MLOPs Zoomcamp Week 2 (part 1): Experiment Tracking

Gabriel

2.1 Experiment Tracking Intro

Experiment Tracking

- ML experiment: the process of building an ML model
- Experiment run: each trial in an ML exp
- Run artifact: any file that is associated with an ML run
- Experiment metadata

Tracking relevant information

- Source code
- Environment
- Data
- Model
- Hyperparameters
- Metrics
- ...

For:

- Reproducibility
- Organization
- Optimization

MLflow

- Open source tool for ML lifecycle (python package)
- Four main modules:
  - Tracking
  - Models (special type of model)
  - Model Registry
  - Projects

Allows you to keep track of..

- Parameters
- Metrics
- Metadata: tags (name of the developer, algorithm)
- Artifacts: visualization, dataset (but hard to scale), dictionary vectorizer
- Models

Also auto log:

- Source code
- Version of the code (git commit)
- Start and end time
- Author

Install and run (with sqlite backend)

$pip install mlflow

$mlflow ui –backend-store-uri sqlite:///mlflow.db

```
(exp-tracking-env) → experiment_tracking git:(main) × mlflow ui --backend-store-uri sqlite:/
//mlflow.db
[2022-05-09 16:53:53 +0200] [6995] [INFO] Starting gunicorn 20.1.0
[2022-05-09 16:53:53 +0200] [6995] [INFO] Listening at: http://127.0.0.1:5000 (6995)
[2022-05-09 16:53:53 +0200] [6995] [INFO] Using worker: sync
[2022-05-09 16:53:53 +0200] [6996] [INFO] Booting worker with pid: 6996
```

