

Assignment 4

Group number: 12

Students' Names:

Ahmed Ahmed Alwasefy

Ibrahim Mostafa Elshenhapy

Mohamed Salah Gabr

Date:

2023/07/08

Task 4

1- Modifications in code

1. Please provide images of modifications in code. Highlight the parts you have changed.

File run.py modifications

Q1 and Q2 Code modifications (run.py)

1- Making weighted aggregation with these participation rates

2- Calculate each client's participation rate and print them

```
run.py 7 x
E: > Smart Cities > Assignments > Assignment 4 > Smart_Ass4 > run.py > ...
88
89 #####
90 #1- Finds each Client's data size and print them for proper client.
91 client_data_sizes = {client_name: len(client_data_split[client_name][0]) for client_name in client_list}
92 # Calculate summation of all client data sizes (30000)
93 summation_of_all_client_data_size = sum(client_data_sizes.values())
94 for client_name, data_size in client_data_sizes.items():
95     print("Client {}: Data size = {}".format(client_name, data_size))
96
97 #2- Calculate each client's participation rate and print them
98 #Tip 4- Participation rate for each client can be calculated by: (clients_data_size)/(summation_of_all_client_data_size)
99 participation_rates = {client_name: data_size / summation_of_all_client_data_size for client_name, data_size in client_data_sizes.items()}
100 print("Participation Rates:")
101 for client_name, participation_rate in participation_rates.items():
102     print("{}: {:.4f}".format(client_name, participation_rate))
103
104 #Tip 3- You can write assessment code that make sure all clients participation rate summation is 1.
105 sum_participation_rate = sum(participation_rates.values())
106 print("Sum of Participation Rates: {:.4f}".format(sum_participation_rate))
107 #####
```

Q3 Code modifications (run.py)

```
run.py 7 ●
E: > Smart Cities > Assignments > Assignment 4 > Smart_Ass4 > run.py > ...
154 #Tip 1- For task 1, you need to change federated_averaging(global_model,received_clients) function
155 #and write your own federated averaging function. It can be done by adding additional
156 #parameter that holds clients data size.
157 #Tip 2- New data structure must be created for weighted aggregation. This structure holds data size
158 #for each client. It can be a dictionary that has key indicates client nam and value indicates
159 #data siz. You can use client_data_split dictionary create this new data structure.
160     helper.federated_averaging(global_model, received_clients, client_data_sizes)
161
```

File helper.py modifications

Q3 Code modifications (helper.py)

3- Making weighted aggregation with these participation rates.

```
helper.py 9+ ●
E: > Smart Cities > Assignments > Assignment 4 > Smart_Ass4 > helper.py > federated_averaging
270 #Tip 1- For task 1, you need to change federated_averaging(global_model,received_clients) function
271 #and write your own federated averaging function. It can be done by adding additional
272 #parameter that holds clients data size.
273 #Tip 2- New data structure must be created for weighted aggregation. This structure holds data size
274 #for each client. It can be a dictionary that has key indicates client name and value indicates
275 #data size. You can use client_data_split dictionary create this new data structure.
276 #####
277 def federated_averaging(server, clients, client_data_sizes):
278     target = {name: value.to(device) for name, value in server.named_parameters()}
279     sources = []
280     data_sizes = []
281     #Loop in recived clients
282     for client_name, client_model in clients.items():
283         source = {name: value.to(device) for name, value in client_model.named_parameters()}
284         sources.append(source)
285         #Calculate data_size again but this time save it as tensor
286         #Tried not to repeat this steps but i had some errors and this already working
287         data_size = client_data_sizes[client_name] * torch.ones(1).to(device)
288         data_sizes.append(data_size)
289
290     # Calculate the summation of data size (30000)
291     total_data_size = sum(client_data_sizes.values())
292
293     # Perform weighted aggregation
294     for name in target:
295         # Just loop again to apply the weighted aggregation
296         # by multiply with the weight of data which is (data_size / total_data_size)
297         weighted_aggregation = torch.stack(
298             [source[name].data * (data_size / total_data_size)
299              for source, data_size in zip(sources, data_sizes)])
300         target[name].data = torch.sum(weighted_aggregation , dim=0).clone()
301     #####
```


