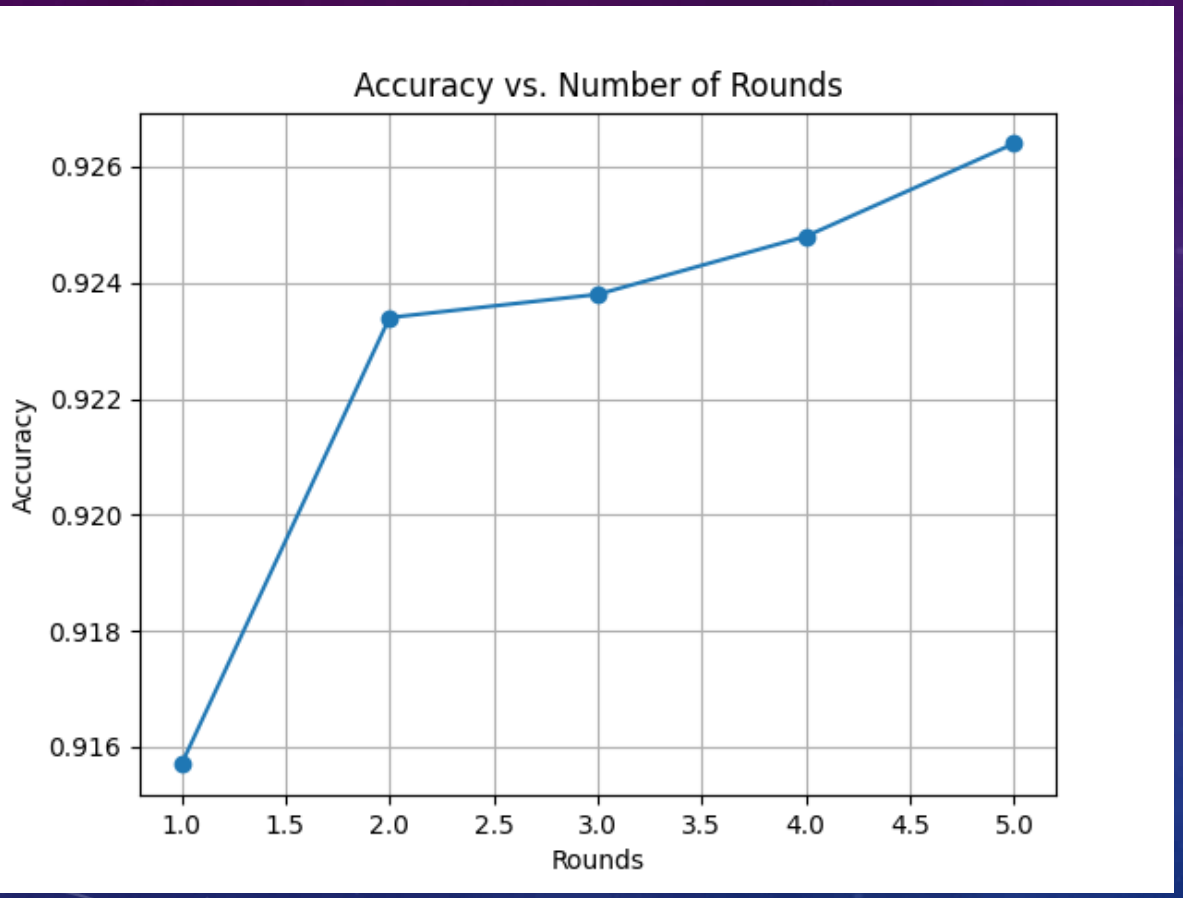
PLOTTING GRAPHS OF ACCURACY VS NUMBER OF CLIENTS

Plot of Accuracy Vs No. of Clients

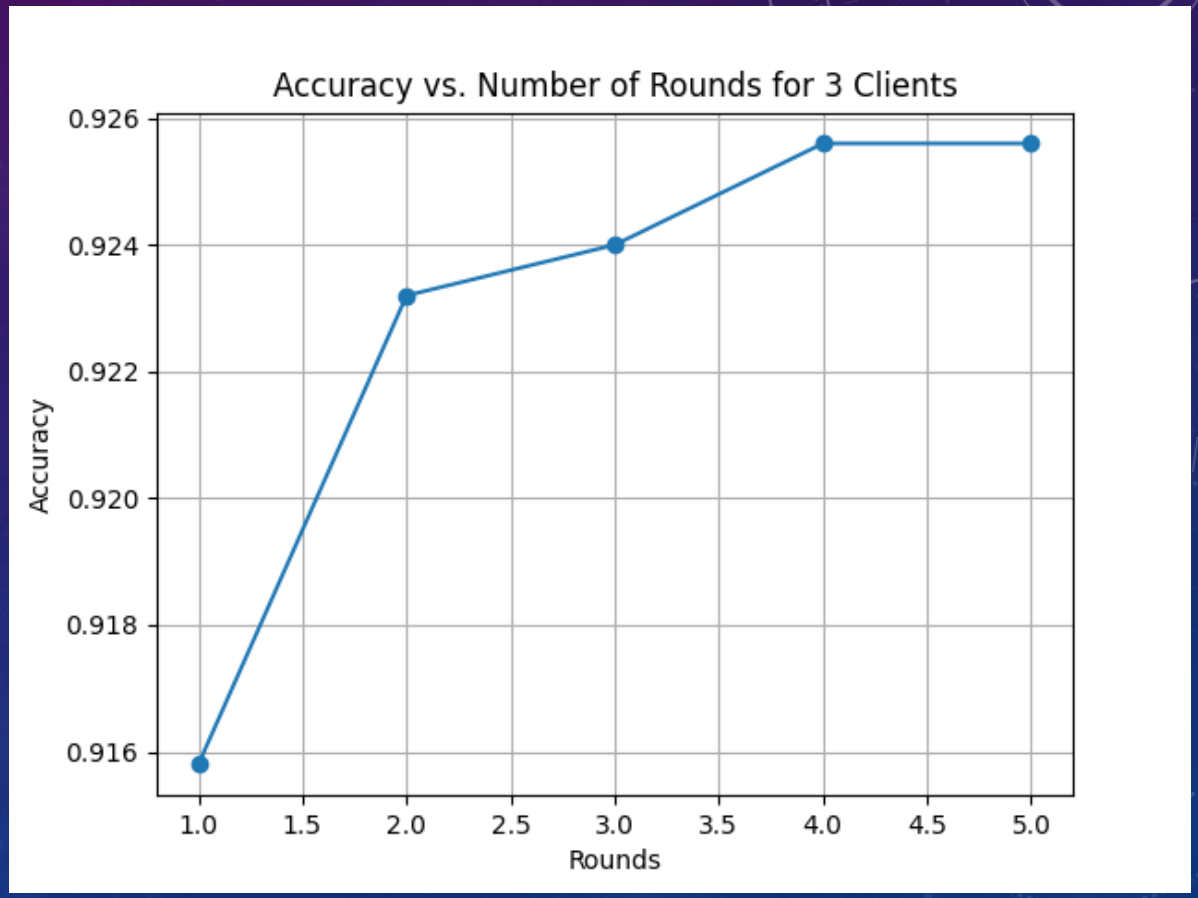
Details:

1. Optimizer used ='adam’,
2. loss='sparse_categorical_crossentropy',
3. metrics=['accuracy’]
4. Number of Rounds = 5 for all cases.
5. Plots only Accuracy for Clients: 2,3,4,5,6,7,8,9,10
6. Aggregation Strategy used="FedAvg"

