HOUSE APPRAISAL SYSTEM

MINI PROJECT REPORT

Submitted to the APJ Abdul Kalam Technological University in partial fulfilment of requirements for the award of degree

Bachelor of Technology
in
Computer Science and Engineering
by

ABIJITH G.	-	**JCE22CS002**
ASWIN P S.	-	**JCE22CS017**
ASWIN S.	-	**JCE22CS018**
YADUKRISHNAN T M.	-	**JCE22CS066**

Guided by

Ms. NISNA HANEEFA K.

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

JAWAHARLAL COLLEGE OF ENGINEERING AND TECHNOLOGY

LAKKIDI, MANGALAM P.O, PALAKKAD, KERALA – 679301

APRIL 2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Mini Project report titled **"HOUSE APPRAISAL SYSTEM"** submitted by **ABIJITH G.** (JCE22CS002), **ASWIN P S.** (JCE22CS017), **ASWIN S.** (JCE22CS018) & **YADUKRISHNAN T M.** (JCE22CS066) to the APJ Abdul Kalam Technological University, Kerala in partial fulfilment of the B.Tech. Degree in Computer Science and Engineering is a bonafide record of the Mini Project work carried out by him/her under the guidance **Ms. NISNA HANEEFA K.** This report in any form has not been submitted to any other University or Institute for any purpose.

Guide   **Head of the Department**

Ms. Nisna Haneefa K.   **Dr. R. Satheesh Kumar.**

Assistant Professor,

Department Of CSE

JAWAHARLAL COLLEGE
OF ENGINEERING AND TECHNOLOGY
JAWAHAR GARDENS, LAKKIDI, MANGALAM, PALAKKAD 679 301
(AN AUTONOMOUS INSTITUTION AFFILIATED TO APJ-AKTU)
APPROVED BY AICTE | ACCREDITED BY NAAC "A+"
NBA ACCREDITED | ISO 9001:2015 CERTIFIED INSTITUTION

ACCREDITED PROGRAMMES
(B.TECH - AERO, CIVIL, CSE)

INSTITUTION VISION

Emerge as a centre of excellence for professional education to produce high quality engineers and entrepreneurs for the development of the region and the Nation.

INSTITUTION MISSION

■ To become an ultimate destination for acquiring latest and advanced knowledge in the multidisciplinary domains.

■ To provide high quality education in engineering and technology through innovative teaching learning practices, research and consultancy, embedded with professional ethics.

■ To promote intellectual curiosity and thirst for acquiring knowledge through outcome-based education.

■ To have partnership with industry and reputed institutions to enhance the employability skills of the students and pedagogical pursuits.

■ To leverage technologies to solve real life societal problems through community services.

DEPARTMENT VISION

To produce competent professionals with research and innovative skills, by providing them with the most conducive environment for quality academic and research oriented undergraduate and postgraduate education along with moral values committed to build a vibrant nation.

DEPARTMENT MISSION

■ Provide a learning environment to develop creativity and problem-solving skills in a professional manner.

■ Exposure to latest technologies and tools used in the field of computer science.

■ Provide a platform to explore the industries to understand the work culture and expectation of an organization.

■ Enhance Industry Institute Interaction program to develop entrepreneurship skills.

■ Develop research interest among students which will impart a better life for the society and the nation.

PROGRAM OUTCOMES (POs)

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

The Graduates in Computer Science and Engineering will be able to:

PEO1: Provide high quality knowledge in computer science and engineering required for a computer professional to identify and solve problems in various application domains.

PEO2: Persist with the ability in innovative ideas in computer support systems and transmit the knowledge and skills for research and advanced learning.

PEO3: Manifest the motivational capabilities and turn on social and economic commitment to community services.

PROGRAM SPECIFIC OUTCOMES (PSOs)

The Students in Computer Science and Engineering will be able to

PSO1: Use fundamental knowledge of mathematics to solve problems using suitable analysis methods, data structure and algorithms.

PSO2: Interpret the basic concepts and methods of computer systems and technical specifications to provide accurate solutions.

PSO3: Apply theoretical and practical proficiency with a wide area of programming knowledge, design new ideas and innovations towards research.

DECLARATION

We undersigned hereby declare that the Mini Project report "**HOUSE APPRAISAL SYSTEM**", submitted for partial fulfilment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala, is a Bonafide work done by us under the guidance of **Ms. Nisna Haneefa K.**, Assistant Professor, Department of Computer Science and Engineering, Jawaharlal College of Engineering and Technology. This submission represents our ideas in our own words and where ideas or words of others have been included we have adequately and accurately cited and referenced the original sources. We also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in our submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

ABIJITH G.

ASWIN P S.

LAKKIDI
ASWIN S.

05-04-2025
YADUKRISHNAN T M.

ACKNOWLEDGEMENT

ABSTRACT

This study focuses on predicting house prices using machine learning to assist buyers in making informed decisions based on specific preferences such as location, number of rooms, and budget. Traditional methods often overlook future market trends, leading to inaccurate valuations. The proposed system addresses these issues by using real-world data and applying regression-based machine learning algorithms to predict prices effectively.

The process involves four key stages: data collection, data preprocessing (handling missing values and formatting), model development using machine learning algorithms, and finally training, testing, and validating the model. Techniques like Linear Regression, Decision Tree, K-Means, and Random Forest Regression are employed, with Random Forest showing the highest accuracy in results.

The goal is to provide a reliable tool for predicting property values tailored to individual requirements without relying on brokers. This model not only enhances prediction accuracy but also reflects current and future market trends. It allows buyers to make smarter investments and sellers to price competitively based on data-driven insights.

Furthermore, the system improves transparency in the real estate sector by reducing dependency on subjective human judgment. With constant updates and learning from new data, the model adapts to market fluctuations over time. This makes it a scalable, efficient, and user-friendly solution for real estate forecasting and decision-making.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ML	-	Machine Learning
GIS	-	Geographic Information System
SVM	-	Support Vector Machines
CNN	-	Convolution Neural Network
RNN	-	Recurrent Neural Network
MAE	-	Mean Absolute Error
RMSE	-	Root Mean Squared Error
MSE	-	Mean Squared Error
ANN	-	Artificial Neural Network
SVR	-	Support Vector Regression
CMA	-	Comparative Market Analysis
DFD	-	Data Flow Diagram
AR	-	Augmented Reality

CHAPTER 1

INTRODUCTION

Housing is a fundamental human need, essential for shelter and livelihood, and its demand has surged as living standards have improved. The housing market significantly impacts a country's economy, influencing currency, consumer spending, and construction activity. With increasing urbanization and migration, particularly in cities, predicting housing prices has become a crucial task for homeowners, real estate agents, and investors. Accurate house price predictions are necessary for stakeholders to make informed decisions in a constantly fluctuating market.

In the past, predicting housing prices was done manually, which led to errors and potential financial losses. With the rise of Machine Learning (ML), these predictions have become more precise, as ML models can analyse large datasets and predict house prices based on various factors like location, house type, and construction year. Machine Learning algorithms, such as Linear Regression, Decision Tree Regression, K-Means Regression, and Random Forest Regression, are now used for this purpose. These models are trained on datasets, with 80% of data used for training and 20% for testing, improving prediction accuracy.

However, challenges remain in regions, where reliable datasets are scarce, and forecasting tools are not widely adopted. In such markets, forecasting house prices is difficult due to a lack of scientific research and data. As a result, real estate companies are working to develop algorithms for more accurate property price predictions, highlighting the importance of data and machine learning in the real estate industry. This research area is critical for creating efficient tools for price prediction in emerging markets.

1.1 OBJECTIVE

The primary objective of this project is to develop an advanced and configurable house price prediction system leveraging machine learning techniques. Through this effort, we aim to address the challenges prevalent in current real estate price prediction systems, including limited accuracy and efficiency, particularly in diverse market conditions.

Firstly, we aim to design a robust architecture capable of accommodating various configurations to adapt to different housing markets and customer requirements. This flexibility will allow homeowners, real estate agents, and investors to customize the prediction system according to specific needs, enhancing its applicability across a wide range of scenarios, from individual property purchases to large-scale real estate investments.

Secondly, our goal is to integrate advanced data pre-processing techniques to clean and transform housing datasets effectively. By applying specialized algorithms optimized for real estate data, we aim to mitigate challenges such as missing values, outliers, and inconsistencies, which often hinder accurate price prediction.

Furthermore, we plan to incorporate machine learning methodologies into the prediction framework. Specifically, regression models like linear regression, decision tree regression, and random forest regression, will be trained on annotated housing datasets. These models will enable the system to predict house prices based on various input parameters, such as location, size, number of rooms, and other relevant features, with high accuracy and efficiency.

Moreover, we seek to optimize the computational efficiency of the prediction system to ensure real-time performance, which is essential for applications such as online property platforms and real estate analysis tools. This optimization will involve fine-tuning the machine learning models and streamlining the data processing pipeline to achieve a balance between accuracy and speed.

Overall, the objectives of this project focus on improving the accuracy, efficiency, configurability, and computational performance of house price prediction systems through machine learning techniques. By achieving these objectives, we aim to contribute to advancements in real estate analysis, making property investment decisions more data-driven and accessible.

