



## **Week 5: CLOUD AND API DEPLOYMENT**

**Name: Zain Ul Haq**

**Batch Code: LISUM07**

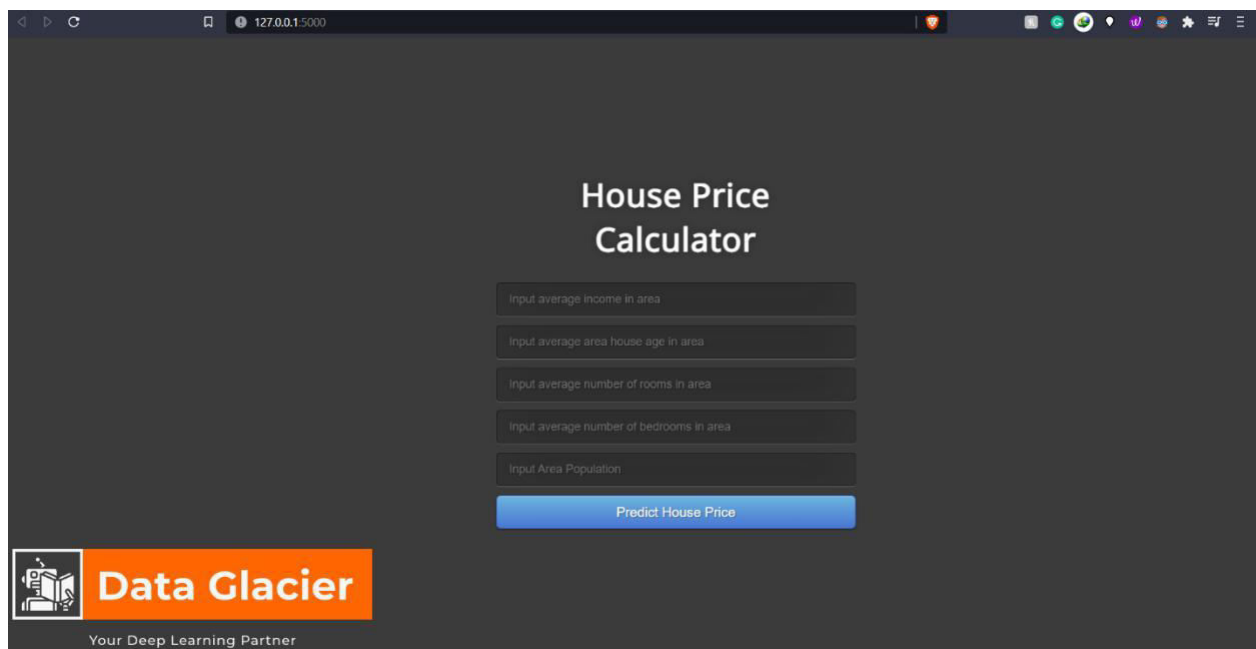
**Submission Data: 04/02/2021**

**Submitted to: Data Glacier**

# Introduction

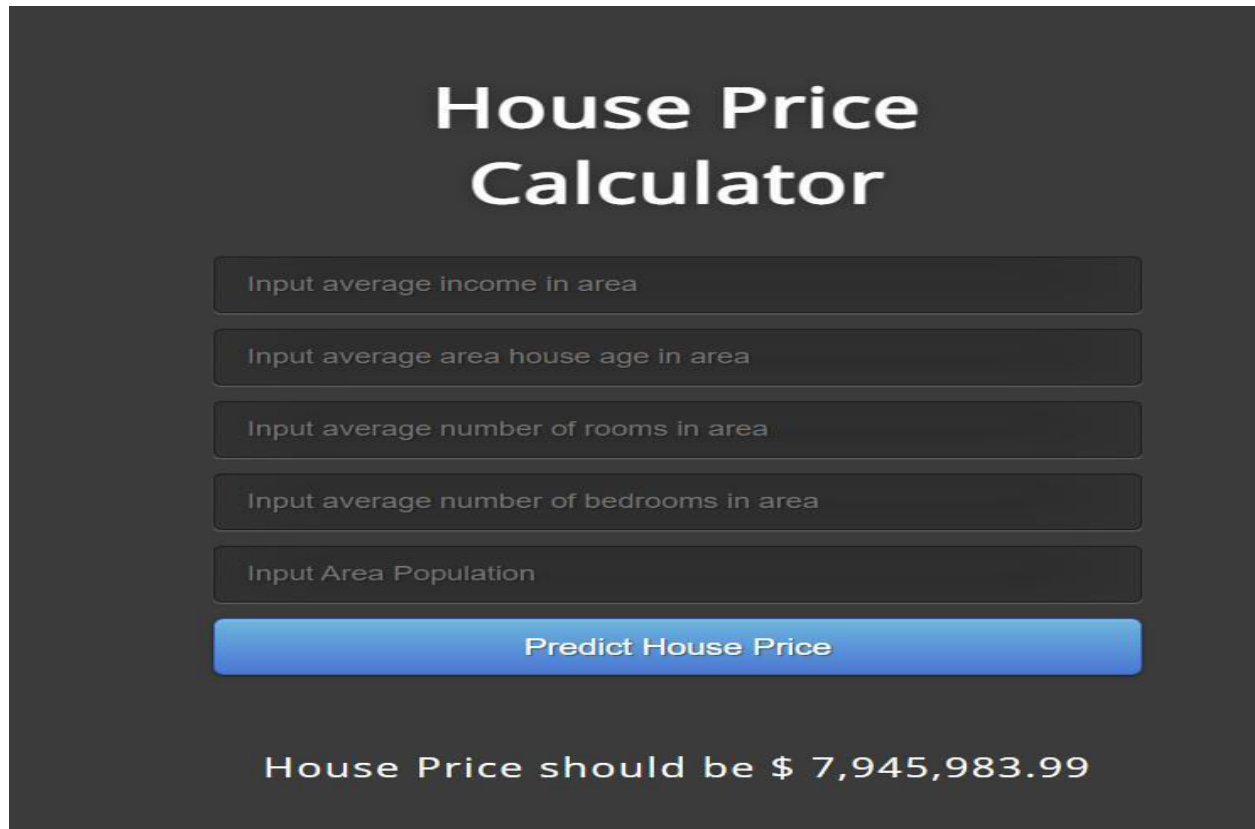
The Flask web had been developed to predict the house price based on different variables in the selected area from “USA Housing” [dataset](#) from Kaggle: Average Income, Average House Age, Average Number of Room, Average of Bedroom in Area, and Area Population. In the following sections, I will explain each of the project of the project separately in detail:

Here is the picture of the web app interface:



The screenshot shows a web application interface for a "House Price Calculator". The interface is dark-themed with a central form. The form contains five input fields for the following variables: "Input average income in area", "Input average area house age in area", "Input average number of rooms in area", "Input average number of bedrooms in area", and "Input Area Population". Below these fields is a blue button labeled "Predict House Price". In the bottom left corner, there is a logo for "Data Glacier" with the tagline "Your Deep Learning Partner". The browser's address bar at the top shows the URL "127.0.0.1:5000".

After a user put values and press the “Predict House Price” button, the answers will come immediately:



The image shows a web-based calculator interface with a dark gray background. At the top, the title "House Price Calculator" is displayed in a large, white, sans-serif font. Below the title, there are five input fields, each with a light gray border and a light gray placeholder text. The inputs are: "Input average income in area", "Input average area house age in area", "Input average number of rooms in area", "Input average number of bedrooms in area", and "Input Area Population". Below these inputs is a prominent blue button with a gradient and rounded corners, labeled "Predict House Price" in white text. At the bottom of the interface, the result is displayed in a white, monospace-style font: "House Price should be \$ 7,945,983.99".

