<HOUSE PRICE PREDICTION>

A

Mini Project Report

Submitted in partial fulfilment of the Requirements for the award of the Degree of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

By

<BUDDI HARSHASRI><1602-23-737-085>

<SUHASINI REDDY><1602-23-737-312>

Department of Information Technology Vasavi

College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University and Approved by AICTE)

Ibrahimbagh, Hyderabad-31

2024

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University and Approved by AICTE)

Hyderabad-500 031

Department of Information Technology

DECLARATION BY THE CANDIDATE

We, **<BUDDI HARSHASRI>, <SUHASINI REDDY>**,
bearing hall ticket numbers, **<1602-23-737-085>, <1602-23-737-312>,** hereby declare that the project report entitled **<"HOUSE PRICE PREDICTION">** is submitted in partial fulfilment of the requirement for the award of the degree of **Bachelor of Engineering** in **Information Technology**

This is a record of bonafide work carried out by us and the results embodied in this project report havenot been submitted to any other university or institute for the award of any other degree or diploma.

<BUDDI HARSHASRI>

<1602-23-737-085.>

<SUHASINI REDDY>

<1602-23-737-312>

(Faculty In-Charge) (External Examiner) (Head,Dept of IT)

ACKNOWLEDGMENT

We extend our sincere thanks to Dr. S. V. Ramana, Principal, Vasavi College of Engineering, for his encouragement.

We express our sincere gratitude to Dr. K. Ram Mohan Rao, Professor & Head, Department of Information Technology, Vasavi College of Engineering, for introducing the Mini-Project module in our curriculum, and for his suggestions, motivation, and co-operation for the successful completion of our Mini Project.

We also want to thank and convey our gratitude towards our mini project coordinators <DIVYA LINGANENI> and <T ANJANI DEVI> for guiding us in understanding the process of project development & giving us timely suggestions at every phase.

We would also like to sincerely thank the project reviewers for their valuable input and suggestions.

Abstract

This project, *Real Estate Price Prediction Website*, focuses on building a comprehensive web application to predict property prices based on user inputs such as location, area, bedrooms, and bathrooms. Using a dataset of Bangalore home prices sourced from Kaggle, the project employs data science techniques and machine learning algorithms to create an accurate prediction model.
The development process is divided into three phases:

1. **Model Building:**

   Utilizing Python libraries like Numpy, Pandas, Matplotlib, and Sklearn, the project includes data cleaning, outlier detection, feature engineering, and model optimization using linear regression. Advanced methods like dimensionality reduction and cross - validation ensure high model performance.

2. **Backend Development:**

   A Flask server is implemented to serve HTTP requests, connecting the machine learning model to the user interface.

3. **Frontend Development:**

   The user interface, created with HTML, CSS, and JavaScript, allows users to input property details and view the predicted price.

This project showcases the integration of machine learning with web development and highlights practical applications of data science in real estate. The outcome is a functional and user-friendly platform that demonstrates predictive analytics

Table of Contents

Abstract & Introduction

Abstract

The *Real Estate Price Prediction Website* is designed to provide users with a reliable tool to estimate property prices based on key inputs such as location, square footage, and the number of bedrooms and bathrooms. The project integrates machine learning techniques with web development to create a seamless and user-friendly platform.

The project is structured into three main components:

1. **Model Building:**

   A machine learning model, trained using a dataset of Bangalore home prices from Kaggle, leverages linear to ensure accuracy.

2. **Backend Development:**

   A Python Flask server connects the machine learning model with the web interface, enabling efficient data exchange and predictions.

3. **Frontend Development:**

   An interactive user interface developed with HTML, CSS, and JavaScript allows users to input property details and view the predicted prices.

This project not only addresses the growing demand for data-driven tools in the real estate sector but also demonstrates the practical application of data science concepts in real-world scenarios.

Introduction

a. Information About the Project Domain

The real estate sector is a cornerstone of economic growth, and accurate property valuation is critical for buyers, sellers, and investors. Traditional property valuation methods often rely on subjective analysis, leading to inconsistent and unreliable pricing. This project addresses this challenge by using data science to provide a more scientific and accurate approach to price prediction.
The domain of real estate prediction is inherently data-driven, with factors like location, property size, and amenities influencing prices. Machine learning techniques enable the extraction of patterns and trends from historical data, leading to more precise predictions. This project, specifically focused on Bangalore's real estate market, demonstrates the application of predictive analytics to address these challenges.

b. Features Specific to the Project

The project is designed with the following prioritized features:

1. **Accurate Price Prediction:**
    - Using a robust linear regression model to provide precise predictions.
    - Accounting for key variables like location, square footage, number of bedrooms, and bathrooms.

2. **User-Friendly Interface:**
    - A simple and intuitive UI developed with HTML, CSS, and JavaScript.
    - Allows users to input property details and view predictions seamlessly.

3. **Backend Integration with Flask:**
    - Efficiently connects the frontend with the machine learning model through API endpoints.
    - Handles HTTP requests for real-time predictions.

4. **Data-Driven Approach:**
    - Comprehensive data cleaning and preprocessing steps to ensure the model's reliability.
    - Implementation of advanced techniques like outlier detection, dimensionality reduction, and feature engineering.

5. **Scalability and Extendibility:**
    - Modular design to enable future extensions, such as including more location-specific variables or integrating with other datasets.

