



# Week5: Cloud and API deployment

Name: Munirah Alfahaid

Batch code: LISUM35

Submission date: 28/7/2024

Submitted to: Data Glacier

## Introduction

This assignment involves building and deploying a machine learning model for predicting real estate prices based on various features. The primary objectives are to:

1. Train a regression model using a provided real estate dataset.
2. Save the trained model using pickle for later use.
3. Create a web application using Flask to accept input data, predict the house price per unit area, and display the results.
4. Deploy the web application on Heroku for accessible and scalable usage.

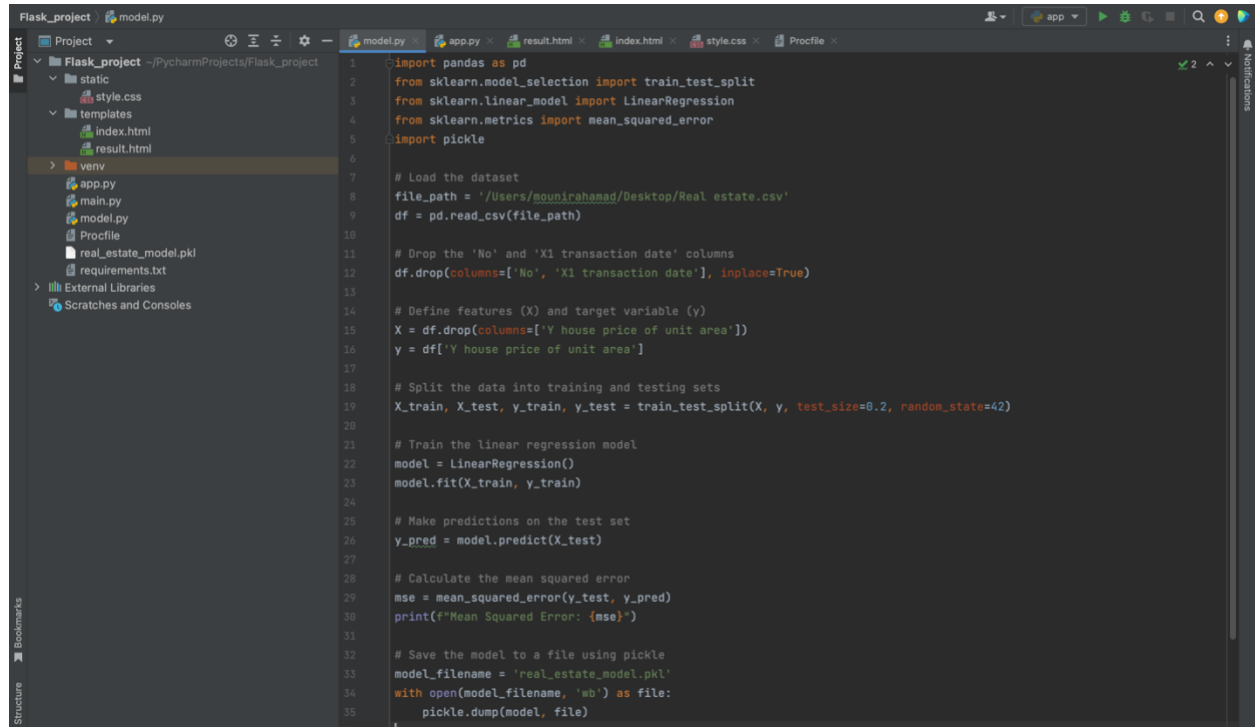
The dataset consists of information about house transactions, including the age of the house, distance to the nearest MRT station, number of convenience stores nearby, and the geographical coordinates (latitude and longitude). The target variable is the house price per unit area.

To complete this assignment, the project is structured with separate files for training the model, setting up the Flask web application, and defining HTML templates and CSS for the user interface.

The web application allows users to input the necessary features, predicts the house price using the trained model, and displays the prediction on a results page. This assignment demonstrates the practical application of machine learning in a real-world scenario, integrating model training, persistence, and deployment in a user-friendly web interface hosted on Heroku.

