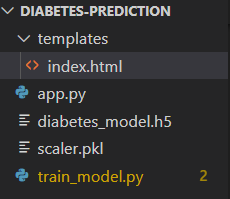
# Version 1



app.py

from flask import Flask, render\_template, request

import numpy as np

import tensorflow as tf

import joblib

# Tải mô hình đã lưu

model = tf.keras.models.load\_model("diabetes\_model.h5")

scaler = joblib.load("scaler.pkl")

app = Flask(\_\_name\_\_)

@app.route("/", methods=["GET", "POST"])

def index():

    prediction = None

    if request.method == "POST":

        try:

            # Lấy dữ liệu từ form

            features = [float(request.form[f]) for f in ["Pregnancies", "Glucose", "BloodPressure",

                                                         "SkinThickness", "Insulin", "BMI",

                                                         "DiabetesPedigreeFunction", "Age"]]

            # Chuẩn hóa dữ liệu

            input\_data = scaler.transform([features])

            # Dự đoán

            pred\_prob = model.predict(input\_data)[0][0]

            prediction = "Diabetic" if pred\_prob > 0.5 else "Not Diabetic"

        except Exception as e:

            prediction = f"Lỗi: {e}"

    return render\_template("index.html", prediction=prediction)

if \_\_name\_\_ == "\_\_main\_\_":

    app.run(debug=True)

train\_model.py

import pandas as pd

import numpy as np

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

import joblib

# Tải dataset

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"

columns = ["Pregnancies", "Glucose", "BloodPressure", "SkinThickness", "Insulin",

           "BMI", "DiabetesPedigreeFunction", "Age", "Outcome"]

df = pd.read\_csv(url, names=columns)

# Chuẩn bị dữ liệu

X = df.drop("Outcome", axis=1)

y = df["Outcome"]

# Chuẩn hóa dữ liệu

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# Chia tập train và test

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)

# Xây dựng mô hình deep learning

def create\_model():

    model = Sequential([

        Dense(128, activation="relu", input\_shape=(X\_train.shape[1],)),

        Dropout(0.3),

        Dense(64, activation="relu"),

        Dropout(0.3),

        Dense(32, activation="relu"),

        Dense(16, activation="relu"),

        Dense(1, activation="sigmoid")

    ])

    model.compile(optimizer="adam", loss="binary\_crossentropy", metrics=["accuracy"])

    return model

# Huấn luyện mô hình

model = create\_model()

model.fit(X\_train, y\_train, epochs=100, batch\_size=10, validation\_data=(X\_test, y\_test))

# Lưu mô hình và scaler

model.save("diabetes\_model.h5")

joblib.dump(scaler, "scaler.pkl")

print("Mô hình đã được lưu thành công!")

index.html

<!DOCTYPE html>

<html>

<head>

    <title>Diabetes Prediction</title>

</head>

<body>

    <h2>Diabetes Prediction Form</h2>

    <form method="post">

        <label>Pregnancies:</label><input type="number" name="Pregnancies" required><br>

        <label>Glucose:</label><input type="number" name="Glucose" required><br>

        <label>Blood Pressure:</label><input type="number" name="BloodPressure" required><br>

        <label>Skin Thickness:</label><input type="number" name="SkinThickness" required><br>

        <label>Insulin:</label><input type="number" name="Insulin" required><br>

        <label>BMI:</label><input type="number" name="BMI" step="0.1" required><br>

        <label>Diabetes Pedigree Function:</label><input type="number" name="DiabetesPedigreeFunction" step="0.01" required><br>

