Week 4: Model Deployment with Flask

Name: Bao Khanh Nguyen

Batch Code: LISUM04

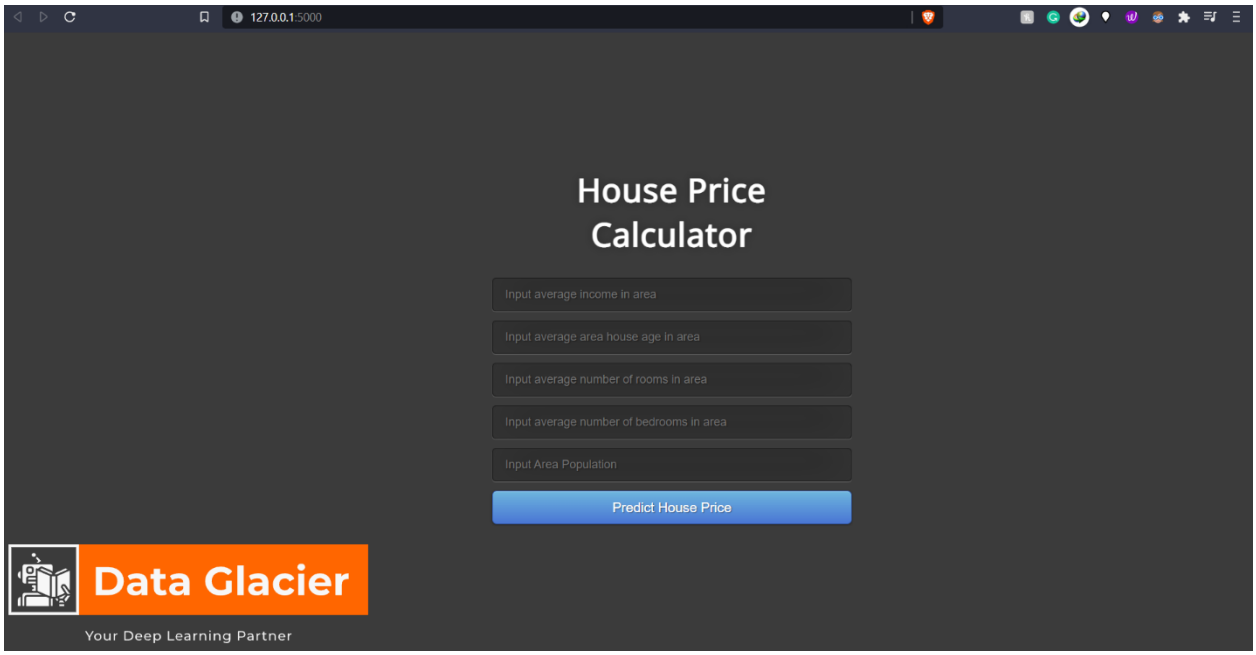
Submission Data: 10/17/2021

Submitted to: Data Glacier

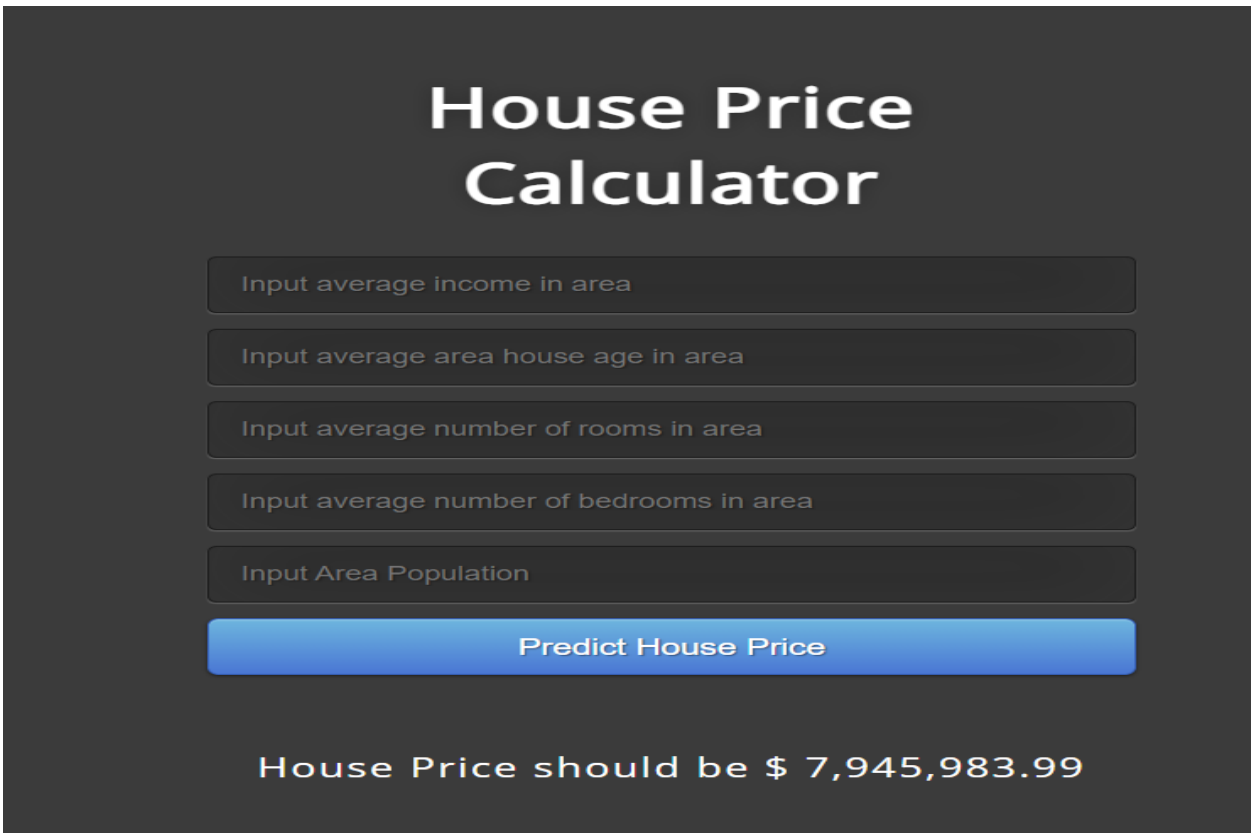
Introduction

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:



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



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. Here is the model valuation based on MAE, MSE, RMSE, R2square and Cross Validation

