Shano Fog Presents:

How MlFlow flows

26-09-2022 / Q3

|  |
| --- |
| Facts about   * It has built in integration with Databricks and Docker * It has built in support for a lot of different machine learning libraries, e.g Scikit-learn, PyTorch, XGBoost, Keras, Tensorflow and so on |
|  |
|  |

# What is MLFlOw?

MLFlow is a platform that allows you to manage end-to-end machine learning lifecycles, and it has 4 main functions

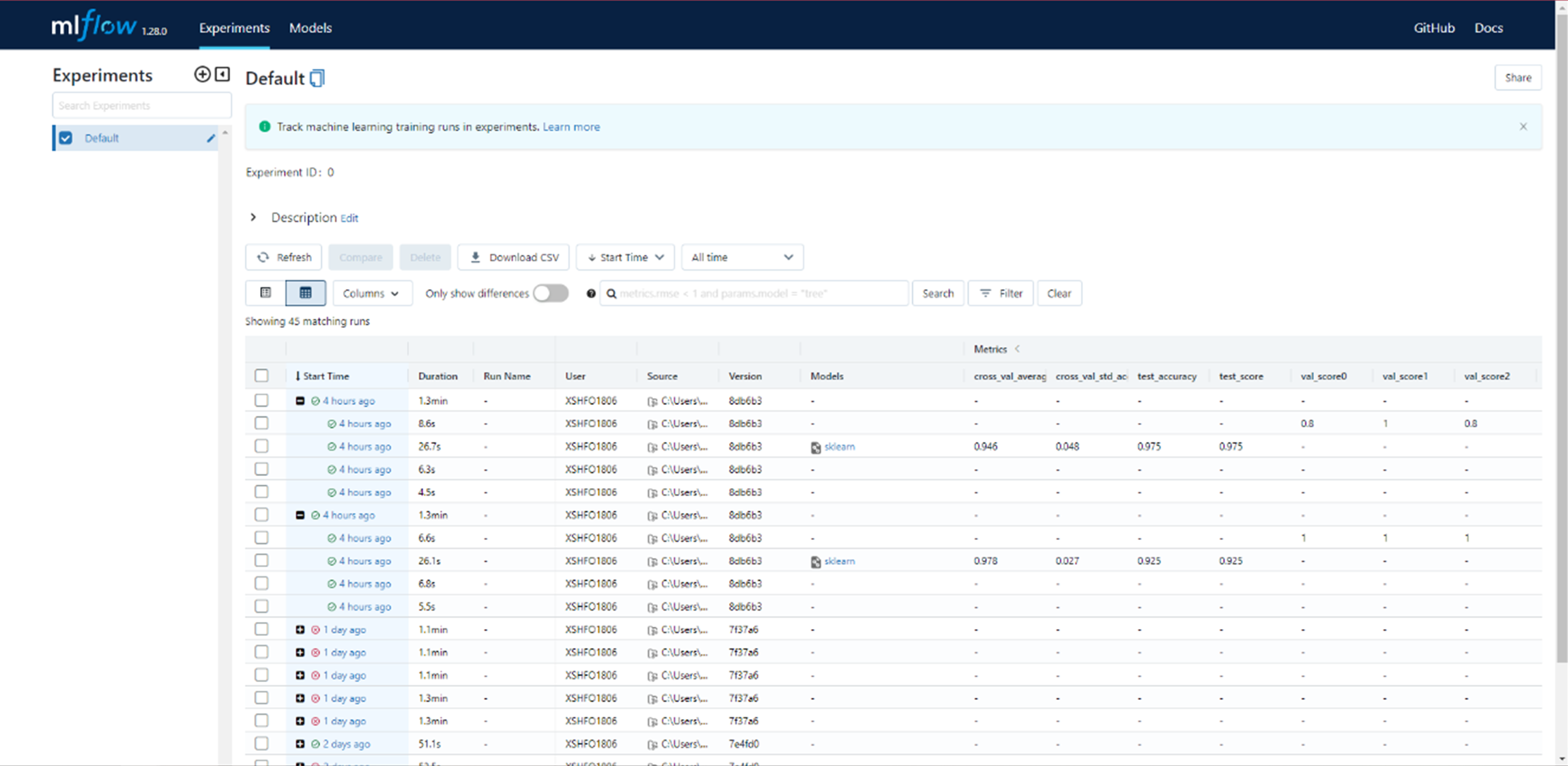
* Tracking
* Project packaging
* Managing and deployment
* Model registration and management

