# Getting Started with MLflow

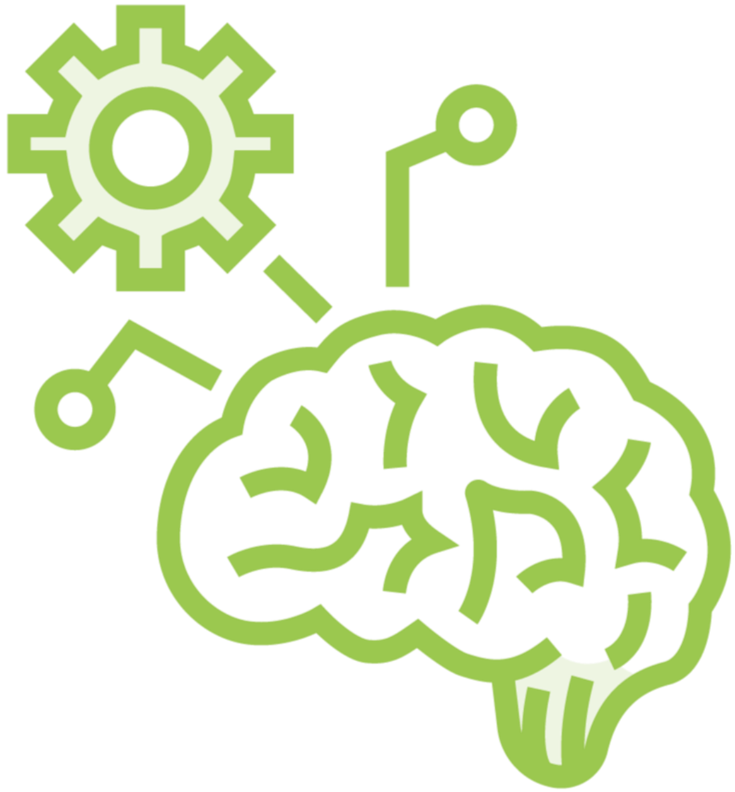
## Understanding MLflow

**Paweł Kordek**

Software Engineer – Data and Machine Learning

@pawel_kordek

# Why MLflow?



**Machine learning is everywhere**

**Growth of ML has created new challenges**

**Need for new tools gave rise to MLOps**

**MLflow solves certain challenges that MLOps aims to address**

# Basics

- Instructor (me)
- Target audience (you)
- Prerequisites
- Demo story description
- MLflow and problems it solves

# Me

**Passionate about data and machine learning**

**Fraud detection**

**Experienced first-hand the challenges I talk about**

# You

- Data scientist
- Machine learning engineer
- Hobbyist
- Engineering manager
- Infrastructure engineer
- Anything related to ML

