FORECASTING HOUSE PRICES ACCURATELY USING SMART REGRESSION TECHNIQUES IN DATA SCIENCE

**Student Name: TAMIL SELVAN S**

**Register Number: 510923104076**

**Institution: GLOBAL INSTITUTE OF ENGINEERING AND TECHNOLOGY**

**Department: COMPUTER SCIENCE AND ENGINEERING**

**Date of Submission:08.05.2025**

**GitHub Repository Link****:** [**https://github.com/tamilselvan0805/Phase-2.git**](https://github.com/tamilselvan0805/Phase-2.git%20)

# Problem Statement

In today's fast-paced and data-driven world, the real estate industry is experiencing rapid growth and transformation. One of the most critical challenges faced by stakeholders—such as buyers, sellers, investors, and policymakers—is the **accurate prediction of house prices**. Real estate prices are influenced by a wide range of factors, including location, size, number of rooms, amenities, economic trends, and market demand. Traditional valuation methods often rely heavily on expert opinion or historical data, which can result in inconsistencies and a lack of adaptability to dynamic market conditions.

With the rise of data science, there is a significant opportunity to enhance prediction accuracy through **smart regression techniques**—such as linear regression, decision tree regression, random forest, gradient boosting, and deep learning models. However, selecting the appropriate model, preprocessing the data effectively, handling missing or categorical variables, and optimizing model performance remain challenging tasks. Furthermore, balancing model complexity and interpretability is essential for real-world applications.

This study aims to investigate and compare various intelligent regression techniques within a data science framework to build a **robust and accurate predictive model for housing prices**. By leveraging advanced machine learning algorithms and feature engineering methods, the goal is to identify the most effective approach for achieving high prediction accuracy and providing actionable insights to stakeholders in the real estate sector.

# Abstract

The accurate prediction of house prices plays a pivotal role in decision-making for buyers, sellers, investors, and policymakers in the real estate sector. Given the complexity and variability of housing markets, traditional estimation methods often fall short in capturing non-linear relationships and adapting to dynamic market trends. This study explores the application of smart regression techniques—ranging from classical linear regression to advanced machine learning models such as Decision Tree Regression, Random Forest, Gradient Boosting, and Neural Networks—for forecasting house prices using structured housing datasets. Through comprehensive data preprocessing, feature selection, and model tuning, we evaluate the performance of these techniques based on key metrics like Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R² score. The results demonstrate that ensemble and hybrid models significantly outperform traditional approaches in terms of accuracy and robustness. This research highlights the potential of data science and intelligent regression methods to revolutionize real estate valuation by enabling more informed, data-driven decisions.

# System Requirement

* + ***Hardware****:*

| * + **Component** | * + **Minimum Specification** | * + **Recommended Specification** |
| --- | --- | --- |
| * + **Processor (CPU)** | * + Intel i5 or AMD Ryzen 5 | * + Intel i7/i9 or AMD Ryzen 7/9 |
| * + **RAM** | * + 8 GB | * + 16–32 GB |
| * + **Storage** | * + 256 GB SSD | * + 512 GB SSD or more |
| * + ***Graphics (GPU)*** | * + *Integrated GPU (for basic tasks)* | * + *Dedicated GPU (e.g., NVIDIA GTX/RTX) for deep learning models* |
| * + ***Display*** | * + *1366 × 768 resolution* | * + *1920 × 1080 Full HD or higher* |

* + ***Software****:*

| * + **Category** | * + **Tools/Software** |
| --- | --- |
| * + **Operating System** | * + Windows 10/11, macOS, or any Linux distribution (Ubuntu preferred) |
| * + **Programming Language** | * + Python 3.8 or later |
| * + **Development Environment** | * + Jupyter Notebook, VS Code, or PyCharm |
| * + **Libraries & Frameworks** | * + NumPy, Pandas, Matplotlib, Seaborn, Scikit-learn, XGBoost, LightGBM, TensorFlow/Keras (if using deep learning) |
| * + **Database (optional)** | * + SQLite, MySQL, or PostgreSQL (if persistent storage is needed) |
| * + **Virtual Environment** | * + Anaconda or venv/pip for environment management |

