# Federated learning with NVFlare

Kickstarting your data-driven and medical AI projects

November 28th, 2024

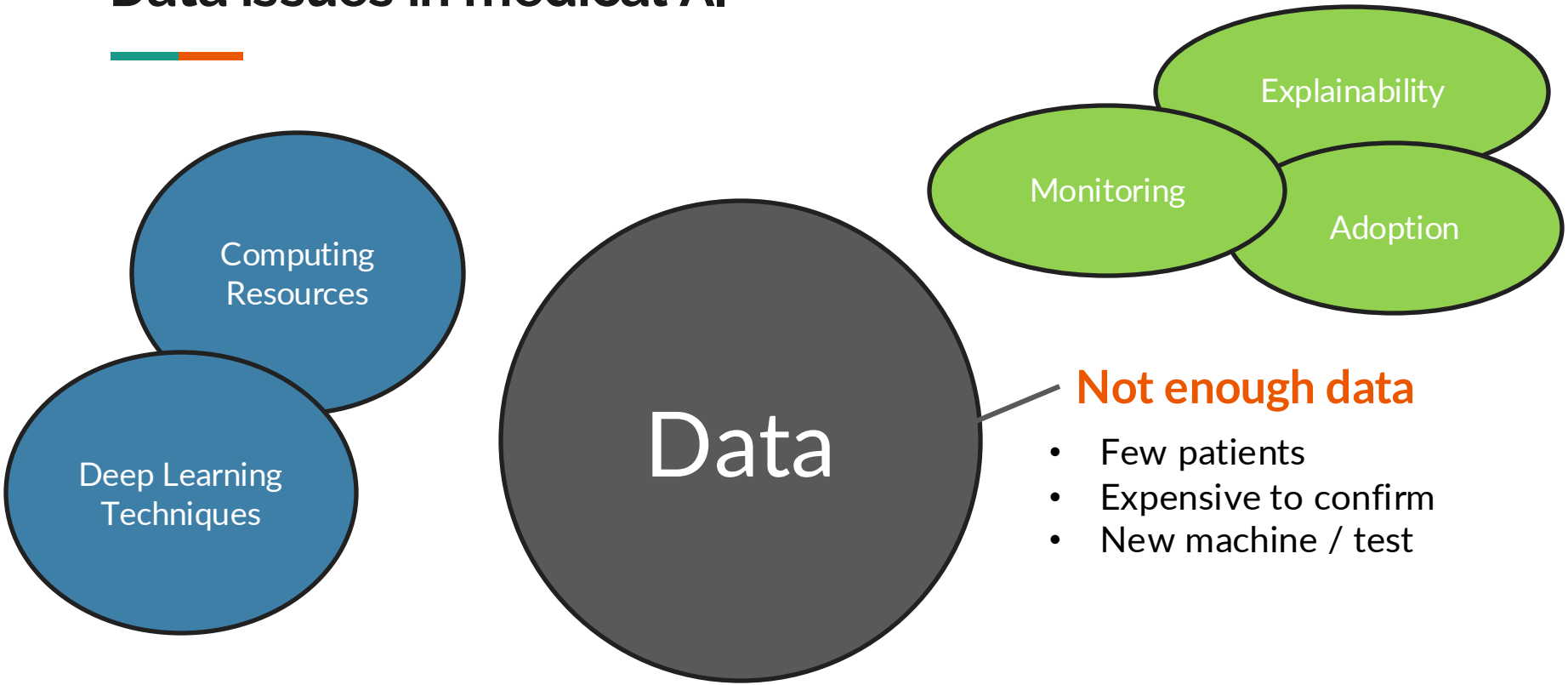


**Sira Sriswasdi, PhD**

- Research Affairs, Faculty of Medicine, Chulalongkorn University
- Computational Molecular Biology Group (CMB)
- Center for Artificial Intelligence in Medicine (CU-AIM)