        <label>Age:</label><input type="number" name="Age" required><br>

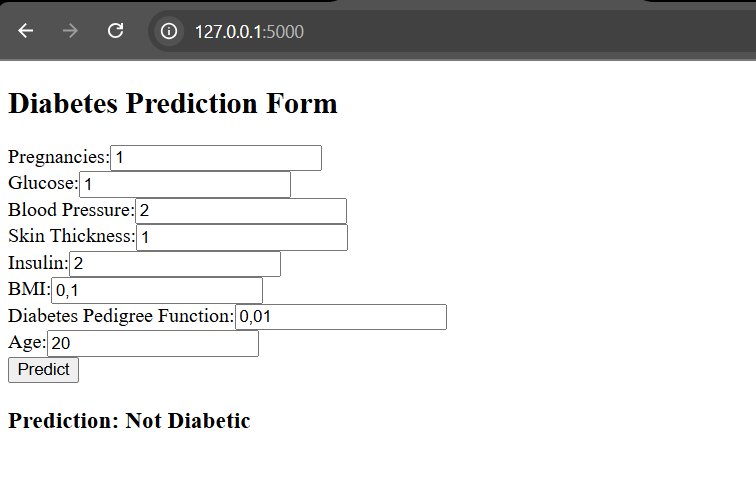
        <input type="submit" value="Predict">

    </form>

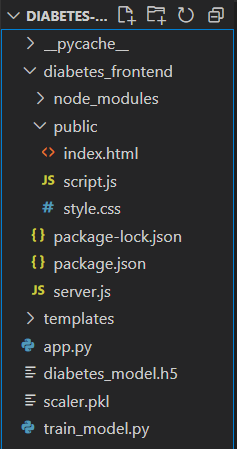
    <h3>Prediction: {{ prediction }}</h3>

</body>

</html>



# Version 2



train\_model.py

import pandas as pd

import numpy as np

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

import joblib

# Tải dataset

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"

columns = ["Pregnancies", "Glucose", "BloodPressure", "SkinThickness", "Insulin",

           "BMI", "DiabetesPedigreeFunction", "Age", "Outcome"]

df = pd.read\_csv(url, names=columns)

# Chia dữ liệu thành features (X) và labels (y)

X = df.drop("Outcome", axis=1)

y = df["Outcome"]

# Chuẩn hóa dữ liệu

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# Chia dữ liệu thành tập train và test

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)

# Xây dựng mô hình

def create\_model():

    model = Sequential([

        Dense(128, activation="relu", input\_shape=(X\_train.shape[1],)),

        Dropout(0.3),

        Dense(64, activation="relu"),

        Dropout(0.3),

        Dense(32, activation="relu"),

        Dense(16, activation="relu"),

        Dense(1, activation="sigmoid")

    ])

    model.compile(optimizer="adam", loss="binary\_crossentropy", metrics=["accuracy"])

    return model

# Huấn luyện mô hình

model = create\_model()

model.fit(X\_train, y\_train, epochs=100, batch\_size=10, validation\_data=(X\_test, y\_test))

# Lưu mô hình và scaler

model.save("diabetes\_model.h5")

joblib.dump(scaler, "scaler.pkl")

print("Model and scaler saved successfully!")

app.py

from flask import Flask, request, jsonify

import numpy as np

import tensorflow as tf

import joblib

from flask\_cors import CORS

# Tải mô hình và scaler

model = tf.keras.models.load\_model("diabetes\_model.h5")

scaler = joblib.load("scaler.pkl")

app = Flask(\_\_name\_\_)

CORS(app)  # Cho phép frontend giao tiếp với API

@app.route("/predict", methods=["POST"])

def predict():

    try:

        data = request.json  # Nhận dữ liệu từ frontend

        features = np.array(data["features"]).reshape(1, -1)  # Chuyển thành mảng numpy

        scaled\_features = scaler.transform(features)  # Chuẩn hóa dữ liệu

        prediction = model.predict(scaled\_features)[0][0]  # Dự đoán

        result = "Diabetic" if prediction > 0.5 else "Not Diabetic"

        return jsonify({"prediction": result})

    except Exception as e:

        return jsonify({"error": str(e)})

if \_\_name\_\_ == "\_\_main\_\_":

    app.run(debug=True, port=5000)

index.html

<!DOCTYPE html>

<html>

<head>

    <title>Diabetes Prediction</title>

</head>

<body>

    <h2>Diabetes Prediction Form</h2>

    <form method="post">

        <label>Pregnancies:</label><input type="number" name="Pregnancies" required><br>

        <label>Glucose:</label><input type="number" name="Glucose" required><br>

        <label>Blood Pressure:</label><input type="number" name="BloodPressure" required><br>

