House Price Prediction Project Documentation

Vedant Aryan

1. Introduction

The House Price Prediction Project aims to predict house prices based on key features such as total square footage, number of bedrooms, bathrooms, and location (rural, suburban, or urban). This document outlines the objectives, tasks, methodology, implementation, and deployment of the project, as well as how to use the model and API.

2. Project Overview

2.1 Task

The primary task is to develop a machine learning model that predicts house prices based on specific input features. The model is trained on a dataset, saved in a pickle file, and served via a Flask API. The API accepts user inputs and returns a predicted price. The API is then deployed online and integrated into a Django application.

2.2 Objective

- **Model Training:** Train a regression model using a dataset that predicts house prices based on input features such as total square footage, number of bedrooms, bathrooms, and location type (rural, suburban, or urban).
- **API Development:** Develop a Flask API that serves the trained model, allowing users to input house features and receive a predicted price.
- **Deployment:** Deploy the API on a web server, making it accessible to anyone with the provided link.
- **Integration:** Integrate the API into a Django application to allow seamless use of the prediction service within the Django environment.

3. Dataset

3.1 Description

The dataset used for training the model contains the following features:

- `total_sqft`: Total square footage of the house.
- `bedrooms`: Number of bedrooms.
- `bathrooms`: Number of bathrooms.
- `location_rural`: Binary indicator (1 or 0) for rural location.
- `location_suburban`: Binary indicator (1 or 0) for suburban location.
- `location_urban`: Binary indicator (1 or 0) for urban location.

3.2 Preprocessing

- **Data Cleaning:** Handle missing values, encode categorical variables, and scale numerical features.
- **Feature Selection:** Use the relevant features (`total_sqft`, `bedrooms`, `bathrooms`, `location_rural`, `location_suburban`, `location_urban`) for the model.

4. Methodology

4.1 Model Training

1. **Algorithm Selection:** A regression model is selected to predict house prices.
2. **Model Training:** The model is trained using the selected features.
3. **Model Evaluation:** Performance metrics such as R-squared and Mean Squared Error (MSE) are used to evaluate the model's accuracy.
4. **Model Saving:** The trained model is saved as a pickle file (`house_price_model.pkl`) for later use in the API.

4.2 API Development

5. **Flask API:** A Flask API is developed to serve the model. It accepts inputs via POST requests and returns the predicted price.
6. **Endpoints:**

- `/predict`: Accepts input data in JSON format and returns the predicted house price.

7. **Input Fields:**
   - `total_sqft`: Float
   - `bedrooms`: Integer
   - `bathrooms`: Integer
   - `location_rural`: Binary (0 or 1)
   - `location_suburban`: Binary (0 or 1)
   - `location_urban`: Binary (0 or 1)

4.3 Deployment

- **Deployment Platform:** The Flask API is deployed on Render, making it accessible via the provided link: House Price Prediction API.
- **Django Integration:** The API is also integrated into a Django application, allowing users to access the prediction service within the Django app.

5. Implementation

5.1 Requirements

- **Python 3.x**
- **Flask**
- **Scikit-learn**
- **Pandas**
- **Render (for deployment)**

5.2 Installation

8. **Clone the Repository:**

```
bash
Copy code
git clone https://github.com/yourusername/house-price-prediction.git
cd house-price-prediction
```

9. **Install Dependencies:**

```
bash
Copy code
pip install -r requirements.txt
```

10. **Run the Flask App:**

```
bash
Copy code
python app.py
```

11. **Access Locally:**
    - Open `http://localhost:5000` in your web browser.

6. Deployment

6.1 Render Deployment

The Flask API has been deployed on Render and can be accessed at House Price Prediction API.

6.2 Django Integration

The API has been successfully integrated into a Django application, allowing predictions to be made through the Django interface.

