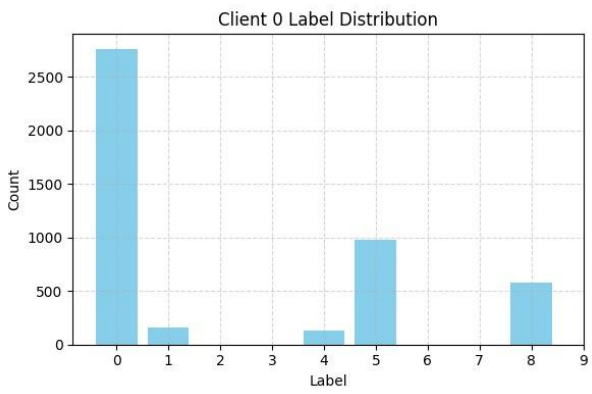
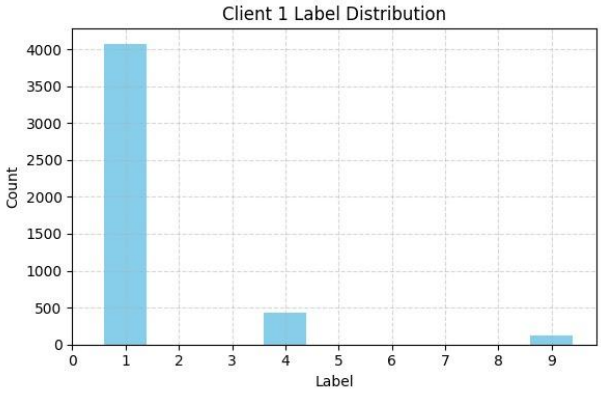
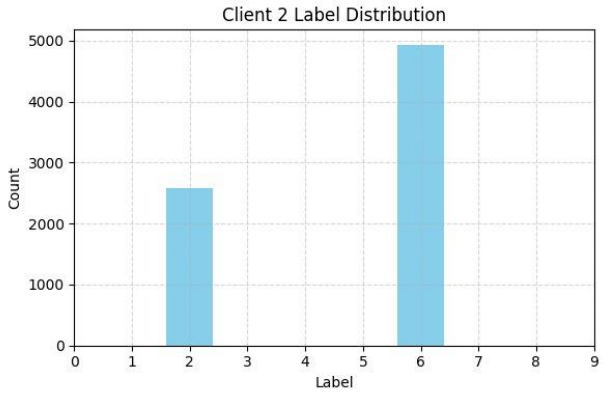
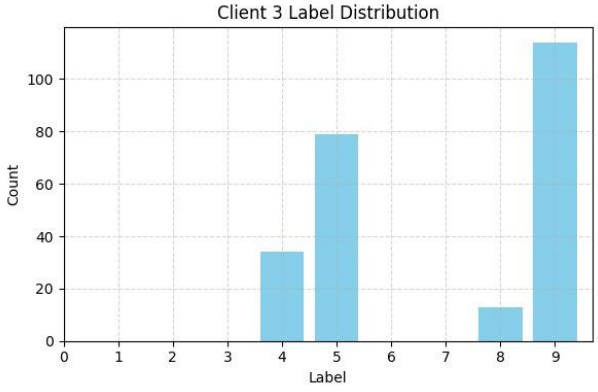
1. Per-Client Data Analysis:

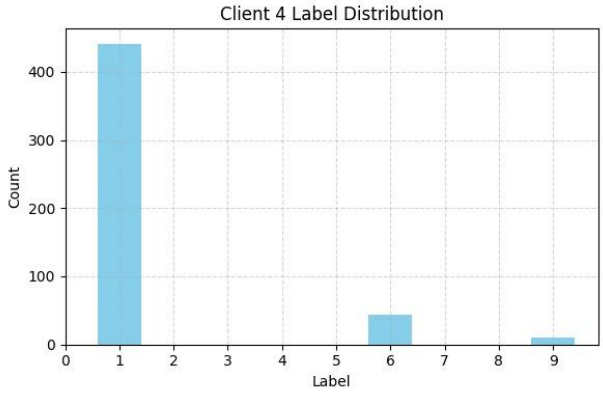
- What is the label distribution (class imbalance) per client?
- How many samples does each client have?
- Are there significant differences in data size or class diversity across clients?

