Department Of Computer Applications

Madurai Kamaraj University

House Price Prediction Project

A Machine Learning Based Real Estate Price Estimation System

PROJECT REPORT

Submitted By,

S. Priya

# Table of Contents

|  |
| --- |
| 1. Overview |
| 1. Objective |
| 1. Dataset Description |
| 1. Data Preparation and Cleaning |
| 1. Feature Engineering |
| 1. Model Building |
| 1. Model Evaluation |
| 1. Deployment with Streamlit |
| 1. Example Use Case |
| 1. Results and Conclusion |
| 1. Future Scope |
| 1. Applications |
| 1. Python Code Implementation |

Bengaluru House Price Prediction – DetaileProject Description

1.Overview

● The Bengaluru House Price Prediction project aims to predict the price of residential properties in Bengaluru, India, using machine learning techniques. The real estate market in Bengaluru is one of the fastest-growing in India, with thousands of properties being bought and sold each month.

● Predicting the right house price can help buyers make informed decisions and prevent overpaying, while sellers can use this tool to list properties at competitive prices. The model uses various.

● Features such as the property's location, area type, size in square feet, number of bedrooms, and number of bathrooms to estimate an accurate price.

2.Objective

● The main objective of this project is to develop a predictive system that can accurately estimate the price of a house in Bengaluru based on the key property features.

● This project demonstrates how machine learning can be applied to solve real-world problems in the real estate sector and make housing data more transparent and accessible.

3.Dataset Description

● The dataset used in this project is the Bengaluru House Data, which was originally published on Kaggle. It contains thousands of records of residential properties across different localities in Bengaluru.

● Each record represents one property with multiple attributes that describe its characteristics and price. These attributes help the model learn relationships between house features and prices.

Key features include:

* area\_type – Describes the measurement type such as Super built-up Area, Plot Area, or Carpet

Area.

* availability – Indicates when the property is available for possession (e.g., Ready To Move,

Immediate).

* location – The locality of the property within Bengaluru.
* size – The number of bedrooms (e.g., 2 BHK, 3 BHK).
* society – The housing society or community name.
* total\_sqft – The total area of the house in square feet.
* bath – Number of bathrooms.
* balcony – Number of balconies.
* price – The selling price of the property in lakhs of Indian Rupees (INR).

4.Data Preparation and Cleaning

● Data preprocessing is an essential step in building an accurate predictive model. The dataset had several missing and inconsistent values, which were handled by removing or replacing them appropriately.

● The ‘size’ column, which contained values like '2 BHK' or '3 Bedroom', was converted

to a numerical format. The ‘total\_sqft’ column also had inconsistent formats such as ranges (e.g.,1200–1500), which were replaced with average values.

● Duplicate entries and outliers were removed to ensure that the model learns only from valid and realistic data.

5. Feature Engineering

● Feature engineering helps the model understand the dataset better. Categorical variables such as ‘area\_type’ and ‘location’ were converted into numerical format using encoding techniques.

● The ‘price per square foot’ feature was also calculated to better understand the relationship between size and price.

● Outlier detection techniques were applied to remove unrealistic values such as

extremely low or high prices.

6.Model Building

● After cleaning and preparing the dataset, several machine learning algorithms were tested, such as Linear Regression, Decision Tree, and Random Forest Regressor. The dataset was divided into training and test sets to evaluate performance.

● Linear Regression gave a good balance of simplicity and accuracy, while Random Forest provided slightly higher accuracy due to its ability to handle nonlinear relationships between features and price.

7.Model Evaluation

● To measure how well the model performs, evaluation metrics such as R² Score, Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE) were used.

● A high R² value indicates that the model can explain most of the variation in the data. Visualizations such as scatter plots between actual and predicted prices were also used to assess model accuracy.

8.Deployment with Streamlit

● The final trained model was deployed as an interactive web application using Streamlit.

The app allows users to input house details such as location, square feet, number of bedrooms, and bathrooms, and receive an instant price prediction.

● Streamlit provides an easy-to-use interface and allows anyone to access the prediction model without coding knowledge.

9.Example Use Case

If a user enters the following details: Location – Whitefield, Area Type – Super built-up Area, Total Sqft – 1200, Bath – 2, BHK – 3, the model may predict a price around ■78.5 Lakhs.

The output can help users estimate the fair value of the property and negotiate accordingly.

# Python Code Implementation

# model\_training.py

import pandas as pd

import joblib

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

from sklearn.ensemble import RandomForestRegressor

from sklearn.impute import SimpleImputer

# -------------------------------

# Load dataset

# -------------------------------

df = pd.read\_csv('Bengaluru\_House\_Data.csv')

# Drop rows with missing target or total\_sqft

df = df.dropna(subset=['total\_sqft', 'price'])

# -------------------------------

# Convert total\_sqft to numeric

# -------------------------------

def convert\_sqft(x):

try:

x = str(x)

if '-' in x: # range like "1200 - 1500"

tokens = x.split('-')

return (float(tokens[0].strip()) + float(tokens[1].strip())) / 2

if 'Sq. Yards' in x:

return float(x.replace('Sq. Yards','').strip()) \* 9

if 'Acres' in x:

return float(x.replace('Acres','').strip()) \* 43560

return float(x)

except:

return None

df['total\_sqft'] = df['total\_sqft'].apply(convert\_sqft)

df = df.dropna(subset=['total\_sqft'])

