VIT BHOPAL

House Price Predictor

Project Documentation & Report

Submitted by: BhaweshKumar Gautam **Reg No:** 25BSA10052 **Language:** Python

Introduction

- **Concept:** Introduce the project as a Machine Learning application designed to assist in real estate valuation.
- **Scope:** The system analyzes historical housing data (Square footage, bedrooms, age, location) to predict the market price of a house.
- **Relevance:** Explain that manual property valuation is time-consuming and prone to bias; this tool automates the process using statistical algorithms.

Problem Statement

- **The Issue:** "Determining the fair market value of a property is complex due to the variety of influencing factors (location, age, size)."
- **The Goal:** To build a software solution that inputs specific house features and outputs a reliable price estimate using a Linear Regression model.

Functional Requirements

- **Data Ingestion:** The system must load a dataset containing housing features.
- **Data Cleaning:** The system must identify and handle missing values (specifically in 'Square Footage').

- **Model Training:** The system must use 80% of the data to train a prediction model.
- **Prediction Interface:** The system must accept user inputs (Square footage, Beds, Age, Location) via a Command Line Interface (CLI). **Visualization:** The system must
- generate a scatter plot comparing actual vs.
  predicted prices.

Non-functional Requirements

- **Accuracy:** The model should achieve an R2 score close to 1.0 (high correlation).
- **Performance:** Predictions should be generated instantly (<1 second) after user input.
- **Usability:** The CLI should handle invalid inputs gracefully (e.g., typing text instead of numbers).

System Architecture

- **Description:** A monolithic script architecture.
- **Components:**
  - **Input Module:** pandas (Data Frames) and input() function.
  - **Processing Module:** sklearn (Linear Regression, Simple Imputer).
  - **Output Module:** matplotlib (Graphing) and Console Print.

Design Diagrams

- **Use Case Diagram:**
  - **Actor:** User.
  - **Use Cases:** "Load Data", "Train Model", "Input House Details", "View Predicted Price", "View Accuracy Graph".
- **Workflow Diagram:** * *Draw this flow:* Raw Data → Impute Missing Values (Clean) → One-Hot Encode (Location) → Split Data → Linear Regression Fit → User Input → Prediction.
- **Sequence Diagram:**
  - User starts script → System initializes data → System cleans data → System trains model → System requests input → User types details → System calculates price → Display result.
- **ER Diagram:**

- *Note:* Since you aren't using a database, you can skip this or show a single "Entity" representing your CSV/Data Frame structure with attributes: ID, Sq Ft, Bedrooms, Age, Location, Price.

Design Decisions & Rationale

- **Algorithm Choice:** Selected **Linear Regression** because the relationship between house size and price is generally linear (as size goes up, price goes up). It is computationally efficient and easy to interpret.
- **Handling Missing Data:** Used **Mean Imputation** (Simple Imputer) for 'Square Footage' because deleting rows with missing data would reduce an already small dataset.
- **Categorical Encoding:** Used **One-Hot Encoding** (get_dummies) for 'Location'. Machine learning models need numbers, not words; this converts "Rural" into binary columns (0s and 1s).

Implementation Details

- **Language:** Python 3.x.
- **Key Libraries:**
  - Pandas: For Data Frame manipulation.
  - Scikit-learn: For the Linear Regression algorithm and train_test_split.
  - Matplotlib: For visualizing the regression line.
- **Key Snippet:** Highlight the prediction logic:

$$Price = (Coef1 \times SqFt)+(Coef2 \times Beds)+(Coef3 \times Age)+Intercept$$

Screenshots / Results

- **Console Output:** Take a screenshot of the "Model Evaluation" section of your terminal showing the **Mean Squared Error (MSE)** and **R2 Score**.
- **Interactive Test:** Take a screenshot of the "Live Prediction" section where you enter a dummy house (e.g., 2000 sqft, 3 beds) and show the estimated price.
- **Graph:** Include the plot generated by plt.show().

Testing Approach

- **Train/Test Split:** Explain that you split the data 80/20. The model never saw the "Test" data during training, ensuring the evaluation is fair.
- **Manual Testing:** Inputting known values (e.g., extremely high age) to see if the price drops logic holds true.
- **Exception Handling:** Testing what happens if a user enters "ABC" instead of a number (handled by your try...except block).

Challenges Faced

- **Small Dataset:** The synthetic dataset is small (20 rows), which can lead to overfitting or variance in results.
- **Categorical Data:** Converting the text-based "Location" column into a format the math equation could understand was a hurdle solved by One-Hot Encoding.

Learnings & Key Takeaways

- Understanding the full ML pipeline: Preprocessing → Training → Evaluation.
- The importance of data cleaning (handling the NaN values you artificially inserted).
- How to interpret coefficients (e.g., how much 1 extra bedroom adds to the price).

Future Enhancements

- **Real Data:** Connect to a real estate API (like Zillow or Kaggle datasets) instead of using a dictionary.
- **GUI:** Build a web interface using **Stream lit** or **Flask** so users don't have to use the console.
- **Better Models:** Implement Random Forest or Gradient Boosting for higher accuracy on complex data.

References

- Scikit-Learn Documentation (Linear Regression).
- Pandas Documentation (Data frames and Dummies).
- Matplotlib Documentation (Scatter plots).