FLATR—Federated Learning Across Telecommunication Resources

Ribbon Intern Workshop – Spring 2024

Ved Nigam, Subre Moktar, Kenan Stredic, Orlando Malanco

May 2024

Agenda

- Intro –
  - University of Texas at Dallas Capstone Group (4 Credit Hours)
  - Federated Learning on Ribbon Resources
- Project Details –
  - Improved the previous year's federated model on Ribbon telecommunication data
  - Also, added explainable AI to evaluate the model
  - To identify features that contribute to call quality and inconsistency
- Next Steps –
  - Would like to improve model accuracy
  - More on this towards the end

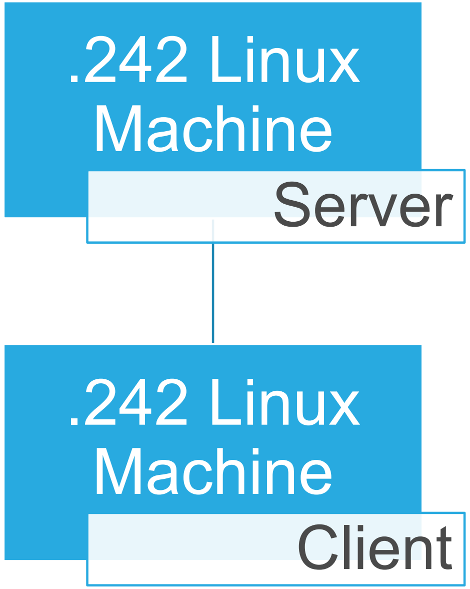
What is Federated Learning

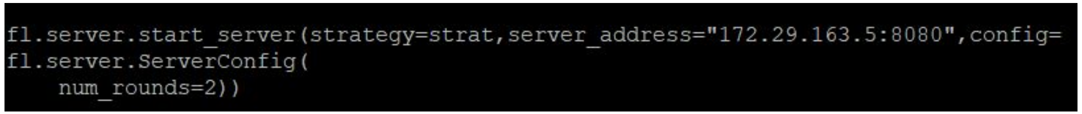
- Federated Learning is a type of machine learning where data is distributed across many entities and a singular model is beneficial for all parties
- Models are trained locally and aggregated in some way to build a centralized model
- Federated learning can be implemented Vertically or Horizontally

Why Use Federated Learning

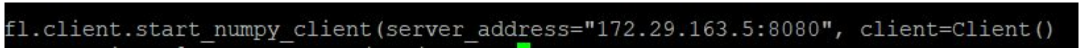
- Federated Learning:
  - Allows companies to share their data without losing company secrets
  - Allows for Data Privacy to be maintained
  - Allows for greater data security
  - Allows companies to benefit from the aggregated model created

Basic Architecture of Federated Learning



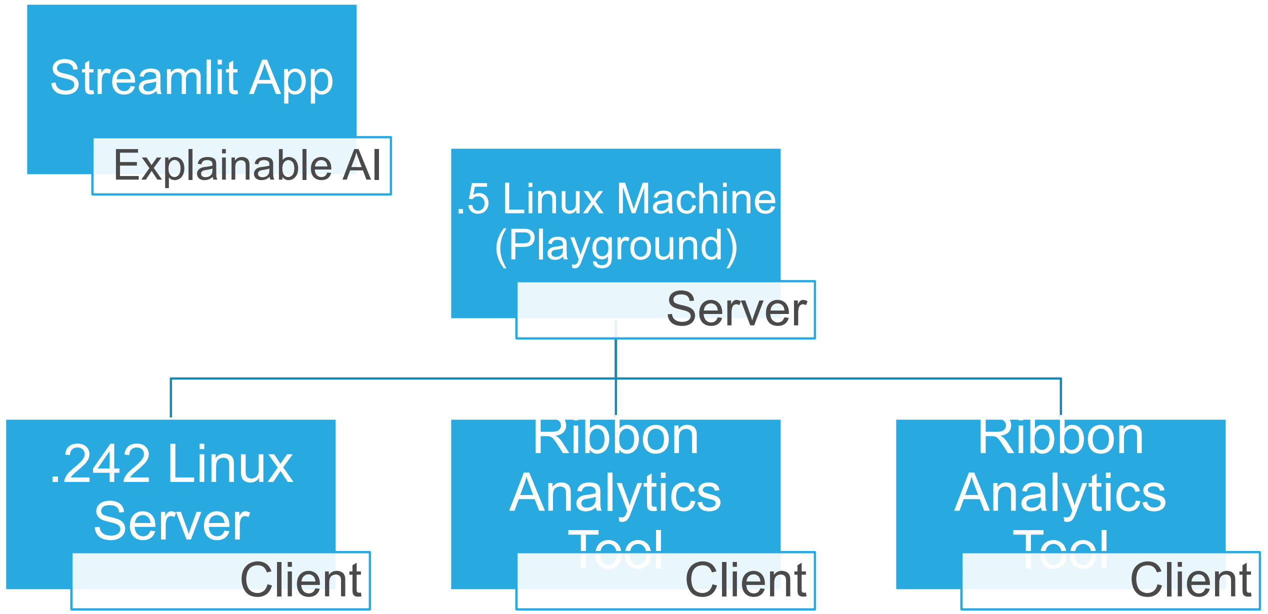


```
fl.server.start_server(strategy=strat,server_address="172.29.163.5:8080",config=
fl.server.ServerConfig(
    num_rounds=2))
```



