for data manipulation and numpy for numerical operations.

 Load the dataset into a pandas DataFrame for easy data handling. You can use pd.read\_csv() for CSV files or other appropriate functions for different file formats.

**Program**:

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

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

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import r2\_score,

mean\_absolute\_error,mean\_squared\_error

from sklearn.linear\_model import LinearRegression

from sklearn.linear\_model import Lasso

from sklearn.ensemble import RandomForestRegressor

from sklearn.svm import SVR

import xgboost as xg

%matplotlib inline

import warnings

warnings.filterwart\_\_.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:**

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

**Output**:



**3. Data Exploration:**

Explore the dataset to understand its structure and contents.Check for the presence of missing values, outliers, and data types ofeach feature.

**4. Data Cleaning:**

Handle missing values by either removing rows with missing data or imputing values based on the nature of the data.

**5. Feature Selection:**

Identify relevant features for house price prediction. Features like the number of bedrooms, square footage, location, and amenities are often important.

We are selecting numerical features which have more than 0.50 or less than -0.50 correlation rate based on Pearson important 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

**6. Feature Engineering:**

Create new features or transform existing ones to capture additional information that may impact house prices. For example, you Can calculate the price per square foot.

**7. Data Encoding:**

Convert categorical variables (e.g., location) into numerical Format using techniques like one-hot encoding.

**8. Train-Test Split:**

Split the dataset into training and testing sets to evaluate the machine learning model’s performance.

**Program**:

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)

**4.PERFORMING DIFFERENT ACTIVITIES LIKE FEATURE ENGINEERING, MODEL TRAINING,EVALUATION etc.,**

**1.Feature Engineering:**

 As mentioned earlier, feature engineering is crucial. It involves Creating new features or transforming existing ones to provide Meaningful information for your model.

 Extracting information from textual descriptions (e.g., presence of keywords like “pool” or “granite countertops”).

 Calculating distances to key locations (e.g., schools, parks) if you have location data.

**2.Data Preprocessing & Visualisation:**

Continue data preprocessing by handling any remaining Missing values or outliers based on insights from your data exploration.

