Model Monitoring

MLflow, Tensorboard, Weights & Biases

Credit to TA.Cheetah & TA.Phu

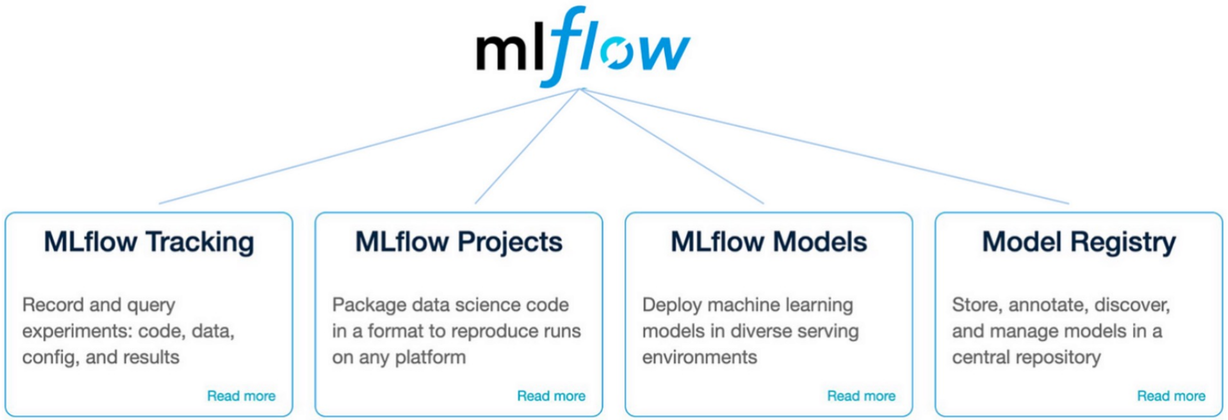
Outline

- Mlflow
- Tensorboard
- Weights & Biases

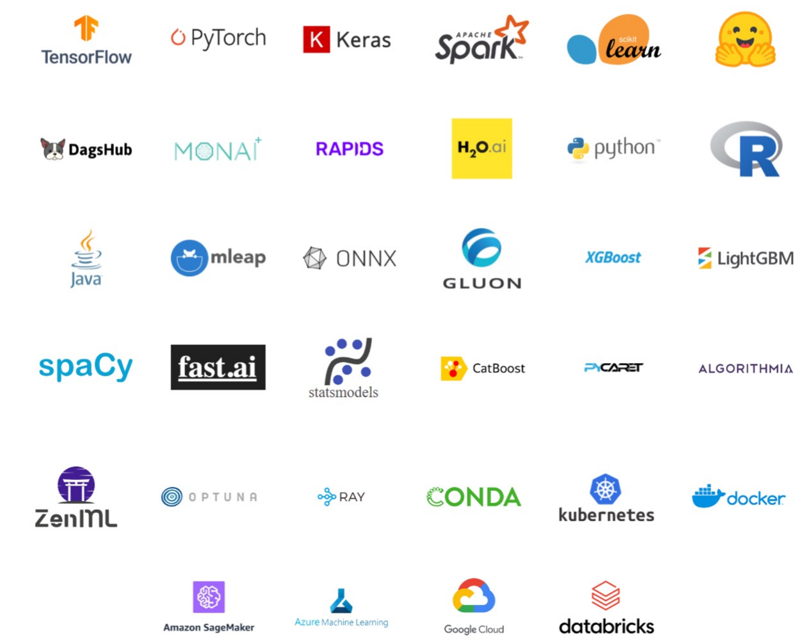
MLflow

What is MLflow?

- MLflow makes it simple to construct end-to-end Machine Learning pipelines in production, and this article will teach you all you need to know about the platform. This implies that at the conclusion of this tutorial, you'll be able to utilize MLflow for Machine Learning pipelines from model experimentation through model deployment.