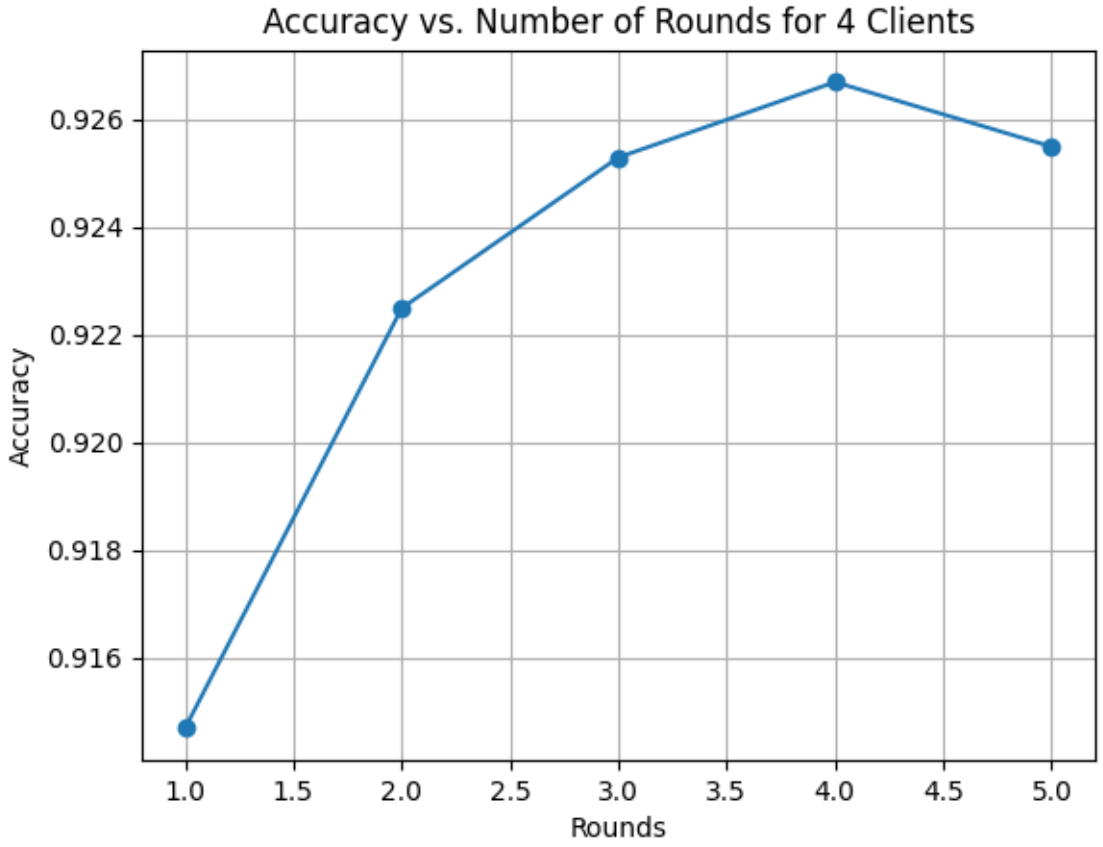
2 CLIENTS



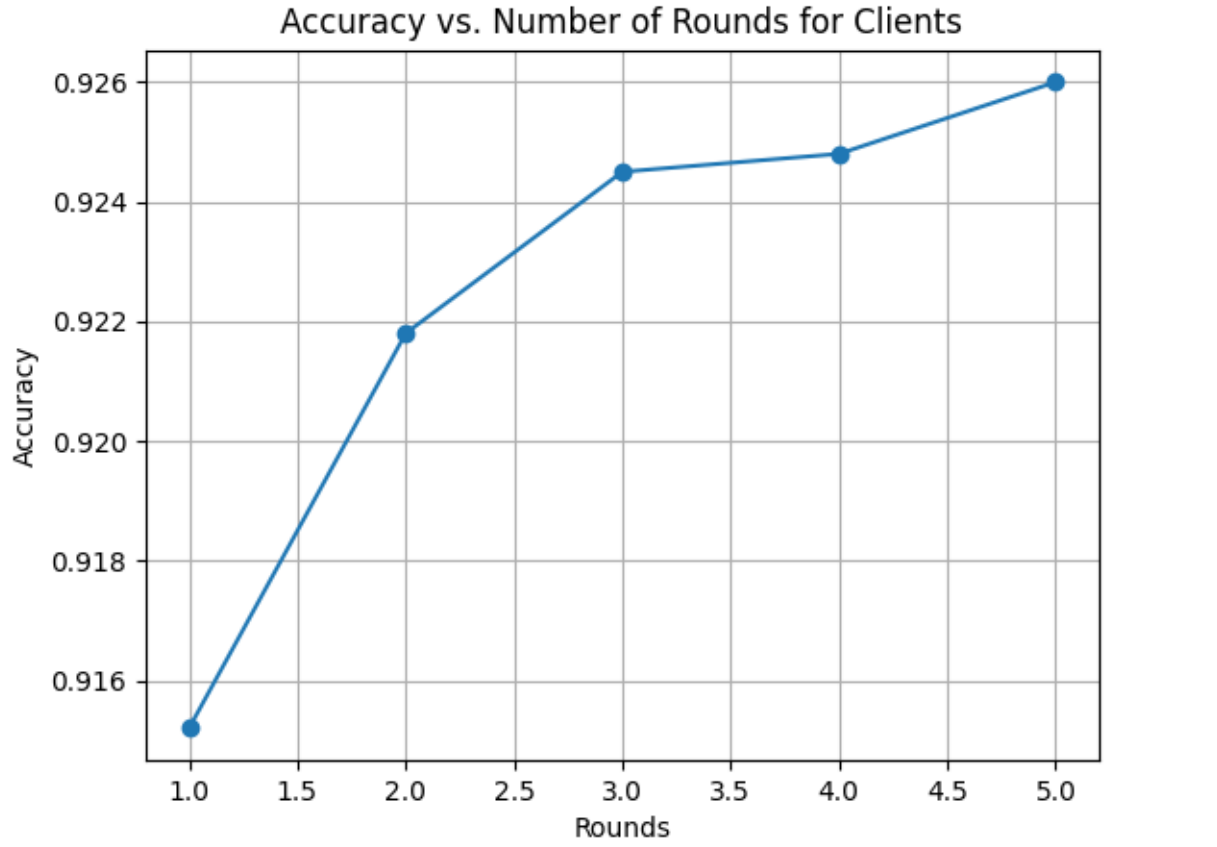
3 CLIENTS



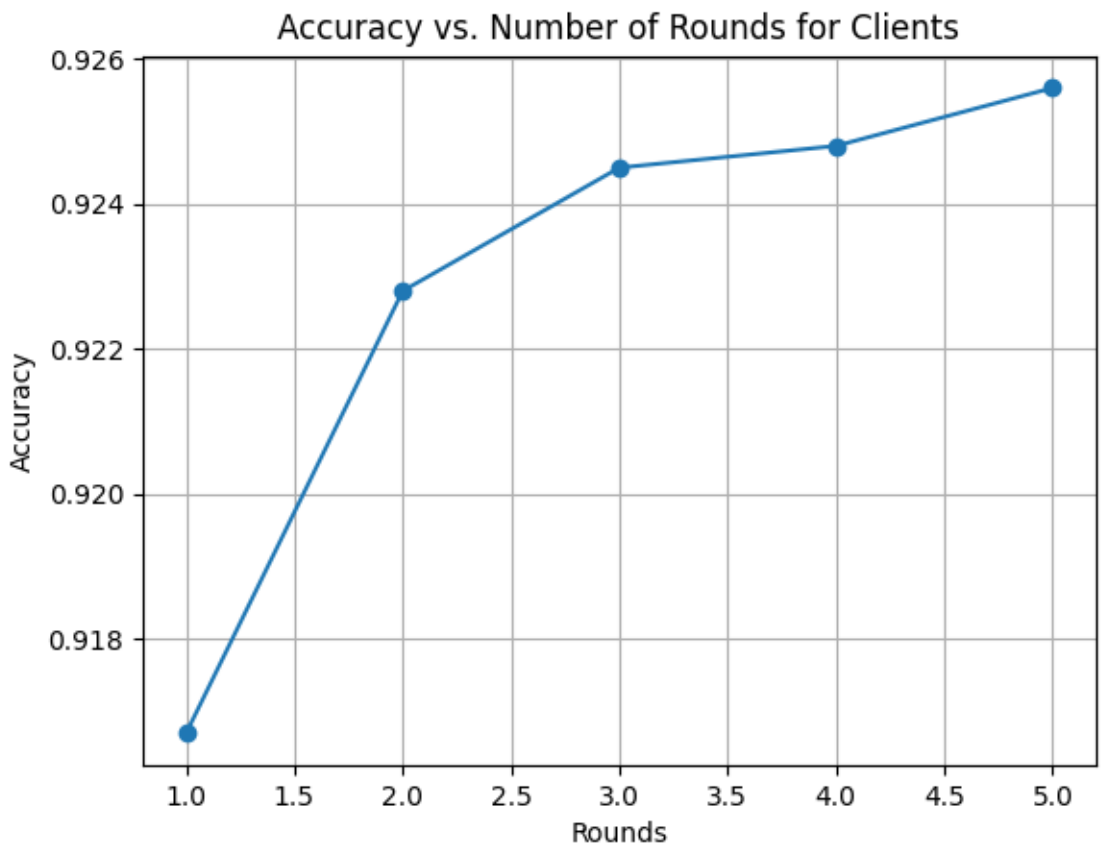
4 CLIENTS



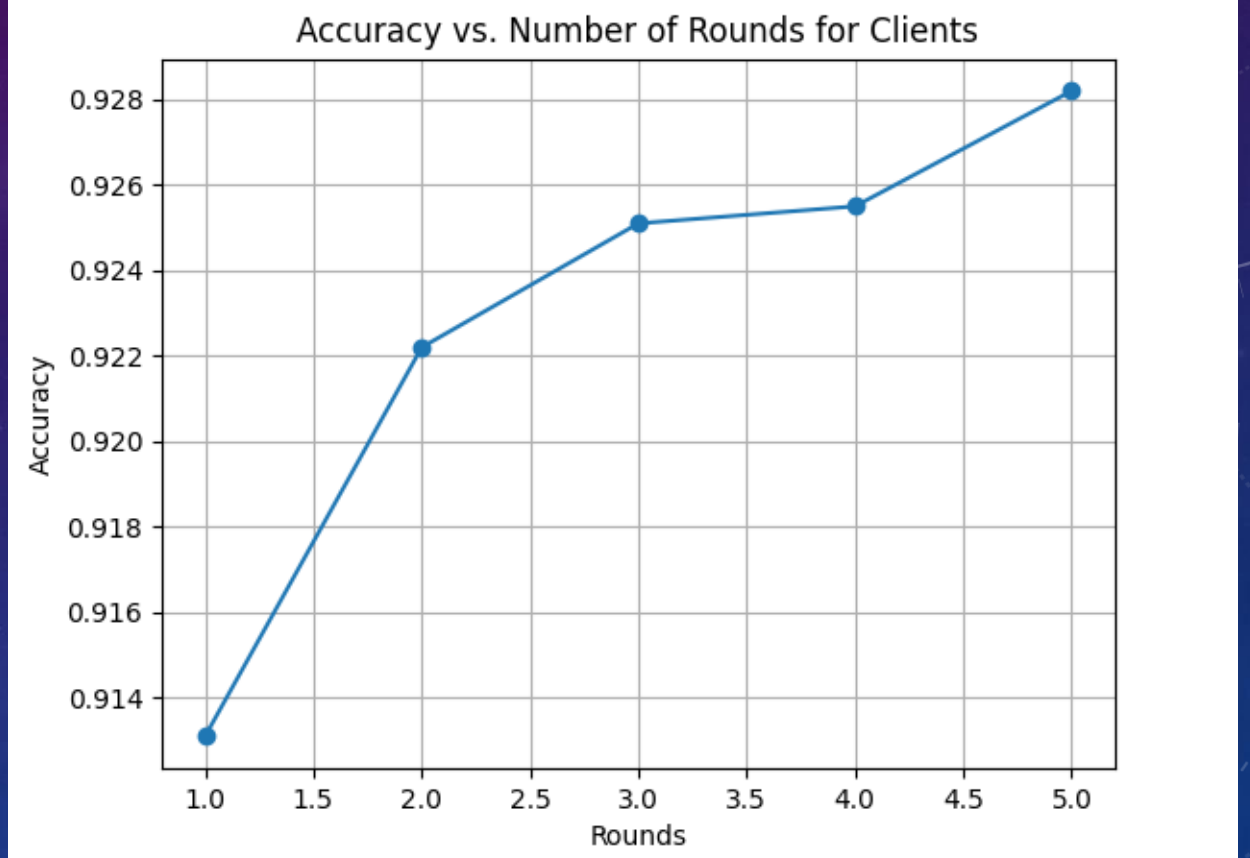
5 CLIENTS



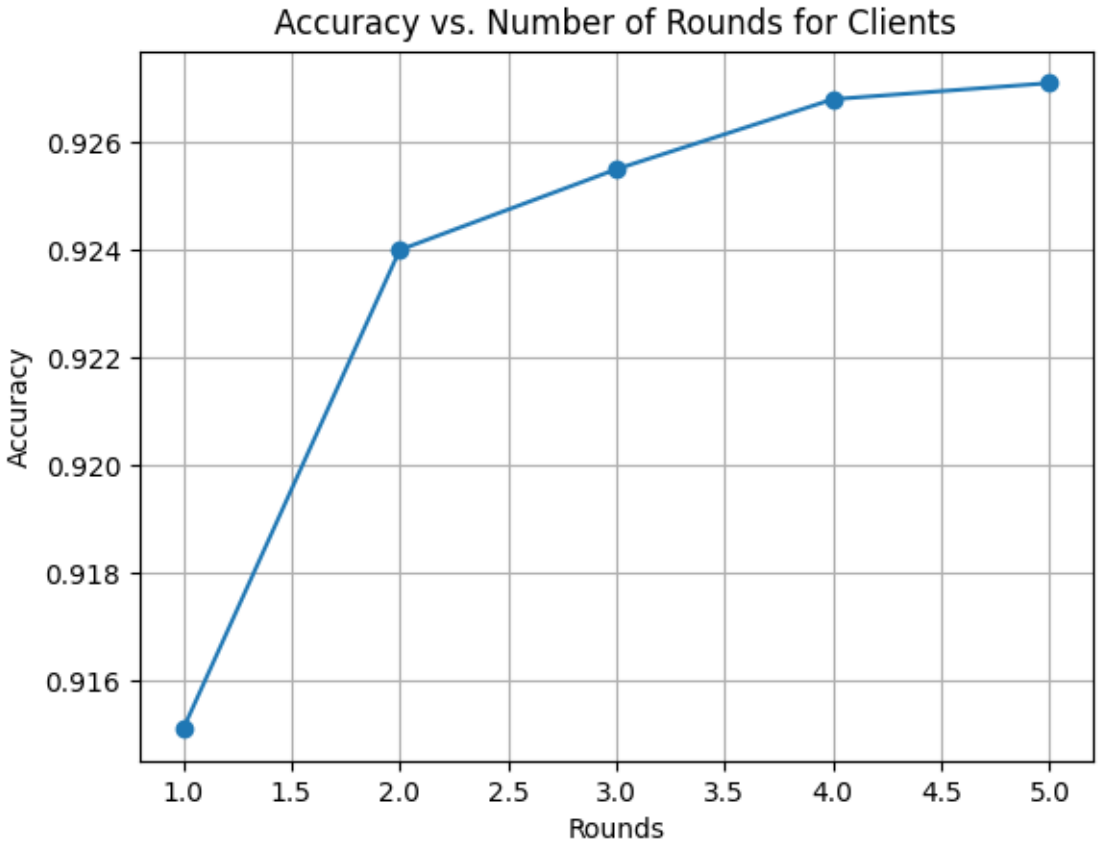
6 CLIENTS



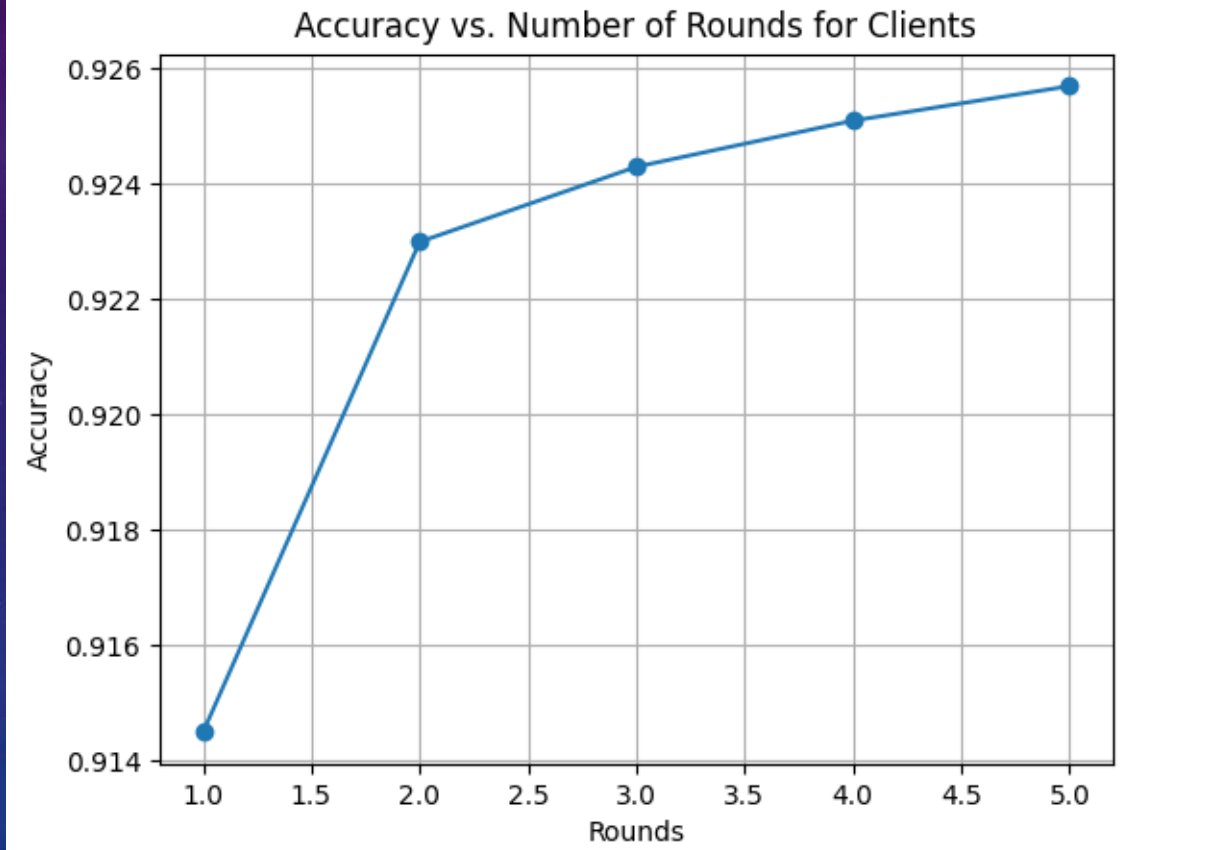
7 CLIENTS



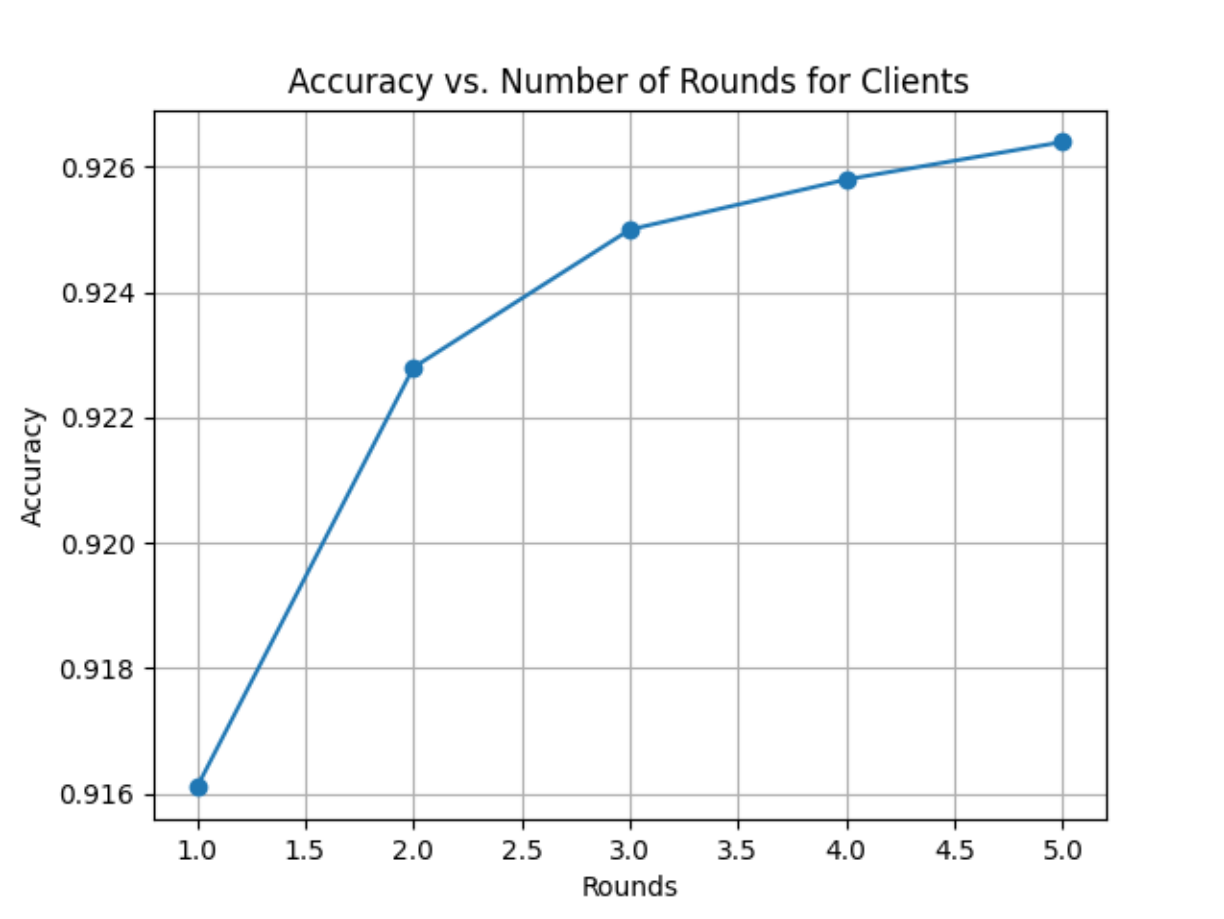
8 CLIENTS



9 CLIENTS



10 CLIENTS


Accuracy vs. Number of Rounds for Clients
Accuracy
0.926
0.924
0.922
0.920
0.918
0.916
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Rounds

KEY OBSERVATIONS

