Flask is a powerful python microwebserver framework that allows us to build REST API based web-services quickly with minimum configuration hassle. Let’s dive into the code:

Apr 14, 2019

# Serve model as a flask application

import pickle

import numpy as np

from flask import **Flask, request**

model = None

app = Flask(\_\_name\_\_)

def load\_model():

global model

# model variable refers to the global variable

with open('iris\_trained\_model.pkl', 'rb') as f:

model = pickle.load(f)

@app.route('/')

def **home\_endpoint():**

return 'Hello World!'

@app.route('/predict', methods=['POST'])

def **get\_prediction():**

# Works only for a single sample

if request.method == 'POST':

# Get data posted as a json

data = request.get\_json()

# converts shape from (4,) to (1, 4)

data = np.array(data)[np.newaxis, :]

prediction = model.predict(data)

return str(prediction[0])

if \_\_name\_\_ == '\_\_main\_\_':

load\_model() # load model one time

app.run(host='0.0.0.0', port=80)

At this point, the web-service is ready to be run locally. Let’s test this.

Execute the command python app.py from the terminal.

Go to the browser 0.0.0.0:80 to get a message Hello World! displayed.

This corresponds to the home endpoint return message.

**NOTE**: A permission error may be received at this point. In this case, change the port number to 5000 in app.run() command in app.py. (Port 80 is a privileged port, so change it to some port that isn't, eg: 5000)

Next, let’s test if we can get predictions using this web-service using the following curl post request on the terminal:

curl -X POST \  
 0.0.0.0:80/predict \  
 -H 'Content-Type: application/json' \  
 -d '[5.9,3.0,5.1,1.8]'

The curl request posts one test sample [5.9,3.0,5.1,1.8] to our web-server and returns a single class label.