Hands-on MLFlow

Managing the end-to-end machine learning lifecycle in practice.

sit.org

Toolkit

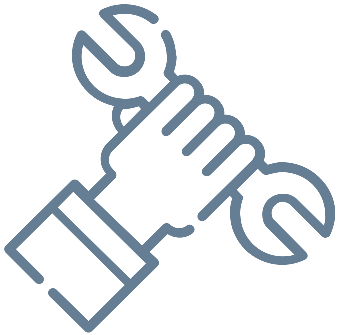
Python,

Google Colab,

Ngrok,

GitHub,

ML libraries: scikit-learn, Pandas, Numpy, Matplotlib

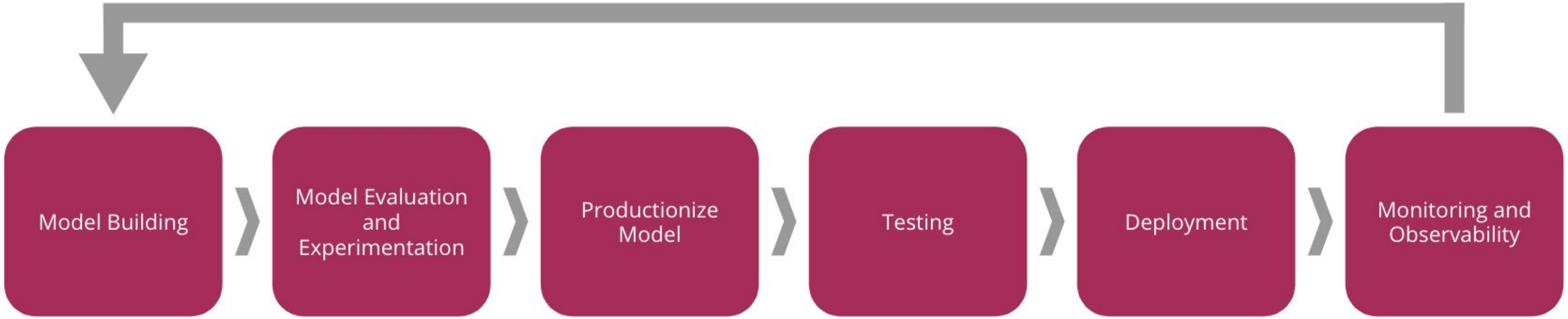


01

Introduction to MLFlow

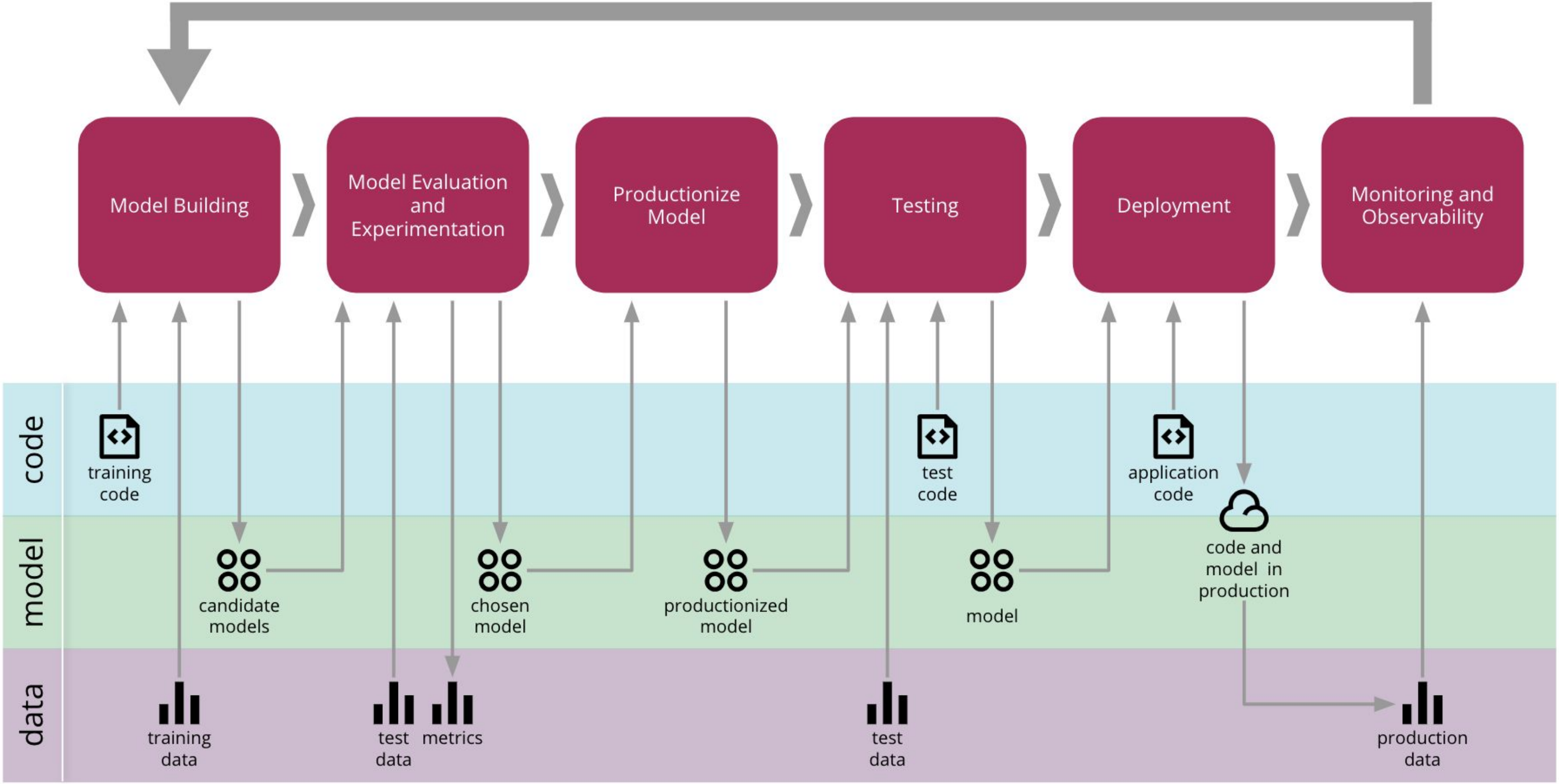
The Data Science Project Life Cycle, and how MLFlow fits in it