1.2 PROBLEM DEFINITION

The main task of house price prediction through machine learning is to create a correct and consistent model that predicts the market price of houses based on a given list of influencing factors. These characteristics could include structural traits (e.g., square area, number of bedrooms, age of the house), locational attributes (e.g., closeness to schools, crime indexes, public transit), and macroeconomic variables (e.g., interest rates, inflation). The difficulty is pinpointing the most predictive variables, while considering non-linear relationships, multicollinearity, and changing market environments. Precise price forecasting helps buyers, sellers, and real estate professionals make better decisions, thus mitigating the danger of overpaying or underestimating. Technically, the challenge entails preprocessing of heterogeneous data sources, dealing with missing values, and feature normalization to guarantee robust models.

Machine learning models like linear regression, decision trees, random forests, and gradient boosting are widely used, with deep learning methods like neural networks being investigated for the detection of complex patterns. A major challenge is making the model generalizable across various geographic areas and time shifts since housing markets are extremely responsive to local economic changes and policy interventions. Moreover, interpretability is still an issue, since stakeholders will tend to demand clear explanations of price forecasts over the use of black-box models. Buyers experience a number of difficulties in the property market that amplify the need for precise price forecasting. Information asymmetry where agents or sellers have better information regarding the actual value of a property is typically responsible for skewed pricing.

In addition, purchasers can have difficulty with affordability analysis because of fluctuating interest rates and concealed expenses (e.g., maintenance, taxes) that are not necessarily factored into listing prices. Emotional choice and restricted access to complete historical transaction data also make it challenging to fairly assess fair market value. A well-designed machine learning system can address these concerns by giving buyers data-driven insights, thus enabling them to negotiate with confidence and place profitable investments.

1.3 OUTLINE

Chapter 2 explains the background which includes the literature survey, current scenario and methodologies. Chapter 3 deals with the proposed approach which includes description of work done, algorithm used and some limitations. Chapter 4 describes the design and implementation details which includes software requirements, data design, architectural design, procedural design, testing and validations, simulation and results and performance comparison. Architectural design contains data and control flow diagrams. Procedural design contains design language description, major module description and proposed method and implementation. Chapter 5 deals with the conclusions made and chapter 6 gives a scope of future work.

CHAPTER 2

BACKGROUND

House price prediction began with traditional appraisals, where experts manually estimated property values using local knowledge and comparable sales. These methods were often slow, subjective, and inconsistent due to human bias and limited data. In the late 20th century, statistical techniques like linear regression emerged, allowing for more objective, data-driven predictions by analyzing historical data and identifying how factors such as location, size, and amenities influenced prices.

In parallel, the development of **Geographic Information Systems (GIS)** and **digital property databases** enhanced access to spatial and transactional data, improving the accuracy of traditional models. However, early models still struggled with capturing complex, non-linear relationships in real estate data, especially in rapidly changing markets.

The rise of **machine learning** in the 21st century brought new capabilities for house price prediction. Researchers and developers began applying algorithms such as **decision trees, support vector machines (SVMs), and random forests**, which could learn intricate patterns from large datasets without relying on explicit programming or domain-specific rules.

A significant breakthrough came with the emergence of **deep learning**, particularly **neural networks**. These models, especially **convolutional neural networks (CNNs)** and **recurrent neural networks (RNNs)**, provided improved performance by automatically learning features from both structured and unstructured data such as images of properties or historical time-series data.

The rise of large-scale real estate datasets from platforms like Zillow, Redfin, and government sources has enabled the development of advanced machine learning models with high prediction accuracy. These datasets include detailed property information, transaction history, economic data, and even satellite imagery. Today, house price prediction has evolved from manual methods to AI-driven systems that leverage real-time data and predictive analytics. Ongoing research focuses on improving model adaptability and transparency, making property valuations more accurate, fair, and useful for all stakeholders.

2.1 LITERATURE SURVEY

1. **Ayush Chauhan & Anubhav Singh – Advanced Machine Learning Algorithms for House Price Prediction**
   This study applies Support Vector Regression (SVR), Artificial Neural Networks (ANN), and Linear Regression to forecast housing prices. It emphasizes the role of feature selection in improving model accuracy. However, it notes limited use of deep learning and potential dataset-specific limitations. The work is useful for understanding traditional ML-based price estimation techniques.

2. **Predicting House Prices with Machine Learning Methods-**
   Nisha Dhaka, Avantika Chaudary, Dhrithi Sisodia, Mayank Sharma, Satish Babu.
   This paper compares the performance of k-Nearest Neighbors (k-NN) and Random Forest Regression using the Ames housing dataset. It concludes that Random Forest is well-suited for complex datasets, whereas k-NN performs better with smaller datasets but struggles with high-dimensional data. The comparison gives insight into model suitability based on data complexity.

3. **"House Price Prediction via Visual Cues and Estate Attributes"-**
   Sai Surya Vaddi, Amir Yousif, Samah Baraheem, Ju Shen,Tam V.Nguyen.
   This novel approach incorporates Convolutional Neural Networks (CNNs) along with textual features for improved prediction accuracy. The integration of image-based and attribute-based data demonstrates promise for AI-driven real estate tools. However, it demands large datasets and comes with high computational costs.

4. **"House Price Prediction Using Machine Learning"-Prof.J.Kalidass, T.Dharshalini, R.Nivetha, AP.Subasri.**
   This work employs Multiple Regression and Random Forest models. The Random Forest algorithm is shown to significantly enhance prediction reliability, and feature engineering is key to boosting performance. The study is practical for real-world applications, although deep learning techniques remain underexplored.

2.2 EXISTING SYSTEM

In the current landscape, **house price prediction systems** have advanced with the integration of **real-time data pipelines**, **interactive platforms**, and **AI-powered analytics**.

Modern methodologies primarily rely on **machine learning models**, such as **XGBoost, Random Forests, Gradient Boosting, and Neural Networks**, trained on large datasets comprising housing features (e.g., size, location, number of rooms), historical prices, and economic variables. These models excel in capturing non-linear relationships and can generalize well across different market conditions.

Real-time prediction systems are deployed via **web and mobile applications** where users can enter property details to get instant price estimates. These platforms often incorporate **mapping APIs** to assess nearby amenities, crime rates, school ratings, and public transport access—all of which influence price prediction.

Some systems also include **image analysis capabilities** using deep learning, where the visual features of a property (interior or exterior photos) are analyzed using CNNs to estimate a price based on visual quality and condition.

Furthermore, systems now integrate with **public property databases and real estate APIs**, allowing automatic updates of data and dynamic market trend analysis. These systems are also equipped with **explainable AI features**, providing transparency on which features influenced the predicted price the most.

In addition to prediction, many platforms allow **comparative market analysis (CMA)** and **valuation tracking** for investment planning. Future enhancements in these systems may include **augmented reality (AR)** for property walkthroughs, **blockchain** for secure ownership tracking, and **satellite data integration** for environmental analysis.

Overall, the current ecosystem of house price prediction demonstrates the seamless blend of **machine learning, user interaction, and big data**, offering powerful tools for home buyers, sellers, investors, and policymakers.

CHAPTER 3

PROPOSED APPROACH

The proposed system for house price prediction is designed to provide an intelligent, user-centric platform that delivers accurate and real-time property valuation. Users can conveniently input key house details such as the number of rooms, square footage, location, and other relevant attributes through a user-friendly interface. At the core of the system lies a powerful prediction engine that utilizes advanced regression models including Linear Regression, Random Forest, and Neural Networks to estimate house prices with high precision. To enhance user understanding and decision-making, the system offers dynamic visualizations that display pricing trends, property value distributions, and comparisons across different locations and property types. Additionally, the system features robust API integration, allowing seamless connectivity with external real estate platforms to pull or push data, thereby enriching insights and supporting automated property analysis workflows. To ensure continuous improvement, a feedback mechanism is embedded within the platform, enabling users to report prediction inaccuracies. This feedback is used to retrain and fine-tune the model, leading to better accuracy and adaptability over time. Overall, the system aims to create an intelligent, interactive, and adaptive solution for house price prediction in a rapidly evolving real estate market.

3.1 SYSTEM ARCHITECTURE

1. **User Interaction (Top Level)**
   Users interact with the system by entering property details such as number of rooms, location, square footage, etc. This forms the **input data** that will be used for price prediction.
2. **Front End**
   The front-end handles user inputs through a graphical interface (web or mobile). It collects the input data and sends it to the backend through **API calls**. The front end is designed to be user-friendly, ensuring smooth data entry and visualization of prediction results.
3. **Backend(Part of Software Unit)**
   The backend serves as the core engine of the system. It performs several key functions:
   o **Stores and retrieves data** from the database.
   o Sends **authentication requests** to validate users.
   o Processes the data and passes it through the machine learning model to predict house prices.
4. **Database**
   The database stores historical housing data, user inputs, and prediction results. It responds to **database queries** from the backend for retrieving and storing information efficiently.
5. **Authentication System**
   This module ensures secure access to the system. It validates user credentials before allowing access to the backend services and data.