## Snapshot of deployment steps:

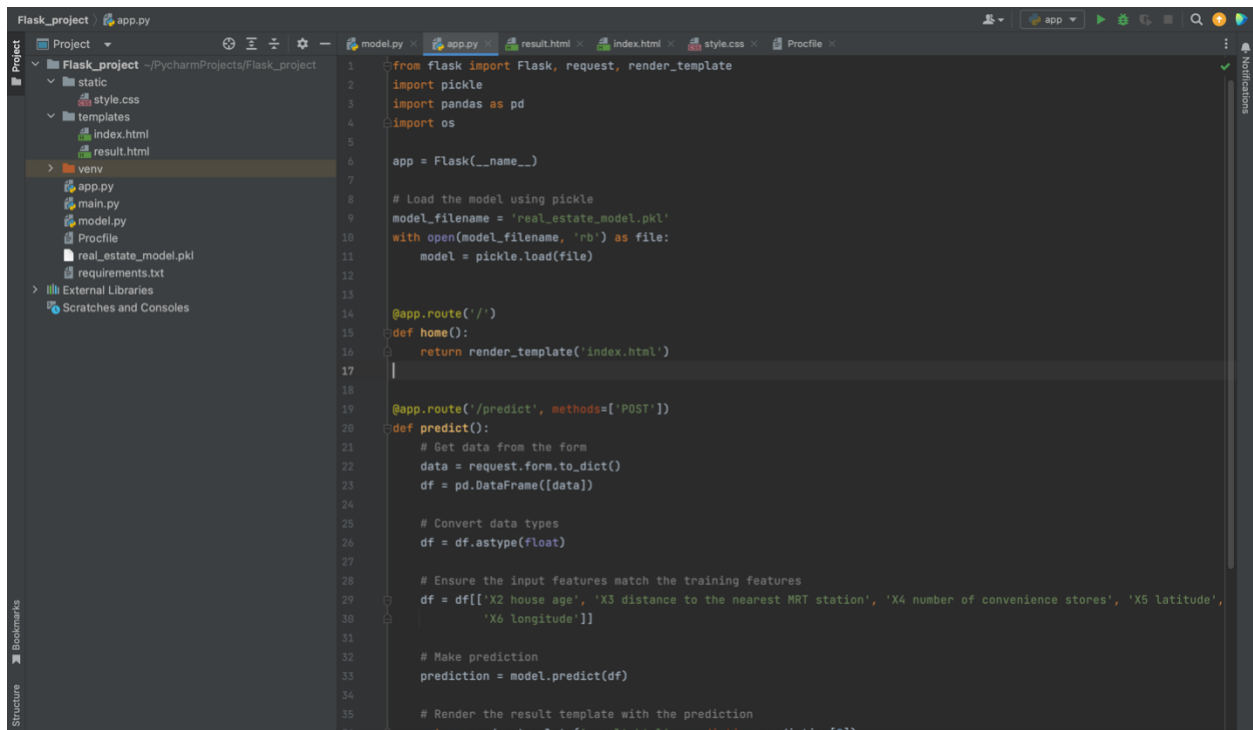
### 1-model.py



The screenshot shows a PyCharm IDE with a project named 'Flask\_project'. The left sidebar displays the project structure, including files like 'static', 'style.css', 'templates', 'index.html', 'result.html', 'venv', 'app.py', 'main.py', 'model.py', 'Procfile', 'real\_estate\_model.pkl', and 'requirements.txt'. The main editor window shows the code for 'model.py'.

