**Phase-3**

**Student Name:** Lakshmi Devi. V

**Register Number:** 2303717710422025

**Institution:** Government College Of Technology

**Department:** Computer Science and Engineering

**Date of Submission:** 9.5.2025

**Github Repository Link:** https://github.com/Lakshmidevi125110/House-Prediction.git

# Problem Statement

The goal of this project is to predict housing prices based on various features such as the number of bedrooms, area, furnishing status, availability of hot water, and more. This is a **regression** problem because the target variable (price) is continuous. Initially, the problem was understood as a basic prediction task, but upon further exploration of the dataset, additional categorical and numerical variables were discovered that influence the model. Solving this problem is important for real estate agencies, homeowners, and buyers as it assists in setting fair market prices and making informed investment decisions.

# Abstract

This project addresses the challenge of accurately predicting house prices in the real estate market. The main objective is to develop a machine learning model that estimates prices based on structured data like location, area, and number of rooms. The approach includes data collection, EDA, feature engineering, and model development using Linear Regression and Random Forest. A web interface is created using Flask and HTML for easy user interaction. Users can enter property details and receive instant price predictions. The outcome is a reliable and accessible tool to support property valuation and decision-making.

# System Requirements

**Hardware Requirements:**

A Laptop with 8GB RAM

Intel i3 or equivalent processor

**Software Requirements:**

* Python 3.x
* VS Code

Required Python Libraries:

* pandas
* NUMPY
* matplotlib
* seaborn
* scikit-learn
* XGBOOST
* LIGHTGBM
* StreamLit

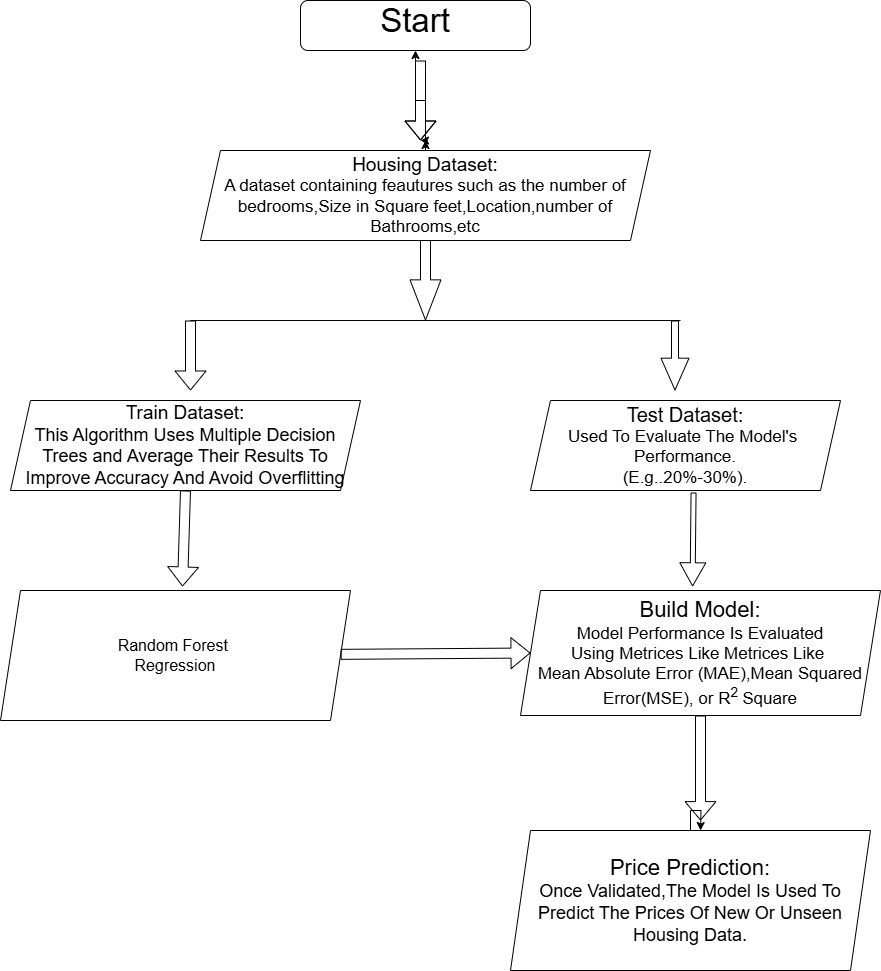
**OPERATING SYSTEM:**

* Windows 11

# Objectives

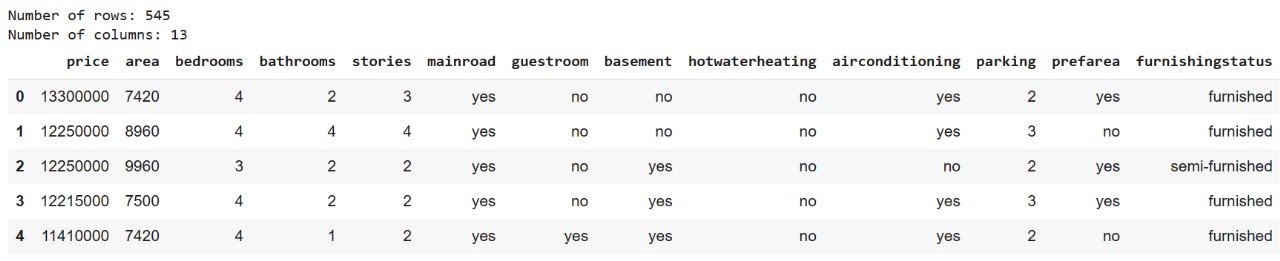
The objective of this project is to develop a machine learning model that predicts house prices based on various input features like location, square footage, and the number of rooms. The model will output accurate price estimates for different properties. By identifying key factors affecting house prices, this project provides insights that help buyers, sellers, and real estate agents make informed decisions. The business impact includes optimizing pricing strategies, improving transaction fairness, and accelerating decision-making in the real estate market.