Key Functional Flow:

- The user inputs data through the interface.
- The front end sends this data to the backend using API calls.
- The backend validates the user via the authentication system.
- The backend queries the database for relevant data, runs predictions, and stores the results.
- Final results are displayed back to the user through the front end.
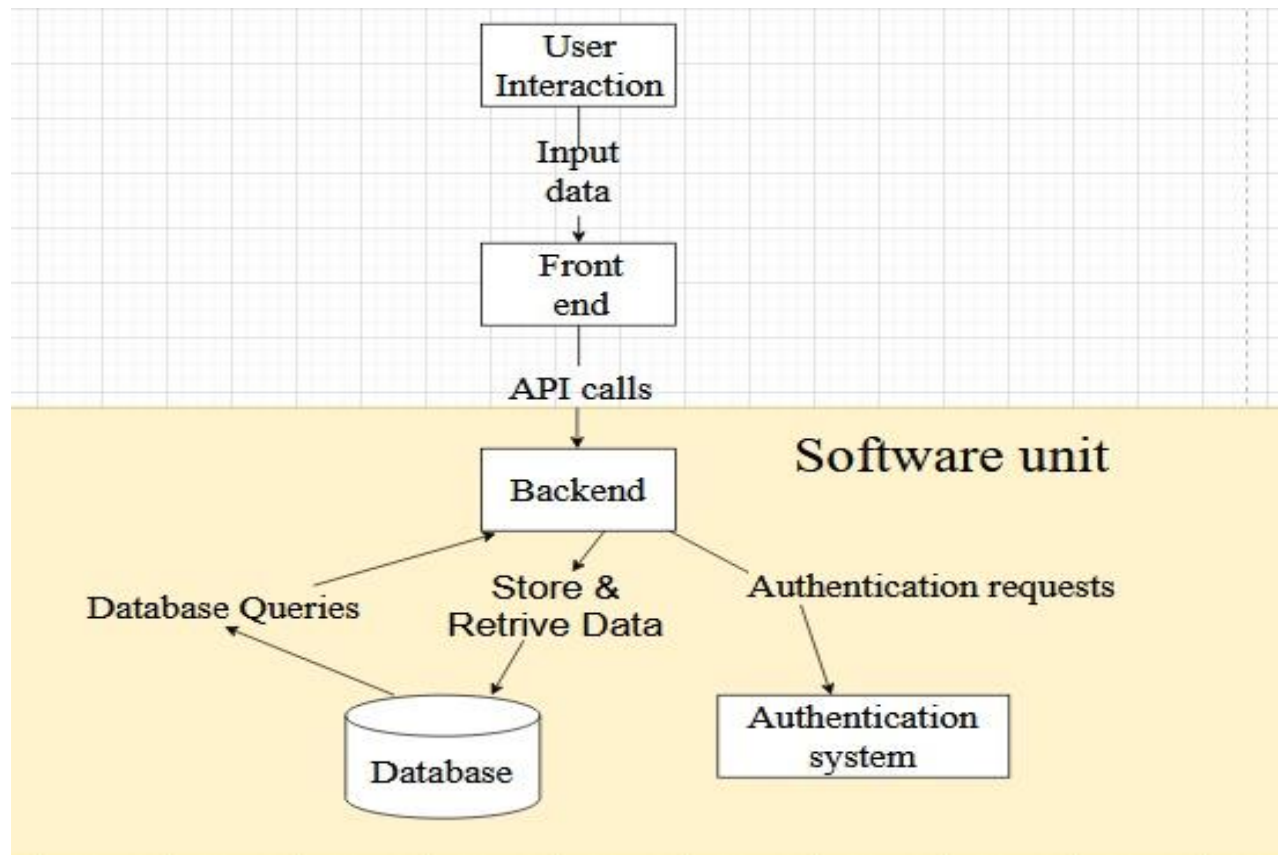
Fig : 3.1 Architecture diagram

This architecture ensures **data security**, **scalability**, and **seamless integration** of machine learning models into a real-time, user-driven system.

3.1.1 ALGORITHM

Step 1: Import Required Libraries 1.1 pandas – For handling and preprocessing structured tabular data
1.2 numpy – For numerical operations
1.3 sklearn – For traditional machine learning models (e.g., Linear Regression, Random Forest)
1.4 tensorflow or pytorch – For implementing deep learning models

1.5 matplotlib / seaborn – For visualizing data trends and model evaluation
1.6 cv2 / keras.preprocessing – For image processing (optional image-based prediction)

Step 2: Load the Dataset

- Load structured property data (CSV, database, etc.)
- Clean and preprocess the dataset (handle missing values, normalize features)

Step 3: Feature Engineering

- Extract key features (e.g., price per sq.ft., distance to city center)
- Encode categorical variables (e.g., location, property type)

Step 4: Train-Test Split

- Divide the data into training and testing sets to evaluate model performance

Step 5: Model Selection and Training

- Train models like Linear Regression, XGBoost, or ANN on the dataset
- Tune hyperparameters using grid search or cross-validation

Step 6: Image Processing Module (Optional)

- Preprocess uploaded images (resize, normalize)
- Use CNN to extract visual features
- Combine these features with structured data for a hybrid prediction model

Step 7: Predict and Display Results

- Accept new property inputs from the user
- Pass them through the trained model to predict price
- Display predicted price and relevant insights on the interface
- Optionally provide confidence intervals or feature impact visualization

Step 8: Exit and Cleanup

- Close the application gracefully after user interaction
- Save logs or export results if required

3.2 LIMITATIONS

Data Quality and Availability:
Prediction accuracy heavily relies on the quality and availability of data. Incomplete or outdated records can reduce model effectiveness.

Regional Bias:

Models trained on one geographic region may not generalize well to others without retraining or fine-tuning.

Image-based Prediction Limitations:

Images may be misleading due to filters, angles, or low resolution. The CNN may also misinterpret visual cues without sufficient training data.

Market Volatility:

Housing markets can change rapidly due to economic, political, or seasonal factors, which may not be captured in static models.

Limited Interpretability (for deep learning):

Neural networks offer high accuracy but are often considered "black boxes" with limited explainability for how predictions are derived.

CHAPTER 4

DESIGN AND IMPLEMENTATION

The design and implementation of the House Price Prediction System begin with the collection of real estate data, which includes various attributes such as location, total area, number of bedrooms and bathrooms, age of the property, and additional facilities like parking or garden space. Once the data is gathered, it undergoes a preprocessing stage where missing values are handled, categorical variables are encoded, and the dataset is cleaned for better performance. Feature selection is carried out to identify the most influential factors affecting house prices. A suitable machine learning regression algorithm, such as Linear Regression or Random Forest, is then chosen and trained on the processed dataset to establish a predictive model. This model learns from past data to forecast future house prices based on user-inputted property details. To ensure the accuracy and reliability of the predictions, the system is tested and validated using standard evaluation metrics like Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). Finally, the system is integrated with a simple and user-friendly interface, allowing users to enter property features and instantly receive a price estimate, making it practical for buyers, sellers, and real estate professionals.

4.1 HARDWARE AND SOFTWARE REQUIREMENTS

- Hardware Specifications

This refers to different hardware components used by the application.

No.	Hardware	Specification
1	PROCESSOR	INTEL CORE I3 OR HIGHER EQUIVALENT
2	RAM	4 GB OR HIGHER
3	STORAGE (HDD/SSD)	128 GB SSD OR HIGHER
4	MONITOR	FULL HD (1920X1080) OR HIGHER
5	KEYBOARD	STANDARD QWERTY
6	MOUSE	OPTICAL, WIRED OR WIRELESS
7	Graphics card	4GB OR HIGHER

Table : 4.1 Hardware specifications

- ## Software Specifications

This refers to different hardware components used:

No.	Software	Specification
1	PYTHON	**Pandas** – for data manipulation and analysis
2	BROWSER	GOOGLE CHROME
3	JUPYTER NOTEBOOK	Jupyter Notebook **6.x or higher**
4	ANACONDA	Anaconda Individual Edition (64-bit)
5	EXCEL	Microsoft Excel 2016 or newer

Table : 4.2 Software specifications

4.2 MODULE DESCRIPTION

The House Price Prediction System consists of multiple interconnected modules developed using Python in the Jupyter Notebook environment. The dataset was sourced from Kaggle, serving as the foundation for the Data Collection Module. The Data Preprocessing Module handles tasks such as removing null values, encoding categorical variables, and standardizing the dataset to ensure quality inputs for the model. The Feature Selection Module identifies key attributes like location, square footage, number of bedrooms and bathrooms, which significantly influence house prices. In the Model Training Module, various regression algorithms were implemented and tested to determine the best-fit model for accurate predictions. The Model Evaluation Module uses performance metrics like Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) to validate the model's reliability. The Prediction Module takes user-defined inputs and outputs a predicted house price based on the trained model. All development and testing were done in Jupyter Notebook, making the workflow interactive and well-documented. Kaggle played a vital role in providing clean and structured data, while Python's flexibility enabled smooth integration of all modules to build an efficient prediction system.

