APPLICATION BUILDING

BUILD PYTHON CODE

Date	19 November 2022
Team ID	PNT2022TMID04221
Project Name	Project – Statistical Machine Learning Approaches to Liver Disease Prediction

Let us build flask file 'app.py' which is a web framework written in python for server-side scripting.

Let's see step by step procedure for building the backend application.

- App starts running when "__name__" constructor is called in main.
- render_template is used to return html file.
- "GET" method is used to take input from the user.
- "POST" method is used to display the output to the user.

Importing Libraries

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

Libraries required for the app to run are to be imported.

Creating our flask app and loading the model

```
app = Flask(__name__)
```

Now after all the libraries are imported, we will be creating our flask app. and the load our model into our flask app.

Routing to the html Page:

@app.route is used to route the application where it should route to.

‘/’ URL is bound with the home.html function. Hence, when the home page of the web server is opened in the browser, the html page is rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method. Here, “index.html” is rendered when the home button is clicked on the UI and “predict.html” is rendered when the predict button is clicked.

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

@app.route('/prediction_form')
def predict1():
    return render_template('predict.html')
```

Firstly, we are rendering the index.html template and from there we are navigating to our prediction page that is predict.html. We enter input values here and these values are sent to the loaded model and the resultant output is displayed on result.html.

```
@app.route('/predict', methods=['POST'])
def predict():
    age = float(request.form['Age'])
    gender = float(request.form['Gender'])
    tb = float(request.form['Total_Bilirubin'])
    db = float(request.form['Direct_Bilirubin'])
    ap = float(request.form['Alkaline_phosphatase'])
    aa1 = float(request.form['Alamine_Aminotransferase'])
    aa2 = float(request.form['Aspartate_Aminotranferase'])
    tp = float(request.form['Total_Proteins'])
    al = float(request.form['Albumin'])
    agr = float(request.form['Albumin_and_Globulin_Ratio'])

    # converting data into float
    X = [[age,gender,tb,db,ap,aa1,aa2,tp,al,agr]]

    # NOTE: manually define and pass the array(s) of values to be scored in the next line
    payload_scoring = {"input_data": [{"field": [['age','gender','tb','db','ap','aa1','aa2','tp','al','agr']], "values": X }]}

    response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/06acb9ea-228d-4770-8841-afb3a9cedc6c/prediction
    headers={'Authorization': 'Bearer ' + mltoken})
    print(response_scoring.json())
    print(response_scoring)
    predictions = response_scoring.json()
    predict = predictions['predictions'][0]['values'][0][0]
    print(f"Final prediction : {predict}")
    # prediction = model.predict(data)[0]

    return render_template('result.html',result = predict)

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

Here the route for prediction is given and necessary steps are performed in order to get the predicted output.

Lastly, we run our app on the local host. Here we are running it on localhost:5000