        <label>Skin Thickness:</label><input type="number" name="SkinThickness" required><br>

        <label>Insulin:</label><input type="number" name="Insulin" required><br>

        <label>BMI:</label><input type="number" name="BMI" step="0.1" required><br>

        <label>Diabetes Pedigree Function:</label><input type="number" name="DiabetesPedigreeFunction" step="0.01" required><br>

        <label>Age:</label><input type="number" name="Age" required><br>

        <input type="submit" value="Predict">

    </form>

    <h3>Prediction: {{ prediction }}</h3>

</body>

</html>

server.js

const express = require("express");

const cors = require("cors");

const bodyParser = require("body-parser");

const app = express();

app.use(cors());

app.use(bodyParser.json());

app.use(express.static("public"));  // Phục vụ file giao diện

const PORT = 3000;

app.listen(PORT, () => {

  console.log(`Frontend server running at http://localhost:${PORT}`);

});

index.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Diabetes Prediction</title>

    <link rel="stylesheet" href="style.css">

</head>

<body>

    <div class="container">

        <h2>Diabetes Prediction</h2>

        <form id="predictionForm">

            <label>Pregnancies:</label><input type="number" name="Pregnancies" required><br>

            <label>Glucose:</label><input type="number" name="Glucose" required><br>

            <label>Blood Pressure:</label><input type="number" name="BloodPressure" required><br>

            <label>Skin Thickness:</label><input type="number" name="SkinThickness" required><br>

            <label>Insulin:</label><input type="number" name="Insulin" required><br>

            <label>BMI:</label><input type="number" name="BMI" step="0.1" required><br>

            <label>Diabetes Pedigree Function:</label><input type="number" name="DiabetesPedigreeFunction" step="0.01" required><br>

            <label>Age:</label><input type="number" name="Age" required><br>

            <button type="submit">Predict</button>

        </form>

        <h3 id="result"></h3>

    </div>

    <script src="script.js"></script>

</body>

</html>

style.css

body {

    font-family: Arial, sans-serif;

    text-align: center;

    background-color: #f4f4f4;

}

.container {

    width: 50%;

    margin: auto;

    background: white;

    padding: 20px;

    box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);

    border-radius: 10px;

}

input, button {

    width: 90%;

    padding: 10px;

    margin: 5px;

}

script.js

document.getElementById("predictionForm").addEventListener("submit", async function(event) {

    event.preventDefault();

    let formData = new FormData(event.target);

    let features = Object.values(Object.fromEntries(formData.entries())).map(Number);