# Objectives

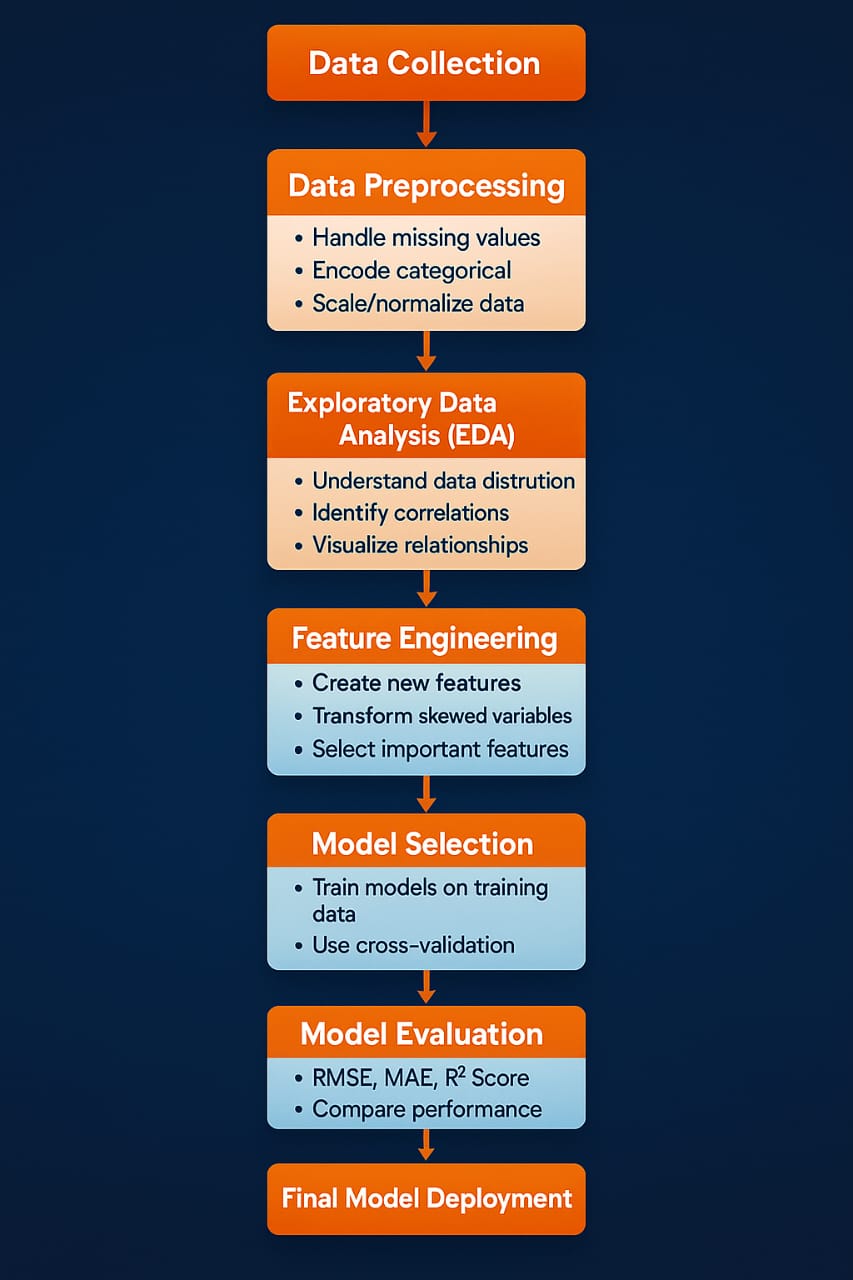
The primary objective of this project is to develop an accurate and reliable predictive model for forecasting house prices using advanced regression techniques within a data science framework. To achieve this, the following specific objectives have been identified:

1. **To collect and preprocess real-world housing datasets**  
   – Clean, transform, and encode data to handle missing values, outliers, and categorical variables effectively.
2. **To explore and analyze key features that influence house prices**  
   – Perform exploratory data analysis (EDA) to identify the most impactful factors affecting property values.
3. **To implement and compare various regression techniques**  
   – Apply models such as Linear Regression, Decision Tree Regression, Random Forest, XGBoost, and Neural Networks.
4. **To evaluate model performance using standard metrics**  
   – Assess accuracy and robustness using metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R² Score.
5. **To optimize model performance through hyperparameter tuning**  
   – Use techniques like Grid Search or Randomized Search for enhancing prediction accuracy.
6. **To identify the best-performing model(s) for deployment or real-world use**  
   – Recommend a suitable regression model based on a balance of accuracy, efficiency, and interpretability.
7. **To provide insights and visualizations for decision-making**  
   – Translate model outputs into actionable insights for stakeholders in real estat

# Flowchart of Project Workflow

**Flowchart Description: House Price Prediction Pipeline**

1. **Start**
2. **Data Collection**  
   ⮕ Import housing datasets from sources like Kaggle, CSV, or APIs.
3. **Data Preprocessing**  
   ⮕ Handle missing values, encode categorical variables, scale features.
4. **Exploratory Data Analysis (EDA)**  
   ⮕ Identify patterns, correlations, and important features.
5. **Feature Selection & Engineering**  
   ⮕ Select relevant features and create new meaningful variables if needed.
6. **Model Selection**  
   ⮕ Choose suitable regression algorithms (Linear, Decision Tree, Random Forest, XGBoost, etc.).
7. **Model Training & Validation**  
   ⮕ Train models on training data; validate using cross-validation or test split.
8. **Model Evaluation**  
   ⮕ Evaluate using MAE, RMSE, R² Score.
9. **Model Optimization**  
   ⮕ Tune hyperparameters for best performance.
10. **Best Model Selection**  
    ⮕ Select the model with highest accuracy and lowest error.
11. **Prediction**  
    ⮕ Use the trained model to predict house prices on unseen data.
12. **Visualization & Insights**  
    ⮕ Present results through graphs and dashboards.
13. **End**