# Flowchart of Project Workflow

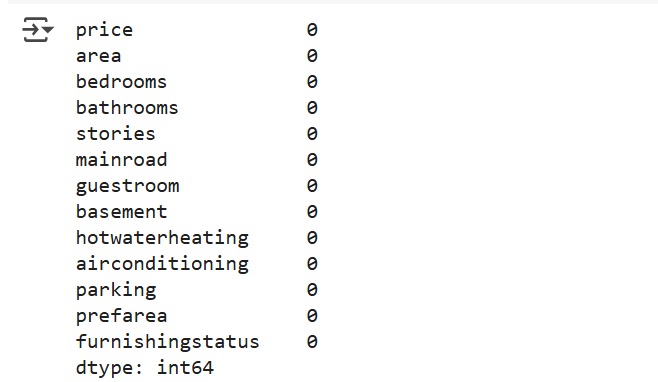


# Dataset Description

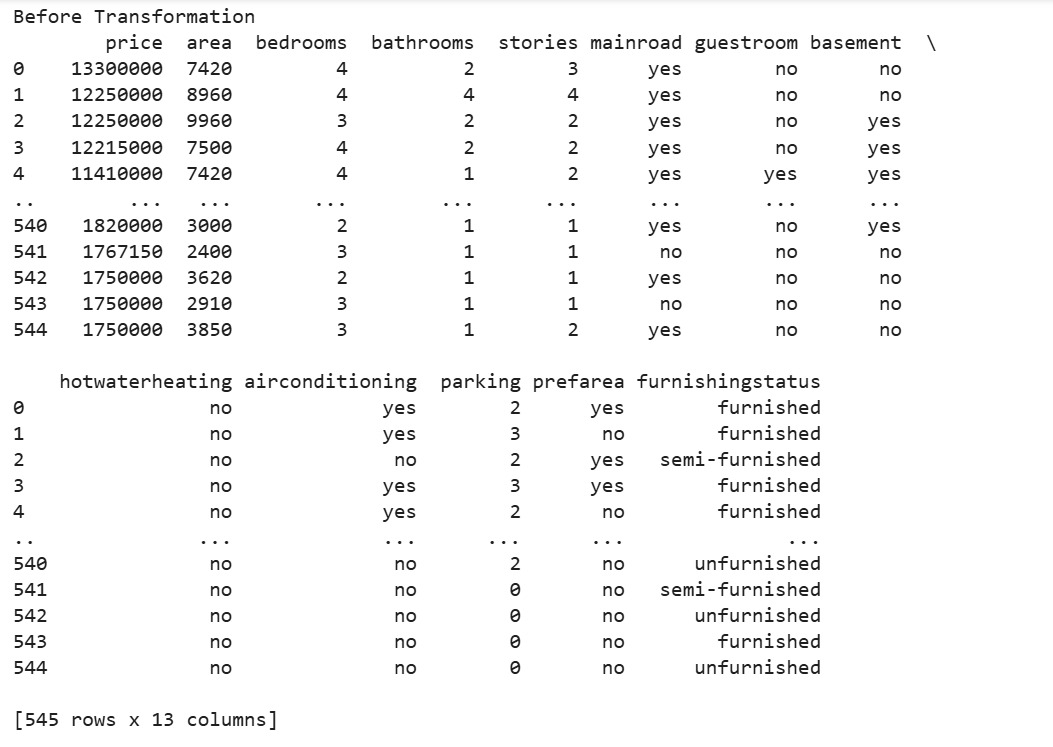
* Source: Kaggle
* Type: public
* Size and structure (number of rows 545/columns 13)