Integrations with:



How to run MLflow

- Install mlflow

```
!pip install mlflow --quiet
```

Python

- Import libraries

```
# Importing all Libraries
import mlflow
import mlflow.sklearn

import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
```

Python

- Load dataset and define evaluation metrics

```
# Load and split dataset
X, Y = load_breast_cancer(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=0)
print("Training Data Shape: ", X_train.shape, y_train.shape)
print("Testing Data Shape: ", X_test.shape, y_test.shape)

def eval_metrics(actual, pred):
    accuracy = accuracy_score(actual, pred)
    return accuracy
```

Model tracking

1. Start an experiment using **mlflow.start_run()** which switches the context of your existing model code to enable mlflow tracking.
2. We log the run parameters with **mlflow.log_param()**
3. We log the model metrics (mean accuracy on the training set in this case) with **mlflow.log_metric().**
4. After model training and evaluation, I have logged the model using **mlflow.sklearn.log_model().**

```python
def train_model(criterion, max_depth):

    # Starting the Experiement
    with mlflow.start_run():
        # Model building
        model = DecisionTreeClassifier(criterion=criterion, max_depth=max_depth,random_state=0)
        model.fit(X_train, y_train)   # Model Training
        y_pred = model.predict(X_test)  # Model Prediction on Testing data
        (accuracy) = eval_metrics(y_test, y_pred)

        print('Decirion tree (criterion=%s, max_depth=%d):'%(criterion, max_depth))
        print('Accuracy: {:.4f}'.format(accuracy))

        # Logging Parameters
        mlflow.log_param("criterion", criterion)
        mlflow.log_param("max_depth", max_depth)

        # Logging Metrics
        mlflow.log_metric("accuracy", accuracy_score(y_test, y_pred))

        # Model Logging
        mlflow.sklearn.log_model(model, 'model') #input_example=input_example)

        return model
```

Python

Train model and search best 5 runs

- Train 10 models with different hyperparameters

```python
train_model('gini', 1)
```

Python

```
Decirion tree (criterion=gini, max_depth=1):
Accuracy: 0.9035
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=1, random_state=0)
```

```python
#Search best 5 runs
best_run_df = mlflow.search_runs(order_by=['metrics.accuracy DESC'], max_results=5)
best_run_df
```

Python

```
INFO  [alembic.runtime.migration] Context impl SQLiteImpl.
INFO  [alembic.runtime.migration] Will assume non-transactional DDL.
```

artifact_uri	start_time	end_time	metrics.accuracy	params.max_depth	params.criterion	tags.mlflow.sour
./mlruns/1/5d63abf908a643ac93724bd076496e19/ar...	2023-08-23 01:47:10.328000+00:00	2023-08-23 01:47:10.584000+00:00	0.964912	3	gini	/usr/local/lib/python3 packages
./mlruns/1/d845b4259e634b5cb8485012cc6bc01b/ar...	2023-08-23 01:47:10.014000+00:00	2023-08-23 01:47:10.263000+00:00	0.964912	2	gini	/usr/local/lib/python3 packages
./mlruns/1/4746bcfe0a5b4936b3350514515e0536/ar...	2023-08-23 01:47:10.639000+00:00	2023-08-23 01:47:10.961000+00:00	0.956140	4	gini	/usr/local/lib/python3 packages
./mlruns/1/09ad160ad4734e68a6044e59228c805a/ar...	2023-08-23 01:47:11.931000+00:00	2023-08-23 01:47:12.131000+00:00	0.947368	3	entropy	/usr/local/lib/python3 packages
./mlruns/1/d16fe0518d2d4bff80f343f9a58be6b4/ar...	2023-08-23 01:47:11.200000+00:00	2023-08-23 01:47:11.406000+00:00	0.947368	5	gini	/usr/local/lib/python3 packages

Load best model (MLflow models)

artifact_uri	start_time	end_time	metrics.accuracy	params.max_depth	params.criterion	tags.mlflow.sour
./mlruns/1/5d63abf908a643ac93724bd076496e19/ar...	2023-08-23 01:47:10.328000+00:00	2023-08-23 01:47:10.584000+00:00	0.964912	3	gini	/usr/local/lib/python3 packages
./mlruns/1/d845b4259e634b5cb8485012cc6bc01b/ar...	2023-08-23 01:47:10.014000+00:00	2023-08-23 01:47:10.263000+00:00	0.964912	2	gini	/usr/local/lib/python3 packages
./mlruns/1/4746bcfe0a5b4936b3350514515e0536/ar...	2023-08-23 01:47:10.639000+00:00	2023-08-23 01:47:10.961000+00:00	0.956140	4	gini	/usr/local/lib/python3 packages
./mlruns/1/09ad160ad4734e68a6044e59228c805a/ar...	2023-08-23 01:47:11.931000+00:00	2023-08-23 01:47:12.131000+00:00	0.947368	3	entropy	/usr/local/lib/python3 packages
./mlruns/1/d16fe0518d2d4bff80f343f9a58be6b4/ar...	2023-08-23 01:47:11.200000+00:00	2023-08-23 01:47:11.406000+00:00	0.947368	5	gini	/usr/local/lib/python3 packages

```python
# Load model as a PyFuncModel.
loaded_model = mlflow.pyfunc.load_model(model_uri=f"runs:/{run_id}/model") # run_id of best model

# Predict on a Pandas DataFrame.
predicted = loaded_model.predict(pd.DataFrame(X_test))
print(classification_report(y_test, predicted, target_names=['Non-DD', 'DD'], digits=4))
```