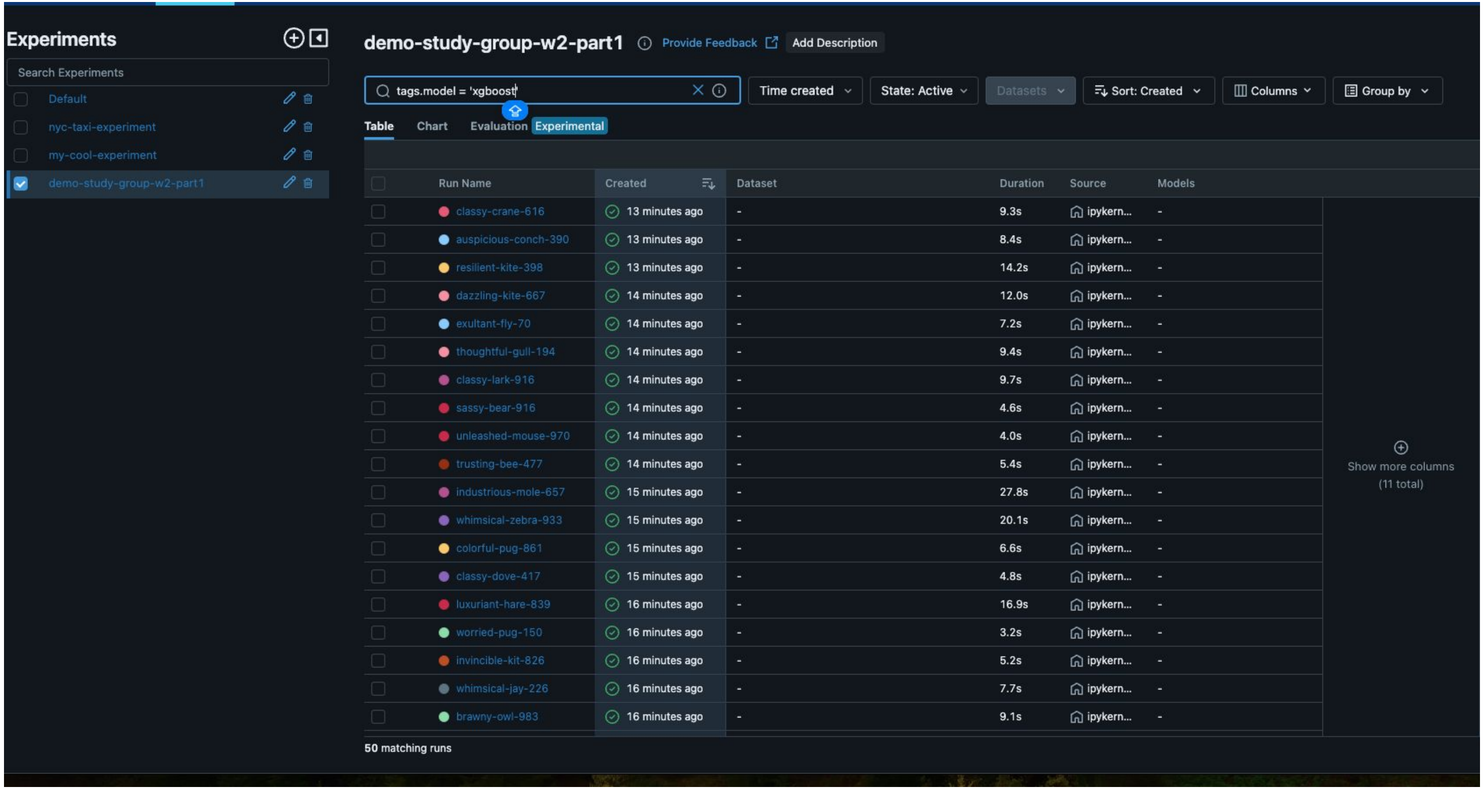
2.2 Getting Started with MLflow

In notebook file

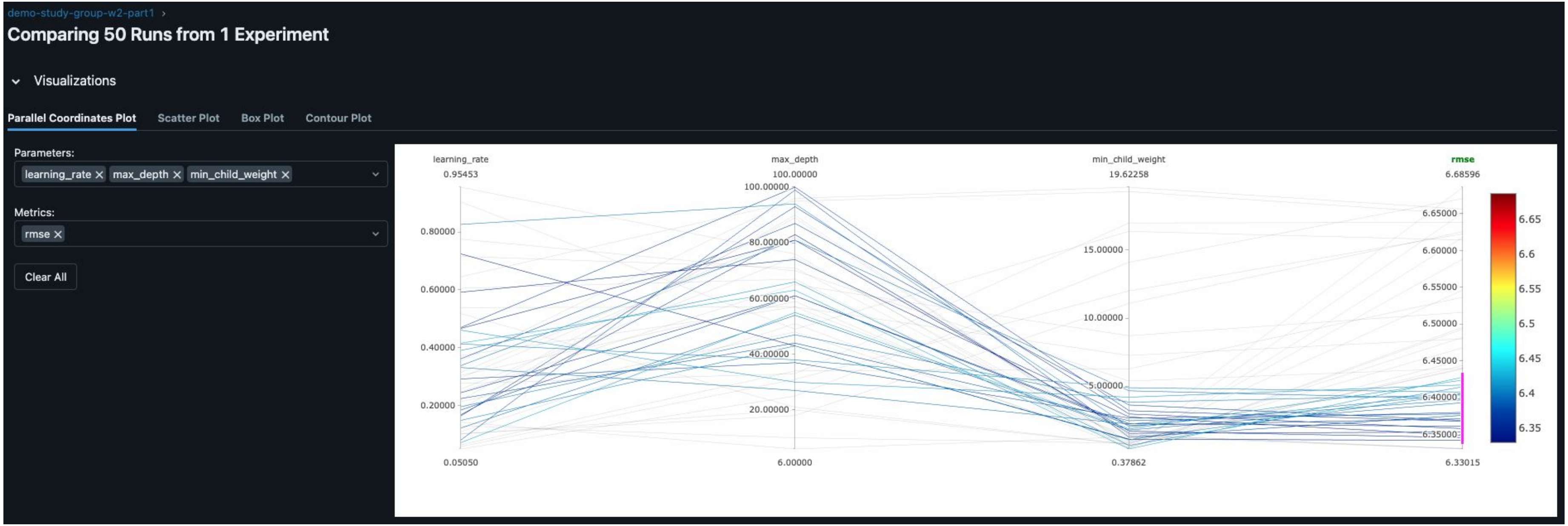
- Check duration-prediction.ipynb

2.3 Experiment Tracking with MLflow

2.3 Tags


Experiments
Search Experiments
Default
nyc-taxi-experiment
my-cool-experiment
demo-study-group-w2-part1
demo-study-group-w2-part1
Provide Feedback
Add Description
tags.model = 'xgboost'
Time created
State: Active
Datasets
Sort: Created
Columns
Group by
Table
Chart
Evaluation
Experimental
Run Name
Created
Dataset
Duration
Source
Models
classy-crane-616 13 minutes ago - 9.3s ipykern... -
auspicious-conch-390 13 minutes ago - 8.4s ipykern... -
resilient-kite-398 13 minutes ago - 14.2s ipykern... -
dazzling-kite-667 14 minutes ago - 12.0s ipykern... -
exultant-fly-70 14 minutes ago - 7.2s ipykern... -
thoughtful-gull-194 14 minutes ago - 9.4s ipykern... -
classy-lark-916 14 minutes ago - 9.7s ipykern... -
sassy-bear-916 14 minutes ago - 4.6s ipykern... -
unleashed-mouse-970 14 minutes ago - 4.0s ipykern... -
trusting-bee-477 14 minutes ago - 5.4s ipykern... -
industrious-mole-657 15 minutes ago - 27.8s ipykern... -
whimsical-zebra-933 15 minutes ago - 20.1s ipykern... -
colorful-pug-861 15 minutes ago - 6.6s ipykern... -
classy-dove-417 15 minutes ago - 4.8s ipykern... -
luxuriant-hare-839 16 minutes ago - 16.9s ipykern... -
worried-pug-150 16 minutes ago - 3.2s ipykern... -
invincible-kit-826 16 minutes ago - 5.2s ipykern... -
whimsical-jay-226 16 minutes ago - 7.7s ipykern... -
brawny-owl-983 16 minutes ago - 9.1s ipykern... -
Show more columns
(11 total)
50 matching runs

2.3 compare Xgboost tags



Auto logging

Automatic Logging

Automatic logging allows you to log metrics, parameters, and models without the need for explicit log statements.

There are two ways to use autologging:

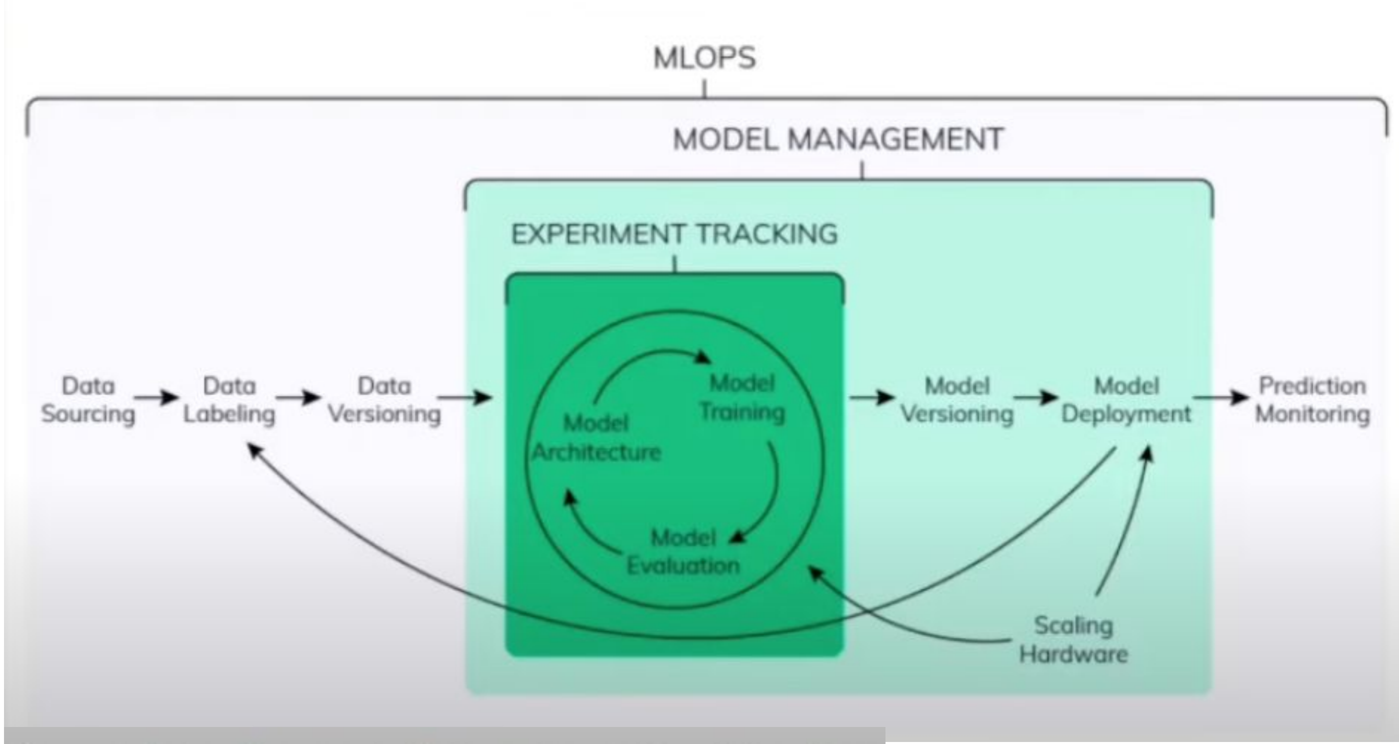
1. Call `mlflow.autolog()` before your training code. This will enable autologging for each supported library you have installed as soon as you import it.
2. Use library-specific autolog calls for each library you use in your code. See below for examples.

The following libraries support autologging:

- Scikit-learn
- TensorFlow and Keras
- Gluon
- XGBoost
- LightGBM
- Statsmodels
- Spark
- Fastai
- Pytorch

2.4 Model Management

ML Lifecycle

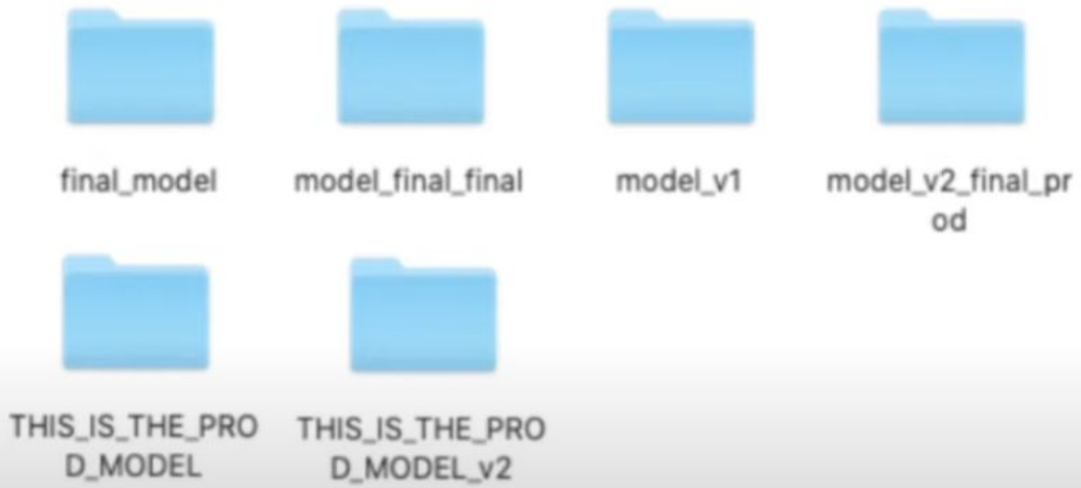


Source: https://neptune.ai/blog/ml-experiment-tracking

Model management

What's wrong with this?