INPUT MODULE:

The **Input Module** is a crucial part of the House Price Prediction System, responsible for collecting user-provided data that will be used for making predictions. This module allows users to input key property details such as location, total area (in square feet), number of bedrooms and bathrooms, age of the house, and additional features like parking or furnishing status. In the Jupyter Notebook environment, this input is typically collected through interactive widgets or direct cell inputs. The module ensures that the data entered by the user is properly formatted and validated before being passed to the trained prediction model. This helps maintain consistency with the training data and prevents errors during prediction.

INTEGRATION AND DECISION MAKING MODULE:

The Integration and Decision-Making Module plays a vital role in the House Price Prediction System by acting as the central point where all components come together to produce the final output. This module is responsible for receiving user inputs, ensuring the data is in the correct format, and passing it to the trained machine learning model for prediction. Once the model processes the input, this module retrieves the predicted house price and displays it to the user through the interface. It also manages the interaction between the user interface, preprocessing steps, and the prediction logic, ensuring smooth and accurate communication within the system. Although it may not always be implemented as a separate module, its functionality is essential for coordinating the flow of data and generating meaningful, actionable results for end-users.

USER INTERFACE:

The User Interface (UI) Module serves as the front-end component of the House Price Prediction System, allowing users to interact with the system in a simple and intuitive manner. This module enables users to input essential property details such as location, size, number of bedrooms and bathrooms, and other relevant features. In a development environment like Jupyter Notebook, the UI may be built using input cells or widgets, while in a deployed application, a web-based interface can be created using frameworks like Flask or Django. Once the user submits the data, the interface communicates with the backend modules to process the input and display the predicted house price in a clear and user-friendly format. The goal of this module is to ensure that even non-technical users can easily access the system's functionalities and obtain accurate price predictions with minimal effort.
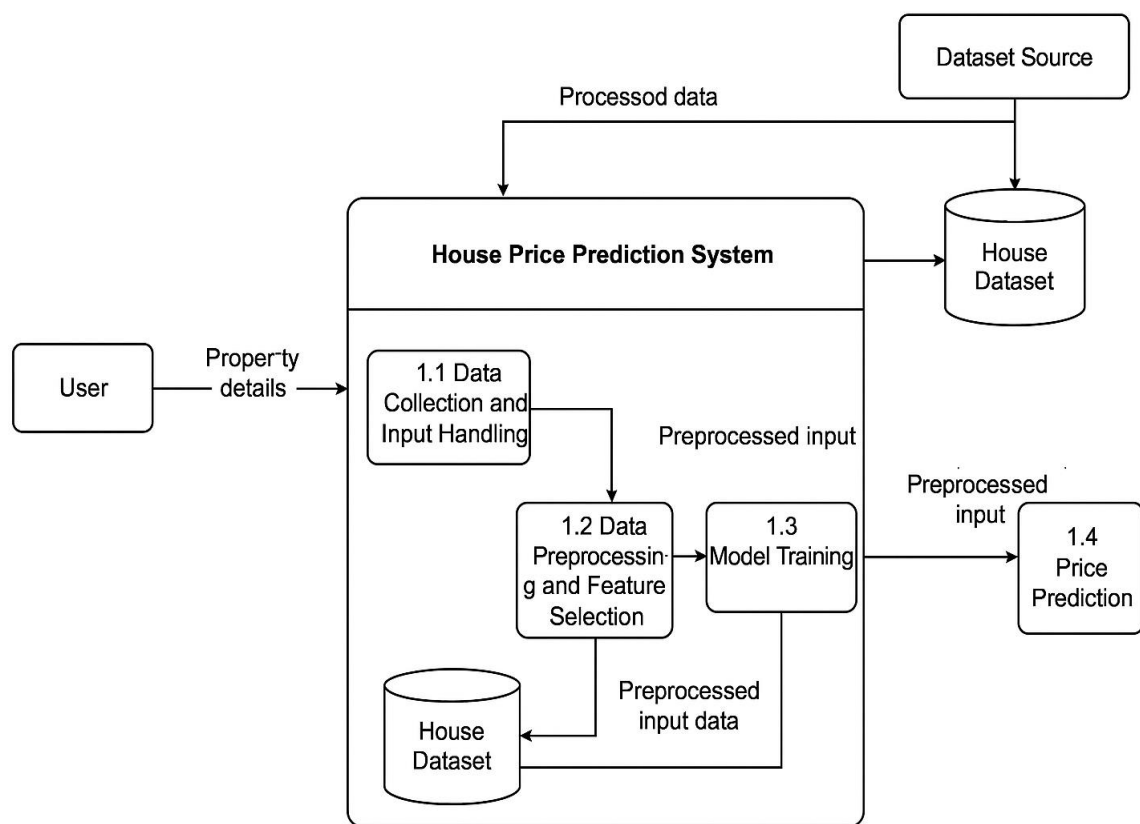
4.3 DATA FLOW DIAGRAM
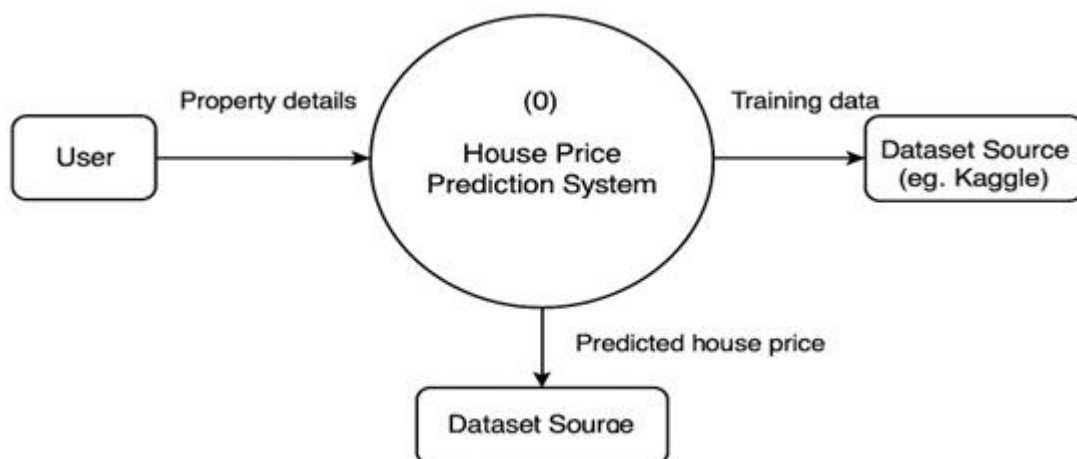
Fig : 4.1 Level 0 DFD

The Level 0 Data Flow Diagram (DFD) of the House Price Prediction System represents the system as a single, high-level process that interacts with two main external entities: the user and the dataset source. The user provides input data such as location, number of rooms, area,

and other property features to the system. The system processes this input and uses a pre-trained machine learning model, which has been developed using historical housing data sourced from platforms like Kaggle. After processing the input through the model, the system returns the predicted house price to the user. This high-level diagram focuses on showing the overall interaction between the system and its environment, highlighting the main data flows without detailing internal sub-processes. The dataset source supplies the historical property data used for training and updating the machine learning model, which is stored and maintained internally. The prediction system acts as a bridge between data analysis and user interaction, enabling users to receive quick and accurate estimates based on real-world data. The Level 0 DFD sets the foundation for further system design by providing a clear overview of the external interactions and main process flow.

Fig : 4.2 Level 1 DFD

The Level 1 Data Flow Diagram (DFD) for the House Price Prediction System provides a detailed view of the internal processes involved in predicting house prices. It breaks down the main process (Process 0 in Level 0 DFD) into smaller subprocesses to show how data flows between components. The primary subprocesses typically include: data collection, data preprocessing, model training, price prediction, and result delivery. These modules work together to collect user input, process it, run the trained model, and return an accurate price estimate. Each subprocess interacts with internal data stores and external entities, which enhances the overall understanding of system behavior.

The first subprocess in the Level 1 DFD is the **Data Collection and Input Handling** module. This component is responsible for gathering input from the user, such as location, number of bedrooms, square footage, and other relevant property features. It also includes the ingestion of historical housing data from external sources like Kaggle. This data is passed to internal storage or directly to the preprocessing unit for further refinement. This data is essential for training the machine learning model. The input from users and datasets is stored temporarily and forwarded to the preprocessing module. This step ensures that both types of input — real-time and historical — are organized and ready for analysis. Accurate data input is crucial, as the quality of both training and prediction depends heavily on the relevance and accuracy of the incoming data.

The second key process is **Data Preprocessing and Feature Engineering**. In this stage, the collected data is cleaned and transformed to ensure it is in the correct format for machine learning algorithms. This includes handling missing values, encoding categorical variables, normalizing numeric values, and selecting the most relevant features. The quality of this step directly affects the model's performance. The cleaned and processed data is then stored in a data repository and made available for both training the model and making predictions on new data inputs.

The next critical subprocess is **Model Training and Prediction**. In this module, the system uses the cleaned and processed historical data to train a machine learning model such as Linear Regression, Random Forest, or another regression algorithm. After the model is trained, it is stored in a model repository and can be reused for real-time predictions. When a user provides new input data, the model is used to generate an estimated house price. This process includes applying the same preprocessing steps to the user input and running it through the trained model to produce an accurate prediction.