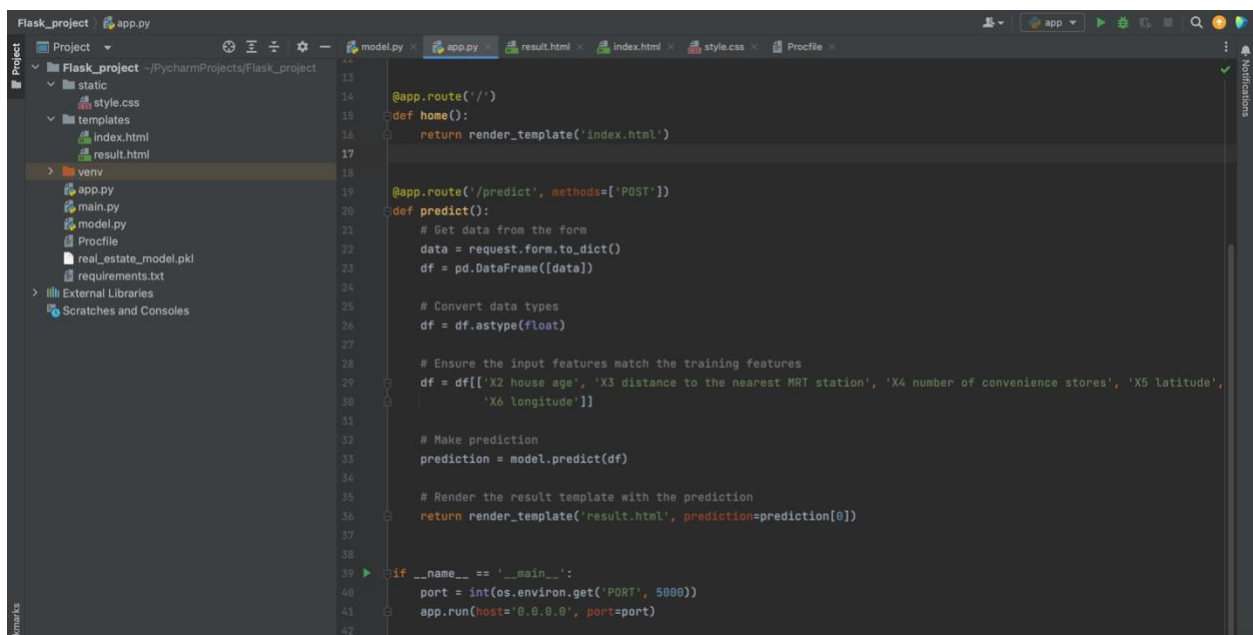
```
1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3 from sklearn.linear_model import LinearRegression
4 from sklearn.metrics import mean_squared_error
5 import pickle
6
7 # Load the dataset
8 file_path = '/Users/gounirahmad/Desktop/Real estate.csv'
9 df = pd.read_csv(file_path)
10
11 # Drop the 'No' and 'X1 transaction date' columns
12 df.drop(columns=['No', 'X1 transaction date'], inplace=True)
13
14 # Define features (X) and target variable (y)
15 X = df.drop(columns=['Y house price of unit area'])
16 y = df['Y house price of unit area']
17
18 # Split the data into training and testing sets
19 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
20
21 # Train the linear regression model
22 model = LinearRegression()
23 model.fit(X_train, y_train)
24
25 # Make predictions on the test set
26 y_pred = model.predict(X_test)
27
28 # Calculate the mean squared error
29 mse = mean_squared_error(y_test, y_pred)
30 print(f"Mean Squared Error: {mse}")
31
32 # Save the model to a file using pickle
33 model_filename = 'real_estate_model.pkl'
34 with open(model_filename, 'wb') as file:
35     pickle.dump(model, file)
```