- Error prone
- No versioning
- No model lineage

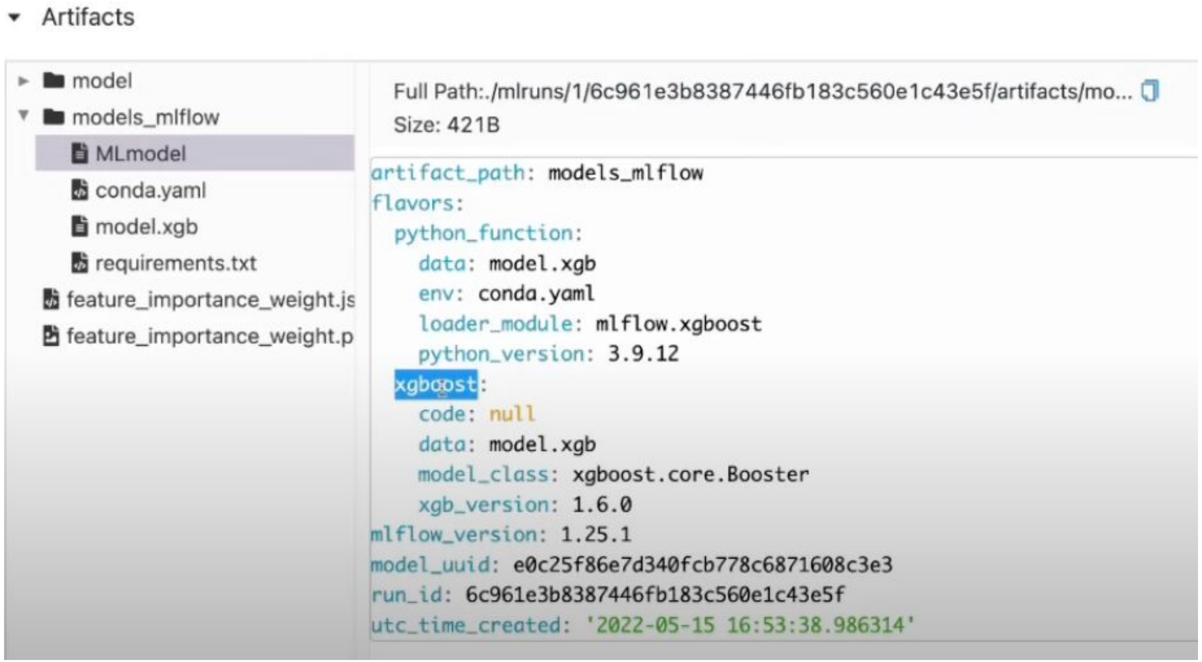


Log artifacts

```python
# 2.4 log artifacts
 mlflow.log_artifact(local_path='./models/lin_reg.bin' ,artifact_path='models_pickle')
```