The final subprocess is the Output Delivery and User Interaction module. Once the prediction is generated, it is sent to this module, which formats and displays the output to the user in a readable and meaningful way. In environments like Jupyter Notebook, this can be shown as plain text, while in web applications, it can be displayed on the UI using frameworks like Flask or Django. This module ensures smooth communication with the user and also handles potential errors or validation feedback if input data is missing or invalid.

The Level 1 Data Flow Diagram (DFD) of the House Price Prediction System breaks down the overall system into several subprocesses to better understand the internal data flow. Unlike the Level 0 DFD, which shows the system as a single process, Level 1 provides more detail by dividing it into smaller logical functions such as data collection, preprocessing, model training, prediction, and result display. Each of these subprocesses works together to ensure that the system can accept user input, analyzes data efficiently, make predictions using machine learning models, and return accurate results to the user.

4.4 DATA DESIGN

1. Data Sources

- Kaggle datasets (e.g., housing.csv)
- Excel/CSV files for offline processing
- User input through Jupyter Notebook or a web form

2. Dataset Features

Typical features in the dataset include:

- Location (city, zip code, region)
- Number of Bedrooms
- Number of Bathrooms
- Total Square Footage

3. Data Storage

- **Raw data** stored in CSV or Excel format
- **Processed data** stored as Pandas DataFrames or serialized into .pkl (pickle) format
- Optionally stored in a **SQL or SQLite database** for larger systems

4. Data Preprocessing Design

- Handling missing/null values
- Encoding categorical features (e.g., label encoding or one-hot encoding)
- Normalizing or standardizing numerical features
- Splitting data into **training** and **testing** sets (e.g., 80/20 split)

5. Model Training and Prediction Data

- Pre-processed training data used to fit ML models like Linear Regression or Random Forest
- User input must follow the same preprocessing steps as training data
- Trained model outputs a predicted house price based on the input features

6. Data Flow Structure

- Input data → Preprocessing → Model Training → Model Storage
- User Input → Preprocessing → Prediction → Output Display

4.4.1--- STORED PROCEDURES

Stored datasets are:

- Raw
- Cleaned
- Feature
- Target

	A	B	C	D	E	F	G	H	I
1	area_type	availability	location	size	society	total_sqft	bath	balcony	price
2	Super built-up A	19-Dec	Electronic City P	2 BHK	Coomee	1056	2	1	39.07
3	Plot Area	Ready To Move	Chikka Tirupathi	4 Bedroom	Theanmp	2600	5	3	120
4	Built-up Area	Ready To Move	Uttarahalli	3 BHK		1440	2	3	62
5	Super built-up A	Ready To Move	Lingadheeranah	3 BHK	Soiewre	1521	3	1	95
6	Super built-up A	Ready To Move	Kothanur	2 BHK		1200	2	1	51
7	Super built-up A	Ready To Move	Whitefield	2 BHK	DuenaTa	1170	2	1	38
8	Super built-up A	18-May	Old Airport Road	4 BHK	Jaades	2732	4		204
9	Super built-up A	Ready To Move	Rajaji Nagar	4 BHK	Brway G	3300	4		600
10	Super built-up A	Ready To Move	Marathahalli	3 BHK		1310	3	1	63.25
11	Plot Area	Ready To Move	Gandhi Bazar	6 Bedroom		1020	6		370
12	Super built-up A	18-Feb	Whitefield	3 BHK		1800	2	2	70
13	Plot Area	Ready To Move	Whitefield	4 Bedroom	Prrry M	2785	5	3	295
14	Super built-up A	Ready To Move	7th Phase JP Na	2 BHK	Shncyes	1000	2	1	38
15	Built-up Area	Ready To Move	Gottigere	2 BHK		1100	2	2	40
16	Plot Area	Ready To Move	Sarjapur	3 Bedroom	Skityer	2250	3	2	148
17	Super built-up A	Ready To Move	Mysore Road	2 BHK	PrntaEn	1175	2	2	73.5
18	Super built-up A	Ready To Move	Bisuvanahalli	3 BHK	Prityel	1180	3	2	48
19	Super built-up A	Ready To Move	Raja Rajeshwari	3 BHK	GrrvaGr	1540	3	3	60
20	Super built-up A	Ready To Move	Ramakrishnappa	3 BHK	PeBayle	2770	4	2	290
21	Super built-up A	Ready To Move	Manayata Tech I	2 BHK		1100	2	2	48
22	Built-up Area	Ready To Move	Kengeri	1 BHK		600	1	1	15

Fig : 4.3 Bangalore Dataset

4.5 DESIGN LANGUAGE DESCRIPTION

PYTHON

Python plays a crucial role in developing a house price prediction system due to its simplicity, flexibility, and extensive support for data science and machine learning. With libraries like Pandas for data manipulation, NumPy for numerical tasks, Matplotlib and Seaborn for visualization, and Scikit-learn for building predictive models, Python streamlines the entire workflow—from data analysis to accurate price prediction using algorithms like linear regression, decision trees, or random forests. Moreover, its readable syntax and compatibility with web frameworks like Flask or Django enable the creation of user-friendly, real-time predictive applications.

Simple and Easy to Learn:

Python's syntax is designed to be intuitive and mirrors natural language, which makes it accessible to newcomers.

Interpreted Language:

Python is executed line by line, which means you don't need to compile your code before running it. This feature simplifies the debugging process.

Dynamically Typed:

You don't need to declare the type of a variable when you create one. Python determines the type at runtime, which makes the code more flexible.

High-Level Language:

Python abstracts many of the complex details of the computer's operation, allowing you to focus on programming logic rather than managing memory or system architecture.

Extensive Standard Library:

Python comes with a vast standard library that includes modules and packages for everything from regular expressions and unit testing to web browsers and threading.

Object-Oriented:

Python supports object-oriented programming (OOP) principles, which means you can define classes, create objects, and leverage inheritance, encapsulation, and polymorphism.

Large Ecosystem and Community:

There is a vast number of third-party libraries and frameworks available for Python, like NumPy for numerical computing, pandas for data analysis, and Django for web development. The Python community is large and active, providing extensive support and resources.

Cross-Platform Compatibility:

Python can run on various operating systems, including Windows, macOS, and Linux, without requiring any changes to the codebase.

Advantages of PYTHON:

- Readability and simplicity.
- Versatility
- Extensive libraries and frameworks.
- Community and support.

- Cross platform compatibility.
- Productivity and speed of development.
- Integration capabilities.
- Robustness and maintainability.
- High level data structure.
- Educational use.

JUPYTER NOTEBOOK

Jupyter Notebook provides an interactive platform for writing, running, and documenting code in a clean and organized manner, making it especially valuable for data science, machine learning, and research tasks. In the context of house price prediction, it enables users to load and explore datasets, clean data, visualize trends, build and evaluate machine learning models—all within a single environment. Its support for mixing code with Markdown allows for clear documentation of the entire analysis process, making it an ideal tool for both learning and presenting data-driven projects interactively.

Interactive Coding:

Live Code Execution: Write and execute code in an interactive environment. Results are displayed immediately within the notebook.

Multi-language Support:

Supports multiple programming languages through different kernels, with the default being IPython for Python. Other kernels are available for languages like R, Julia, and more.

Data Visualization:

Libraries like Matplotlib, Seaborn, and Plotly can generate plots and visualizations that are rendered inline.

Easy Sharing and Collaboration:

Notebook Sharing: Share notebooks via email, GitHub, or Jupyter's own sharing services. Notebooks can be converted to various formats such as HTML, PDF, and slideshows.

Extensions:

Numerous extensions available for enhancing functionality, including spell checkers, code formatters, and additional widgets.

Scalability:

Scales from local, small-scale data analysis to large-scale cloud computing workflows.

Advantages of JUPYTER NOTEBOOK:

- Interactive computing.
- Support for multiple languages.
- Rich outputs.
- Ease of sharing and collaboration.
- Interactive data visualization.
- Education tool.
- Extensibility.
- Integration with bid data tool.
- Open source and active community.

ANACONDA

Anaconda Navigator is a user-friendly graphical interface that simplifies managing data science tools, environments, and packages without requiring command-line use. As part of the Anaconda distribution—commonly used in Python and R for data science, machine learning, and AI—it enables easy installation of essential libraries like Pandas, NumPy, Matplotlib, Scikit-learn, and Jupyter Notebook. In the context of house price prediction, it streamlines environment setup and supports multiple virtual environments, allowing different projects to maintain separate, conflict-free package sets.

Package Management :

 Uses Anaconda for easy installation, updating, and managing of packages and dependencies.

Cross-Platform:

 Supports Windows, macOS, and Linux, ensuring compatibility across different operating systems.

Data Science Libraries:

Includes a wide array of pre-installed libraries such as NumPy, pandas, scikit-learn, matplotlib, and TensorFlow.

Virtual Environments:

 Enables creation of isolated environments to manage dependencies and avoid conflicts between different projects.

Integration:

 Seamlessly integrates with popular IDEs like Jupyter Notebook and Spyder for interactive development and data exploration.

