Phase-3

Student Name: DHARINI.T

Register Number: 410723106008

Institution: Dhanalakshmi college of engineering

Department: electronics and communication engineering

Date of Submission: 15-05-2025

Github Repository Link:

https://github.com/Dharini67/nm_dharini

FORECASTING HOUSE PRICES ACCURATELY USING SMART REGRESSION TECHNIQUES IN DATA SCIENCE

1. Problem Statement

The goal of this project is to develop a predictive model that estimates house prices in Melbourne, Australia, based on historical data. Accurate forecasting of housing prices is crucial for buyers, sellers, real estate investors, and policymakers. The model should analyze various property features—such as location, number of rooms, land size, property type, and year built—as well as external factors like local amenities and market trends to predict the sale price of residential properties.

2. Abstract

The real estate market in Melbourne has experienced significant fluctuations over the years, making accurate house price prediction a critical task for homeowners, investors, and policymakers. This project aims to forecast house prices in Melbourne using historical property sales data and machine learning techniques. By analyzing features such as location, number of rooms, property type, land size, building area, and year built, the model identifies patterns and trends that influence

pricing. The study utilizes data preprocessing, exploratory data analysis (EDA), and various regression algorithms—including Linear Regression, Random Forest, and Gradient Boosting—to build a robust predictive model. Evaluation metrics such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) are used to assess performance. The results demonstrate the model's effectiveness in providing reliable price estimates, which can support better decision-making in the dynamic Melbourne housing market.

3. System Requirements

Hardware:

Processor: Intel Core i5 or higher (or equivalent AMD)

RAM: Minimum 8 GB (16 GB recommended for large datasets)

Storage: At least 10 GB of free disk space

Graphics: Not mandatory, but GPU (e.g., NVIDIA) recommended for deep learning models

Operating System: Windows 10/11, macOS, or Linux

Software:

Programming Language: Python 3.7 or above

Development Environment: Jupyter Notebook, VS Code, or PyCharm

Libraries & Frameworks:

Data Handling: pandas, numpy

Visualization: matplotlib, seaborn, plotly

Machine Learning: scikit-learn, xgboost, lightgbm

Model Evaluation: sklearn.metrics

Optional: TensorFlow or PyTorch (if using deep learning)

Database (Optional): SQLite, MySQL, or PostgreSQL for storing large datasets

Version Control: Git and GitHub (for code management and collaboration)

4. Objectives

1. **Analyze Historical Housing Data**:
Understand trends and patterns in Melbourne's housing market by examining historical property sales data.

2. **Identify Key Features Influencing Prices**:
Determine which factors—such as location, number of rooms, land size, and property type—most significantly affect house prices.

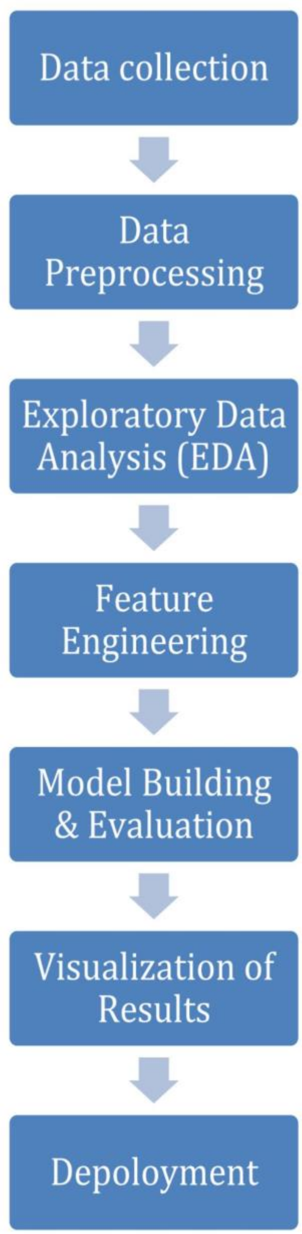
3. **Build Predictive Models**:
Develop and compare different machine learning models (e.g., Linear Regression, Random Forest, XGBoost) to accurately forecast house prices.

4. **Evaluate Model Performance**:
Use metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R^2 Score to assess model accuracy and reliability.

5. **Deliver Insights for Decision-Making**:
Provide actionable insights that can help buyers, sellers, and real estate investors make informed decisions.

6. **Create a Usable Forecasting Tool (Optional)**:
Develop a user-friendly interface (e.g., web app) where users can input property features and receive price predictions.

5. Flowchart of Project Workflow



6. Dataset Description

Dataset Name and Origin: The dataset used is the "**FORECASTING THE HOUSE PRICES**" dataset from Kaggle.