# Federated learning

# Data issues in medical AI

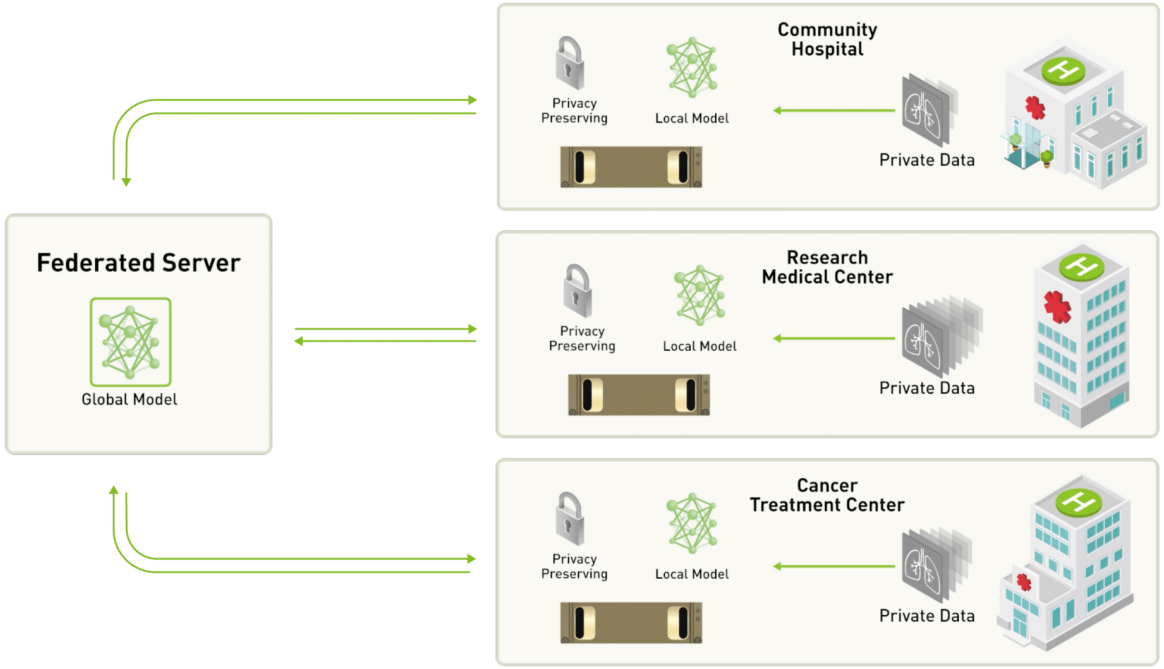



## Not enough data

- Few patients
- Expensive to confirm
- New machine / test

# Federated learning

- Exchange model weights, **not data**
- Preserve privacy, as no data is shared
- A more generalized model, with feedback from every dataset



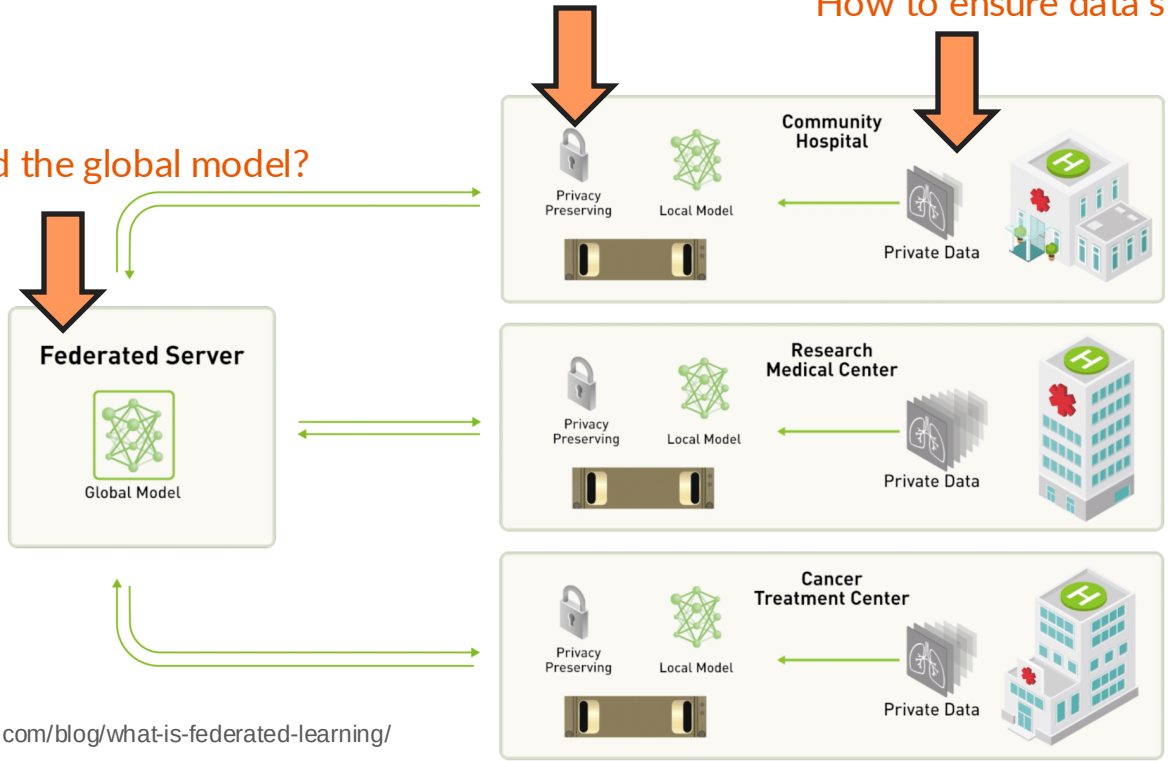

https://blogs.nvidia.com/blog/what-is-federated-learning/

# Key concerns in federated learning

How to ensure security and privacy?

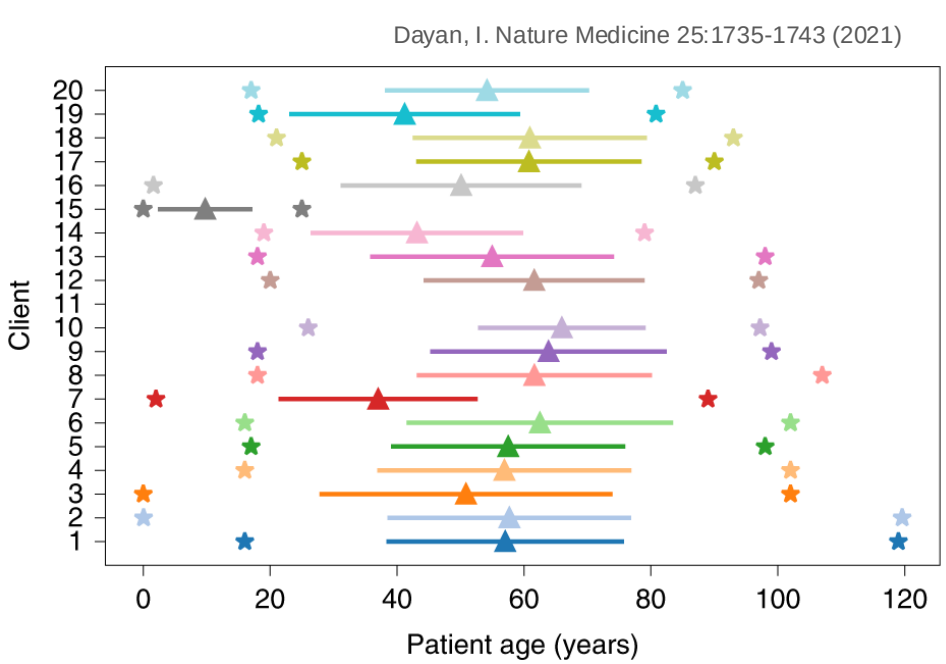
How to ensure data standard and quality?