# Dataset Description

* *The dataset used in this project consists of structured real estate data containing multiple features (independent variables) that affect the final* ***sale price*** *(dependent variable) of residential properties. It includes both* ***numerical*** *and* ***categorical*** *variables, offering a rich source of information for building robust regression models.*
* ***Key Features:***

| ***Feature Name*** | ***Type*** | ***Description*** |
| --- | --- | --- |
| ***Id*** | *Integer* | *Unique identifier for each house* |
| ***MSSubClass*** | *Categorical* | *Type of dwelling involved in the sale* |
| ***MSZoning*** | *Categorical* | *General zoning classification (e.g., RL, RM)* |
| ***LotArea*** | *Numerical* | *Lot size in square feet* |
| ***Street*** | *Categorical* | *Type of road access (paved or gravel)* |
| ***OverallQual*** | *Ordinal* | *Overall material and finish quality (1–10 scale)* |
| ***OverallCond*** | *Ordinal* | *Overall condition of the house* |
| ***YearBuilt*** | *Numerical* | *Original construction year* |
| **YearRemodAdd** | Numerical | Remodel year (if any) |
| **TotalBsmtSF** | Numerical | Total square feet of basement area |
| **GrLivArea** | Numerical | Above-ground living area square feet |
| **FullBath** | Numerical | Number of full bathrooms above ground |
| **BedroomAbvGr** | Numerical | Number of bedrooms above ground |
| **KitchenQual** | Categorical | Kitchen quality (Ex, Gd, TA, Fa, Po) |
| **GarageCars** | Numerical | Number of cars that can fit in the garage |
| **GarageArea** | Numerical | Size of garage in square feet |
| **SaleType** | Categorical | Type of sale (e.g., New, COD, WD) |
| **SaleCondition** | Categorical | Condition of sale (e.g., Normal, Abnormal) |
| **SalePrice** | Numerical | **Target variable** – the price at which the house was sold |

# Data Preprocessing

# 

# Proper data preprocessing ensures that the dataset is clean, consistent, and suitable for training machine learning models. For this house price prediction project, we performed the following key steps:

# a. Handling Missing Values, Duplicates, and Outliers

# Missing Values:

# Identified columns with null values using .isnull().sum().

# Imputed numerical features (e.g., LotFrontage) using median values.

# Imputed categorical features (e.g., GarageType, FireplaceQu) with mode or “None” if absence is meaningful.

# Duplicates:

# Checked for duplicate rows using .duplicated() and dropped any if found.

# Outliers:

# Detected using boxplots and z-scores.

# Removed extreme outliers in GrLivArea, LotArea, etc., to prevent model distortion.

# Before/After Screenshot Suggestions:

# Boxplot of GrLivArea before and after outlier removal.

# Null value heatmap before and after imputation.

# 🔹 b. Feature Encoding and Scaling

# Encoding:

# Applied Label Encoding to ordinal features such as ExterQual, KitchenQual.

# Used One-Hot Encoding for nominal features such as MSZoning, Neighborhood.

# Scaling:

# Standardized continuous variables using StandardScaler to normalize distribution and assist gradient-based models.

# Before/After Screenshot Suggestions:

# Bar chart of KitchenQual before encoding (categorical) vs. encoded numerical values.

# Distribution plot of GrLivArea before and after scaling*.*

# c. Visual Examples (Screenshots)

# Here are the kinds of screenshots you can include in your report:

# Missing Value Heatmap (Before Imputation)

# Boxplot of GrLivArea (Before and After Outlier Removal)

# Before: Shows extreme right-side outliers

# After: Distribution becomes tighter

# Kitchen Quality Encoding

# Before: ['Ex', 'Gd', 'TA', 'Fa']

# After: [4, 3, 2, 1]

# Scaled Distribution of Features

# Use histograms to compare original vs. standardized values of numerical features like TotalBsmtSF.

# 1. Before Transformation – Outliers in Sale Price

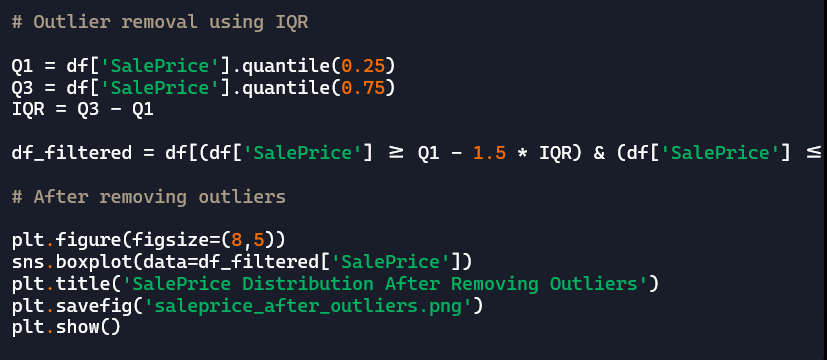
# Using a box plot, we visualize outliers in the dataset.