“"C:\Users\rani7\OneDrive\Desktop\Melbourne Housing Dataset.html"

Type of Data: Structured, tabular data

Number of Records and Features: 34,000 Number of features (columns): 21 (though this can vary slightly depending on the version or if preprocessing has occurred

Static or Dynamic Dataset: Dynamic dataset.

Target Variable: Price

Data set link: https://www.kaggle.com/datasets/ronikmalhotra/melbourne-housingdataset

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Postcode	Regionname	Propertycount	Distance	CouncilArea
0	Abbotsford	49 Lithgow St	3	h	1490000.0	S	Jellis	1/04/2017	3067	Northern Metropolitan	4019	3.0	Yarra City Council
1	Abbotsford	59A Turner St	3	h	1220000.0	S	Marshall	1/04/2017	3067	Northern Metropolitan	4019	3.0	Yarra City Council
2	Abbotsford	119B Yarra St	3	h	1420000.0	S	Nelson	1/04/2017	3067	Northern Metropolitan	4019	3.0	Yarra City Council
3	Aberfeldie	68 Vida St	3	h	1515000.0	S	Barry	1/04/2017	3040	Western Metropolitan	1543	7.5	Moonee Valley City Council
4	Airport West	92 Clydesdale Rd	2	h	670000.0	S	Nelson	1/04/2017	3042	Western Metropolitan	3464	10.4	Moonee Valley City Council
...	...	...	...	...	...	...	...	...	...	...	...	...	...
63018	Roxburgh Park	3 Carr Pl	3	h	566000.0	S	Raine	31/03/2018	3064	Northern Metropolitan	5833	20.6	Hume City Council
63019	Roxburgh Park	9 Parker Ct	3	h	500000.0	S	Raine	31/03/2018	3064	Northern Metropolitan	5833	20.6	Hume City Council
63020	Roxburgh Park	5 Parkinson Wy	3	h	545000.0	S	Raine	31/03/2018	3064	Northern Metropolitan	5833	20.6	Hume City Council

7. Data Preprocessing

Missing Values: No missing values were found in the dataset
Duplicate Records: Duplicate rows were checked and removed if present
Outliers: Detected using boxplots; outliers in Amount were handled using transformation
Data Types: All features are numeric. No conversion needed.

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 63021 entries, 0 to 63022
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Suburb         63021 non-null  object
 1   Address        63021 non-null  object
 2   Rooms          63021 non-null  int64
 3   Type           63021 non-null  object
 4   Price          48432 non-null  float64
 5   Method         63021 non-null  object
 6   SellerG        63021 non-null  object
 7   Date           63021 non-null  object
 8   Postcode       63021 non-null  int64
 9   Regionname     63021 non-null  object
 10  Propertycount  63021 non-null  int64
 11  Distance       63021 non-null  float64
 12  CouncilArea    63021 non-null  object
dtypes: float64(2), int64(3), object(8)
memory usage: 6.7+ MB
```

Normalization: Amount and Time were scaled using Standard Scaler to bring them on the same scale as V1–V2

