Diabetic Prediction Framework: A Privacy-Enhanced Approach with Metaheuristic Optimization and Blockchain

1. Project Overview

This project develops a cutting-edge **Diabetic Prediction System** that integrates advanced machine learning techniques with blockchain technology and a user-friendly web interface. Its primary goal is to provide a robust, accurate, and transparent solution for assessing diabetes risk based on various patient health parameters.

The system leverages an **ensemble** of powerful machine learning models, enhanced by **metaheuristic optimization** for feature selection, to classify individuals into Non-Diabetic, Pre-Diabetic, or Diabetic categories. Crucially, every prediction made is securely recorded onto a custom-built **blockchain**, ensuring an immutable and auditable history of all health assessments. The entire solution is made accessible through an intuitive Flask-based web interface, facilitated by Ngrok for public access, allowing users to input health parameters and receive real-time predictions with verifiable integrity.

2. Key Features

- **Accurate Diabetes Prediction:** Utilizes an ensemble of machine learning models (XGBoost and Logistic Regression) for high predictive accuracy.
- **Optimized Feature Selection:** Incorporates **Metaheuristic Optimization** (specifically PSO - Particle Swarm Optimization) to select the most relevant features, enhancing model performance and efficiency.
- **Class Imbalance Handling:** Employs SMOTE (Synthetic Minority Over-sampling Technique) during training to address skewed class distributions, ensuring fair predictions across all diabetes stages.
- **Immutable Prediction Records:** Integrates a custom **Blockchain** to securely store every prediction, providing a tamper-proof and auditable history.
- **User-Friendly Web Interface:** A Flask-based web application with a clean HTML/Tailwind CSS frontend for easy patient data input and result visualization.
- **Real-time Risk Assessment:** Provides instant predictions along with confidence scores and tailored health recommendations.
- **Public Accessibility:** Deployed using **Ngrok** for easy access and demonstration from any web browser.

3. Technologies Used

- **Frontend:**
  - HTML
  - Tailwind CSS
  - Jinja Templating
- **Backend:**
  - Python (Core language for all logic)
  - Scikit-learn (Machine Learning models, preprocessing)
  - NumPy (Numerical operations)
  - Pandas (Data manipulation)
  - Joblib (Model persistence)
  - Custom Python Blockchain Implementation (Block and Blockchain classes)
- **Frameworks:**
  - Flask (Web application framework)
  - Pyngrok (For creating public tunnels)

4. Project Structure

```
Diabetic_Prediction_Model/
├── app.py                  # Main Flask application logic
├── blockchain.py           # Custom Blockchain implementation
├── diabetic.csv.csv        # Original dataset (ensure this is in your Drive)
├── ensemble_model.pkl      # Trained ensemble ML model
├── gender_encoder.pkl      # LabelEncoder for 'Gender'
├── original_feature_names.pkl  # List of feature names used by the model
├── scaler.pkl              # StandardScaler for feature scaling
├── selected_columns.pkl    # Indices of features selected by PSO
├── diabetes_blockchain.json    # Persistent storage for the blockchain
└── templates/              # HTML templates for the web interface
    ├── index.html          # Main prediction input form and results display
    └── history.html        # Blockchain history viewer
```

(Note: Your Colab notebook file, e.g., YourProjectNotebook.ipynb, would typically reside outside this Diabetic_Prediction_Model folder, often in Colab Notebooks/.)

5. How to Run the Project (in Google Colab)

1. **Upload diabetic.csv.csv:** Ensure your diabetic.csv.csv dataset is uploaded to your Google Drive, preferably in the same Diabetic_Prediction_Model folder or accessible via '/content/drive/MyDrive/Colab Notebooks/Diabetic/diabetic.csv'.
2. **Open Your Colab Notebook:** Open your project's .ipynb file in Google Colab.

3. **Run Cells Sequentially:** Execute all code cells in your Colab notebook from top to bottom.
    - **Cell 1:** Creates templates directory and writes blockchain.py.
    - **Cell 2:** Writes templates/index.html.
    - **Cell 3:** Writes templates/history.html.
    - **Cell 4:** Writes app.py (contains the ML logic, blockchain integration, and Flask routes).
    - **Cell 5:** Installs libraries, mounts Google Drive, copies all necessary files (app.py, blockchain.py, templates/) to your Diabetic_Prediction_Model folder on Drive, changes the working directory, sets up Ngrok, and launches the Flask application.
        - **Important:** You will need to replace "YOUR_NGROK_AUTH_TOKEN_HERE" with your actual Ngrok authentication token in Cell 5.
        - The Flask app will run on http://localhost:5000 and Ngrok will expose it publicly.
4. **Access the Web App:** Once Cell 5 finishes executing, it will print a Flask App Public URL: https://... link. Click this link to open your diabetes prediction web application in your browser.

6. Sample Usage

Navigate to the public URL provided by Ngrok. You can input patient data into the form fields.

Example Inputs:

- **Non-Diabetic (N):**
    - Gender: F, AGE: 30, Urea: 3.0, Cr: 55, HbA1c: 5.0, Chol: 4.0, TG: 1.0, HDL: 1.5, LDL: 2.0, VLDL: 0.5, BMI: 22.0
- **Pre-Diabetic (P):**
    - Gender: M, AGE: 45, Urea: 4.5, Cr: 70, HbA1c: 6.0, Chol: 5.0, TG: 1.8, HDL: 1.0, LDL: 2.8, VLDL: 0.8, BMI: 26.0
- **Diabetic (Y):**
    - Gender: F, AGE: 60, Urea: 7.0, Cr: 90, HbA1c: 7.5, Chol: 6.0, TG: 3.0, HDL: 0.8, LDL: 3.5, VLDL: 1.2, BMI: 32.0

After submitting, the prediction, confidence, and recommendations will be displayed. You can then click "View History" to see the immutable blockchain record of your prediction.

7. Resetting Blockchain History

To clear the diabetes_blockchain.json file and reset the prediction history, you can run the following command in a new Colab cell (after mounting Drive and setting PROJECT_DIR_ON_DRIVE):

```
import os
BLOCKCHAIN_FILE = os.path.join(PROJECT_DIR_ON_DRIVE,
'diabetes_blockchain.json')
if os.path.exists(BLOCKCHAIN_FILE):
    os.remove(BLOCKCHAIN_FILE)
    print(f"Removed existing blockchain file: {BLOCKCHAIN_FILE}")
else:
    print(f"Blockchain file not found at {BLOCKCHAIN_FILE}. Nothing to remove.")
```

After running this, restart your Colab runtime and re-run all cells to start with a fresh blockchain.

8. Future Enhancements

- Implement user authentication and role-based access control.
- Explore more advanced deep learning models for prediction.
- Integrate with a decentralized storage solution (e.g., IPFS) for patient data, linking hashes to the blockchain.
- Enhance the web interface with interactive visualizations of patient data and prediction trends.
- Implement a peer-to-peer network for the blockchain to achieve true decentralization.