Integrate Flask with Scoring End Point :

TeamID – PNT2022TMID30742

```
import flask
from flask import request, render_template, Flask
from flask_cors import CORS


import requests

# you must manually set API_KEY below using information retrieved from your
IBM Cloud account.

API_KEY = "hAaNpiuYlDodbW1xwrpmxZTycN5gMBJW_06m9m2fU2r1"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token',
data={"apikey":
 API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' +
mltoken}

app = Flask(__name__)  # initialising flask app
# model = joblib.load('car performance') # load machine learning  model

@app.route('/', methods=['GET'])
def home():
    return render_template('ibm.html')
@app.route('/ibm.html')
def formpg():
    return render_template('ibm.html')
@app.route('/predict', methods=['POST', 'GET'])
def predict():
    if request.method == 'POST':
        CYLINDERS = int(request.form['cylinders'])
        DISPLACEMENT = int(request.form['displacement'])
        HOESEPOWER = int(request.form['horsepower'])
        WEIGHT = int(request.form['weight'])
        MODEL_YEAR = int(request.form['model_year'])
        ORIGIN = int(request.form['origin'])

        X = [[CYLINDERS, DISPLACEMENT, HOESEPOWER, WEIGHT, MODEL_YEAR,
ORIGIN]]
```

```
        # NOTE: manually define and pass the array(s) of values to be scored
in the next line
        payload_scoring = {"input_data": [{"fields": [['cylinders',
'displacement', 'horsepower', 'weight', 'model_year', 'origin']], "values":
X}]}

        response_scoring = requests.post('https://us-
south.ml.cloud.ibm.com/ml/v4/deployments/da27f9c6-0faf-452a-952c-
c400d4115758/predictions?version=2022-11-09', json=payload_scoring,
        headers={'Authorization': 'Bearer ' + mltoken})
        # print("Scoring response")
        predictions = response_scoring.json()

        predict = predictions['predictions'][0]['values'][0][0]
        # print("Final prediction : ", predict)

        # showing the prediction results in a ui

        output=predict
        if(output<=9):
            return render_template('submit.html', prediction_text="Worst
performance with mileage " + str(predict) +". Carry extra fuel")
        if(output>9 and output<=17.5):
            return render_template('submit.html', prediction_text="Low
performance with mileage " +str(predict) +". Don't go to long distance")
        if(output>17.5 and output<=29):
            return render_template('submit.html', prediction_text="Medium
performance with mileage " +str(predict) +". Go for a ride nearby.")
        if(output>29 and output<=46):
            return render_template('submit.html', prediction_text="High
performance with mileage " +str(predict) +". Go for a healthy ride")
        if(output>46):
            return render_template('submit.html', prediction_text="Very high
performance with mileage " +str(predict)+". You can plan for a Tour")
    else:
        return render_template('ibm.html')

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