Python

```
              precision    recall  f1-score   support

      Non-DD     0.9778    0.9362    0.9565        47
          DD     0.9565    0.9851    0.9706        67

    accuracy                         0.9649       114
   macro avg     0.9671    0.9606    0.9636       114
weighted avg     0.9653    0.9649    0.9648       114
```

Model registry

- The MLflow Model Registry component is a centralized model store, set of APIs, and UI, to collaboratively manage the full lifecycle of an MLflow Model. It provides model lineage, model versioning, stage transitions (for example from staging to production), and annotations.

```python
#Register best model
mlflow.register_model(model_uri=model_uri, name="breast_cancer")
```
Python

```
Successfully registered model 'breast_cancer'.
2023/08/23 14:12:33 INFO mlflow.tracking._model_registry.client: Waiting up to 300 seconds for model version to finish creation. Model name: breast_cancer,
Created version '1' of model 'breast_cancer'.
```

- Load model from registered model

```python
model_name = "breast_cancer"
model_version = 1
# Load model as a PyFuncModel.
loaded_model = mlflow.pyfunc.load_model(model_uri=f"models:/{model_name}/{model_version}")

# Predict on a Pandas DataFrame.
predicted = loaded_model.predict(pd.DataFrame(X_test))
```
Python

```python
print(classification_report(y_test, predicted, target_names=['Non-DD', 'DD'], digits=4))
```
Python

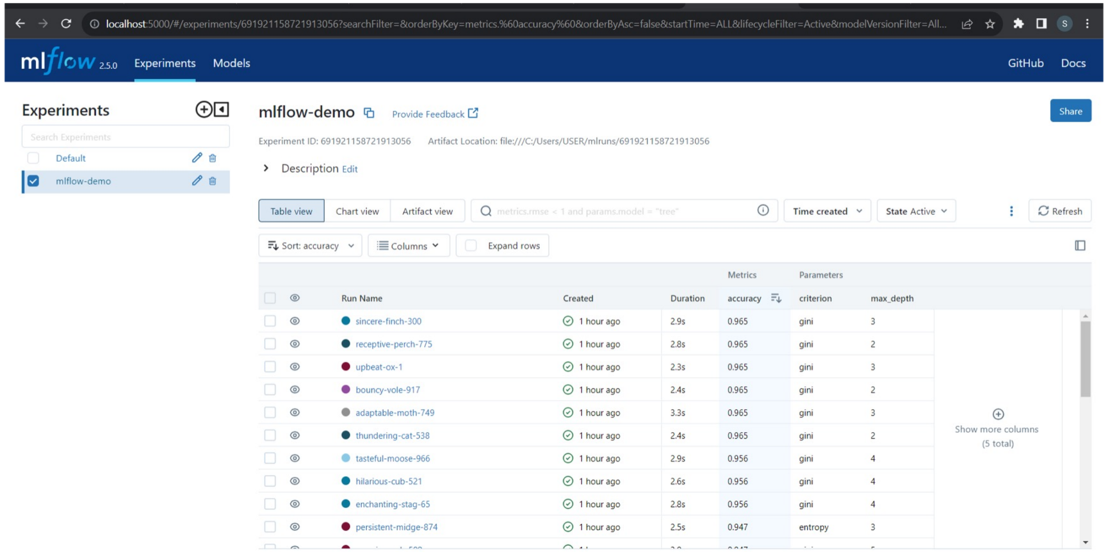
```
              precision    recall  f1-score   support

      Non-DD     0.9778    0.9362    0.9565        47
          DD     0.9565    0.9851    0.9706        67

    accuracy                         0.9649       114
   macro avg     0.9671    0.9606    0.9636       114
weighted avg     0.9653    0.9649    0.9648       114
```

