

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

<input type="checkbox"/>	Run Name	Created	Dataset	Duration	Source	Models
<input type="checkbox"/>	classy-crane-616	13 minutes ago	-	9.3s	ipykern...	-
<input type="checkbox"/>	auspicious-conch-390	13 minutes ago	-	8.4s	ipykern...	-
<input type="checkbox"/>	resilient-kite-398	13 minutes ago	-	14.2s	ipykern...	-
<input type="checkbox"/>	dazzling-kite-667	14 minutes ago	-	12.0s	ipykern...	-
<input type="checkbox"/>	exultant-fly-70	14 minutes ago	-	7.2s	ipykern...	-
<input type="checkbox"/>	thoughtful-gull-194	14 minutes ago	-	9.4s	ipykern...	-
<input type="checkbox"/>	classy-lark-916	14 minutes ago	-	9.7s	ipykern...	-
<input type="checkbox"/>	sassy-bear-916	14 minutes ago	-	4.6s	ipykern...	-
<input type="checkbox"/>	unleashed-mouse-970	14 minutes ago	-	4.0s	ipykern...	-
<input type="checkbox"/>	trusting-bee-477	14 minutes ago	-	5.4s	ipykern...	-
<input type="checkbox"/>	industrious-mole-657	15 minutes ago	-	27.8s	ipykern...	-
<input type="checkbox"/>	whimsical-zebra-933	15 minutes ago	-	20.1s	ipykern...	-
<input type="checkbox"/>	colorful-pug-861	15 minutes ago	-	6.6s	ipykern...	-
<input type="checkbox"/>	classy-dove-417	15 minutes ago	-	4.8s	ipykern...	-
<input type="checkbox"/>	luxuriant-hare-839	16 minutes ago	-	16.9s	ipykern...	-
<input type="checkbox"/>	worried-pug-150	16 minutes ago	-	3.2s	ipykern...	-
<input type="checkbox"/>	invincible-kit-826	16 minutes ago	-	5.2s	ipykern...	-
<input type="checkbox"/>	whimsical-jay-226	16 minutes ago	-	7.7s	ipykern...	-
<input type="checkbox"/>	brawny-owl-983	16 minutes ago	-	9.1s	ipykern...	-

Show more columns (11 total)

50 matching runs



2.3 compare Xgboost tags

demo-study-group-w2-part1 >

Comparing 50 Runs from 1 Experiment

Visualizations

Parallel Coordinates Plot Scatter Plot Box Plot Contour Plot

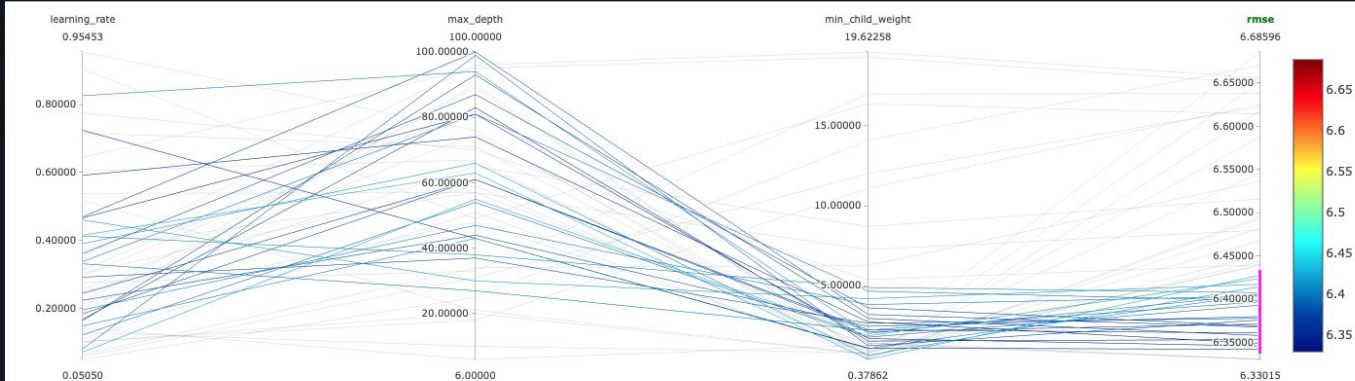
Parameters:

learning_rate X max_depth X min_child_weight X

Metrics:

rmse X

Clear All





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:

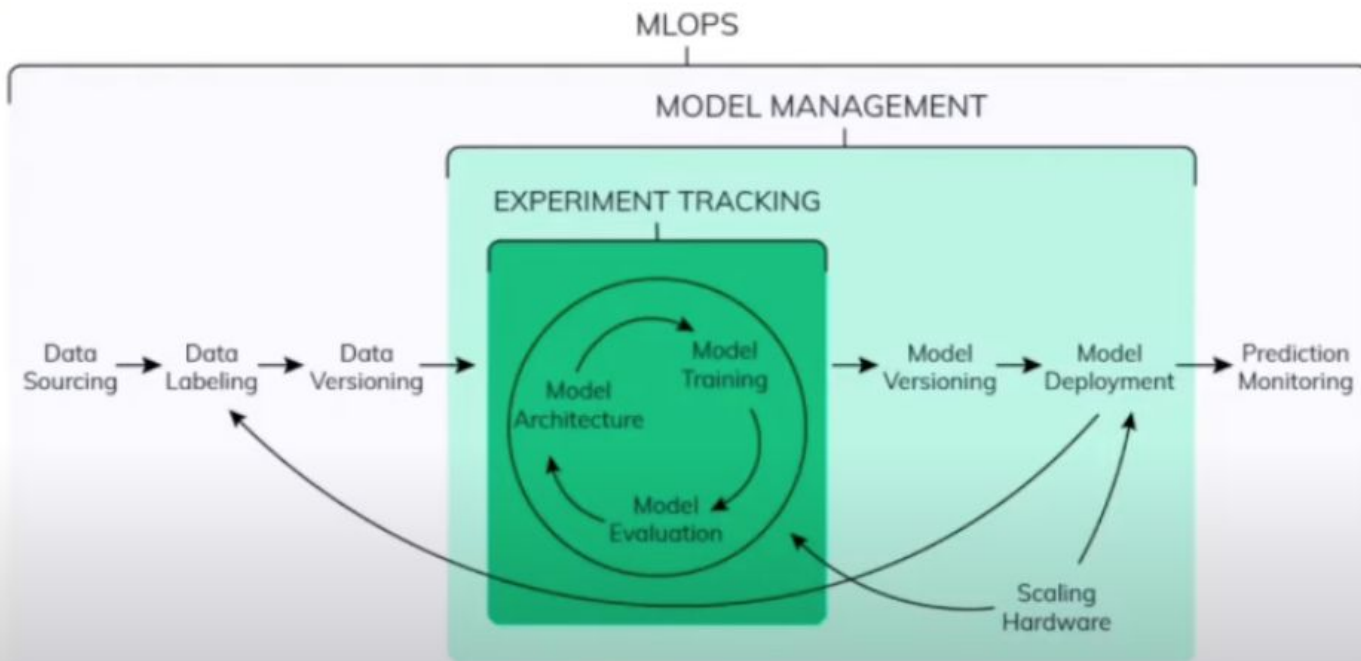
1. Call `mlflow.autolog()` before your training code. This will enable autologging for each supported library you have installed as soon as you import it.
2. Use library-specific autolog calls for each library you use in your code. See below for examples.

The following libraries support autologging:

- [Scikit-learn](#)
- TensorFlow and Keras
- Gluon
- XGBoost
- LightGBM
- Statsmodels
- Spark
- Fastai
- Pytorch

