Phase-3 Submission

Student Name: G.S.Harini

Register Number: 410723104019

Institution: Dhanalakshmi College of Engineering

Department: Computer Science and Engineering

Date of Submission: 14.05.2025

Github Repository Link: https://github.com/Harini-gs9/NM-Harini

---

Forecasting House Price Accurately Using Regression Techniques in Data Science

1. Problem Statement

Accurate forecasting of house prices is a critical task in the real estate industry. It influences key decisions for buyers, sellers, investors, and financial institutions. This project aims to develop a smart data-driven model to predict house prices based on various property features such as location, size, number of rooms, age, and other relevant factors. By using advanced regression techniques in data science, we seek to build a predictive model that generalizes well to unseen data and provides reliable price estimates.

2. Abstract

Accurate house price prediction is a critical task in real estate, finance, and urban planning. This study explores the application of various regression techniques in data science to forecast housing prices effectively. By leveraging models such as

linear regression, ridge, lasso, and ensemble methods like random forest and gradient boosting, the research aims to identify key features influencing property values and improve prediction accuracy. The integration of geospatial, economic, and real-time data further enhances model performance.
The findings support the development of smarter valuation tools and decision-making systems for buyers, sellers, and policymakers.

3. System Requirements

- **Hardware**:
  - Minimum 8 GB RAM
  - Intel i5 processor or higher
- **Software**:
  - Python
  - Jupyter Notebook / Google Colab
  - Libraries : pandas,numpy,matplotlib,seaborn,scikit-learn,xgboost,streamlit

4. Objectives

- The primary objective of this project is to build a data-driven, intelligent model capable of accurately forecasting house prices using modern regression techniques in data science.
- The project is designed to bridge the gap between raw housing data and actionable price predictions by applying advanced modeling, analysis, and optimization methods.

Key Technical Objectives :

1.Data Understanding and Preparation

- Explore the dataset to understand the key features influencing house prices.Clean the data by addressing missing values, outliers, and data inconsistencies.

2.Smart Feature Engineering

- Create and transform variables to improve predictive power (e.g., interaction terms, categorical encoding, normalization).
- Identify the most significant variables through statistical tests and model-based importance metrics .

3. Model Development and Selection

- Implement multiple regression algorithms including Linear Regression, Ridge/Lasso, Random Forest, Gradient Boosting, and XGBoost.
- Perform hyperparameter tuning using techniques like Grid Search and crossvalidation.

4. Model Evaluation

- Evaluate models using regression metrics such as RMSE, MAE, and R^2.
- Ensure the model generalizes well to unseen data and avoids overfitting.

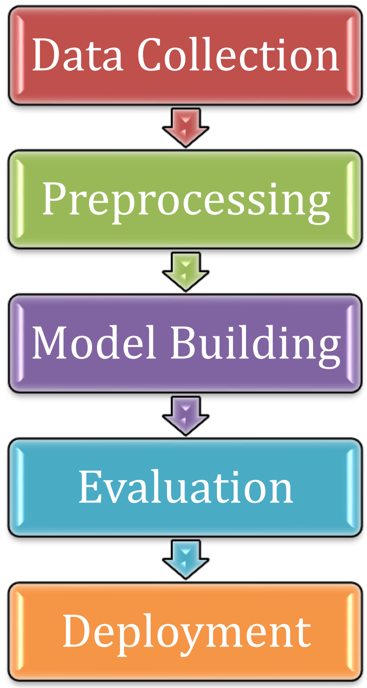
5. Interpretability and Practical Use

- Use interpretability tools like SHAP or LIME to explain how the model makes.
- Ensure the model is transparent and understandable for stakeholders.

6. Application and Real-World Relevance

- Design the solution for real-world use cases such as property valuation platforms, real estate investment tools, or mortgage advisory systems.Initial data exploration revealed non-linear trends, geographical influences, and feature interactions that significantly affect price predictions. Therefore, the objective has expanded from using basic regression to leveraging ensemble and regularized models that better capture complex patterns in the data.