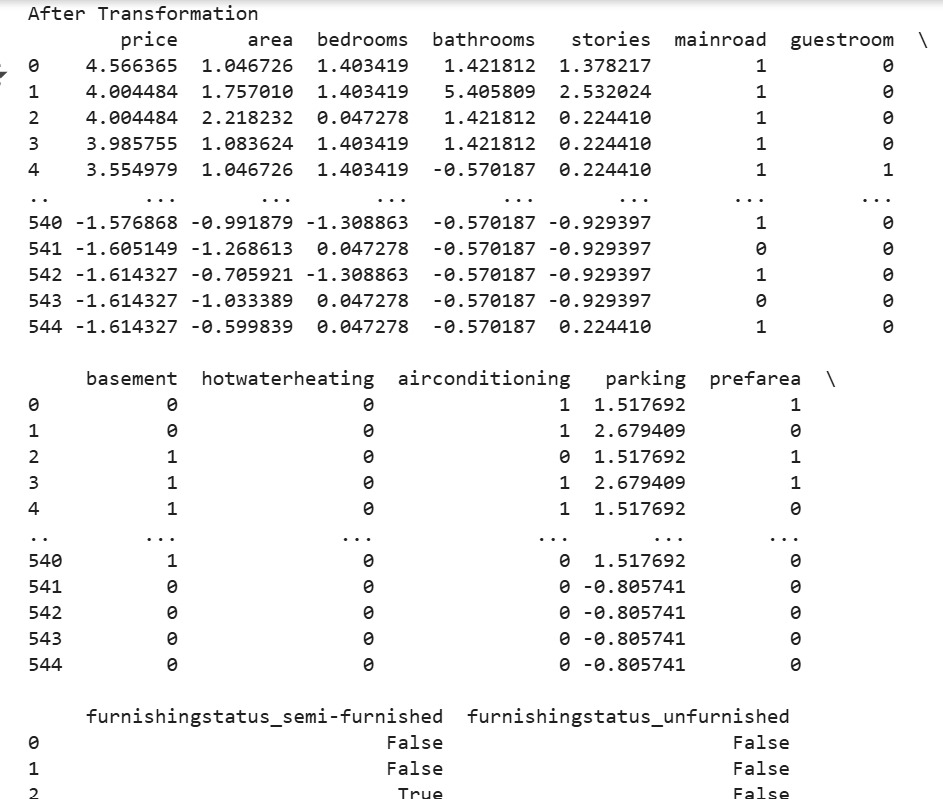


# Data Preprocessing

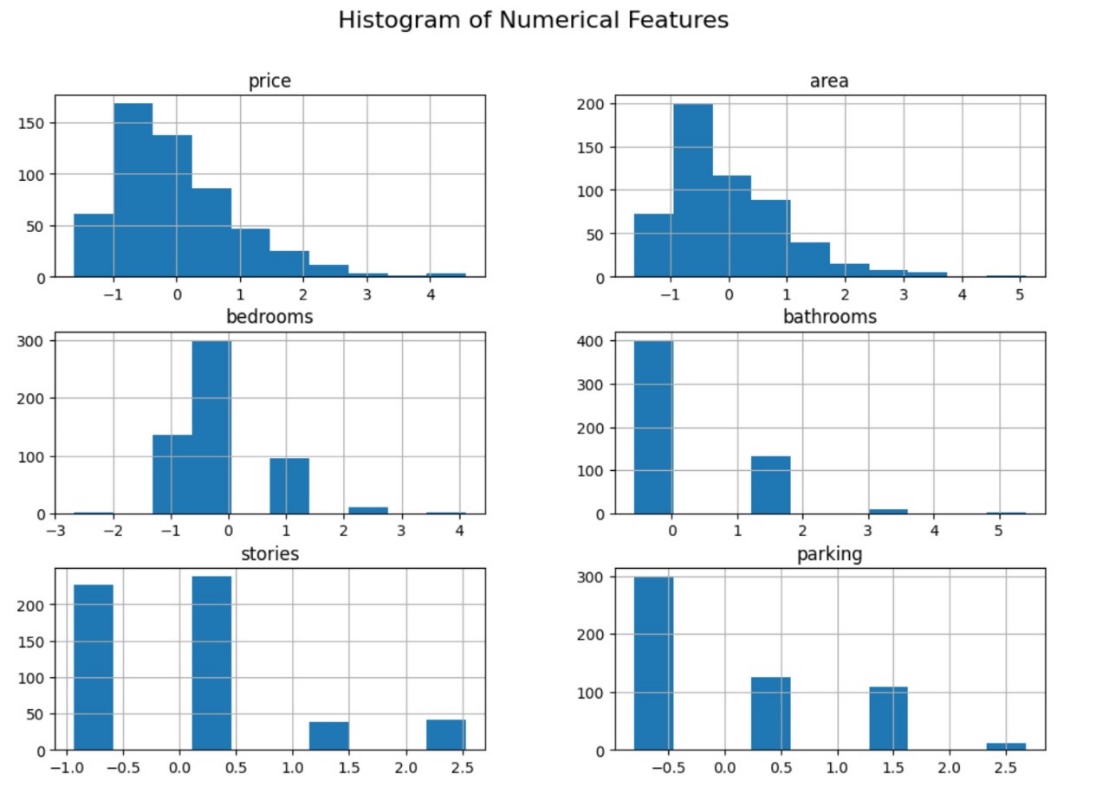


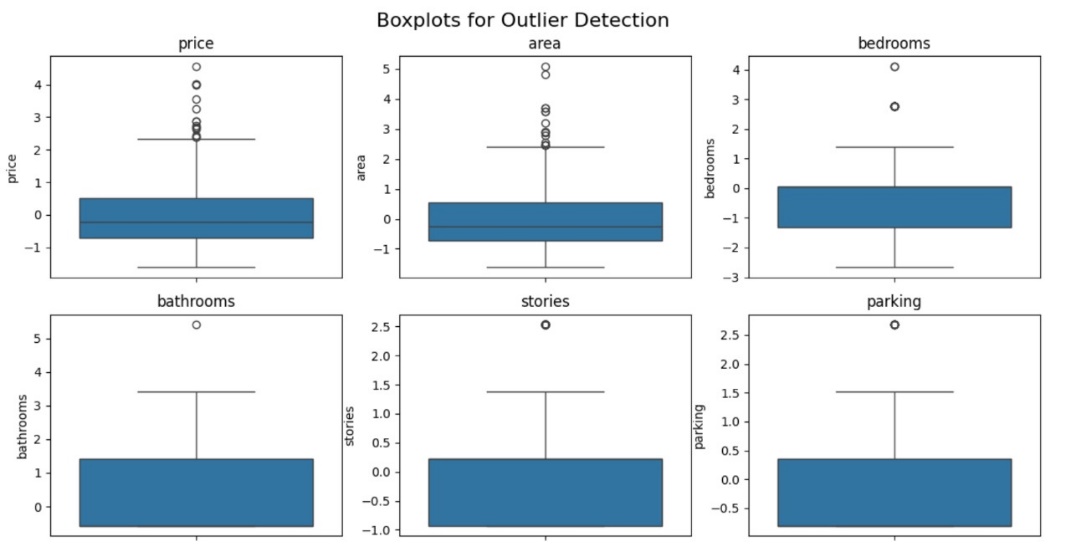
* One Hot encoding and Label encoding

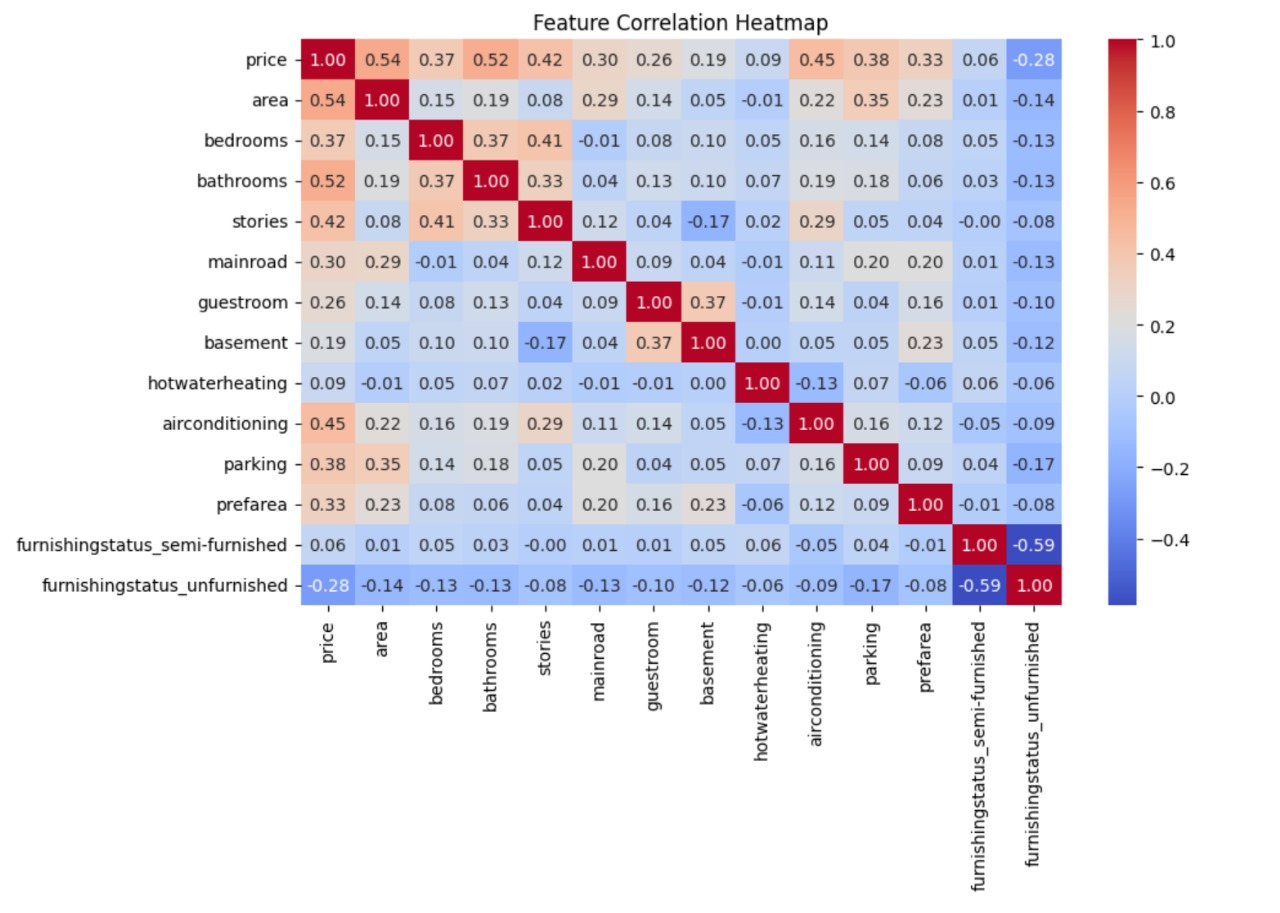




# Exploratory Data Analysis (EDA)







**Distribution:**

* Area, price, and stories show fairly normal distributions after scaling.

bedrooms and parking may have skewed distribution (e.g., few values like 0 or 4 dominating).

**Outliers:**

* Boxplots may highlight some outliers in area or price (though less visible post-scaling).

**Correlations:**

* Price is strongly correlated with area and stories, which makes sense in housing.
* Guestroom, basement, and airconditioning have mild positive correlation with price, suggesting that luxury features increase home value.
* Furnishingstatus\_semi-furnished and furnishingstatus\_unfurnished may show negative correlation with price, depending on encoding.

# 9.Feature Engineering

**1. New Feature Creation**

**Added Features:**

Price\_per\_sqft = price / area

**Why? Gives a normalized view of price across varying house sizes.**

Impact: Helps compare property values independent of size. Useful in real estate pricing models.

Total\_rooms = bedrooms + bathrooms

**Why? Aggregates living space usability into one feature.**

Impact: Simplifies model interpretation and avoids splitting importance across two correlated features.

**2. Feature Selection**

**After feature creation, it’s important to:**

* Remove redundant or less informative features.