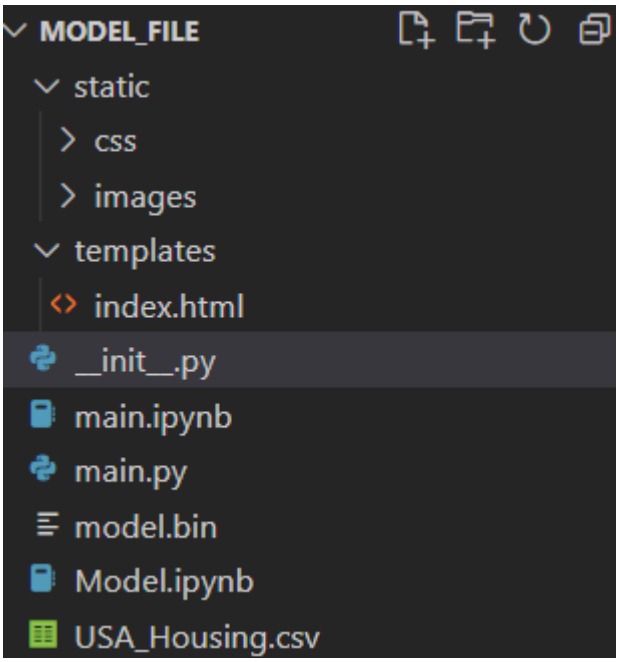
```
# Have * before evaluate func or will getting error
results_df = pd.DataFrame(data=[["Linear Regression", *evaluate(y_test, test_pred) , cross_val(LinearRegression())]],
                          columns=['Model', 'MAE', 'MSE', 'RMSE', 'R2 Square', "Cross Validation"])
results_df
```

	Model	MAE	MSE	RMSE	R2 Square	Cross Validation
0	Linear Regression	81135.57	10068422551.40	100341.53	0.91	0.92