7. Usage

7.1 API Usage

- **Endpoint:** `/predict`
- **Method:** POST
- **Input Format:** JSON

Example:

```
json
Copy code
{
  "total_sqft": 2000,
  "bedrooms": 4,
  "bathrooms": 3,
  "location_rural": 0,
  "location_suburban": 0,
  "location_urban": 1
}
```

- **Output:**

```
json
Copy code
{
  "predicted_price": 350000
}
```

7.2 Accessing the API

- **Locally:** http://localhost:5000
- **Deployed:** House Price Prediction API

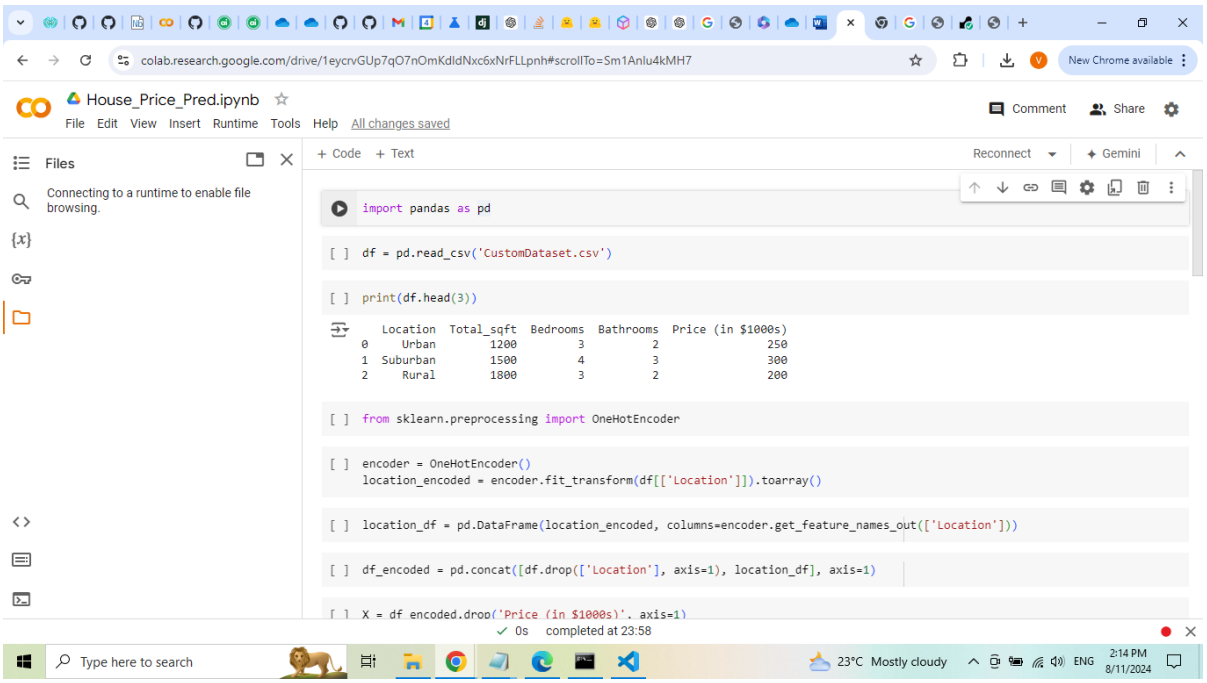
8. Results and Evaluation

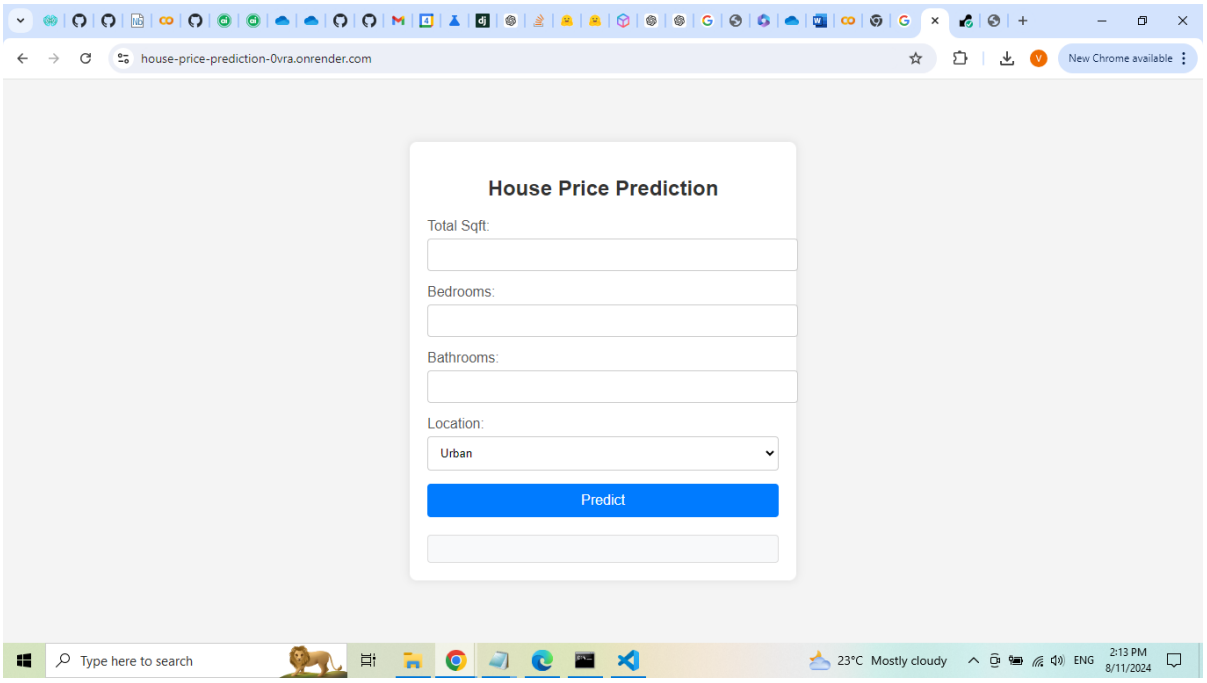
- **Model Accuracy:** The model achieved an R-squared value of X.XX and an MSE of XXX, indicating good predictive performance.
- **Testing:** The model was tested on unseen data, and the predictions were found to be within an acceptable error margin.