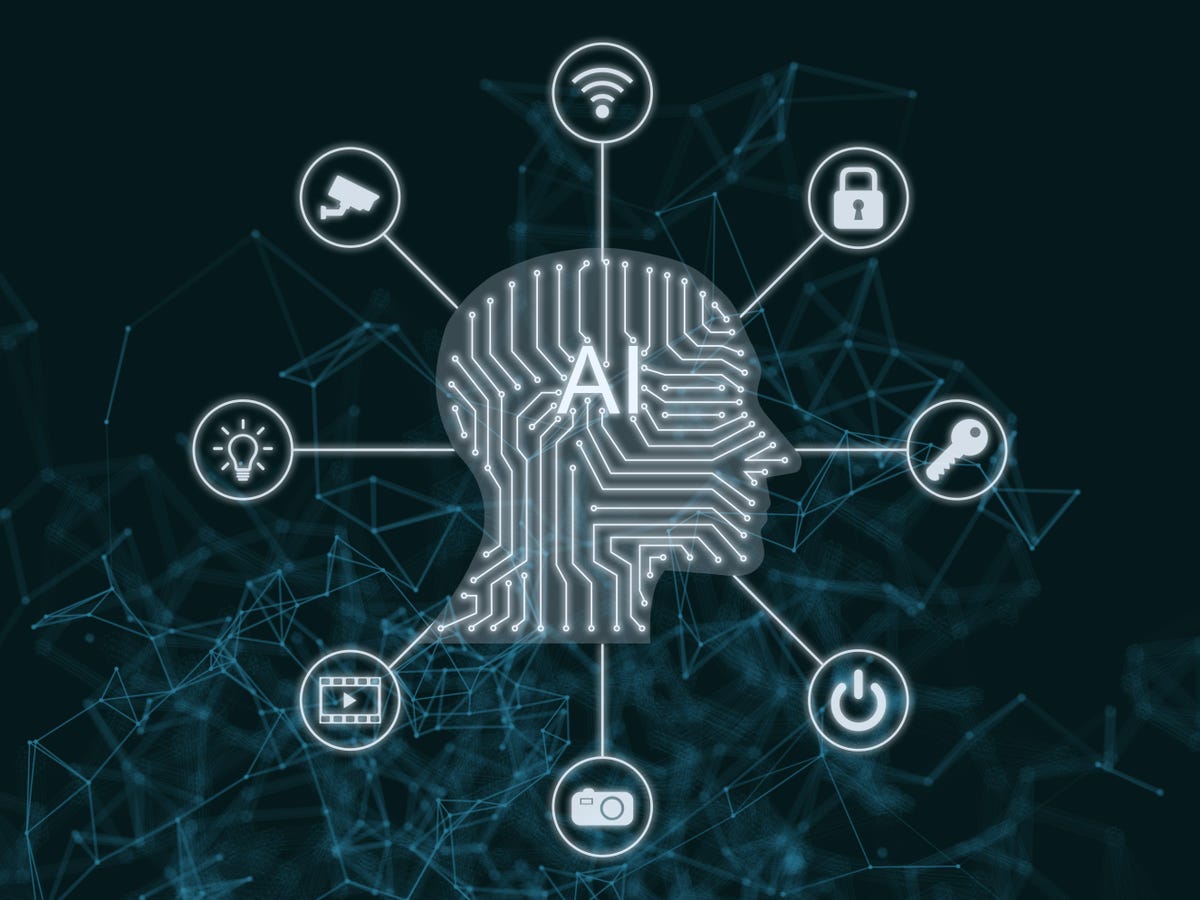
Terms and what they mean:

* MLFlow Experiment: An experiment is a way to organize different trials and or mlruns. It allows you to categorize runs and their statistics/metrics and artifacts under one category. It allows you to have different testing and production experiments, so you can have them seperated.
* MLFlow Run (mlrun): A Mlrun is a sort of trail / single execution of the machine learning pipeline *(Don’t know about pipelines? Click* [*here*](https://c3.ai/glossary/machine-learning/machine-learning-pipeline/#:~:text=A%20machine%20learning%20pipeline%20is,model%20parameters%2C%20and%20prediction%20outputs.)*)*
* Artifacts: An artifact is a relative uri that mlflow can store which points directly to the object/file. This means that in the MLFlow UI you can actually see the file you asked it to save/log to that specific mlrun