    let response = await fetch("http://127.0.0.1:5000/predict", {

        method: "POST",

        headers: {"Content-Type": "application/json"},

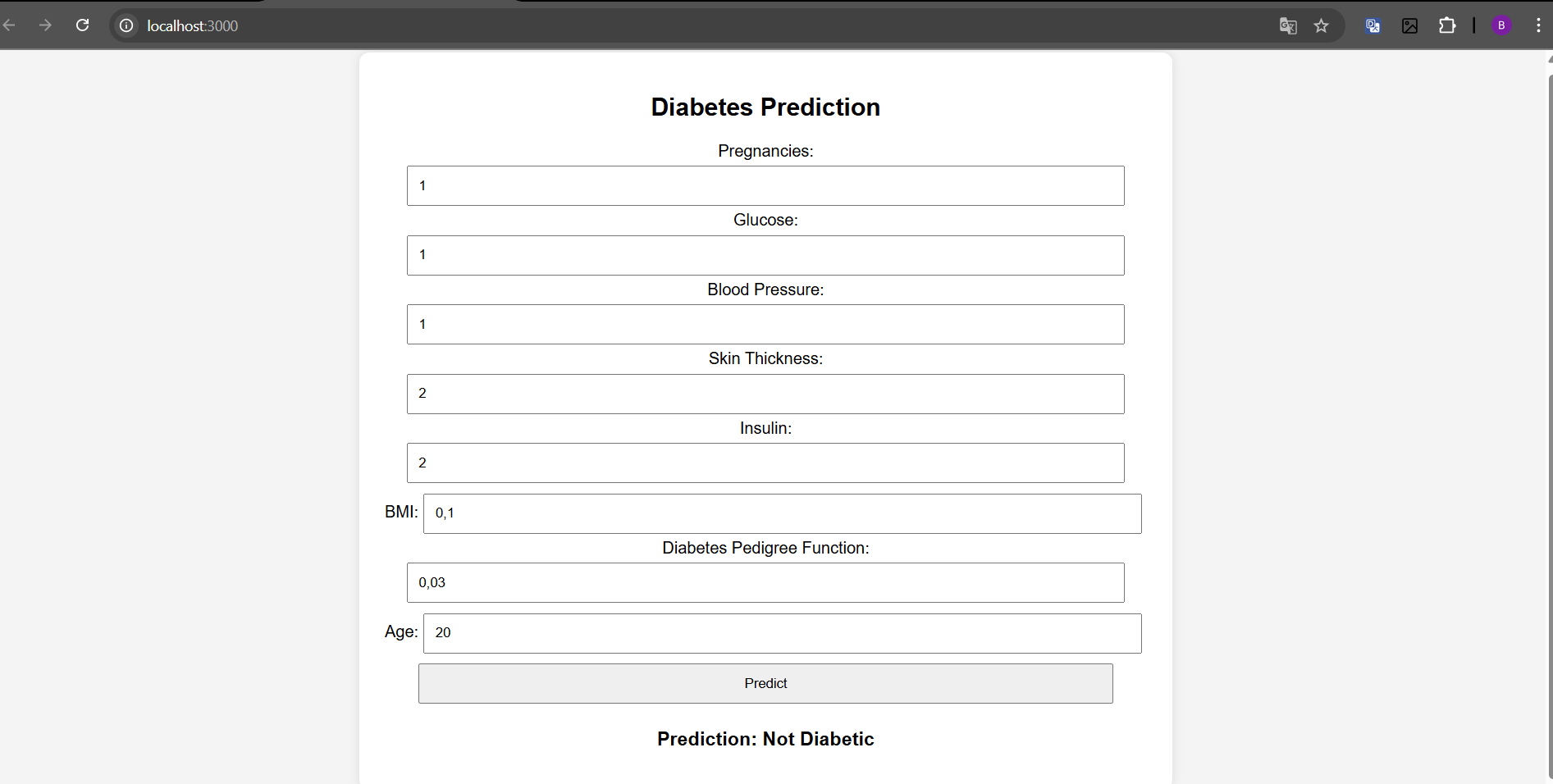
        body: JSON.stringify({ features: features })

    });

    let result = await response.json();

    document.getElementById("result").innerText = "Prediction: " + result.prediction;

});



# Version 3

server.py

from flask import Flask, request, jsonify

from flask\_cors import CORS

import numpy as np

import tensorflow as tf

# Load mô hình đã huấn luyện (đặt đúng đường dẫn của bạn)

model = tf.keras.models.load\_model("diabetes\_model.h5")

# Khởi tạo Flask app

app = Flask(\_\_name\_\_)

CORS(app)  # Cho phép truy cập từ frontend

@app.route("/predict", methods=["POST"])

def predict():

    try:

        data = request.json  # Nhận dữ liệu JSON từ frontend

        features = np.array(data["features"]).reshape(1, -1)  # Chuyển thành mảng NumPy

        prediction = model.predict(features)[0][0]  # Lấy giá trị dự đoán

        result = "Diabetic" if prediction > 0.5 else "Non-Diabetic"

        return jsonify({"prediction": result, "probability": float(prediction)})

    except Exception as e:

        return jsonify({"error": str(e)})

if \_\_name\_\_ == "\_\_main\_\_":

    app.run(debug=True, port=5000)

index.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Diabetes Prediction</title>

