Flask Deployment Documentation

Igwebuike Eze

30/04/2024

1 Introduction

This document describes the deployment of a Flask-based web application for a machine learning model. This project has five major code snipets:

- HRRecruitment.py - This contains Machine Learning model code to predict employee salaries based on trainig data in 'HRRecruitmentData.csv' file.
- app.py - This contains flask API that receives employee details through API calls, computes the predited value based on our model and returns it.
- request.py - This uses requests module to call APIs already defined in app.py and dispalys the returned value.
- idex.html file is used to design the output of the form.
- style.css file is used to apply aesthetics to the form.

The following sections provide a detailed step-by-step guide.

2 Step 1: Select a Toy Dataset

For this deployment, we used a simple toy dataset for regression analysis. Below is an example of the dataset used:

EXPERIENCE	ASSESSMENT_SCORE	INTERVIEW_SCORE	EXPECTED_SALARY
one	8	9	40000
two	8	6	50000
three	6	7	60000
four	10	9	70000
five	9	6	80000
six	7	8	60000
seven	5	7	80000
eight	7	8	75000
nine	6	6	45000
ten	5	7	40000
eleven	5	8	80000
twelve	6	6	75000

3 Step 2: Train and Save the Model

I trained a simple linear regression model and saved it using the Python ‘pickle‘ module. The code snippet below demonstrates how the model was trained and saved:

```
import pandas as pd
from sklearn.linear_model import LinearRegression
import pickle

# Importing the libraries
import pandas as pd
import pickle

dataset = pd.read_csv('HR_Recruitment_Data.csv')


X = dataset.iloc[:, :3]

#Converting words to integer values
def convert_to_int(word):
    word_dict = {'one':1, 'two':2, 'three':3, 'four':4, 'five':5, 'six':6,
        'seven':7, 'eight':8,
                'nine':9, 'ten':10, 'eleven':11, 'twelve':12, 'zero':0}
    return word_dict[word]

X['experience'] = X['experience'].apply(lambda x : convert_to_int(x))

y = dataset.iloc[:, -1]

#Splitting Training and Test Set
#Since we have a very small dataset, we will train our model with all
   availabe data.

from sklearn.linear_model import LinearRegression
regressor = LinearRegression()

#Fitting model with trainig data
regressor.fit(X, y)

# Saving model to disk
pickle.dump(regressor, open('HR_Recruitment.pkl','wb'))

# Loading model to compare the results
HR_Recruitment = pickle.load(open('HR_Recruitment.pkl','rb'))
```

4 Step 3: Deploy the Model on Flask

Next, I created a simple Flask application to deploy the model. The code snippet below shows the main structure of the Flask app:

```
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle
```

```
app = Flask(__name__)
HR_Recruitment = pickle.load(open('HR_Recruitment.pkl', 'rb'))

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict',methods=['POST'])
def predict():
    '''
    For rendering results on HTML GUI
    '''
    int_features = [int(x) for x in request.form.values()]
    final_features = [np.array(int_features)]
    prediction = HR_Recruitment.predict(final_features)

    output = round(prediction[0], 2)

    return render_template('index.html', prediction_text='Employee␣
       Starting␣Salary␣Forcast␣$␣{}'.format(output))

@app.route('/predict_api',methods=['POST'])
def predict_api():
    '''
    For direct API calls trought request
    '''
    data = request.get_json(force=True)
    prediction = HR_Recruitment.predict([np.array(list(data.values()))])

    output = prediction[0]
    return jsonify(output)

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

5 Step 4: Create a Web Form

This section describes the document creation process. The code snippet below demonstrates how the form was designed:

```
<!DOCTYPE html>
<html lang="en">
<!-- Modified to improve formatting -->
<head>
  <meta charset="UTF-8">
  <title>ML API</title>
  <!-- Include custom CSS -->
  <link rel="stylesheet" href="{{␣url_for('static',␣filename='css/style.
     css')␣}}">

  <!-- Google Fonts -->
  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='
     stylesheet' type='text/css'>
```

```
  <link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed
     :300' rel='stylesheet' type='text/css'>
</head>

<body>
 <!-- Main login form -->
 <div class="login">
    <h1>Salary Prediction</h1>

    <!-- Form for input and predict button -->
    <form action="{{ url_for('predict') }}" method="post">
        <input type="text" name="xperience" placeholder="Enter Experience"
            required="required" />
        <input type="text" name="assessment_score" placeholder="Enter
           Assessment Score" required="required" />
        <input type="text" name="interview_score" placeholder="Enter
           Interview Score" required="required" />

        <button type="submit">Predict</button>
    </form>

    <!-- Prediction text area -->
    <div class="prediction-text">
       {{ prediction_text }}
    </div>
 </div>

</body>
</html>
```

To beautify the form , I applied style to it.

```
/* style.css */
body {
    background-color: #f5f5f5;  /* Soft background color */
    font-family: 'Open Sans Condensed', sans-serif;  /* Font for the page
       */
}

.login {
    max-width: 300px;  /* Set a fixed width for the form */
    margin: 50px auto;  /* Center horizontally with margin */
    padding: 20px;  /* Padding for spacing within the form */
    border: 1px solid #ccc;  /* Border for the form */
    background: #fff;  /* White background for form */
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);  /* Soft shadow */
}

h1 {
    font-family: 'Pacifico', cursive;  /* Title font */
    font-size: 24px;  /* Title size */
    text-align: center;  /* Center align title */
}

input {
```

```
    width: 100%;  /* Ensure inputs take the full width */
    padding: 10px;  /* Padding for comfort */
    margin-bottom: 10px;  /* Space between inputs */
    border: 1px solid #ccc;  /* Border for inputs */
}

button {
    width: 100%;  /* Button should take full width */
    padding: 10px;  /* Padding for comfort */
    background-color: #3498db;  /* Blue background for button */
    color: white;  /* White text */
    border: none;  /* No border */
    cursor: pointer;  /* Change cursor on hover */
}

button:hover {
    background-color: #2980b9;  /* Darker blue on hover */
}
```

6 Step 4: Running the Program

This section describes the process of executing the project.

First navigate to the project home directory. Create the machine learning model by running the command - python HRRecruitment.py which creates a serialized version of our model into a file HRRecruitment.pkl.

Then execute python app.py using below command to start Flask API. By default, flask will run on port 5000. Navigate to URL http://localhost:5000 to view the homepage.

Enter valid numerical values in all 3 input boxes and hit Predict.