# Prerequisites

**Comfortable with Python**

**Minimal practical experience with building ML models**

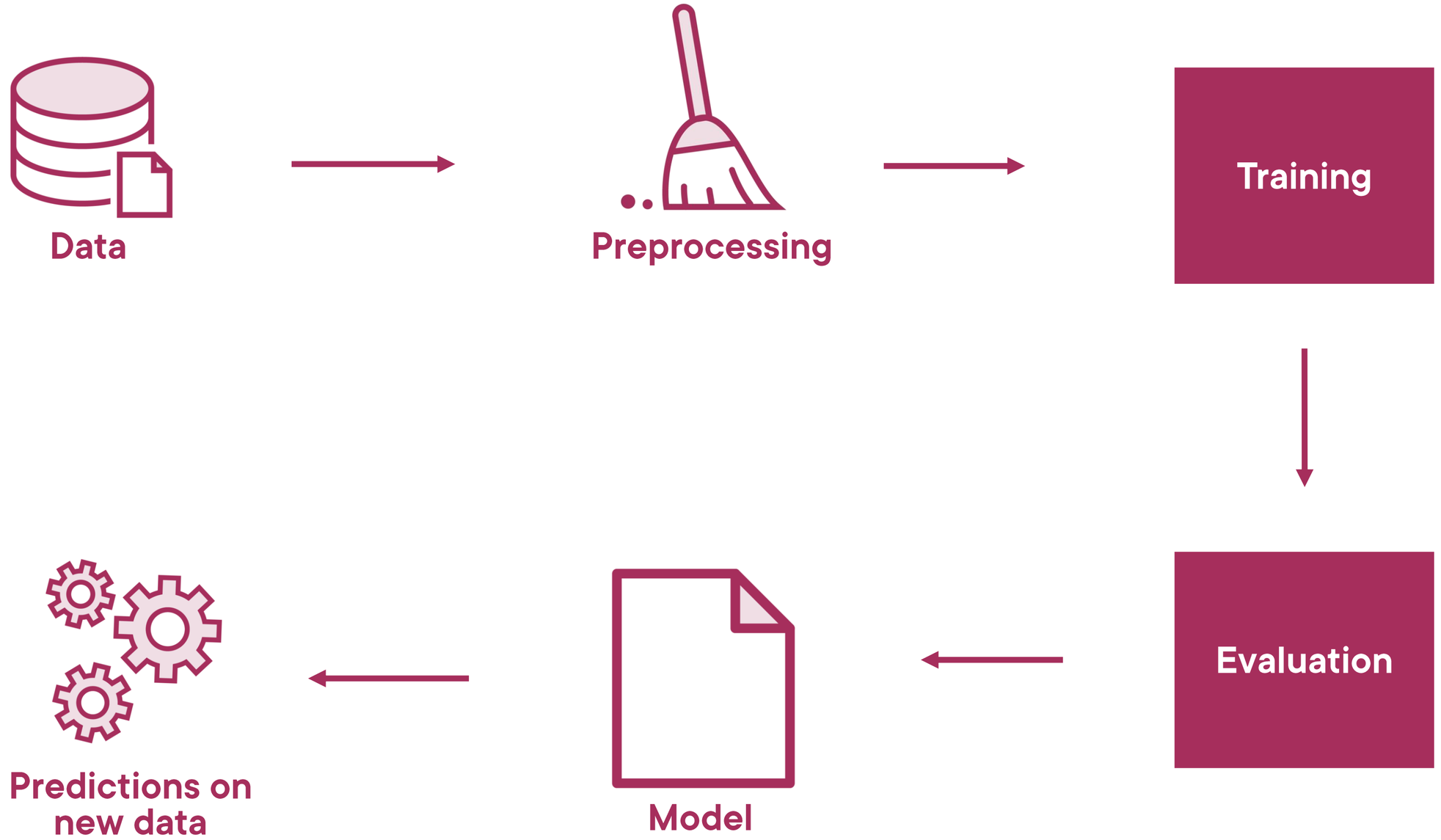
# Story

**Real estate company**

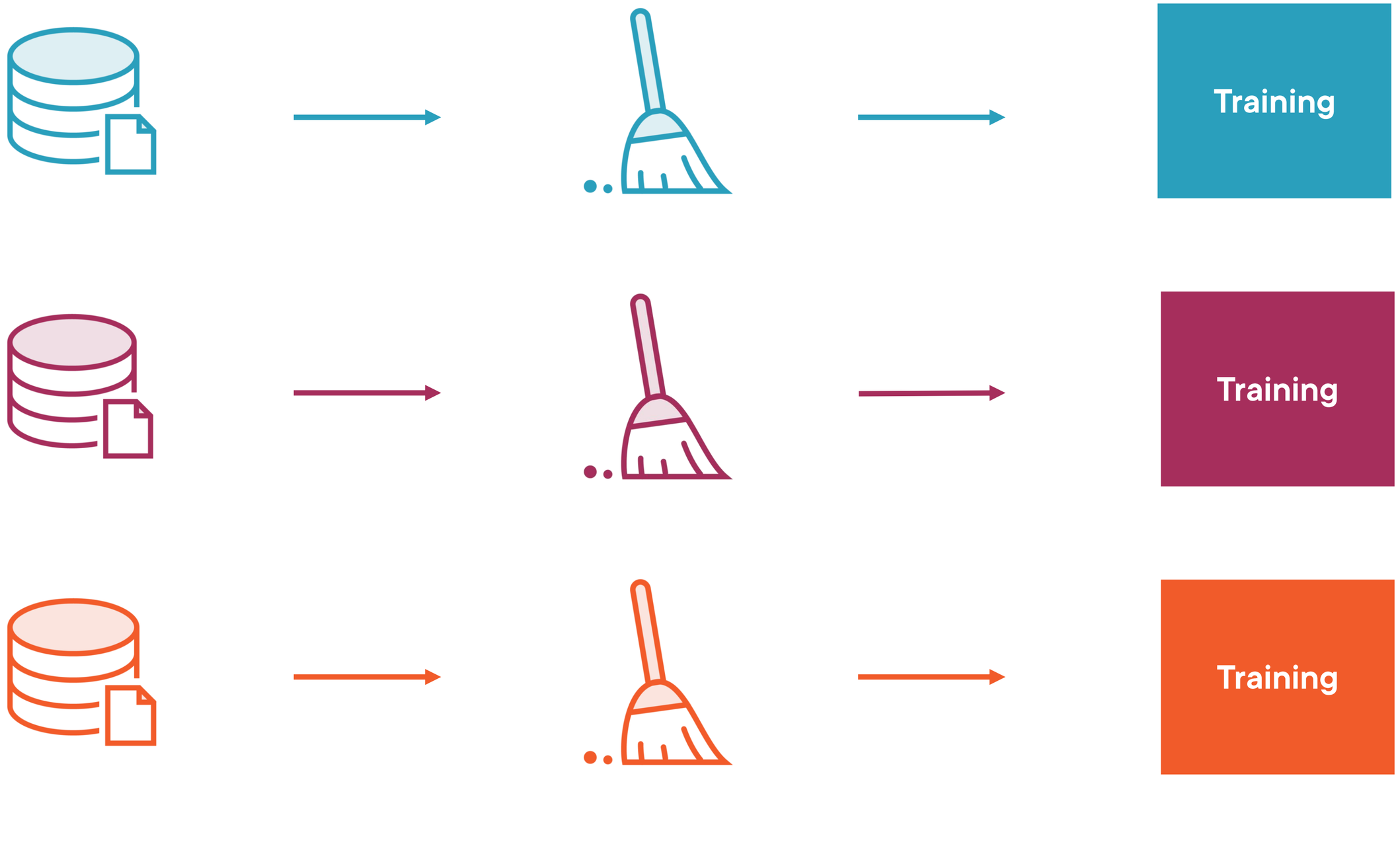
**One of the first data scientists**

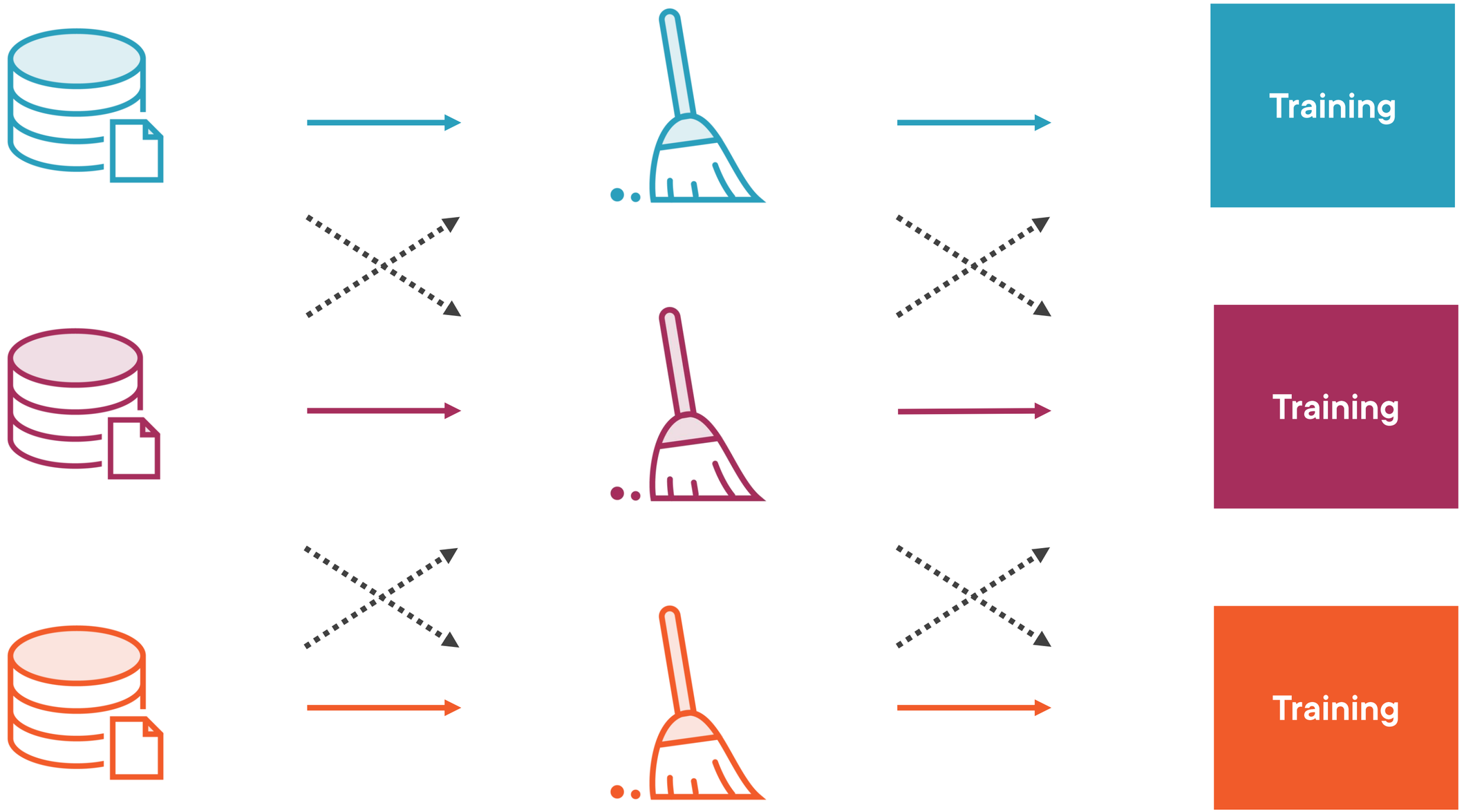
**Prediction of house prices**

# What Is MLflow?


Data
Preprocessing
Training
Evaluation
Model
Predictions on new data


Training


Training
Training
Training


Training
Training