4- Creating CNN architecture.

```

helper.py 9+
E: > Smart Cities > Assignments > Assignment 4 > Smart_Ass4 > helper.py > federated_averaging
164 # MODEL CREATION
165 #####
166 #Creating CNN architecture.
167 class Net(nn.Module):
168     def __init__(self, num_class, dim):
169         super(Net, self).__init__()
170         #First CNN Layer 1 -> 32 units
171         self.conv1 = nn.Conv2d(1, 32, kernel_size=3, stride=1, padding=1)
172         #Second CNN layer 32 -> 64 units
173         self.conv2 = nn.Conv2d(32, 64, kernel_size=3, stride=1, padding=1)
174         #The 64 unit need to be flatted first (old)
175         # So the 64 * dim[0] divided by 4 and make it floor (use // instead of /)
176         self.flatten_size = 64 * (dim[0]//4) * (dim[1]//4)
177         #First dense layer (Fully Connected) 64 -> 128
178         self.fc1 = nn.Linear(self.flatten_size, 128)
179         #Second and last dense layer will take 128 unit to number of classes
180         #128 -> len(classes)
181         self.fc2 = nn.Linear(128, num_class)
182
183     def forward(self, x):
184         # Apply the first convolutional layer (conv1) with ReLU activation.
185         x = F.relu(self.conv1(x))
186         # Perform max pooling with a kernel size of 2 and stride of 2.
187         x = F.max_pool2d(x, kernel_size=2, stride=2)
188         # Apply the second convolutional layer (conv2) with ReLU activation.
189         x = F.relu(self.conv2(x))
190         # Perform max pooling with a kernel size of 2 and stride of 2.
191         x = F.max_pool2d(x, kernel_size=2, stride=2)
192         # Reshape the tensor by flattening it. (old)
193         x = x.view(-1, self.flatten_size)
194         # Apply the first fully connected layer (fc1) with ReLU activation. (old)
195         x = F.relu(self.fc1(x))
196         # Apply the second and final fully connected layer (fc2).
197         x = self.fc2(x)
198         return x
199 #####

```

2- Report the assignment

In this assignment, we present our findings on the implementation of weighted aggregation in federated learning. We aimed to address the issue of varying client data sizes and their impact on model performance. By assigning participation rates to each client and incorporating them into the aggregation process, we sought to improve the overall accuracy of the model.

1- Client Data Size Calculation:

We iterated through the client list to determine the size of each client's data. We saved this information in a dictionary for future reference.

2- Participation Rate Calculation:

Next, we calculated the participation rate for each client. It was obtained by dividing the client's data size by the sum of all data sizes. We also verified the accuracy of the rates by summing them; they should ideally add up to 1.

3- Weighted Aggregation:

The participation rates obtained were utilized in the weighted aggregation of the model results. The aggregation process ensured that clients with larger data sizes had a proportionally greater impact on the model's outcome. This approach accounted for the data-driven nature of deep learning and mitigated the negative impact of clients with limited data.

4- Model Architecture:

To evaluate the effectiveness of the weighted aggregation, we designed a deep learning model using a Convolutional Neural Network (CNN). The model consisted of two convolutional layers followed by two dense layers. We chose a moderate number of units for each layer to strike a balance between complexity and running time.

Results: We observed the following outcomes during our experimentation:

Accuracy Improvement: When comparing the aggregated results to those of non-aggregated data, we found a notable increase in accuracy. Initially, we trained the model on 10 clients with three rounds, resulting in an accuracy of 93.2%. However, after incorporating weighted aggregation, the accuracy improved to 96.1%. This significant boost indicates the effectiveness of the approach in enhancing model performance.

Impact of CNN Architecture: The implementation of a Convolutional Neural Network (CNN) architecture played a crucial role in further improving the model's accuracy. The CNN architecture allowed the model to learn more complex representations from the data, resulting in improved classification accuracy.

