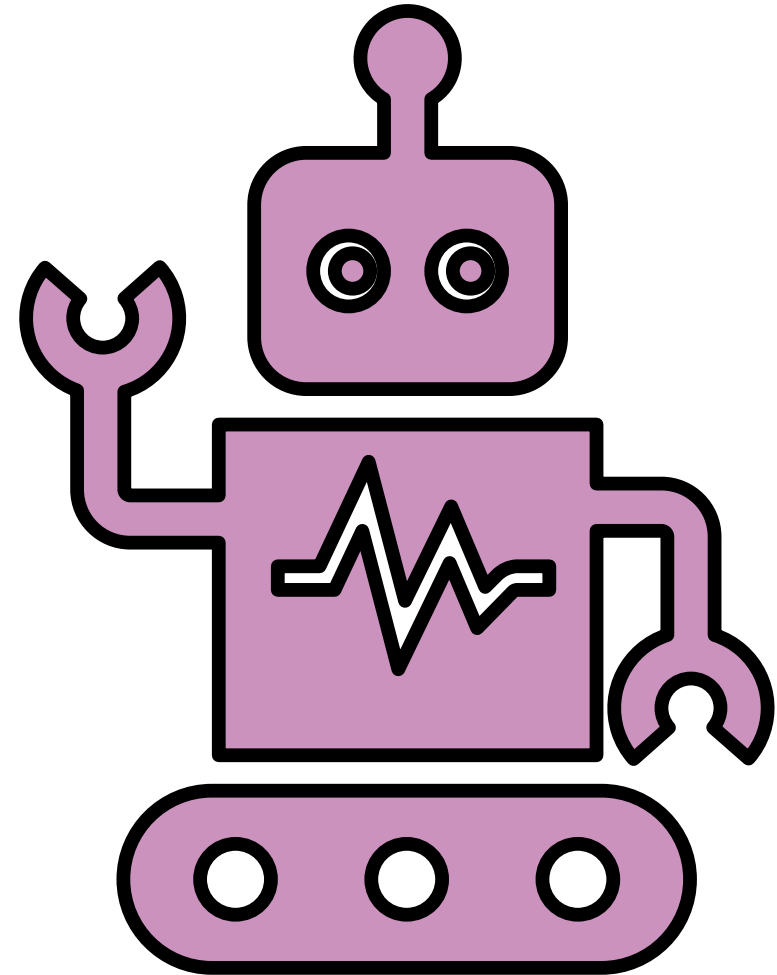


Introduction to MLflow

A Comprehensive Overview of MLflow for Machine
Learning

Introduction to Mlflow:

- MLflow is a comprehensive **open-source platform** developed to address the challenges associated with the **end-to-end machine learning lifecycle**. It facilitates seamless management from the **initial experimentation phase** to the **deployment of machine learning models**.





Key Objectives:

- MLflow aims to **simplify** and enhance the machine learning development process by providing tools and components that streamline **experiment tracking, project packaging, and model deployment**. It acts as a unified platform for data scientists and machine learning engineers.

Integrations with:



PyTorch



HuggingFace



OpenAI



LangChain



Spark



Keras



TensorFlow



Prophet



scikit-learn



XGBoost



LightGBM



CatBoost



XGBoost



LightGBM



CatBoost

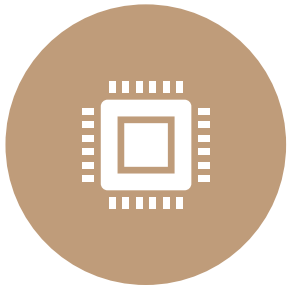
Components Overview:



Tracking: Enables logging and querying of experiments, capturing essential information such as **parameters**, **metrics**, and **artifacts**.



Projects: A system for packaging code into a reproducible format, ensuring that experiments can be **easily reproduced**.



Models: Provides tools for **packaging** and **deploying** machine learning models with support for various frameworks and flavors.



Registry: A centralized repository for **managing** and **versioning** machine learning models.

Tracking:

- The tracking component in MLflow is designed to capture and log essential information during the experimentation phase of machine learning development. It ensures that every aspect of an experiment is recorded systematically for later analysis and reproducibility.

A simple example of logging:

```
import mlflow

with mlflow.start_run():
    mlflow.log_param("learning_rate", 0.01)
    mlflow.log_param("hidden_layers", 3)
    # Other code for the experiment
```

Projects:

- The Projects component in MLflow provides a standardized structure for packaging code into a format that ensures reproducibility. It encapsulates dependencies, code, and configurations, making it easy to share and reproduce experiments across different environments.

