Data Glacier Week 4 Assignment

Name: Muadh Faizan

Batch Code: LISUM01

Submission Date: 04-Jul-2021

Submitted To: Data Glacier

Summary

I used the Iris dataset which is included in the Sci-kit learn library, and applied a simple linear regression model on it. After pickling the model, I created a program which used Flask to deserialize and deploy the model which would successfully predict the class type based on input parameters. I tested this using Postman, and the results were successful.

Model Deployment Steps

1. Creating Linear Regression Model.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
import pickle
```

In [3]:
```
iris = load_iris()
X = pd.DataFrame(iris.data,columns=iris.feature_names)
y = pd.Series(iris.target)
```

In [6]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state=1)
```

In [7]:
```
model = LogisticRegression()
```

In [8]:
```
model.fit(X_train,y_train)
```

Out[8]:
```
LogisticRegression()
```

In [9]:
```
model.predict([[5.9,3,5.1,1.8]])
```

Out[9]:
```
array([2])
```

In [10]:
```
y_pred = model.predict(X_test)
```

In [11]:
```
metrics.confusion_matrix(y_test, y_pred)
```

Out[11]:
```
array([[14,  0,  0],
       [ 0, 17,  1],
       [ 0,  0, 13]], dtype=int64)
```

In [12]:
```
metrics.accuracy_score(y_test, y_pred)
```

Out[12]:
```
0.9777777777777777
```

In [14]:
```
filename = 'simple_model.pkl'
pickle.dump(model, open(filename, 'wb'))
```

2. Creating program to deploy the model.

model_dep.py

```python
# -*- coding: utf-8 -*-
"""
Created on Thu Jul  1 15:29:52 2021

@author: Muadh
"""

from flask import Flask, request, jsonify
import pandas as pd
import pickle

app = Flask(__name__)

@app.route('/')
def home():
    data = 'hello world'
    return jsonify({'data':data})

@app.route('/predict')
def predict():
    model = pickle.load(open('simple_model.pkl', 'rb'))
    sepal_len = request.args.get('sepal_len')
    sepal_wid = request.args.get('sepal_wid')
    petal_len = request.args.get('petal_len')
    petal_wid = request.args.get('petal_wid')

    pred = pd.DataFrame({'Sepal Length':[sepal_len], 'Sepal Width':[sepal_wid],
                         'Petal Length':[petal_len], 'Petal Width':[petal_wid]})
    pred_price = model.predict(pred)
    return jsonify({'Flower class':str(pred_price)})

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

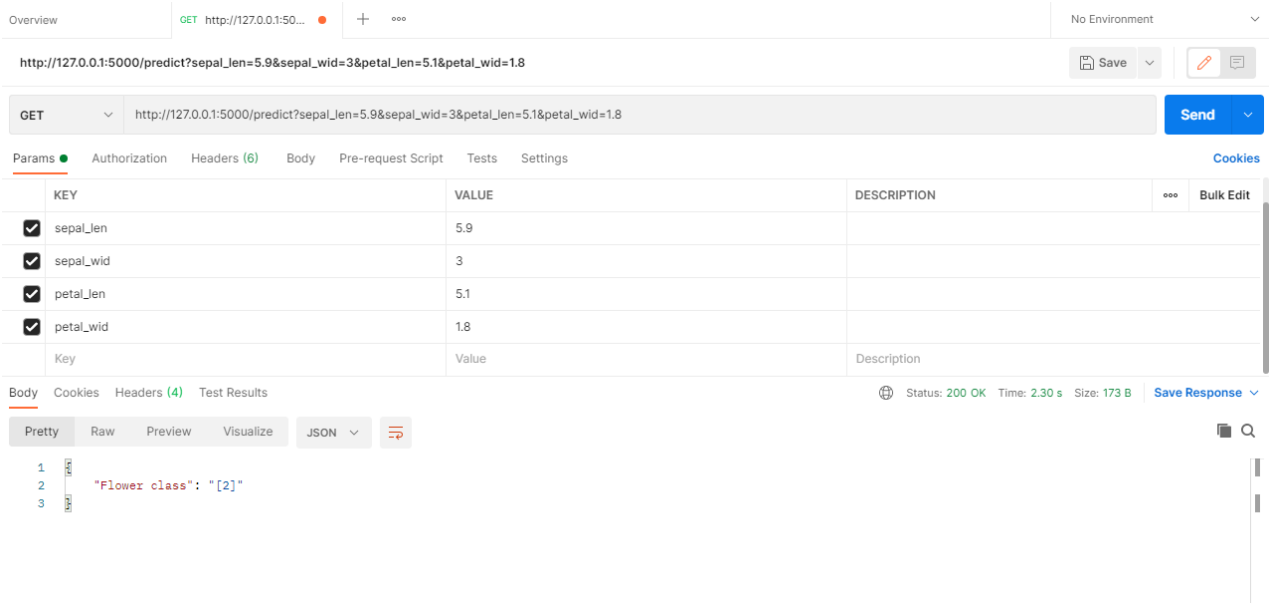
3. Running the python application

Anaconda Prompt (anaconda3) - python model_dep.py

```
(base) C:\Users\Muadh>cd ./Documents/Data Glacier

(base) C:\Users\Muadh\Documents\Data Glacier>python model_dep.py
 * Serving Flask app 'model_dep' (lazy loading)
 * 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 watchdog (windowsapi)
 * Debugger is active!
 * Debugger PIN: 179-988-915
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

4. Testing the model using Postman



As shown above, the model correctly predicts the class of flower after being given the 4 input values.