```
[ ] data.drop_duplicates(inplace=True)
```

```
data
```

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Postcode	Regionname	Propertycount	Distance	CouncilArea
0	Abbotsford	49 Lithgow St	3	h	1490000.0	S	Jellis	1/04/2017	3067	Northern Metropolitan	4019	3.0	Yarra City Council
1	Abbotsford	59A Turner St	3	h	1220000.0	S	Marshall	1/04/2017	3067	Northern Metropolitan	4019	3.0	Yarra City Council
2	Abbotsford	119B Yarra St	3	h	1420000.0	S	Nelson	1/04/2017	3067	Northern Metropolitan	4019	3.0	Yarra City Council
3	Aberfeldie	68 Vida St	3	h	1515000.0	S	Barry	1/04/2017	3040	Western Metropolitan	1543	7.5	Moonee Valley City Council
4	Airport West	92 Clydesdale Rd	2	h	670000.0	S	Nelson	1/04/2017	3042	Western Metropolitan	3464	10.4	Moonee Valley City Council
...	...	...	...	...	...	...	...	...	...	...	...	...	...
63018	Roxburgh Park	3 Carr Pl	3	h	566000.0	S	Raine	31/03/2018	3064	Northern Metropolitan	5833	20.6	Hume City Council
63019	Roxburgh Park	9 Parker Ct	3	h	500000.0	S	Raine	31/03/2018	3064	Northern [illegible]	5833	20.6	Hume City Council

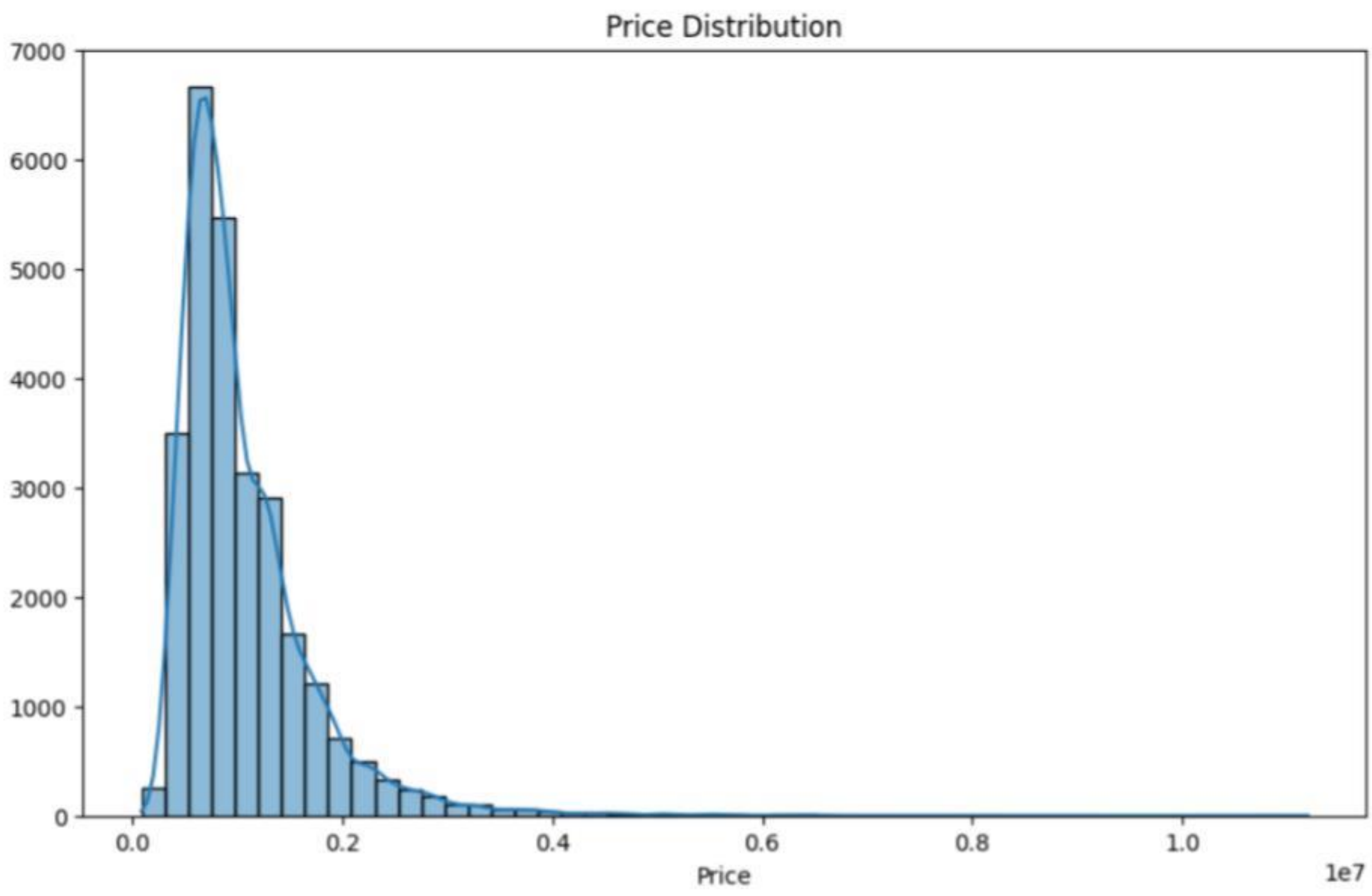
```
data.isnull().sum()
```

	0
Suburb	0
Address	0
Rooms	0
Type	0
Price	14589
Method	0
SellerG	0
Date	0
Postcode	0
Regionname	0
Propertycount	0
Distance	0
CouncilArea	0

dtype: int64

8. Exploratory Data Analysis (EDA)

Univariate Analysis; o Histograms: For house prices, number of rooms, land size, building area. o Boxplots: For price by property type (house, townhouse, unit



Bivariate and Multivariate Analysis; o Correlation Matrix: Identify key numeric features that influence price