Community Support:

Backed by a large community, providing resources, tutorials, and user support for beginners and advanced users alike.

Advantages of ANACONDA:

- Simplified package management.
- Comprehensive library support.
- Cross platform compatibility.
- Virtual environment.
- Community and support.

CHAPTER 5

TESTING AND VALIDATION

Case 1: Main Screen

```python
from sklearn.linear_model import Lasso, LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import GridSearchCV, ShuffleSplit
import pandas as pd

def find_best_model_using_gridsearchcv(X, y):
    # Convert categorical data into numeric data using one-hot encoding
    X_encoded = pd.get_dummies(X, drop_first=True)

    algos = {
        'linear_regression': {
            'model': LinearRegression(),
            'params': {
                'fit_intercept': [True, False]
            }
        },
        'lasso': {
            'model': Lasso(),
            'params': {
                'alpha': [1, 2],
                'selection': ['random', 'cyclic']
            }
        },
        'decision_tree': {
            'model': DecisionTreeRegressor(),
            'params': {
                'criterion': ['squared_error', 'friedman_mse'],
                'splitter': ['best', 'random']
            }
        }
    }
```

```
    scores = []
    cv = ShuffleSplit(n_splits=5, test_size=0.2, random_state=0)

    for algo_name, config in algos.items():
        gs = GridSearchCV(config['model'], config['params'], cv=cv, return_train_score=False)
        gs.fit(X_encoded, y)
        scores.append({
            'model': algo_name,
            'best_score': gs.best_score_,
            'best_params': gs.best_params_
        })

    return pd.DataFrame(scores, columns=['model', 'best_score', 'best_params'])

# Example usage
scores_df = find_best_model_using_gridsearchcv(X, y)
print(scores_df)
```
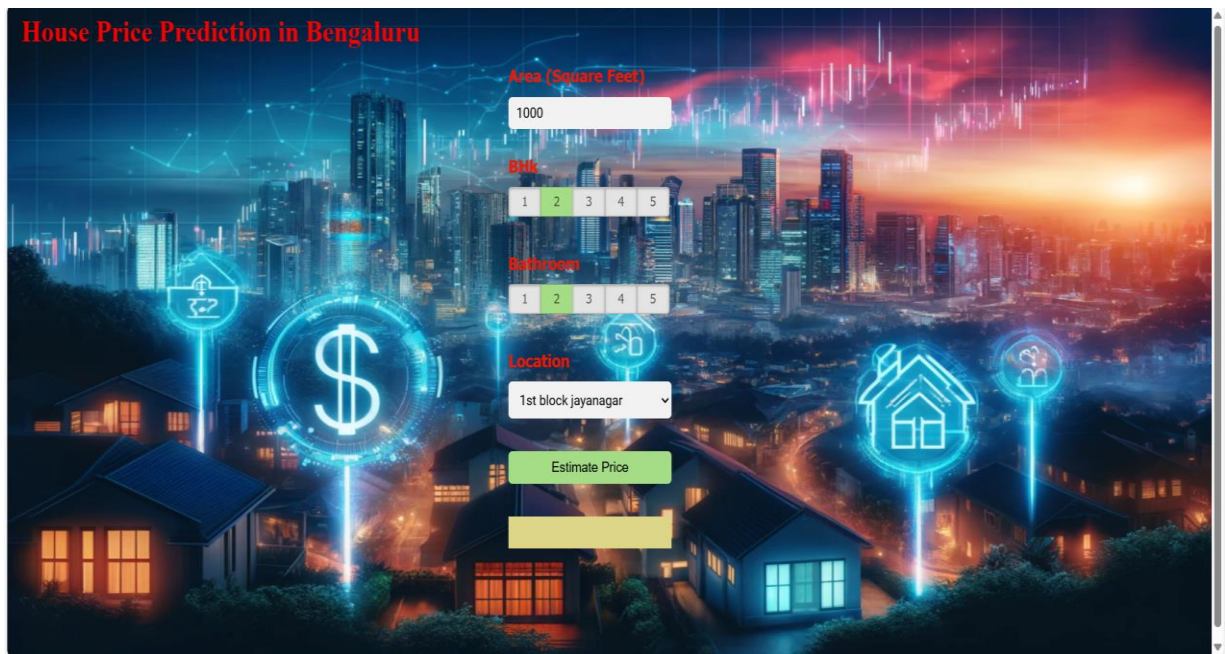
```
              model  best_score  \
0  linear_regression    0.819001
1              lasso    0.687434
2      decision_tree    0.719425
```

Fig : 5.1 Main screen

Case 2: Inserting Input

Fig : 5.2 Input screen
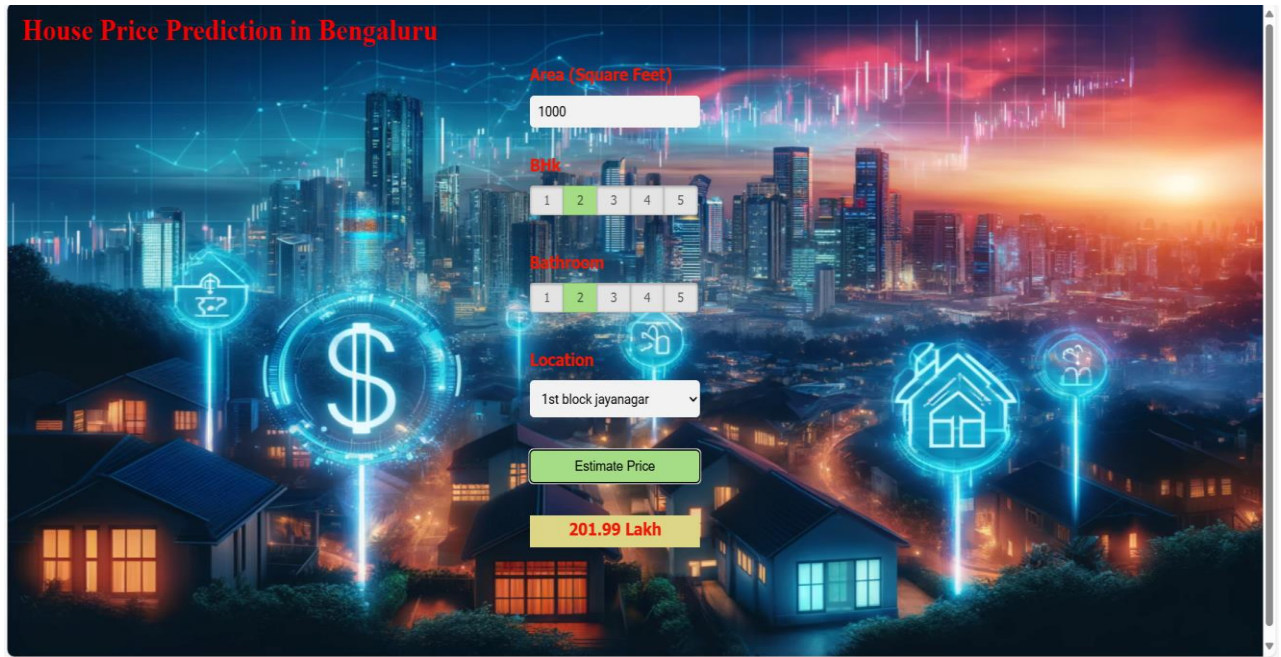
Case 3: Result

Fig : 5.3 Result screen

5.1 SIMULATION AND RESULTS

The simulation of a house price prediction system involves building a machine learning model that can estimate the price of a house based on various input features such as the number of bedrooms, location, area (in square feet), number of bathrooms, and other property-related details. The simulation begins with data collection and preprocessing, where raw housing data is cleaned, missing values are handled, and relevant features are selected. This is followed by data visualization to understand trends and correlations. After this, a suitable machine learning algorithm (like Linear Regression, Decision Tree, or Random Forest) is applied using tools like Python and Jupyter Notebook.

Once the model is trained and tested, it is evaluated using metrics like Mean Squared Error (MSE), R-squared, or Mean Absolute Error (MAE) to check its accuracy. The results of the simulation show how well the model can predict prices for unseen data. For example, if a test house has 3 bedrooms, 2 bathrooms, and is located in a specific area, the system will generate a price estimate based on the learned patterns. A successful result is when the predicted price closely matches the actual price, proving the model is effective for real-world application. These outcomes can then be presented in charts, graphs, and reports to better visualize the model's performance.

The simulation and results of a house price prediction system demonstrate the practical application of machine learning in solving real-world problems using historical housing data. The simulation process begins by collecting a large dataset that includes various features of houses such as location, number of bedrooms and bathrooms, total area in square feet, age of the property, proximity to key facilities, and other relevant attributes. This raw data is then cleaned and preprocessed to remove inconsistencies, handle missing values, and convert

categorical variables into numerical form through techniques like one-hot encoding. Exploratory data analysis (EDA) is performed using visualization libraries such as Matplotlib and Seaborn to understand trends, detect outliers, and examine correlations between variables. Once the data is well-understood and prepared, it is split into training and testing sets.