A simple example of an 'mlproject' File :

```
name: my_project
conda_env: conda.yaml

entry_points:
  main:
    parameters:
      input_path: path
    command: "python my_script.py {input_path}"
```

Example: Running a project with the CLI

mlflow run my_project -P input_path=data.csv

Models:

- The Models component in MLflow is designed to facilitate the packaging and deployment of machine learning models. It supports various flavors, allowing flexibility in packaging models for different frameworks and libraries.

MLflow provides APIs for packaging models. For a scikit-learn model

```
import mlflow.sklearn
from sklearn.linear_model import LogisticRegression

# Train a scikit-learn model
model = LogisticRegression()
model.fit(X_train, y_train)

# Log the model with MLflow
mlflow.sklearn.log_model(model, "my_model")
```


Tracking:

- The Registry component in MLflow serves as a centralized repository for managing and versioning machine learning models. It plays a crucial role in organizing, tracking changes, and promoting collaboration in the model development lifecycle.

A simple example of logging:

```
import mlflow.pyfunc

# Assuming 'my_model' is a registered model
mlflow.pyfunc.register(model_path="runs:/<run-id>/my_model",
name="my_registered_model")
```

MLFLOW UI

MLflow 2.7.1 Experiments Models GitHub Docs

Experiments

Search Experiments

- ☒ Default
- ☐ live_mnist_exp_20231007

Default

Experiment ID: 0

Artifact Location: file:///c:/Users/at0m/Desktop/mlflow-main/Main/mlruns/0

Description Edit

Search: metrics.rmse < 1 and params.model = "tree"

Time created: Last year State: Active

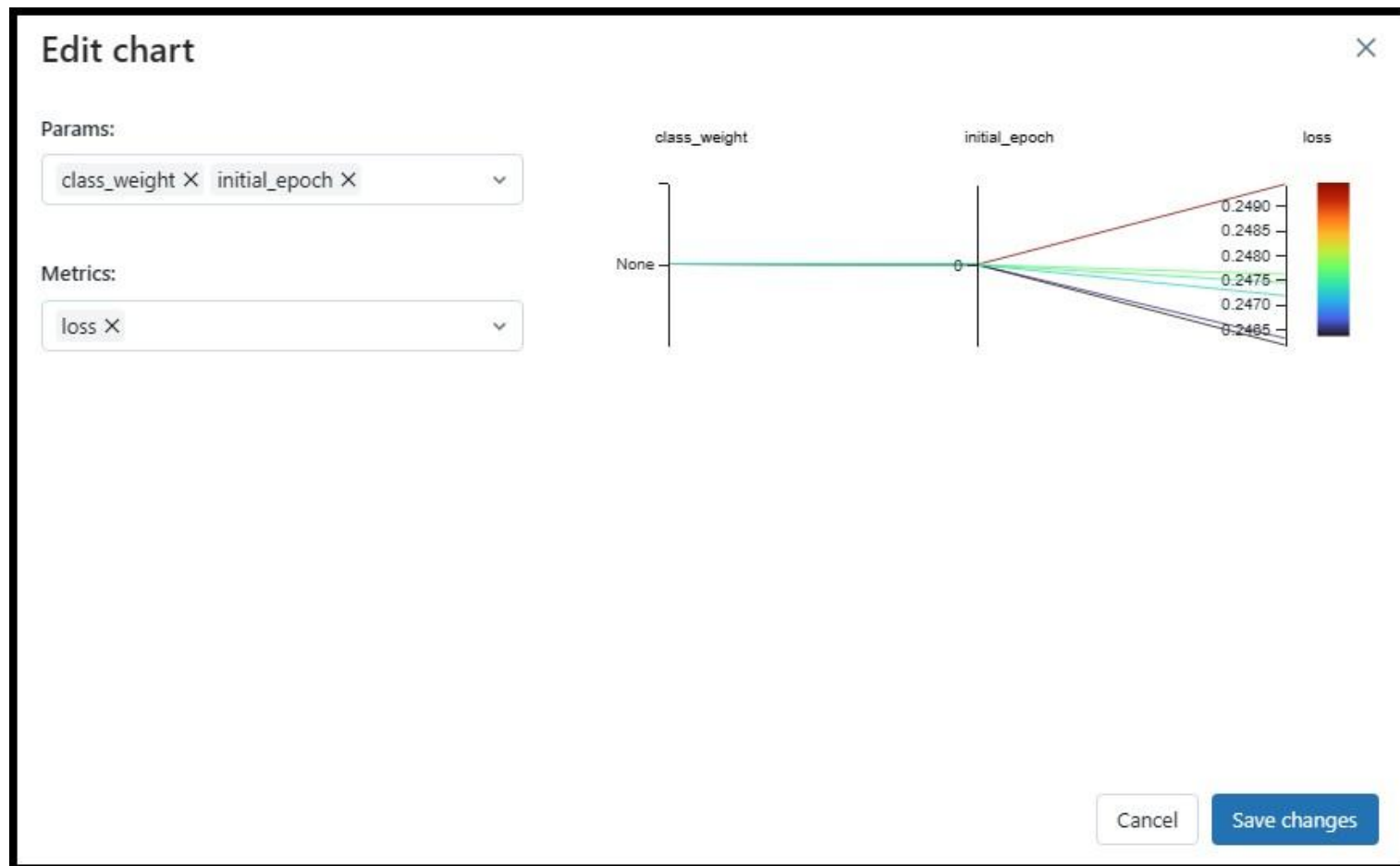
Sort: Run Name Columns Expand rows New run