## 2-app.py



This screenshot shows the first 35 lines of the `app.py` file in a PyCharm IDE. The code imports Flask, request, render\_template, pickle, pandas, and os. It initializes a Flask app and loads a pre-trained model from `real_estate_model.pkl`. A `home()` route is defined to render `index.html`. A `predict()` route is defined to handle POST requests, parse the form data into a DataFrame, convert it to floats, and use the loaded model to make a prediction.

```
1 from flask import Flask, request, render_template
2 import pickle
3 import pandas as pd
4 import os
5
6 app = Flask(__name__)
7
8 # Load the model using pickle
9 model_filename = 'real_estate_model.pkl'
10 with open(model_filename, 'rb') as file:
11     model = pickle.load(file)
12
13
14 @app.route('/')
15 def home():
16     return render_template('index.html')
17
18
19 @app.route('/predict', methods=['POST'])
20 def predict():
21     # Get data from the form
22     data = request.form.to_dict()
23     df = pd.DataFrame([data])
24
25     # Convert data types
26     df = df.astype(float)
27
28     # Ensure the input features match the training features
29     df = df[['X2 house age', 'X3 distance to the nearest MRT station', 'X4 number of convenience stores', 'X5 latitude',
30             'X6 longitude']]
31
32     # Make prediction
33     prediction = model.predict(df)
34
35     # Render the result template with the prediction
```



This screenshot shows the continuation of the `app.py` file from line 36 to 42. It completes the `predict()` function by returning the prediction result in the `result.html` template. It also includes a standard `if __name__ == '__main__':` block to run the app on port 5000.

```
36     return render_template('result.html', prediction=prediction[0])
37
38
39 if __name__ == '__main__':
40     port = int(os.environ.get('PORT', 5000))
41     app.run(host='0.0.0.0', port=port)
42
```

### 3-index.html

```
Flask_project templates index.html
Project
  Flask_project - PycharmProjects/Flask_project
    static
    templates
      index.html
      result.html
    venv
      app.py
      main.py
      model.py
      Profile
      real_estate_model.pkl
      requirements.txt
    External Libraries
    Scratches and Consoles

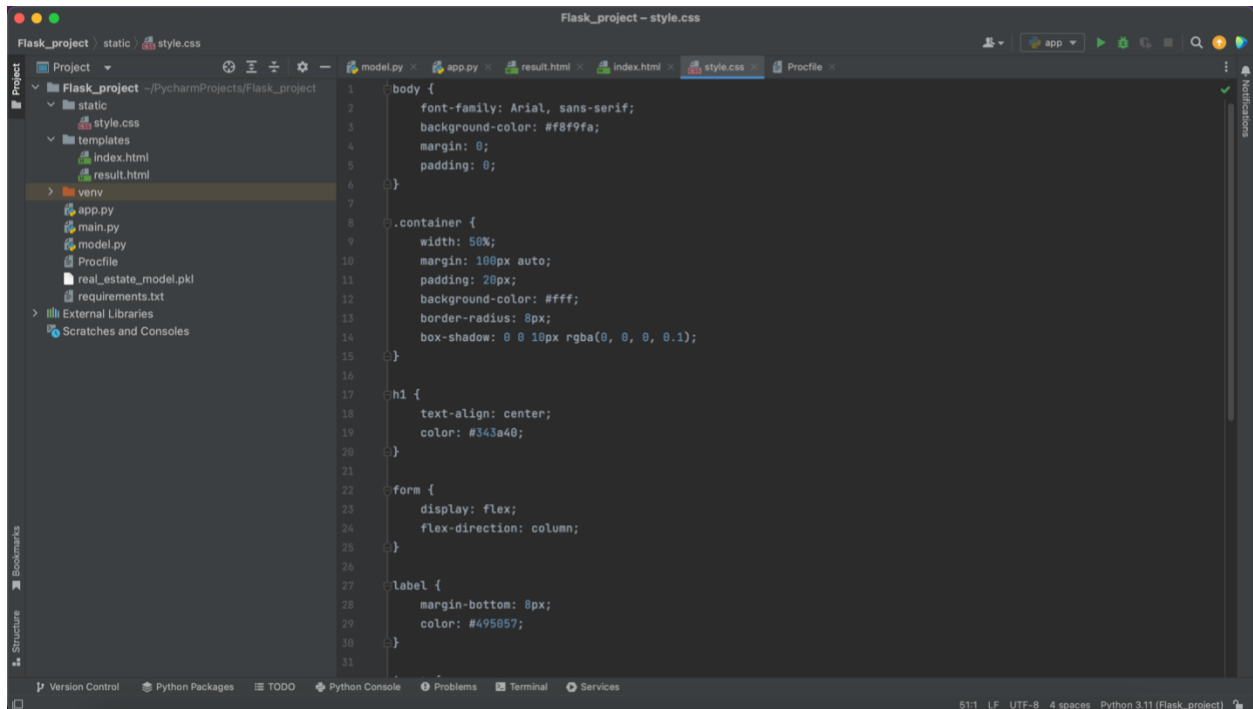
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
7   <title>Real Estate Price Prediction</title>
8 </head>
9 <body>
10  <div class="container">
11    <h1>Real Estate Price Prediction</h1>
12    <form action="/predict" method="post">
13      <label for="X2 house age">House Age:</label>
14      <input type="text" id="X2 house age" name="X2 house age" required><br>
15
16      <label for="X3 distance to the nearest MRT station">Distance to MRT:</label>
17      <input type="text" id="X3 distance to the nearest MRT station" name="X3 distance to the nearest MRT station"
18
19      <label for="X4 number of convenience stores">Number of Convenience Stores:</label>
20      <input type="text" id="X4 number of convenience stores" name="X4 number of convenience stores" required><br>
21
22      <label for="X5 latitude">Latitude:</label>
23      <input type="text" id="X5 latitude" name="X5 latitude" required><br>
24
25      <label for="X6 longitude">Longitude:</label>
26      <input type="text" id="X6 longitude" name="X6 longitude" required><br>
27
28      <button type="submit">Predict</button>
29    </form>
30  </div>
31 </body>
32 </html>
33
```

