00:00-21:10

Building a Python Flask Server for Home Price Prediction

Introduction

In this article, we will discuss how to build a Python Flask server that can handle HTTP requests from a user interface (UI) and predict home prices. The Flask server will serve as the backend for our UI application. We will guide you through the process of setting up the server, creating the necessary routines, and testing the endpoints.

Setting up the Project

To begin, make sure you have PyCharm Community Edition installed, which can be downloaded for free from the JetBrains website. Once installed, open the project in PyCharm. The project directory should have three subfolders: client, server, and model. The client folder will house the UI application, the server folder will contain the Flask server code, and the model folder will store the notebook and exported artifacts (saved model and columns.json) from previous tutorials.

Creating the Flask Server

1. In the server folder, create a new file called server.py.
2. Import the Flask module in the server.py file. Flask allows us

to write Python services that can handle HTTP requests.
3. Configure the Python interpreter to use Anaconda (or any other preferred interpreter) as it comes with Flask pre-installed.
4. Create a Flask app using the `app = Flask(__name__)` line.
5. In the main function, run the application using `app.run()`.
6. Write a simple "hello" route by adding the following code:

   ```
   @app.route('/hello')
   def hello():
       return 'Hi'
   ```

   This route will return the response 'Hi' when accessed.

Testing the Flask Server

1. Run the server by executing `python server.py`.
2. Copy the URL displayed in the terminal and paste it into your browser.
3. Append `/hello` to the URL and press Enter.
4. You should see the response 'Hi' displayed in your browser, indicating that the Flask server is up and running.

Retrieving Location Names

Next, we will create a routine to retrieve location names from a JSON file. This routine will be used to populate a dropdown menu in the UI application.

1. Create a subdirectory called "artifacts" within the server directory.
2. Copy the exported artifacts (model and columns.json) into the

artifacts directory.
3. Read the columns.json file to extract the location names.
4. Create a new file called util.py in the server directory to store utility functions.
5. In util.py, define a function called `get_location_names()` to retrieve the location names.
6. Return the location names using the `jsonify()` method.
7. Test the `get_location_names()` function to ensure it returns the expected results.

Estimating Home Prices

We will now create a function to estimate home prices based on location, square footage, number of bedrooms, and number of bathrooms.

1. Define a function called `get_estimated_price(location, sq_ft, bhk, bathroom)` that takes these parameters.
2. Load the saved artifacts (model and columns.json) into memory.
3. Create a numpy array with zeros to hold the input values.
4. Find the index corresponding to the provided location in the columns list.
5. Set the corresponding element in the numpy array to 1 and the remaining elements to 0 using one-hot encoding.
6. Use the loaded model to predict the home price by calling the `predict()` method.
7. Round the predicted price to two decimal places.
8. Return the estimated price.

Implementing the Predict Home Price Endpoint

In the server.py file, create a new endpoint called `/predict_home_price` using the HTTP POST method.

1. Import the `request` class to retrieve the input parameters from the request form.
2. Get the input values (total square footage, location, BHK, and bathroom) from the request form.
3. Call the `get_estimated_price()` function with the input parameters to get the estimated price.
4. Return the estimated price as the response.

Testing the Endpoints

To test the endpoints, we can use an application like Postman.

1. Open Postman and select the GET method.
2. Enter the URL with the endpoint `/get_location_names` and click Send.
3. The response will contain a list of all the location names.
4. Switch to the POST method and set the URL to `/predict_home_price`.
5. Specify the input parameters (total square footage, location, BHK, and bathroom) in the form data.
6. Click Send, and the response will contain the estimated home price.

By following these steps, you can successfully build a Python Flask server that can handle HTTP requests and predict home prices based on user input. With the Flask server in place, you can now proceed to develop the UI for a complete application.

In the next article, we will guide you through building the UI, which will integrate with the Flask server to provide a seamless user experience.