Training

# Tracking the Changes

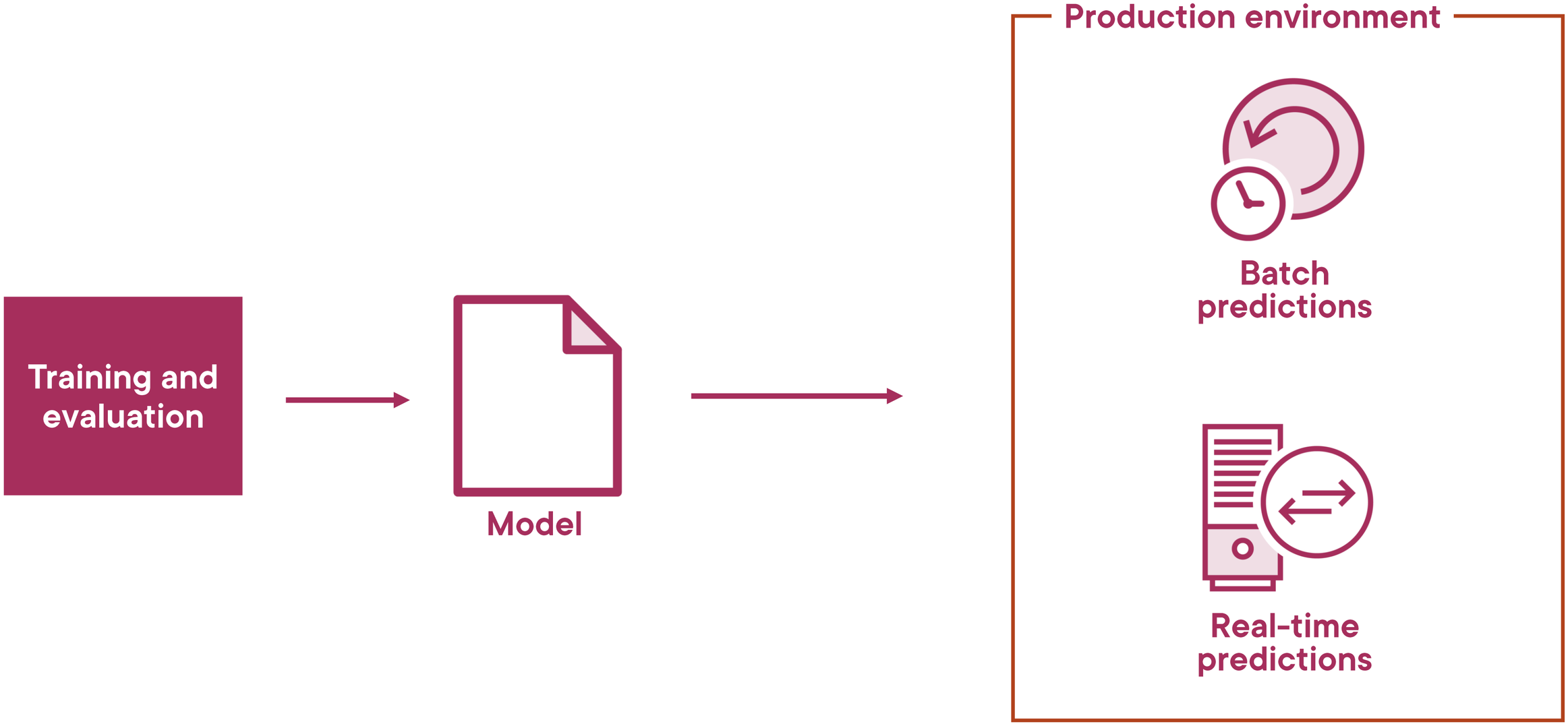
**Reproducibility**

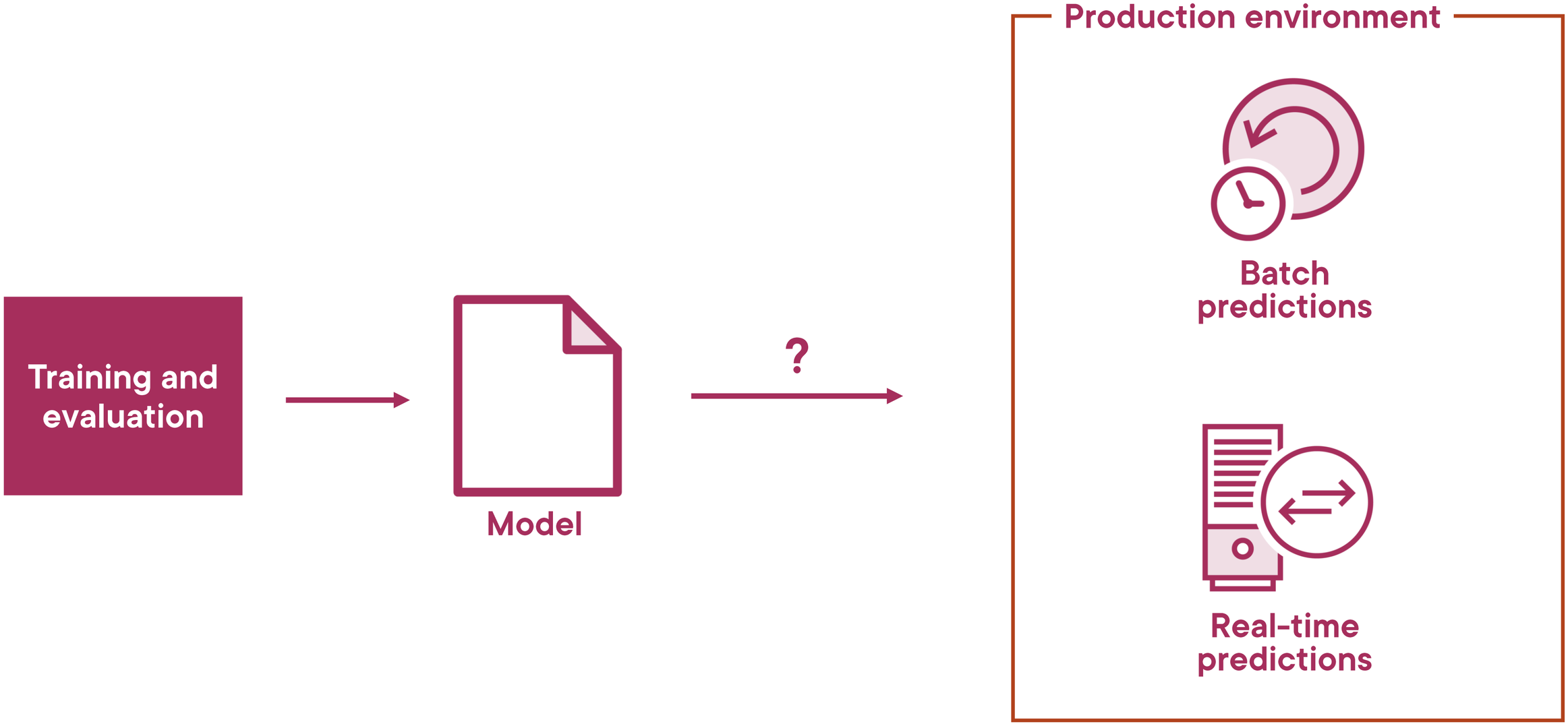
**Traceability**

**Notes and spreadsheets are not scalable**

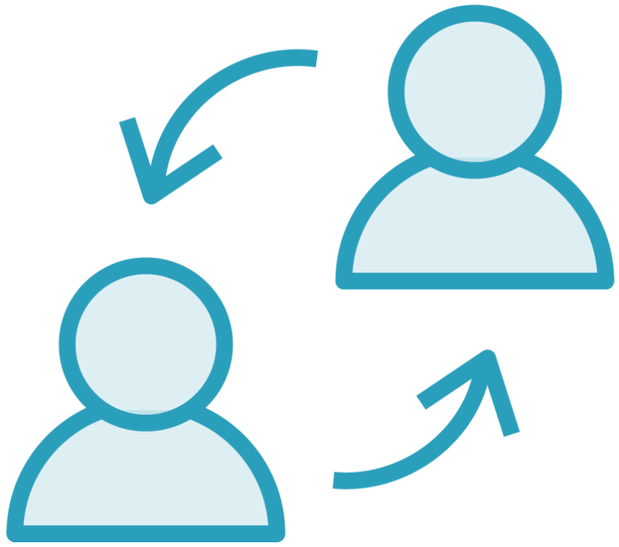
**MLflow Tracking**

- Stores data for all experiments
- User interface
- Programmatic querying

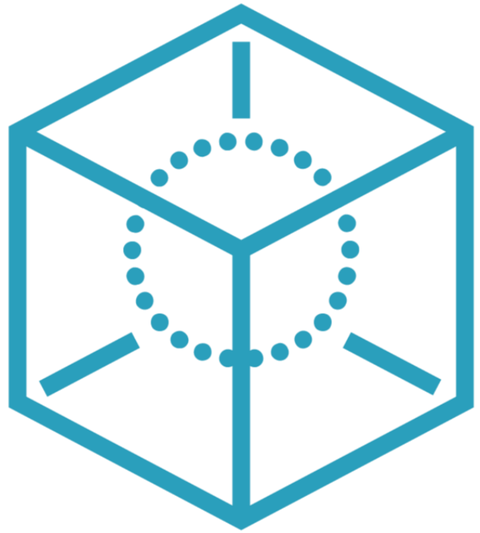

Training and evaluation
Model
Production environment
Batch predictions
Real-time predictions