This project not only provides a functional solution for property price prediction but also lays the groundwork for future enhancements and applications in the real estate domain.

Technology

a. Software Requirements

The project relies on the following software tools and technologies:

1. **Python**

    - Primary language for data preprocessing, machine learning model development, and backend integration.

2. **Libraries and Frameworks**
    - **Numpy** and **Pandas:**

      For data cleaning, preprocessing, and manipulation.

    - **Matplotlib:**

      For data visualization and exploratory data analysis.

    - **Sklearn (Scikit-learn):**

      For building the machine learning model, performing and implementing linear regression.

    - **Flask:**

      To create a lightweight web server for handling HTTP requests and serving predictions.

3. **Integrated Development Environments (IDEs):**

    - **Jupyter Notebook:**

      For exploratory data analysis and model building.

    - **Visual Studio Code**

      For Flask server development and integrating the backend with the frontend.

4. **Frontend Technologies:**

    - **HTML:**
      To structure the web pages.

    - **CSS:**
      For styling the user interface.

- **JavaScript:**

  To handle dynamic interactions and send requests to the Flask server.

5. **Other Tools:**

   - **Kaggle:**

     To source the Bangalore home prices dataset.

   - **Git/GitHub:**

   - For version control and repository management.

b. Hardware Requirements

To develop and run the project, the following hardware specifications are recommended:

1. **Processor:**

   - Intel Core i5/i7 or AMD Ryzen 5/7 (or equivalent) for faster computations and smoother execution of ML workflows.

2. **RAM:**

   - Minimum 8GB (16GB recommended) to handle data processing and model training efficiently.

3. **Storage:**

   - At least 256GB SSD (solid-state drive) for faster read/write speeds, or 1TB HDD for larger storage needs.

4. **Graphics:**

   - Integrated graphics are sufficient, but a discrete GPU (e.g., NVIDIA GTX series) can accelerate model training if needed for larger datasets.

5. **Operating System:**

   - 64-bit OS: Windows 10/11, macOS, or Linux (Ubuntu 20.04+).

6. **Internet Access:**

   Reliable internet for downloading libraries, datasets, and accessing external resources.

These specifications ensure a smooth development and deployment process for the real estate price prediction website.

Proposed Work

A . Design

I . High-Level Design Diagram

The high-level architecture of the *Real Estate Price Prediction Website* integrates three main components:

1. **Frontend (User Interface):**
    - Designed with HTML, CSS, and JavaScript to enable user interaction.
    - Collects user inputs such as location, square footage, number of bedrooms, and bathrooms.
2. **Backend (Flask Server):**
    - Processes HTTP requests from the frontend and interacts with the machine learning model.
    - Returns the predicted price to the frontend.
3. **Machine Learning Model:**
    - Trained using the Bangalore home prices dataset.
    - Implements linear regression for prediction and for higher accuracy.

Diagram Description:

- **Frontend Layer:**

  Displays input form, sends data to Flask API, and receives predictions to display.

- **Backend Layer:**

  Processes input data, loads the machine learning model, and sends the response back to the frontend.

- **Machine Learning Layer:**

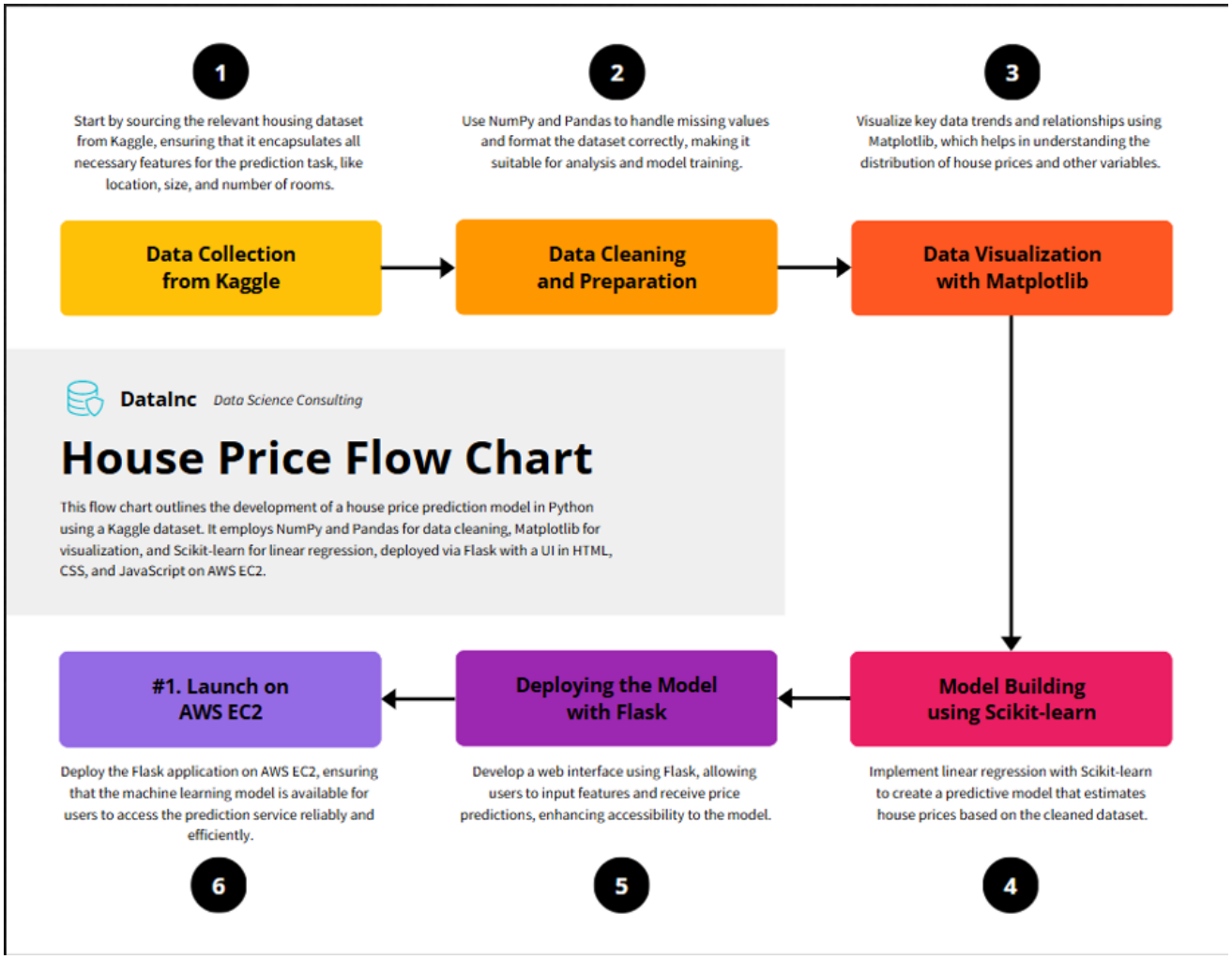
  Contains the trained model and handles data transformations for predictions.