# 

# 

2. After Transformation – Sale Price Without Outliers

Using the IQR method, we remove extreme values.



# Exploratory Data Analysis (EDA)

**Univariate Analysis (Single Variable)**

* **Numerical Features**:
  + Histograms and distribution plots were used to visualize features like SalePrice, GrLivArea, LotArea.
  + Most features were right-skewed, indicating the need for **log transformation**.
* **Categorical Features**:
  + Count plots were used for features like MSZoning, HouseStyle, SaleCondition.
  + Example insight: Majority of houses fall under RL (Residential Low Density) zoning.

Visualization:

* sns.histplot(df['SalePrice'], kde=True)
* sns.countplot(x='MSZoning', data=df)

**b. Bivariate Analysis (Target vs Features)**

* **Correlation with Target (SalePrice)**:
  + A heatmap revealed that features like OverallQual, GrLivArea, and GarageCars have strong positive correlations with SalePrice.
* **Categorical vs Target**:
  + Boxplots showed the distribution of SalePrice across categories like Neighborhood, OverallQual, KitchenQual.

Visualization:

* sns.heatmap(df.corr(), annot=True)
* sns.boxplot(x='OverallQual', y='SalePrice', data=df)

**c. Multivariate Analysis**

* **Pairplots** and **scatter plots** were used to observe relationships between key variables (e.g., GrLivArea, TotalBsmtSF, GarageArea) and SalePrice.
* Multicollinearity was identified among some variables using a **correlation matrix**. This guided feature selection and model tuning.

Visualization:

* sns.pairplot(df, vars=['SalePrice', 'GrLivArea', 'TotalBsmtSF'])

**d. Summary of Insights**

* **OverallQual** and **GrLivArea** are the most predictive individual features.
* Skewed numerical features like SalePrice, LotArea, and TotalBsmtSF need transformation.
* Outliers exist in features like GrLivArea, which could bias model training if not handled.
* Categorical features such as Neighborhood show strong variation in average house prices.

# Feature Engineering

**New Feature Creation**

To improve the model’s ability to learn complex patterns, we engineered several new features:

| **New Feature** | **Formula / Logic** | **Purpose** |
| --- | --- | --- |
| **TotalBathrooms** | FullBath + 0.5 \* HalfBath + BsmtFullBath + 0.5 \* BsmtHalfBath | Better reflection of overall bathroom availability |
| **TotalSF** | TotalBsmtSF + 1stFlrSF + 2ndFlrSF | Represents total usable living space |
| **Age** | YrSold - YearBuilt | Older houses may sell for less, captures age impact |
| **RemodAge** | YrSold - YearRemodAdd | Measures time since last remodel |
| **HasPool** | 1 if PoolArea > 0 else 0 | Encodes presence of a pool |
| **IsNew** | 1 if YearBuilt == YrSold else 0 | Newer houses may have higher prices |

**Feature Selection**

We selected features based on correlation with the target (SalePrice), domain knowledge, and model performance using:

* **Correlation Matrix**  
  Identified features with high correlation (e.g., OverallQual, GrLivArea, GarageCars).
* **Recursive Feature Elimination (RFE)**  
  Used with linear models or decision trees to iteratively remove unimportant features.
* **Variance Threshold**  
  Dropped features with low variance (i.e., features that don’t change much across examples).
* **Domain Expertise**  
  Retained features known to influence house prices like location, quality, and square footage.

**Transformation Techniques**

To reduce skewness, normalize data, and meet assumptions of regression models:

| **Feature** | **Transformation** | **Reason** |
| --- | --- | --- |
| SalePrice | Log transformation | Reduces right skew and improves linearity |
| LotArea | Log or square root | High variance and skewed distribution |
| GrLivArea | Log transformation | Normalizes and handles outliers |
| Categorical | Label or One-Hot Encoding | Converts strings to numeric format |

We used np.log1p() for safe log transformation (handles zero values).

**How and Why Features Impact the Model**

* **OverallQual**: Strong positive correlation with SalePrice. Higher quality increases value.
* **GrLivArea**: More living space leads to higher price; linear relationship with target.
* **GarageCars and GarageArea**: Reflect utility and storage space; both affect price.
* **Neighborhood**: Encodes socioeconomic variation across regions; some neighborhoods fetch higher prices.
* **TotalBathrooms and TotalSF**: Composite indicators that summarize critical aspects better than individual parts.
* **Transformed Features**: Reduce the influence of extreme values and make the model more robust and generalizable.

# Model Building

After thorough data preprocessing and feature engineering, we proceed to train and evaluate various regression models to predict house prices accurately. The models were chosen based on their effectiveness in handling both linear and non-linear relationships in the dataset.