→ Not good, don’t have instruction of how to use the model

After logging the model, 2 ways to load

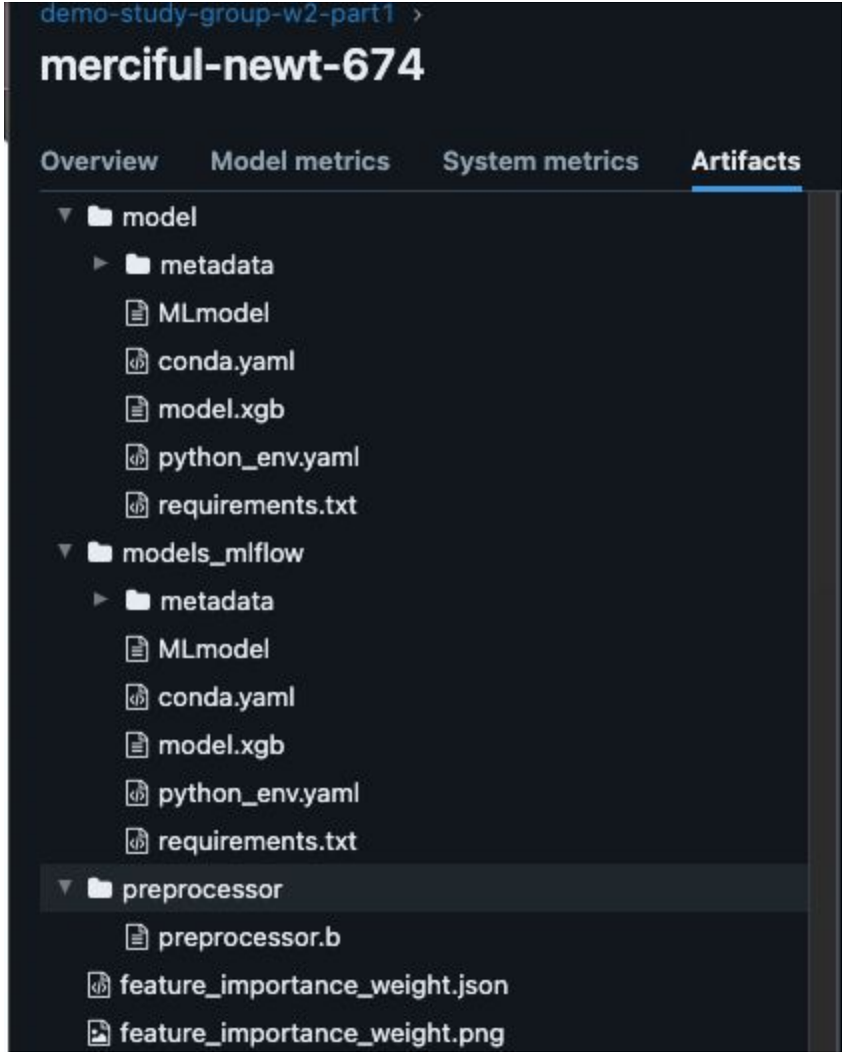


Logged model, artifacts

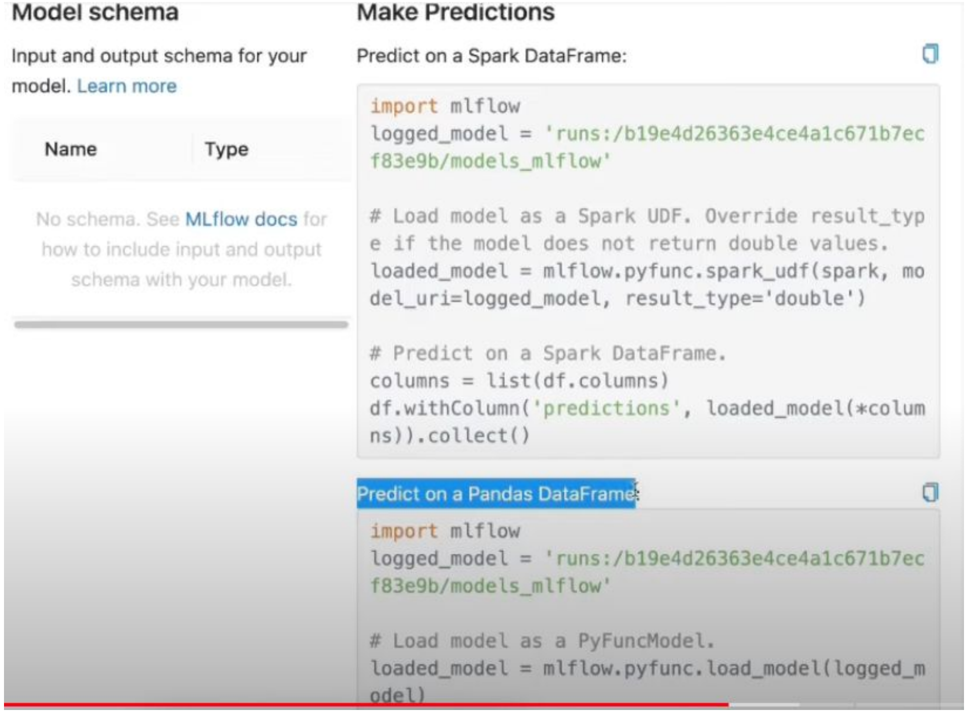
Model: auto logging

Models_mlflow: manual logging