# House Price Calculator

Input average income in area

Input average area house age in area

Input average number of rooms in area

Input average number of bedrooms in area

Input Area Population

Predict House Price

House Price should be \$ 7,945,983.99

## Dataset

The dataset that I used it is “USA housing” Dataset. The below picture illustrates all features of this dataset Since house price is a continues variable, this is a regression problem. The data contains the following columns:

- ‘Avg. Area Income’: Avg. Income of residents of the city house is located in.
- ‘Avg. Area House Age’: Avg Age of Houses in same city
- ‘Avg. Area Number of Rooms’: Avg Number of Rooms for Houses in same city
- ‘Avg. Area Number of Bedrooms’: Avg Number of Bedrooms for Houses in same city
- ‘Area Population’: Area population
- ‘Price’: Price that the house sold at
- ‘Address’: Address for the house

## MODEL

I have deployed a model to predict house price based on 5 variables:

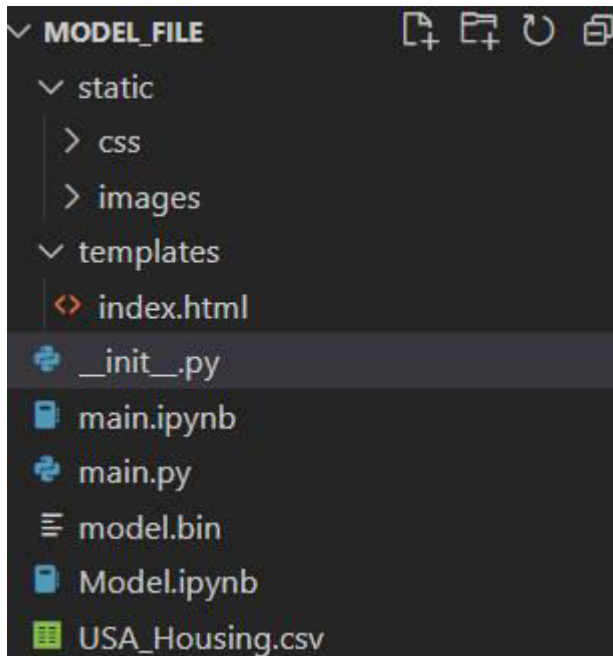
- ‘Avg. Area Income’
- ‘Avg. Area House Age’
- ‘Avg. Area Number of Rooms’
- ‘Avg. Area Number of Bedrooms’
- ‘Area Population’

Linear Regression is the method in this project to predict the target and train the model with whole dataset. The code has been included in my Git Folder.

The code is written in Jupyter Notebook, and my model is stored as “model.bin” for deploying in Flask

# FLAKS DEPLOYMENT


After install Flask, we will need some other important material for the deployment such as .html for web design, style with .css, image, model.bin to store my model, the main code .py to launch it. The following image is the summary of all necessary file need for the Flask deployment:



**Model. Bin:** is the model for our deployment. We could store as .pkl as well

**Style.css** This file will help create a beautiful view for our app. Here is the snapshot of this file:

```

static > css > # style.css >  body
1  @import url(https://fonts.googleapis.com/css?family=Open+Sans);
2  .btn { display: inline-block; *display: inline; *zoom: 1; padding: 4px 10px 4px; margin-bottom: 0; font-size: 13px; line-height: 18px; co
3  .btn:hover, .btn:active, .btn.active, .btn.disabled, .btn[disabled] { background-color: #e6e6e6; }
4  .btn-large { padding: 9px 14px; font-size: 15px; line-height: normal; -webkit-border-radius: 5px; -moz-border-radius: 5px; border-radius:
5  .btn:hover { color: #333333; text-decoration: none; background-color: #e6e6e6; background-position: 0 -15px; -webkit-transition: back
6  .btn-primary, .btn-primary:hover { text-shadow: 0 -1px 0 rgba(0, 0, 0, 0.25); color: #ffffff; }
7  .btn-primary.active { color: rgba(255, 255, 255, 0.75); }
8  .btn-primary { background-color: #4a77d4; background-image: -moz-linear-gradient(top, #6eb6de, #4a77d4); background-image: -ms-line
9  .btn-primary:hover, .btn-primary:active, .btn-primary.active, .btn-primary.disabled, .btn-primary[disabled] { filter: none; background-co
10 .btn-block { width: 100%; display: block; }
11
12 * { -webkit-box-sizing: border-box; -moz-box-sizing: border-box; -ms-box-sizing: border-box; -o-box-sizing: border-box; box-sizing: border-box
13
14 html { width: 100%; height: 100%; overflow: hidden; }
15
16 body {
17   width: 100%;
18   height: 100%;
19   font-family: 'Open Sans', sans-serif;
20   color: #fff;
21   font-size: 18px;
22   text-align: center;
23   letter-spacing: 1.2px;
24   background: #3B3B3B !important;
25   filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#3E1D6D', endColorstr='#092756', GradientType=1 );
26
27 }
28 .login {
29   position: absolute;
30   top: 40%;
31   left: 50%;
32   margin: -150px 0 0 -150px;
33   width: 400px;
34   height: 400px;
35 }
36
37 .login h1 { color: #fff; text-shadow: 0 0 10px rgba(0,0,0,0.3); letter-spacing: 1px; text-align: center; }
38

```

```

.login h1 { color: #fff; text-shadow: 0 0 10px rgba(0,0,0,0.3); letter-spacing: 1px; text-align: center; }

input {
  width: 100%;
  margin-bottom: 10px;
  background: rgba(0,0,0,0.3);
  border: none;
  outline: none;
  padding: 10px;
  font-size: 13px;
  color: #fff;
  text-shadow: 1px 1px 1px rgba(0,0,0,0.3);
  border: 1px solid rgba(0,0,0,0.3);
  border-radius: 4px;
  box-shadow: inset 0 -5px 45px rgba(100,100,100,0.2), 0 1px 1px rgba(255,255,255,0.2);
  -webkit-transition: box-shadow .5s ease;
  -moz-transition: box-shadow .5s ease;
  -o-transition: box-shadow .5s ease;
  -ms-transition: box-shadow .5s ease;
  transition: box-shadow .5s ease;
}
input:focus { box-shadow: inset 0 -5px 45px rgba(100,100,100,0.4), 0 1px 1px rgba(255,255,255,0.2); }

```

**Index.html:** This is the main view page that contains all static elements so the client can see in UI:

```

templates > < index.html > ...
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>House Price Calculator</title>
6      <link href="https://fonts.googleapis.com/css?family=Pacifico" rel="stylesheet" type="text/css">
7      <link href="https://fonts.googleapis.com/css?family=Arimo" rel="stylesheet" type="text/css">
8      <link href="https://fonts.googleapis.com/css?family=Hind:300" rel="stylesheet" type="text/css">
9      <link href="https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300" rel="stylesheet" type="text/css">
10     <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
11
12 </head>
13
14 <body>
15     <div class="login">
16         <h1>House Price Calculator</h1>
17
18         <!-- Main Input For Receiving Query to our ML -->
19         <form action="{{ url_for('predict') }}" method="post">
20             <input type="text" name="Avg. Area Income" placeholder=" Input average income in area" required="required" />
21             <input type="text" name="Avg. Area House Age" placeholder=" Input average area house age in area" required="required" />
22             <input type="text" name="Avg. Area Number of Rooms" placeholder=" Input average number of rooms in area" required="required" />
23             <input type="text" name="Avg. Area Number of Bedroom" placeholder=" Input average number of bedrooms in area" required="required" />
24             <input type="text" name="Area Population" placeholder=" Input Area Population" required="required" />
25
26             <button type="submit" class="btn btn-primary btn-block btn-large">Predict House Price </button>
27         </form>
28
29         <br>
30         <br>
31         {{ prediction_text }}
32     </div>
33
34     
35
36 </body>
37 </html>