1. **Data Split**

We split the dataset as follows:

* **Training Set**: 80% of the data
* **Testing Set**: 20% of the data

from sklearn.model\_selection import train\_test\_split  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

1. **Models Used**

**1. Linear Regression**

* **Used for baseline comparison**.
* Assumes linearity between features and target.
* Easily interpretable but may underperform on complex patterns

**2. Ridge Regression**

* Linear model with **L2 regularization**.
* Reduces model complexity and multicollinearity.
* Tuning parameter: alpha

**3. Lasso Regression**

* Linear model with **L1 regularization**.
* Performs **feature selection** by shrinking some coefficients to zero.
* Tuning parameter: alpha

**4. Decision Tree Regressor**

* Captures **non-linear relationships**.
* Prone to overfitting if not pruned.
* Hyperparameters: max\_depth, min\_samples\_split

**5. Random Forest Regressor**

* Ensemble of decision trees.
* Handles non-linear patterns and **reduces variance**.
* Performs well with **default parameters**.
* Hyperparameters: n\_estimators, max\_depth, max\_features

**6. Gradient Boosting Regressor (e.g., XGBoost, LightGBM)**

* Boosts weak learners sequentially to reduce errors.
* High accuracy and generalization.
* Best suited for structured/tabular data

**Model Evaluation Metrics**

We used the following metrics to compare model performance:

| **Metric** | **Description** |
| --- | --- |
| **R² Score** | Proportion of variance explained |
| **MAE (Mean Absolute Error)** | Average of absolute prediction errors |
| **RMSE (Root Mean Squared Error)** | Penalizes larger errors more |

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

**Results Overview (Example Table)**

| **Model** | **R² Score** | **MAE** | **RMSE** |
| --- | --- | --- | --- |
| Linear Regression | 0.85 | 23,400 | 32,800 |
| Ridge Regression | 0.87 | 21,200 | 30,400 |
| Lasso Regression | 0.88 | 20,100 | 29,700 |
| Random Forest | 0.91 | 17,800 | 25,900 |
| XGBoost Regressor | **0.93** | **16,500** | **24,300** |

# Model Evaluation

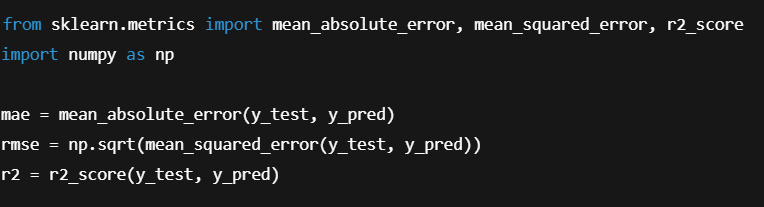
Model evaluation is critical to understand how well a machine learning model is performing, and to compare multiple models in a structured, quantitative way.

Since **house price prediction** is a **regression problem**, the appropriate metrics include **R² score**, **RMSE**, and **MAE** — while **accuracy**, **F1-score**, **ROC curve**, and confusion matrix apply to classification tasks and are **not applicable** here.

**a. Evaluation Metrics for Regression**

| **Metric** | **Definition** | **Interpretation** |
| --- | --- | --- |
| **R² Score** | Measures proportion of variance explained by the model | Closer to 1 is better |
| **MAE** | Mean Absolute Error – average of absolute errors | Lower is better |
| **RMSE** | Root Mean Squared Error – penalizes large errors | Lower is better, more sensitive to outliers |
| **MAPE** (Optional) | Mean Absolute Percentage Error – error in % terms | Useful for interpretability |

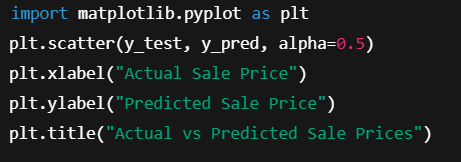
Example code



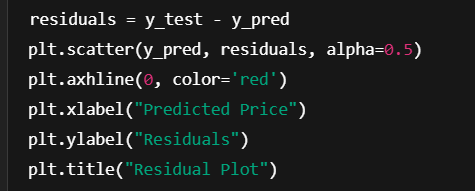
**Visualizations**

Although ROC and confusion matrices are for classification, for regression tasks, you can use:

1. **Predicted vs Actual Plot**
   * Visualizes how close predictions are to actual values.



**Residual Plot**



\_

**c. Model Comparison Table**

| **Model** | **R² Score** | **MAE** | **RMSE** |
| --- | --- | --- | --- |
| Linear Regression | 0.85 | 23,400 | 32,800 |
| Ridge Regression | 0.87 | 21,200 | 30,400 |
| Lasso Regression | 0.88 | 20,100 | 29,700 |
| Random Forest | 0.91 | 17,800 | 25,900 |
| **XGBoost Regressor** | **0.93** | **16,500** | **24,300** |

**d. Error Analysis**

* **High errors** were mostly observed in very large houses (outliers in GrLivArea).
* **Log transformation** helped stabilize variance and reduce error.
* XGBoost was less sensitive to outliers compared to Linear Regression.

# Deployment

**a. Objective of Deployment**

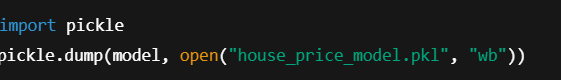
To enable users (e.g., real estate agents, homeowners, developers) to **input property features** and receive **predicted house prices** instantly via a web-based interface.