Workflow Diagram

The workflow diagram illustrates the step-by-step flow of data within the system:

1. **User Input:**
    - The user enters details like location, square footage, bedrooms, and bathrooms on the website form.
2. **Frontend Request:**
    - The frontend sends the input data as a POST request to the Flask server.
3. **Flask Backend:**
    - Validates the received data.
    - Passes the data to the machine learning model for prediction.
4. **Machine Learning Model:**
    - Loads the trained linear regression model.
    - Processes the input data and calculates the predicted price.
5. **Backend Response:**
    - The Flask server receives the prediction from the model.
    - Sends the predicted price as a JSON response to the frontend.
6. **Result Display:**
    - The frontend displays the predicted price to the user in a user-friendly format.

DIAGRAMATIC REPRESENTATION:



Implementation

I . Module-wise Code for the Entire Project

The project consists of the following main modules:

1. Machine Learning Model (Model Training and Saving)

CODE :

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
import matplotlib
matplotlib.rcParams["figure.figsize"] = (20,10)
df1 = pd.read_csv("C:/Users/harsh/OneDrive/Desktop/miniproject/bengaluru_house_prices.csv")
df1.head()
df1.shape
df1['area_type'].value_counts()
df2 = df1.drop(['area_type','society','balcony','availability'],axis = 'columns')
df2.head()
df2.isnull().sum()
df3 = df2.dropna()
df3.isnull().sum()
df3.shape
df3['size'].unique()
df3['bhk'] = df3['size'].apply(lambda x: int(x.split(" ")[0]))
df3.head()
df3['bhk'].unique()
df3[df3['bhk']>20]
df3.total_sqft.unique()
def is_float(x):
    try:
        float(x)
    except:
        return False
    return True
df3[~df3['total_sqft'].apply(is_float)].head(10)
def convert(x):
    token = x.split('-')
    if len(token) == 2:
        return (float(token[0])+float(token[1]))/2
    try:
        return float(x)
    except:
        return None
convert('2100 - 2850')
```

```
df4 = df3.copy()
df4['total_sqft'] = df4['total_sqft'].apply(convert)
df4.head()
df4.loc[30]
df4.head(3)
df5 = df4.copy()
df5['price_per_sqft'] = df5['price']*100000/df5['total_sqft']
df5.head()
len(df5.location.unique())
df5.location = df5.location.apply(lambda x: x.strip())
location_stats = df5.groupby('location')['location'].agg('count').sort_values(ascending = False)
location_stats
df5['location'].value_counts()
len(location_stats[location_stats<=10])
location_stats_lessthan10 = location_stats[location_stats<=10]
location_stats_lessthan10
df5.location = df5.location.apply(lambda x : 'other' if x in location_stats_lessthan10 else x)
len(df5.location.unique())
df5.head(10)
df5[df5.total_sqft/df5.bhk<300].head()
df5.shape
df6 = df5[~(df5.total_sqft/df5.bhk<300)]
df6.shape
df6.price_per_sqft.describe()
def remove(df):
    df_out = pd.DataFrame()
    for key,subdf in df.groupby('location'):
        m = np.mean(subdf.price_per_sqft)
        st = np.std(subdf.price_per_sqft)
        reduced_df = subdf[(subdf.price_per_sqft > (m-st))&(subdf.price_per_sqft <= (m+st))]
        df_out = pd.concat([df_out,reduced_df],ignore_index = True)
    return df_out
df7 = remove(df6)
df7.shape
def plot(df,location):
    bhk2 = df[(df.location == location) & (df.bhk == 2)]
    bhk3 = df[(df.location == location) & (df.bhk == 3)]
    matplotlib.rcParams['figure.figsize'] = (15,10)
    plt.scatter(bhk2.total_sqft,bhk2.price,color = 'blue',label = '2bhk',s = 50)
    plt.scatter(bhk3.total_sqft,bhk3.price,marker = '+',color = 'green',label = '3bhk',s = 50)
    plt.xlabel("Total sqft area")
    plt.ylabel("price per sqft")
    plt.title(location)
    plt.legend()
plot(df7,"Hebbal")
def remove(df):
    ex = np.array([])
    for location,location_df in df.groupby('location'):
        bhk_stats = {}
        for bhk,bhk_df in location_df.groupby('bhk'):
            bhk_stats[bhk] = {
```

