AIML ASSIGNMENT - 2

1. Importing Libraries and Initializing Flask

```
from flask import Flask, request, render_template_string
import joblib
import numpy as np
import os
```

Flask: The Flask web framework allows us to build web applications in Python.

request: Used to access data submitted in the form, like user inputs.

render template string: Used to directly render HTML templates in the code.

joblib: A library for loading and saving models, which is optimized for larger data. Here, we use it to load a pre-trained model and scaler.

numpy: Provides support for numerical data manipulation, especially arrays. **os**: Used to construct paths to ensure the model files are loaded correctly.

2. App Initialization and Model Loading

```
app = Flask(__name__)

# Load the model and scaler with local paths using joblib
model_path = os.path.join(os.getcwd(), 'diabetes_model.pkl')
scaler_path = os.path.join(os.getcwd(), 'scaler.pkl')

model = joblib.load(model_path)
scaler = joblib.load(scaler_path)
```

- app = Flask(name): Initializes the Flask app, where __name__ helps Flask identify the app's resources.
- **model_path and scaler_path**: Define paths to the model and scaler files, making the app flexible to run in different directories.
- **joblib.load(model path)**: Loads the trained diabetes prediction model.

• **joblib.load(scaler_path)**: Loads the scaler to standardize input data (ensuring consistent predictions).

```
@app.route('/')
                                                                           Copy code
def home():
    home_html = '''
    <!DOCTYPE html>
        <meta charset="UTF-8">
        <title>Diabetes Prediction</title>
    </head>
        <h1>Diabetes Prediction</h1>
            <label>Pregnancies: <input type="number" name="pregnancies" step="any"</pre>
            <label>Glucose: <input type="number" name="glucose" step="any" required</pre>
            <label>Blood Pressure: <input type="number" name="blood_pressure" step=</pre>
            <label>Skin Thickness: <input type="number" name="skin_thickness" step=</pre>
            <label>Insulin: <input type="number" name="insulin" step="any" required</pre>
            <label>BMI: <input type="number" name="bmi" step="any" required></labe</pre>
            <label>Diabetes Pedigree Function: <input type="number" name="diabetes</pre>
            <label>Age: <input type="number" name="age" required></label><br>
            <button type="submit">Predict</button>
    </body>
    return render_template_string(home_html)
```

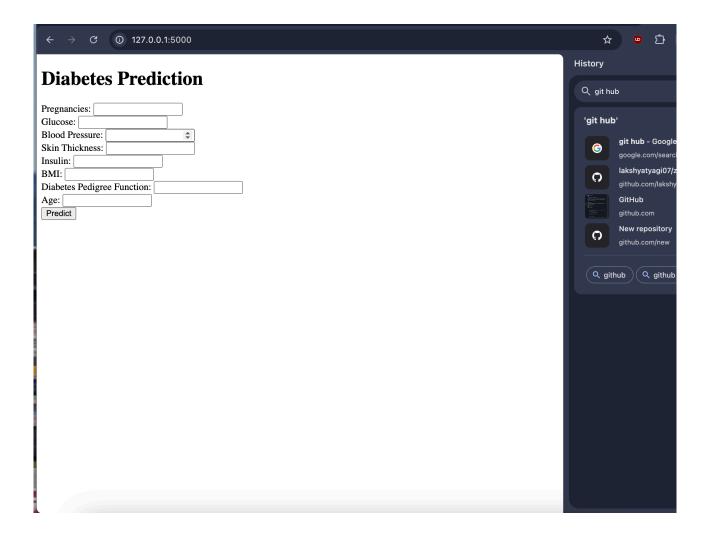
- @app.route('/'): Defines the home route, which loads the initial form.
- **home_html**: Contains the HTML form for the user to input data, such as pregnancies, glucose, and other health metrics.
- **return render_template_string(home_html)**: Renders the home_html as a webpage. This displays an input form with fields for each required feature in the model.

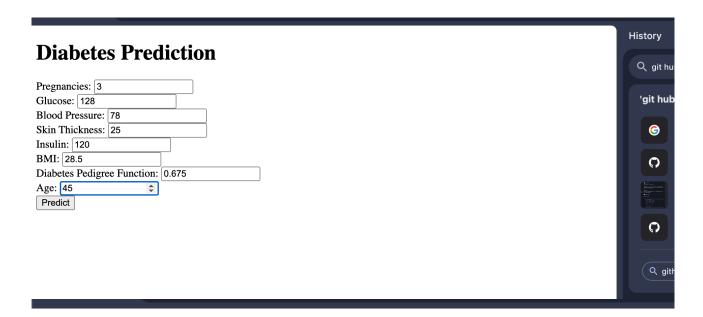
4. Prediction Route

```
@app.route('/predict', methods=['POST'])
                                                                       Copy code
def predict():
   try:
        # Get form data
        input_data = [float(request.form[key]) for key in request.form.keys()]
        # Reshape and scale input data
        input_array = np.array(input_data).reshape(1, -1)
        input_scaled = scaler.transform(input_array)
       # Make prediction
        prediction = model.predict(input_scaled)
        result = "Diabetes Detected" if prediction[0] == 1 else "No Diabetes Detect
   except Exception as e:
        result = f"An error occurred: {e}"
   result_html = f'''
   <!DOCTYPE html>
        <meta charset="UTF-8">
       <title>Prediction Result</title>
   </head>
       <h1>Prediction Result</h1>
        {result}
                                    \downarrow
   </body>
```

- @app.route('/predict', methods=['POST']): Defines the route that handles prediction logic when the form is submitted.
- **try-except Block**: Catches errors that might occur during data processing or prediction.

- input_data = [float(request.form[key]) for key in request.form.keys()]: Extracts and converts the form data into a list of floating-point numbers, one for each feature.
- **np.array(input_data).reshape(1, -1)**: Converts the input data into a 2D array, which is required for model input.
- **input_scaled = scaler.transform(input_array)**: Scales the data to match the format used during model training.
- **model.predict(input scaled)**: Generates a prediction using the trained model.
- **result**: Holds a message based on the prediction. If prediction[0] is 1, it implies that diabetes was detected; otherwise, "No Diabetes Detected."
- **result_html**: Contains HTML to display the prediction result, as well as a link to return to the form.





Prediction Result

Diabetes Detected

Try Again