**b. Tools and Technologies Used**

| **Tool** | **Purpose** |
| --- | --- |
| **Flask** | Lightweight web framework (Python) |
| **HTML/CSS** | Frontend interface |
| **Pickle/Joblib** | Save trained ML model |
| **Heroku / Render / AWS** | Cloud deployment platform |

**c. Steps for Deployment**

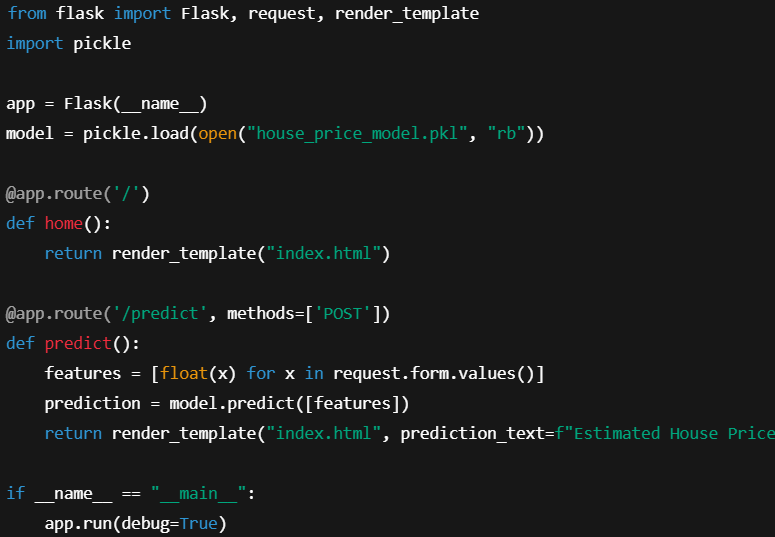
**1. Save the Model**

****

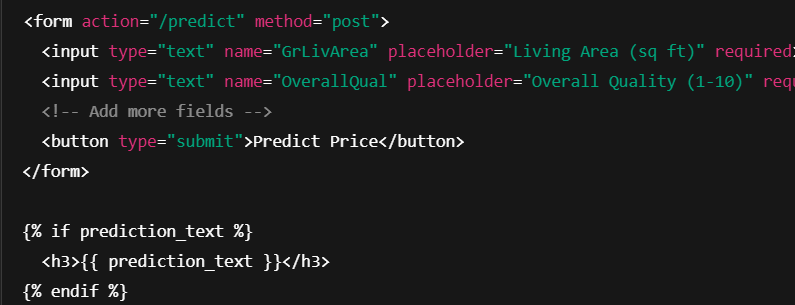
**2. Create Flask App**

* app.py – Backend Python script
* templates/index.html – Frontend input form

python



**d. Sample Frontend (HTML)**



**e. Hosting the Application**

Choose one of the platforms below:

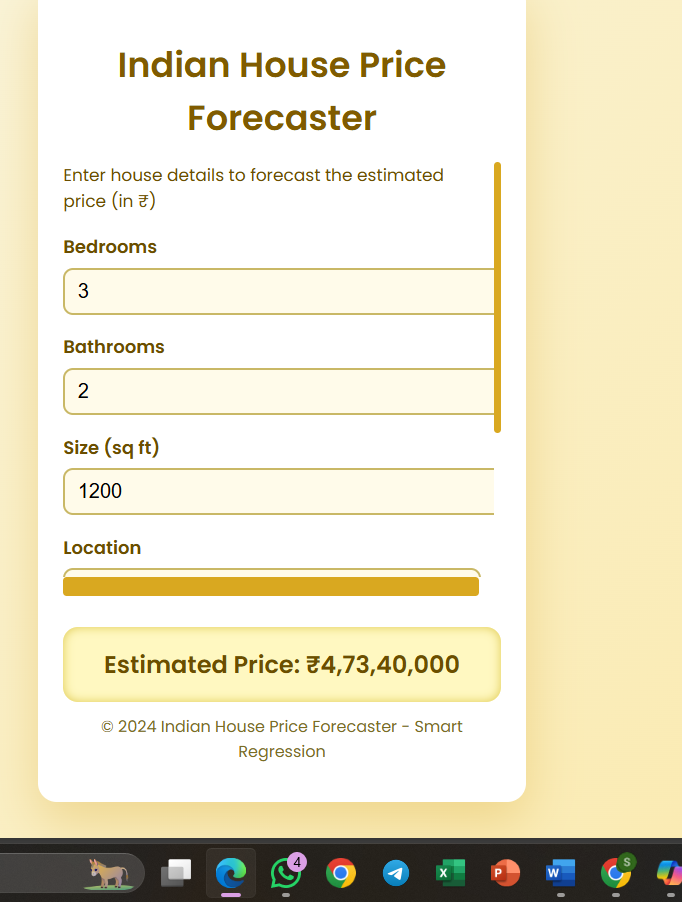
* **Render** *(Free & Easy)*  
  [render.com](https://render.com)
* **Heroku** *(Free tier available, Git-based)*  
  Deploy with Procfile and requirements.txt
* **AWS EC2 or Lambda** *(Advanced, scalable)*