```
            'mean':np.mean(bhk_df.price_per_sqft),
            'std':np.std(bhk_df.price_per_sqft),
            'count':bhk_df.shape[0]
            }
            for bhk,bhk_df in location_df.groupby('bhk'):
                stats = bhk_stats.get(bhk-1)
                if stats and stats['count']>5:
                    ex = np.append(ex,bhk_df[bhk_df.price_per_sqft < (stats['mean'])].index.values)
    return df.drop(ex,axis = 'index')
df8 = remove(df7)
df8.shape
plot(df8,"Hebbal")
import matplotlib
matplotlib.rcParams["figure.figsize"] = (20,10)
plt.hist(df8.price_per_sqft,rwidth = 0.8)
plt.xlabel("price per sqft")
plt.ylabel("Count")
df8.bath.unique()
df8[df8.bath>10]
plt.hist(df8.bath,rwidth = 0.8)
plt.xlabel("no of bathrooms")
plt.ylabel("count")
df8[df8.bath>df8.bhk+2]
df9 = df8[df8.bath<df8.bhk+2]
df9.shape
df10 = df9.drop(['size','price_per_sqft'],axis = 'columns')
df10.head(3)
dummies = pd.get_dummies(df10.location)# Set display option to show all columns
pd.set_option('display.max_columns', None) # Now display the DataFrame
print(dummies.head())
df11 = pd.concat([df10,dummies.drop('other',axis = 'columns')],axis = 'columns')
df11.head(3)
df12 = df11.drop('location',axis = 'columns')
df12.head(3)
df12.shape
x = df12.drop('price', axis=1)
print("Total columns:", len(x.columns))
print("Column names:", x.columns.tolist())
y = df12.price
y.head()
# Check the names of the columns related to location
location_columns = [col for col in x.columns if 'location' in col]
print(location_columns)
# Check if the location column exists in training data
loc_index = np.where(x.columns == 'Indira Nagar')[0]
print("Location index:", loc_index)  # It should print an index if the location exists
import sklearn
import sklearn.model_selection
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.2,random_state = 10)
from sklearn.linear_model import LinearRegression
```

```
lr_cf = LinearRegression()
lr_cf.fit(x_train,y_train)
lr_cf.score(x_test,y_test)
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import cross_val_score
cv = ShuffleSplit(n_splits = 5,test_size = 0.2,random_state = 0)
cross_val_score(LinearRegression(),x,y,cv = cv)
import numpy as np
def predict(location, sqft, bath, bhk):
    # Create an input vector of zeros with the correct number of features (244)
    x_input = np.zeros(len(x.columns))  # 244 features in the model
    # Set known values for sqft, bath, bhk in the corresponding indices (assuming they are the first
three features)
    x_input[0] = sqft  # Square footage
    x_input[1] = bath   # Number of bathrooms
    x_input[2] = bhk    # Number of BHK

    # Handle the location

if location in x.columns:
        loc_index = np.where(x.columns == location)[0][0]  # Get the index of the location column
        x_input[loc_index] = 1  # Set the corresponding location column to 1
    else:
        print(f"Location '{location}' not found in the model columns. Available locations:
{list(x.columns)}")
        return None  # Or assign a default location if preferred

    # Perform the prediction
    return float(lr_cf.predict([x_input])[0])

# Example usage:

result = predict('Indira Nagar', 1000, 2, 2)
if result is not None:
    print(f"Estimated price: ₹{result}")
predict('Yelaha',1000,2,2)
import pickle
with open('bangalore_home_prices_model.price','wb') as f:
    pickle.dump(lr_cf,f)
import json
columns = {
    'data_columns' : [col.lower() for col in x.columns]
}
with open("columns.json","w") as f:
    f.write(json.dumps(columns))
predict('Yelahanka',1000,2,2)
```

2. APP.PY (FLASK):

```
from flask import Flask, render_template, request, jsonify
from util import load_saved_artifacts, get_location_names, get_estimated_price
app = Flask(__name__, template_folder='templates', static_folder='static')
@app.route("/")
def index():
    return render_template('index.html')
@app.route('/api/location_names', methods=['GET'])
def api_get_location_names():
    locations = get_location_names()
    return jsonify({'locations': locations})
@app.route('/api/predict_home_price', methods=['POST'])
def api_predict_home_price():
    try:
        total_sqft = float(request.form['total_sqft'])
        location = request.form['location']
        bhk = int(request.form['bhk'])
        bath = int(request.form['bath'])
        estimated_price = get_estimated_price(location, total_sqft, bhk, bath)
        response = jsonify({
            'estimated_price': estimated_price
        })
        response.headers.add('Access-Control-Allow-Origin', '*')
        return response

    except Exception as e:
        print("Error in prediction API:", e)
        return jsonify({'error': str(e)}), 400
if __name__ == "__main__":
    print("Starting Python Flask Server For Home Price Prediction...")
    load_saved_artifacts()
    app.run(debug=True)
```

3. UTIL.PY

