Summary

This lab consists of:

- Example model (`train.py` , `validate.py`)
- MLFlow/project configuration (`MLproject` , `python_env.yaml` , `requirements.*.txt`)
- Service to deply model as a REST API (`run.py`)
- Model configuration (`params.env`)
- Docker configuration for model deployment (`Dockerfile`)
- Docker configuration for MLFlow server deployment (`mlflow/Dockerfile`)
- Pipeline configuration for infrastructure and model deployment (`.github/workflows/`)
- Kubernetes configuration for model deployment and autoscaling (`k8s/*`)
- Terraform infrastructure configuration (`terraform/`)

Setup

Setup environment and install dependencies for mlflow:

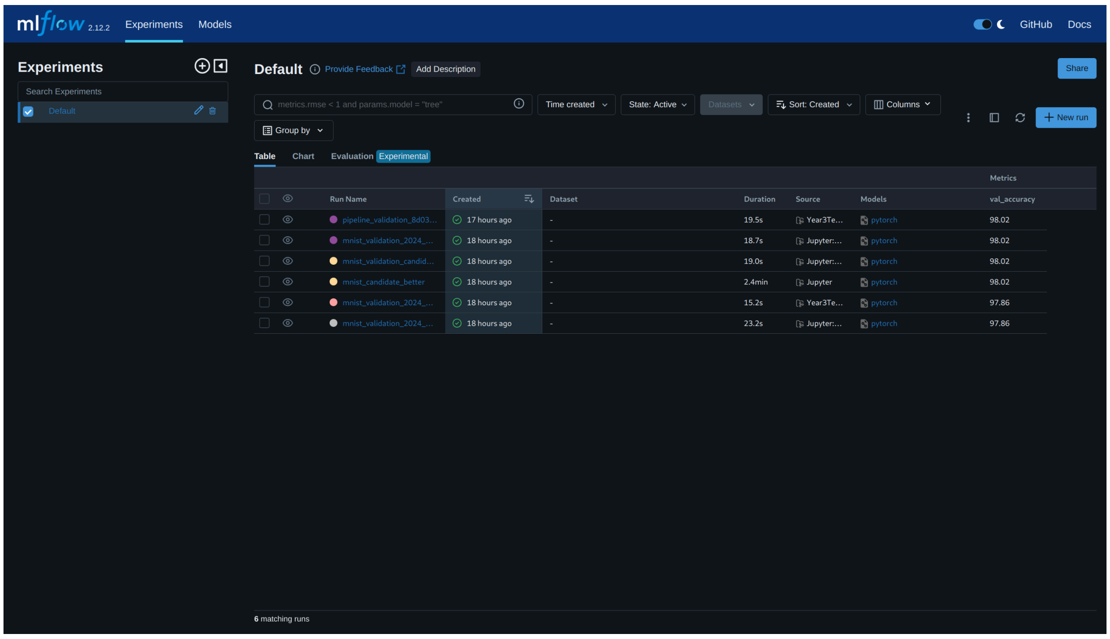
In [ ]:
```
! python3.8 -m venv .venv
! source .venv/bin/activate
! pip install -r requirements.mlflow.txt
```

Try it

As you develop the model, you can use MLFlow to store metrics and artifacts, using command:

In [ ]:
```
! MLFLOW_TRACKING_URI='https://mlflow-y3omahtamq-uc.a.run.app' mlflow run
```

Then visit the link in `MLFLOW_TRACKING_URI` , to see the effect:

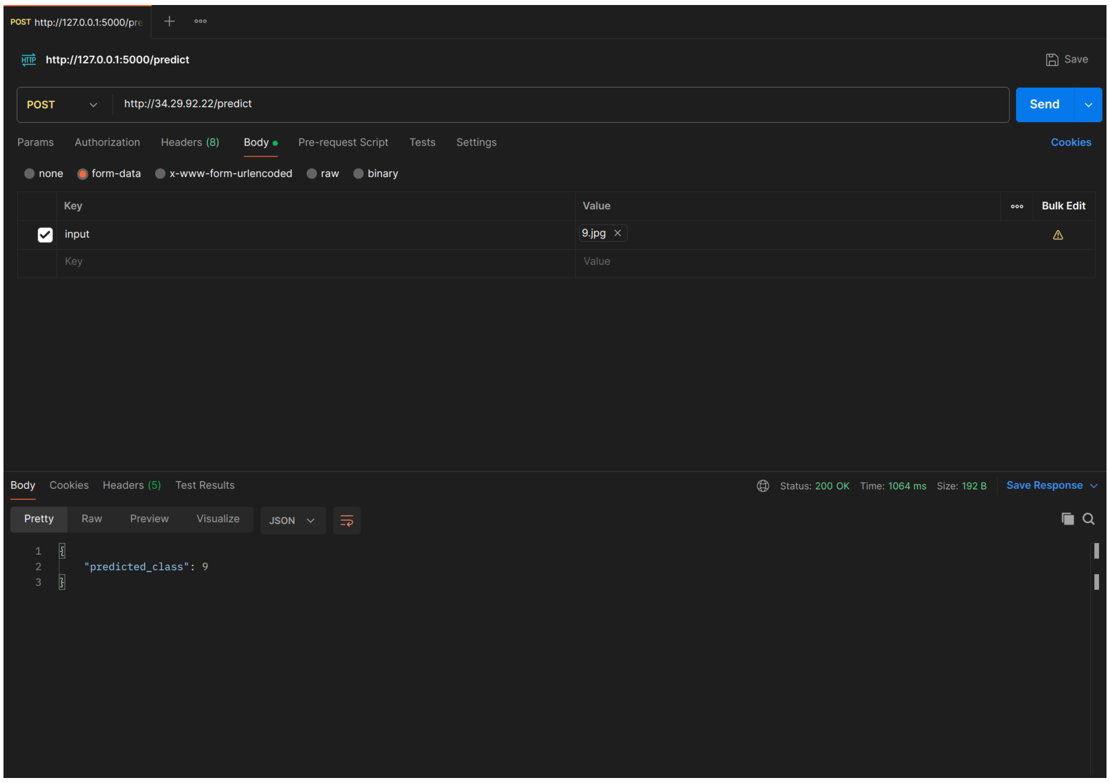


Default

Run Name	Created	Dataset	Duration	Source	Models	val_accuracy
pipeline_validation_8d03...	17 hours ago	-	19.5s	Year3Te...	pytorch	98.02
mnist_validation_2024_...	18 hours ago	-	18.7s	Jupyter:...	pytorch	98.02
mnist_validation_candid...	18 hours ago	-	19.0s	Jupyter:...	pytorch	98.02
mnist_candidate_better	18 hours ago	-	2.4min	Jupyter	pytorch	98.02
mnist_validation_2024_...	18 hours ago	-	15.2s	Year3Te...	pytorch	97.86
mnist_validation_2024_...	18 hours ago	-	23.2s	Jupyter:...	pytorch	97.86

6 matching runs

The deployed model can also be accessed, current address is `http://34.29.92.22/`. `POST` requests to `http://34.29.92.22/predict` with request body parameter `input` as an image of a number will yield a prediction. Example:



POST http://34.29.92.22/predict

Body: form-data

Key	Value
input	9.jpg

Status: 200 OK Time: 1064 ms Size: 192 B

```
{
    "predicted_class": 9
}
```

How it works

MLFlow commands

The project has several options for commands: `main`, `validate` and `deploy`. `main` trains and validates the model, `validate` performs validation only, and `deploy` builds a docker image for the model to run on the cloud as a REST service. Use `main`, or just `mlflow run .`, to develop the model, `validate` to test the model, and `deploy` to ship model to production.

Pipeline

Used in this project is GitHub Actions. The following steps are run:

- Check code formatting
- Test/validate model
- Push of metrics and artifacts (model, ROC graphs for each class) into the cloud-hosted MLFlow server
- Build model as a REST API, and package it as a docker container
- Push docker container into the artifact registry
- Apply updated Tearrform configuration, which includes kubernetes cluster, to update it to use newly built docker image

Nuances of implementation:

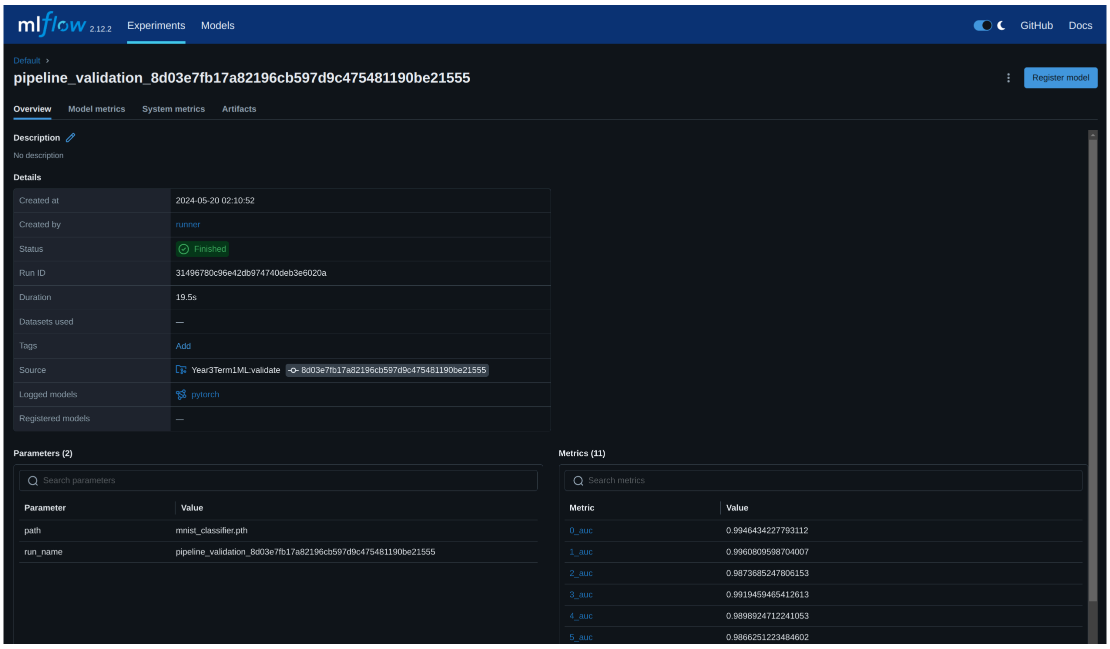
- Variables like mysql user and password and github credentials ares stored using GitHub secret manager.

Terraform layout

- Cloud storage bucket for syncing terraform state
- Managed kubernetes cluster for model deployment with horizontal autoscaling and load balancing.
- MySQL instance to store MLFlow data
- Cloud storage to store MLFlow artifacts
- IAM configuration for GitHub actions, MLFlow, Deployed instance and Kubernetes cluster
- Cloud run deployment for hosting MLFlow server docker image
- Artifact registry configuration to store REST model services
- Firewall configuration for access to MySQL database

Artifact tracking

During train runs, model learning curve of loss, and its parameters are tracked. During validation, models validation accuracy and ROC curves for each class are tracked. For all runs, model is stored to artifact registry.



pipeline_validation_8d03e7fb17a82196cb597d9c475481190be21555

Description

No description

Details

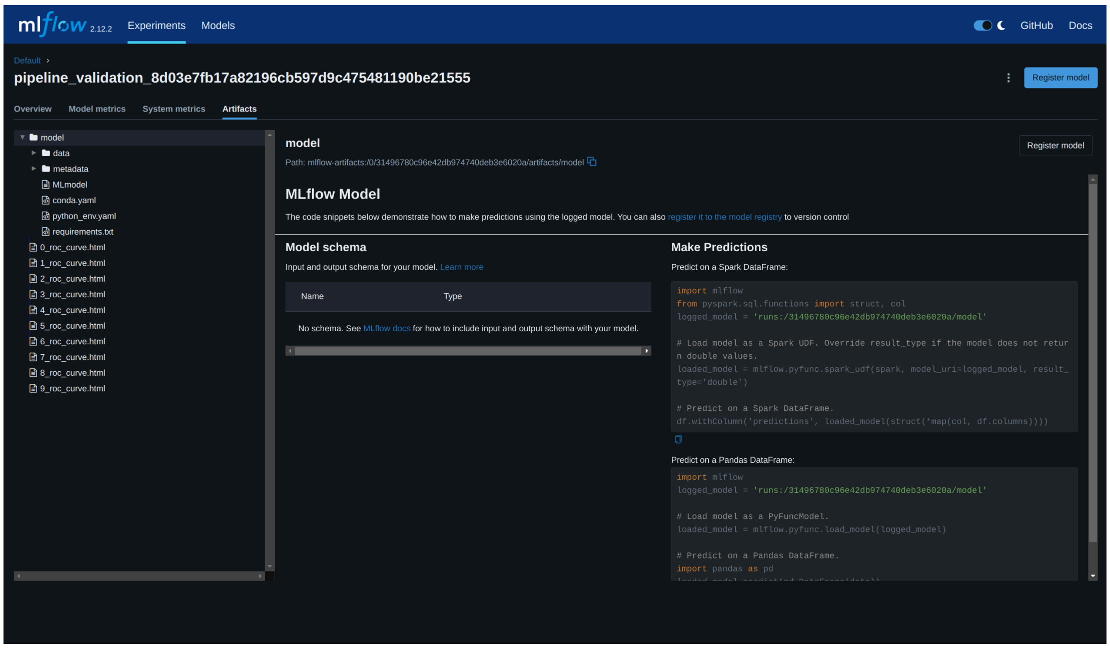
Created at	2024-05-20 02:10:52
Created by	runner
Status	Finished
Run ID	31496780c96e42db974740deb3e6020a
Duration	19.5s
Datasets used	—
Tags	Add
Source	Year3Term1ML:validate 8d03e7fb17a82196cb597d9c475481190be21555
Logged models	pytorch
Registered models	—

Parameters (2)

Parameter	Value
path	mnist_classifier.pth
run_name	pipeline_validation_8d03e7fb17a82196cb597d9c475481190be21555

Metrics (11)

Metric	Value
0_auc	0.9946434227793112
1_auc	0.9960809598704007
2_auc	0.9873685247806153
3_auc	0.9919459465412613
4_auc	0.9898924712241053
5_auc	0.9866251223484602



pipeline_validation_8d03e7fb17a82196cb597d9c475481190be21555

model

Path: mlflow-artifacts:/0/31496780c96e42db974740deb3e6020a/artifacts/model

MLflow Model

The code snippets below demonstrate how to make predictions using the logged model. You can also register it to the model registry to version control

Model schema

Input and output schema for your model. Learn more

Name	Type

No schema. See MLflow docs for how to include input and output schema with your model.

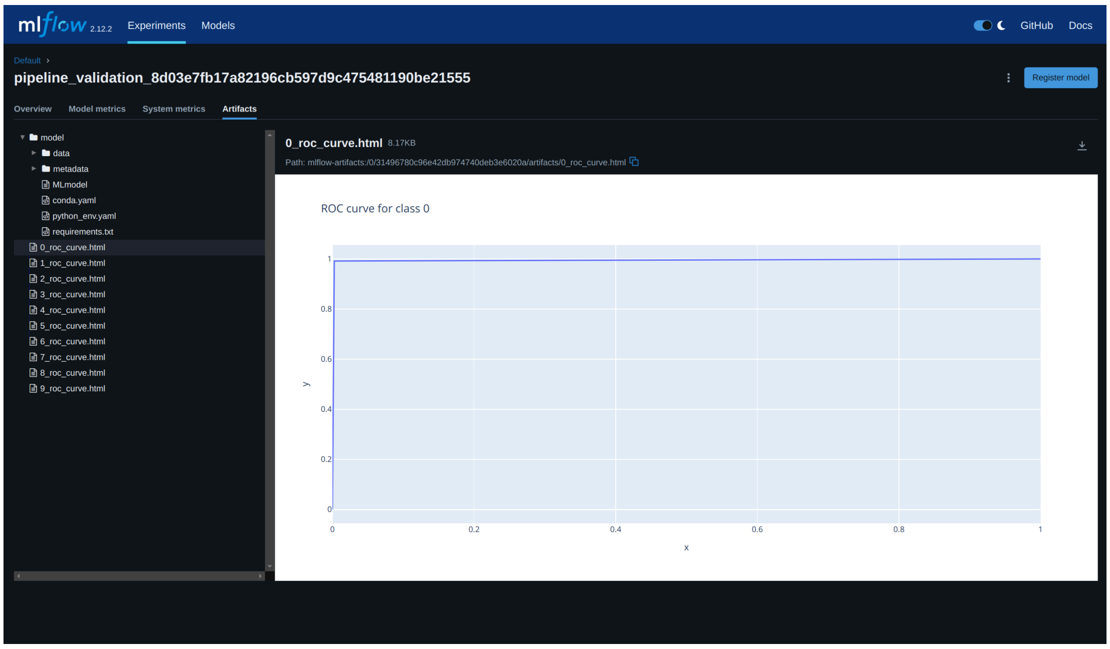
Make Predictions

Predict on a Spark DataFrame:

```
import mlflow
from pyspark.sql.functions import struct, col
logged_model = 'runs:/31496780c96e42db974740deb3e6020a/model'

# Load model as a Spark UDF. Override result_type if the model does not retur
n double values.
loaded_model = mlflow.pyfunc.spark_udf(spark, model_uri=logged_model, result_
type='double')

# Predict on a Spark DataFrame.
df.withColumn('predictions', loaded_model(struct(*map(col, df.columns))))
```

Predict on a Pandas DataFrame:

```
import mlflow
logged_model = 'runs:/31496780c96e42db974740deb3e6020a/model'

# Load model as a PyFuncModel.
loaded_model = mlflow.pyfunc.load_model(logged_model)

# Predict on a Pandas DataFrame.
import pandas as pd
```



pipeline_validation_8d03e7fb17a82196cb597d9c475481190be21555

0_roc_curve.html 8.17KB

Path: mlflow-artifacts:/0/31496780c96e42db974740deb3e6020a/artifacts/0_roc_curve.html