Project Overview

This project, **AI in Mental Health: Personalized Stress Management**, uses Python and React to help users analyze stress and get personalized recommendations. The backend is built with Flask, and the AI model predicts stress levels based on user inputs. Each file in the project has a specific purpose.

Files and Their Purpose

1. **feedback_loop.py**:
    - Collects user feedback (e.g., actual stress level) to improve the AI model over time.
    - Uses pandas to save feedback in a CSV file.
2. **generate_synthetic_data.py**:
    - Creates fake data for training and testing the AI model.
    - Uses pandas to save the data and random to generate it.
3. **logger.py**:
    - Keeps logs of important events, like errors or processes.
    - Uses logging to create separate logs for training and API activity.
4. **mood_analysis.py**:
    - Loads the trained AI model to analyze user mood and predict stress levels.
    - Uses joblib for loading the model and sklearn for processing text.
5. **predict_stress.py**:
    - Connects the frontend with the backend for mood predictions.
    - Uses flask to handle requests and send responses.
6. **recommendations.py**:
    - Provides personalized suggestions based on stress levels.
    - Simple Python logic without external libraries.
7. **server.py**:
    - Manages the backend server.
    - Handles routes like:
        - /predict: Predicts stress.
        - /feedback: Saves user feedback.
        - /trends: Shows past stress trends.
    - Uses flask and pandas for these functions.
8. **train_model.py**:
    - Trains the AI model using labeled data.

- Uses sklearn for model training and pandas for data handling.
- Saves the model with joblib.

How It All Works

1. **Frontend (React)**: Users input their mood.
2. **Backend (Flask)**: Processes inputs and sends stress predictions.
3. **AI Model**: Analyzes data and predicts stress levels.
4. **Feedback Loop**: Improves the system over time with user feedback.

The libraries (e.g., pandas, joblib, sklearn, and flask) are used to handle data, train models, save progress, and manage the server

Processed Folder:

1. **mood_data_cleaned.csv**:
    - **Purpose**: This is the preprocessed dataset where all the raw data from various sources has been cleaned, normalized, and organized for further machine learning tasks.
    - **Use**: Used in the training process of the machine learning model. This file ensures that unnecessary noise and inconsistencies are removed from the raw data.

Raw Folder:

1. **emotion_sentimen_dataset.csv**:
    - **Purpose**: Contains raw data about emotions and sentiment, possibly labeled with emotional categories like "happy," "sad," "stressed," etc.
    - **Use**: Acts as a foundational dataset for emotion analysis tasks.
    - **To Know If Labeled**: Check if there is a column in the dataset specifying categories for emotions (e.g., "Label" or "Category"). If present, it is labeled.
2. **test.csv**:
    - **Purpose**: Contains raw or processed data for testing the trained model's performance.
    - **Use**: Ensures the model's accuracy and reliability during testing.
    - **Labeling**: Check if there is a ground truth (output column) in the file to compare predictions against. If yes, it's labeled.
3. **wesad_dataset.csv**:
    - **Purpose**: WESAD (Wearable Stress and Affect Detection) dataset is a public dataset commonly used for stress detection using wearable device data like heart rate and breathing patterns.
    - **Use**: Provides physiological data for training and evaluating the stress detection model.
    - **Labeling**: Analyze the columns for predefined stress levels (e.g., "High Stress" or "Low Stress") to confirm if it's labeled.

Synthetic Data Folder:

1. **generate_synthetic_data.py**:
    - **Purpose**: Python script to generate synthetic data for training or testing purposes when real data is insufficient or imbalanced.
    - **Use**: Creates artificial datasets resembling the original dataset distribution to improve model robustness.

2. **generate_synthetic_data.csv**:
    - **Purpose**: A CSV file containing the generated synthetic data created by generate_synthetic_data.py.
    - **Use**: Acts as an additional dataset for model training or testing.
3. **synthetic_mood_data.csv**:
    - **Purpose**: Contains synthetic mood data generated to simulate different emotional states for analysis.
    - **Use**: Complements the training dataset with diverse scenarios.

Notebooks Folder:

1. **data_preprocessing.ipynb**:
    - **Purpose**: A Jupyter notebook that contains steps to clean, normalize, and preprocess raw datasets.
    - **Use**: Helps prepare raw data for model training by removing inconsistencies, handling missing values, and extracting features.
    - **Dataset Cleaning**: This is the primary tool for cleaning datasets, producing files like mood_data_cleaned.csv.
2. **data_preprocessing.py**:
    - **Purpose**: A Python script equivalent of data_preprocessing.ipynb for batch or automated preprocessing.
    - **Use**: Automates the data cleaning pipeline to handle large datasets efficiently.
3. **data_preprocessing.txt**:
    - **Purpose**: Likely contains instructions, descriptions, or logs related to the preprocessing tasks.
    - **Use**: Provides documentation for preprocessing methodologies used in the project.
4. **model_training.ipynb**:
    - **Purpose**: A Jupyter notebook used for training the machine learning model using cleaned and labeled data.
    - **Use**: Trains and evaluates models, and saves the best-performing ones for deployment.

How to Identify If a Dataset is Labeled or Not:

- **Check Columns**: Look for a column that specifies labels, such as "Category," "Emotion," or "Stress_Level."
- **Unique Values**: If a column contains unique categories or numerical codes for classes, the dataset is labeled.
- **Example**: A dataset with columns like ["Text", "Emotion"] where "Emotion" values are "Happy," "Sad," or "Stressed" is labeled.
- **Unlabeled Dataset**: If the dataset lacks any column providing the ground truth, it is unlabeled.

Method Used for Labeling:

- **Manual Labeling**: Data entries are manually tagged by humans based on their meaning.
- **Automatic Labeling**: Machine learning or rule-based models tag data automatically using heuristics or predefined rules.
- **Example in Your Project**:
    - Keywords like "stressed" or "anxious" in text data may have been mapped to the "High Stress" label.
    - Heart rate data above a threshold might automatically classify a row as "High Stress."