# -------------------------------

# Extract BHK from size

# -------------------------------

def extract\_bhk(size\_str):

try:

return int(size\_str.split()[0])

except:

return 0

df['bhk'] = df['size'].apply(extract\_bhk)

# -------------------------------

# Show dataset info

# -------------------------------

print("✅ Bengaluru House dataset loaded and cleaned!")

print("Shape:", df.shape)

print("\n--- First 5 rows ---")

print(df.head())

print("\n--- Dataset Info ---")

print(df.info())

print("\n--- Missing Values ---")

print(df.isnull().sum())

print("\n--- Statistical Summary ---")

print(df.describe())

# -------------------------------

# Features and target

# -------------------------------

X = df.drop(['price', 'size'], axis=1)

y = df['price']

# -------------------------------

# Define numeric and categorical features

# -------------------------------

numerical\_features = ['total\_sqft', 'bath', 'balcony', 'bhk']

categorical\_features = ['area\_type', 'availability', 'location', 'society']

# -------------------------------

# Preprocessing pipeline with imputation

# -------------------------------

numeric\_transformer = Pipeline(steps=[

('imputer', SimpleImputer(strategy='mean')),

('scaler', StandardScaler())

])

categorical\_transformer = Pipeline(steps=[

('imputer', SimpleImputer(strategy='constant', fill\_value='Unknown')),

('onehot', OneHotEncoder(handle\_unknown='ignore'))

])

preprocessor = ColumnTransformer([

('num', numeric\_transformer, numerical\_features),

('cat', categorical\_transformer, categorical\_features)

])

# -------------------------------

# Full pipeline

# -------------------------------

pipeline = Pipeline([

('preprocessor', preprocessor),

('model', RandomForestRegressor(n\_estimators=200, max\_depth=20, random\_state=42))

])

# -------------------------------

# Train/Test split

# -------------------------------

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# -------------------------------

# Train model

# -------------------------------

pipeline.fit(X\_train, y\_train)

# -------------------------------

# Evaluate

# -------------------------------

r2 = pipeline.score(X\_test, y\_test)

print(f"\n🎯 Model trained! R² Score: {r2:.3f}")

# -------------------------------

# Save model

# -------------------------------

joblib.dump(pipeline, 'bangalore\_house\_price\_model.pkl')

print("💾 Model saved as 'bangalore\_house\_price\_model.pkl'")

# app.py

import streamlit as st

import pandas as pd

import joblib

import os

# -------------------------------

# File paths

# -------------------------------

MODEL\_FILE = 'bangalore\_house\_price\_model.pkl'

DATA\_FILE = 'Bengaluru\_House\_Data.csv'

# -------------------------------

# Load model

# -------------------------------

if not os.path.exists(MODEL\_FILE):

st.warning("Model not found. Please run model\_training.py first!")

else:

pipeline = joblib.load(MODEL\_FILE)

st.success("✅ Model loaded successfully!")

# -------------------------------

# Helper function for BHK

# -------------------------------

def extract\_bhk(size\_str):

try:

return int(size\_str.split()[0])

except:

return 0

# -------------------------------

# Streamlit UI

# -------------------------------

st.title("🏠 House Price Prediction")

st.write("Enter house details to predict the price:")

# Numeric inputs

total\_sqft = st.number\_input("Total Area (sqft)", min\_value=100.0, value=1000.0, step=10.0)

bath = st.number\_input("Bathrooms", min\_value=0, value=2, step=1)

balcony = st.number\_input("Balconies", min\_value=0, value=1, step=1)

size = st.text\_input("Size (e.g., 2 BHK)", value="2 BHK")

bhk = extract\_bhk(size)

# Load dataset for dynamic dropdowns

df = pd.read\_csv(DATA\_FILE)

area\_type = st.selectbox("Area Type", df['area\_type'].unique())

availability = st.selectbox("Availability", df['availability'].unique())

location = st.selectbox("Location", df['location'].unique())

society = st.text\_input("Society Name", value="Unknown")

# Prepare input

input\_df = pd.DataFrame([{

'total\_sqft': total\_sqft,

'bath': bath,

'balcony': balcony,

'bhk': bhk,

'area\_type': area\_type,

'availability': availability,

'location': location,

'society': society

}])

# Prediction

if st.button("Predict Price"):

prediction = pipeline.predict(input\_df)[0]

# Convert from lakhs to INR

price\_in\_inr = prediction \* 5000

if price\_in\_inr >= 1e7:

st.success(f"🏠 Predicted Price: ₹{price\_in\_inr/1e7:.2f} Crores")

else:

st.success(f"🏠 Predicted Price: ₹{price\_in\_inr/1e5:.2f} Lakhs")

10.Results and Conclusion

● The Bengaluru House Price Prediction project successfully demonstrates the power of data-driven approaches in the real estate market.

● The machine learning model built can predict house prices with reasonable accuracy and can be further improved with more data and additional features like amenities, property age, and proximity to city centers.

● The project highlights the complete lifecycle of a data science project – from data collection to deployment.

11.Future Scope

● In the future, the model can be expanded by integrating live real estate data from APIs or government sources.

● More advanced models such as Gradient Boosting or Neural Networks can also be implemented for higher accuracy.

● Adding visualization dashboards and integrating Google Maps APIs can help users explore location-based price trends.

12.Applications

* Real estate price estimation and comparison tools.
* Investment advisory for property buyers.
* Market trend analysis for developers.
* Real estate research and analytics dashboards.

Output:-