* Keep features that add predictive value and reduce noise

# 10.Model Building

* Baseline model: Linear Regression
* Advanced Model: Random Forest Regressor

**Why Linear Regression?**

Linear Regression is a fundamental, easy-to-implement algorithm that models the relationship between input features and a continuous target variable (like house price) by fitting a straight line (or hyperplane in higher dimensions).

**Reasons for choosing Linear Regression:**

**Simplicity and Interpretability:**

* It helps understand how each feature (like area, bedrooms, etc.) impacts the price.
* Coefficients tell you the weight or importance of each feature.

**Fast Training:**

* Very efficient for small to medium-sized datasets like the Housing.csv.

**Baseline Model:**

* It's often used as a baseline model to compare performance with more complex models.
* Works well when data is linearly separable:
* If the relationship between the features and price is roughly linear, it performs well.

**Why Random Forest Regressor?**

Random Forest is an ensemble learning method that builds multiple decision trees and merges their outputs to make predictions. It often gives better accuracy than individual models, especially when the relationships in data are complex.

**Reasons for choosing Random Forest:**

**Captures Non-Linear Relationships:**

* Unlike linear regression, it handles complex interactions and non-linearities between features.

**Robust to Outliers and Noise:**

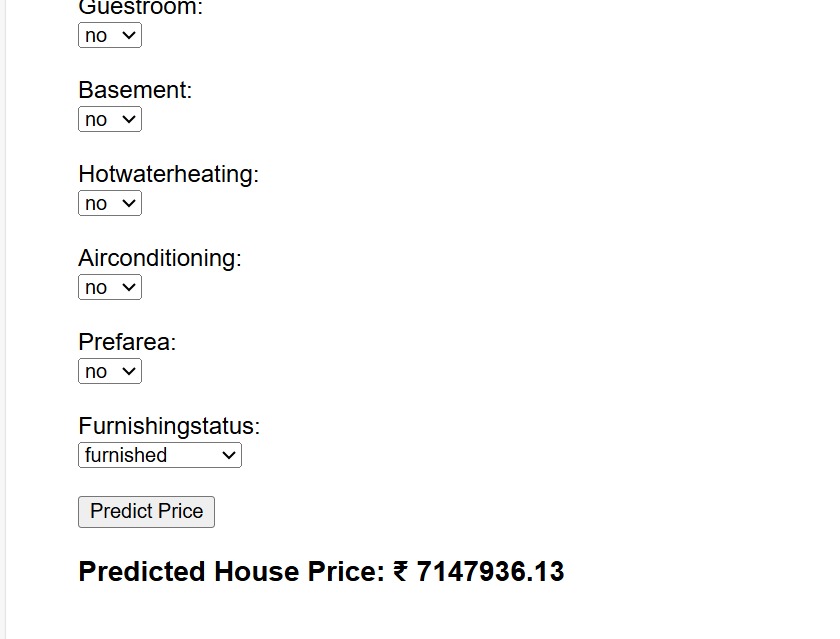
* It reduces overfitting by averaging many trees (low variance).