## MLFlow as a platform

MLFlow provides a lot of different tools for better managing the lifecycle of a machine learning pipelines *(Don’t know about pipelines? Click* [*here*](https://c3.ai/glossary/machine-learning/machine-learning-pipeline/#:~:text=A%20machine%20learning%20pipeline%20is,model%20parameters%2C%20and%20prediction%20outputs.)*)*. Tools such as the above mentioned 4 main functions, in those 4 functions there are smaller tools that can help you. What is also extremely useful is that MLFlow is library-agnostic, this means that it can be used with any machine learning library, along with the fact that you can use it with any programming language since every function can be called from a REST API

One notable feature is the user interface which is part of the tracking function. It gives you a visible overview of the different mlruns

Another really useful feature that is included in the same tracking function, is the artifact logging and metric logging.

“An Awesome tool for making maintanence on Machine learning models easy”

# Core concepts

The core concepts of MLFlow is making a reusable and reproducible machine learning package. You can make it highly specialized for one purpose, or you can make it extremely generic so it serves as a form of template for future projects that use Machine Learning.

# Using MLFlow in practice

How to use mlflow in practice

MLFlow has a nice quickstart guide that will give you an introduction into how you can run the pipeline, it will help you understand just a little more about MLFlow.

The quickstart guide (You can view it [here](https://www.mlflow.org/docs/latest/quickstart.html)) is rather lackluster at times though, and it shouldn’t be used for anything more than a quick introduction into MLFlow, it doesn’t tell you anything about project structure or other really useful things to know in MLFlow. It only gives you a snippet of introduction into the different usages of MLFlow

Therefore, I will give you a quick overview on how to execute my code. How the project structure is and how you can view and fool around with the different aspects of the UI that the tracking function has

**SETUP:**

To get started you will need 2 things. 1. Anaconda installed (used to install python and libraries), you can find the guide on how to install: [here](https://www.datacamp.com/tutorial/installing-anaconda-windows) 2. MLFlow installed.

The easiest way to install MLFlow is searching for: Anaconda Prompt (Anaconda3) , then clicking enter and making sure the Anaconda Prompt is open, after that you can simply use this pip command to install MLFlow into your Anaconda Environment: pip install mlflow

In the case of my project you can follow the README.md that is included in my github repository: <https://github.com/Zenress/Iris-MLFlow/tree/development>

**RUNNING THE PIPELINE:**

There are a lot of different parameters used in the command used to run the pipeline, I will explain some of the more important ones:

**Project URI (local path or the github repository link):** The project uri is used to identify where the project you are trying to run is located, there are some specific things you can do with the project uri as well. You can run Git over SSH by using: user@host:path. You can also specify a subdirectory in the project uri by using # at the end of the path and then writing the subdirectory you want to run, this isn’t as commonly used. It can be pretty useful if you have the project inside a big root folder with other projects though (this practice is not recommended)

**Project version:** This is for Git-based projects. You can use the commit hash or branch name of the Git repository to specify which version you wanna run

**Entry Point:** You can choose to specify a specific step to start the pipeline on or just leave it empty. *(The default is main. So if you wanted to write that just leave it out).* This is especially useful if you have a step that faills and want to figure out how many of the other steps would fail as well.

**Parameters:** They are something you use for different purposes and it’s a longer paragraf just to explain what they are and how to specify them. It’s useful for passing settings you might want to enable or disable through, e.g enabling error detection *(if it’s setup)*

**Deployment Mode:** This is used to specify where the run is executed, you run it from a local machine yes, but in many cases you would just use the local machine to queue a run on Databricks or a similar cluster or computing infrastructure.

**Environment:** This is used to force MLFlow to use another environment than the one that was specified in the mlprojects file. There are uses for this but I have not yet needed to use this for the overall run command (I use the code version of environment specification when starting a step)

Now, onto running the project itself. The prefix command for running a project is: mlflow run

Note: When talking about a command it is generally understood that you are using a console/terminal that already is using a environment with mlflow

So an example command could look like this:

mlflow run iris-pipeline

Here is an empty command template:

mlflow run [project uri] [entry point] [paremeters] [deployment mode] [environment]

As you can see, a lot of the different parameters (in the command itself) can be omitted since they might not always be needed. This is where it can differ a whole lot from other projects as well

**RUNNING THE UI:**

The UI is extremely easy to run since it’s a single command to start the UI. Afterwards it’s more about understanding how to navigate the UI and I’ll give a quick overview after giving the command.

The command for running the UI is:

mlflow ui

Here is a small explanation of the UI