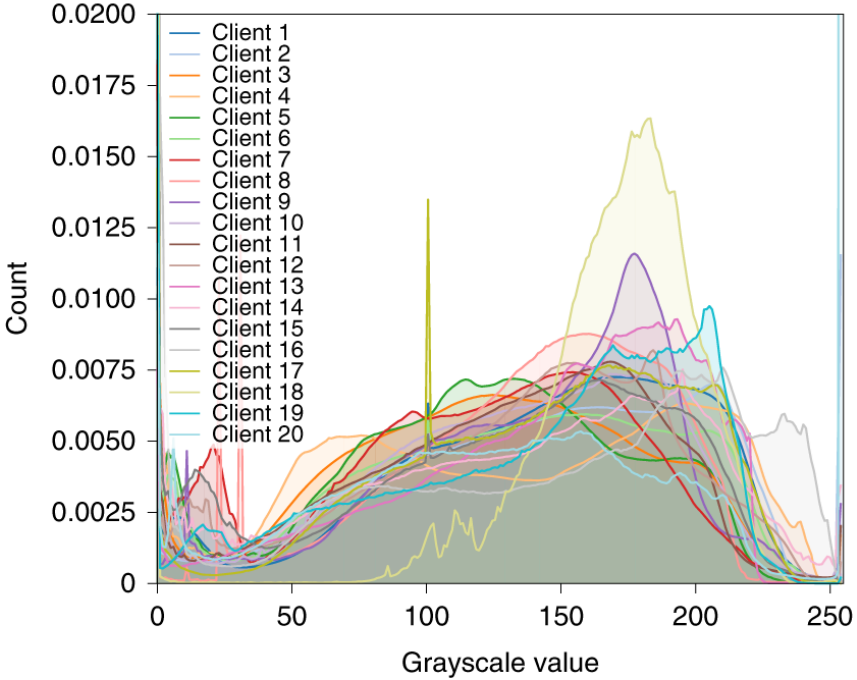
How to build the global model?




https://blogs.nvidia.com/blog/what-is-federated-learning/

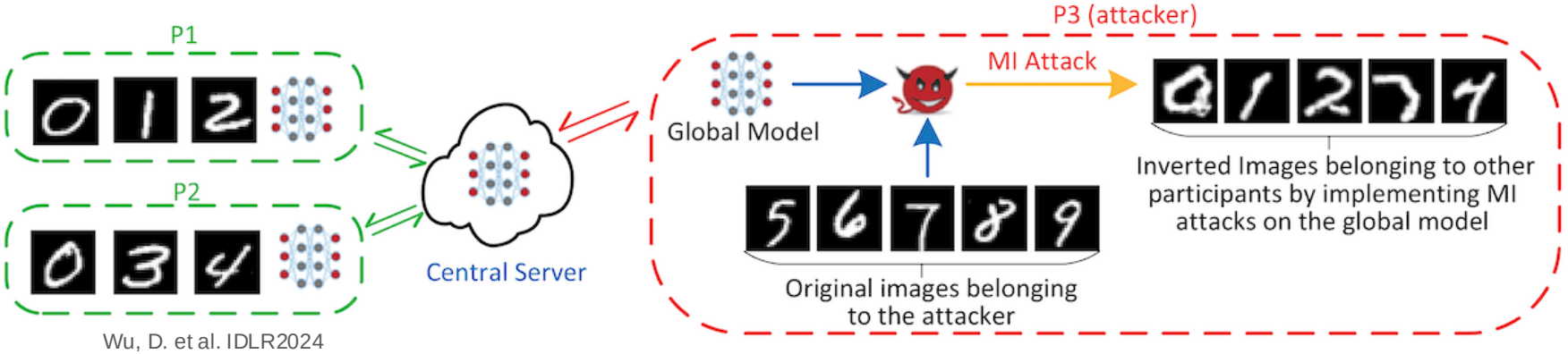
# Data standardization and quality check



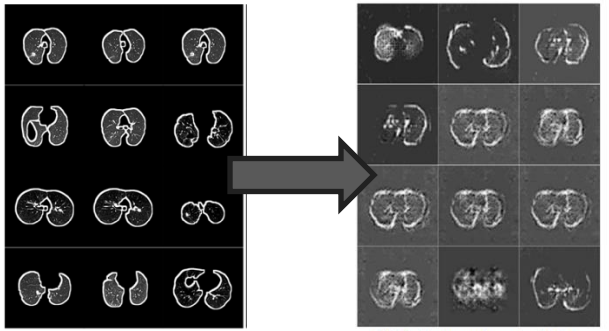

- Perform with every partner before initiating the project

# Security and privacy protection




Wu, D. et al. IDLR2024

- Model inversion attack can partially reconstruct original data
- Partial weights return, add noises