**Feature Importance:**

It ranks features based on how useful they are in predicting the price, which can help in feature selection.

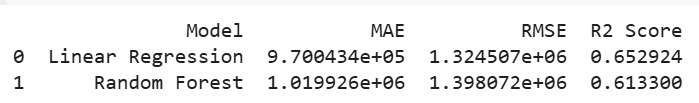
**Good default model:**

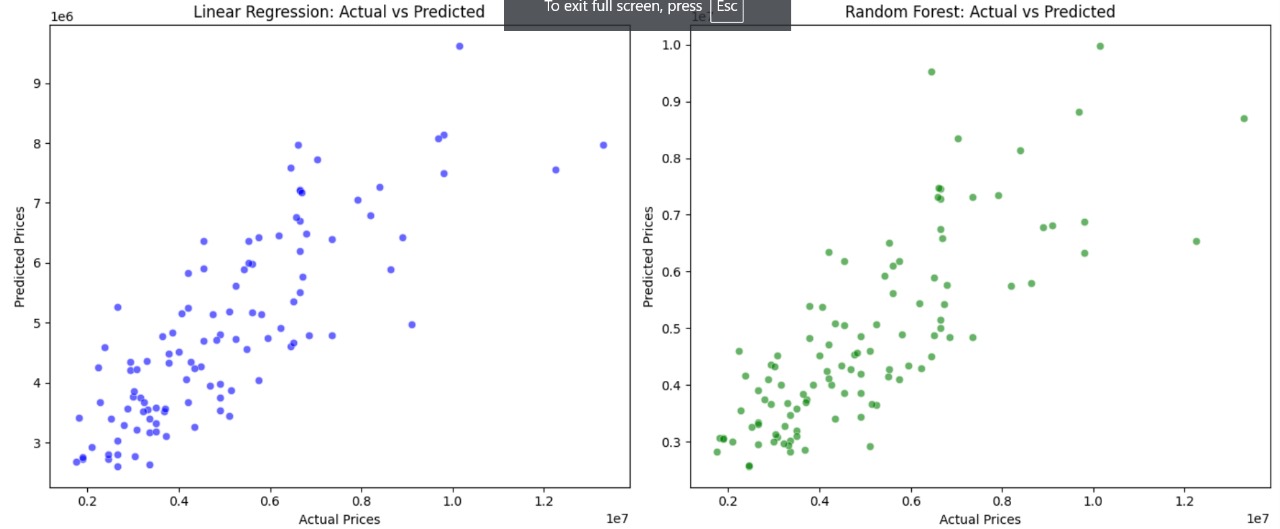
Works well with minimal tuning and on most types of datasets.

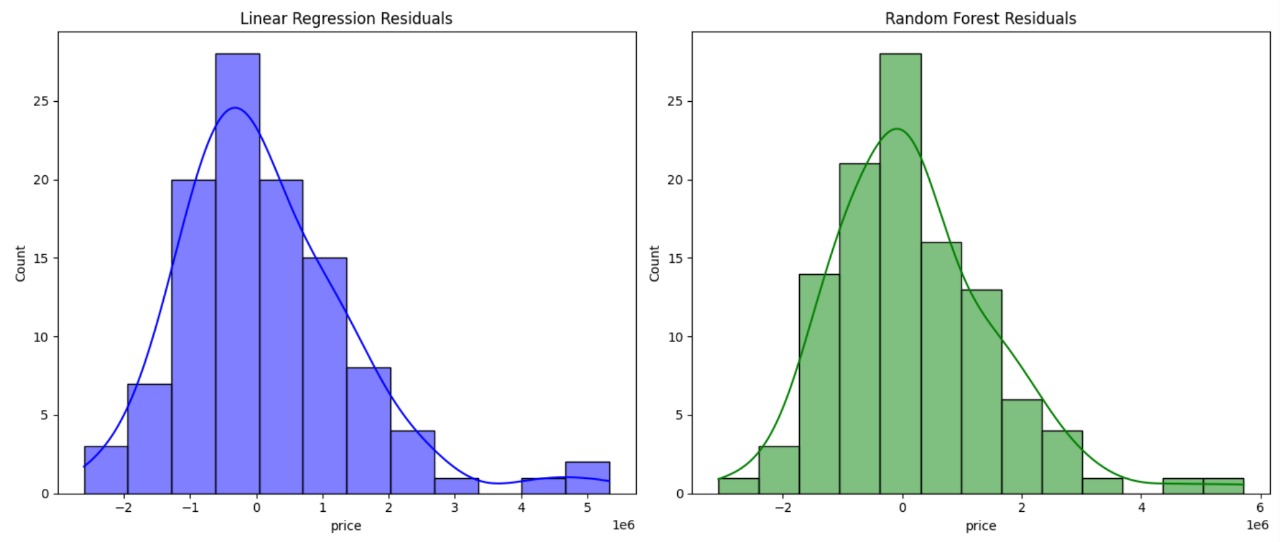




# Model Evaluation







# 

# 11.Deployment

* + Streamlit Cloud
  + Flask API on Render

Deployment method

"Web Service Deployment via GitHub Repository"

**This method includes:**

**Steps Involved:**

* Code is stored on GitHub ( app.py, index.html, housing dataset,

requirements.txt, and Procfile).

* Render connects to our GitHub repo and automatically pulls the code.

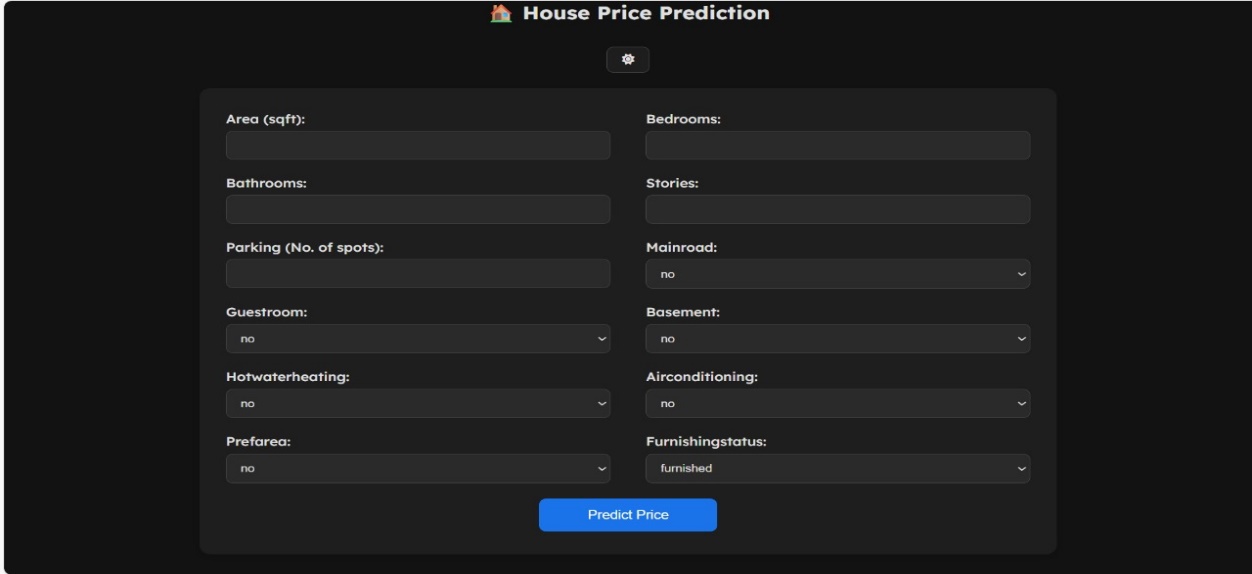
**On deployment:**

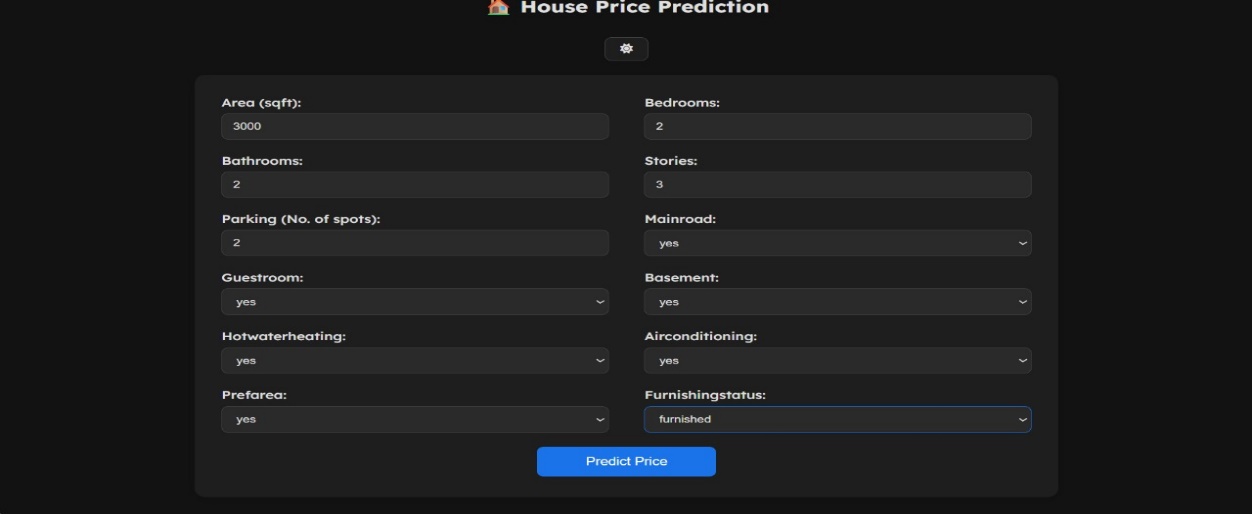
* Render installs dependencies (requirements.txt).
* It uses our Procfile to start the app using a WSGI server like Gunicorn.
* It binds our Flask app to 0.0.0.0 on the $PORT environment variable (assigned by Render).
* Our app is hosted publicly on Render’s infrastructure, and Render handles server provisioning, HTTP routing, and HTTPS.

**Link:**

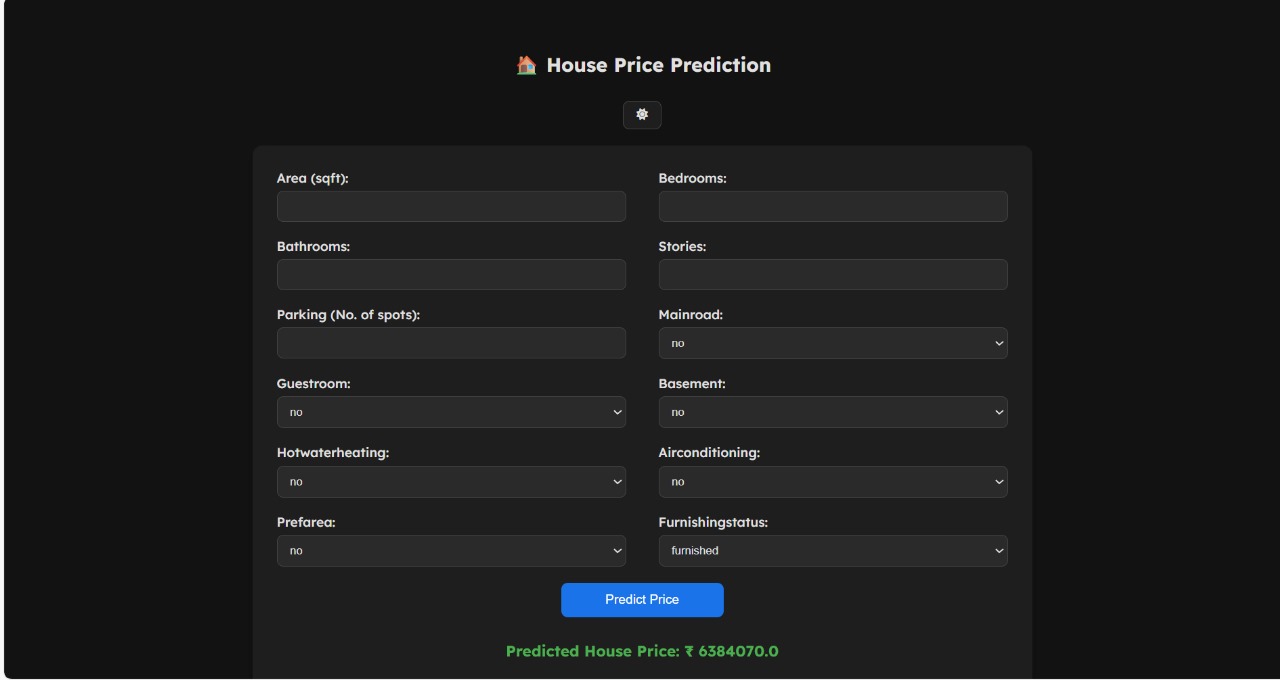
<https://house-prediction-q1yo.onrender.com>