### 4-result.html

```
Flask_project templates result.html
Project
  Flask_project - PycharmProjects/Flask_project
    static
    templates
      index.html
      result.html
    venv
      app.py
      main.py
      model.py
      Profile
      real_estate_model.pkl
      requirements.txt
    External Libraries
    Scratches and Consoles










1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
7   <title>Prediction Result</title>
8 </head>
9 <body>
10  <div class="container">
11    <h1>Prediction Result</h1>
12    <p>The predicted house price per unit area is: {{ prediction }}</p>
13    <a href="/">Go back</a>
14  </div>
15 </body>
16 </html>
17
```

## 5-style.css

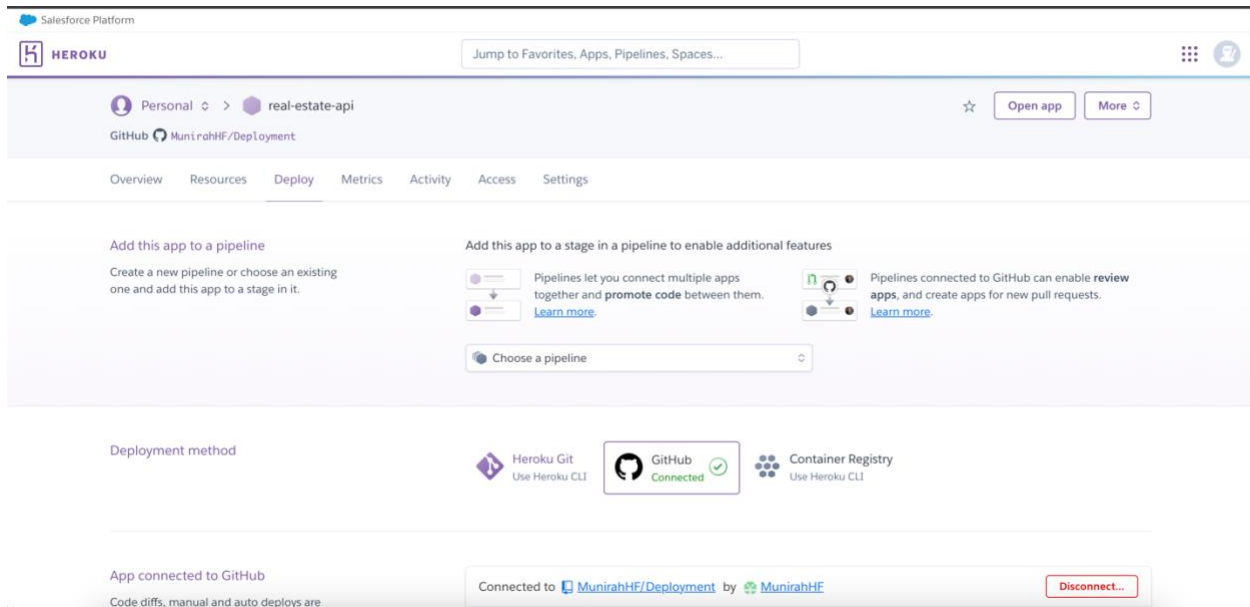


## 6-Deployment using Heroku

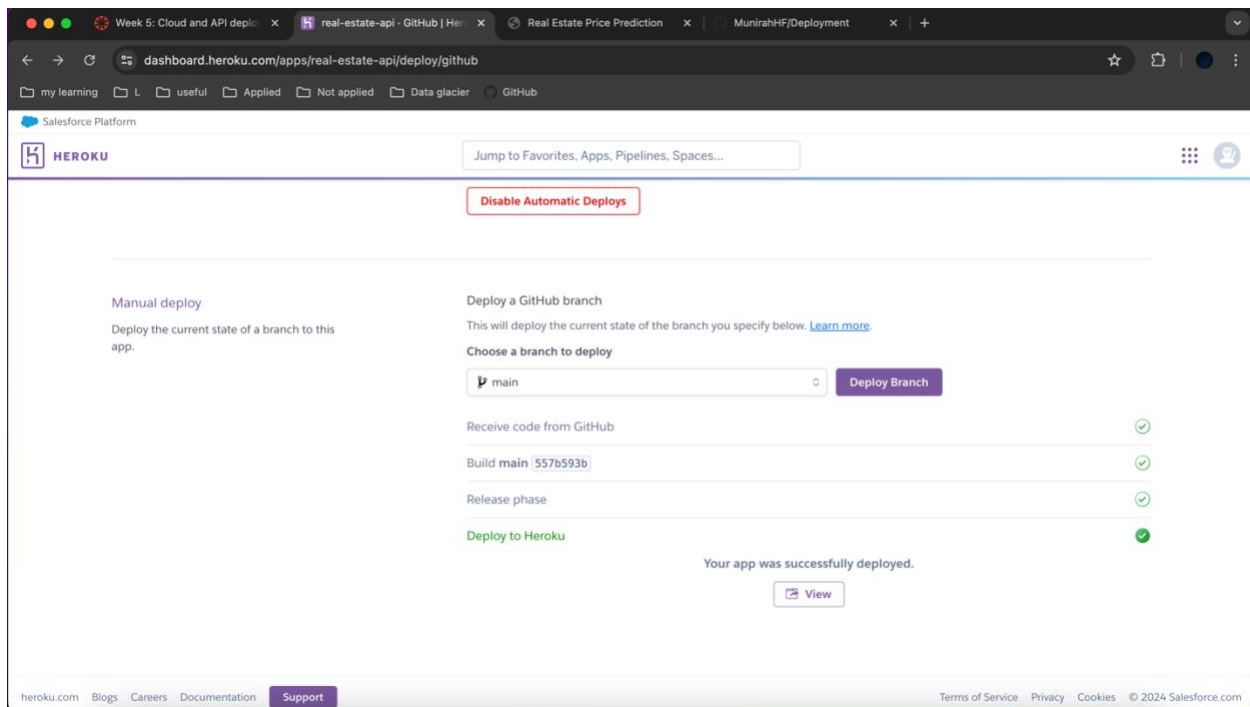
### Preparing the repository

 MunirahHF	Add files via upload	5e48488 · 3 minutes ago	🕒 2 Commits
 static	Add files via upload		3 hours ago
 templates	Add files via upload		3 hours ago
 Procfile	Add files via upload		3 minutes ago
 app.py	Add files via upload		3 hours ago
 main.py	Add files via upload		3 hours ago
 model.py	Add files via upload		3 hours ago
 real_estate_model.pkl	Add files via upload		3 hours ago
 requirements.txt	Add files via upload		3 hours ago