1. For Lesser Number of Clients i.e. 2 to 7, the slope is somewhat zig-zag and undulating. The accuracy suddenly decreases at some point and again increases. The graph is NOT smooth.
2. For higher number of clients(8 to 10), the graph shows lesser changes with increase in rounds. The graph gets SMOOTHER.
3. Accuracy is maximum for 7 clients(0.928) but is near 0.926 for rest graphs.

PROBLEMS FACED

1. Time Complexity Increases with increase in number of Clients.

2. Not all the Clients are able to complete training for all 5 rounds. For example, while training for 8 clients, two clients ended up training for only 4 rounds and one client ended up training for 3 rounds, while the rest could successfully train for 5 rounds.

3. The system slows down for more no. of clients and some clients take too late to connect to the server.

CODE USED:

SERVER CODE:

```
import flwr as fl
strategy=fl.server.strategy.FedAvg()

#Start server
fl.server.start_server(
    server_address="0.0.0.0:8080",
    config=fl.server.ServerConfig(num_rounds=5),
    strategy=strategy,
)
```

CLIENT CODE:

```
# Step 1: Import necessary libraries
import flwr as fl
import tensorflow as tf
import matplotlib.pyplot as plt

# Step 2: Loading dataset
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()

# Step 3: Normalization
x_train, x_test = x_train / 255.0, x_test / 255.0

# Step 4: Define the Model
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(10, activation='softmax')
])

# Step 5: Compile model
model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
```

CLIENT CODE:

```python
# Step 6: Define the Client
class SimpleClient(fl.client.NumPyClient):
    def __init__(self):
        self.accuracy_list = []  # Store accuracy values after each round

    def get_parameters(self, config):
        return model.get_weights()

    def fit(self, parameters, config):
        model.set_weights(parameters)
        model.fit(x_train, y_train, epochs=1, batch_size=32)
        return model.get_weights(), len(x_train), {}

    def evaluate(self, parameters, config):
        model.set_weights(parameters)
        loss, accuracy = model.evaluate(x_test, y_test)
        self.accuracy_list.append(accuracy)  # Append accuracy to the list
        return loss, len(x_test), {"accuracy": accuracy}
```

CLIENT CODE:

```
# Step 7: Start the client and plot accuracy
client = SimpleClient()

def client_training():
    fl.client.start_numpy_client(
        server_address="127.0.0.1:8080",
        client=client,
    )

# Run client training
client_training()

# Step 8: Plot accuracy vs. number of rounds
rounds = range(1, len(client.accuracy_list) + 1)
plt.plot(rounds, client.accuracy_list, marker='o')
plt.title('Accuracy vs. Number of Rounds for 3 Clients')
plt.xlabel('Rounds')
plt.ylabel('Accuracy')
plt.grid(True)
plt.show()
```