

Week 5 – API based deployment using Postman

Name: Archana Devi Ramesh

Batch code: LISUM16

Submission date: 5th January 2023

Submitted to: Data Glacier

Submission Link: <https://github.com/ArchanaDeviRamesh/Data-Glacier-Week5>

API based Deployment steps

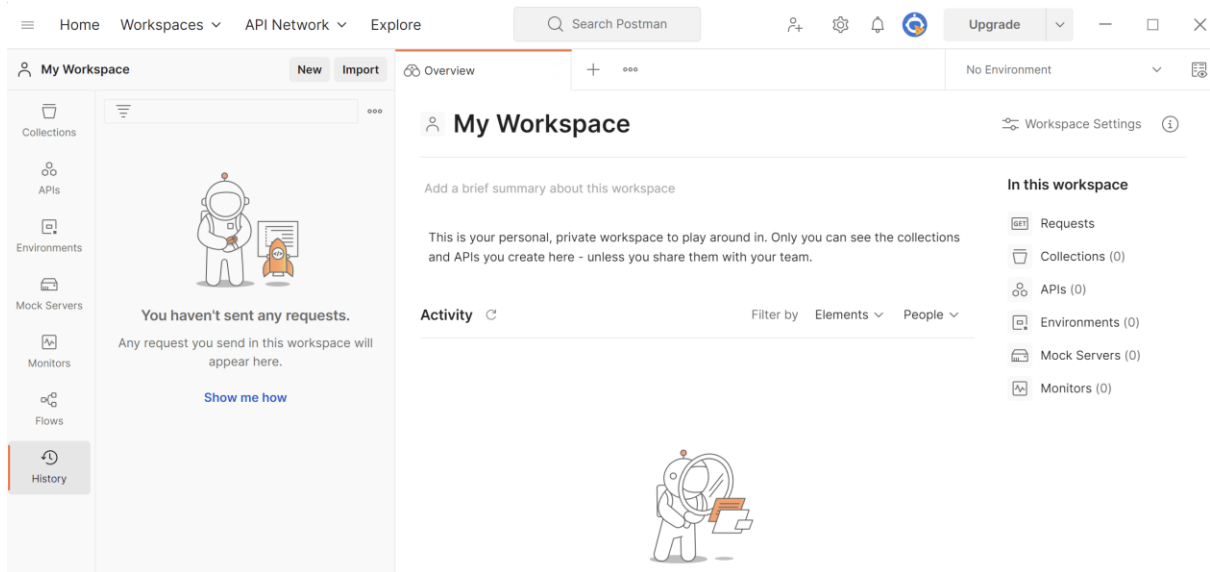
1. Since **Postman** accepts input in the form of **json** objects, the data returned from the flask app is converted to json, instead of returning a web page.

```
app_api.py •
app_api.py
1  import numpy as np
2  import pandas as pd
3  from flask import Flask, request, jsonify
4  import pickle
5
6  # Create flask app
7  app_api = Flask(__name__)
8  model = pickle.load(open("iris_model.pkl", "rb"))
9
10 @app_api.route("/", methods = ['GET', 'POST'])
11 def Home():
12     if request.method == 'GET':
13         data = 'Testing with API'
14         return jsonify({'data':data})
15
16 @app_api.route("/predict/", methods = ['GET', 'POST'])
17 def predict():
18     sepal_length = request.args.get("sepal_length")
19     sepal_width = request.args.get("sepal_width")
20     petal_length = request.args.get("petal_length")
21     petal_width = request.args.get("petal_width")
22     test_df = pd.DataFrame({'sepal.length': [sepal_length], 'sepal.width':[sepal_width], 'petal.length':[petal
23     prediction = model.predict(test_df)
24     return jsonify({'Iris Flower':str(prediction)})
25
26 if __name__ == "__main__":
27     app_api.run(debug=True)
```

