Document Overview

This document explains a Python script designed to create a web application using Flask. The application predicts the risk of medical conditions based on a user's gender and blood group. The predictions are derived from a trained machine learning model, specifically a RandomForestClassifier, trained on a healthcare dataset. The user's inputs and the predicted medical conditions are stored in a CSV file for record-keeping.

Detailed Explanation

1. Imports and Setup

- **Libraries:** Uses standard libraries such as **os** for operating system interfaces, **pandas** for data manipulation, **pickle** for object serialization, and components from **sklearn** for model training and preprocessing.
- **Flask:** Utilized for creating the web application, handling routes, and rendering HTML forms.

2. Data Preparation

- **Data Loading:** Loads healthcare data from a CSV file containing columns like 'Blood Group Type', 'Gender', and 'Medical Condition'.
- **Feature Encoding:** Uses **LabelEncoder** to convert categorical text data into model-understandable numerical data. Each category in the columns is encoded separately.

3. Model Training

- **Feature Selection:** Selects 'Blood Group Type' and 'Gender' as features and 'Medical Condition' as the target.
- **Data Splitting:** Splits the data into training and testing sets to evaluate the model's performance.
- **Random Forest Model:** Trains a RandomForestClassifier with the training data.
- **Model Serialization:** Saves the trained model and encoders to disk using **pickle** for later use in the application.

4. Web Application Setup

- **Flask App Initialization:** Initializes the Flask application.
- **HTML Form:** An HTML form embedded in the Python script allows users to submit their name, gender, and blood group.
- **Route Definition:** Defines a route that handles both GET and POST requests. GET serves the form, while POST processes submitted data.

5. Prediction and Response

Form Submission Handling: On form submission, the script:

- Retrieves input values from the form.
- Prepares data for prediction using the stored encoders and model.
- Performs prediction to determine the most likely medical conditions.

- Generates a response that lists the top probable medical conditions based on the user's inputs.

Data Storage: Stores the user's input along with the predicted condition in a CSV file, creating the file if it doesn't exist or appending to it if it does.

6. Application Execution

- **Main Block:** Ensures that the Flask app runs only if the script is executed directly, not if it's imported as a module.
- **Debug Mode:** Runs the app in debug mode to facilitate development and troubleshooting.

Conclusion

This script provides a complete solution for predicting medical conditions using a RandomForestClassifier trained on healthcare data. It also demonstrates practical application development with Flask, including user interaction through web forms and backend data storage.

This setup is ideal for demonstration purposes or initial deployment in a controlled environment.