3- Results

Please provide your output (figures, texts, ...):

1. A snapshot of the command line output

Q1

```
Creating Model...
Total Data Size:30000
Client client_1: Data size = 2250
Client client_2: Data size = 750
Client client_3: Data size = 2250
Client client_4: Data size = 750
Client client_5: Data size = 2250
Client client_6: Data size = 750
Client client_7: Data size = 2250
Client client_8: Data size = 750
Client client_9: Data size = 2250
Client client_10: Data size = 750
Client client_11: Data size = 2250
Client client_12: Data size = 750
Client client_13: Data size = 2250
Client client_14: Data size = 750
Client client_15: Data size = 2250
Client client_16: Data size = 750
Client client_17: Data size = 2250
Client client_18: Data size = 750
Client client_19: Data size = 2250
Client client_20: Data size = 750
```

Q2

```
Participation Rates:
client_1: 0.0750
client_2: 0.0250
client_3: 0.0750
client_4: 0.0250
client_5: 0.0750
client_6: 0.0250
client_7: 0.0750
client_8: 0.0250
client_9: 0.0750
client_10: 0.0250
client_11: 0.0750
client_12: 0.0250
client_13: 0.0750
client_14: 0.0250
client_15: 0.0750
client_16: 0.0250
client_17: 0.0750
client_18: 0.0250
client_19: 0.0750
client_20: 0.0250
Sum of Participation Rates: 1.0000
```

2. A snapshot of the Model training and results

For the last Round (Round 9) [Total 10 Rounds 0 ~> 9]

```
Create nodes.
Assign IP Addresses.
Create sockets.
Run Simulation.
Server: 10.2.1.1
Client: 10.1.0.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.1.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.2.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.3.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.4.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.5.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.6.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.7.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.8.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.9.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.10.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.11.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.12.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.13.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.14.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.15.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.16.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.17.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.18.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.19.1 m_peer:03-07-0a:02:01:01:7d:11:00
All Packets Are Sent by Client_1!!!
All Packets Are Sent by Client_2!!!
All Packets Are Sent by Client_3!!!
All Packets Are Sent by Client_4!!!
All Packets Are Sent by Client_5!!!
All Packets Are Sent by Client_6!!!
All Packets Are Sent by Client_7!!!
All Packets Are Sent by Client_8!!!
All Packets Are Sent by Client_9!!!
All Packets Are Sent by Client_10!!!
All Packets Are Sent by Client_11!!!
All Packets Are Sent by Client_12!!!
All Packets Are Sent by Client_13!!!
All Packets Are Sent by Client_14!!!
All Packets Are Sent by Client_15!!!
All Packets Are Sent by Client_16!!!
All Packets Are Sent by Client_17!!!
All Packets Are Sent by Client_18!!!
All Packets Are Sent by Client_19!!!
All Packets Are Sent by Client_20!!!
```

Each client accuracy and the final accuracy after 10 rounds.

```
PYTHON:: Received Client is: 1
Client_1 Accuracy: 0.967667
PYTHON:: Received Client is: 2
Client_2 Accuracy: 0.962333
PYTHON:: Received Client is: 3
Client_3 Accuracy: 0.972333
PYTHON:: Received Client is: 4
Client_4 Accuracy: 0.9695
PYTHON:: Received Client is: 5
Client_5 Accuracy: 0.969167
PYTHON:: Received Client is: 6
Client_6 Accuracy: 0.968
PYTHON:: Received Client is: 7
Client_7 Accuracy: 0.969333
PYTHON:: Received Client is: 8
Client_8 Accuracy: 0.9685
PYTHON:: Received Client is: 9
Client_9 Accuracy: 0.966167
PYTHON:: Received Client is: 10
Client_10 Accuracy: 0.969
PYTHON:: Received Client is: 11
Client_11 Accuracy: 0.971
PYTHON:: Received Client is: 12
Client_12 Accuracy: 0.9685
PYTHON:: Received Client is: 13
Client_13 Accuracy: 0.968667
PYTHON:: Received Client is: 14
Client_14 Accuracy: 0.97
PYTHON:: Received Client is: 15
Client_15 Accuracy: 0.969833
PYTHON:: Received Client is: 16
Client_16 Accuracy: 0.968167
PYTHON:: Received Client is: 17
Client_17 Accuracy: 0.968833
PYTHON:: Received Client is: 18
Client_18 Accuracy: 0.967333
PYTHON:: Received Client is: 19
Client_19 Accuracy: 0.972333
PYTHON:: Received Client is: 20
Client_20 Accuracy: 0.9645
All Packets are Received!!!
PYTHON:: All Clients are Received - Aggregation Starts!!!
PYTHON:: Aggregation is finished - Model Downloading!!!

cleaning

Overall Accuracy Result: 0.977
shenhapy@shenhapy:~/Downloads/ns-allinone-3.35/ns-3.35/scratch/Tutorial_81$
```