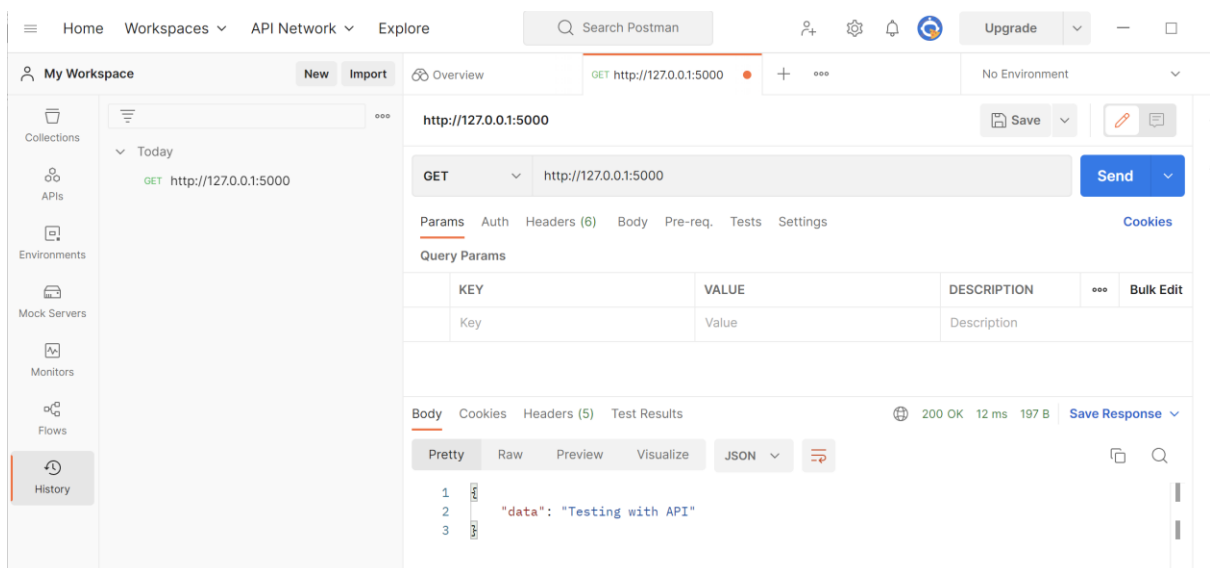
2. Run the above python file as `python app_api.py`, copy the URL <http://127.0.0.1:5000> from the terminal

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
(D:\mydocuments\CANADA\Internship\DataGlacier\Week4\myenv) D:\mydocuments\CANADA\Internship\DataGlacier\Week4>python a
pp_api.py
* Serving Flask app 'app_api'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 528-755-425
```

3. Open Postman and click on **Requests**, under **In this workspace**



4. Enter the URL in the tab near **GET** and click **Send**.
Under GET request, an object **data** was passed with the value **Testing with API** which is getting displayed under the results



5. For the prediction, enter the URL <http://127.0.0.1:5000/predict> and give key and value for the four features (**sepal_length**, **sepal_width**, **petal_length**, **petal_width**). Make sure to have the same key names given under predict method for these features.

```
@app_api.route("/predict/", methods = ['GET', 'POST'])
def predict():
    sepal_length = request.args.get("sepal_length")
    sepal_width = request.args.get("sepal_width")
    petal_length = request.args.get("petal_length")
    petal_width = request.args.get("petal_width")
```

6. After entering the key and value for all four features, click Send. The prediction will be displayed in the results section

The screenshot shows the Postman interface with a GET request to `http://127.0.0.1:5000/predict/?sepal_length=5.1&sepal_width=3.5&petal_length=1.5&petal_width=0.2`. The parameters are listed in a table:

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> sepal_length	5.1	
<input checked="" type="checkbox"/> sepal_width	3.5	
<input checked="" type="checkbox"/> petal_length	1.5	
<input checked="" type="checkbox"/> petal_width	0.2	

The response body is shown in JSON format:

```
{
  "Iris Flower": "Virginica"
}
```

7. If any of the parameters is not passed, it will throw an error as the model expects four parameters

The screenshot shows the Postman interface with a GET request to `http://127.0.0.1:5000/predict/?sepal_length=5.1&sepal_width=3.5&petal_length=1.5`. The parameters are listed in a table:

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> sepal_length	5.1	
<input checked="" type="checkbox"/> sepal_width	3.5	
<input checked="" type="checkbox"/> petal_length	1.5	
<input type="checkbox"/> petal_width	0.2	

The response status is `500 INTERNAL SERVER ERROR`. The response body is shown in HTML format:

```
<!doctype html>
<html lang=en>
<head>
  <title>ValueError: Input X contains NaN.
  RandomForestClassifier does not accept missing values encoded as NaN natively. For supervised learning, you
  might want to consider classes ensemble HistGradientBoostingClassifier and
```