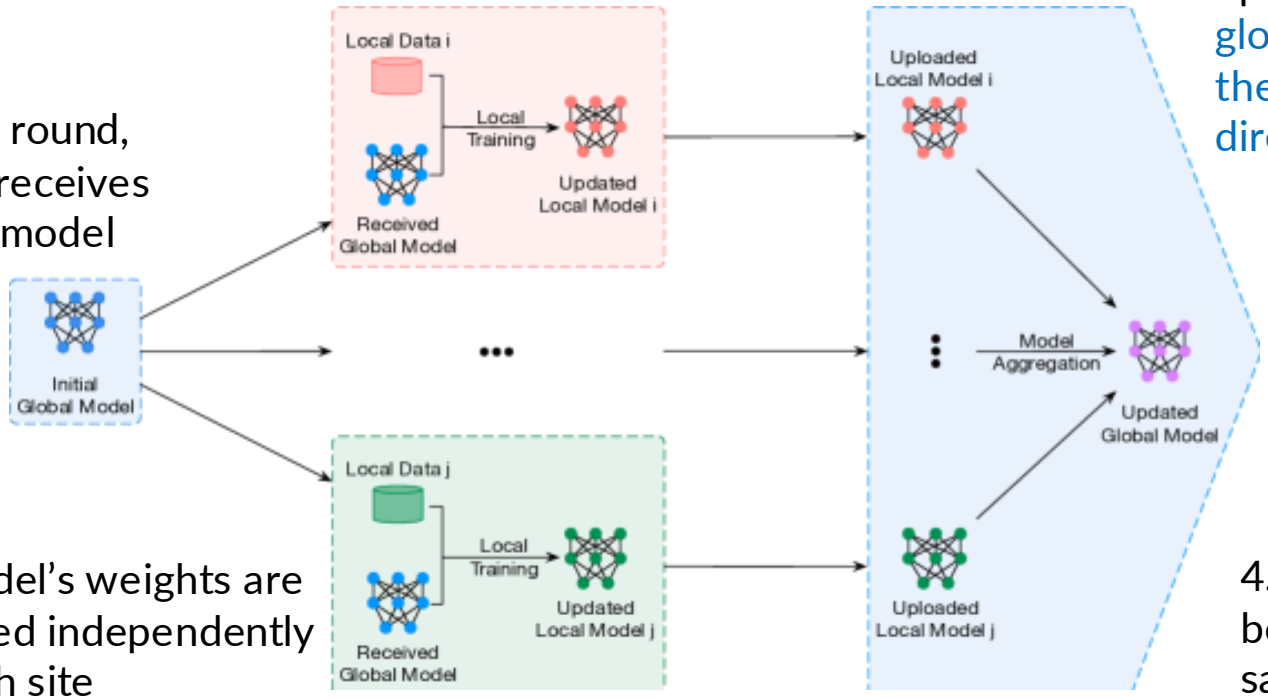


**(b)** Segmented Target Data (Fed-LIDC-IDRI)

**(e)** FedInverse Inverted Data with 100% prior data

# Global model update

1. In each round, each site receives the same model




2. Model's weights are updated independently at each site

3. Averaging the updates push the global model in the good general direction

4. Updates may be weighed by sample size or performance

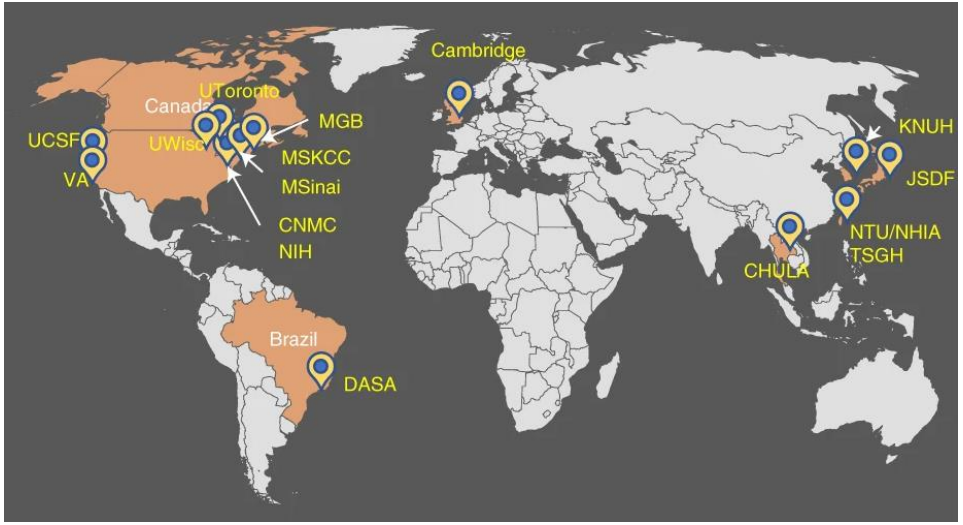
Yao, X. https://arxiv.org/abs/1910.08234 (2020)

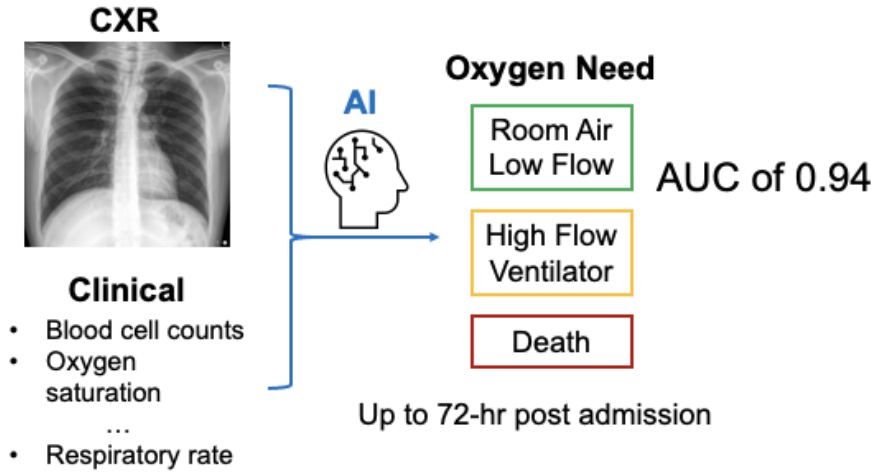
# A global FL for COVID-19

## Triaging COVID-19 Patients: 20 Hospitals in 20 Days Build AI Model that Predicts Oxygen Needs

NVIDIA Clara federated learning predicts requirements without sharing data and builds a more generalizable AI model regardless of geographical location, patient population or data size.

October 5, 2020 by MONA FLORES




Dayan, I. Nature Medicine 27:1735-1743 (2021)

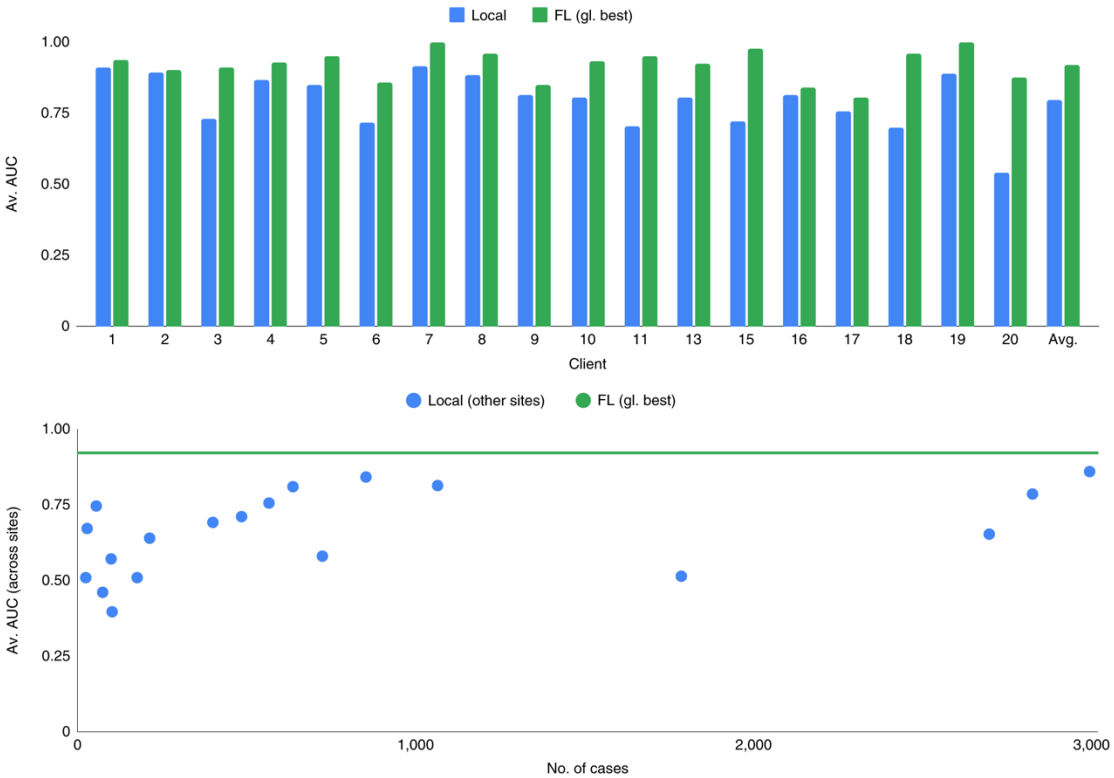
# Combination of clinical data and CXR images

FEAT_VITAL_DBP_FIRST: 54.0
FEAT_VITAL_SBP_FIRST: 136.0
FEAT_PT_AGE: 87
FEAT_LAB_LDH_FIRST: NaN
FEAT_LAB_CRP_FIRST: NaN
FEAT_VITAL_SPO2_FIRST: 97.0
FEAT_VITAL_RR_FIRST: 17.0
FEAT_LAB_AST_FIRST: 26.0
FEAT_LAB_PCLC_FIRST: NaN
FEAT_LAB_LAC_FIRST: NaN
FEAT_LAB_NEUT_FIRST: 4.23
FEAT_LAB_GLU_FIRST: 79.0
FEAT_LAB_WBC_FIRST: 6.34
FEAT_LAB_TNT_FIRST: 16.0
FEAT_LAB_GFR_FIRST: 45.0
FEAT_LAB_CR_FIRST: 1.1
FEAT_LAB_DDMR_FIRST: NaN
FEAT_ED_OD: RA
PCR POS ED: True
PCR POS EVER : True

Dayan, I. Nature Medicine 27:1735-1743 (2021)

- FL is just a framework
- Accommodate any model architecture and data
- Data need to be standardized across participants

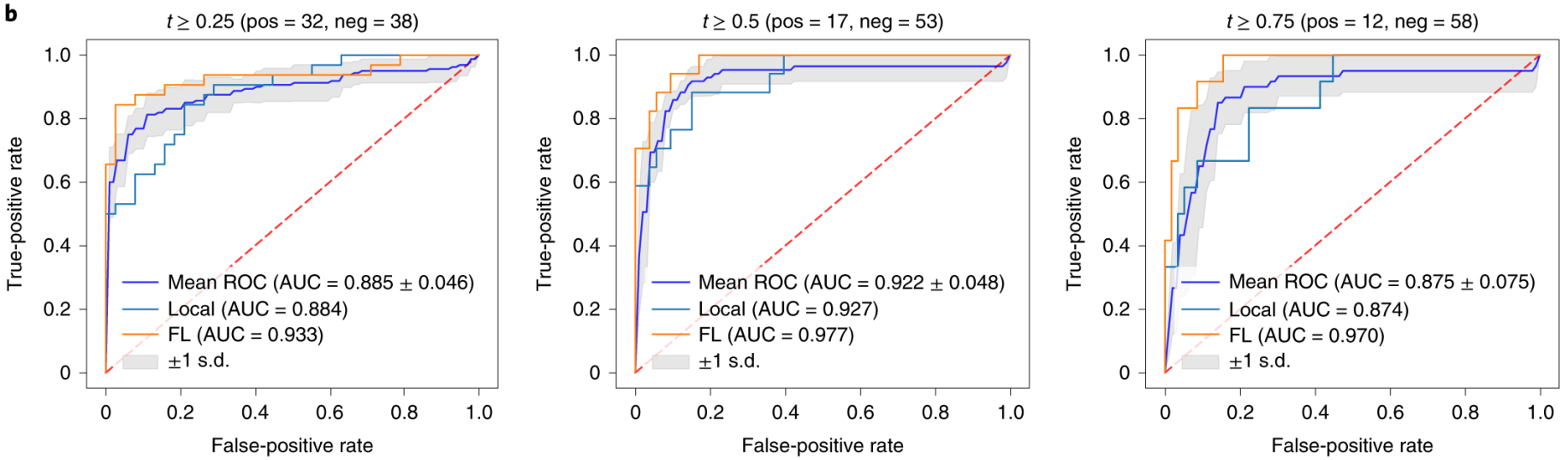
# Benefits of FL




- FL model is beneficial, even for sites with the largest datasets
- FL model performs well on every dataset

Dayan, I. Nature Medicine 27:1735-1743 (2021)

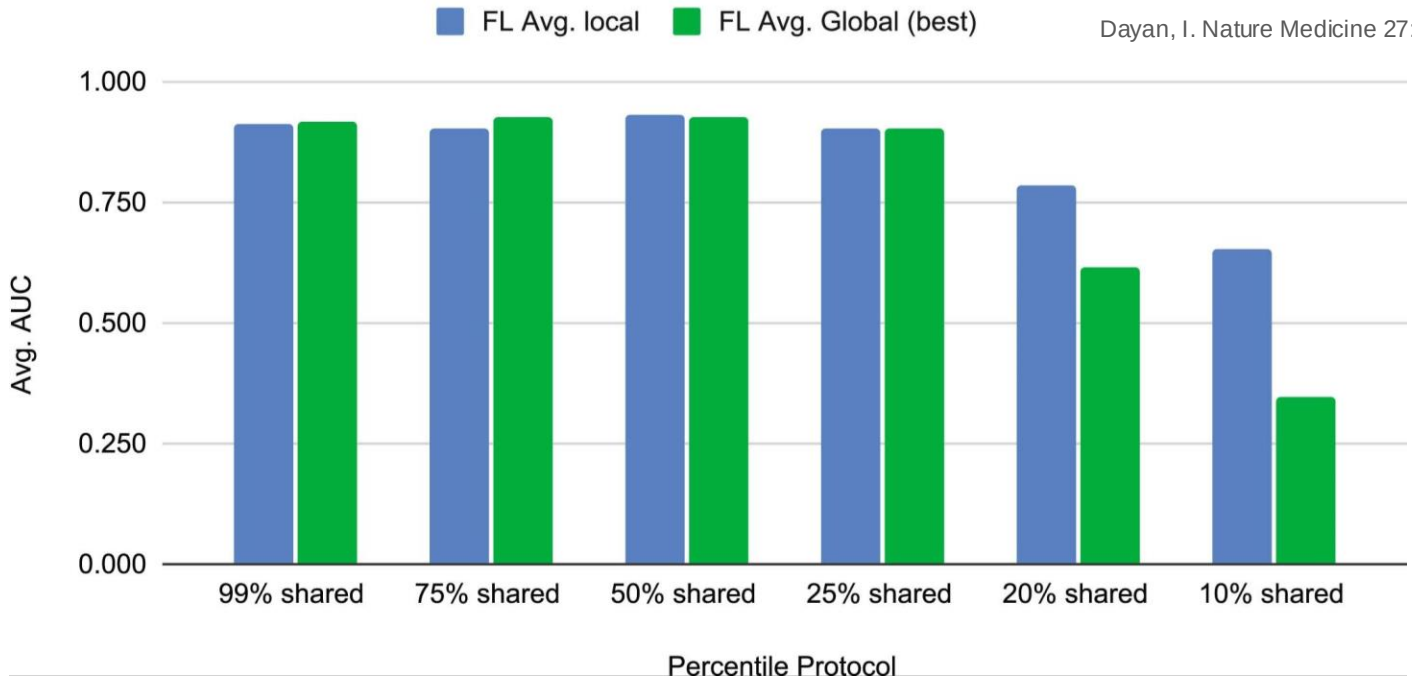
# Generalization of FL




Dayan, I. Nature Medicine 27:1735-1743 (2021)

- ROC of FL vs models from 5 largest sites and locally trained model

# Privacy protection with partial model update




Dayan, I. Nature Medicine 27:1735-1743 (2021)

- Return only top 25% changed weights is enough

# What will you need to do FL?

- Define your ML task and data standard
  - What are the inputs and outputs?
  - Measurement techniques and units
  - Clinical decision guideline
- Identify a good preliminary model
- Find partners (local or online)
- Prepare computing resources (cloud)

**Gold Standard CT Scans of Lung Cancer with Human-Generated Lung-RADS Assessments**

This dataset was generated to create diagnostic algorithms for malignant nodules found in CT scans of the lungs. AI developers can use this multimodal dataset to train or evaluate novel algorithms for radiological oncology.

Note: The image displayed is for illustrative purposes only. Federated Datasets contain a wide range of images and data types.

