PREDICTING HOUSE PRICE USING MACHINE LEARNING

TEAM MEMBER

210221104029 : SARAVANAN.S

Phase 5 submission document

Project Title: House Price Predictor

Phase 5: Project Documentation & Submission

Topic: *In this section we will document the completeproject and prepare it for submission.*

House Price Prediction

Introduction:

- The real estate market is a dynamic and complex arena, where property values can fluctuate significantly due to a multitude of factors. For both homebuyers and sellers, accurately determining thefair 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 ofits most compelling applications is predicting house prices with remarkable accuracy.
- Traditional methods of property valuation, relying on factors such aslocation, square footage, and recent sales data, are undoubtedly useful. However, they often fall short in capturing the intricacies andnuances 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-drivenpredictions.
- 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 datato create predictive models that consider an array of variables, such as neighborhood characteristics, property features, economic indicators, and even social trends.
-

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

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

5000 Rows x 7 Columns

Here's a list of tools and software commonly used in theprocess:

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, GoogleColab, 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 dataexploration and visualization.

5. Data Preprocessing Tools:

- Libraries like pandas help with data cleaning, manipulation, andpreprocessing.

6. Data Collection and Storage:

- Depending on your data source, you might need web scrapingtools *(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 Notebooksor Markdown for creating *README* files and documentation.

9. Hyperparameter Tuning:

- Tools like GridSearchCV or RandomizedSearchCV fromscikit-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 beuseful.

1.DESIGN THINKING AND PRESENT IN FORMOF DOCUMENT

1. Empathize:

► Understand the needs and challenges of all stakeholders involved inthe 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 fortheir decision-making.

2. Define:

- Clearly articulate the problem statement, such as "How might we predict house prices more accurately and transparently using machinelearning?"
- Identify the key goals and success criteria for the project, such as increasing prediction accuracy, reducing bias, or improving user trustin the valuation process.

3. Ideate:

- Brainstorm creative solutions and data sources that can enhance theaccuracy and transparency of house price predictions.
- Encourage interdisciplinary collaboration to generate a wide range ofideas, including the use of alternative data, new algorithms, or improved visualization techniques.

4. Prototype:

- Create prototype machine learning models based on the ideasgenerated during the ideation phase.
- Test and iterate on these prototypes to determine which approachesare most promising in terms of accuracy and usability.

2.DESIGN INTO INNOVATION

1. Data Collection:

Gather a comprehensive dataset that includes features such aslocation, 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 numericalfeatures as necessary.

PYHON PROGRAM:

```
# Import necessary libraries
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import
train_test_splitfrom 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:

```
   Avg. Area Income Avg. Area House Age  Avg. Area Number of Roo
ms \
0     79545.458574          5.682861              7.009188
1     79248.642455          6.002900              6.730821
2     61287.067179          5.865890              8.512727
3     63345.240046          7.188236              5.586729
4     59982.197226          5.040555              7.839388

   Avg. Area Number of Bedrooms  Area Population     Price \
0                     4.09    23086.80050  1.059034e+0
                                        3            6
1                     3.09    40173.07217  1.505891e+0
                                        4            6
2                     5.13    36882.15940  1.058988e+0
                                        0            6
```

3	3.26	34310.242831 1.260617e+06
4	4.23	26354.109472 6.309435e+05

Address

0 208 Michael Ferry Apt. 674\nLaurabury, NE 3701...

1 188 Johnson Views Suite 079\nLake Kathleen, CA...

2 9127 Elizabeth Stravenue\nDanieltown, WI 06482...

3 USS Barnett\nFPO AP 44820

4 USNS Raymond\nFPO AE 09386

Preprocessed Data:

[[-0.19105816 -0.13226994 -0.13969293 0.12047677 -0.83757985 -1.0
0562872]

[-1.3945016 0.42786736 0.79541275 -0.55212509 1.15729018 9 1.61
946754]

[-0.3513786 0.46394489 1.70199509 0.03133676 -0.32671213 5 1.63
886651]

[-0.1394414 0.1104872 0.22289331 -0.75471601 -0.90401197 3 -1.54
810704]

[2.20969666 0.42984356 -0.45488144 0.12566216 0.62516685 0.98
830821]]

4227 1.094880e+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 valuable information. For example, you can calculate the distance to thenearest public transportation, or create a feature for the overall conditionof 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, GradientBoosting, and Neural Networks.*

5. Training:

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

3.BUILD LOADING AND PREPROCESSING THE DATASET

1. Data Collection:

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

2. Load the Dataset:

- Import relevant libraries, such as pandas for data manipulation andnumpy 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_splitfrom sklearn.preprocessing

import StandardScaler

```
from sklearn.metrics import r2_score,
mean_absolute_error,mean_squared_e
rror

from sklearn.linear_model import

LinearRegressionfrom sklearn.linear_model

import Lasso

from sklearn.ensemble import

RandomForestRegressorfrom sklearn.svm import

SVR

import xgboost as xg

%matplotlib

inlineimport

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
forthis 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:

	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 ?

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 missingdata or imputing values based on the nature of the data.