2.4 Model Management

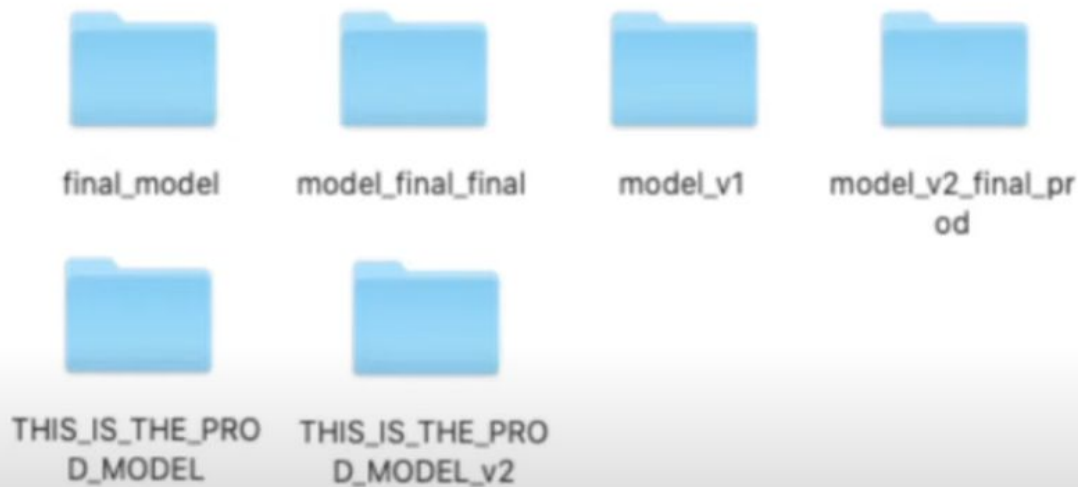
ML Lifecycle



Model management

What's wrong with this?

- Error prone
- No versioning
- No model lineage



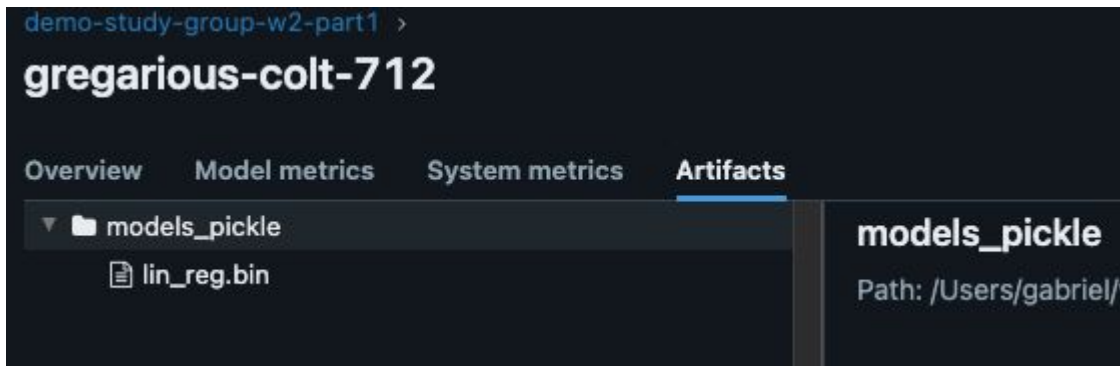


Log artifacts

```
# 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

▼ Artifacts

- ▶ model
- ▼ models_mlflow
 - MLmodel
 - conda.yaml
 - model.xgb
 - requirements.txt
- feature_importance_weight.js
- feature_importance_weight.p

Full Path: ./mlruns/1/6c961e3b8387446fb183c560e1c43e5f/artifacts/mo... 

Size: 421B