</head>

<body>

    <h2>Enter Patient Data</h2>

    <form id="predictForm">

        <input type="text" id="features" placeholder="Enter values comma-separated">

        <button type="submit">Predict</button>

    </form>

    <h3 id="result"></h3>

    <script>

        document.getElementById("predictForm").addEventListener("submit", function(event) {

            event.preventDefault();

            let features = document.getElementById("features").value.split(",").map(Number);

            fetch("http://127.0.0.1:5000/predict", {

                method: "POST",

                headers: { "Content-Type": "application/json" },

                body: JSON.stringify({ features: features })

            })

            .then(response => response.json())

            .then(data => {

                document.getElementById("result").innerText =

                `Prediction: ${data.prediction} (Probability: ${data.probability.toFixed(2)})`;

            })

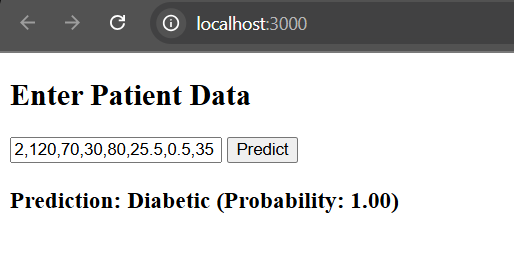
            .catch(error => console.error("Error:", error));

        });

    </script>

</body>

</html>



# Version 4

server.py

from flask import Flask, request, jsonify

import numpy as np

import tensorflow as tf

app = Flask(\_\_name\_\_)

# Load mô hình đã huấn luyện

model = tf.keras.models.load\_model("diabetes\_model.h5")

@app.route("/predict", methods=["POST"])

def predict():

    data = request.json

    features = np.array(data["features"]).reshape(1, -1)

    prediction = model.predict(features)[0][0]

    result = "Diabetic" if prediction > 0.5 else "Non-Diabetic"

    return jsonify({"prediction": result, "probability": float(prediction)})

if \_\_name\_\_ == "\_\_main\_\_":

    app.run(host="0.0.0.0", port=5000)

train\_model.py

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout, BatchNormalization

from tensorflow.keras.optimizers import AdamW

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

# Tải dataset

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"

columns = ["Pregnancies", "Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI", "DiabetesPedigreeFunction", "Age", "Outcome"]

df = pd.read\_csv(url, names=columns)

# Chia dữ liệu thành tập train/test

X = df.iloc[:, :-1].values

y = df.iloc[:, -1].values

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Chuẩn hóa dữ liệu

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Xây dựng mô hình cải tiến

def create\_model():

    model = Sequential([

        Dense(128, input\_shape=(X\_train.shape[1],)),

        BatchNormalization(),

        tf.keras.layers.LeakyReLU(alpha=0.1),

        Dropout(0.4),

        Dense(64),

        BatchNormalization(),

        tf.keras.layers.LeakyReLU(alpha=0.1),

        Dropout(0.3),

        Dense(32),

        BatchNormalization(),

        tf.keras.layers.LeakyReLU(alpha=0.1),

        Dense(16, activation="relu"),

        Dense(1, activation="sigmoid")

    ])

    model.compile(optimizer=AdamW(learning\_rate=0.001), loss="binary\_crossentropy", metrics=["accuracy"])

    return model

# Huấn luyện mô hình

model = create\_model()

early\_stopping = tf.keras.callbacks.EarlyStopping(monitor="val\_loss", patience=10, restore\_best\_weights=True)

model.fit(X\_train, y\_train, epochs=100, validation\_data=(X\_test, y\_test), batch\_size=16, callbacks=[early\_stopping])

# Lưu mô hình

model.save("diabetes\_model.h5")

DockerFile

# Sử dụng hình ảnh Python chính thức

FROM python:3.9

# Đặt thư mục làm việc

WORKDIR /app

# Sao chép file vào container

COPY server.py diabetes\_model.h5 /app/

# Cài đặt các thư viện cần thiết

RUN pip install flask tensorflow numpy

# Mở cổng 5000

EXPOSE 5000

# Chạy server

CMD ["python", "server.py"]

requirements.txt

flask

tensorflow

numpy

joblib

scikit-learn