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>



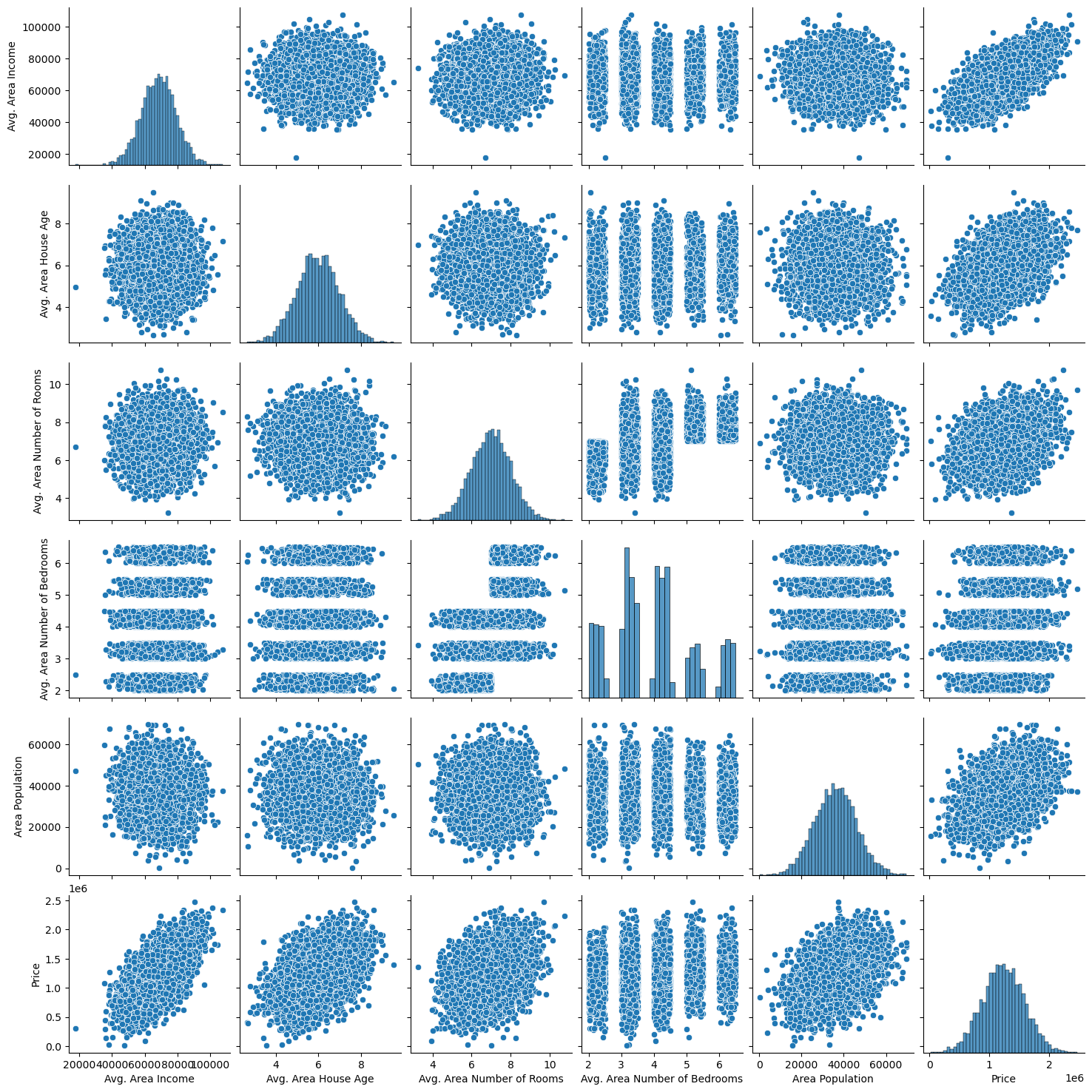
In [5]:

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)



from sklearn.ensemble import RandomForestRegressor

from sklearn.svm import SVR

import xgboost as xg

%matplotlib inline

Visualising Correlation:

In [7]:

dataset.corr(numeric\_only=True)

Out[7]:

In [8]:

plt.figure(figsize=(10,5))sns.heatmap(dataset.corr(numeric\_only = Tru

e), annot=True)

Out[8]:

<Axes: >



3. Model Selection:

Choose an appropriate machine learning model for your

regression task. Common choices include:

 Linear Regression

 Decision Trees

 Random Forest

 Gradient Boosting (e.g., XGBoost or LightGBM)

 Neural Networks (Deep Learning)

Program:

Importing Dependencies

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

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

Model 1 - Linear Regression

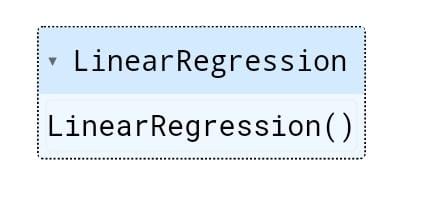
In [1]:

model\_lr=LinearRegression()

In [2]:

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

Out[2]:



Predicting Prices

In [3]:

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

Evaluation of Predicted Data

In [4]:

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 Tr

end')

plt.xlabel('Data')

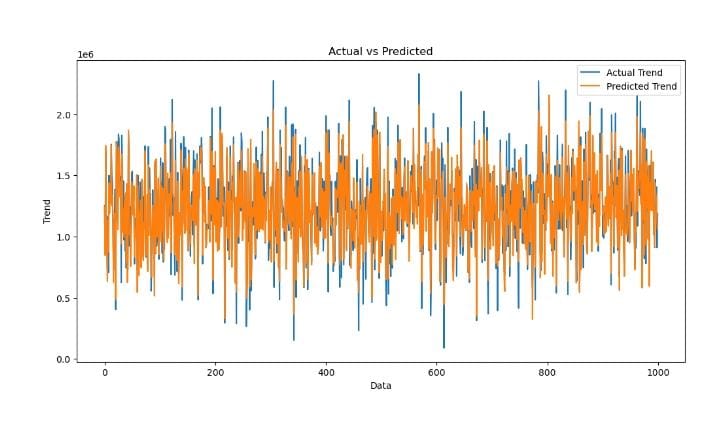
plt.ylabel('Trend')

plt.legend()

plt.title('Actual vs Predicted')