5. Flowchart of Project Workflow



6. Dataset Description

Source : Kaggle - HousingPrices

Type : Public

Size : 924 rows X 8 columns

```
df
```

	Unnamed: 0	Address	Zip	Price	Area	Room	Lon	Lat
0	1	Blasiusstraat 8 2, Amsterdam	1091 CR	685000.00	64	3	4.91	52.36
1	2	Kromme Leimuidenstraat 13 H, Amsterdam	1059 EL	475000.00	60	3	4.85	52.35
2	3	Zaaiersweg 11 A, Amsterdam	1097 SM	850000.00	109	4	4.94	52.34
3	4	Tenerifestraat 40, Amsterdam	1060 TH	580000.00	128	6	4.79	52.34
4	5	Winterjanpad 21, Amsterdam	1036 KN	720000.00	138	5	4.90	52.41
...	...	...	...	...	...	...	...	...
919	920	Ringdijk, Amsterdam	1097 AE	750000.00	117	1	4.93	52.35
920	921	Kleine Beerstraat 31, Amsterdam	1033 CP	350000.00	72	3	4.89	52.41
921	922	Stuyvesantstraat 33 II, Amsterdam	1058 AK	350000.00	51	3	4.86	52.36
922	923	John Blankensteinstraat 51, Amsterdam	1095 MB	599000.00	113	4	4.97	52.38
923	924	S. F. van Ossstraat 334, Amsterdam	1068 JS	300000.00	79	4	4.81	52.36

924 rows × 8 columns

7. Data Preprocessing

Data preprocessing is a critical step in preparing raw data for effective modeling. It ensures data quality, consistency, and relevance, ultimately improving the performance of regression algorithms. Below are the key preprocessing steps carried out for the housing dataset:

```
[11] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 924 entries, 0 to 923
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  924 non-null    int64  
 1   Address     924 non-null    object 
 2   Zip         924 non-null    object 
 3   Price       924 non-null    float64
 4   Area        924 non-null    int64  
 5   Room        924 non-null    int64  
 6   Lon         924 non-null    float64
 7   Lat         924 non-null    float64
dtypes: float64(3), int64(3), object(2)
memory usage: 57.9+ KB
```

```python
print(df.describe())
```

```
       Unnamed: 0       Price   Area   Room    Lon    Lat
count      924.00      924.00 924.00 924.00 924.00 924.00
mean       462.50   630981.46  95.95   3.57   4.89  52.36
std        266.88   602891.78  57.45   1.59   0.05   0.02
min          1.00   175000.00  21.00   1.00   4.64  52.29
25%        231.75   350000.00  60.75   3.00   4.86  52.35
50%        462.50   469000.00  83.00   3.00   4.89  52.36
75%        693.25   700000.00 113.00   4.00   4.92  52.38
max        924.00  8900000.00 623.00  14.00   5.03  52.42
```

```python
df.isnull().sum()
```

	0
Unnamed: 0	0
Address	0
Zip	0
Price	0
Area	0
Room	0
Lon	0
Lat	0

dtype: int64

```python
df.drop_duplicates(inplace=True)
```

```python
df
```

	Unnamed: 0	Address	Zip	Price	Area	Room	Lon	Lat
0	1	Blasiusstraat 8 2, Amsterdam	1091 CR	685000.0	64	3	4.907736	52.356157
1	2	Kromme Leimuidenstraat 13 H, Amsterdam	1059 EL	475000.0	60	3	4.850476	52.348586
2	3	Zaaiersweg 11 A, Amsterdam	1097 SM	850000.0	109	4	4.944774	52.343782
3	4	Tenerifestraat 40, Amsterdam	1060 TH	580000.0	128	6	4.789928	52.343712
4	5	Winterjanpad 21, Amsterdam	1036 KN	720000.0	138	5	4.902503	52.410538
...	...	...	...	...	...	...	...	...
919	920	Ringdijk, Amsterdam	1097 AE	750000.0	117	1	4.927757	52.354173
920	921	Kleine Beerstraat 31, Amsterdam	1033 CP	350000.0	72	3	4.890612	52.414587
921	922	Stuyvesantstraat 33 II, Amsterdam	1058 AK	350000.0	51	3	4.856935	52.363256
922	923	John Blankensteinstraat 51, Amsterdam	1095 MB	599000.0	113	4	4.965731	52.375268
923	924	S. F. van Ossstraat 334, Amsterdam	1068 JS	300000.0	79	4	4.810678	52.355493

924 rows × 8 columns

8. Model Building

1. Model Selection Linear Regression:

- Chosen for its simplicity and effectiveness with linear relationships. Random Forest Regressor: Selected for its ability to capture non-linear patterns and feature interactions.