Overall Accuracy Result is 0.977

3. Log for all the Overall Accuracy Result for the 10 rounds

Round 0: Overall Accuracy Result: 0.912167

Round 1: Overall Accuracy Result: 0.9495

Round 2: Overall Accuracy Result: 0.960167

Round 3: Overall Accuracy Result: 0.965167

Round 4: Overall Accuracy Result: 0.969

Round 5: Overall Accuracy Result: 0.9715

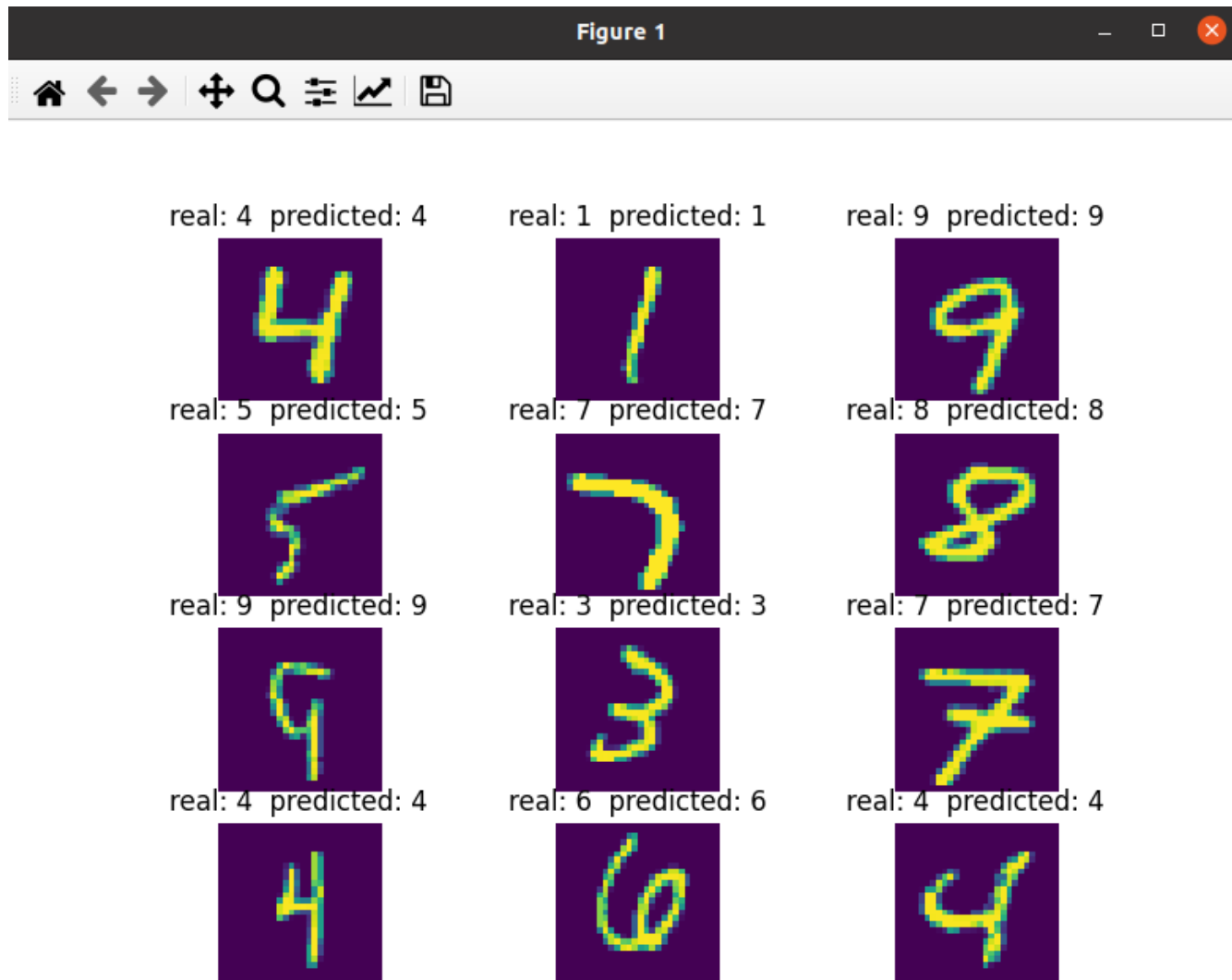
Round 6: Overall Accuracy Result: 0.973667