**UI Snapshot:**





**Output:**



# 12.Source code

# Index.html:

# <!DOCTYPE html>

# <html>

# <head>

# <title>House Price Predictor</title>

# <link rel="preconnect" href="https://fonts.googleapis.com">

# <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

# <link href="https://fonts.googleapis.com/css2?family=Lexend:wght@100..900&display=swap" rel="stylesheet">

# <!-- Font Awesome CDN for icons -->

# <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.0/css/all.min.css">

# <style>

# :root {

# --bg-color: #f0f4f8;

# --text-color: #333;

# --form-bg: #fff;

# --input-bg: #fff;

# --border-color: #ccc;

# --btn-bg: #007bff;

# --btn-hover: #0056b3;

# --success-color: #27ae60;

# }

# body.dark {

# --bg-color: #121212;

# --text-color: #e0e0e0;

# --form-bg: #1e1e1e;

# --input-bg: #2a2a2a;

# --border-color: #444;

# --btn-bg: #1a73e8;

# --btn-hover: #0c5adb;

# --success-color: #4caf50;

# }

# body {

# font-family: 'Lexend', sans-serif;

# background-color: var(--bg-color);

# color: var(--text-color);

# padding: 40px;

# transition: all 0.3s ease;

# }

# h2 {

# color: var(--text-color);

# margin-bottom: 30px;

# text-align: center;

# }

# .toggle-container {

# text-align: center;

# margin-bottom: 20px;

# }

# .form-container {

# background: var(--form-bg);

# padding: 30px;

# border-radius: 12px;

# box-shadow: 0 4px 20px rgba(0, 0, 0, 0.1);

# max-width: 900px;

# margin: auto;

# }

# form {

# display: grid;

# grid-template-columns: 1fr 1fr;

# gap: 20px 40px;

# }

# .form-group {

# display: flex;

# flex-direction: column;

# }

# label {

# font-weight: 500;

# margin-bottom: 6px;

# }

# input[type="number"],

# select {

# padding: 10px 12px;

# border: 1px solid var(--border-color);

# background-color: var(--input-bg);

# border-radius: 8px;

# font-size: 14px;

# color: var(--text-color);

# }

# input[type="number"]:focus,

# select:focus {

# border-color: var(--btn-bg);

# outline: none;

# }

# .full-width {

# grid-column: span 2;

# text-align: center;

# }

# input[type="submit"] {

# background-color: var(--btn-bg);

# color: white;

# border: none;

# padding: 12px 20px;

# border-radius: 8px;

# cursor: pointer;

# font-size: 16px;

# width: 200px;

# transition: background-color 0.3s ease;

# }

# input[type="submit"]:hover {

# background-color: var(--btn-hover);

# }

# h3 {

# margin-top: 30px;

# color: var(--success-color);

# text-align: center;

# }

# .dark-toggle {

# cursor: pointer;

# background-color: var(--form-bg);

# border: 1px solid var(--border-color);

# padding: 8px 16px;

# border-radius: 8px;

# font-size: 14px;

# color: var(--text-color);

# transition: background 0.3s;

# }

# .dark-toggle:hover {

# background-color: var(--btn-bg);

# color: white;

# }

# @media (max-width: 768px) {

# form {

# grid-template-columns: 1fr;

# }

# .full-width {

# grid-column: span 1;

# }

# }

# </style>

# </head>

# <body>

# <h2>🏠 House Price Prediction</h2>

# <div class="toggle-container">

# <button class="dark-toggle" onclick="toggleDarkMode()" id="darkModeBtn">

# <i class="fas fa-moon" id="darkIcon"></i>

# </button>

# </div>

# <div class="form-container">

# <form action="/predict" method="post">

# <!-- Left column -->

# <div class="form-group">

# <label>Area (sqft):</label>

# <input type="number" name="area" required>

# </div>

# <div class="form-group">

# <label>Bedrooms:</label>

# <input type="number" name="bedrooms" required>

# </div>

# <div class="form-group">

# <label>Bathrooms:</label>

# <input type="number" name="bathrooms" required>

# </div>

# <div class="form-group">

# <label>Stories:</label>

# <input type="number" name="stories" required>

# </div>

# <div class="form-group">

# <label>Parking (No. of spots):</label>

# <input type="number" name="parking" required>

# </div>

# <div class="form-group">

# <label>Mainroad:</label>

# <select name="mainroad" required>

# {% for value in options.mainroad %}

# <option value="{{ value }}">{{ value }}</option>

# {% endfor %}

# </select>

# </div>

# <!-- Right column -->

# <div class="form-group">

# <label>Guestroom:</label>

# <select name="guestroom" required>

# {% for value in options.guestroom %}

# <option value="{{ value }}">{{ value }}</option>

# {% endfor %}

# </select>

# </div>

# <div class="form-group">

# <label>Basement:</label>

# <select name="basement" required>

# {% for value in options.basement %}

# <option value="{{ value }}">{{ value }}</option>

# {% endfor %}

# </select>

# </div>

# <div class="form-group">

# <label>Hotwaterheating:</label>

# <select name="hotwaterheating" required>

# {% for value in options.hotwaterheating %}

# <option value="{{ value }}">{{ value }}</option>

# {% endfor %}

# </select>

# </div>

# <div class="form-group">

# <label>Airconditioning:</label>

# <select name="airconditioning" required>

# {% for value in options.airconditioning %}

# <option value="{{ value }}">{{ value }}</option>

# {% endfor %}

# </select>

# </div>

# <div class="form-group">

# <label>Prefarea:</label>

# <select name="prefarea" required>

# {% for value in options.prefarea %}

# <option value="{{ value }}">{{ value }}</option>

# {% endfor %}

# </select>

# </div>

# <div class="form-group">

# <label>Furnishingstatus:</label>

# <select name="furnishingstatus" required>

# {% for value in