The Data Science Process



Source

The Data Science Process



Source

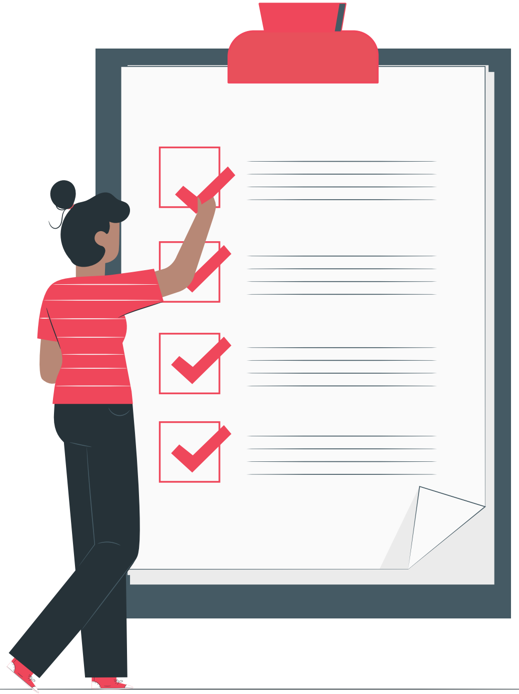
Challenges of Machine Learning Development

Goal

= Optimizing metrics. Metrics can always be improved and can degrade if models are not updated.

Quality

Depends on data, model, training parameters and code => we need to track all of these.



Tools

Data Scientists use various tools/libraries for data preparation and modelling => experiment **tracking** and **sharing** is challenging.

Deployment

A same model might need to be deployed in various environments.

MLFlow Solutions

MLFlow Tracking

Record and query experiments: code, data, config, and results

MLFlow Projects

Package data science code in a format to reproduce runs on any platform

MLFlow Models

Deploy machine learning models in diverse serving environments

MLFlow Registry

Store, annotate, discover, and manage models in a central repository

Source

Outline of the day

9:00-10:30	10:45-12:15	13:15-14:45	15:00-16:30
Introduction to MLFlow & MLFlow Tracking	MLFlow Projects	MLFlow Models	MLFlow Model Registry

02

MLFlow Tracking

log parameters, code versions, metrics, and artifacts when running your machine learning code.

MLFlow Tracking Concepts

In MLFlow tracking, a **run** is the execution of some piece of data science code. For each run, you can log:

Code Version

Git commit hash of the run

Start & End Time

Start and end Time of the run

Source

Name of the file to launch the run, or the project name and entry point for the run

Parameters

Key-value input parameters of your choice. Both keys and values are strings.

Metrics

Key-value metrics, where the value is numeric and can be updated throughout the course of the run

Artifacts

Output files in any format (images, pickled models, data files...).

Source

MLFlow Tracking Python API

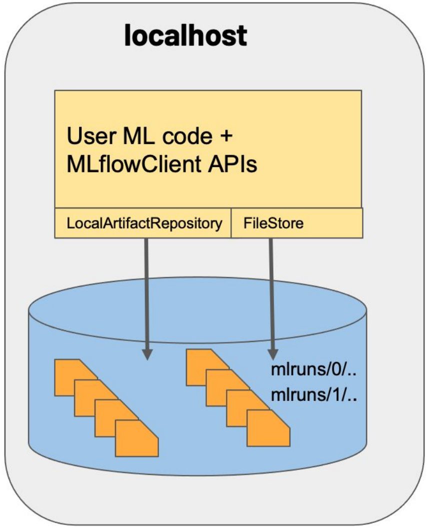
```python
import numpy as np
from sklearn.linear_model import LinearRegression
import mlflow

X = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])
y = np.dot(X, np.array([1, 2])) + 3

with mlflow.start_run():
    mlflow.sklearn.autolog()
    reg = LinearRegression().fit(X, y)
```