Out[4]:

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

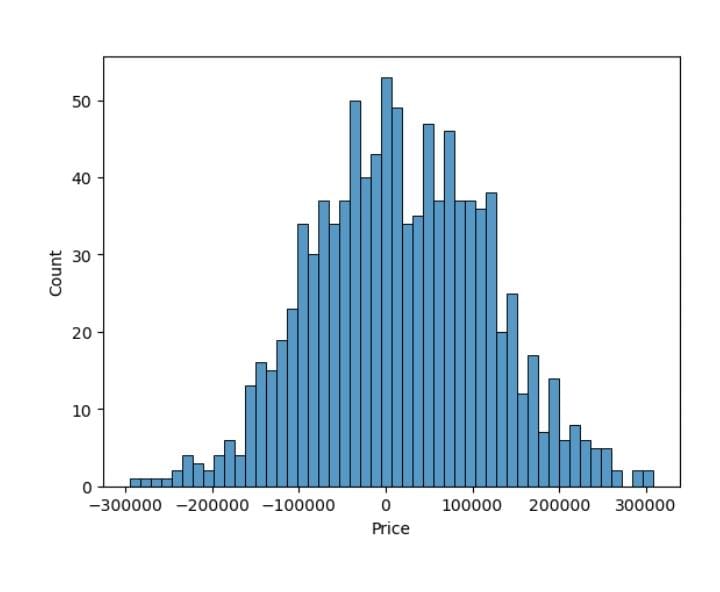


In [5]:

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

Out[5]:

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



In [6]:

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

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

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

Out[6]:

0.9182928179392918

82295.49779231755

10469084772.975954

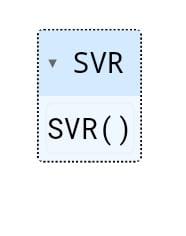
Model 2 - Support Vector Regressor

In [7]:

model\_svr = SVR()

In [8]:

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

Out[8]:

Predicting Prices

In [9]:

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

Evaluation of Predicted Data

In [10]:

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 Tr

end')

plt.xlabel('Data')

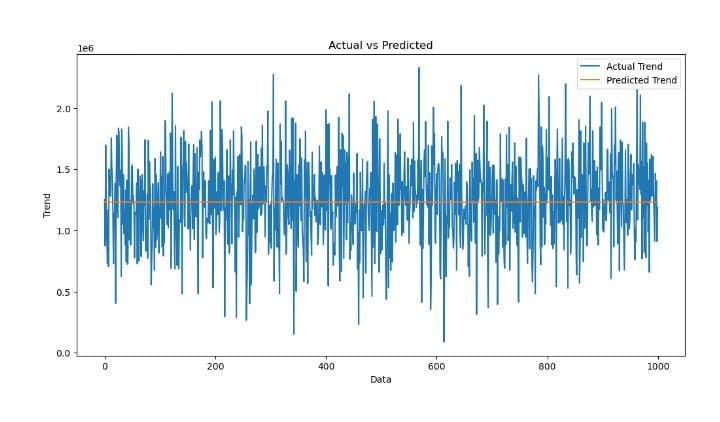
plt.ylabel('Trend')

plt.legend()

plt.title('Actual vs Predicted')

Out[10]:

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

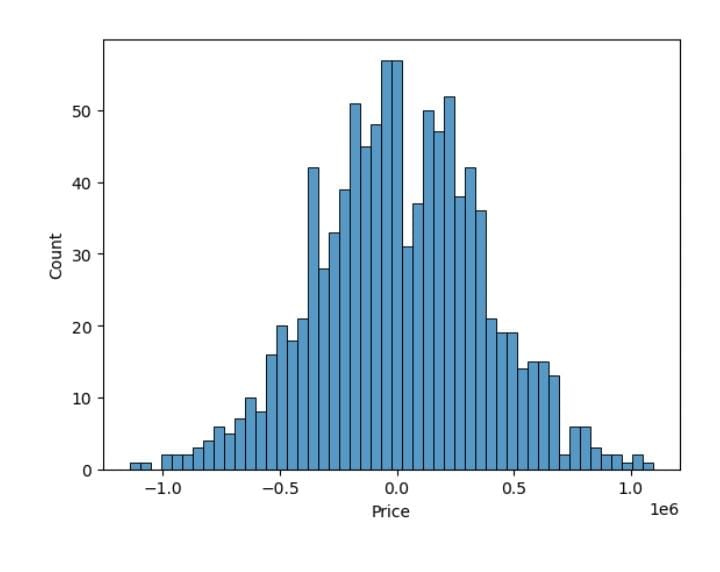


In [11]:

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

Out[12]:

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



In [12]:

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

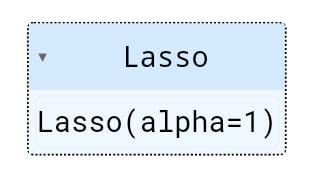
In [13]:

model\_lar = Lasso(alpha=1)

In [14]:

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

Out[14]:



Predicting Prices

In [15]:

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

Evaluation of Predicted Data

In [16]:

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 Tr

end')

plt.xlabel('Data')

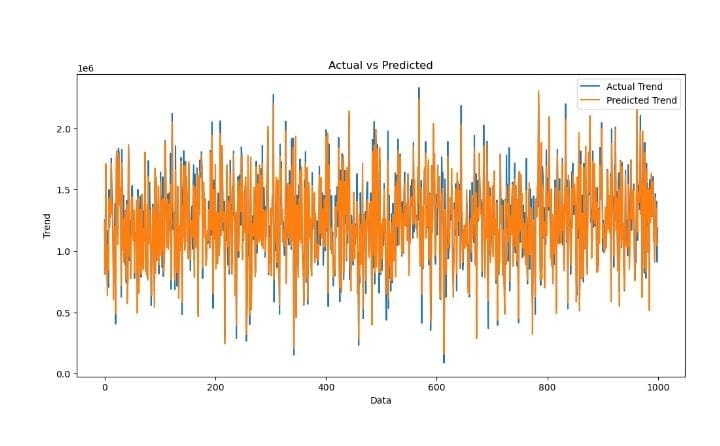
plt.ylabel('Trend')

plt.legend()

plt.title('Actual vs Predicted')

Out[16]:

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



In [17]:

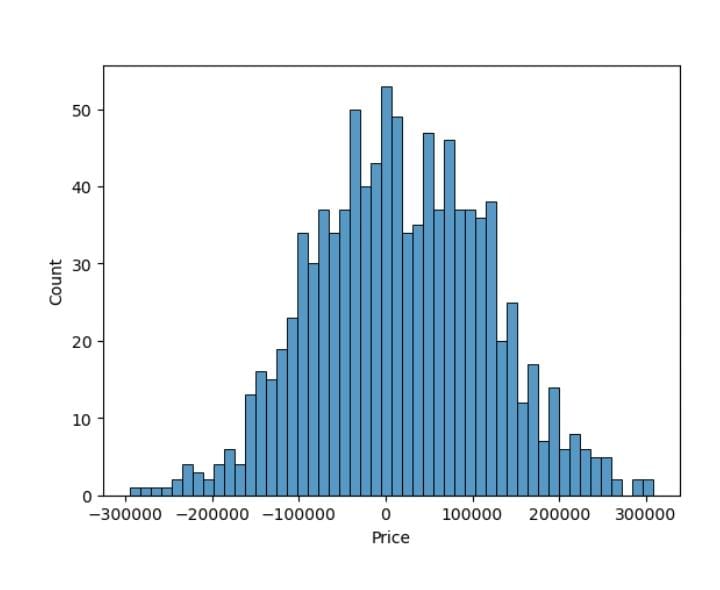
sns.histplot((Y\_test-Prediction3), bins=50)