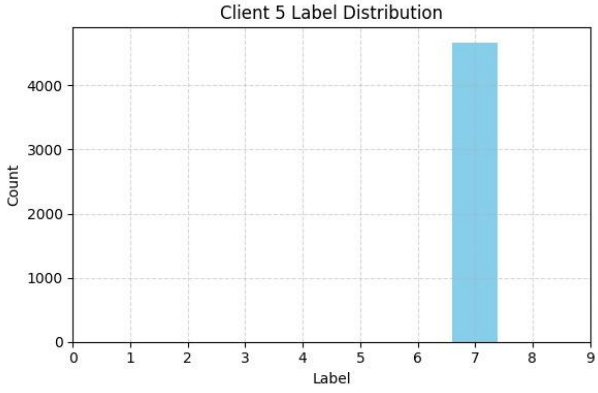


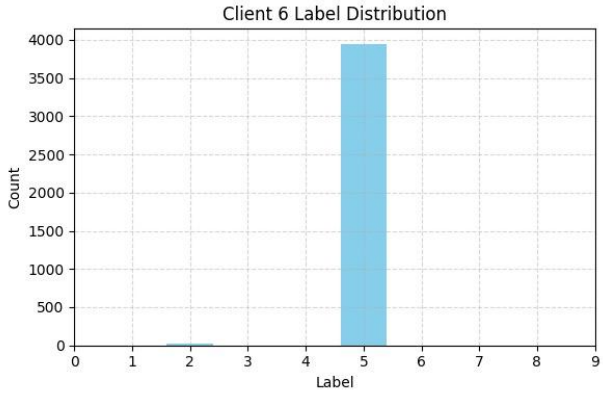


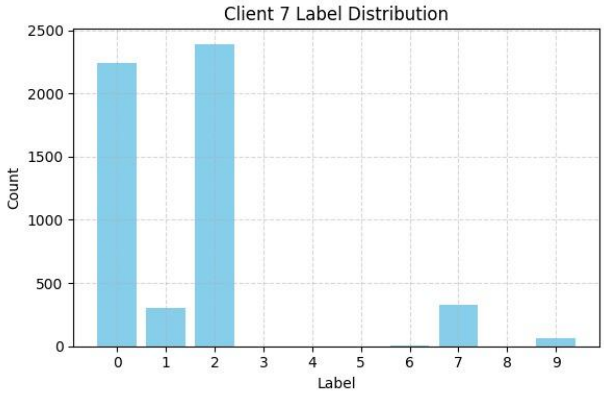


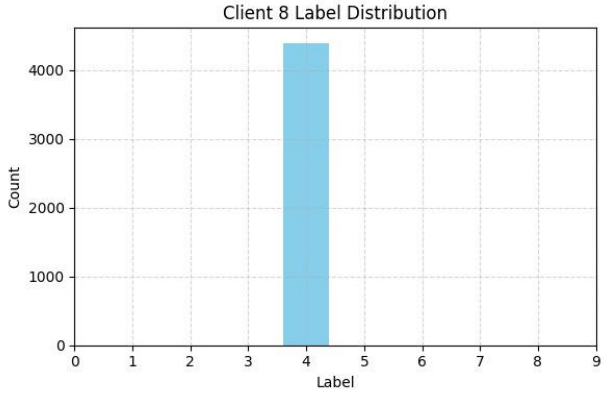


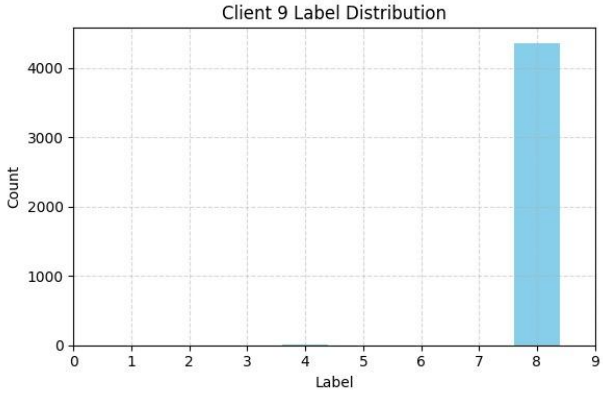


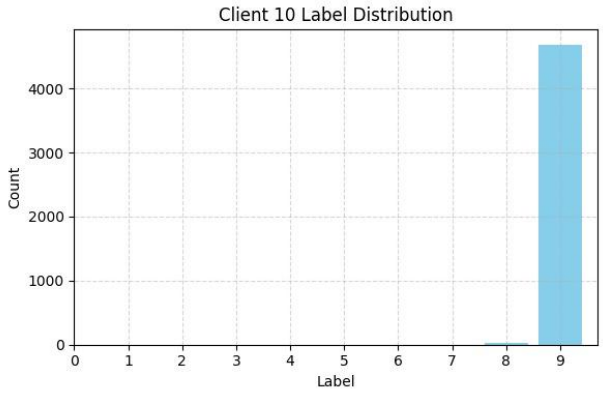


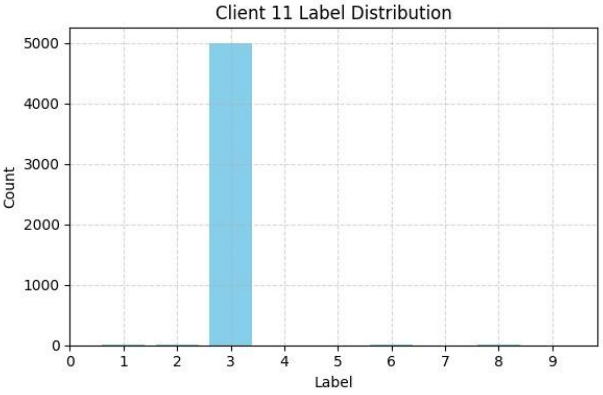








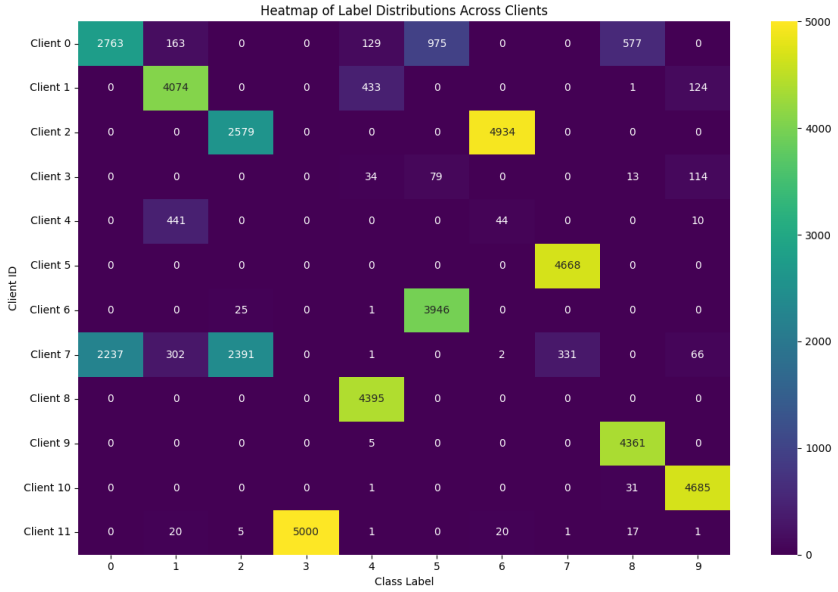




Total number of samples each client has:

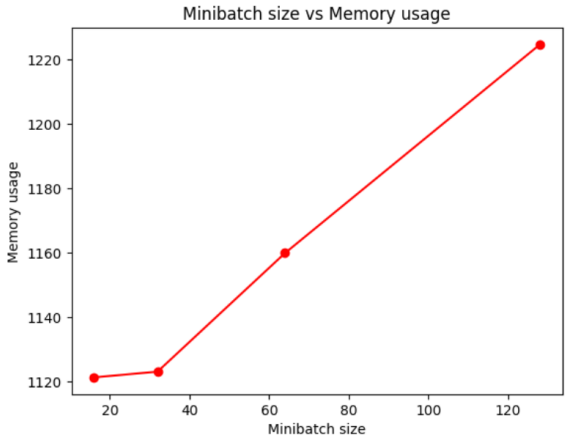
Client ID	Sample count
0	4607
1	4632

2	7513
3	240
4	495
5	4668
6	3972
7	5330
8	4395
9	4366
10	4717
11	5065



2. Profiling Client Training Time

Perform an ablation study by measuring the memory used(using top/htop) and training time per minibatch for different batch sizes for 100 mini-batches of local training.



● Do you see any variability in the minibatch times and memory usage? Why?

Yes, we have seen variability in the minibatch times and memory usage across increasing batch sizes 16, 32, 64, 128. For a higher batch size, the time taken is increasing as the training is being done on larger data; memory usage, on the other hand, is increasing marginally due to the same reason to capture more data in each minibatch.

● What should be the optimal minibatch size to use from what you have observed?

The observed optimal minibatch size is 128.

3. Baseline FL Experiment with Random Client Selection

Using the provided FL simulation code:

● Plug the optimal batch size and the corresponding minibatch timings into the FL simulation framework config for use in the next steps.

● Run 20 rounds of FL with random client selection, selecting 3 clients out of 12 every round. Summarize what you observe—do some clients contribute more to the model's performance? Is training time well-utilized?