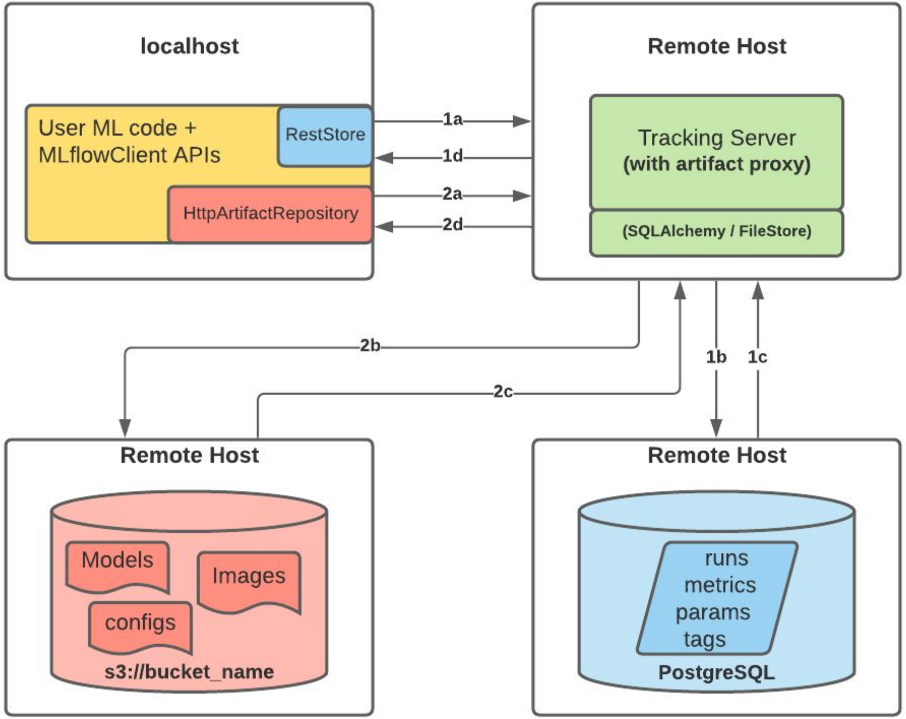
- Creates a new **run**
- Logs all **parameters** for LinearRegression
- Logs training **Metrics**
- Logs **Model**

By default, all this information is stored in the folder **mlrun**:

```
<experiment id>
├── <run id>
│   ├── artifacts
│   │   └── model
│   ├── meta.yaml
│   ├── metrics
│   ├── params
│   └── tags
└── meta.yaml
```

MLFlow Tracking Storage Configurations

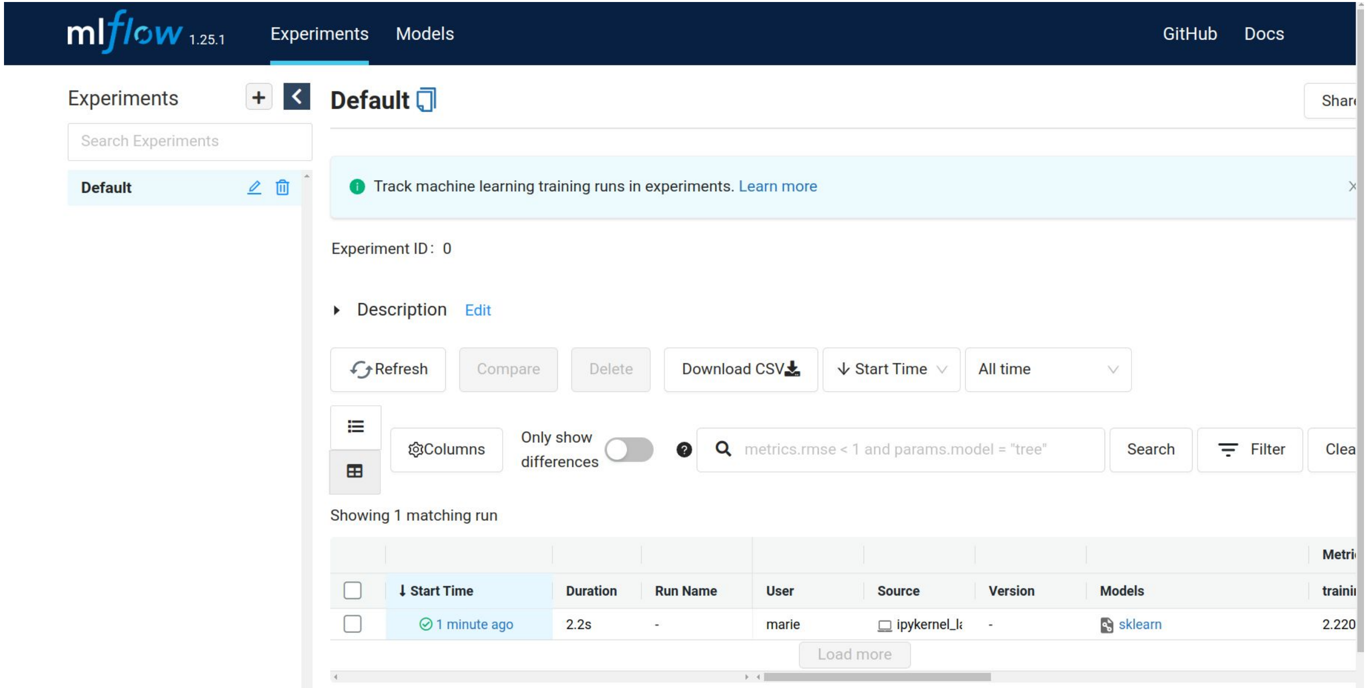




Source

MLFlow Tracking UI

Once you have executed a **run** in MLFlow tracking, you can explore all the entities you logged using the MLFlow UI:



Source

MLFlow Tracking: experiments

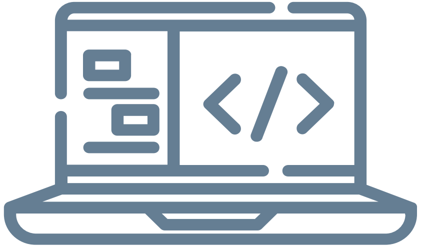
- Minimal Example
- Regression problem
- Classification Problem
- Exercise: Tracking with your own code

Tracking Livecoding!

Go to Google Colab

The notebooks are available here:
LC0_MLFlow_Tracking_MinExample

LC1_MLFlow_Tracking_Regression_SwissHousing



03

MLFlow Projects

package data science code in a
reusable way

Overview

- Concepts and motivation
- Documentation
- Build and share an MLflow Project with Git

MLFlow Projects

MLFlow Projects

Package data science code in a format to reproduce runs on any platform

- <u>Reproducibility</u>

...and how to not become another statistic in the **reproducibility crisis**

MLFlow Projects - Virtual environments

Reproducibility:

- Data (processing methods)
- Code (package versions, logging)

Other benefits:

- Platform-agnostic
- Automation
- Troubleshooting

MLFlow Projects - Virtual environments

Reproducibility:

- Code (package versions, logging)

MLFlow Projects – *MLProject* YAML file

***MLProject**.yaml*

- **Entry points**
  - Commands run within project
  - ***Need at least one!***
- **Parameters**
  - Variables to be passed to entry point
    - *File paths, options, etc.*
- **Environment**
  - Conda, Docker, venv, etc.

MLFlow Projects – *MLProject* YAML file

MLProject.yaml

```
name: My Project

conda_env: my_env.yaml
# Can have a docker_env instead of a conda_env, e.g.
# docker_env:
#    image:  mlflow-docker-example

entry_points:
  main:
    parameters:
      data_file: path
      regularization: {type: float, default: 0.1}
    command: "python train.py -r {regularization} {data_file}"
```

Project name

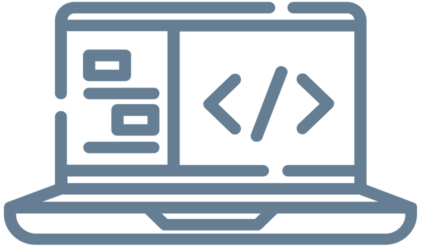
Environment type
& config file

Final command to be
run in terminal

Projects Livecoding!