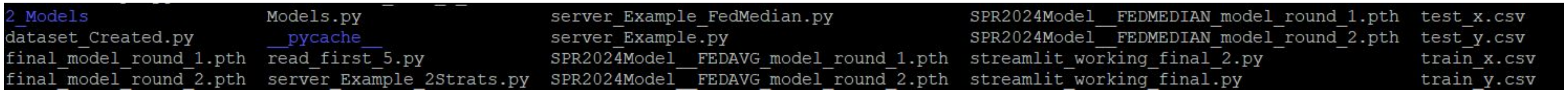
```
fl.client.start_numpy_client(server_address="172.29.163.5:8080", client=Client()
```

Details on the Architecture and Beginning Work



Saving the Model

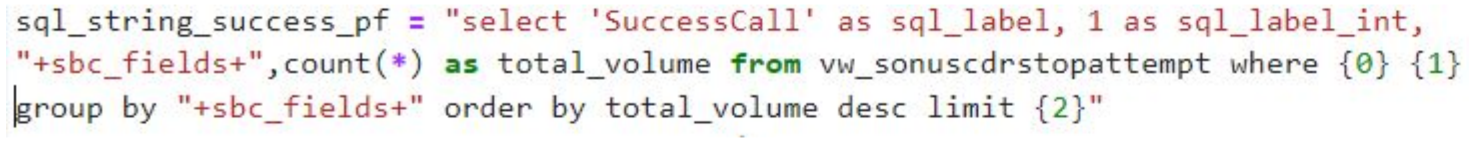
- As the clients are training the model, the weights of the model are being shared into the server
- The model will be saved on the server side, which is where we will begin evaluating the model



```
2_Models                Models.py                 server_Example_FedMedian.py         SPR2024Model__FEDMEDIAN_model_round_1.pth  test_x.csv
dataset_Created.py      __pycache__               server_Example.py                   SPR2024Model__FEDMEDIAN_model_round_2.pth  test_y.csv
final_model_round_1.pth read_first_5.py           SPR2024Model__FEDAVG_model_round_1.pth streamlit_working_final_2.py            train_x.csv
final_model_round_2.pth server_Example_2Strats.py SPR2024Model__FEDAVG_model_round_2.pth streamlit_working_final.py              train_y.csv
```

Ribbon Analytics Tool

- The Analytics tool will ideally be on each client side, connected to their database
- We would like to query their data and form datasets to federate the models across data from different companies



```
sql_string_success_pf = "select 'SuccessCall' as sql_label, 1 as sql_label_int,
"+sbc_fields+",count(*) as total_volume from vw_sonuscdrstopattempt where {0} {1}
group by "+sbc_fields+" order by total_volume desc limit {2}"
```

Training the Prior Semester's Model with Call Record Data

- Next, our aim was to integrate one of the flower clients directly within the Jupyter Notebook workspace of our analytics tool, this time utilizing call record data instead of MNIST/CIFAR-10 data.
- Initially, we gathered call record data encompassing both successful and failed phone calls lasting longer than 2 seconds through SQL queries.
- This dataset served as the foundation for training a multi-layer perceptron model that was developed by last year's Ribbon capstone group.

Running a Client on Analytics Tool

- Following the model's training with the new query data, we proceeded to implement federated learning, leveraging both the prior semester's query data and the newly trained model.
- The federated learning process was seamlessly executed, with one of the three flower clients integrated directly into the analytics tool's Jupyter Notebook environment. Meanwhile, the remaining two flower clients and the flower host server were running on the ribbon servers.
- Next, we wanted to use explainable ai to visualize features from this model.

Explainable AI App

- We created a Explainable AI (Streamlit) app to show off the model, and used Shap for its explainability AI
  - Explainability AI is trying to make a black box model (Highly complex model) more explainable. For example, a decision tree, is just one tree and you can easily see what causes each split. Then how would you explain a random forest of 200 trees. This is hard therefore we need explainability ai to make it easier to understand a random forest.
- We wanted to have better understanding of what the federated model was learning for more transparency of our model.

Now we will give you a demo of the Streamlit app with the MNIST Model

What We Learned

- Throughout this project, we acquired proficiency in various programming languages and tools essential for data science, including Python, SQL, and Linux/Bash, as well as development environments like Visual Studio Code, Jupyter Notebook, and Streamlit. Moreover, we gained hands-on experience in accessing data from and working on servers, utilizing tools such as PuTTY and WinSCP for server navigation.
- Our exploration into Federated Learning machine learning techniques involved extensive coding, researching research papers, and scouring the internet for valuable insights, which will undoubtedly benefit our future careers in data science. Additionally, we developed essential teamwork skills in a professional environment, collaborating effectively with our peers.
- Furthermore, we received invaluable guidance and tools for success from our mentors, enhancing our learning journey and bolstering our skill set for future endeavors.

Impact on Ribbon

- The research conducted in this project will play a pivotal role in integrating federated learning into the model training process for error code analysis in call records.
- Federated learning promises enhanced prediction accuracy while safeguarding the privacy of all users involved in phone calls.
- By continually updating the aggregated model with fresh data, Ribbon will expedite error code identification and prevention, facilitating quicker and easier resolution.

Future Works

- Expand Streamlit app functionalities:
  - Incorporate additional model evaluation metrics.
  - Explore federated learning strategies.
  - Integrate models from alternative datasets.
  - Augment data visualization options for specific call record error codes through plots.
- Implement LIME for explainable AI:
  - Assess its performance compared to SHAP.
- Aim to run all three clients on the analytics tool:
  - Server code execution on the playground server.

Where to find our work

- All project code, readmes, and pertinent files will be accessible via the Ribbon Communications Microsoft Teams platform within the Ribbon Labs/00_Federated Learning/FL_Projects folder, under FLATR_SPR24.

Thank You

ribbon®