MLflow UI

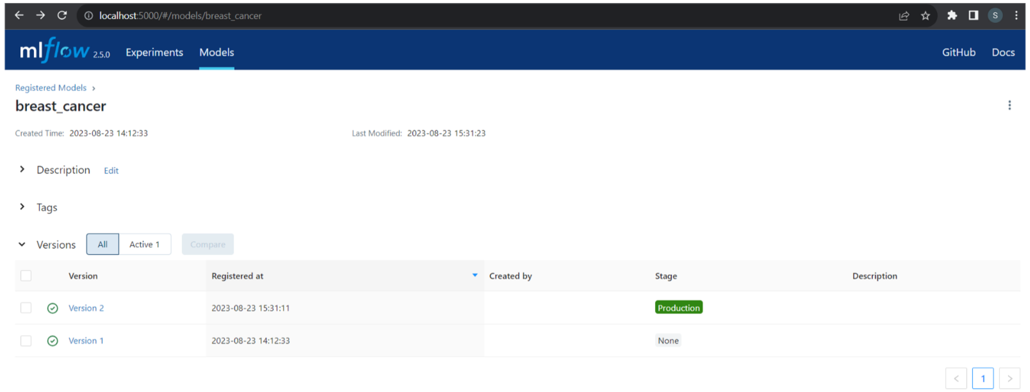
- View MLflow runs and experiments

```python
!mlflow ui
# Access this link: http://localhost:5000/
```
Python

Compare performance



Registered model



For run mlflow ui on google colab

```
# Load and split dataset
X, Y = load_breast_cancer(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=0)
print("Training Data Shape: ", X_train.shape, y_train.shape)
print("Testing Data Shape: ", X_test.shape, y_test.shape)

local_registry = "sqlite:///mlruns.db"
mlflow.set_tracking_uri(local_registry)
experiment_id = mlflow.set_experiment('test_experiment')

def eval_metrics(actual, pred):
    accuracy = accuracy_score(actual, pred)
    return accuracy
```

add 3 lines before with mlflow.start_run():

Get authtoken from https://dashboard.ngrok.com/auth

```
!pip install pyngrok --quiet
```

Python

```
from pyngrok import ngrok
ngrok.kill()

#Setting the authtoken (optional)
#Get your authtoken from https://dashboard.ngrok.com/auth
NGROK_AUTH_TOKEN = '' # Your authtoken
ngrok.set_auth_token(NGROK_AUTH_TOKEN)

# Open an HTTPs tunnel on port 5000 for http://localhost:5000
ngrok_tunnel = ngrok.connect(addr='5000', proto='http', bind_tls=True)
print("MLflow Tracking UI: ", ngrok_tunnel.public_url)
```

Python

```
WARNI [pyngrok.process.ngrok] t=2023-08-23T01:47:21+0000 lvl=warn msg="ngrok config file found at legacy location, move to XDG location" xdg_path=/root/.con
MLflow Tracking UI: https://79c5-34-136-157-242.ngrok-free.app
```

access from this link

```
!mlflow ui --backend-store-uri sqlite:///mlruns.db
```