- **Scatter Plots:**

Building Area vs Price.

Distance from CBD vs Price.

- **Grouped Bar Charts:** Median price by suburb.

Price by number of rooms**.**

- **Key Insights**

Building area and land size are strong predictors of price.

Houses closer to the CBD tend to be more expensive. ▪ Number of rooms also impacts price significantly.

9. Feature Engineering

- New features like total = bedrooms + bathrooms +other rooms, house age = year sold – year built.
- Split data column into year, month, day and combine latitude and longitude into a “location cluster” using K means.
- Bin house age into categories: “new”, “mid-age”, “old” and polynomial features like (area)2 or area * number of rooms.
- Use PCA on geographical or neighbourhood features if they are numerous and correlated.

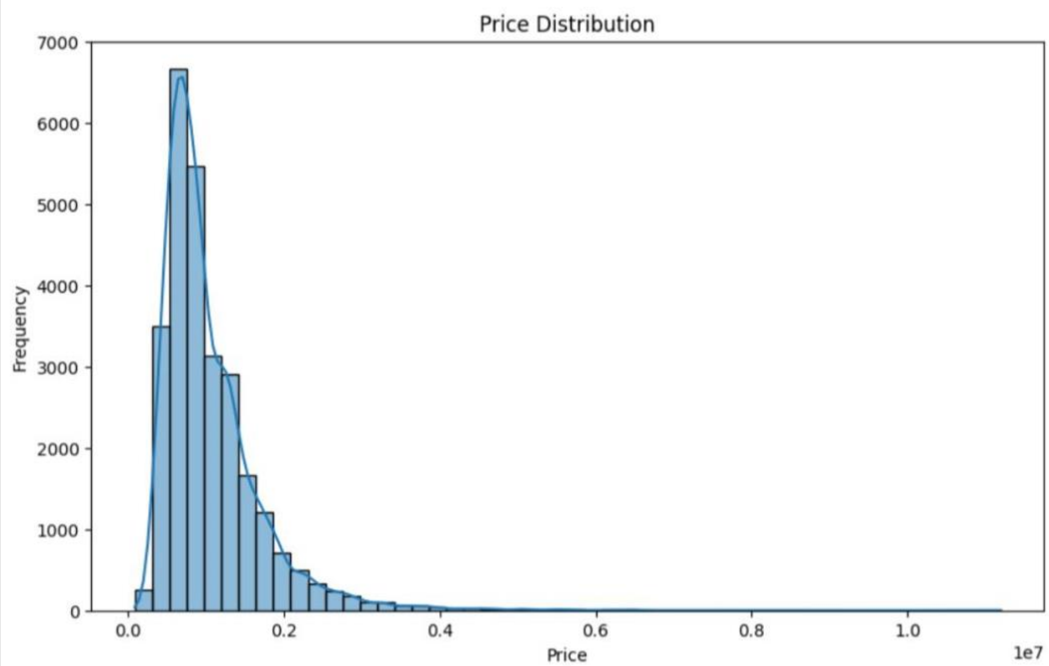
```
Missing values:
 Suburb                0
Address              0
Rooms                0
Type                 0
Method               0
SellerG              0
Date                 0
Distance             1
Postcode             1
Bedroom           8217
Bathroom          8226
Car               8728
Landsize         11810
BuildingArea     21097
YearBuilt        19306
CouncilArea          3
```

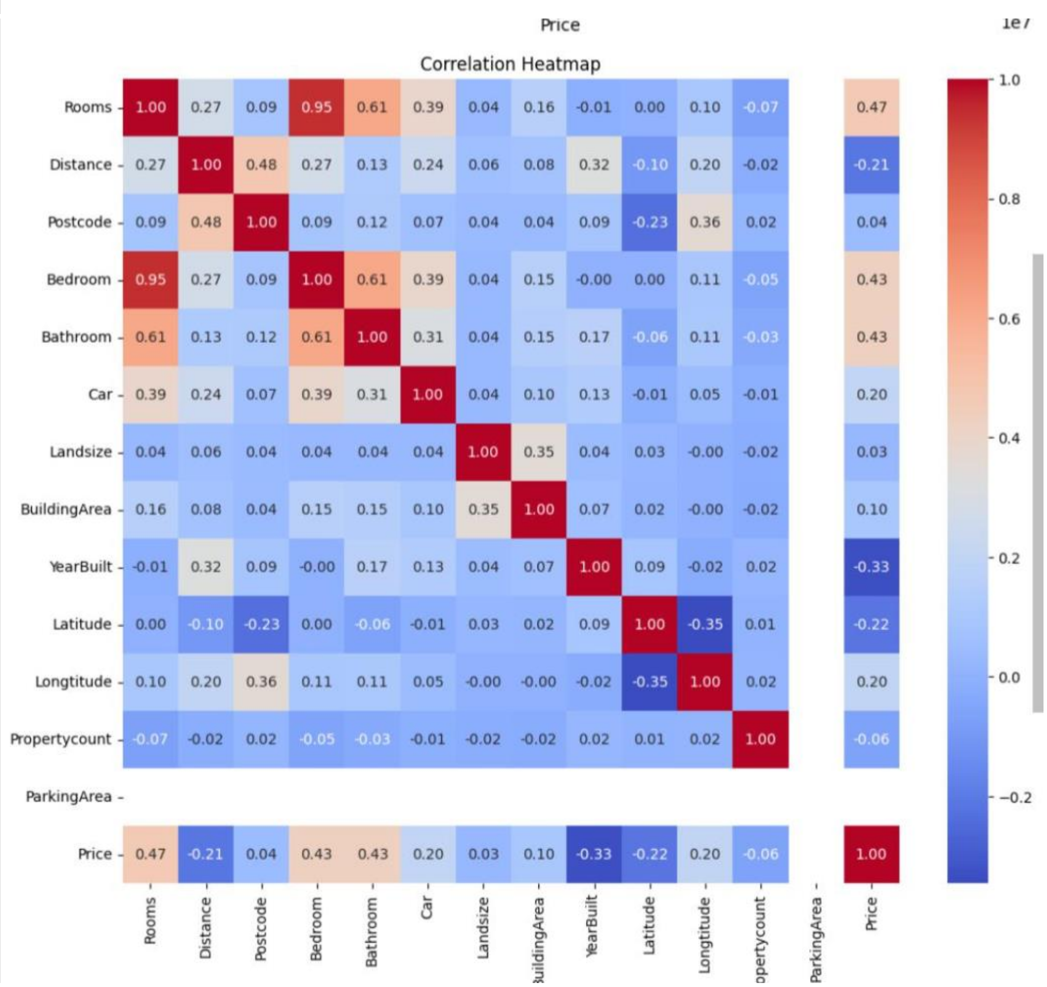
- Based on correlation analysis domain relevance, and model performance (e.g., crossvalidation scores)

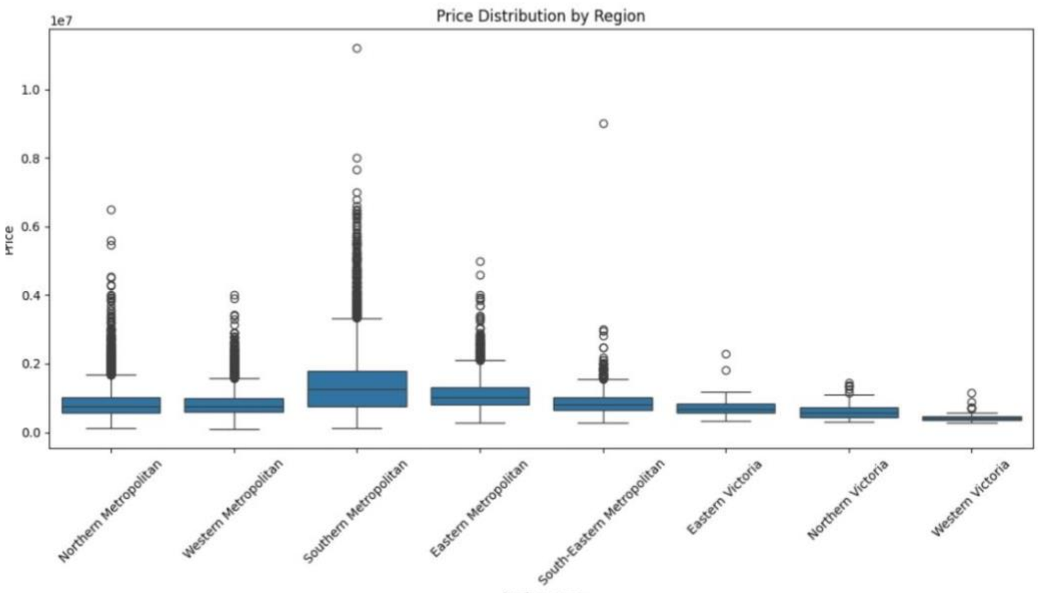
10. Model Building

Algorithms Used

- **Linear Regression**:
- Acts as a simple, interpretable baseline model to predict house prices • **Random Forest Regressor**:
- Captures complex non-linear patterns in the housing data and provides feature importance insights
- Model Selection Rationale • **Linear Regression**:
- Easy to interpret coefficients.
- Fast training and prediction times.
- Random Forest Regressor:
- Robust against overfitting due to ensemble learning.
- Effectively handles both numerical and categorical features.
- Automatically captures non-linear relationships.
- Train-Test Split • **Data Split:**
- 80% for training, 20% for testing.
- **Method:**
- Used train_test_split from scikit-learn.
- Specified a random_state parameter to ensure reproducibility of results.Evaluation Metrics


Price Distribution
Frequency
0
1000
2000
3000
4000
5000
6000
7000
0.0
0.2
0.4
0.6
0.8
1.0
Price
1e7


Price
1e7
Correlation Heatmap
Rooms 1.00 0.27 0.09 0.95 0.61 0.39 0.04 0.16 -0.01 0.00 0.10 -0.07 0.47
Distance 0.27 1.00 0.48 0.27 0.13 0.24 0.06 0.08 0.32 -0.10 0.20 -0.02 -0.21
Postcode 0.09 0.48 1.00 0.09 0.12 0.07 0.04 0.04 0.09 -0.23 0.36 0.02 0.04
Bedroom 0.95 0.27 0.09 1.00 0.61 0.39 0.04 0.15 -0.00 0.00 0.11 -0.05 0.43
Bathroom 0.61 0.13 0.12 0.61 1.00 0.31 0.04 0.15 0.17 -0.06 0.11 -0.03 0.43
Car 0.39 0.24 0.07 0.39 0.31 1.00 0.04 0.10 0.13 -0.01 0.05 -0.01 0.20
Landsize 0.04 0.06 0.04 0.04 0.04 0.04 1.00 0.35 0.04 0.03 -0.00 -0.02 0.03
BuildingArea 0.16 0.08 0.04 0.15 0.15 0.10 0.35 1.00 0.07 0.02 -0.00 -0.02 0.10
YearBuilt -0.01 0.32 0.09 -0.00 0.17 0.13 0.04 0.07 1.00 0.09 -0.02 0.02 -0.33