Go to Google Colab

The notebook is available here:
LC2_MLFlow_Projects_Regression_SwissHousing



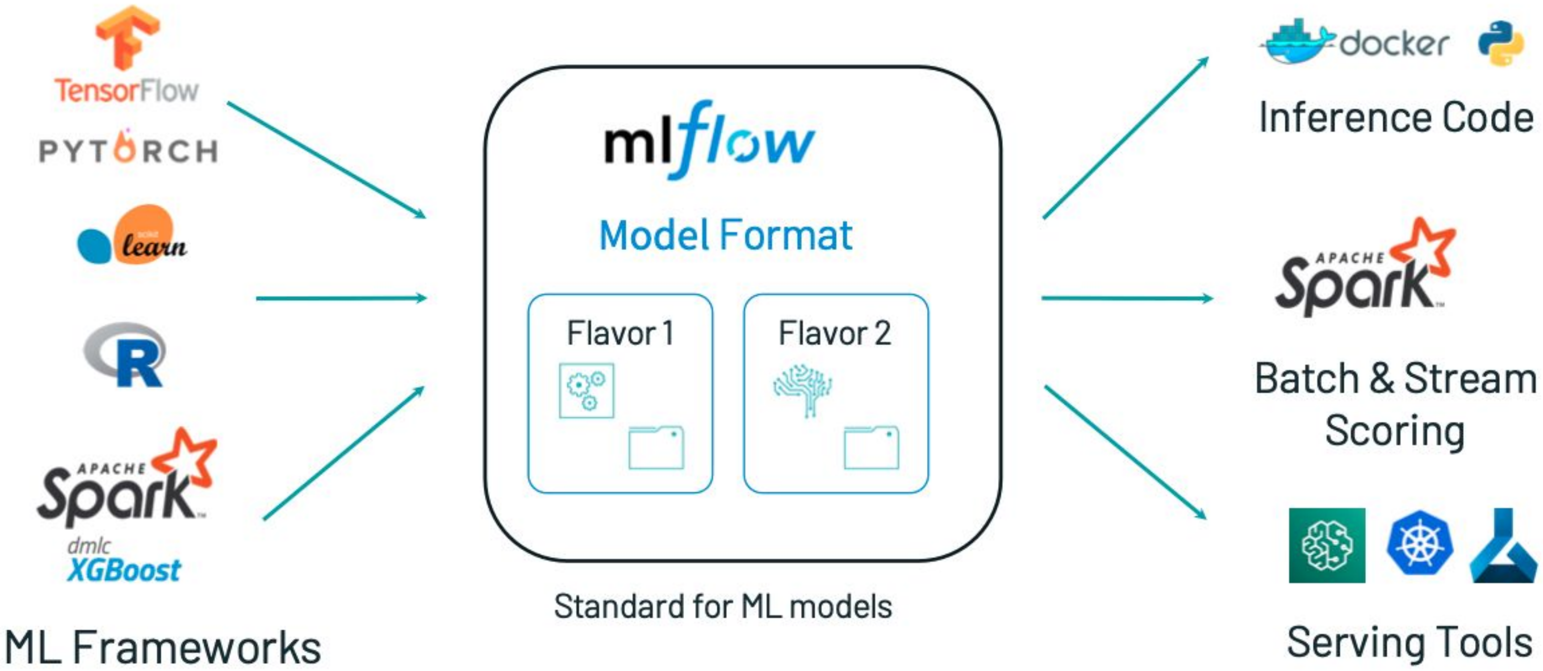
04

MLFlow Models

Package Machine Learning Models
and deploy them

MLFlow Models Motivation:

Deploy machine learning models in diverse serving environments



Source

MLFlow Models Documentation:

Deploy machine learning models in diverse serving environments

- Storage Format, Model Flavors
- Model API, Model Evaluation
- Deployment Tools

Source

Example MLFlow Model (option 1):

By default, all this information is stored in the folder model:

```
model/
├── MLmodel
├── model.pkl
├── conda.yaml
├── python_env.yaml
└── requirements.txt
```

```
artifact_path: model
  flavors:
    python_function:
      env: conda.yaml
      loader_module: mlflow.sklearn
      model_path: model.pkl
      python_version: 3.7.13
    sklearn:
      code: null
      pickled_model: model.pkl
      serialization_format: cloudpickle
      sklearn_version: 1.0.2
  mlflow_version: 1.26.1
  model_uuid: 07d32968a75345a9b28a8cd8f0b3f093
  run_id: f5d7ac67f6454523a880adc52f72242b
  utc_time_created: '2022-06-09 11:12:27.594199'
```

Usable by:

Any tool that can run Python (Docker, Spark,etc)

Tools that under-stand Scikit-Learn model format

Built-In Model Flavors

Model Scikit-Learn Flavor Example

Train a model

```
mlflow.sklearn.log_model(pipeline_rf, "model")
```

model format

Flavor 1:
python_function

```
predict =
mlflow.pyfunc.load_model(...)

predict(pandas.input_dataframe)
```

Flavor 2:
sklearn

```
model =
mlflow.sklearn.load_model(...)

model.predict(data['X_test'])
```

Model Scikit-Learn API and Evaluation

Model API

Save and load MLflow Models in multiple ways:

1) mlflow.sklearn contains save_model, log_model, and load_model functions for scikit-learn models.

2) mlflow.models.Model class to create and write models.

Model Evaluation

mlflow.evaluate()* API to evaluate MLflow Model performance

*currently supports the python_function (pyfunc) model flavor for classification and regression tasks, evaluation results are logged to MLflow Tracking

Examples:

```
import mlflow
import mlflow.sklearn

iris = load_iris()
sk_model = tree.DecisionTreeClassifier()
sk_model = sk_model.fit(iris.data, iris.target)
# set the artifact_path to location where experiment artifacts will be saved

…

# log model
mlflow.sklearn.log_model(sk_model, "sk_models")


with mlflow.start_run() as run:
    model_info = mlflow.sklearn.log_model(model, "model")
    result = mlflow.evaluate(
        model_info.model_uri,
        eval_data,
        targets="label",
        model_type="classifier",
        dataset_name="adult",
        evaluators=["default"],
    )
```

MLFlow Models Deployment:

Built-In Deployment Tools

- on a local machine
- to several production environments

Deploy MLflow models

Deploy a python_function model on Microsoft Azure ML

Deploy a python_function model on Amazon SageMaker

Export a python_function model as an Apache Spark UDF

Not all deployment methods are available for all model flavors.

Example:

- Serve a model saved with MLflow.
- Launch a webserver on the specified host and port. Supports models with the python_function or crate (R Function) flavor.
- Make requests to POST /invocations in pandas split- or record-oriented formats.

```
$ mlflow models serve -m
runs:/my-run-id/model-path &
```

```
$ curl http://127.0.0.1:5000/invocations -H
'Content-Type: application/json' -d '{
    "columns": ["a", "b", "c"],
    "data": [[1, 2, 3], [4, 5, 6]]
 }'
```

MLFlow Models Summary:

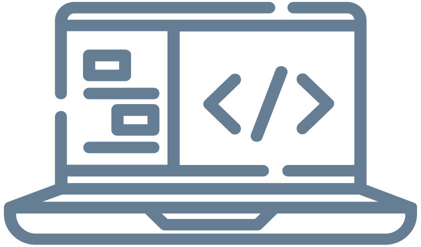
- Packaging format for ML Models
    - Any directory with MLmodel file
- Defines dependencies for reproducibility
    - Conda environment can be specified in MLmodel configuration
- Model creation and loading utilities
    - mlflow.<model_flavor>.save_model(...) or log_model(...)
    - mlflow.<model_flavor>.load_model(...)
- Deployment APIs
    - CLI/Python/R/Java
    - mlflow models [OPTIONS] COMMAND [ARGS]...
        - mlflow models serve [OPTIONS [ARGS] ....
        - mlflow models predict [OPTIONS [ARGS] ...

