Experiment Tracking with MLFlow

MLflow

MLflow is an open source platform for managing the end-to-end machine learning lifecycle. It provides a set of tools and APIs for tracking experiments, packaging code into reproducible runs, and sharing and deploying models.

MLflow allows data scientists to easily track and compare experiments, so they can see how different model architectures, hyperparameters, and datasets perform. It also provides a model registry for storing and versioning trained models, and a deployment API for serving models in real-time.

MLflow is designed to be language-agnostic, so it can be used with a wide range of machine learning frameworks and libraries, including TensorFlow, PyTorch, Scikit-learn, and more. It is a popular tool in industry and academia for managing the machine learning workflow, making it easier to develop, test, and deploy machine learning models.

Introduction to Experiment Tracking

Terminologies:

1. Experiment
2. Run
3. Metadata (i.e. Tags, Parameters, Metrics)
4. Artifacts (i.e. Output files associated with experiment runs)

What do you want to track for each Experiment Run?

1. Training and Validation Data Used
2. Hyperparameters
3. Metrics
4. Models

Why Track?

Organization Optimization Reproducibility

Tool - MLFlow
MLFlow helps you to organize your experiments into runs.

MLFlow keeps track of:

Tags
Parameters
Metrics
Models
Artifact
Source code, Start and End Time, Authors etc..

Run below mentioned commands to install mlflow on your system:

```
pip install mlflow
mlflow ui --backend-store-uri sqlite:///mlflow.db
```

Introduction to MLFlow

Step 1 - Import MLFlow

```
import mlflow
```

Step 2 - Set the tracker and experiment

```
mlflow.set_tracking_uri(DATABASE_URI)
mlflow.set_experiment("EXPERIMENT_NAME")
```

Step 3 - Start a experiment run

```
with mlflow.start_run():
```

Step 4 - Logging the metadata

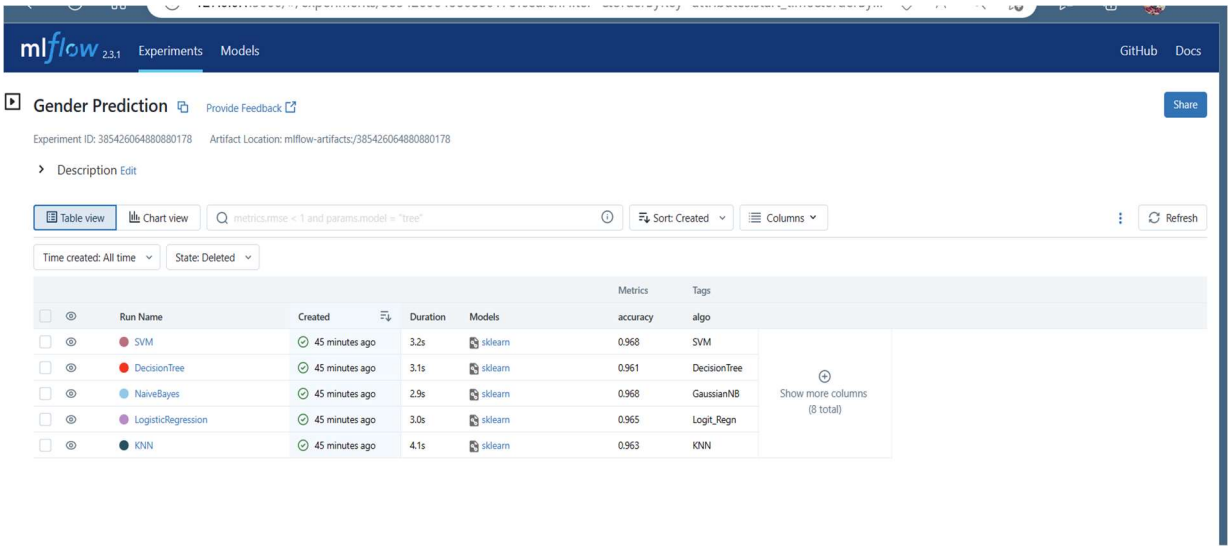
```
mlflow.set_tag(KEY, VALUE)
mlflow.log_param(KEY, VALUE) mlflow.log_metric(KEY, VALUE)
```

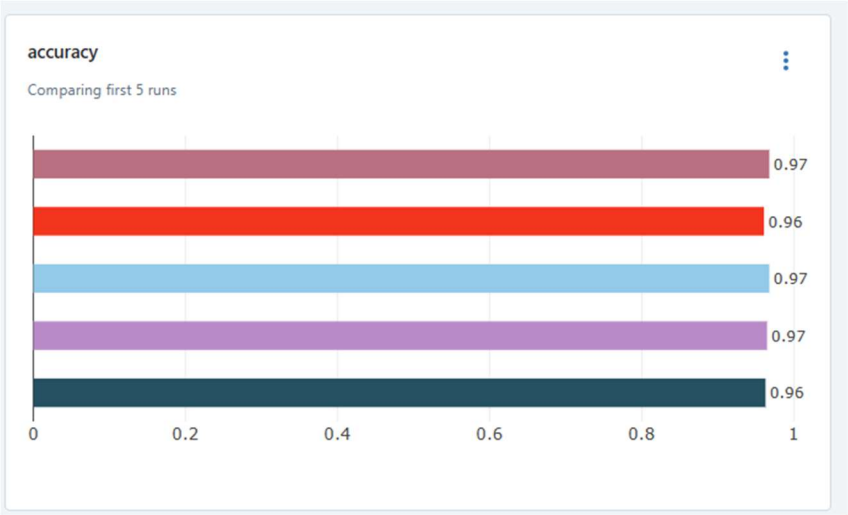
Step 5 - Logging the model and other files (2 ways)