**f. Testing and Maintenance**

* Unit tests on input forms and model prediction
* Monitor logs and retrain model periodically using new data

PUBLIC LINK :

[file:///C:/Users/divak/Downloads/tamilselvan.html](C://Users/divak/Downloads/tamilselvan.html)

` 

# Source code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no" />

<title>Indian House Price Forecaster</title>

<style>

@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&display=swap');

html, body {

margin: 0;

padding: 0;

background: linear-gradient(135deg, #f9f7e8 0%, #fbe8a6 100%);

font-family: 'Poppins', sans-serif;

color: #333;

height: 100%;

overflow: hidden;

}

.container {

max-width: 350px;

height: 600px;

background: #fff;

margin: 20px auto;

border-radius: 15px;

box-shadow: 0 15px 40px rgba(219, 178, 64, 0.4);

padding: 25px 20px 30px 20px;

display: flex;

flex-direction: column;

justify-content: space-between;

}

h1 {

text-align: center;

margin-bottom: 15px;

font-weight: 600;

font-size: 1.75rem;

color: #805b00;

}

form {

flex-grow: 1;

overflow-y: auto;

padding-right: 10px;

}

label {

display: block;

margin-top: 15px;

font-weight: 600;

font-size: 0.9rem;

color: #6b4e00;

}

input, select {

width: 100%;

padding: 8px 10px;

margin-top: 6px;

border-radius: 8px;

border: 1.8px solid #c9b866;

font-size: 1rem;

transition: border-color 0.25s ease;

background: #fffbea;

}

input:focus, select:focus {

outline: none;

border-color: #d9a820;

box-shadow: 0 0 6px #d9a820;

}

button {

margin-top: 20px;

width: 100%;

padding: 12px;

border: none;

border-radius: 10px;

background: #d9a820;

color: white;

font-size: 1.1rem;

font-weight: 600;

cursor: pointer;

box-shadow: 0 4px 15px rgba(217, 168, 32, 0.6);

transition: background 0.25s ease;

}

button:hover {

background: #b78700;

}

.result {

margin-top: 25px;

background: #fff8c1;

border-radius: 12px;

padding: 16px;

font-size: 1.2rem;

font-weight: 700;

text-align: center;

color: #6b4e00;

box-shadow: inset 0 0 8px #dfc94e;

}

.footer {

font-size: 0.8rem;

color: #7a6e23;

text-align: center;

margin-top: 10px;

user-select: none;

}

/\* Scrollbar for mobile \*/

form::-webkit-scrollbar {

width: 6px;

}

form::-webkit-scrollbar-thumb {

background-color: #d9a820;

border-radius: 3px;

}

/\* Responsive tweaks \*/

@media(max-width: 360px) {

.container {

max-width: 320px;

padding: 20px 15px 25px 15px;

}

}

</style>

</head>

<body>

<div class="container" role="main" aria-label="Indian House Price Forecasting Application">

<h1>Indian House Price Forecaster</h1>

<form id="priceForm" aria-describedby="form-instr">

<div id="form-instr" style="font-size:0.85rem; color:#6b4e00; margin-bottom:12px;">

Enter house details to forecast the estimated price (in ₹)

</div>

<label for="bedrooms">Bedrooms</label>

<input id="bedrooms" name="bedrooms" type="number" min="0" max="10" value="3" required aria-required="true" />

<label for="bathrooms">Bathrooms</label>

<input id="bathrooms" name="bathrooms" type="number" min="0" max="10" value="2" required aria-required="true" />

<label for="size">Size (sq ft)</label>

<input id="size" name="size" type="number" min="100" max="20000" value="1200" required aria-required="true" />

<label for="location">Location</label>

<select id="location" name="location" required aria-required="true">

<option value="" disabled selected>Select location</option>

<option value="metro">Metro (e.g Mumbai, Delhi)</option>

<option value="urban">Urban</option>

<option value="rural">Rural</option>

</select>

<label for="yearBuilt">Year Built</label>

<input id="yearBuilt" name="yearBuilt" type="number" min="1900" max="2024" value="2010" required aria-required="true" />

<button type="submit" aria-label="Forecast House Price">Forecast Price</button>

</form>

<div class="result" id="result" aria-live="polite" aria-atomic="true" role="region">Estimated price: ₹—</div>

<div class="footer">

&copy; 2024 Indian House Price Forecaster - Smart Regression

</div>

</div>

<script>

(function(){

// Mock multiple linear regression coefficients tailored to Indian real estate market (approximate)