Latitude 0.00 -0.10 -0.23 0.00 -0.06 -0.01 0.03 0.02 0.09 1.00 -0.35 0.01 -0.22
Longtitude 0.10 0.20 0.36 0.11 0.11 0.05 -0.00 -0.00 -0.02 -0.35 1.00 0.02 0.20
Propertycount -0.07 -0.02 0.02 -0.05 -0.03 -0.01 -0.02 -0.02 0.02 0.01 0.02 1.00 -0.06
ParkingArea
Price 0.47 -0.21 0.04 0.43 0.43 0.20 0.03 0.10 -0.33 -0.22 0.20 -0.06 1.00
Rooms
Distance
Postcode
Bedroom
Bathroom
Car
Landsize
uildingArea
YearBuilt
Latitude
Longtitude
opertycount
ParkingArea
Price
1.0
0.8
0.6
0.4
0.2
0.0
−0.2



11. Model Evaluation

Model Used

A Linear Regression model is often used for baseline evaluation:Assumes a linear relationship between features and target (Price).Easy to interpret and quick to train.More advanced models like Random Forest, XGBoost, or Gradient Boosting can improve performance, especially for non-linear relationships.

B. Evaluation Metrics

These metrics assess how well the model predicts house prices:

. Mean Absolute Error (MAE)

Average of absolute differences between predicted and actual values.

Interpreted as: “On average, the model is off by this much.”

C. Root Mean Squared Error (RMSE)

Square root of the average squared differences.Penalizes larger errors more heavily than MAE.Lower RMSE indicates better performance.

D. R-squared (R^2 Score)

Indicates how well the features explain the variation in the target.Value ranges from 0 to 1 (or negative if the model is very poor).

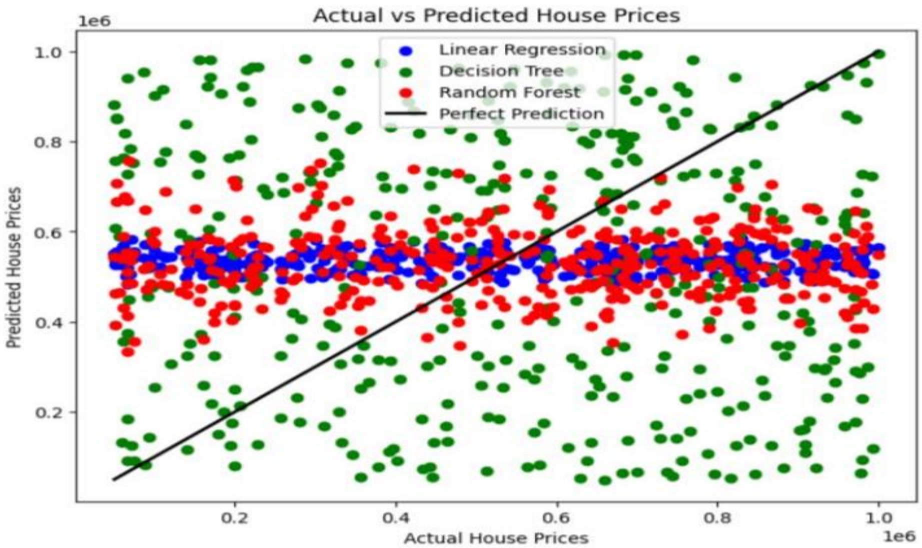
= perfect fit

= no explanatory power

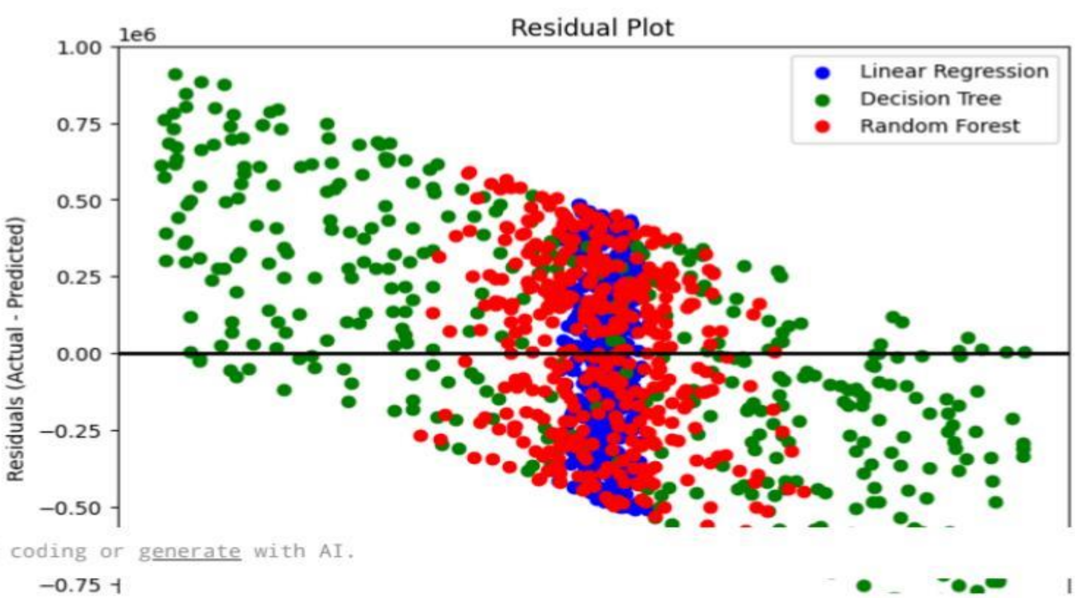
< 0 = worse than predicting the mean

