# **Assignment 4 – Privacy-Preserving Federated Learning Simulation (Breast Cancer Dataset)**

## **Objective**

Simulate federated learning across two “hospitals” using the **Breast Cancer Wisconsin (Diagnostic)** dataset without sharing raw data. Add a differential-privacy layer, compare against centralized training, and document privacy/utility trade-offs.

## **Inputs (we provide)**

* Data source: sklearn.datasets.load\_breast\_cancer()
* Feature count: 30 numeric features
* Target: 0 = malignant, 1 = benign (or vice-versa depending on scikit-learn; you must confirm and state it)
* Starter package list: numpy, pandas, scikit-learn, matplotlib, flower (or flwr), torch, opacus

Note: Although this is a breast cancer dataset, you should **frame findings** in a way that could be ported to SeleneX (e.g., how to handle site heterogeneity, privacy budgets, and model monitoring when we later swap in ovarian features).

## **Expected Outputs (deliverables)**

1. **GitHub repo** containing:
   * notebooks/assignment4\_federated\_dp.ipynb (clean, runnable end-to-end)
   * src/ with any helper modules (data split, training loops, DP wrappers)
   * requirements.txt (exact versions)
   * README.md (how to run, environment setup, expected results)
2. **PDF report (3–5 pages)** in the repo and **uploaded to Google Drive**:
   * Methods (data split logic, FL setup, DP params)
   * Results (tables + plots)
   * Discussion (privacy–utility trade-offs, limitations, how to adapt to SeleneX)
3. **45–60 minute video** (headshot, screen share) walking through:
   * Step-by-step code
   * Design choices & alternatives
   * Results & reasoning
   * How you’d adapt this to ovarian multimodal data
4. **One Google Drive folder** named Assignment4\_<YourName> sharing:
   * PDF report
   * Video file or unlisted link
   * Link to GitHub repo
   * Any generated artifacts (e.g., plots)

## **Grading focus**

* Reproducibility (one-click run)
* Correct FL setup (no raw data leakage)
* DP correctly applied and explained (ε, δ, clipping, noise)
* Clear, honest analysis of privacy vs. accuracy
* Code quality, docs, and narrative clarity

## **Step-by-Step Tasks**

### **1) Data loading and EDA (centralized, local only)**

* Load with load\_breast\_cancer(return\_X\_y=True, as\_frame=True).
* Confirm target mapping (document “0=malignant, 1=benign” or vice-versa).
* Basic EDA:
  + Class balance
  + Feature means/SD
  + Correlation heatmap
* **Output**: eda\_summary.md (or section in notebook) + 2–3 plots.

### **2) Create two non-IID hospital splits**

Simulate site heterogeneity so FL is realistic.

* Split logic:
  + Shuffle with fixed seed.
  + **Option A (simple):** 60% of malignant cases go to Hospital A, 40% to B (and vice-versa for benign) to make distributions different.
  + **Option B (feature shift):** Assign by a feature threshold (e.g., mean radius above median → Hospital A; below → B), which induces distribution shift.
* Keep an **external test set** (hold-out 20% of the whole dataset before splitting to hospitals).
* **Outputs**:
  + hospital\_A.csv, hospital\_B.csv, test\_set.csv
  + A small table showing class distribution per hospital.

### **3) Baseline centralized model (reference)**

Train a centralized model on **A ∪ B** (train split only), evaluate on the external test set.

* Models: **Logistic Regression** and **RandomForestClassifier**
* Preprocessing:
  + Train/validation split on the centralized train set.
  + StandardScaler for LR; RF can use raw features.
* Metrics on test set: Accuracy, Precision, Recall, F1, ROC-AUC
* **Outputs**:
  + Table: Centralized LR vs RF metrics
  + ROC curve(s)
  + Save best centralized model as models/centralized\_best.joblib

### **4) Federated learning without DP (Flower)**

* Framework: **Flower (flwr)**
* Architecture:
  + One server process orchestrating training rounds
  + Two clients (Hospital A, Hospital B), each trains **local PyTorch model** on its local data
* Model: a small **MLP in PyTorch** (e.g., 2 hidden layers with ReLU, Dropout)
* Training loop:
  + Rounds: 5–20
  + Local epochs per round: 1–5
  + Optimizer: Adam
  + Loss: BCEWithLogitsLoss
* Aggregation: FedAvg (default in Flower)
* Evaluation: After each round, evaluate the global model on the **external test set** (central process).
* **Outputs**:
  + Plot: Global test ROC-AUC vs. rounds
  + Final metrics on test set
  + Save global model models/fed\_nodp\_final.pt

### **5) Federated learning with Differential Privacy (Opacus)**

* Apply **DP-SGD** on each client during local training:
  + Use **Opacus** to wrap the optimizer
  + Set **clipping norm (C)**, **noise multiplier (σ)**, and **sample rate**
  + Track the **privacy budget (ε, δ)** with Opacus’ accountant; choose δ ≈ 1/N² (document your choice)
* Run same number of FL rounds and local epochs as non-DP run.
* **Outputs**:
  + Final ε reported
  + Plot: test ROC-AUC vs. rounds (DP vs. non-DP vs. centralized)
  + Save model models/fed\_dp\_final.pt

### **6) Comparative analysis**

Produce one concise table (and one chart) comparing **Centralized vs Federated (no-DP) vs Federated (DP)** on the **same external test set**:

* Metrics: Accuracy, Precision, Recall, F1, ROC-AUC
* Training time per epoch/round (rough)
* Privacy: ε (for DP run), clipping norm, noise multiplier
* Discussion prompts:
  + Where did accuracy drop most and why?
  + How sensitive are results to noise multiplier and clipping norm?
  + Would more rounds or more clients help?
  + How would you adapt this to ovarian cancer multi-site training (data schema changes, heterogeneity, monitoring)?

### **7) Reproducibility checklist**

* Fix all random seeds (numpy, torch, sklearn).
* Package versions pinned in requirements.txt.
* README.md with:
  + Setup (venv/conda)
  + How to run centralized baseline
  + How to launch Flower server and clients
  + How to run DP version
  + Expected outputs (example metrics/plots)
* Notebook must run top-to-bottom on a clean machine.

## **Required Plots and Tables**

* Class distribution per hospital
* ROC curves: centralized vs FL no-DP vs FL-DP
* Metric table: all three setups
* FL learning curve: test ROC-AUC vs rounds (with and without DP)
* Optional: calibration plot (reliability curve) for the final global model

## **Hints and pointers**

* Flower quick start: simple NumPy/torch client example → adapt to Pandas/torch tensors.
* Keep models **small** to avoid overfitting on local shards.
* Non-IID split matters: show it with one histogram comparing a key feature across hospitals.
* For Opacus:
  + Start with moderate clipping (e.g., C=1.0) and σ≈1.0–1.3, then tune.
  + Report final ε at the end of training (and how you computed δ).
* Always evaluate on the **same external test set** to keep comparisons fair.

## **What “ideal” looks like**

* Clean repo, single notebook that runs end-to-end.
* Clear, labeled plots and neat tables.
* Thoughtful discussion of privacy–utility trade-offs (not hand-wavy).
* Reproducible numbers with seeds set.
* Clear path to “How this ports to SeleneX with ovarian features and more sites.”