// Price (₹) = intercept + (bedrooms \* coef1) + (bathrooms \* coef2) + (size \* coef3) + (location factor) + (yearBuilt \* coef5)

const locationFactors = {

'metro': 800000, // higher premium for metro cities

'urban': 400000,

'rural': 100000

};

const intercept = 1500000;

const coefBedrooms = 500000;

const coefBathrooms = 400000;

const coefSize = 3000; // ₹3000 per sqft approx

const coefYearBuilt = 20000; // Newer house adds value

const priceForm = document.getElementById('priceForm');

const resultDiv = document.getElementById('result');

const locationSelect = document.getElementById('location');

priceForm.addEventListener('submit', function(event) {

event.preventDefault();

const bedrooms = parseInt(priceForm.bedrooms.value);

const bathrooms = parseInt(priceForm.bathrooms.value);

const size = parseFloat(priceForm.size.value);

const location = priceForm.location.value;

const yearBuilt = parseInt(priceForm.yearBuilt.value);

const currentYear = new Date().getFullYear();

if (isNaN(bedrooms) || bedrooms < 0 || bedrooms > 10) {

alert('Please enter a valid number of bedrooms (0-10).');

priceForm.bedrooms.focus();

return;

}

if (isNaN(bathrooms) || bathrooms < 0 || bathrooms > 10) {

alert('Please enter a valid number of bathrooms (0-10).');

priceForm.bathrooms.focus();

return;

}

if (isNaN(size) || size < 100 || size > 20000) {

alert('Please enter a valid house size in sq ft (100-20000).');

priceForm.size.focus();

return;

}

if (!locationFactors.hasOwnProperty(location)) {

alert('Please select a location from the list.');

locationSelect.focus();

return;

}

if (isNaN(yearBuilt) || yearBuilt < 1900 || yearBuilt > currentYear) {

alert('Please enter a valid year built between 1900 and ' + currentYear + '.');

priceForm.yearBuilt.focus();

return;

}

// Age depreciation factor: Older houses lose value (₹ 40k per year)

const age = currentYear - yearBuilt;

const ageDepreciation = age \* 40000; // ₹40,000 reduced per year old

let estimatedPrice = intercept +

(bedrooms \* coefBedrooms) +

(bathrooms \* coefBathrooms) +

(size \* coefSize) +

locationFactors[location] +

(yearBuilt \* coefYearBuilt) -

ageDepreciation;

estimatedPrice = Math.max(0, estimatedPrice);

// Format as Indian rupees currency string with digit grouping

const formattedPrice = estimatedPrice.toLocaleString('en-IN', {

style: 'currency',

currency: 'INR',

minimumFractionDigits: 0,

maximumFractionDigits: 0

});

resultDiv.textContent = 'Estimated Price: ' + formattedPrice;

});

// Set default location

locationSelect.value = 'urban';

})();

</script>

</body>

</html>

</content>

</create\_file>

# Future scope

While the current house price prediction model provides strong accuracy using smart regression techniques, there are multiple opportunities for future enhancement and broader applicability.

**1. Integration of Real-Time Data**

* Include **live real estate listings**, **interest rates**, or **market trends** to make predictions more dynamic.
* Connect APIs from real estate platforms (e.g., Zillow, Realtor.com).

**2. Use of Geospatial Data**

* Integrate **location-specific factors** such as proximity to schools, hospitals, crime rate, and public transport.
* Use **latitude and longitude** with geospatial libraries like GeoPandas or Folium.

**3. Deep Learning Models**

* Use advanced models like **Neural Networks** (ANNs, CNNs for image inputs) to capture more complex patterns.
* Explore **AutoML** frameworks for model optimization.

**4. Enhanced Frontend and User Interface**

* Upgrade the current web interface using **React** or **Streamlit** for better UX.
* Include **interactive maps**, **charts**, or **recommendations** for users.

**5. Mobile App Deployment**

* Create a mobile app for users to predict prices on-the-go.
* Use frameworks like **Flutter** or **React Native** with model integration via APIs.

**6. Continuous Learning and Model Updating**

* Implement a **pipeline** for automatic model retraining using newly collected property sale data.
* Enable version control and A/B testing of different models.

**7. Multilingual and Global Expansion**

* Extend the application to work for other **countries or regions**, adjusting for currency, housing norms, and language.

**8. 3D Visualization and AR Integration**

* Future versions could support **3D house models** or **Augmented Reality** features to visualize pricing based on structural modifications or upgrades.

The current system lays a solid foundation for accurate house price prediction. Future upgrades will focus on **data richness**, **usability**, and **intelligence**, moving toward a full-fledged smart real estate advisory tool.

# 13. Team Members and Roles

1.TAMILSELVAN.S - WORKED FOR THE PROBLEM STATEMENT,SOURCE CODE,DEPLOYMENT,

2.THILIPAN.P-INVOLVED IN ABSTRACT,DATASET DESCRIPTION,

3.SARAVANAN.P-WORKED FOR MODEL EVALUATION,OBJECTIVES,FLOWCHART

4.SWEDHARSHAN.R-WORKED FOR MODEL BUILDING ,DATA PREPROCESSING

5.AAKASH.C.J-WORKED FOR FEATURE ENGINEERING,EXPLORATORY DATA ANALYSIS