Table Chart Evaluation Experimental

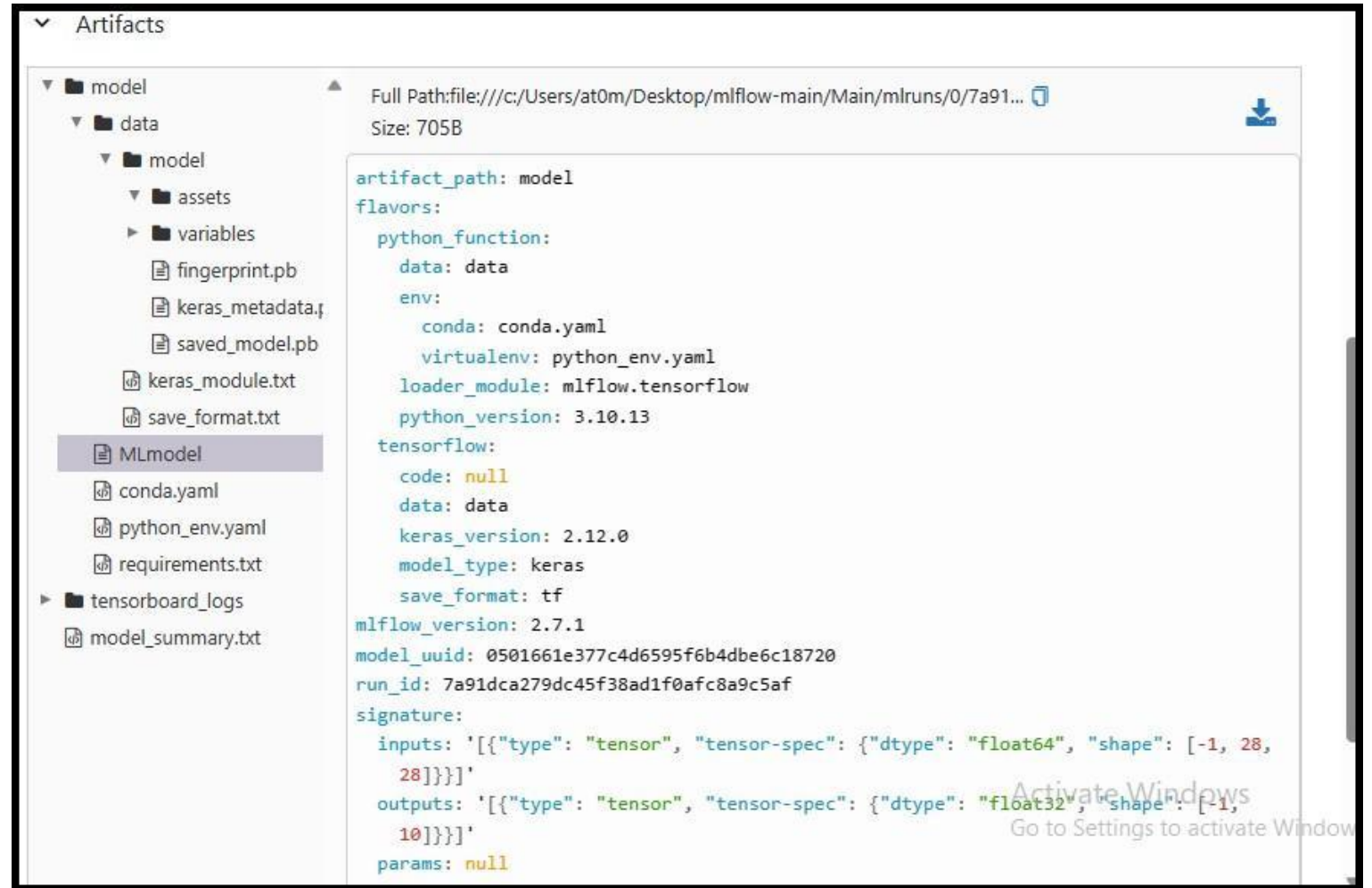
		Run Name	Created	Data
<input type="checkbox"/>	<input type="radio"/>	agreeable-shrimp-118	18 minutes ago	
<input type="checkbox"/>	<input type="radio"/>	carefree-fawn-812	1 hour ago	
<input type="checkbox"/>	<input type="radio"/>	chill-fawn-160	10 minutes ago	
<input type="checkbox"/>	<input type="radio"/>	orderly-toad-85	20 minutes ago	