```
import os
import json
import pickle
import numpy as np
__locations = None
__data_columns = None
__model = None
def get_estimated_price(location, sqft, bhk, bath):
    """Function to get estimated price based on the inputs."""
    try:
        loc_index = __data_columns.index(location.lower())
    except ValueError:
        loc_index = -1
    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():
    """Function to load the saved model and other artifacts."""
    print("loading saved artifacts...start")
    global __data_columns
    global __locations
    current_folder = os.path.dirname(__file__)
    artifacts_folder = os.path.join(current_folder, 'artifacts')
    columns_path = os.path.join(artifacts_folder, 'columns.json')
    with open(columns_path, 'r') as f:
        columns_data = json.load(f)
        __data_columns = columns_data['data_columns']
        __locations = __data_columns[3:]
    global __model
    if __model is None:
        model_path = os.path.join(artifacts_folder, 'banglore_home_prices_model.pickle')
        with open(model_path, 'rb') as f:
            __model = pickle.load(f)
    print("loading saved artifacts...done")
def get_location_names():
    """Return the list of location names."""
    return __locations
def get_data_columns():
    """Return the list of all data columns (including location and other features)."""
    return __data_columns
if __name__ == '__main__':
    load_saved_artifacts()
    print(get_location_names())
    print(get_estimated_price('1st Phase JP Nagar', 1000, 3, 3))
    print(get_estimated_price('1st Phase JP Nagar', 1000, 2, 2))
    print(get_estimated_price('Kalhalli', 1000, 2, 2))
    print(get_estimated_price('Ejipura', 1000, 2, 2))
```

4. INDEX.HTML : (FRONTEND)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Home Price Prediction</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='app.css') }}?v=1">
    <link rel="icon" href="{{ url_for('static', filename='favicon.ico') }}" type="image/x-icon">
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
</head>
<body>
    <div class="img"></div>
```

```
<div id="main" class="form">
  <h2>Home Price Prediction</h2>
  <label for="uiLocations">Location</label>
  <select id="uiLocations" class="location" name="location">
    <option value="">Select Location</option>
  </select>
  <label for="uiSqft">Total Square Feet</label>
  <input type="number" id="uiSqft" class="area" name="total_sqft" placeholder="Enter total
 square feet">
  <div class="switch-group">
    <label class="group-label">BHK</label>
    <div class="switch-field">
      <input type="radio" id="bhk1" name="uiBHK" value="1">
      <label for="bhk1">1 BHK</label>

      <input type="radio" id="bhk2" name="uiBHK" value="2">
      <label for="bhk2">2 BHK</label>

      <input type="radio" id="bhk3" name="uiBHK" value="3">
      <label for="bhk3">3 BHK</label>

      <input type="radio" id="bhk4" name="uiBHK" value="4">
      <label for="bhk4">4 BHK</label>
    </div>
  </div>
  <div class="switch-group">
    <label class="group-label">Bathrooms</label>
    <div class="switch-field">
      <input type="radio" id="bath1" name="uiBathrooms" value="1">
      <label for="bath1">1 Bathroom</label>

      <input type="radio" id="bath2" name="uiBathrooms" value="2">
      <label for="bath2">2 Bathrooms</label>

      <input type="radio" id="bath3" name="uiBathrooms" value="3">
      <label for="bath3">3 Bathrooms</label>

      <input type="radio" id="bath4" name="uiBathrooms" value="4">
      <label for="bath4">4 Bathrooms</label>
    </div>
  </div>
  <button id="estimatePriceButton" class="submit">Estimate Price</button>
  <div id="uiEstimatedPrice" class="result">
    Estimated Price: ₹<span>0</span> L
  </div>
</div>
<script>
  $(document).ready(function() {
    $.get("/api/location_names", function(data) {
      if (data.locations && data.locations.length > 0) {
        data.locations.forEach(function(location) {
```

```
                $('#uiLocations').append('<option value="' + location + '">' + location +
    '</option>');
                });
            } else {
                console.log("No locations found.");
            }
        });
        $('#estimatePriceButton').on('click', function() {
            var total_sqft = $('#uiSqft').val();
            var location = $('#uiLocations').val();
            var bhk = $("input[name='uiBHK']:checked").val();
            var bath = $("input[name='uiBathrooms']:checked").val();
            if (!total_sqft || !location || !bhk || !bath) {
                alert("Please fill in all the fields!");
                return;
            }
            $.post("/api/predict_home_price", {
                total_sqft: total_sqft,
                location: location,
                bhk: bhk,
                bath: bath
            })
            .done(function(response) {
                $('#uiEstimatedPrice span').text(response.estimated_price);
            })
            .fail(function(xhr, status, error) {
                console.error("Error fetching price:", error);
                console.error("Response:", xhr.responseText);
                alert("Error in fetching the price. Please try again.");
            });
        });
    });
  </script>
</body>
</html>
```