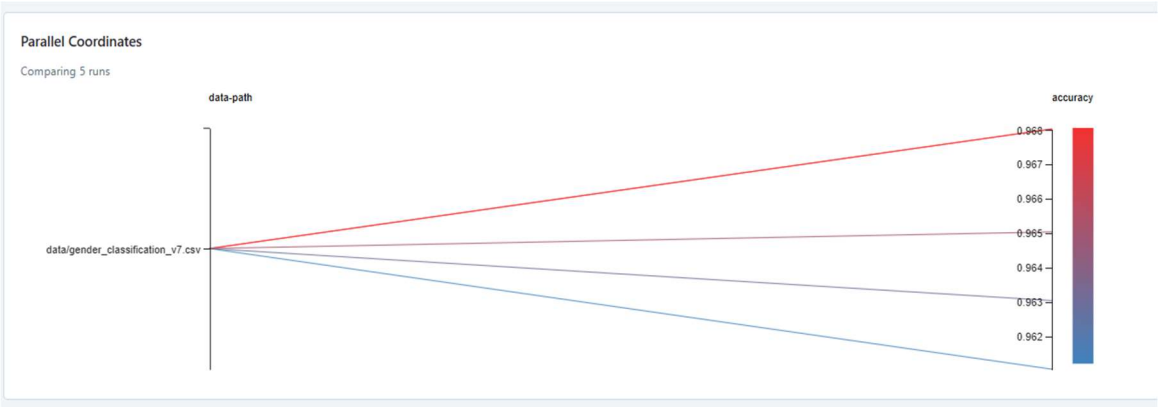
Way 1 - `mlflow.<FRAMEWORK>.log_model(MODEL_OBJECT, artifact_path="PATH")`
Way 2 - `mlflow.log_artifact(LOCAL_PATH, artifact_path="PATH")`

MLflow Experiments Page

MLflow Experiments is a tool for tracking and managing machine learning experiments. It provides a way to organize, compare, and reproduce machine learning runs, as well as visualize and share the results. With MLflow Experiments, you can track and log parameters, metrics, and artifacts for each run, and easily compare the results across runs. You can also search and filter runs based on various criteria, such as tags, dates, and performance metrics. Additionally, MLflow Experiments supports integration with other MLflow components, such as MLflow Tracking and MLflow Models, to provide a complete platform for the machine learning lifecycle.



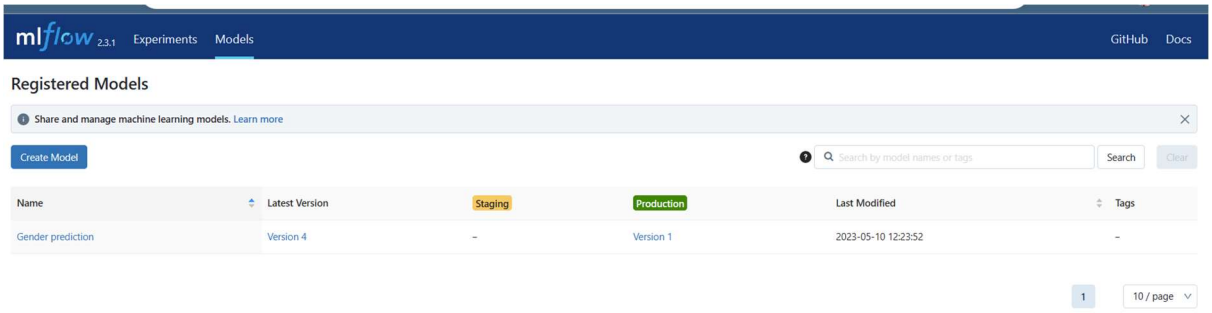




MLflow Models page

MLflow Models is a component of the MLflow platform for managing the end-to-end machine learning lifecycle. It allows you to package machine learning models in a standard format and deploy them to various environments such as batch, real-time serving, or inference in the cloud. You can track model training runs and compare performance metrics using MLflow tracking, and version model artifacts and metadata

in a centralized model registry. MLflow Models supports a wide range of machine learning frameworks and deployment platforms, and provides an API and command-line interface for interacting with models programmatically.



MLflow Models versions before stages

Before the introduction of model stages in MLflow 1.8.0, MLflow Models Registry did not exist as a separate component. Instead, model versioning and management were handled within the MLflow Tracking component.

Users could use the MLflow Tracking API to log different versions of a model, and track its metadata and artifacts over time. However, there was no dedicated UI or API to manage model versions as a separate entity, and users had to manually track which versions of a model were ready for deployment, which ones were in development, and which ones were no longer needed.

Overall, while users could use the MLflow Tracking component to manage different versions of their models, the lack of a dedicated Model Registry made it more difficult to manage and share models across teams and organizations. The introduction of the Model Registry in later versions of MLflow addressed these limitations and provided a more streamlined way to manage and share models.