Discover More Details +

https://www.rhinohealth.com/federated-network

# Take-home messages

- FL does:
  - Resolves limitation in data access / sharing
  - Produce a more generalized model
- FL does not:
  - Replace data curation duty
  - Guarantee privacy and security, by itself
- Being able to anonymize and share data is still the best choice for co-developing medical AI

# NVFlare

# What is NVFlare?

NVIDIA FLARE (NVIDIA Federated Learning Application Runtime Environment) is a domain-agnostic, open-source, extensible SDK that allows researchers and data scientists to adapt existing ML/DL workflows (PyTorch, RAPIDS, Nemo, TensorFlow) to a federated paradigm

- **FL Simulator** for rapid development and prototyping
- **FLARE Dashboard** for simplified project management and deployment
- **Reference FL algorithms** (e.g., FedAvg, FedProx) and workflows (e.g., Scatter and Gather, Cyclic)
- **Privacy preservation** with differential privacy, homomorphic encryption, and more
- **Management tools** for secure provisioning and deployment, orchestration, and management

# How does NVFlare facilitate FL?

## Security & Privacy

- Data encryption
- Spare Vector
- Truncation

## FL Algorithms

- FedAvg
- FedProx
- FedOpt
- SCAFFOLD
- Ditto

## Monitoring & Management

- FL routine
- Cross-site validation
- Admin dashboard

# A look inside NVFlare code: Server-side

```
n_clients = 4                                                    ## NUMBER OF CLIENT
num_rounds = 10                                                  ## NUMBER OF TRAINING ROUNDS
train_script = "nvflare-demo/sklearn-linear/sgd_fl.py”           ## MODEL TRAINING
workspace = "nvflare-demo/sklearn-linear/output"                 ## OUTPUT LOCATION
script_args = "--data_root_dir /content/nvflare-demo/crc-data"   ## PATH TO DATA

initial_params = dict(                                           ## MODEL HYPERPARAMETERS
        n_classes=2, learning_rate="constant", eta0=1e-03,
        loss="log_loss", penalty="l2", fit_intercept=True,
        max_iter=1, random_state=4649)


                                                                 ## DEFINE FL WORKERS
job.to(JoblibModelParamPersistor(initial_params=initial_params), "server")
job.to(FullModelShareableGenerator(), "server")
job.to(InTimeAccumulateWeightedAggregator(expected_data_kind=DataKind.WEIGHTS), "server")

ctrl = ScatterAndGather(min_clients=n_clients,                   ## FL PROCESS
                        num_rounds=num_rounds,
                        start_round=0)
```

# A look inside NVFlare code: Client-side

```
while flare.is_running():                                   ## REPEAT FOR THE DURATION OF PROJECT
    input_model = flare.receive()                           ## RECEIVE MODEL FROM NVFLARE
    global_params = input_model.params
    curr_round = input_model.current_round

    if curr_round == 0:                                     ## INITIALIZE LOCAL MODEL
        model = MLModel()
    else:
        model.coef_ = global_params["coef"]

    model.fit(x_train, y_train)                             ## TRAIN GLOBAL MODEL ON THE LOCAL DATASET
    local_auc, local_report, local_acc = evaluate_model(x_test, model, y_test)

    params = {"coef": model.coef_, "intercept": model.intercept_}
    metrics = {"accuracy": global_auc}
    output_model = flare.FLModel(params=params, metrics=metrics)
    flare.send(output_model)                                ## SEND THE UPDATED MODEL BACK TO NVFLARE
```

# Steps for launching an FL project

## 1. Feasibility

Gather local and public datasets

Develop local models to identify the best model architectures and hyperparameters

Define data and label standard

## 2. Local Testing

Use NVFlare's *Simulator* to test the FL scripts and workflow locally

## 4. Provisioning

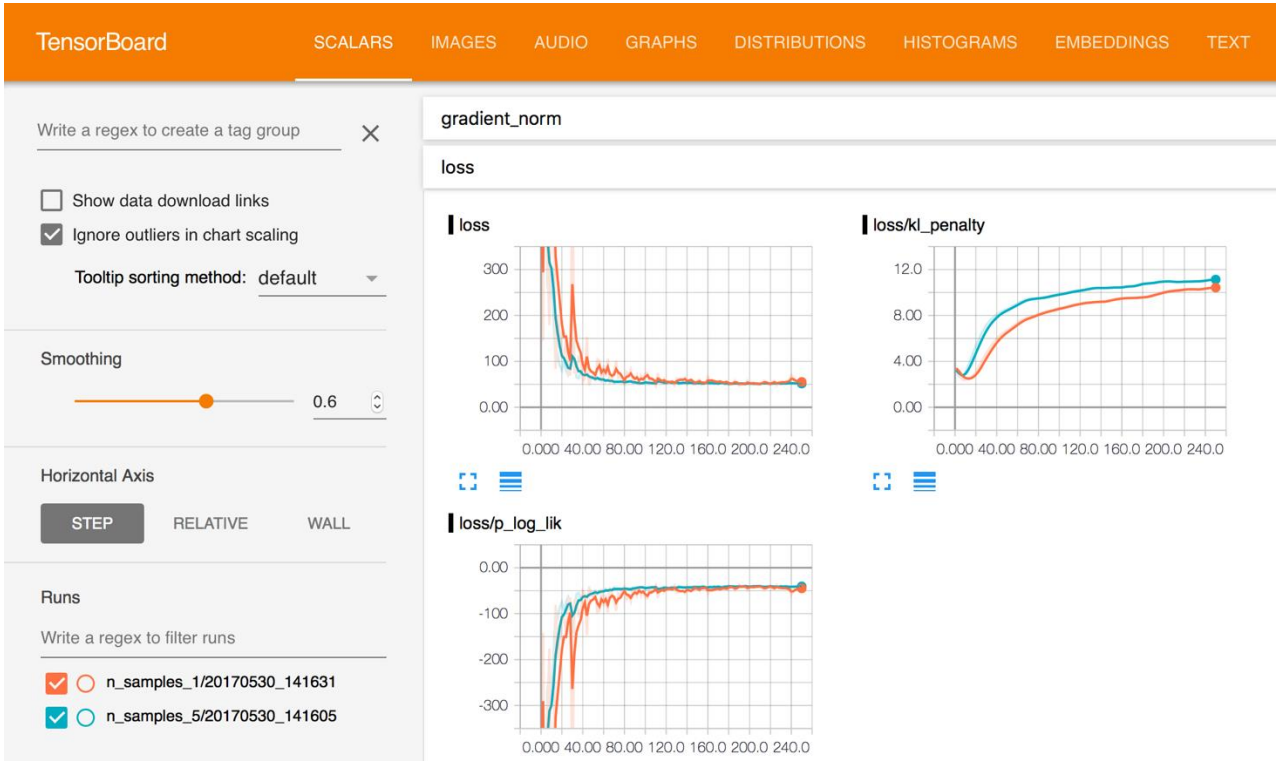
Initiate each client site and share FL scripts