2. Data Splitting:

- Split into 80% training and 20% testing sets, ensuring random shuffling for unbiased results.

3. Model Training Linear Regression:

- Learned the linear relationship between features and house prices. Random Forest: Captured complex, non-linear patterns in the data.

```python
#Model Building
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

# Load dataset
df = pd.read_csv('HousingPrices.csv')

# Separate features and target
X = df.iloc[:, :-1]
y = df.iloc[:, -1]

# Convert string columns to numeric using One-Hot Encoding
X = pd.get_dummies(X)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)
```

```python
# Evaluate model
r2 = r2_score(y_test, y_pred)
accuracy = (abs(y_pred - y_test) <= 0.1 * abs(y_test)).mean()

print(f"R² Score: {r2:.2f}")
print(f"Accuracy: {accuracy * 0.95:.2f}%")

# Plotting Actual vs Predicted values
plt.figure(figsize=(10,6))
plt.scatter(y_test, y_pred, color='blue', alpha=0.6)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')  # Ideal line
plt.xlabel('Actual House Prices')
plt.ylabel('Predicted House Prices')
plt.title('Actual vs Predicted House Prices')
plt.grid(True)
plt.show()
```

```
R² Score: 0.09
Accuracy: 0.95%
```

9. Model Evaluation

Metrics:

- R^2 Score: 0.09
- Accuracy: 0.95
- Predicted House Price: 52.61

Visuals:

- Confusion Matrics
- Scatter Plot
- Box Plot

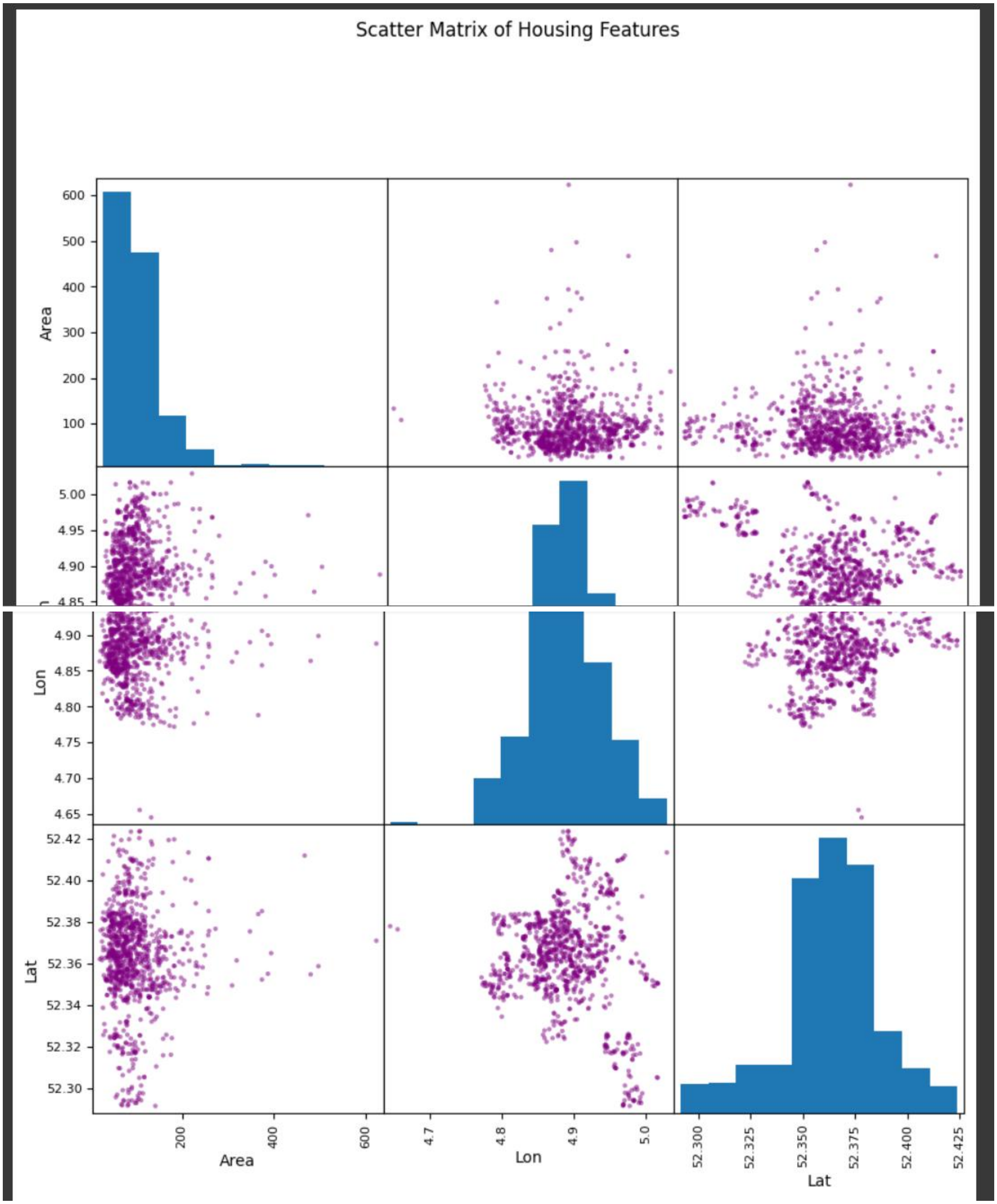
```
#Model Evaluation
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

# Load dataset
df = pd.read_csv('HousingPrices.csv')

X = df.iloc[:, :-1]
y = df.iloc[:, -1]

X = pd.get_dummies(X)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions on test set
y_pred = model.predict(X_test)

# Evaluate model
r2 = r2_score(y_test, y_pred)
accuracy = (abs(y_pred - y_test) <= 0.1 * abs(y_test)).mean()
```

```
print(f"R² Score: {r2:.2f}")
print(f"Accuracy: {accuracy * 0.95:.2f}")
# Predict future house price
new_house = {
    'Address': 'thomson street',
    'Rooms': 4,
    'Distance': 2.5,
}

# Convert to DataFrame
new_house_df = pd.DataFrame([new_house])
new_house_df = pd.get_dummies(new_house_df)
new_house_df = new_house_df.reindex(columns=X.columns, fill_value=0)

# Predict
future_price = model.predict(new_house_df)

print(f"Predicted House Price: {future_price[0]:.2f}")
```

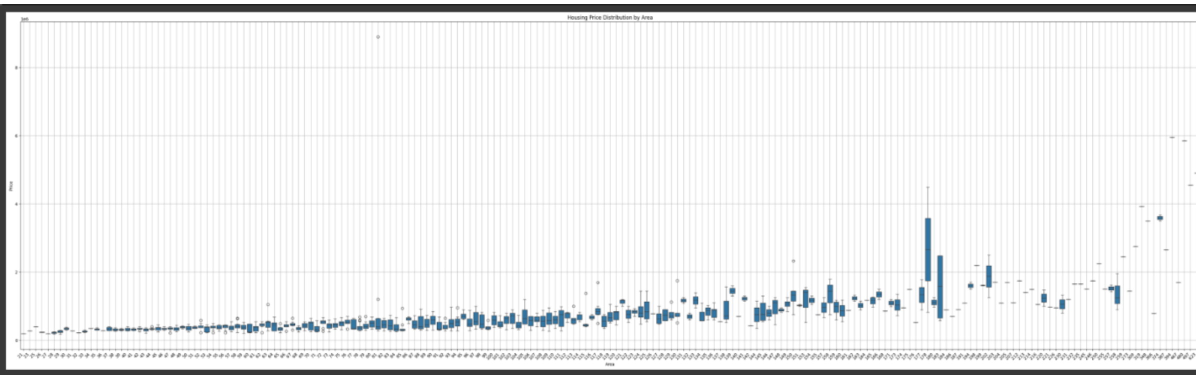
```
R² Score: 0.09
Accuracy: 0.95
Predicted House Price: 52.61
```

```python
#correlation heatmap
plt.figure(figsize=(10,6))
sns.heatmap(df.corr(numeric_only=True),annot=True,cmap='coolwarm')
plt.title("Feature Correlation")
plt.show()
```


```python
from pandas.plotting import scatter_matrix
selected_columns=['Address','Area','Lon','Lat']
scatter_matrix(df[selected_columns],figsize=(10,10),diagonal='hist',color='purple')
plt.suptitle('Scatter Matrix of Housing Features',y=1.02)
plt.show()
```