Models Livecoding!

Go to Google Colab

The notebooks are available here:
LC3_MLFlow_Models_Regression_SwissHousing

LC3_MLFlow_Models_Exe



05

MLFlow Model Registry

get familiar with the centralized
model store

MLFlow Model Registry - Data Stores

mlflow™

Artifact store

Model files, plots, config files, etc.

(./mlruns, S3 bucket, etc.)

Backend store

Metrics, IDs, parameters, etc.

(Local file, SQL DB, etc.)

Artifact store

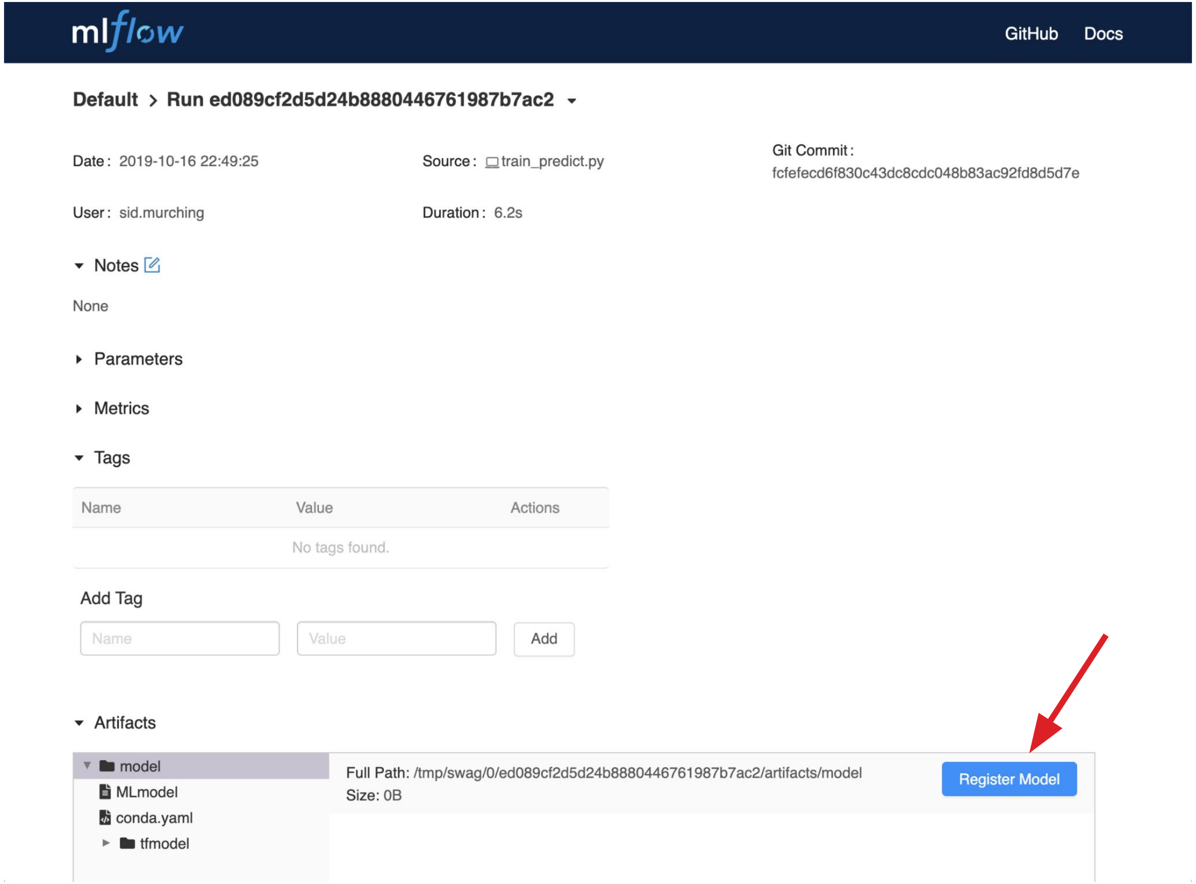
Model files, plots, config files, etc.

