# PCA+CNN+RANDOM_FOREST IMPLEMENTATION

# PSEUDO CODE

1. Import required libraries for image processing, visualization, machine learning, deep learning, and federated learning setup.

2. Mount Google Drive and define the path where the image dataset is stored.

3. Define a function to:
   - Load images from path
   - Resize each image to 224x224
   - Extract labels from directory structure
   - Return the image array and label array

4. List image file paths and apply the image loader function to get data and labels.

5. Encode the class labels using label binarization (One-vs-All binary matrix).

6. Split the dataset into training and test sets (90% train, 10% test).

7. Reshape training and test image data into 2D format (samples x features).

8. Standardize the image pixel values using StandardScaler.

9. Apply PCA to reduce dimensionality to 20 principal components for both train and test sets.

10. Define a function to:
    - Create multiple clients (e.g., 6)
    - Randomly shuffle and shard training data
    - Return a dictionary with client names and their corresponding data

11. Define a function to:
    - Convert each client’s data into a TensorFlow dataset
    - Shuffle and batch the data

12. Process and batch all clients' training data.
    Also prepare the test dataset in batch format.

13. Define training parameters:
    - Number of communication rounds
    - Loss function, optimizer, and metrics

14. Define helper functions:
    - To compute weight scaling factor based on data size per client
    - To scale model weights
    - To sum all scaled weights across clients
    - To evaluate the model using accuracy and loss

15. Define a class to create a model combining:
    - A basic CNN (Dense layers)
    - Followed by a simulated Random Forest (fully connected output layer)
    - Output layer uses softmax activation

16. Initialize a global model using the CNN + Random Forest class.

17. Start the federated learning training loop:
    a. For each communication round:
        i. Get the global model's weights
        ii. For each client:
            - Create and compile a new local model

- Load global weights into local model

- Train on client data

- Scale and store local model weights

- Clear session to free memory

iii. Average the scaled local weights and update the global model

iv. Evaluate the global model on the test set

v. Store performance metrics like accuracy, loss, precision, recall, F1 score, sensitivity, and specificity

18. After training:

- Plot graphs for Accuracy, Precision, Recall, F1 Score, Sensitivity, and Specificity over communication rounds

**PCA+CNN+RANDOM FOREST**