5. APP.CSS : (STYLING)

```
@import url('https://fonts.googleapis.com/css?family=Roboto:300');
.img {
  background: url('https://s3-ap-southeast-1.amazonaws.com/housingman-
v2/perspectives/images/3874/original/2.jpg?1471327564');
  background-repeat: no-repeat;
  background-size: cover;
  filter: blur(15px);
  position: fixed;
  width: 100%;
  height: 100%;
  top: 0;
  left: 0;
```

```
    z-index: -1;
}
body, html {
    height: 100%;
    font-family: 'Roboto', sans-serif;
    margin: 0;
    padding: 0;
}

.form {
    max-width: 300px;
    margin: 50px auto;
    background-color: rgba(255, 255, 255, 0.8);
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.3);
    z-index: 2;
}

h2 {
    font-size: 20px;
    color: #3c3c3c;
    text-align: center;
    margin-bottom: 20px;
}
.area, .location {
    width: 100%;
    padding: 10px;
    margin: 10px 0;
    background: #f9f9f9;
    border: 1px solid #ccc;
    border-radius: 5px;
    box-sizing: border-box;
    font-size: 14px;
}
.switch-group {
    margin-bottom: 20px;
}

.group-label {
    font-weight: bold;
    display: block;
    margin-bottom: 8px;
    color: #3c3c3c;
}

.switch-field {
    display: flex;
    flex-wrap: nowrap;
    justify-content: space-between;
    margin-bottom: 36px;
```

```
    overflow: hidden;
}

.switch-field label {
    background-color: #f0e3d3;
    color: rgba(0, 0, 0, 0.7);
    font-size: 14px;
    text-align: center;
    padding: 8px 16px;
    border: 1px solid rgba(0, 0, 0, 0.2);
    box-shadow: inset 0 1px 3px rgba(0, 0, 0, 0.3), 0 1px rgba(255, 255, 255, 0.1);
    flex: 1;
    margin: 0 2px;
    transition: all 0.1s ease-in-out;
}

.switch-field input {
    position: absolute !important;
    clip: rect(0, 0, 0, 0);
    height: 1px;
    width: 1px;
    border: 0;
    overflow: hidden;
}

.switch-field input:checked + label {
    background-color: #b7e0b5;
    box-shadow: none;
}

.switch-field label:hover {
    cursor: pointer;
}

.switch-field label:first-of-type {
    border-radius: 4px 0 0 4px;
}

.switch-field label:last-of-type {
    border-radius: 0 4px 4px 0;
}
.submit {
    background: #a5dc86;
    width: 100%;
    border: 0;
    padding: 10px;
    margin: 20px 0;
    font-size: 16px;
    font-weight: bold;
    color: #fff;
    text-align: center;
```

```
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s ease, box-shadow 0.3s ease;
}

.submit:hover {
    background: #8fc7a5;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.3);
}
.result {
    background: #f0e3d3;
    width: 100%;
    padding: 10px;
    font-size: 16px;
    font-weight: bold;
    text-align: center;
    border-radius: 5px;
    color: #3c3c3c;
}
@media screen and (max-width: 480px) {
    .form {
        max-width: 90%;
        margin: 20px auto;
    }

    h2 {
        font-size: 18px;
    }

    .submit {
        font-size: 14px;
        padding: 8px;
    }

    .result {
        font-size: 14px;
    }
}
```

6. APP.JS :

```
$(document).ready(function() {
    $.get("/api/location_names", function(data) {
        if (data.locations && data.locations.length > 0) {
            data.locations.forEach(function(location) {
                $('#uiLocations').append('<option value="' + location + '">' + location + '</option>');
            });
        } else {
            console.log("No locations found.");
        }
```

```
    });
    $('#estimatePriceButton').on('click', function() {
        var total_sqft = $('#uiSqft').val();
        var location = $('#uiLocations').val();
        var bhk = $("input[name='uiBHK']:checked").val();
        var bath = $("input[name='uiBathrooms']:checked").val();
        $('#uiEstimatedPrice span').text('Loading...');
        $.post("/api/predict_home_price", {
            total_sqft: total_sqft,
            location: location,
            bhk: bhk,
            bath: bath
        }, function(response) {
            if(response && response.estimated_price) {
                $('#uiEstimatedPrice span').text('₹' + response.estimated_price);
            } else {
                $('#uiEstimatedPrice span').text('Error: Price could not be determined');
            }
        }).fail(function(xhr, status, error) {
            console.log("Error: " + error);
            $('#uiEstimatedPrice span').text('Error: Could not fetch price');
        });
    });
});
```

FOLDER STRUCTURE :

```
mini project/
├── data/
│   ├── columns.json
│   ├── bangalore_home_prices_model.price
│   └── bangalore_home_prices_final.ipynb
├── model/
│   └── bangalore_home_prices_final.ipynb
├── server/
│   ├── app.py
│   ├── util.py
│   ├── artifacts/
│   │   ├── columns.json
│   │   ├── bangalore_home_prices_model.price
│   │   └── bangalore_home_prices_model.pickle
│   ├── static/
│   │   ├── app.css
│   │   ├── app.js
│   │   └── favicon.ico
│   ├── templates/
│   │   └── index.html
```