options.furnishingstatus %}

# <option value="{{ value }}">{{ value }}</option>

# {% endfor %}

# </select>

# </div>

# <div class="full-width">

# <input type="submit" value="Predict Price">

# </div>

# </form>

# {% if prediction\_text %}

# <h3>{{ prediction\_text }}</h3>

# {% endif %}

# </div>

# <script>

# // Check localStorage for dark mode preference

# if (localStorage.getItem('darkMode') === 'true') {

# document.body.classList.add('dark');

# document.getElementById('darkIcon').classList.remove('fa-moon');

# document.getElementById('darkIcon').classList.add('fa-sun');

# }

# function toggleDarkMode() {

# document.body.classList.toggle('dark');

# const icon = document.getElementById('darkIcon');

# if (document.body.classList.contains('dark')) {

# icon.classList.remove('fa-moon');

# icon.classList.add('fa-sun');

# localStorage.setItem('darkMode', 'true'); // Save dark mode in localStorage

# } else {

# icon.classList.remove('fa-sun');

# icon.classList.add('fa-moon');

# localStorage.setItem('darkMode', 'false'); // Save light mode in localStorage

# }

# }

# </script>

# </body>

# </html>

# App.py:

# from flask import Flask, request, render\_template

# import pandas as pd

# from sklearn.ensemble import RandomForestRegressor

# from sklearn.preprocessing import OneHotEncoder

# from sklearn.compose import ColumnTransformer

# app = Flask(\_name\_)

# # Load the dataset

# df = pd.read\_csv("Housing.csv")

# # Define features and target

# features = ['area', 'bedrooms', 'bathrooms', 'stories', 'mainroad', 'guestroom',

# 'basement', 'hotwaterheating', 'airconditioning', 'parking',

# 'prefarea', 'furnishingstatus']

# X = df[features]

# y = df['price']

# # Preprocessing

# categorical\_features = ['mainroad', 'guestroom', 'basement', 'hotwaterheating',

# 'airconditioning', 'prefarea', 'furnishingstatus']

# numerical\_features = ['area', 'bedrooms', 'bathrooms', 'stories', 'parking']

# preprocessor = ColumnTransformer(

# transformers=[

# ('cat', OneHotEncoder(drop='first'), categorical\_features)

# ],

# remainder='passthrough' # Keep numerical features

# )

# # Preprocess and train the model

# X\_processed = preprocessor.fit\_transform(X)

# model = RandomForestRegressor(n\_estimators=100, random\_state=42)

# model.fit(X\_processed, y)

# # Get list of possible values for dropdowns

# dropdown\_options = {

# 'mainroad': sorted(df['mainroad'].unique()),

# 'guestroom': sorted(df['guestroom'].unique()),

# 'basement': sorted(df['basement'].unique()),

# 'hotwaterheating': sorted(df['hotwaterheating'].unique()),

# 'airconditioning': sorted(df['airconditioning'].unique()),

# 'prefarea': sorted(df['prefarea'].unique()),

# 'furnishingstatus': sorted(df['furnishingstatus'].unique())

# }

# @app.route('/')

# def home():

# return render\_template('index.html', options=dropdown\_options)

# @app.route('/predict', methods=['POST'])

# def predict():

# # Get values from form

# input\_data = {

# 'area': int(request.form['area']),

# 'bedrooms': int(request.form['bedrooms']),

# 'bathrooms': int(request.form['bathrooms']),

# 'stories': int(request.form['stories']),

# 'mainroad': request.form['mainroad'],

# 'guestroom': request.form['guestroom'],

# 'basement': request.form['basement'],

# 'hotwaterheating': request.form['hotwaterheating'],

# 'airconditioning': request.form['airconditioning'],

# 'parking': int(request.form['parking']),

# 'prefarea': request.form['prefarea'],

# 'furnishingstatus': request.form['furnishingstatus']

# }

# # Convert to DataFrame

# input\_df = pd.DataFrame([input\_data])

# # Transform input

# input\_processed = preprocessor.transform(input\_df)

# # Predict

# predicted\_price = model.predict(input\_processed)[0]

# predicted\_price = round(predicted\_price, 2)

# return render\_template('index.html', prediction\_text=f"Predicted House Price: ₹ {predicted\_price}", options=dropdown\_options)

# #if \_name\_ == '\_main\_':

# #app.run(debug=True)

# if \_name\_ == '\_main\_':

# import webbrowser

# import threading

# def open\_browser():

# webbrowser.open\_new("http://127.0.0.1:5000/")

# threading.Timer(1.25, open\_browser).start()

# app.run(debug=True)

# Future scope

* Real-time data integration from live property websites
* Web/mobile app deployment for user-friendly access
* Incorporating more features like proximity to schools, hospitals, and public transport
* Multi-city support with dynamic datasets
* Deep learning integration for improving accuracy and learning complex patterns
* Integration with GIS systems for map-based predictions

# 

# 13. Team Members and Roles

**1. Lakshmi Devi**

**Role:** Data Collection and Documentation

**Responsibilities:**

Collecting structured data

Validating data sources

Preparing the Software Requirement Specification (SRS) document

**2. Indhuja**

**Role:** Exploratory Analysis and Feature Engineering

**Responsibilities:**

Performing Exploratory Data Analysis (EDA)

Selecting and engineering relevant features from structured data

**3. Subha Paramesh**

**Role:** Model Development

**Responsibilities:**

Designing and developing machine learning models

Implementing Linear Regression and Random Forest algorithms

**4. Mounika**

**Role:** Web Development and Deployment

**Responsibilities:**

Building the web interface using Flask and HTML

Deploying the final model for end-user access