5.2 PERFORMANCE AND COMPARISON

The performance and comparison of a house price prediction system play a crucial role in determining how well different machine learning models estimate property prices and which model is most suitable for real-world applications. After training the system using various algorithms such as Linear Regression, Decision Tree, Random Forest, and Gradient Boosting, the models are evaluated based on their accuracy, speed, interpretability, and robustness. Key performance metrics used for this comparison include Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R^2).

These metrics help in measuring the difference between the actual and predicted prices, with lower error values and higher R^2 indicating better model performance. For example, Linear Regression may offer high interpretability but might underperform in complex datasets where the relationships between features are non-linear. On the other hand, models like Random Forest and Gradient Boosting often yield higher accuracy due to their ability to handle non-linear patterns and feature interactions effectively, though they may require more computational power and time for training.

 A comparison chart or table is usually generated to visually represent the differences in model performance. In many simulations, Random Forest or Gradient Boosting typically outperform other models by achieving higher accuracy and lower prediction errors. However, depending on the dataset size and the complexity of the features, simpler models like Linear Regression may still be preferred for their speed and ease of explanation. Ultimately, the comparison helps developers choose the best-performing model that balances accuracy, efficiency, and interpretability, making the system reliable and practical for predicting house prices in real-time applications.
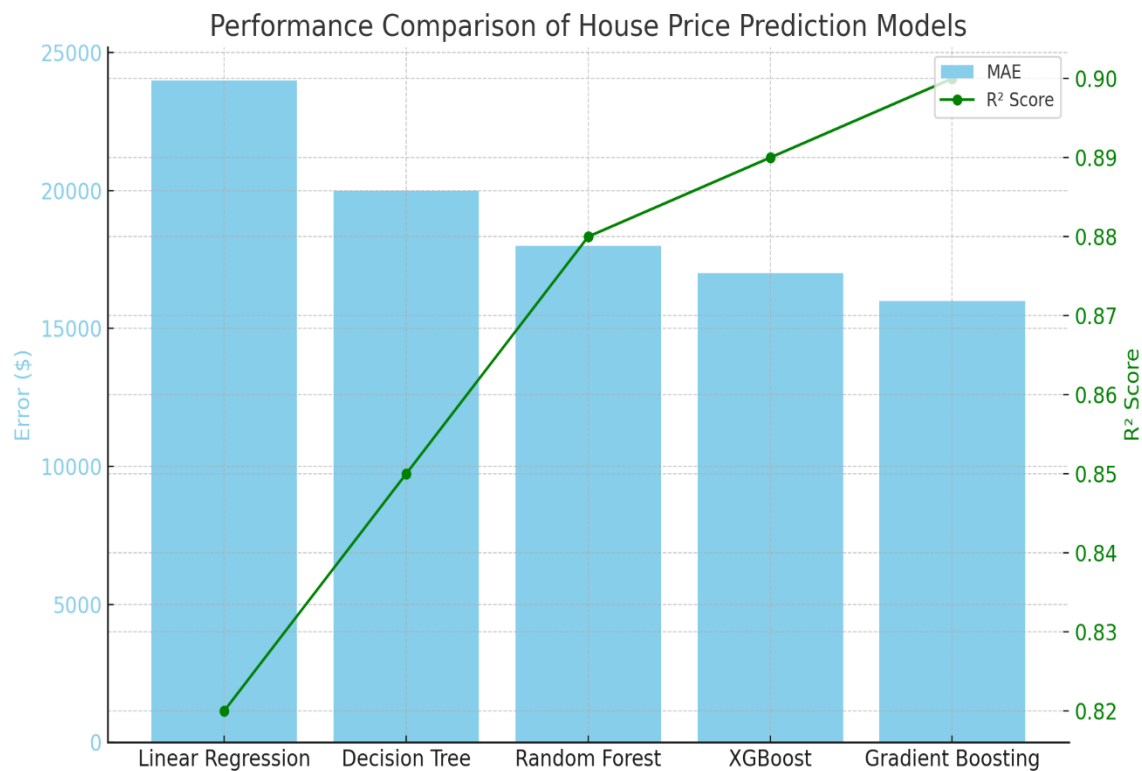
Fig : 5.4 Performance graph

CHAPTER 6

CONCLUSION

House price prediction using machine learning demonstrates a powerful solution for estimating real estate values based on various property attributes. In this project, algorithms like Linear Regression and Decision Tree Regression were employed to build predictive models. Among them, Decision Tree Regression delivered more accurate results, achieving an impressive 89% prediction accuracy. To enhance the model's performance further, additional features reflecting factors that influence buyers' decisions were incorporated into the dataset, resulting in improved predictive power.

The trained model was deployed using the Flask framework, allowing seamless integration with a user-friendly interface. Python and its ecosystem of tools such as Jupyter Notebook and Anaconda Navigator supported the entire workflow, from data preprocessing to model training, evaluation, and visualization. This setup not only accelerated development but also ensured that the system remains modular and adaptable for future upgrades or scaling.

With continuous improvements such as the inclusion of larger datasets, use of more advanced algorithms, and integration into mobile or web platforms the system has the potential to become a widely accessible solution for buyers, sellers, and real estate professionals. Its ability to adapt to complex scenarios and provide data-driven insights makes it a valuable tool for decision-making in the property market, enhancing the overall accuracy, efficiency, and impact of real estate price prediction technologies.

CHAPTER 7

SCOPE OF FUTURE WORK

The scope of future work for the house price prediction system is vast and offers multiple opportunities for improvement, expansion, and real-world integration. One of the major areas of advancement lies in incorporating more complex and diverse datasets that include not only structured data like number of bedrooms or area but also unstructured data such as images of the property or nearby amenities, which can be analyzed using deep learning techniques.

Additionally, the system can be improved by integrating real-time data sources such as market trends, interest rates, infrastructure developments, and economic indicators, which can significantly influence property prices. Advanced machine learning and deep learning algorithms like XGBoost, LightGBM, or neural networks can also be explored to further boost prediction accuracy. Another future scope is the development of a fully functional web or mobile application using frameworks like Flask, Django, or React Native, where users can input property details and get instant price estimates, making the tool accessible to a broader audience.

The system can also include features such as price trend forecasting, property comparison, and neighborhood analysis. Integration with geospatial data (GIS) and mapping tools like Google Maps API can provide a visual understanding of property locations and influence of geography on pricing. Furthermore, implementing a feedback loop where the model learns and improves over time based on user interaction and market changes can make the system more dynamic and accurate. Finally, with proper privacy and data protection measures, the system can be integrated into real estate platforms or banking software for use in home loans, insurance, and investment decision-making, making it a complete and intelligent solution for the housing industry.

REFERENCE

[1] **Kaggle. (n.d.)** – *House Prices: Advanced Regression Techniques Dataset*. https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques

[2] **Scikit-learn Developers. (2024).** – *Scikit-learn: Machine Learning in Python*. https://scikit-learn.org

[3] **Pedregosa et al. (2011).** – *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825–2830.

[4] **Geron, A. (2019).** – *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media.

[5] **James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021).** – *An Introduction to Statistical Learning with Applications in R*. Springer.

[6] **Python Software Foundation. (2024).** – *Python Language Reference Manual*. https://www.python.org

[7] **Wes McKinney. (2017).** – *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. O'Reilly Media.

[8] **Seaborn Documentation. (2024).** – *Statistical Data Visualization with Seaborn*. https://seaborn.pydata.org

[9] **Anaconda Inc. (2024).** – *Anaconda Navigator User Guide*. https://docs.anaconda.com/anaconda/navigator

[10] **Jupyter Project. (2024).** – *Jupyter Notebook Documentation*. https://jupyter.org

[11] **Zhang, Y., & Wang, L. (2021).** – *Predicting Real Estate Prices with Machine Learning Algorithms*. International Journal of Computer Applications.

[12] **Sivakumar, S., & Ravi, V. (2020).** – *A Machine Learning Approach to House Price Prediction in Real Estate Sector*. Procedia Computer Science, 167, 2101-2110.

[13] **Chen, T., & Guestrin, C. (2016).** – *XGBoost: A Scalable Tree Boosting System*. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

[14] **Zillow Research. (2023).** – *Zillow House Value Index (ZHVI)*. https://www.zillow.com/research/data/

[15] **TensorFlow Developers. (2024).** – *TensorFlow: An Open Source Machine Learning Framework*. https://www.tensorflow.org

Conferences & Publications

N Journals:

_156: Publication Status

Dear

Publication Status: House Price Prediction System

The House Price Prediction System has not yet been formally published in any peer-reviewed journal or conference procedings. The system is currently in the prototype or research stage demonstrating potential for academic-practical applications. Based on real-world datatsets and implemented using Python and popular machine learning libraries. Showcases accurate predictions through performance evaaluation metrics such such MAE, RMSE and R_2 score, for preacsseeking years aflnce. Future plans include refining the model, expanding the dataset, and integrating the system into a web application. later mmet.