GITHUB LINK :

https://github.com/Harshasri26/home-price-prediction

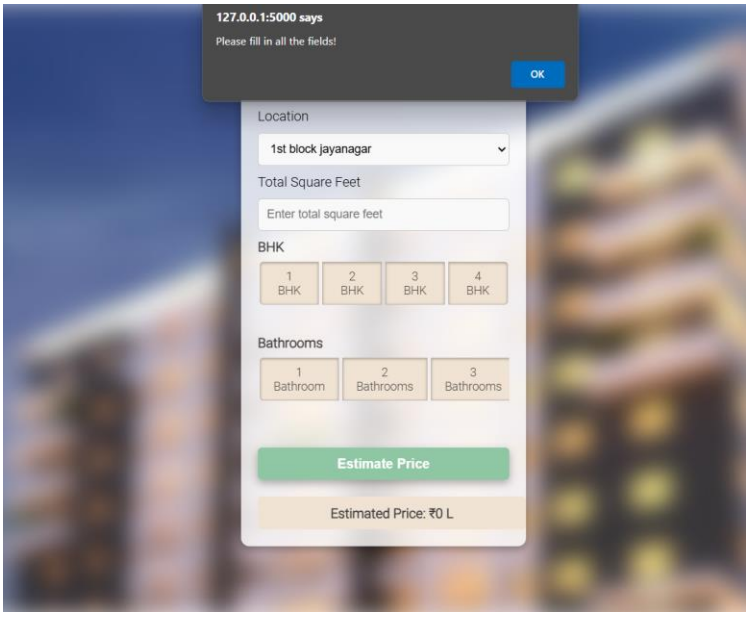
C . TESTING :

Test Case Template

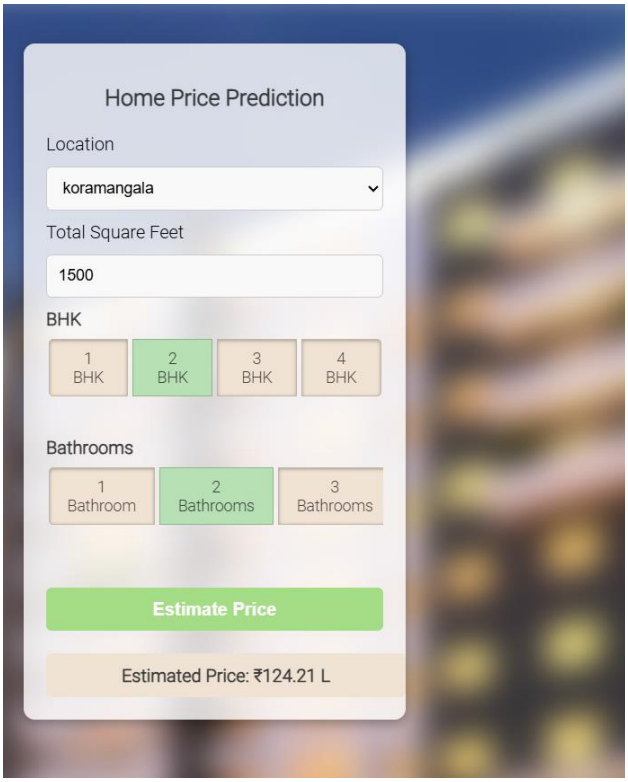
Test Case ID	Feature	Test Steps	Expected Result	Actual Result	Status
TC001	Predict Price	Input valid location, sqft, BHK, and bathroom data	Correct price prediction displayed	Pass	Pass
TC002	Input Validation	Leave any input field empty	Displays "Invalid input" or similar message	Pass	Pass

RESULTS

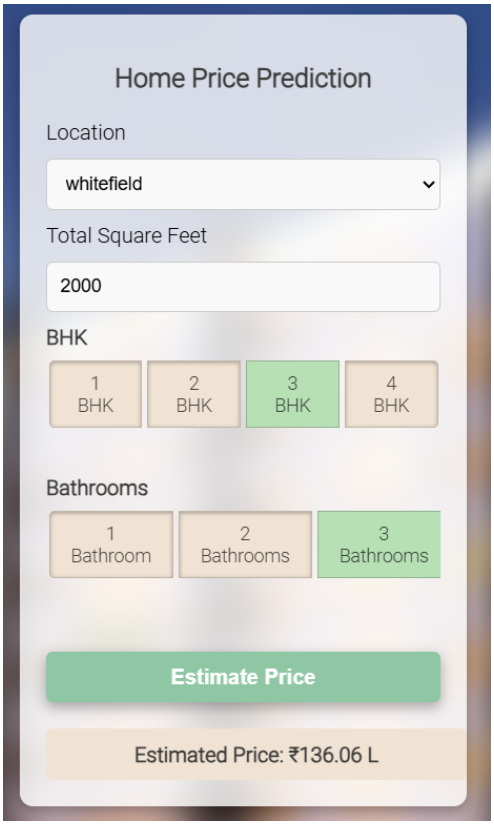
CASE 1:



CASE 2:



CASE 3 :


Home Price Prediction
Location
whitefield
Total Square Feet
2000
BHK
1 BHK
2 BHK
3 BHK
4 BHK
Bathrooms
1 Bathroom
2 Bathrooms
3 Bathrooms
Estimate Price
Estimated Price: ₹136.06 L

Additional Knowledge Gained

Implementing this mini-project on house price prediction went beyond the core syllabus of Python programming and provided extensive practical exposure to real-world applications of programming and data science. The additional knowledge gained includes:

1. Machine Learning Fundamentals

- Learned the process of building a predictive model, from data cleaning to evaluation.
- Gained hands-on experience with **linear regression**, a key supervised learning algorithm.
- Understood the importance of feature engineering and dimensionality reduction in improving model accuracy.

2. Data Manipulation and Visualization

- Mastered the use of libraries like **NumPy** and **Pandas** for data preprocessing, handling missing values, and transforming datasets.
- Enhanced skills in data visualization using **Matplotlib**, enabling better insights into patterns and trends in the dataset**.**

2. Backend Development with Flask

- Learned how to set up a **Python Flask server** to handle HTTP requests and serve a machine learning model.
- Understood routing in Flask, API integration, and handling POST requests for dynamic interactions.
- Implemented file handling for loading pre-trained models and JSON data.

3. Frontend Development and Integration

- Developed a functional web interface using **HTML**, **CSS**, and **JavaScript**.
- Integrated the backend with the frontend to create a seamless user experience for inputting house details and fetching predictions.
- Used CSS for styling and JavaScript for interactive elements, enhancing the visual appeal and usability of the website.

4. Working with JSON and Artifacts

- Gained experience in managing artifacts such as the trained model (pickle) and column mappings (JSON).
- Learned to serialize and deserialize data for efficient storage and retrieval.

Project Structuring and Best Practices

- Organized the project into modular components for better scalability and maintainability.
- Followed a clean folder structure to separate data, server, and frontend components.

5. Version Control and Collaboration

- Understood the importance of using **GitHub** for version control and collaboration.
- Learned how to document and share projects with clear folder structures and instructions for reproducibility.

6. Problem-Solving and Debugging

- Gained experience in troubleshooting issues during development, such as model integration errors, frontend-backend communication problems, and styling challenges.

This project has provided a comprehensive understanding of integrating Python programming with machine learning, backend development, and frontend technologies to build a complete application. It bridges the gap between theoretical learning and real-world application, preparing for future projects and advanced topics in software and data engineering.

Conclusion and Future Work

Conclusion

The house price prediction mini-project has been an insightful and valuable learning experience. By combining various aspects of programming, data science, machine learning, and web development, it has provided a comprehensive understanding of how to build an end-to-end predictive system. The project successfully demonstrated the full workflow, starting from data preprocessing and model training, to deploying the model via a Flask backend, and integrating it with a frontend for real-time predictions. The use of **linear regression**, data cleaning, feature engineering, and model evaluation through techniques helped in achieving a robust and efficient prediction model. Additionally, building a dynamic web interface in HTML, CSS, and JavaScript provided hands-on experience in developing full-stack applications.
Through this project, I have gained valuable knowledge in machine learning, data manipulation, and full-stack development. The integration of machine learning with web technologies is crucial in the current landscape of software applications, and this project has helped me bridge the gap between theoretical knowledge and practical implementation.

Future Work

Although the current implementation serves its purpose effectively, there are several areas where the project can be enhanced:

1. **Model Improvement**
    - **Advanced Machine Learning Models:**

      The current project uses linear regression. Exploring more sophisticated models like **Random Forest**, **Gradient Boosting**, or **XGBoost** could potentially improve prediction accuracy

    - **Ensemble Methods:**

      Combining multiple models through techniques like bagging and boosting could also help enhance the overall model performance.

2. **Data Expansion**
    - Currently, the model is based on a limited dataset for Bangalore. Expanding the dataset to include more cities or regions would allow for more generalized predictions, improving the versatility of the model.

3. **Real-Time Data Integration**
    - To make the predictions even more accurate, integrating real-time data such as market trends, interest rates, or economic factors could provide more context for the pricing predictions.

4. **User Interface Enhancements**

- The current web interface can be made more interactive by implementing real-time validation of user inputs, and providing more detailed feedback (e.g., range of predicted prices based on varying input features).
- The website could be optimized for mobile users to provide a better experience across devices.

5. **Deployment and Scaling**

   - Deploying the project on a cloud platform (such as **AWS**, **Azure**, or **Heroku**) could make the application more accessible. This would also allow for scalability, supporting more users and requests without performance degradation.
   - Implementing **Docker** could help containerize the application, ensuring consistent deployment across different environments.

6. **Additional Features**

   - **User Account Management:** Allow users to save their predicted results or access historical data.
   - **Data Visualization:** Adding graphs and charts for visual representation of predicted house prices based on location, area, etc., would make the results more insightful for users.

References

1. Kaggle. **“Bangalore Home Prices Dataset.”** Retrieved from https://www.kaggle.com.

2. Python Software Foundation. **“Python Documentation.”** Retrieved from https://docs.python.org/3/.

3. Numpy Developers. **“Numpy: Scientific Computing with Python.”** Retrieved from https://numpy.org/.

4. Pandas Developers. **“Pandas Documentation.”** Retrieved from https://pandas.pydata.org/.

5. Scikit-learn Developers. **“Scikit-learn: Machine Learning in Python.”** Retrieved from https://scikit-learn.org/stable/.

6. Matplotlib Developers. **“Matplotlib: Python Plotting Library.”** Retrieved from https://matplotlib.org/.

7. Flask Developers. **“Flask Documentation.”** Retrieved from https://flask.palletsprojects.com/.

8. W3Schools. **“Web Development Tutorials: HTML, CSS, and JavaScript.”** Retrieved from https://www.w3schools.com/.

9. GeeksforGeeks. **“Introduction to Flask and Full-Stack Development.”** Retrieved from https://www.geeksforgeeks.org/.

10. GitHub. **"Code Repository Management and Collaboration."** Retrieved from https://github.com/.

11. Stack Overflow. **“Programming Community Q&A.”** Retrieved from https://stackoverflow.com/.