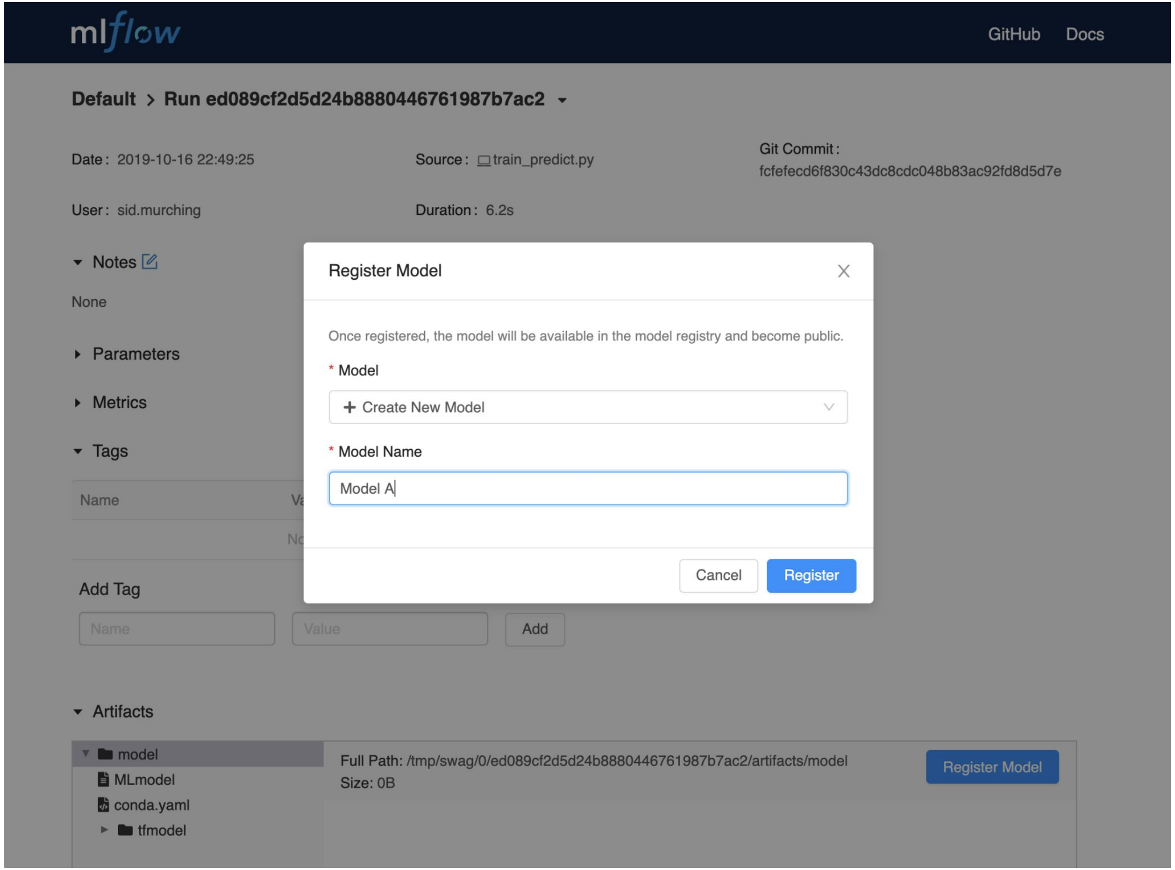
(./mlruns, S3 bucket, etc.)

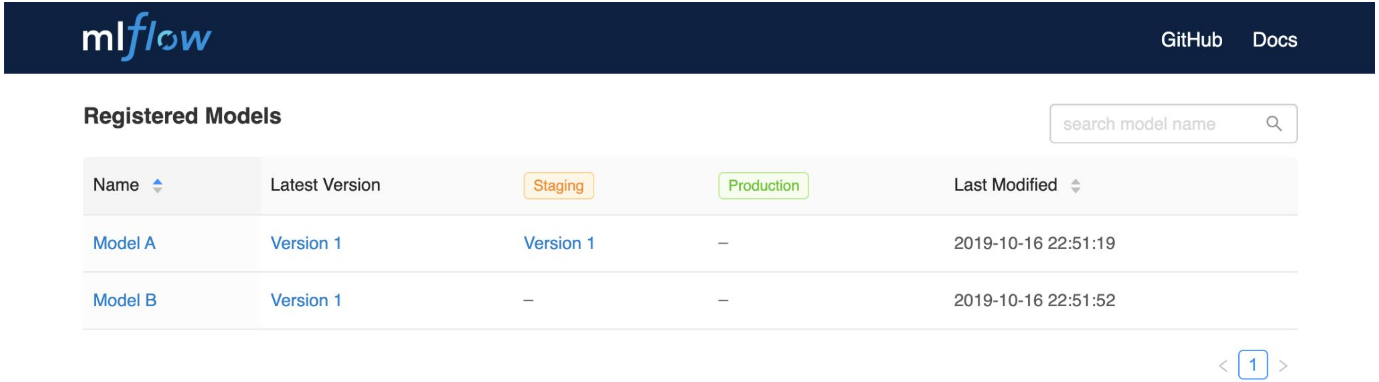
- Register different versions *(v.1, v.2, v.n)*
- Register different stages *(staging, production, etc.)*