Round 7: Overall Accuracy Result: 0.975167

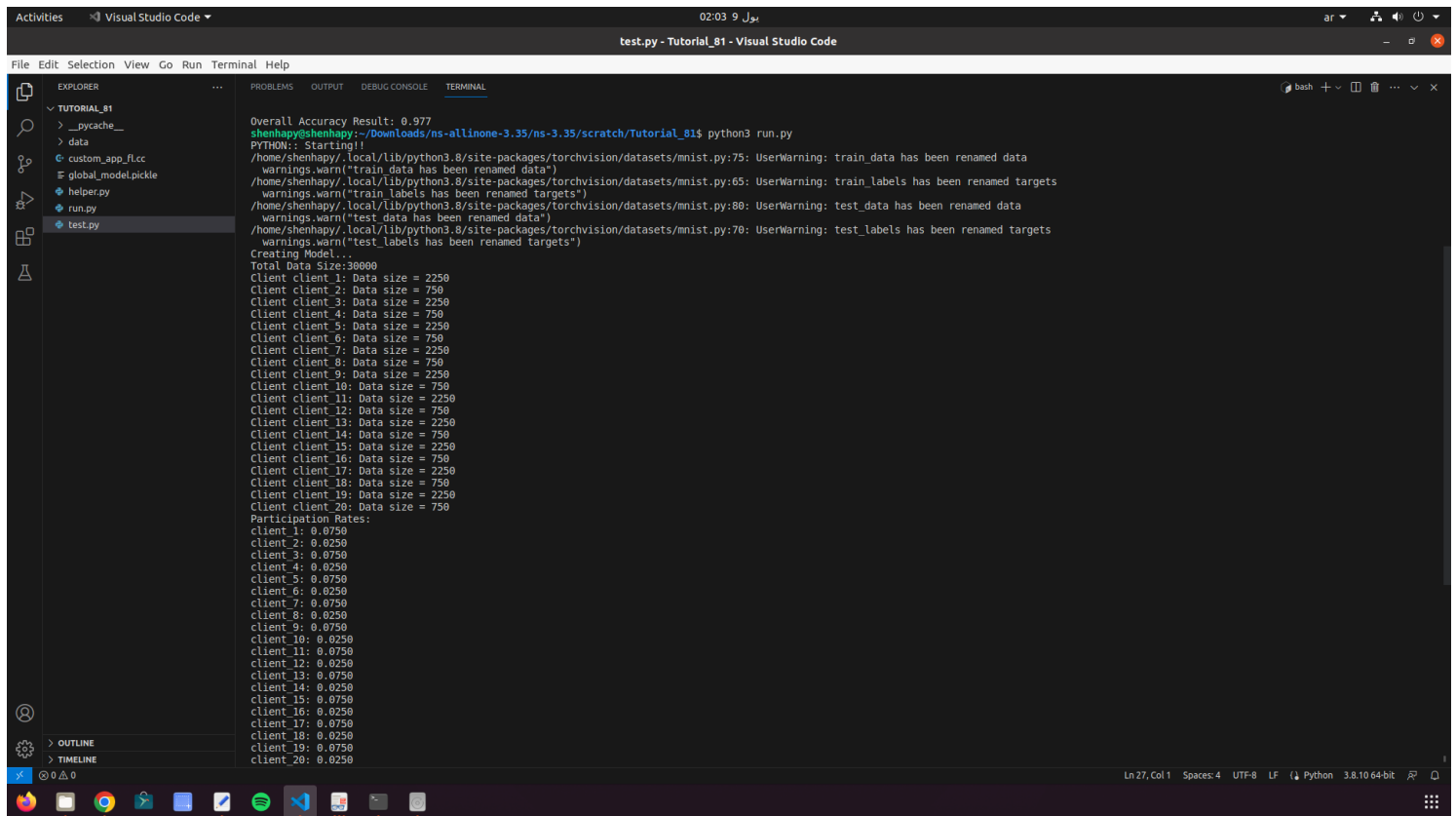
Round 8: Overall Accuracy Result: 0.976333