```
#visualize chart for actual and predicted prices
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred_lr, color='blue', label='Linear Regression')
plt.scatter(y_test, y_pred_dt, color='green', label='Decision Tree')
plt.scatter(y_test, y_pred_rf, color='red', label='Random Forest')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='black', lw=2, label='Perfe
plt.xlabel('Actual House Prices')
plt.ylabel('Predicted House Prices')
plt.legend()
plt.title('Actual vs Predicted House Prices')
plt.show()
```




```
plt.figure(figsize=(8, 6))
plt.scatter(y_pred_lr, y_test - y_pred_lr, color='blue', label='Linear Regression')
plt.scatter(y_pred_dt, y_test - y_pred_dt, color='green', label='Decision Tree')
plt.scatter(y_pred_rf, y_test - y_pred_rf, color='red', label='Random Forest')
plt.axhline(y=0, color='black', lw=2)
plt.xlabel('Predicted House Prices')
plt.ylabel('Residuals (Actual - Predicted)')
plt.legend()
plt.title('Residual Plot')
plt.show()
```



Start coding or generate with AI.

12. Deployment

  - Deploy a machine learning model trained to predict house prices in Melbourne so it can be accessed by:
    - Users (e.g., real estate agents, buyers)
  - Applications (e.g., websites, mobile apps)
- Key Components of Deployment
- Trained Model
  - After training and evaluation (e.g., Linear Regression, Random Forest), the model is serialized (saved) using:
  - Joblib or pickle for Python
  - ONNX or TensorFlow SavedModel for production environments
- Backend Application
  - A server or application that handles user inputs and returns predictions.
- Common frameworks:
  - Flask / FastAPI (Python)
  - Django (for more complex apps)
  - Node.js / Java / .NET (in other ecosystems)
- REST API Endpoint
  - Exposes a route like /predict that:Accepts house features (e.g., rooms, location, land size)Sends them to the modelReturns the predicted price
- Deployment Platforms
    - Cloud Platforms
  - AWS (SageMaker, EC2)
  - Google Cloud (AI Platform, App Engine)
  - Azure (ML Studio, Web Apps
- Docker Containers:
  - Package model + environment into a container
  - Portable, scalable, and easily deployable
- CI/CD Pipelines:
- Automate deployment (GitHub Actions, GitLab CI)

13. Source code