Preprocessor: artifacts for dictionary vectorizer



Load model and make predictions



Recap of logging models in MLflow

Two options:

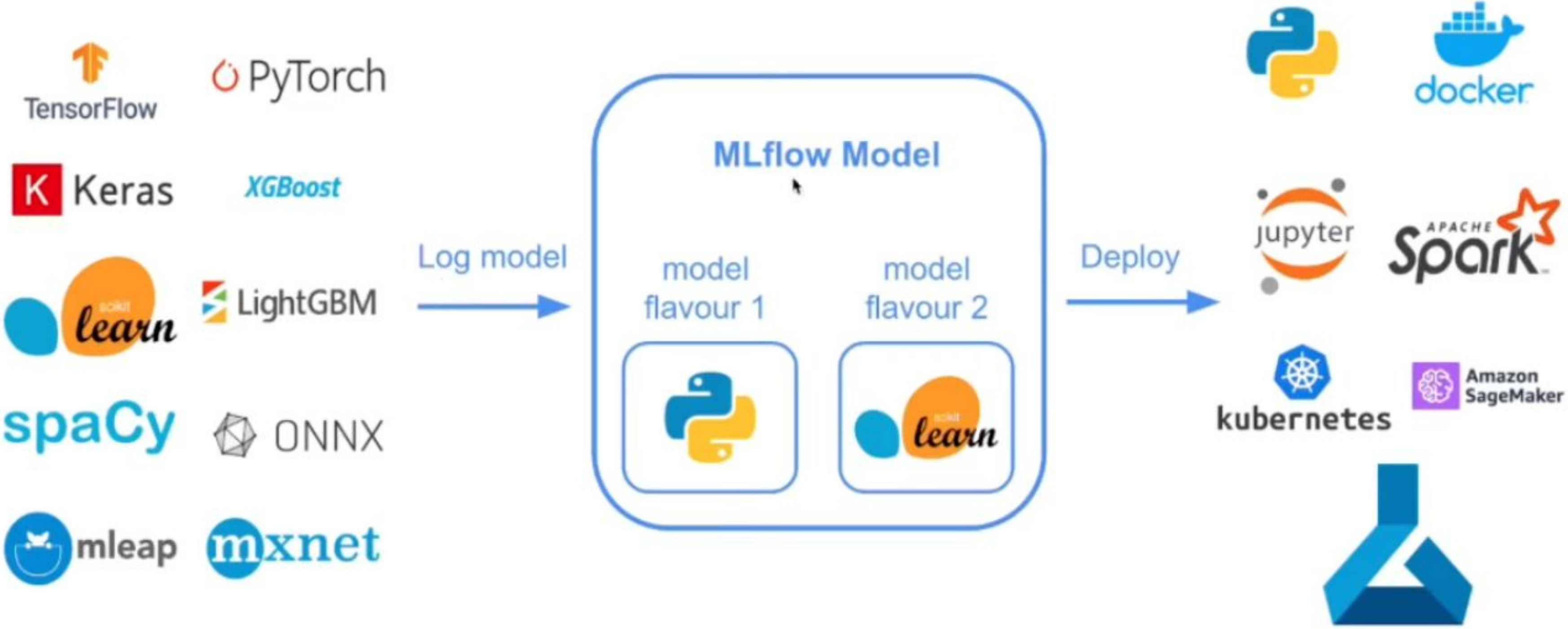
- Log model as an artifact

```
mlflow.log_artifact("mymodel", artifact_path="models/")
```