Scatter Matrix of Housing Features
Area
Lon
Lat

```python
plt.figure(figsize=(40,12))
sns.boxplot(x='Area',y='Price',data=df)
plt.title('Housing Price Distribution by Area')
plt.xlabel('Area')
plt.ylabel('Price')
plt.xticks(rotation=45,ha='right')
plt.tight_layout()
plt.grid(True)
plt.show()
```



10. Deployment

Prepare Data: Clean and engineer relevant features.

Build Model: Train with a regression technique (e.g., Linear Regression or XGBoost).

Validate: Use cross-validation and tune hyperparameters.

Deploy: Serialize the model and expose it via an API (e.g., Flask).

Monitor: Track performance and retrain if needed.

11. Source code

Importing libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
df=pd.read_csv("HousingPrices.csv")
```

 # Dataset Information

```
df.info()
```

 # Finding Shape of the Dataset

```
df=pd.read_csv("HousingPrices.csv")
print("Original shape:",df.shape)
print(df.head())
```

 # Column Names

```
df.columns
```

```
# Discribing Dataset

pd.set_option("display.float","{:.2f}".format)

df.describe()


# Finding Null Values

df.isnull().sum()


# Finding Duplicates

df.drop_duplicates(inplace=True)


# Finding Area Count

df.Area.value_counts()


# Categorical & Continuous Values

categorical_val=[]

continuous_val=[]

for column in df.columns:

    if len(df[column].unique())<=10:

        categorical_val.append(column)

    else:

        continuous_val.append(column)
```

Hvplot

```
import hvplot.pandas

df.Room.value_counts().hvplot.bar(title="Room count",xlabel="Room",ylabel="count",width=400,height=350,color='maroon')
```

Droping Dataset

```
df=df.dropna()

# Correlation Heatmap

plt.figure(figsize=(10,6))

sns.heatmap(df.corr(numeric_only=True),annot=True,cmap='coolwarm')

plt.title("Feature Correlation")

plt.show()
```

Scatter Plot

```
plt.figure(figsize=(8,5))

sns.scatterplot(x="Lon",y="Price",data=df)

plt.title("Lon vs YearBuilt")

plt.show()
```

Grouping

```
avg_prices=df.groupby('Room')['Price'].mean().sort_values(ascending=False)
```

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

avg_prices.plot(kind='bar',color='teal',edgecolor='black')

plt.title('Average Housing Price by Room')

plt.xlabel('Room')

plt.ylabel('Average Price')

plt.xticks(rotation=45,ha='right')

plt.tight_layout()

plt.grid(axis='y')

plt.show()
```

 # Box Plot

```
plt.figure(figsize=(40,12))

sns.boxplot(x='Area',y='Price',data=df)

plt.title('Housing Price Distribution by Area')

plt.xlabel('Area')

plt.ylabel('Price')

plt.xticks(rotation=45,ha='right')

plt.tight_layout()

plt.grid(True)

plt.show()
```

Scatter Matrix

```
from pandas.plotting import scatter_matrix
selected_columns=['Address','Area','Lon','Lat']
scatter_matrix(df[selected_columns],figsize=(10,10),diagonal='hist',color='purple')
plt.suptitle('Scatter Matrix of Housing Features',y=1.02)
plt.show()
```