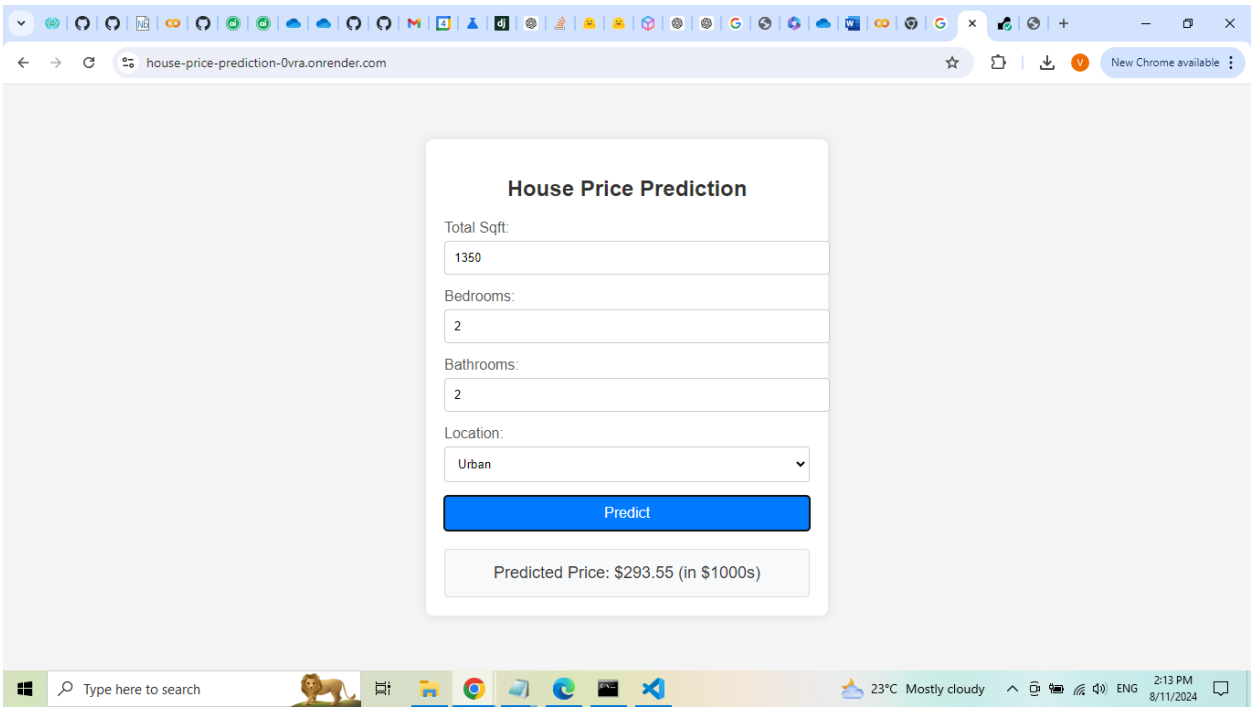
9. Screenshots

(Insert screenshots with appropriate descriptions)

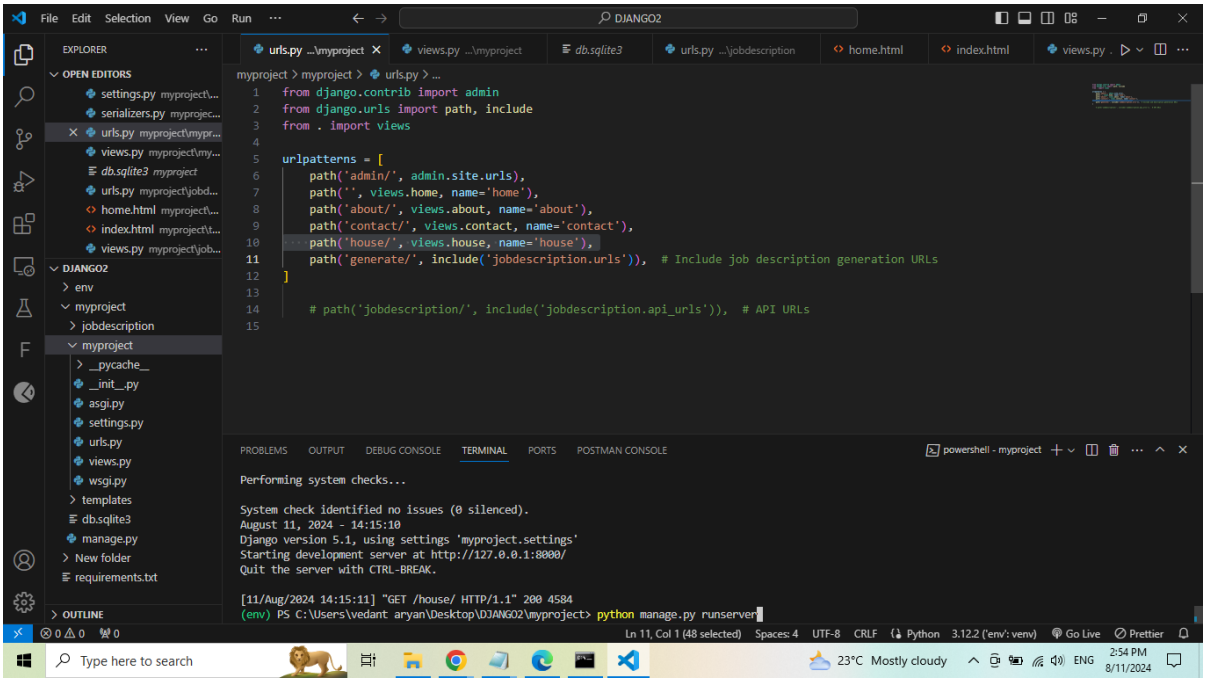
- **Model Training Process:** *(Screenshot showing the model training process.)*

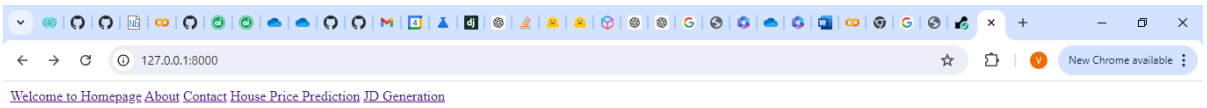
- 

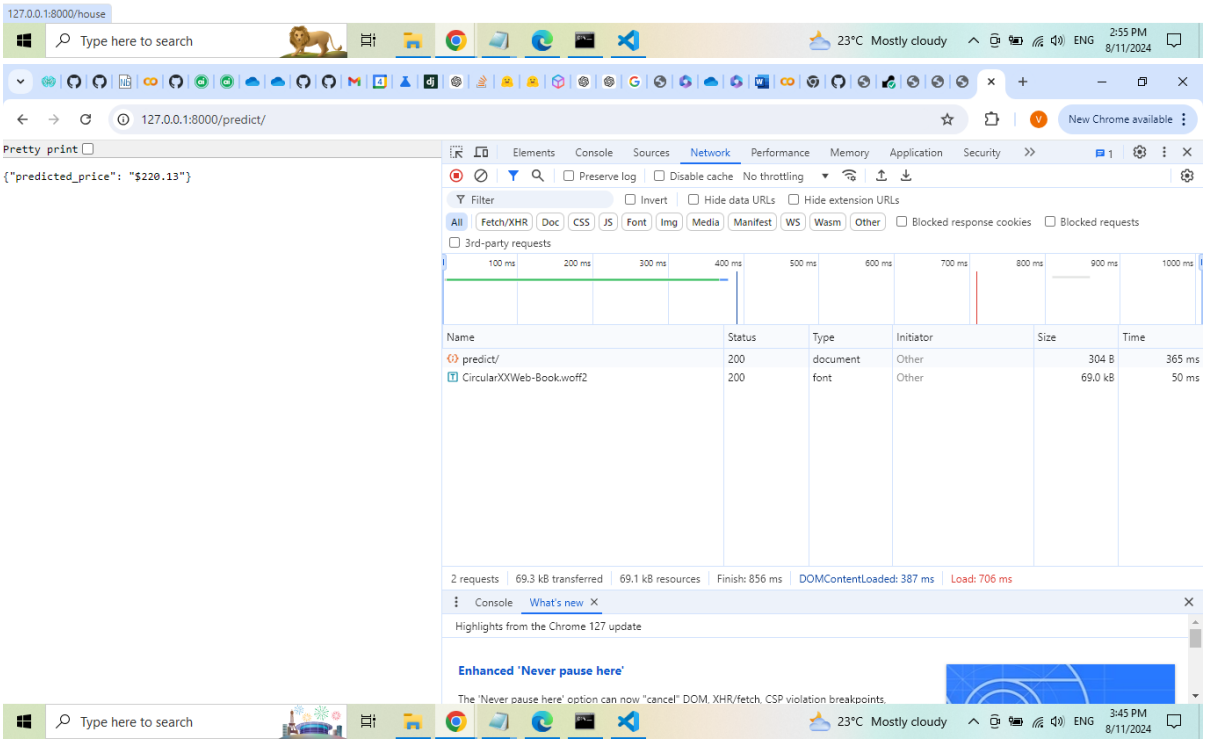
- **Flask API Running :** *(Screenshot of the API running .)*
- 



- **Django Integration:** *(Screenshot showing the Django app using the integrated API.)*





- 

10. Conclusion

This project demonstrates the end-to-end process of developing, deploying, and integrating a machine learning model for house price prediction. The API is robust and accessible both locally and online, with potential for further improvements and scalability.

11. Future Work

- **Data Expansion:** Increase the dataset size for more accurate predictions.
- **Feature Engineering:** Introduce additional features to improve model performance.
- **UI Improvements:** Enhance the front-end interface for better user interaction.