Department of Computer Science and Engineering

Activity Report for the course

Mobile Communication Tutorial (MCN21)

Submitted in partial fulfillment of the requirements for the award of degree in

Master of Technology in Computer Science & Engineering

by

Nikhil K Rohidekar 1MS24SCN010

Under the guidance of

Dr. Shilpa S Chaudhari

Associate Professor

M S RAMAIAH INSTITUTE OF TECHNOLOGY

(Autonomous Institute, Affiliated to VTU)

BANGALORE-560054

www.msrit.edu

May 2025

5G Simulation with

Federated Learning-Based Resource Allocation

The goal of this project is to simulate a 5G environment consisting of multiple gNodeBs and user equipment (UEs), generate synthetic network performance data, and build a federated machine learning model to predict the number of allocated Resource Blocks (RBs) to UEs based on quality-of-service (QoS) parameters.

🔷 2. MATLAB Simulation

2.1 Simulation Parameters

- **Number of gNodeBs**: 5
- **UEs per gNodeB**: 10
- **Total UEs**: 50
- **Area**: 100 x 100 unit grid

2.2 gNodeB Deployment

Five gNodeBs are positioned at fixed coordinates to simulate coverage across the area:

```
gNB_pos = [20 20; 20 80; 50 50; 80 20; 80 80];
```

2.3 UE Deployment

UEs are distributed randomly within ±10 units around their associated gNodeB to simulate real-world scattering.

2.4 Metric Generation

Synthetic values are computed for each UE based on distance from its serving gNodeB:

- **RSSI**: Signal strength decreases with distance.
- **SINR**: Decreases with distance plus noise.
- **Throughput**: Decreases with distance.
- **Latency & Jitter**: Increase with distance.
- **HARQ Retransmissions**: Randomized between 0–5.
- **CQI**: Random integers from 1–15.
- **RBs (Target Variable)**: Random but distance-influenced.

2.5 Visualization

A scatter plot shows:

- Red circles: gNodeBs
- Blue dots: UEs
- Dashed lines: Connections
- Labels: Identifiers for gNodeBs and UEs

2.6 Export

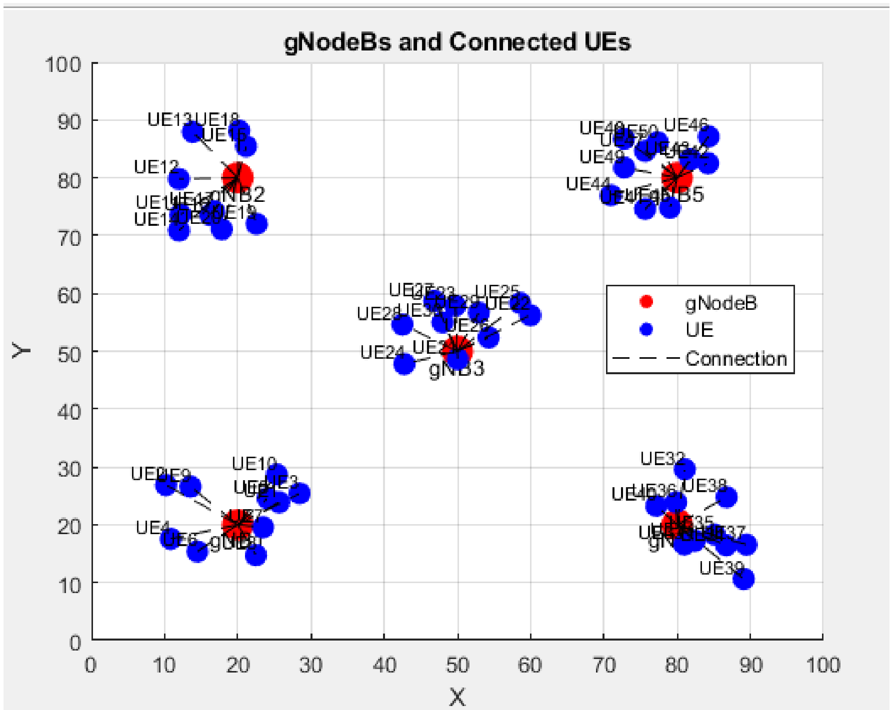
All data is saved into a CSV file:

UE_gNB_metrics.csv

Code:

```
% Number of gNodeBs and UEs per gNodeB

num_gNBs = 5;
ues_per_gNB = 10;
total_UEs = num_gNBs * ues_per_gNB;

% Define gNodeB positions
gNB_pos = [20 20; 20 80; 50 50; 80 20; 80 80];

% Initialize UE positions and connections
UE_pos = zeros(total_UEs, 2);
UE_to_gNB = zeros(total_UEs, 1);

idx = 1;
for g = 1:num_gNBs
   for u = 1:ues_per_gNB
      % Generate UE positions around gNB (±10 unit range)
      UE_pos(idx, :) = gNB_pos(g, :) + (rand(1, 2)-0.5)*20;
      UE_to_gNB(idx) = g;
      idx = idx + 1;
   end
end

% Plotting
figure;
hold on;
title('gNodeBs and Connected UEs');
xlabel('X');
ylabel('Y');
```

```
% Plot gNodeBs
scatter(gNB_pos(:,1), gNB_pos(:,2), 200, 'r', 'filled');
text(gNB_pos(:,1), gNB_pos(:,2), "gNB" + string(1:num_gNBs), ...
    'VerticalAlignment','top','HorizontalAlignment','center', 'FontSize', 10);

% Plot UEs
scatter(UE_pos(:,1), UE_pos(:,2), 100, 'b', 'filled');
for i = 1:total_UEs
    % Draw connection line
    gnb = UE_to_gNB(i);
    plot([gNB_pos(gnb,1), UE_pos(i,1)], [gNB_pos(gnb,2), UE_pos(i,2)], 'k--');
    text(UE_pos(i,1), UE_pos(i,2), "UE" + string(i), ...
        'VerticalAlignment','bottom','HorizontalAlignment','right', 'FontSize', 8);
end

legend('gNodeB', 'UE', 'Connection');
axis([0 100 0 100]);
grid on;
```



UE_gNB_metrics.csv

	A	B	C	D	E	F	G	H	I	J
	UEgNBmetrics									
	UE_ID	gNB_ID	RSSI	SINR	Throughput	RBs	HARQ_retx	CQI	Latency	Jitter
	Number	Number	Number	Number	Number	Number	Number	Number	Number	Number
1	UE_ID	gNB_ID	RSSI	SINR	Throughput	RBs	HARQ_retx	CQI	Latency	Jitter
2	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

🔷 3. Python Machine Learning (Federated Setting)

3.1 Data Preparation

- Loaded the CSV into a Pandas DataFrame.
- Handled missing values (if any) using mean imputation.
- Defined:
  - **Features**: RSSI, SINR, Throughput, HARQ_retx, CQI, Latency, Jitter
  - **Target**: RBs

3.2 Federated Setup

Each gNB_ID was treated as a separate federated learning client. Data was split:

- **70% for training**
- **30% for testing**

3.3 Model Training

- A **Random Forest Regressor** was trained independently on each client's data.
- Model used: n_estimators=50, random_state=client_id

3.4 Federated Aggregation

- Instead of sharing model parameters (true federated learning), predictions from each client model were **averaged (ensemble)** to simulate aggregation.

3.5 Evaluation

- Combined all test sets into a **global test set**.
- Performed ensemble prediction.
- Metrics:
  - **MSE**: 381.61
  - **R^2 Score**: -0.06 → indicates poor model fit

3.6 Visualization

Scatter plot of actual vs. predicted RBs:

- Blue dots: Predictions
- Red dashed line: Ideal prediction line ($y = x$)

- Most predictions were in a narrow range, indicating underfitting.