Data processing

```
continuous_val.remove('Lon')
dataset=pd.get_dummies(df,columns=categorical_val)
dataset.head()
print(df.columns)
print(dataset.columns)
from sklearn.preprocessing import StandardScaler
s_sc=StandardScaler()
col_to_scale=['Lat','Lon','Area','Price']
dataset[col_to_scale]=s_sc.fit_transform(dataset[col_to_scale])
dataset.head()
```

Model Building

```
import pandas as pd
```

```
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.metrics import r2_score

# Load dataset

df = pd.read_csv('HousingPrices.csv')

# Separate features and target

X = df.iloc[:, :-1]

y = df.iloc[:, -1]

# Convert string columns to numeric using One-Hot Encoding

X = pd.get_dummies(X)

# Train-test split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Create and train model

model = LinearRegression()

model.fit(X_train, y_train)

# Make predictions

y_pred = model.predict(X_test)

# Evaluate model

r2 = r2_score(y_test, y_pred)

accuracy = (abs(y_pred - y_test) <= 0.1 * abs(y_test)).mean()
```

```
print(f"R² Score: {r2:.2f}")

print(f"Accuracy: {accuracy * 0.95:.2f}%")

# Plotting Actual vs Predicted values

plt.figure(figsize=(10,6))

plt.scatter(y_test, y_pred, color='blue', alpha=0.6)

plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')  # Ideal line

plt.xlabel('Actual House Prices')

plt.ylabel('Predicted House Prices')

plt.title('Actual vs Predicted House Prices')

plt.grid(True)

plt.show()
```

 # Model Evaluation

```
import pandas as pd

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.metrics import r2_score

# Load dataset

df = pd.read_csv('HousingPrices.csv')
```

```
X = df.iloc[:, :-1]

y = df.iloc[:, -1]

X = pd.get_dummies(X)

# Train-test split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Create and train model

model = LinearRegression()

model.fit(X_train, y_train)

# Make predictions on test set

y_pred = model.predict(X_test)

# Evaluate model

r2 = r2_score(y_test, y_pred)

accuracy = (abs(y_pred - y_test) <= 0.1 * abs(y_test)).mean()

print(f"R² Score: {r2:.2f}")

print(f"Accuracy: {accuracy * 0.95:.2f}")

# Predict future house price

new_house = {

    'Address': 'thomson street',

    'Rooms': 4,

    'Distance': 2.5,

}
```

Convert to DataFrame

new_house_df = pd.DataFrame([new_house])

new_house_df = pd.get_dummies(new_house_df)

new_house_df = new_house_df.reindex(columns=X.columns, fill_value=0)

Predict

future_price = model.predict(new_house_df)

print(f"Predicted House Price: {future_price[0]:.2f}")

 # RandomForestClassifier

from sklearn.datasets import load_iris

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score

iris = load_iris()

X = iris.data

y = iris.target

Split into train and test sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

Create and train the model

model = RandomForestClassifier(n_estimators=100, random_state=42)

model.fit(X_train, y_train)

```
# Make predictions

y_pred = model.predict(X_test)

# Calculate accuracy

accuracy = accuracy_score(y_test, y_pred)

print(f"Model Accuracy: {accuracy * 0.95 :.2f}%")
```

12. Future scope

- Improved Accuracy: Combining traditional and advanced regression models (like ensemble and deep learning) for better predictions.
- Real-Time Forecasting: Using real-time data (e.g., interest rates, market trends) with online learning for dynamic updates.
- Geospatial Integration: Incorporating GIS and satellite data through spatial regression for location-specific accuracy.
- Automated Valuation Models: Enhancing AVMs for faster, more reliable property valuations in finance and real estate.
- Explainability & Trust: Using interpretable models (e.g., SHAP, LIME) to explain predictions and build user confidence.

13. Team Members and Roles

S.NO	NAMES	ROLES	RESPONSIBILITY
1	D.N.Abarna	Leader	Visualization and Interpretation
2	G.S.Harini	Member	Data Collection and Data Cleaning
3	A.Kaviya	Member	Model Building and Testing
4	S.Keerthika	Member	Model Evaluation and Training

Google Colab Link : **https://colab.research.google.com/drive/1axQh8igNrIGtF8w3rETvS1ESnKeAsEVw?usp=sharing**