With the help of local IT

## 3. Participant Recruitment

Identify people with similar interest and data, and assess their IT and computing capabilities

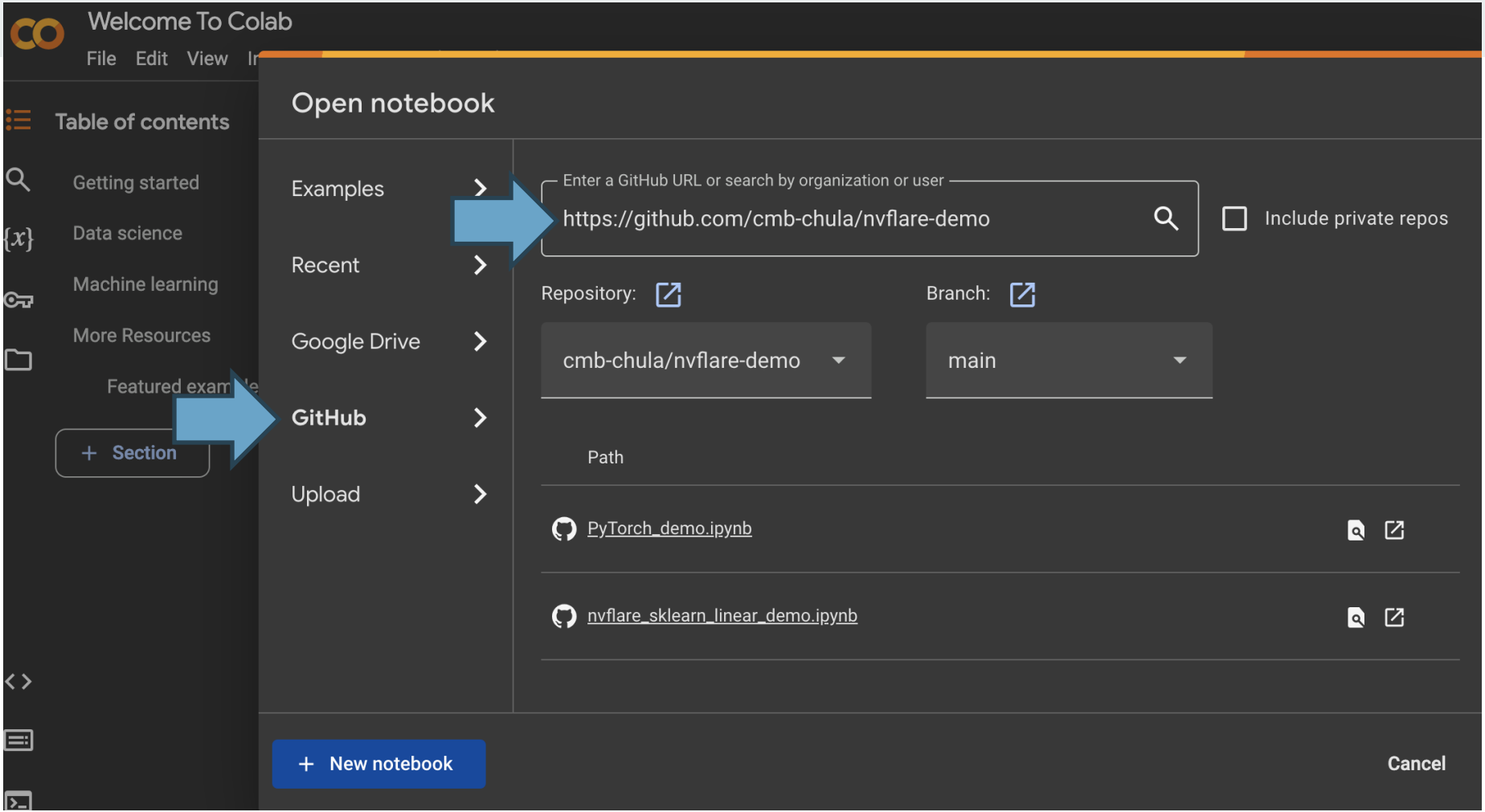
# Progress monitoring with TensorBoard



# Let's do some demo

- Google Colaboratory: Python and GPU services
- **Demo 1**: PyTorch for deep learning development
- **Demo 2**: NVFlare FL simulator
  - 4 clients
  - 10 rounds


https://colab.research.google.com/


Welcome To Colab
File Edit View
Table of contents
Getting started
Data science
Machine learning
More Resources
Featured exam
Section
Open notebook
Examples
Recent
Google Drive
GitHub
Upload
Enter a GitHub URL or search by organization or user
https://github.com/cmb-chula/nvflare-demo
Include private repos
Repository:
Branch:
cmb-chula/nvflare-demo
main
Path
PyTorch_demo.ipynb
nvflare_sklearn_linear_demo.ipynb
New notebook
Cancel