```
artifact_path: models_mlflow
flavors:
  python_function:
    data: model.xgb
    env: conda.yaml
    loader_module: mlflow.xgboost
    python_version: 3.9.12
  xgboost:
    code: null
    data: model.xgb
    model_class: xgboost.core.Booster
    xgb_version: 1.6.0
mlflow_version: 1.25.1
model_uuid: e0c25f86e7d340fcb778c6871608c3e3
run_id: 6c961e3b8387446fb183c560e1c43e5f
utc_time_created: '2022-05-15 16:53:38.986314'
```

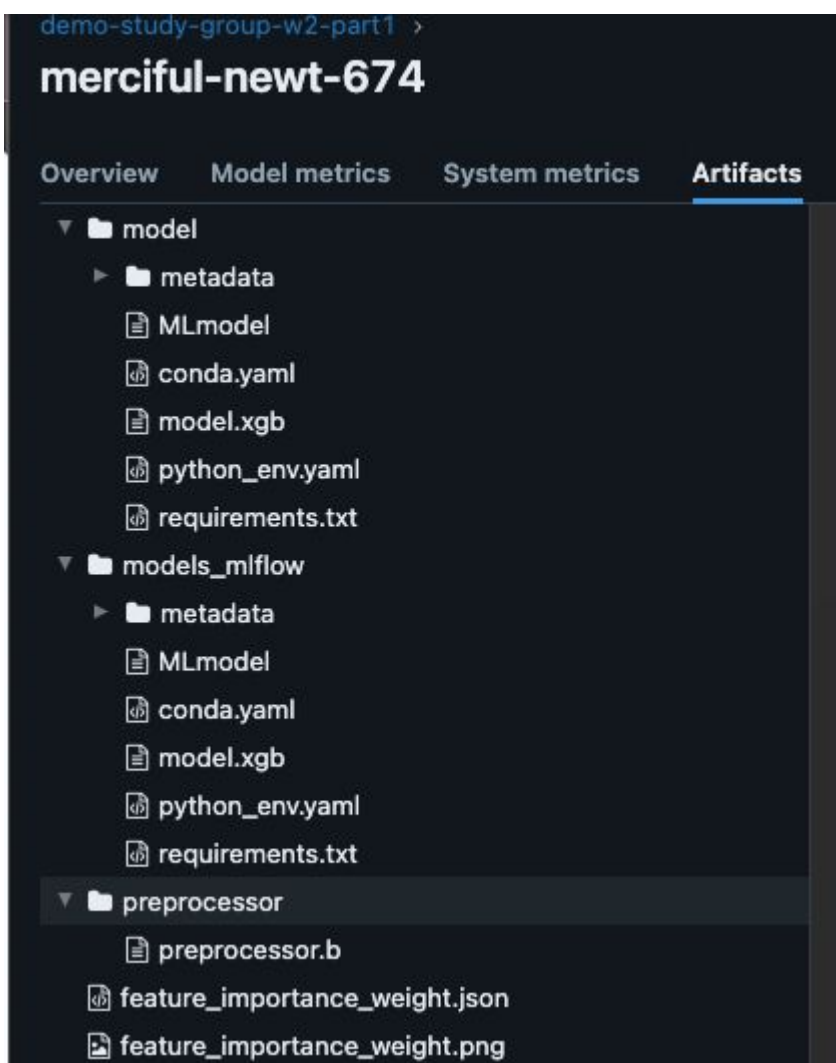


Logged model, artifacts

Model: auto logging

Models_mlflow: manual logging

Preprocessor: artifacts for dictionary vectorizer





Load model and make predictions

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
logged_model = 'runs:/b19e4d26363e4ce4a1c671b7ecf83e9b/models_mlflow'

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

# Predict on a Spark DataFrame.
columns = list(df.columns)
df.withColumn('predictions', loaded_model(*columns)).collect()
```