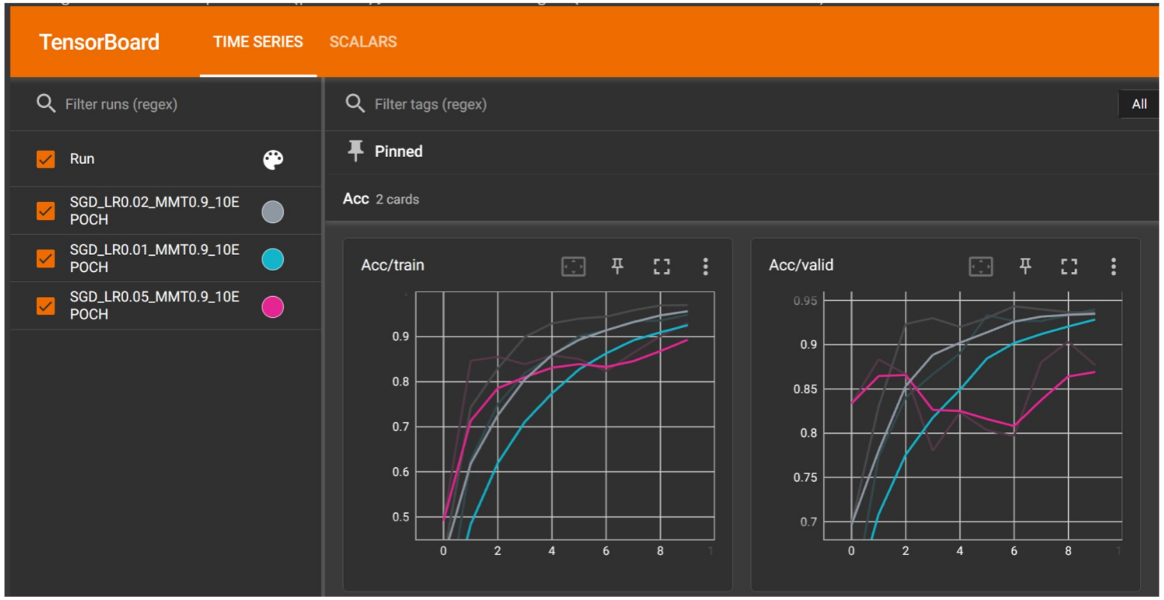
Python

Tensorboard

Guide

What is tensorboard

1) **Visualization toolkit for machine learning training**
2) **Can visualize train/validate loss, accuracy etc.**
3) **Benefits in comparing between runs (adjust hyperparameters)**



Tensorboard steps

1) install `pip install -qq tensorboard`
2) import summarywriter

```
from torch.utils.tensorboard import SummaryWriter
```

1) create directory to save log files e.g. `/content/runs/run1/`
2) instantiate writer `writer = SummaryWriter(log_dir="./runs/run1/")`
3) add scalar `writer.add_scalar("Name", value, round)`
4) write on disk `writer.flush()`
5) close `writer.close()`
6) launch tensorboard

```
%load_ext tensorboard
%tensorboard --logdir runs
```

References

[1] https://www.tensorflow.org/tensorboard

[2] https://pytorch.org/tutorials/recipes/recipes/tensorboard_with_pytorch.html

WandB

Guide

What is WandB

- Special tools by Weights & Biases for
  - *experiments tracking*
  - results visualization
  - *hyperparameter adjustment (sweep)*
  - reproduce models
  - and more!
- Create account https://wandb.ai/site
- Get API key (Need when login) https://wandb.ai/authorize

Steps: Dashboard

1) install `!pip install wandb`
2) import `import wandb`
3) login `wandb.login()  # this one is for the imported wandb library`
4) initiate

```
wandb.init(
                project="Animal-EfficientNetB0",
                config={"learning_rate": 0.02,
                              "architecture": "EfficientNetB0",
                              "dataset": "Animal2",
                              "epochs": 10}
      )
```

5) log `wandb.log({"acc": acc, "loss": loss})`
6) finish `wandb.finish()`

Steps: Sweep

1) install

    `!pip install wandb`

2) import

    `import wandb`

3) login

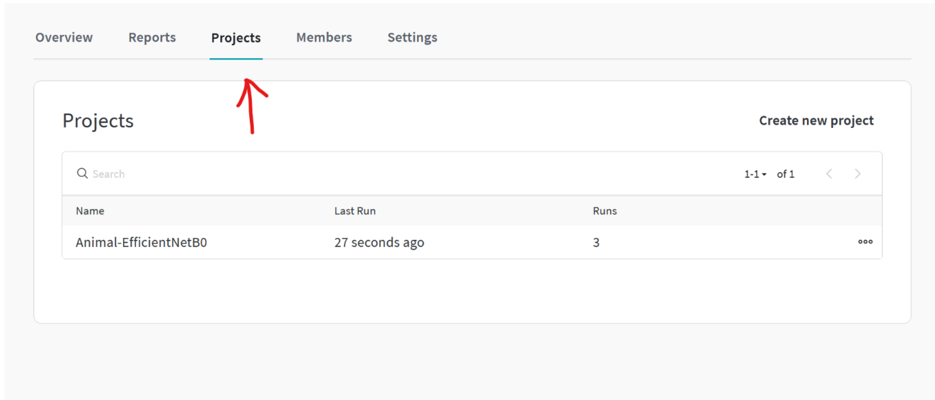
    `wandb.login()`

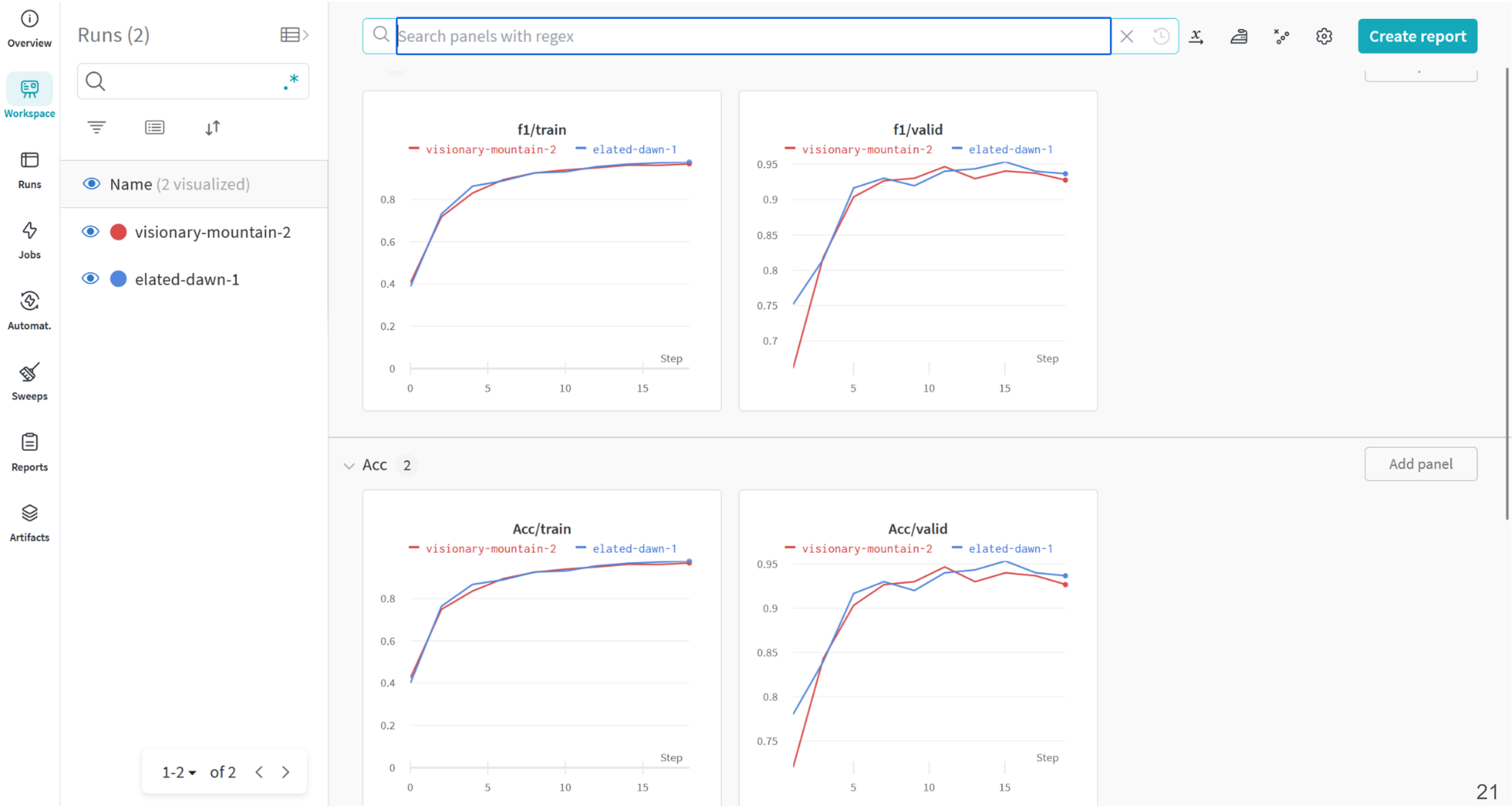
4) create config (dict) — `sweep config = dict()`
5) write your own training function — `train()`
6) write WandB training function on top — `trainer()`
7) initiate sweep (via wandb agent) — `wandb.agent(sweep_id, train)`
8) get results at your account page — https://wandb.ai/

Results

- Run history and run summary in your notebook

- Full dashboard in your wandb profile




Overview
Workspace
Runs
Jobs
Automat.
Sweeps
Reports
Artifacts
Runs (2)
Name (2 visualized)
visionary-mountain-2
elated-dawn-1
1-2 of 2
Search panels with regex
Create report
f1/train
f1/valid
Acc 2
Add panel
Acc/train
Acc/valid
Step

References

[1] https://wandb.ai/home