MLFlow Model Registry - Model Store

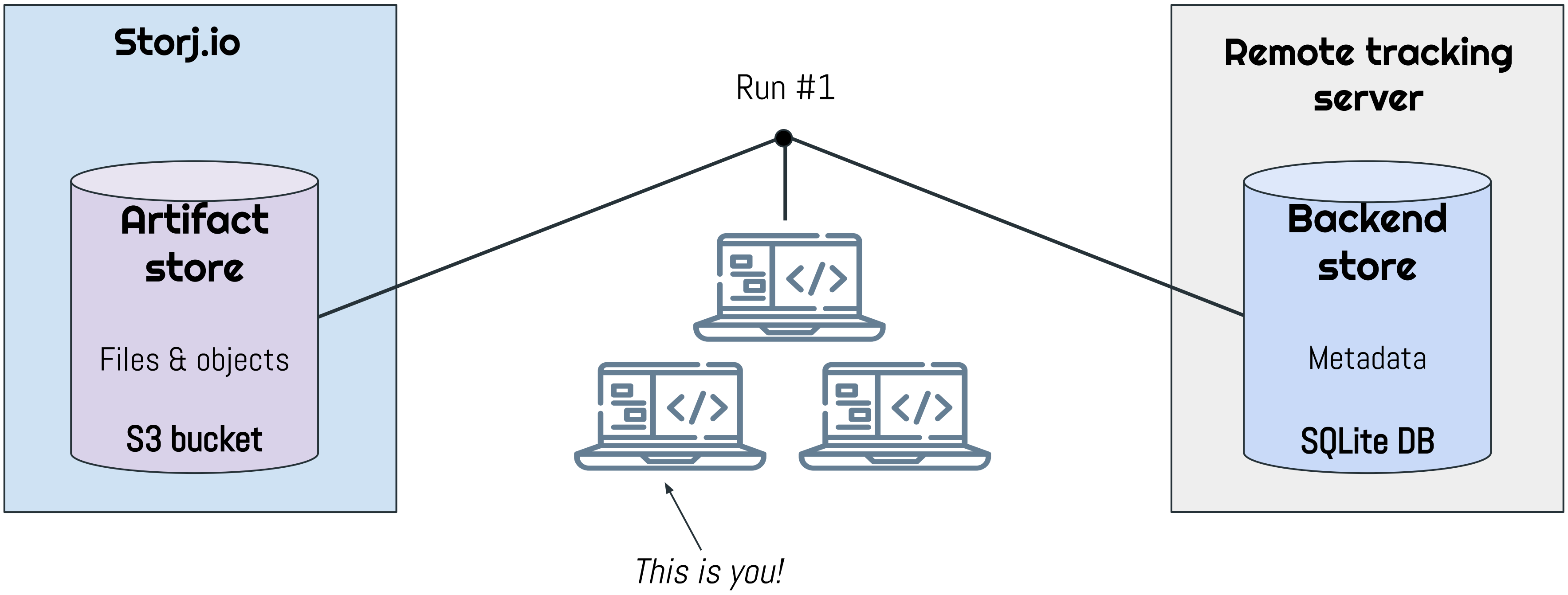
- Model management API
  - From the beginning to the end of a project
- Full model lineage
  - Versioning, stage, MLFlow experiment metadata (run, date, metrics, etc.)
- Higher level annotations
- Easy accessibility for deployment


mlflow
GitHub
Docs
Default > Run ed089cf2d5d24b8880446761987b7ac2
Date : 2019-10-16 22:49:25
Source : train_predict.py
Git Commit :
fcfefecd6f830c43dc8cdc048b83ac92fd8d5d7e
User : sid.murching
Duration : 6.2s
Notes
None
Parameters
Metrics
Tags
Name
Value
Actions
No tags found.
Add Tag
Name
Value
Add
Artifacts
model
MLmodel
conda.yaml
tfmodel
Full Path: /tmp/swag/0/ed089cf2d5d24b8880446761987b7ac2/artifacts/model
Size: 0B
Register Model


mlflow
GitHub
Docs
Default > Run ed089cf2d5d24b8880446761987b7ac2
Date: 2019-10-16 22:49:25
Source: train_predict.py
Git Commit: fcfefecd6f830c43dc8cdc048b83ac92fd8d5d7e
User: sid.murching
Duration: 6.2s
Notes
None
Parameters
Metrics
Tags
Name
Add Tag
Name
Value
Add
Artifacts
model
MLmodel
conda.yaml
tfmodel
Full Path: /tmp/swag/0/ed089cf2d5d24b8880446761987b7ac2/artifacts/model
Size: 0B
Register Model
Register Model
Once registered, the model will be available in the model registry and become public.
* Model
Create New Model
* Model Name
Model A
Cancel
Register


mlflow
GitHub
Docs
Registered Models
search model name
Name
Latest Version
Staging
Production
Last Modified
Model A
Version 1
Version 1
–
2019-10-16 22:51:19
Model B
Version 1
–
–
2019-10-16 22:51:52
1

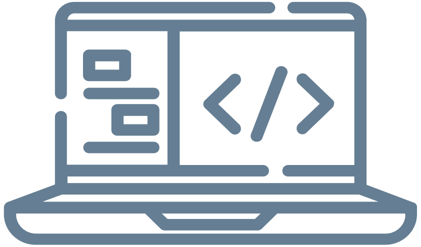
Registry Livecoding!



Registry Livecoding!

Go to Google Colab

The notebooks are available here:
LC4_MLFlow_Registry_Regression_SwissHousing



Sources and References

- Continuous Delivery for Machine Learning
- Databrick’s 3-part tutorial: Managing the Machine Learning Lifecycle using MLFlow
- MLFlow’s official Documentation


SDS2022
WORKSHOPS
SWISS CONFERENCE
ON DATA SCIENCE
Workshop Feedback
data innovation alliance