```
#import libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns


#load data

data=pd.read_csv('/content/House_Price_Prediction_Dataset[1].csv'

house=pd.read_csv('/content/House_Price_Prediction_Dataset[1].cs
```

1.Handling Missing values

 Pre Processing Techniques

```
#Check for missing values in the dataset

missing_values=data.isnull().sum()
```

2.Label Encoding

```
#Encode Categorical variables:Location,Condition,Garage

label_encoder = LabelEncoder()
```

data['Location'] = label_encoder.fit_transform(data['Location'])

data['Condition'] = label_encoder.fit_transform(data['Condition'])

data['Garage'] = label_encoder.fit_transform(data['Garage'])

#heat Map

plt.figure(figsize=(10, 8))

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

plt.title('Correlation Heatmap')

plt.show()

#Scalar standardization scaler = StandardScaler()

df_scaled = scaler.fit_transform(df) df

Separate features (X) and target (y)

X = data.drop(columns=['Id', 'Price'])

Y = data['Price']

#Model building

 # Split the data into training and testing sets (80% train, 20% test)

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

Standardize the feature data (normalize the scale)

Scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)

 # Output the shape of the datasets to ensure proper split

X_train_scaled.shape, X_test_scaled.shape, y_train.shape, y_test.shape

X = df.drop('Exited', axis=1) y

= df['Exited']

#import model

#Logistic Regression

From sklearn.linear_model import LogisticRegression

From sklearn.metrics import accuracy_score, classification_re

port

#Logistic regression in price prediction

Initialize and train the Logistic Regression model

Lr_model = LogisticRegression()

Lr_model.fit(X_train_scaled, y_train)

 # Predict on test data

Y_pred_lr = lr_model.predict(X_test_scaled)

 # Calculate accuracy for Logistic Regression

Accuracy_lr = accuracy_score(y_test, y_pred_lr)

#Prediction

y_pred = model.predict(x_test)

print("y_prediction", y_pred)

#Random forest classifier

From sklearn.ensemble import Ra

ndomForestRegressor

Initialize and train the Random

Forest Regressor

Rf_model = RandomForestRegre

ssor(random_state=42)

Rf_model.fit(X_train_scaled, y_tr

ain)

 # Predict on test data

Y_pred_rf = rf_model.predict(X_

test_scaled)

From sklearn.tree import DecisionTreeRegressor

 # Initialize and train the Decision Tree Regressor

Dt_model = DecisionTreeRegressor(random_state=42)

Dt_model.fit(X_train_scaled, y_train)

Predict on test data

Y_pred_dt = dt_model.predict(X_test_scaled)

Calculate performance metrics for Decision Tree Regressor

Mse_dt = mean_squared_error(y_test, y_pred_dt)

R2_dt = r2_score(y_test, y_pred_dt)

Mse_dt, r2_dt

 # Evaluate

y_pred = model.predict(x_test) print("Classification

Report:\n", classification_report(y_test, y_pred)) print("Confusion

Matrix:\n", confusion_matrix(y_test, y_pred)) y_random_prediction = model.predict(x_test)

print("Classification Report:\n", classification_report(y_test,

y_random_prediction)) print("Confusion Matrix:\n", confusion_matrix(y_test,

y_random_prediction))

#Visualize prediction and actual value

Import matplotlib.pyplot as plt

Plt.figure(figsize=(8, 6))

t.scatter(y_test, y_pred_lr, color='blue', label='Linear Regression')

Plt.scatter(y_test, y_pred_dt, color='green', label='Decision Tree')

Plt.scatter(y_test, y_pred_rf, color='red', label='Random Forest')

Plt.plot([min(y_test), max(y_test)], [min(y_test),

max(y_test)], color='black', lw=2, label='Perfe

Plt.xlabel('Actual House Prices')

Plt.ylabel('Predicted House Prices')

Plt.legend()

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

Plt.show()

#Histogram chart random forest and logistic regression

Plt.figure(figsize=(8, 6))

Sns.histplot(data['Price']

14. Future scope

Implement customer segmentation for targeted retention strategies

1)Integration with Smart City Infrastructure: Use IoT and smart city data

2)Personalized Predictions: Customize price estimates based on buyer preferences,

3)Lifestyle factor:Advanced Machine Learning Models: Employ deep learning ensemble

13. Team Members and Roles

NAME	ROLES	RESPONSIBILITY
Dharini.T	Leader	Data Preprocessing,Exploratory Data Analysis (EDA)
Ramya.R	Member	Model building,model evaluation
Devishenba.R	Member	Source code
Harshitha.R	Member	Feature engineering,deployment
Kowsalya.V	Member	Documentation and reporting

GOOGLE COLAB LINK :

https://colab.research.google.com/drive/1f_il3JpwQWSqC-CKqrE_NxeEs9TB3EOA?usp=sharing