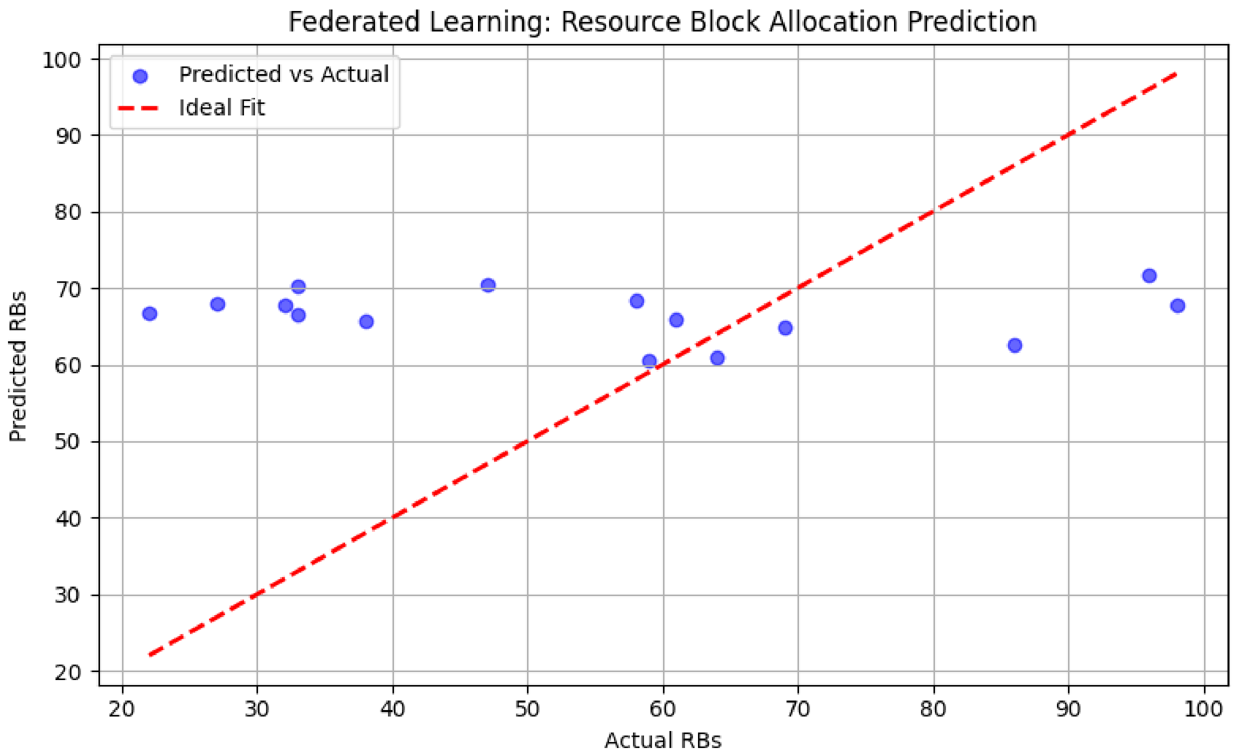
Code :

```
import pandas as pd

import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt

# -----------------------------
# Step 1: Load Data
# -----------------------------
df = pd.read_csv(r"C:\Users\Asus\OneDrive\Documents\MATLAB\New
Folder\Final\UE_gNB_metrics.csv")
# Define features and target
features = ['RSSI', 'SINR', 'Throughput', 'HARQ_retx', 'CQI', 'Latency', 'Jitter']
target = 'RBs'
# Check for missing data
if df.isnull().values.any():
    print("Missing values found. Filling with mean.")
    df.fillna(df.mean(), inplace=True)
# -----------------------------
# Step 2: Split Data by gNB_ID (Federated Clients)
# -----------------------------
clients = {}
for gnb_id in sorted(df['gNB_ID'].unique()):
    df_client = df[df['gNB_ID'] == gnb_id]
    if len(df_client) < 10:
        continue  # Skip small sets
    X_train, X_test, y_train, y_test = train_test_split(df_client[features], df_client[target], test_size=0.3,
random_state=gnb_id)
    clients[gnb_id] = (X_train, X_test, y_train, y_test)

print(f"Total federated clients: {len(clients)}")

# -----------------------------
# Step 3: Train Local Models
# -----------------------------
models = {}
for client_id, (X_train, X_test, y_train, y_test) in clients.items():
    model = RandomForestRegressor(n_estimators=50, random_state=client_id)
    model.fit(X_train, y_train)
```

```python
    models[client_id] = model
    print(f"Client {client_id}: Local model trained")

# -----------------------------
# Step 4: Federated Ensemble Prediction
# -----------------------------
# Combine global test set
global_X = pd.concat([clients[i][1] for i in clients])
global_y = pd.concat([clients[i][3] for i in clients])

# Predict and average from all models
ensemble_predictions = np.mean([model.predict(global_X) for model in models.values()], axis=0)

mse = mean_squared_error(global_y, ensemble_predictions)
r2 = r2_score(global_y, ensemble_predictions)
print(f"\nFederated Ensemble Evaluation:")
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"R2 Score: {r2:.2f}")

# -----------------------------
# Step 5: Visualization
# -----------------------------
plt.figure(figsize=(8, 5))
plt.scatter(global_y, ensemble_predictions, alpha=0.6, c='blue', label='Predicted vs Actual')
plt.plot([global_y.min(), global_y.max()], [global_y.min(), global_y.max()], 'r--', lw=2, label='Ideal Fit')
plt.xlabel("Actual RBs")
plt.ylabel("Predicted RBs")
plt.title("Federated Learning: Resource Block Allocation Prediction")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.savefig("federated_prediction_plot.png")
plt.show()
```

Output:

```
Total federated clients: 5
Client 1: Local model trained
Client 2: Local model trained
Client 3: Local model trained
Client 4: Local model trained
Client 5: Local model trained
Federated Ensemble Evaluation:
Mean Squared Error (MSE): 734.38
R2 Score: -0.29
```



🔷 4. Analysis & Observations

✅ Achievements

- Realistic simulation of UE-gNB interactions.
- End-to-end pipeline from data generation to federated learning.
- Visualization of spatial layout and prediction quality.

🔷 5. Recommendations for Improvement

🔁 Model Enhancements

- Include **distance** as a predictive feature.
- Normalize/standardize features before training.
- Use more powerful models (e.g., XGBoost, LightGBM).
- Hyperparameter tuning with `GridSearchCV`.

🌐 Federated Learning Refinement

- Consider using **Flower** or **FedML** for real federated learning (model parameter aggregation).
- Implement **FedAvg** instead of just averaging predictions.

📊 Evaluation

- Perform per-client evaluation (MSE/R^2 per gNB).
- Add feature importance analysis.

📉 Visual Enhancements

- Use different colors for UEs of different gNodeBs.
- Show coverage range (e.g., circles around gNodeBs).
- Use UE marker size or color to indicate throughput or SINR.

🔷 6. Conclusion

This project demonstrates a practical application of federated learning in a 5G network context, showing how synthetic simulation and decentralized modeling can be integrated. While initial results indicate areas for model improvement, the framework provides a strong foundation for more advanced studies in network optimization, edge intelligence, and federated analytics.

Full code with Documentation

https://github.com/nikhilrohid26/MobileCommunication

Nikhil K Rohidekar