Deployment on flask

Name: Pok Hei Tang
Batch code: LISUM23: 30
Submission date: 28-Jul-2023
Submitted to: https://github.com/keithonpy/data_glacier_log/tree/main/week_04/scripts
Snapshot:

1. Test the flask

```python
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')

def index():
    return render_template('index.html')


if __name__ == "__main__":
    app.run(port=5000, debug= True)
```

2. Index.html file during the test

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>My Website</title>
    <link rel="stylesheet" href="./style.css">
    <link rel="icon" href="./favicon.ico" type="image/x-icon">
  </head>
  <body>
    <main>
        <h1>Welcome to My Website</h1>
    </main>
  <script src="index.js"></script>
  </body>
</html>
```

3. Train the random forest model for Iowa house price prediction

```python
import pandas as pd
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
import pickle

# Path of the file to read
iowa_file_path = 'train.csv'

home_data = pd.read_csv(iowa_file_path)
# Create target object and call it y
y = home_data.SalePrice
# Create X
features = ['LotArea', 'YearBuilt', '1stFlrSF', '2ndFlrSF', 'FullBath', 'TotRmsAbvGrd']
X = home_data[features]

# Split into validation and training data
train_X, val_X, train_y, val_y = train_test_split(X, y, random_state=1)



# Define the model. Set random_state to 1
rf_model = RandomForestRegressor(random_state=1)

# fit your model
rf_model.fit(train_X, train_y)


# save the model to pickel
with open('rf.pkl', 'wb') as f:
    pickle.dump(rf_model, f)
    print("Pickling completed")
```

4. Develop the web app

```python
import numpy as np
import pickle

from flask import Flask, request, render_template

app = Flask(__name__)
model = pickle.load(open("model/rf.pkl", "rb"))

@app.route('/')

def index():
    return render_template('index.html')


@app.route('/predict', 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='The selling price of the house should be $ {}'.format(output))



if __name__ == "__main__":
    app.run(port=5000, debug= True)
```

5. update the index.html for the form

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>My Website</title>
    <link rel="stylesheet" href="./style.css">
    <link rel="icon" href="./favicon.ico" type="image/x-icon">
  </head>
  <body>
    <main>
        <h1>Welcome to My Website</h1>


        <form action="{{url_for('predict')}}" method = "post">
          <label for="lotarea">Lot size (in sq ft): </label>
          <input type="number" name="lotarea"  required>
          <br>
          <label for="Year">Construction Year: </label>
          <input type="text" name="Year"  required>
          <br>
          <label for="1stSF">1st floor (in sq ft): </label>
          <input type="number" name="email"  required>
          <br>
          <label for="2ndSF">2nd floor (in sq ft): </label>
          <input type="number" name="2ndSF" >
          <br>
          <label for="Bath">Number of bathrooms: </label>
          <input type="number" name="Bath"  required>
          <br>
          <label for="Totrms">Total rooms (not include bathroom): </label>
          <input type="number" name="Totrms"  required>
          <br>
          <input type="submit" value="Predict!">
        </form>
        <br>
        <br>
        {{ prediction_text }}

    </main>
  <script src="index.js"></script>
  </body>
</html>
```

6. improve the html file with base file

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>My Website</title>
    <link rel="stylesheet" href={{ url_for('static', filename='css/main.css')}}>

    {% block head %}{% endblock %}
  </head>
  <body>
    {% block body %}{% endblock %}

  </body>
</html>
```

7. add css for styling

```css
@import url('https://fonts.googleapis.com/css?family=Poppins:400,500,600,700&display=swap');
*{
  margin: 0;
  padding: 0;
  outline: none;
  box-sizing: border-box;
  font-family: 'Poppins', sans-serif;
}
body{
  display: flex;
  align-items: center;
  justify-content: center;
  min-height: 100vh;
  padding: 10px;
  font-family: 'Poppins', sans-serif;
  background: linear-gradient(115deg, #56d8e4 10%, #9f01ea 90%);
}
.container{
  max-width: 800px;
  background: #fff;
  width: 800px;
  padding: 25px 40px 10px 40px;
  box-shadow: 0px 0px 10px rgba(0,0,0,0.1);
}
.container .text{
  text-align: center;
  font-size: 41px;
  font-weight: 600;
  font-family: 'Poppins', sans-serif;
  background: -webkit-linear-gradient(right, #56d8e4, #9f01ea, #56d8e4, #9f01ea);
  -webkit-background-clip: text;
  -webkit-text-fill-color: transparent;
}
.container form{
  padding: 30px 0 0 0;
}
.container form .form-row{
  display: flex;
  margin: 32px 0;
}
form .form-row .input-data{
  width: 100%;
  height: 40px;
  margin: 0 20px;
  position: relative;
}
form .form-row .textarea{
  height: 70px;
}
```