Predict on a Pandas DataFrame:

```
import mlflow
logged_model = 'runs:/b19e4d26363e4ce4a1c671b7ecf83e9b/models_mlflow'

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



Recap of logging models in MLflow

Two options:

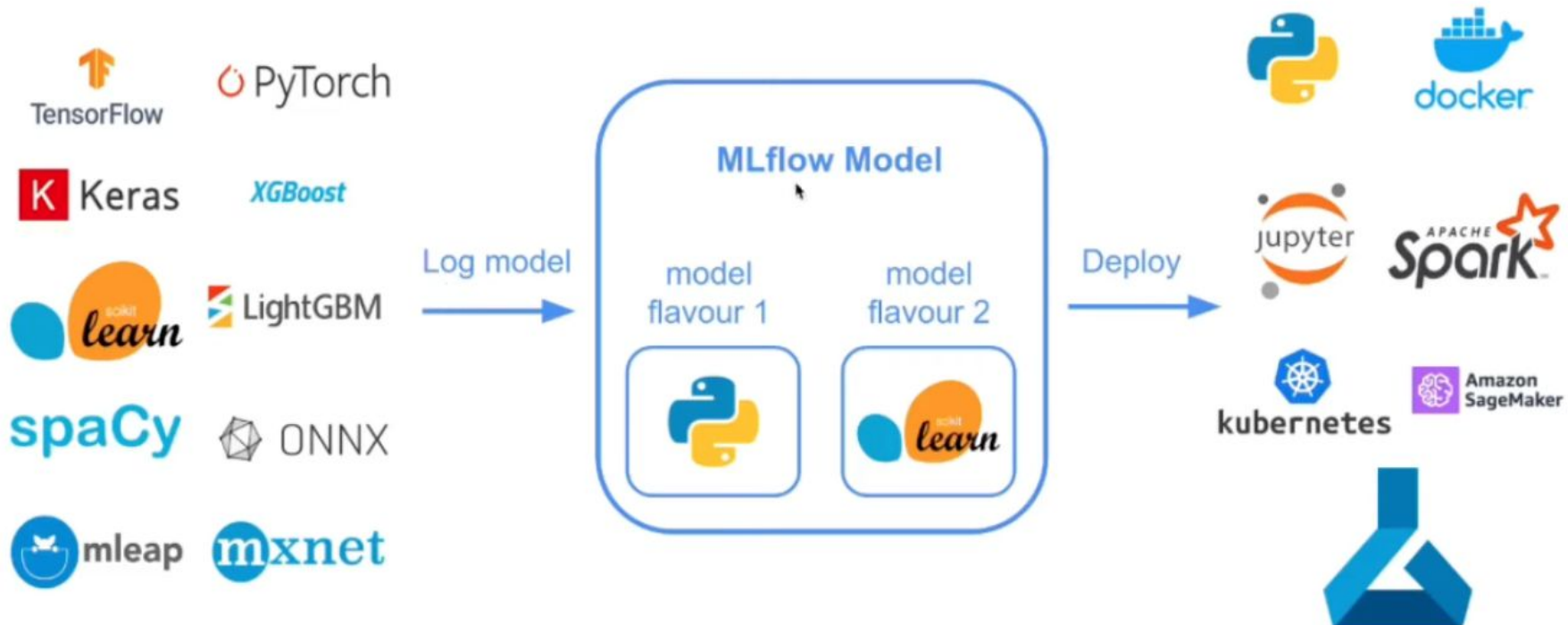
- Log model as an artifact

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

- Log model using the method `log_model`

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

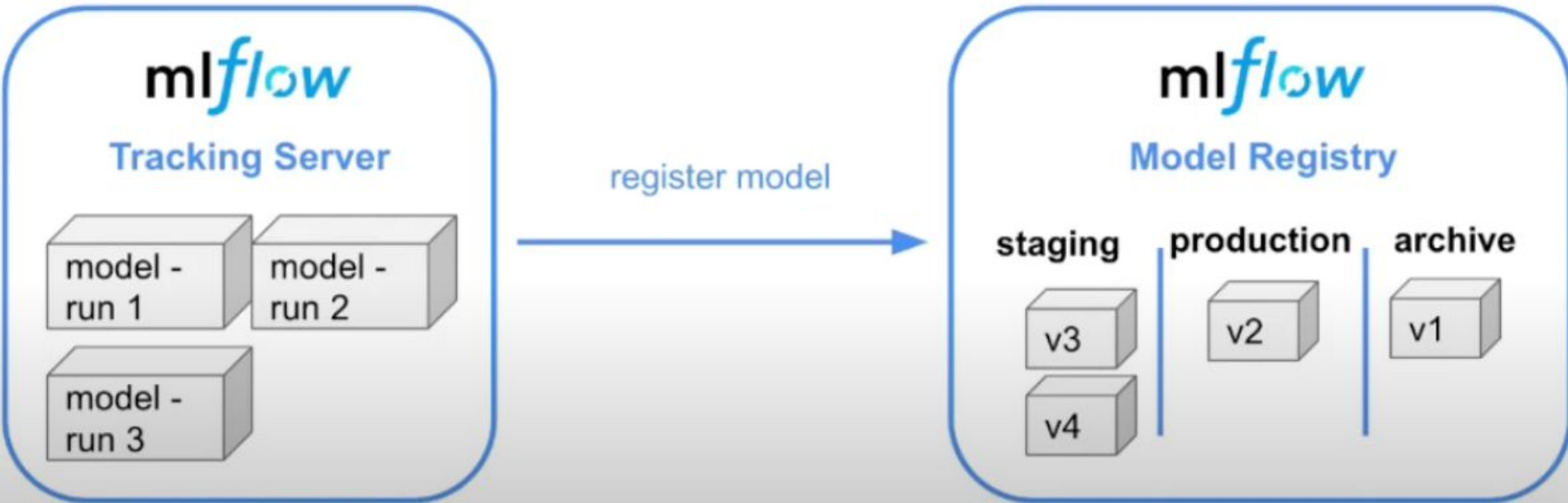
MLflow Model Format





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")
```