In [17]:

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

Out[17]:

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



In [18]:

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

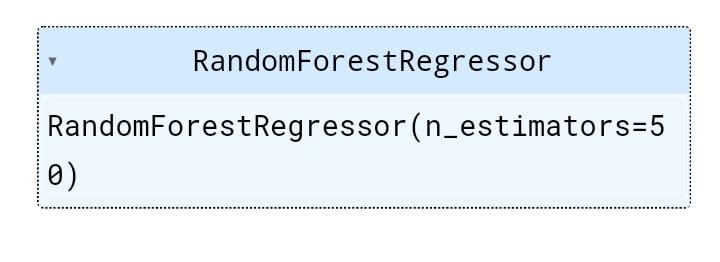
In [19]:

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

In [20]:

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

Out[20]:



Predicting Prices

In [21]:

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

Model 5 - XGboost Regressor

In [25]:

model\_xg = xg.XGBRegressor()

In [26]:

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

Out[26]:

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, ...)

4. Model Training:

Split your dataset into training and testing sets (as shown earlier)

and train the selected model on the training data. Here's an example

using Linear Regression:

5. Model Evaluation:

Evaluate your model's performance using appropriate regression

metrics, such as Mean Absolute Error (MAE), Mean Squared Error

(MSE), and Root Mean Squared Error (RMSE). For example:

PYTHON PROGRAM:

# Import necessary libraries

from sklearn.feature\_selection import SelectKBest, f\_regression

from sklearn.linear\_model import LinearRegression

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_squared\_error, r2\_score

import numpy as np

selector = SelectKBest(score\_func=f\_regression, k=k)

X\_train\_selected = selector.fit\_transform(X\_train, y\_train)

# Model Selection

# Let's choose both Linear Regression and Random Forest Regressor for

comparison.

linear\_reg\_model = LinearRegression()

random\_forest\_model = RandomForestRegressor(n\_estimators=100,

random\_state=42)

# Train the models on the selected features

linear\_reg\_model.fit(X\_train\_selected, y\_train)

random\_forest\_model.fit(X\_train\_selected, y\_train)

# Evaluate the models on the test set

X\_test\_selected = selector.transform(X\_test)

# Make predictions

linear\_reg\_predictions = linear\_reg\_model.predict(X\_test\_selected)

random\_forest\_predictions =

random\_forest\_model.predict(X\_test\_selected)

# Evaluate model performance

def evaluate\_model(predictions, model\_name):

mse = mean\_squared\_error(y\_test, predictions)

r2 = r2\_score(y\_test, predictions)

print(f"{model\_name} Model Evaluation:")

print(f"Mean Squared Error (MSE): {mse}")

print(f"R-squared (R2) Score: {r2}\n")

# Performance Measure

elr\_mse = mean\_squared\_error(y\_test, pred)

elr\_rmse = np.sqrt(lr\_mse)

elr\_r2 = r2\_score(y\_test, pred)

# Show Measures

result = '''

MSE : {}

RMSE : {}

R^2 : {}

'''.format(lr\_mse, lr\_rmse, lr\_r2)

print(result)

# Model Comparision

names.append("elr")

mses.append(elr\_mse)

rmses.append(elr\_rmse)

r2s.append(elr\_r2)

evaluate\_model(linear\_reg\_predictions, "Linear Regression")

evaluate\_model(random\_forest\_predictions, "Random Forest Regressor")

OUTPUT:

Linear Regression Model Evaluation:

Mean Squared Error (MSE): 10089009300.893988

R-squared (R2) Score: 0.9179971706834331

Random Forest Regressor Model Evaluation:

Mean Squared Error (MSE): 14463028828.265167

R-squared (R2) Score: 0.8824454166872736

MSE : 10141766848.330585

RMSE : 100706.33966305491

R^2 : 0.913302484308253

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]:

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()

Evaluation of Predicted Data

In [22]:

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 Tr

end')

plt.xlabel('Data')

plt.ylabel('Trend')

plt.legend()

plt.title('Actual vs Predicted')

Out[22]:

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

In [23]:

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

Out[23]:

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

In [24]:

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

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

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

Out [24] :

-0.0006222175925689744

286137.81086908665

128209033251.4034

6. Hyperparameter Tuning:

Optimize the model's hyperparameters to improve its

performance. Depending on the model, you can use techniques like grid

search or random search.