```

**Main.py:** This is important file for this project

```

# In[1]:
from flask import Flask, request, jsonify, render_template
import pickle
import numpy as np
import sklearn
Run Cell | Run Above | Debug Cell

# In[2]:
#change the save path location for model.bin depends on your computer
save_path = r"C:\Users\nguye\Desktop\Data Science Stuff\Projects\Python\Flask\model_file\model.bin"
# # Deployment of Flask
Run Cell | Run Above | Debug Cell

# In[3]:
app = Flask(__name__)
with open(save_path, 'rb') as f:
    model = pickle.load(f)
Run Cell | Run Above | Debug Cell

# In[4]:
@app.route("/")
def home():
    return render_template("index.html")
Run Cell | Run Above | Debug Cell

# In[5]:
@app.route('/', methods=['POST'])
def predict():
    int_features = [int(x) for x in request.form.values()]
    final_features = [np.array(int_features)]
    prediction = model.predict(final_features)
    output = round(prediction[0], 2)
    return render_template('index.html', prediction_text='House Price should be $ {:.,}'.format(output))
Run Cell | Run Above | Debug Cell

# In[ ]:
if __name__ == '__main__':
    app.run(port=5000, debug=True)

```

# Test Running App

For running application, we will run the main.py, which is the python code, and find the address of our location like : “127.0.0.1:5000/”, or we can customize our address as well.



```
1 #!/usr/bin/env python
2 # coding: utf-8
3 # -*- coding: utf-8 -*-
4 from flask import Flask, request, jsonify, render_template
5 import pickle
6 import numpy as np
7 import os
8 # Datasets
9 # Imports the new path location for model file stored on your computer
10 save_path = r"C:\Users\user\Desktop\data science stuff\Projects\Python\acknowledg\file\model.pkl"
11 # Requirement of flask
12 from flask import Flask
13 app = Flask(__name__)
14 with open(save_path, "rb") as f:
15     model = pickle.load(f)
16 # Datasets
17 @app.route("/")
18 def home():
19     return render_template("index.html")
20 @app.route("/predict", methods=['POST'])
21 def predict():
22     int_features = [int(x) for x in request.form.values()]
23     final_features = np.array(int_features)
24
25 * Environment: production
26 WARNING: This is a development server. Do not use it in a production deployment.
27 Use a production grade server instead.
28 * Debug mode: on
29 * Restarting with stat
30 C:\Users\user\AppData\Local\Programs\Python\Python311\Scripts\python.exe: UserWarning: Trying to unpickle estimator LinearRegression
31 version 0.28.2 when using version 1.0. This might lead to breaking code or invalid results, use at your own risk. For more info please refer to:
32 https://wiki.python.org/moin/PythonsFuture#Deprecation_warnings_about_unpicklable_classes:
33 warnings.warn(
34 * Debugger is active!
35 * Debugger PIN: 402-544-118
36 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

## HEROKU DEPLOYMENT

### Before Deployment

My app name is **main.py**, which will have this command line: web: gunicorn **main**:app. The app name have to match with this command line or it will cause errors. In addition, for the **requirement.txt**, we can run pip freeze command to create this file, but in my case, I choose pipreqs to create the requirement.txt.

Create new GitHub repository, and your repository will have these files:



**Main.py:** for app running

**Notebook.ipynb:** contain the model building process

main


1 branch

0 tags

Go to file








Add file

Code

 ZainUIHaq

Add files via upload

2a804e4 5 minutes ago 12 commits

 LICENSE	Initial commit	42 minutes ago
 Procfile	Add files via upload	5 minutes ago
 README.md	Add files via upload	5 minutes ago
 USA_Housing.csv	Add files via upload	18 minutes ago
 main.py	Add files via upload	5 minutes ago
 model_app.ipynb	Add files via upload	5 minutes ago
 requirements.txt	Add files via upload	5 minutes ago

**Requirement.txt:** can be generated by pip freeze or pipreqs command line

**Procfile:** we must create it with the command line mentioned above

**Templates and statics** folders to contain layout, designs of our models

**Model\_files:** contain the model.bin