- Log model using the method `log_model`

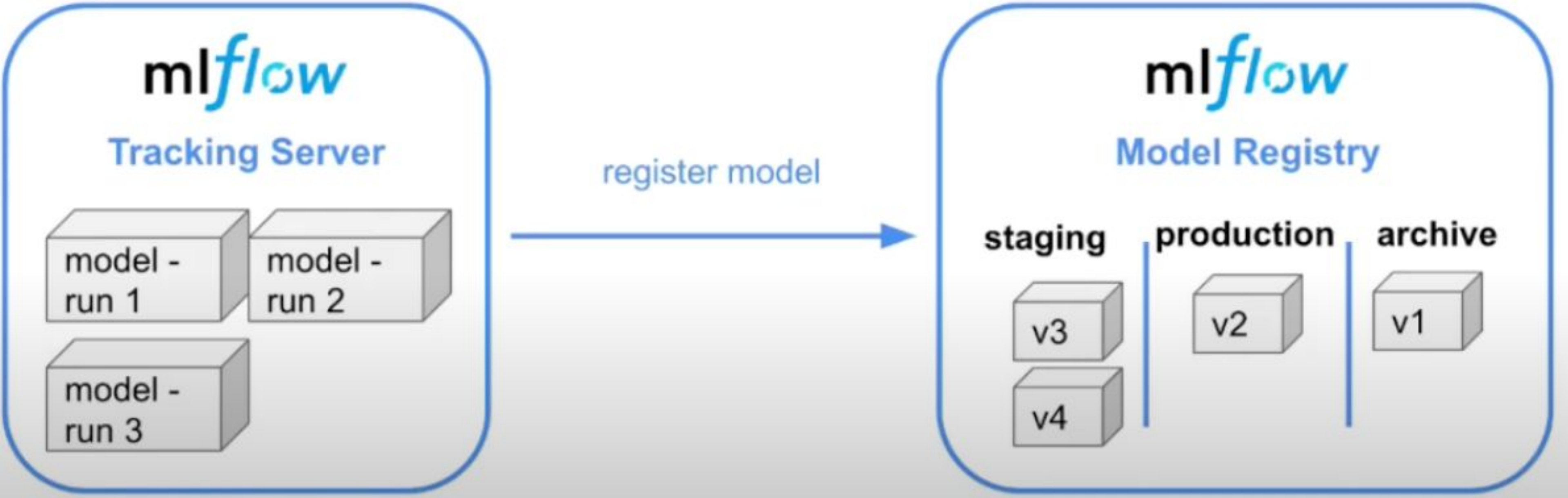
```
mlflow.<framework>.log_model(model, artifact_path="models/")
```

MLflow Model Format



2.5 Model Registry

Model Registry: version control



Registering the models

- Register models (in artifact tab)
- Add to the group
- Check inside the registered models:
  - Lineage (which version links to which run)
  - Add tags (name and model)
- Check model-registry.ipynb

`MlflowClient` Class

- A client of …
  - an MLflow Tracking Server that creates and manages experiments and runs.
  - an MLflow Registry Server that creates and manages registered models and model versions.
- To instantiate it we need to pass a tracking URI and/or a registry URI:

```
from mlflow.tracking import MlflowClient

client = MlflowClient(tracking_uri="sqlite:///mlflow.db")
```