**Flower evaluation**

* **General information**

Flower started as a federated learning package; later it developed into a startup. Flower is still growing and there are many new updates to the package. There can be found many frameworks, ready partitioned datasets and example projects on the official Flower website. The git repository is regularly updated and there are multiple tutorials available.

* **How does it make sure privacy is preserved?**

Flower itself does not enforce security by default; however, it provides all necessary elements of the infrastructure of federated learning. Flower offers a client class, and all the functions defined for it remains private, only client has access to them. Moreover, data never leaves the client, so all the computations performed by those functions, return only the output of the computation (if programmed to do so). FL is a technique ensuring data privacy, Flower ensures secure communication between server and the client. On top of that, you can apply additional privacy features such as other PETs.

Flower introduces additional layer of protection of client information. It is often a worry that the server might be able to inspect the client while collecting parameters of a trained model; or in case of dropouts or configuration difficulties of the client, the server will be able to trace it back and infer some information. Secure aggregation protocols and similar are a solution to this issue. Flowe provides multiple functions that allow for its implementation.

* **Ease of use**

Flower is very straightforward and easy to use. The repository and website provide many tutorials and does not assume familiarity with other Python packages. There is even a tutorial explaining how federated learning works, so anyone who is new to this technique can quickly get familiar with it. The installation does not differ from installing any other packages. Flower has active community where most of your questions can be answered. Despite fast growth and many updates, Flower stays compatible, and dependencies usually are not an issue.

* **What works well**
  + Compatible with Hugging Face
  + Compatible with most of the Python packages
  + Numerous built-in functions
  + Various tools: apps, mods, templates, visualization techniques, prepared databases etc.
  + Many tutorials, guides and videos designed by creators of Flower
  + Possible to apply additional layers of privacy with minimal effort
  + Ready to use (templates are highly functional)
  + Does not assume much prior knowledge
  + Possible to highly personalise to individual needs
  + Deployment ready
  + Allows for broad experimentation
* **What could be improved** 
  + Continuously growing: some features are not available or throw errors with specific versions of other packages.
* **Additional functionality**

*Evaluation*

Flower provides tools for model evaluation. It can be done on a level of a client or server (decentralised vs centralised approach). Depending on the situation and the goal, you can choose which method you will apply in your case. Evaluation on the server is possible only if there is some public data available or mock/representative dataset. Otherwise, the evaluation will not be performed (by default, it does not run; once you provide a centralised dataset this function will run). It can speed-up the evaluation process and make it more consistent, however, if the available data is not representative it might return misleading results. In some cases, it could potentially reveal some hints about real data imposing some privacy risks. In case of the decentralised evaluation, the results of the evaluation performed by each client are aggregated on the central server. It can give more realistic result but also introduce a lot of noise or inconsistency due to differences between clients. In real-world applications usually the decentralised approach is used, while for debugging and experimenting, it is alright to use centralised method for simplicity.

*Build-in mods for flexibility*

Flower offers a wide range of modes. With their help you can write reusable code that separates logic from the rest of the function. This way, instead of writing code that has to be adapted per case, you can specify general logic and customise all the methods. They follow a specific order of execution, where the more general ones are performed first, and only at the end client-specific functions are run. It also specifies the communication between client and server (how information is sent between the two).

*Baselines*

There are many baselines provided on the official Flower website. Those are implementation of different publications. Once you download the same dataset and set up the same environment you should yield the same results. Baselines are provided to speed up your work: you can build on top of it, improve or modify it.

Datasets

Flower supports creation and partition of datasets suitable for federated learning during training/analytics/evaluation. It also helps with data visualisation and supports usage of datasets available online (all datasets available on hugging face can be used and can be processed by Flower). There is also a preview tool available online to visualize your dataset and its partition.

*Projects*

There are some example projects including machine learning models, different use cases and platforms that can be reused or applied as a template to your own project. It is also a great learning material. They provide explanations along the code. Most of the time, an extensive knowledge is not assumed so it is easy to understand for anyone who wishes to start working with a specific template/a package used in that template. They can be directly cloned from the Flower repository or accessed via the official website.

*Apps*

On top of all of the described options provided in Flower, you can also create your own apps. It offers an easy was to switch from experimental setup to deployment. It creates a whole federated learning setup that is highly reusable. You define your clients and server and the way they communicate. It is in a form of a separate file describing components, parameters and how it is supposed to be configurated (resources, strategy, etc). Once you have the app, you can create the other files describing the rest of the functionality and the task (create a ML task, test, train and evaluate). This structure allows for easier deployment for a few reasons. This way logic is separated from execution. The app file also contains all the specifications, and all files are already partitioned (client, server, strategy are separate) allowing for experimentation and swapping different components. It also resembles potential partition for containers in later deployment.

*Messages*

It is a basic building block for communication between server and client in Flower. It includes information about exchanged data and instructions. It has a specific type that describes what kind of information will be retrieved (arbitrary information, model parameters, custom query, etc). To create a message on the client side, you have to use a special decorator, while for server you only specify the type of it.

*Flower Intelligence*

It is a cross-platform inference library available for TypeScript/JavaScript, Kotlin, and Swift only. It allows for interacting with LLMs locally or remotely and creating AI apps. It is not pure federated learning, but a tool for improving LLMs on private user data. It is specifically dedicated for apps. “*When extra power is needed, Flower Confidential Remote Compute steps in as a seamless private extension of the device, without compromising privacy, security or performance. This hybrid approach delivers the best of both worlds: local-first AI that remains powerful, private and compatible with all devices. […] The Flower Confidential Remote Compute service acts as a seamless private extension of the device that uses end-to-end encryption and other techniques to protect sensitive user data.”*

* **How to create your own Flower app?**

1. Once you are in your Flower environment run ‘flwr new <name\_of\_the\_app>’ – this command will create a new app; you can select what kind of template you want to use. Once you have your app, you can either run the selected template or modify it/create your own code. To do that start with the ‘task.py’ file.
2. Select a dataset you want to work on. All Hugging Face datasets are compatible with Flower. If you wish to use another dataset, you have to be mindful of labels and structure of the data.
3. Crete your function for loading data. Flower provides a function to quickly partition data. Next, you can split your partitions into train/test split.
4. Preprocess your data if necessary. Depending on the type of data it might include tokenization, converting to tensors or other types.
5. Specify a model; add train and evaluate function. Additionally, you can adopt set and get weights but usually the provided default one is suitable.
6. If necessary, adopt functionality of the client to the used model architecture. You can also add more functionality or put restrictions on it. Creating messages to communicate with server is also possible.
7. Adopt server file if necessary. You can change the way results are aggregated and processed. Adding messages to communicate with clients is also possible. Adding more advanced functionality is also possible as long as it is compatible with the rest of the code. If you wish to add extra privacy features, you can call them here.
8. Create any additional helper files. If you want to use another strategy or define new functionality that would be called by client or server, you have to add it separately.
9. Run your app from terminal via ‘flwr run .’

* **Requirements**

Flower requires at least Python 3.9 or higher. I run it on Python 3.10.

* **Sources**

Flower on Azure tutorial: <https://flower.ai/docs/framework/how-to-run-flower-on-azure.html>

Flower website: <https://flower.ai>

Flower Intelligence tool: <https://flower.ai/docs/intelligence/>

Flower repository: <https://github.com/adap/flower/tree/main>

Introduction to FL with Flower demonstrating its functionality: <https://github.com/adap/flower/tree/main/examples/flower-in-30-minutes>