Observation : Overall accuracy is increasing, and the accuracy for each label is also improving.

4. Design a Smarter Client Selection Strategy

Devise a new client selection algorithm that aims to maximize accuracy gain per unit time. You can consider factors such as:

● Client training speed (minibatch time)

● Data diversity or label distribution

● Sample size

● Historical contribution to model accuracy

● Whatever else you think will help.

```
def client_selection_random(self, clients, args: dict) -> list:

    acc_per_cl_per_cl -> for each client capture accuracy for each class_label
    time_per_cl -> for each client capture training time

    args contains acc_per_cl_per_cl and time_per_cl

    # Strategy 1
    for each round
      class_label_pick = class_labels[i] where i is increased sequentially
      select 3 clients based
      1. X -> decreasing order of their accuracies for class_label_pick
      2. Y -> increasing order of their training time for class_label_pick

      client_selected[3] = 0.5 * X + 0.5 * Y


    # Strategy 2
    for each round
      class_labels_pick[3] = class_labels[i] where i is increased sequentially but picks 3
       repeat below for 3 times
            select client based
            for each i in class_labels_pick

            1. X -> best based on their accuracies for i
            2. Y -> best based on training time for i

            client_selected = 0.5 * X + 0.5 * Y
```

5. Reflection and Discussion: Prepare a short summary discussing:

● What aspects of client heterogeneity were most critical?

● Why does your strategy outperform (or didn't outperform) random selection?

● Trade-offs between fairness, speed, and accuracy in FL?