Round 9: Overall Accuracy Result: 0.977

4. Test Results (test.py)



5. Whole Screen screenshot



```
Activities Visual Studio Code 02:03 9 بول ar test.py - Tutorial_81 - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER TUTORIAL_81
  > __pycache__
  > data
  custom_app_fl.cc
  global_model.pickle
  helper.py
  run.py
  test.py

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
bash
Overall Accuracy Result: 0.977
shenhappy@shenhappy:~/Downloads/ns-allinone-3.35/ns-3.35/scratch/Tutorial_81$ python3 run.py
PYTHON: Starting!
/home/shenhappy/.local/lib/python3.8/site-packages/torchvision/datasets/mnist.py:75: UserWarning: train_data has been renamed data
warnings.warn("train_data has been renamed data")
/home/shenhappy/.local/lib/python3.8/site-packages/torchvision/datasets/mnist.py:65: UserWarning: train_labels has been renamed targets
warnings.warn("train_labels has been renamed targets")
/home/shenhappy/.local/lib/python3.8/site-packages/torchvision/datasets/mnist.py:80: UserWarning: test_data has been renamed data
warnings.warn("test_data has been renamed data")
/home/shenhappy/.local/lib/python3.8/site-packages/torchvision/datasets/mnist.py:70: UserWarning: test_labels has been renamed targets
warnings.warn("test_labels has been renamed targets")
Creating Model...
Total Data Size: 30000
Client client 1: Data size = 2250
Client client 2: Data size = 750
Client client 3: Data size = 2250
Client client 4: Data size = 750
Client client 5: Data size = 2250
Client client 6: Data size = 750
Client client 7: Data size = 2250
Client client 8: Data size = 750
Client client 9: Data size = 2250
Client client 10: Data size = 750
Client client 11: Data size = 2250
Client client 12: Data size = 750
Client client 13: Data size = 2250
Client client 14: Data size = 750
Client client 15: Data size = 2250
Client client 16: Data size = 750
Client client 17: Data size = 2250
Client client 18: Data size = 750
Client client 19: Data size = 2250
Client client 20: Data size = 750
Participation Rates:
client 1: 0.0750
client 2: 0.0250
client 3: 0.0750
client 4: 0.0250
client 5: 0.0750
client 6: 0.0250
client 7: 0.0750
client 8: 0.0250
client 9: 0.0750
client 10: 0.0250
client 11: 0.0750
client 12: 0.0250
client 13: 0.0750
client 14: 0.0250
client 15: 0.0750
client 16: 0.0250
client 17: 0.0750
client 18: 0.0250
client 19: 0.0750
client 20: 0.0250

Ln 27, Col 1 Spaces: 4 UTF-8 LF Python 3.8.10 64-bit
```

4-List of files

Please mention the name of the files you have attached in Brightspace. Also, attach this file and modified code file in Brightspace.

- 1-run_GP12.py
- 1-helper_GP12.py
- 1-test_GP12.py
- 2-Report_GP12.pdf
- global_model.pickle
- custom_app_fl.cc