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

```
@import url(https://fonts.googleapis.com/css?family=Open+Sans);
.btn { display: inline-block; *display: inline; *zoom: 1; padding: 4px 10px 4px; margin-bottom: 0; font-size: 13px; line-height: 18px; co
.btn:hover, .btn:active, .btn.active, .btn.disabled, .btn[disabled] { background-color: #e6e6e6; }
.btn-large { padding: 9px 14px; font-size: 15px; line-height: normal; -webkit-border-radius: 5px; -moz-border-radius: 5px; border-radius:
.btn:hover { color: #333333; text-decoration: none; background-color: #e6e6e6; background-position: 0 -15px; -webkit-transition: back
.btn-primary, .btn-primary:hover { text-shadow: 0 -1px 0 rgba(0, 0, 0, 0.25); color: #ffffff; }
.btn-primary.active { color: rgba(255, 255, 255, 0.75); }
.btn-primary { background-color: #4a77d4; background-image: -moz-linear-gradient(top, #6eb6de, #4a77d4); background-image: -ms-lin
.btn-primary:hover, .btn-primary:active, .btn-primary.active, .btn-primary.disabled, .btn-primary[disabled] { filter: none; background-co
.btn-block { width: 100%; display:block; }

* { -webkit-box-sizing:border-box; -moz-box-sizing:border-box; -ms-box-sizing:border-box; -o-box-sizing:border-box; box-sizing:border-box

html { width: 100%; height:100%; overflow:hidden; }

body {
    width: 100%;
    height:100%;
    font-family: 'Open Sans', sans-serif;
    color: #fff;
    font-size: 18px;
    text-align:center;
    letter-spacing:1.2px;
    background: #3B3B3B !important;
    filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#3E1D6D', endColorstr='#092756',GradientType=1 );

}
.login {
    position: absolute;
    top: 40%;
    left: 50%;
    margin: -150px 0 0 -150px;
    width:400px;
    height:400px;
}

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

```
.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: The is the main view page that contains all static elements so the client can see in UI:

templates > index.html > ...

```html
<!DOCTYPE html>
<html >
<head>
  <meta charset="UTF-8">
  <title>House Price Calculator</title>
  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>
<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">

</head>

<body>
 <div class="login">
  <h1>House Price Calculator</h1>

    <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict')}}"method="post">
    <input type="text" name="Avg. Area Income" placeholder=" Input average income in area" required="required" />
    <input type="text" name="Avg. Area House Age" placeholder=" Input average area house age in area" required="required" />
    <input type="text" name="Avg. Area Number of Rooms" placeholder=" Input average number of rooms in area" required="required" />
    <input type="text" name="Avg. Area Number of Bedroom" placeholder=" Input average number of bedrooms in area" required="required" />
    <input type="text" name="Area Population" placeholder=" Input Area Population" required="required" />

      <button type="submit" class="btn btn-primary btn-block btn-large">Predict House Price </button>
    </form>

   <br>
   <br>
   {{ prediction_text }}

  </div>
  <img src="/static/images/Original.svg" style="width: 400px;position: absolute;bottom: 10px;left: 10px;" alt="Company Logo"/>

 </body>
 </html>
```

Main.py: This is important file for this project

```python
# 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.

```
main.py
main.py > ...
#!/usr/bin/env python
# coding: utf-8
In[1]:
from flask import Flask, request, jsonify, render_template
import pickle
import numpy as np
import sklearn
# 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
# In[3]:
app = Flask(__name__)
with open(save_path,'rb') as f:
    model = pickle.load(f)
# In[4]:
@app.route("/")
def home():
    return render_template("index.html")
# In[5]:
@app.route('/',methods=['POST'])
def predict():
    int_features = [ int (x) for x in request.form.values()]
    final_features = [np.array(int_features)]
```

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE                    Python Debug Console
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
C:\Users\nguye\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\base.py:324: UserWarning: Trying to unpickle estimator LinearRegression f
0.24.2 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:
https://scikit-learn.org/stable/modules/model_persistence.html#security-maintainability-limitations
  warnings.warn(
 * Debugger is active!
 * Debugger PIN: 402-344-118
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```