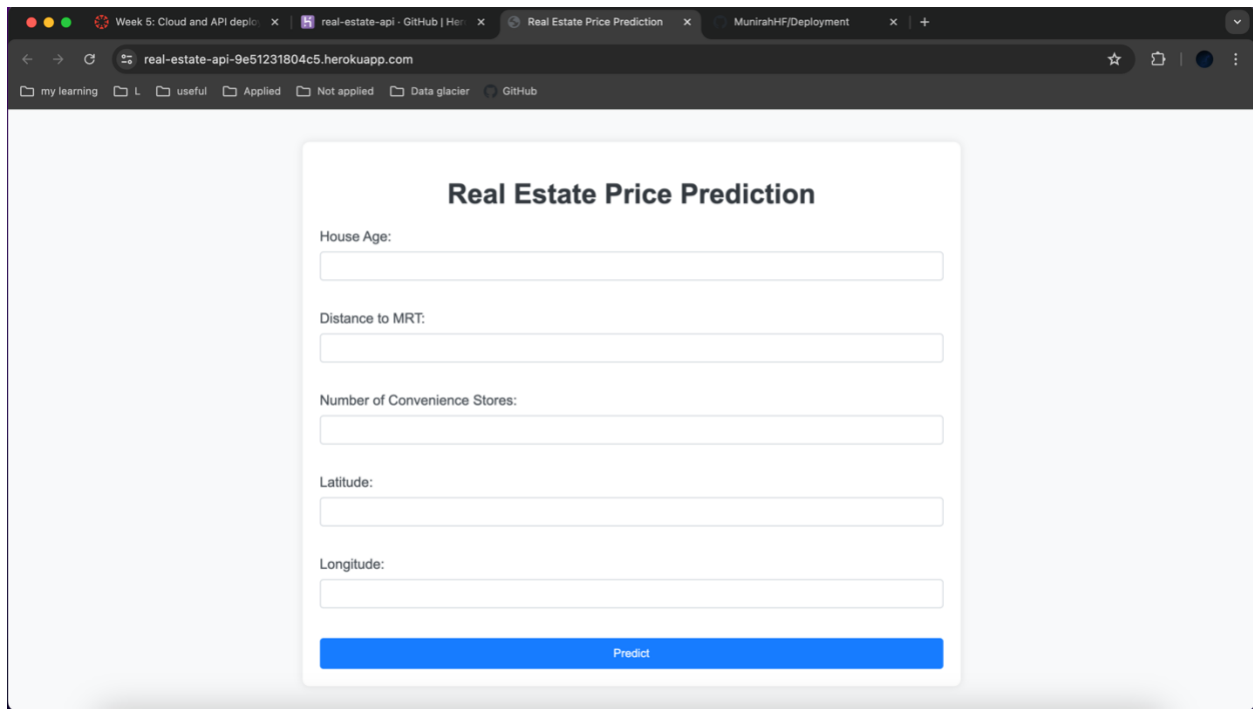
Connect to GitHub repository where code is uploaded.



Deploy branch



<https://real-estate-api-9e51231804c5.herokuapp.com/>



The screenshot shows a web browser window with the URL `real-estate-api-9e51231804c5.herokuapp.com`. The browser's address bar and tabs are visible at the top. The main content is a white form titled "Real Estate Price Prediction". The form contains five input fields, each with a label above it: "House Age:", "Distance to MRT:", "Number of Convenience Stores:", "Latitude:", and "Longitude:". Below these fields is a blue button labeled "Predict".

**Real Estate Price Prediction**

House Age:

Distance to MRT:

Number of Convenience Stores:

Latitude:

Longitude: