Pipeline for Federated Quantum Machine Learning

1. Data Loading and Preprocessing

1. **Load and Label Data:**
    - Load brain tumor data from .mat files.
    - Extract labels and count the number of samples for each tumor type.
    - Create a bar chart to visualize the number of samples per label.
2. **Image and Mask Extraction:**
    - Extract images and tumor masks from the .mat files.
    - Normalize the images and prepare the tumor regions for visualization.
    - Display sample images with tumor regions highlighted.
3. **Load Processed Data:**
    - Load preprocessed data from a .npz file.
    - Print the shapes of loaded images and labels.
    - Display sample images from the processed data.

2. Data Preparation for Federated Learning

1. **Flatten Images and Encode Labels:**
    - Flatten the image data for compatibility with quantum algorithms.
    - Encode the labels using LabelEncoder.
2. **Split Data into Training and Test Sets:**
    - Perform a stratified split to ensure proportional representation of each class.
    - Print the shapes of training and test datasets.
    - Count the number of instances for each label in the training and test sets.

3. Federated Learning Setup

1. **Client Class Definition:**
    - Define a Client class to manage client-specific models and datasets.
2. **Distribute Data to Clients:**
    - Distribute the training data across multiple clients, each receiving a subset of the data for several epochs.
    - Print the label distribution for the first epoch of each client.

4. Quantum Model Training

1. **Model Training Function:**
    - Define a train_model function to train a VQC model on the provided dataset.

- Initialize or continue training a VQC model and evaluate its performance on the training and test sets.

2. Training Callback:
    - Define a training_callback function to monitor training iterations.

5. Federated Learning Implementation

1. Initialize Clients with Models:
    - Initialize clients with their respective datasets.
    - Print sample data points for verification.
2. Weight Averaging Techniques:
    - Define an average_weights_simple function to average model weights.
    - Initialize client arrays for different averaging techniques.

6. Federated Training Loop

1. Epoch-wise Training:
    - Iterate through multiple epochs and perform the following steps for each epoch:
        - Train models on each client's dataset.
        - Collect model weights and test scores.
        - Compute new global model weights using the chosen averaging technique.
        - Initialize new models with the global weights for each client.
        - Evaluate the new global model and record its accuracy.

7. Evaluation

1. Model Evaluation:
    - Define a calculate_accuracy function to evaluate the model's performance on the test set.
    - Print global model accuracy for each epoch and technique