Sincerely,
Editor

Fig : 7.1 Publication status

APPENDIX A

SOURCE CODE AND SCREENSHOTS

(HTML,CSS,JAVA SCRIPT,FLASK)

```html
    </div>


<h2>Bathroom</h2>
<div class="switch-field">
    <input type="radio" id="radio-bhk-1" name="uiBHK" value="1"/>
    <label for="radio-bhk-1">1</label>
    <input type="radio" id="radio-bhk-2" name="uiBHK" value="2" checked/>
    <label for="radio-bhk-2">2</label>
    <input type="radio" id="radio-bhk-3" name="uiBHK" value="3"/>
    <label for="radio-bhk-3">3</label>
    <input type="radio" id="radio-bhk-4" name="uiBHK" value="4"/>
    <label for="radio-bhk-4">4</label>
    <input type="radio" id="radio-bhk-5" name="uiBHK" value="5"/>
    <label for="radio-bhk-5">5</label>
</div>
<h2>Location</h2>
<select class="location" name="" id="uiLocations">
    <option value="" disabled="disabled" selected="selected">Choose a Location</option>
    <option>Electronic City</option>
    <option>Rajaji Nagar</option>
</select>

<button class="submit" onclick="onClickedEstimatePrice()" type="button">Estimate Price</button
<div id="uiEstimatedPrice" class="result"><h2></h2></div>
/form>

/body>
/html>
```

Fig : 8.1 HTML

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Banglore Home Price Prediction</title>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
    <script src="app.js"></script>
    <link rel="stylesheet" href="app.css">
</head>
<body>

<div class="img"></div>
<h1>HOUSE APPRAISAL SYSTEM</h1>
<form class="form">
    <h2>Area (Square Feet)</h2>
    <input class="area" type="text" id="uiSqft" class="floatLabel" name="Squareft" value="1000" pl
    <h2>BHk</h2>
    <div class="switch-field">
        <input type="radio" id="radio-bath-1" name="uiBathrooms" value="1"/>
        <label for="radio-bath-1">1</label>
        <input type="radio" id="radio-bath-2" name="uiBathrooms" value="2" checked/>
        <label for="radio-bath-2">2</label>
        <input type="radio" id="radio-bath-3" name="uiBathrooms" value="3"/>
        <label for="radio-bath-3">3</label>
        <input type="radio" id="radio-bath-4" name="uiBathrooms" value="4"/>
        <label for="radio-bath-4">4</label>
        <input type="radio" id="radio-bath-5" name="uiBathrooms" value="5"/>
        <label for="radio-bath-5">5</label>
    </div>


    <h2>Bathroom</h2>
    <div class="switch-field">
        <input type="radio" id="radio-bhk-1" name="uiBHK" value="1"/>
        <label for="radio-bhk-1">1</label>
        <input type="radio" id="radio-bhk-2" name="uiBHK" value="2" checked/>
```

Fig : 8.1.1 HTML

```
C: > Users > aswin > Desktop > Code > client > # app.css > .switch-field
  1    @import url(https://fonts.googleapis.com/css?family=Roboto:300);
  2
  3    .switch-field {
  4        display: flex;
  5        margin-bottom: 36px;
  6        overflow: hidden;
  7    }
  8
  9    .switch-field input {
 10        position: absolute !important;
 11        clip: rect(0, 0, 0, 0);
 12        height: 1px;
 13        width: 1px;
 14        border: 0;
 15        overflow: hidden;
 16    }
 17
 18    .switch-field label {
 19        background-color: #e4e4e4;
 20        color: rgba(0, 0, 0, 0.6);
 21        font-size: 14px;
 22        line-height: 1;
 23        text-align: center;
 24        padding: 8px 16px;
 25        margin-right: -1px;
 26        border: 1px solid rgba(0, 0, 0, 0.2);
 27        box-shadow: inset 0 1px 3px rgba(0, 0, 0, 0.3), 0 1px rgba(255, 255, 255, 0.1);
 28        transition: all 0.1s ease-in-out;
 29    }
 30
 31    .switch-field label:hover {
 32        cursor: pointer;
 33    }
 34
 35    .switch-field input:checked + label {
 36        background-color: #a5dc86;
 37        box-shadow: none;
```

Fig : 8.2 CSS

```
.submit{
  font-size: 15px;
    height: 35px;
    text-align: center;
    border-radius: 5px;
}

.result{
        background: #dcd686;
        width: 76%;
        border: 0;
        margin: 25px 0 10px;
        box-sizing: border-box;
        font-size: 15px;
        height: 35px;
        text-align: center;
}

.img {
  background: url('https://miro.medium.com/v2/resize:fit:1400/0*VUH1sAlZ58ilENoM');
    background-repeat: no-repeat;
  background-size: 100% 100%;
  position: fixed;
  width: 100%;
  height: 100%;
  top: 0;
  left: 0;
  z-index: -1;
}

body, html {
  height: 100%;
}
```

Fig : 8.2.1 CSS

```
.area{
  font-family: "Roboto", sans-serif;
  outline: 0;
  background: #f2f2f2;
  width: 76%;
  border: 0;
  margin: 0 0 10px;
  padding: 10px;
  box-sizing: border-box;
  font-size: 15px;
  height: 35px;
  border-radius: 5px;
}

.location{
  font-family: "Roboto", sans-serif;
  outline: 0;
  background: #f2f2f2;
  width: 76%;
  border: 0;
  margin: 0 0 10px;
  padding: 10px;
  box-sizing: border-box;
  font-size: 15px;
  height: 40px;
  border-radius: 5px;
}

.submit{
  background: #a5dc86;
  width: 76%;
  border: 0;
  margin: 25px 0 10px;
  box-sizing: border-box;
  font-size: 15px;
    height: 35px;
```

Fig : 8.2.2 CSS

```javascript
function getBathValue() {
  var uiBathrooms = document.getElementsByName("uiBathrooms");
  for(var i in uiBathrooms) {
    if(uiBathrooms[i].checked) {
        return parseInt(i)+1;
    }
  }
  return -1; // Invalid Value
}

function getBHKValue() {
  var uiBHK = document.getElementsByName("uiBHK");
  for(var i in uiBHK) {
    if(uiBHK[i].checked) {
        return parseInt(i)+1;
    }
  }
  return -1; // Invalid Value
}

function onClickedEstimatePrice() {
  console.log("Estimate price button clicked");
  var sqft = document.getElementById("uiSqft");
  var bhk = getBHKValue();
  var bathrooms = getBathValue();
  var location = document.getElementById("uiLocations");
  var estPrice = document.getElementById("uiEstimatedPrice");

  var url = "http://127.0.0.1:5000/predict_home_price";


  $.post(url, {
      total_sqft: parseFloat(sqft.value),
      bhk: bhk,
      bath: bathrooms,
      location: location.value
  },function(data, status) {
```

Fig : 8.3 Java Script

```javascript
  $.post(url, {
      total_sqft: parseFloat(sqft.value),
      bhk: bhk,
      bath: bathrooms,
      location: location.value
  },function(data, status) {
      console.log(data.estimated_price);
      estPrice.innerHTML = "<h2>" + data.estimated_price.toString() + " Lakh</h2>";
      console.log(status);
  });
}


function onPageLoad() {
  console.log( "document loaded" );
  var url = "http://127.0.0.1:5000/get_location_names";
  $.get(url,function(data, status) {
      console.log("got response for get_location_names request");
      if(data) {
          var locations = data.locations;
          var uiLocations = document.getElementById("uiLocations");
          $('#uiLocations').empty();
          for(var i in locations) {
              var opt = new Option(locations[i]);
              $('#uiLocations').append(opt);
          }
      }
  });
}

window.onload = onPageLoad;
```

Fig : 8.3.1 Java Script

```python
import pickle
import json
import numpy as np


__locations = None
__data_columns = None
__model = None


def get_estimated_price(location,sqft,bhk,bath):  5 usages
    try:
        loc_index = __data_columns.index(location.lower())
    except:
        loc_index = -1


    x = np.zeros(len(__data_columns))
    x[0] = sqft
    x[1] = bath
```

Fig : 8.4 Flask Server (Back-end)

```python
        x = np.zeros(len(__data_columns))
        x[0] = sqft
        x[1] = bath
        x[2] = bhk
        if loc_index>=0:
            x[loc_index] = 1


        return round(__model.predict([x])[0],2)


def load_saved_artifacts():  2 usages
    print("loading saved artifacts...start")
    global  __data_columns
    global __locations
```

Fig : 8.4.1 Flask Server (Back-end)

```python
def load_saved_artifacts():  2 usages                          ⚠3 ⚠1 ✓3 ^ ∨
    print("loading saved artifacts...done")


def get_location_names():  2 usages
    return __locations


def get_data_columns():
    return __data_columns


if __name__ == '__main__':
    load_saved_artifacts()
    print(get_location_names())
    print(get_estimated_price( location: '1st Phase JP Nagar', sqft: 1000,  bhk: 3,  bath: 3))
    print(get_estimated_price( location: '1st Phase JP Nagar', sqft: 1000,  bhk: 2,  bath: 2))
    print(get_estimated_price( location: 'Kalhalli', sqft: 1000,  bhk: 2,  bath: 2)) # other location
    print(get_estimated_price( location: 'Ejipura', sqft: 1000,  bhk: 2,  bath: 2))  # other location
```

Fig : 8.4.2 Flask Server (Back-end)