5. Feature Selection:

Identify relevant features for house price prediction. Features likethe 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 Correlation Method—which is the default value of parameter "method" in corr() function. As for selecting categorical features, Iselected 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

0dtype: int64

-________-_-____________

TOTAL MISSING VALUES: 0

Program:

```
X = df.drop('price', axis=1) #

Featuresy = 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 involvescreating new features or transforming existing ones to provide meaningful information for your model.
- Extracting information from textual descriptions *(e.g., presence ofkeywords like "pool" or "granite countertops").*
- Calculating distances to key locations *(e.g., schools, parks)* if youhave 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)
```

Visualising Correlation:

In [7]:

```
dataset.corr(numeric_only=True)
```

Out[7]:

	Avg. Area Income	Avg. Area HouseAge	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 ofRooms	-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), annot=True)
```

Out[8]:

<Axes: >

3. Model Selection:

Choose an appropriate machine learning model for yourregression task. *Common choices include:*

- ✓ Linear Regression
- ✓ Decision Trees
- ✓ Random Forest
- ✓ Gradient Boosting *(e.g., XGBoost or LightGBM)*
- ✓ Neural Networks (Deep Learning)

Program:

Importing

Dependenciesimport

pandas as pd import

numpy as np import

seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model_selection import

train_test_splitfrom sklearn.preprocessing

import StandardScaler

from sklearn.metrics import r2_score,
mean_absolute_error,mean_squared_e
rror

from sklearn.linear_model import

```
from sklearn.ensemble import

RandomForestRegressorfrom 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 requiredfor 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]:

```
▾ LinearRegression
LinearRegression()
```

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 Trend')
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]:

```
▾ SVR
SVR()
```

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 Trend')

    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]:

```
▾ Lasso
Lasso(alpha=1)
```

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 Trend')

    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)
```

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]:

```
RandomForestRegressor
RandomForestRegressor(n_estimators=50)
```

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 regressionmetrics, 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_regressionfrom sklearn.linear_model import

LinearRegression

from sklearn.ensemble import

RandomForestRegressor from sklearn.metrics import

mean_squared_error, r2_scoreimport 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
forcomparison.

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_selec
ted)

# 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_rm
se)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):
10089009300.893988R-squared (R2) Score:
0.9179971706834331

Random Forest Regressor Model Evaluation:

Mean Squared Error (MSE):
14463028828.265167R-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 themodel is.

In [30]:

```
models.sort_values(by="RMSE (Cross-Validation)")
```

Out[30]:

	Model	MAE	MSE	RMSE	R2 Score	RMSE (Cross-Validation)
6	XGBRegressor	1.743992e+04	7.165790e+08	2.676899e+04	9.065778e-01	29698.849618
4	SVR	1.784316e+04	1.132136e+09	3.364723e+04	8.524005e-01	30745.475239
5	RandomForestRegressor	1.811511e+04	1.004422e+09	3.169262e+04	8.690509e-01	31138.863315
1	Ridge	2.343550e+04	1.404264e+09	3.747351e+04	8.169225e-01	35887.852792
2	Lasso	2.356046e+04	1.414338e+09	3.760768e+04	8.156092e-01	35922.769369
0	LinearRegression	2.356789e+04	1.414931e+09	3.761557e+04	8.155318e-01	36326.451445
7	Polynomial Regression (degree=2)	2.382228e+15	1.513991e+32	1.230443e+16	-1.973829e+22	36326.451445

	Model	MAE	MSE	RMSE	R2 Score	RMSE (Cross-Validation)
3	ElasticNet	2.379274e+04	1.718446e+09	4.145414e+04	7.759618e-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()
```

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 Trend')

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
```

ADVANTAGES:

Predicting house prices using machine learning offers severalsignificant advantages:

1.Accuracy:

Machine learning models can process and analyze vast amountsof 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 assessmentof a property's value.

3.Continuous Learning:

Machine learning models can be continually updated with newdata, 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 aproperty quickly.

5. Data Integration:

Machine learning models can incorporate a wide range of datasources, 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.

DISADVANTAGES:

While predicting house prices using machine learning offersnumerous advantages, it also comes with several disadvantages and challenges:

1.Data Quality:

Machine learning models heavily rely on data quality. Inaccurateor 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 withnew, 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 informationfrom 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.

BENEFITS:

Predicting house prices using machine learning offers a wide range of benefits, which can positively impact various stakeholders in the real estate industry and beyond. Here are some of the key benefits ofusing machine learning for house price prediction:

1. Accuracy:

Machine learning models can provide more accurate property valuations by considering a broader set of variables and patterns withinthe data, leading to more precise price predictions.

2. Data-Driven Insights:

Machine learning models uncover valuable insights into the real estate market by identifying trends, factors influencing property values, and neighborhood characteristics. This information can inform strategic decisions for investors, developers, and policymakers.

3. Efficiency:

Automated valuation models powered by machine learning canrapidly assess property values, saving time and effort for appraisers and individuals looking to determine a property's worth quickly.

4. Continuous Learning:

Machine learning models can adapt to changing market conditions and incorporate new data, ensuring that predictions remainrelevant and up-to-date over time.

5. Market Transparency:

Machine learning can contribute to a more transparent andefficient real estate market by reducing overvaluation and undervaluation, thereby promoting fair pricing and reducing market inefficiencies.

6. Risk Assessment:

Machine learning can evaluate the risk associated with a property, which is crucial for mortgage lenders, insurers, and investors.It helps identify potential issues or opportunities related to a property's value.

7. Customization:

Machine learning models can be tailored to specific markets,property types, or regional variations, enabling more accurate and context-specific predictions.

8. Cost Savings:

Using machine learning for property valuation can reduce thecosts associated with manual appraisals, benefiting both businesses and individuals in terms of appraisal expenses.

CONCLUSION:

Predicting house prices using machine learning is a transformativeand 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 ofvariables, 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 thereal estate market by identifying trends, neighborhood characteristics, and other factors that influence property prices. This information can beinvaluable for investors, developers, and policymakers seeking to makestrategic decisions.

Market Efficiency: The increased accuracy in pricing predictions canlead to a more efficient real estate market, reducing overvaluation andundervaluation of properties. This contributes to a fairer and more transparent marketplace.

In conclusion, the application of machine learning in predictng 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.