6 matching runs

Visualize Data Easily



Organized Assets Easily



The screenshot displays the MLflow Artifacts interface. On the left, a file tree shows the artifact's structure: a 'model' folder containing a 'data' folder, which in turn contains an 'MLmodel' file and other files like 'fingerprint.pb', 'keras_metadata.pb', 'saved_model.pb', 'keras_module.txt', 'save_format.txt', 'conda.yaml', 'python_env.yaml', 'requirements.txt', and 'tensorboard_logs'. The 'MLmodel' file is selected. The right pane shows the metadata for this artifact, including its full path, size (705B), and various configuration parameters.

Artifacts

model

data

model

assets

variables

fingerprint.pb

keras_metadata.pb

saved_model.pb

keras_module.txt

save_format.txt

MLmodel

conda.yaml

python_env.yaml

requirements.txt

tensorboard_logs

model_summary.txt

Full Path: file:///c:/Users/at0m/Desktop/mlflow-main/Main/mlruns/0/7a91...
Size: 705B

```
artifact_path: model
flavors:
  python_function:
    data: data
    env:
      conda: conda.yaml
      virtualenv: python_env.yaml
    loader_module: mlflow.tensorflow
    python_version: 3.10.13
  tensorflow:
    code: null
    data: data
    keras_version: 2.12.0
    model_type: keras
    save_format: tf
mlflow_version: 2.7.1
model_uuid: 0501661e377c4d6595f6b4dbe6c18720
run_id: 7a91dca279dc45f38ad1f0afc8a9c5af
signature:
  inputs: '[{"type": "tensor", "tensor-spec": {"dtype": "float64", "shape": [-1, 28, 28]}}]'
  outputs: '[{"type": "tensor", "tensor-spec": {"dtype": "float32", "shape": [-1, 10]}}]'
  params: null
```

Activate Windows
Go to Settings to activate Windows

No Need to Manually LOG Everything

Parameters (29)	
Name	Value
batch_size	32
class_weight	None
epochs	10
initial_epoch	0
max_queue_size	10
opt_amsgrad	False
opt_beta_1	0.9
opt_beta_2	0.999
opt_clipnorm	None
opt_clipvalue	None
opt_ema_momentum	0.99

The screenshot shows the MLflow web interface in a browser. The address bar indicates the URL is 127.0.0.1:5000/#/experiments/0/runs/fe9c0f13476341c68a56... The page title is 'mlflow 2.7.1' with tabs for 'Experiments' and 'Models'. The main content area shows details for a run named 'agreeable-shrimp-118'. The 'Run ID' is 'fe9c0f13476341c68a56f7c09f56af5' and the 'Date' is '2023-10-07 19:10:49'. The 'Source' is 'c:\Users\at0m\conda\envs\kailash\lib\site-packages\ipykernel_launcher.py' and the 'User' is 'at0m'. The 'Duration' is '1.2min' and the 'Status' is 'FINISHED', which is highlighted with a red box. The 'Lifecycle Stage' is 'active'. Below this, there are expandable sections for 'Description', 'Datasets (2)', and 'Parameters (29)'.

MLflow 2.7.1 Experiments Models

Default >

agreeable-shrimp-118

Run ID: fe9c0f13476341c68a56f7c09f56af5 Date: 2023-10-07 19:10:49

Source: c:\Users\at0m\conda\envs\kailash\lib\site-packages\ipykernel_launcher.py User: at0m

Duration: 1.2min Status: FINISHED

Lifecycle Stage: active

> Description Edit

> Datasets (2)

> Parameters (29)

COMPLETE TUTORIAL

- https://drive.google.com/file/d/1lk9a_8AplqdyX53ZMfryD68-B5DRSSDs/view?usp=drive_link



Advantages of Mlflow:

- **Ease of Experiment Tracking:**
 - Captures parameters, metrics, artifacts, and source code.
 - Enables easy comparison of multiple experiments.
- **Reproducibility:**
 - Records the environment, dependencies, and versioning to ensure reproducibility.
- **Model Packaging and Deployment:**
 - Simplifies packaging and sharing of models.
 - Supports deployment to a variety of platforms (cloud, on-premises, edge devices).

Best Practices and Tips

Version Control
for Code and Data

Comprehensive
Documentation

Environment
Reproducibility

Model Registry
Naming
Conventions

Regular Model
Retraining

Security
Measures

Continuous
Integration (CI)
for MLflow
Projects

Regularly Clean
Up Unused
Experiments



Conclusion

- Recap of Key Points
- Empowering the Machine Learning Lifecycle
- Enhancing Collaboration and Reproducibility
- Flexibility in Deployment