7. Cross-Validation:

Implement cross-validation to ensure that your model's

performance is consistent across different subsets of your data. This

helps prevent overfitting.

8. Regularization:

Apply regularization techniques like L1 (Lasso) or L2 (Ridge)

if needed to prevent overfitting and improve model generalization.

Feature Selection:

Use feature importance scores from your model or techniques

like recursive feature elimination to identify the most important features

for predictions.

Interpretability:

Ensure that the model's predictions are interpretable and

explainable. Stakeholders may want to understand how each feature

impacts the predicted house price.

Deployment:

Deploy your trained model in a real-world setting, whether it's

through a web application, API, or any other user-friendly interface.

Users can input property details, and the model provides price

predictions.

Monitoring and Maintenance:

Continuously monitor the model's performance and update it as

needed. Real estate markets change, so it's essential to retrain the model

with new data periodically.

Ethical Considerations:

Ensure that your model doesn't introduce or perpetuate biases

in pricing. Implement fairness and transparency measures.

Innovation:

Explore innovative approaches such as incorporating external

data sources (e.g., satellite imagery, IoT data) for better predictions.

ADVANTAGES:

Predicting house prices using machine learning offers several

significant advantages:

1.Accuracy:

Machine learning models can process and analyze vast amounts

of data, including various property and market factors. This results in

more accurate house price predictions compared to traditional methods

that rely on a limited set of variables.

2.Complex Data Handling:

Machine learning algorithms can handle complex, non-linear

relationships in the data. They can recognize patterns and interactions

among different features, allowing for a more comprehensive assessment

of a property's value.

3.Continuous Learning:

Machine learning models can be continually updated with new

data, enabling them to adapt to changing market conditions and trends.

This ensures that predictions remain relevant and up-to-date.

4.Efficiency:

Automated valuation models powered by machine learning

can evaluate properties rapidly. This efficiency is beneficial for both

property appraisers and individuals looking to determine the value of a

property quickly.

5. Data Integration:

Machine learning models can incorporate a wide range of data

sources, including property characteristics, neighborhood information,

economic indicators, and even social trends. This holistic approach

provides a more complete picture of the factors influencing house prices.

6.Reduced Bias:

Machine learning can help reduce human bias in property

valuation. It evaluates properties objectively based on data, which can

lead to fairer and more consistent pricing.

7.Market Insights:

By analyzing historical data and current market conditions,

machine learning can offer valuable insights into market trends, helping

investors and developers make informed decisions.

8.Risk Assessment:

Machine learning can assess the risk associated with a

property, which is crucial for mortgage lenders and investors. It helps

identify potential issues or opportunities related to a property's value.

9. Transparency:

Machine learning models can provide clear and transparent

explanations for their predictions, which is essential for building trust

among stakeholders in the real estate market.

10. Scalability:

Machine learning models can be deployed at scale, making it

possible to assess property values in large real estate portfolios, entire

neighborhoods, or even across entire cities.

11. Time and Cost Savings:

Using machine learning for property valuation can save time

and reduce costs associated with manual appraisals, making it an

efficient and cost-effective solution for both businesses and individuals.

12. Customization:

Machine learning models can be customized to cater to

specific markets, types of properties, or regional variations, allowing for

more tailored and precise predictions.

DISADVANTAGES:

While predicting house prices using machine learning offers

numerous advantages, it also comes with several disadvantages and

challenges:

1.Data Quality:

Machine learning models heavily rely on data quality. Inaccurate

or incomplete data can lead to unreliable predictions. Ensuring the data

used for training and evaluation is of high quality is essential.

