Dialysis_of_Patient

Problem Statement:

The goal of this project is to **develop a machine learning-based prediction system** to determine whether a patient with Chronic Kidney Disease (CKD) **requires dialysis** based on clinical and biochemical test inputs.

The system should also provide **medical explanations** for predictions using **Google Gemini**, making it easier for patients and non-medical users to understand. The final product includes a **trained Random Forest model** and a **Streamlit web application** for real-time user interaction and predictions.

Steps Carried Out:

1. Data Preprocessing & Model Training (CKD Model Training.py):

1. **Import Required Libraries:**
   - Libraries for data handling (pandas, numpy)
   - Machine learning tools from scikit-learn
   - Model saving with joblib

2. **Load Dataset:**
   - Reads CKD dataset from a CSV file.

3. **Handle Missing Values:**
   - Replace all '?' with NaN.
   - Drops rows with excessive missing data.
   - Fills missing values based on datatype:
     - Numeric columns: median
     - Categorical columns: mode, then label-encoded.

4. **Feature Engineering:**
   - Drops unnecessary columns (e.g., 'id').
   - Ensures the target column is named consistently ('classification').

5. **Split Data:**
   - Splits data into training and testing sets (80-20 split).

6. **Train Model:**
    - Trains a RandomForestClassifier on the training data.

7. **Evaluate Model:**
    - Uses accuracy_score and classification_report to evaluate model performance on the test set.

8. **Save Model:**
    - Stores the trained model as Dialysis_Status.pkl using joblib.

---

2. Streamlit Web App Interface (Dialysis_Status.py):

1. **Import Required Libraries:**
    - Streamlit for web interface
    - joblib for loading the trained model
    - LangChain & Google Generative AI for medical explanation

2. **API Setup:**
    - Loads Gemini API key from .env file
    - Configures Gemini 2.0 Flash model via google.generativeai
    - Initializes LangChain's ChatGoogleGenerativeAI for LLM use

3. **Load Trained Model:**
    - Loads the Random Forest model from Dialysis_Status.pkl

4. **Create UI:**
    - Sets Streamlit page title and instructions
    - Collects user inputs via sliders and dropdowns for all clinical features:
        - E.g., age, blood pressure, sugar, hemoglobin, WBC count, RBC count, hypertension status, etc.

5. **Make Prediction:**
    - On clicking the **“Predict Dialysis Need”** button:
        - Passes input data to the model
        - Displays result: "Dialysis Required" or "Dialysis Not Required"

6. **Explain Prediction (via Gemini):**
    - Sends user input to Gemini with a prompt asking for a simple medical explanation.
    - Displays AI-generated explanation to the user.

Summary:
This two-part project integrates:

- **Machine learning** for predictive analytics,
- **Streamlit** for building an interactive UI,
- **Gemini LLM (Generative AI)** for producing patient-friendly explanations.

It is a comprehensive solution aimed at **assisting healthcare professionals and patients** in understanding and acting on CKD dialysis decisions more efficiently.