2. Overfitting:

Machine learning models can be prone to overfitting, where

they perform exceptionally well on the training data but struggle with

new, unseen data. This can result in overly optimistic or inaccurate

predictions

3.Data Privacy and Security:

Handling sensitive property and financial data for training

models raises privacy and security concerns. Protecting this information

from unauthorized access and breaches is critical.

4.Model Interpretability:

Some machine learning models, such as deep neural networks,

can be challenging to interpret. Understanding why a model makes a

specific prediction is crucial for trust and accountability.

5. Bias and Fairness:

Machine learning models can inherit biases present in the

training data, potentially leading to unfair or discriminatory predictions,

especially in areas where historical biases exist in real estate practices.

6. Lack of Transparency:

While some machine learning models offer interpretability,

others are considered "black boxes," making it difficult to understand the

logic behind their predictions. This can be a barrier to trust and

regulatory compliance.

7. Maintenance and Updates:

Machine learning models require ongoing maintenance and

updates to remain accurate and relevant. This includes updating them

with new data and retraining as market conditions change.

8. High Computational Requirements:

Training and running sophisticated machine learning models

can demand significant computational resources, which can be costly

and require advanced infrastructure.

9. Cost of Implementation:

Integrating machine learning into real estate workflows can be

expensive, particularly for smaller businesses or organizations that lack

the resources for extensive data science and engineering teams.

10. Market Volatility:

Machine learning models may not always perform well during

times of extreme market volatility or significant economic shifts, as they

rely on historical data for predictions.

11. Legal and Regulatory Compliance:

The use of machine learning in real estate must comply with

various legal and regulatory standards. Ensuring that models adhere to

fair housing laws and other regulations is crucial.

12. Limited Data Availability:

In some regions or for certain property types, high-quality

data may be limited, making it challenging to build accurate models.

13. Human Expertise:

While machine learning can enhance the valuation process, it

doesn't eliminate the need for human expertise entirely. Appraisers and

real estate professionals are still crucial for verifying model predictions

and considering unique property characteristics.

14. Model Degradation:

Over time, machine learning models may lose accuracy due

to shifts in market dynamics, and retraining is necessary to maintain

performance.

**CONCLUSION**:

Predicting house prices using machine learning is a transformative

and promising approach that has the potential to revolutionize the real

estate industry. Throughout this exploration, we have uncovered the

remarkable capabilities of machine learning in providing more accurate,

data-driven, and nuanced predictions for property values. As we

conclude, several key takeaways and implications emerge:

Improved Accuracy: Machine learning models consider a myriad of

variables, many of which may be overlooked by traditional methods.

This results in more accurate predictions, benefiting both buyers and

sellers who can make informed decisions based on a property's true

value.

Data-Driven Insights: These models provide valuable insights into the

real estate market by identifying trends, neighborhood characteristics,

and other factors that influence property prices. This information can be

invaluable for investors, developers, and policymakers seeking to make

strategic decisions.

Market Efficiency: The increased accuracy in pricing predictions can

lead to a more efficient real estate market, reducing overvaluation and

undervaluation of properties. This contributes to a fairer and more

transparent marketplace.

Challenges and Considerations: Machine learning for house price

prediction is not without its challenges. Data quality, model

interpretability, and ethical concerns are important considerations.

Addressing these issues is crucial for the responsible and ethical

deployment of this technology.

Continual Advancement: The field of machine learning is continually

evolving, and as it does, so will the accuracy and capabilities of

predictive models. As more data becomes available and algorithms

improve, we can expect even more sophisticated predictions in the future.

In conclusion, the application of machine learning in

predicting house prices is a groundbreaking development with far-

reaching implications. It empowers individuals, businesses, and

governments to navigate the real estate market with more confidence

and precision. However, it is essential to approach this technology with a

clear understanding of its potential and limitations, ensuring that its

benefits are harnessed responsibly for the betterment of the real estate

industry and society as a whole. As machine learning continues to

advance, we can look forward to a future where property valuation

becomes increasingly precise and data-informed.