Training and evaluation
Model
?
Production environment
Batch predictions
Real-time predictions

# Model Handover



**Extensive explanations required**



**Additional layers of code**

# MLflow Models

**Format for packing models**

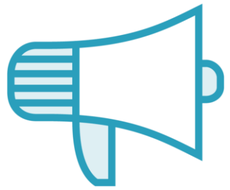
**Running predictions against existing dataset**

**Real-time serving**

# Managing Models

**Model discovery**

**Model metadata management**



**Traditionally requires explicit communication**

**Human error likely to happen**

# MLflow Model Registry

**Centralized model catalogue**

**Models can be used/deployed directly from the registry**

**Can be used as a base for building automated tools**

# MLflow Projects

**Another MLflow component**

**Opinionated – often not compatible with existing choice of tools inside ML teams**

**Won't be explored in this course**

# MLflow

**Tracking**

**Models**

**Model Registry**

# Summary

**Ubiquity of machine learning**

**Target audience and prerequisites**

**MLflow components**

# Tracking ML Experiments

**Paweł Kordek**

Software Engineer – Data and Machine Learning

@pawel_kordek

# Overview

**Dataset**

**Model training pipeline**

**Vocabulary**

**MLflow Tracking: Demo**

# Dataset

# Task

**Predict market price of a property given some of its characteristics**

# Ames Housing Dataset

**De Cock D. 2011. Ames, Iowa: Alternative to the Boston Housing Data as an End of Semester Regression Project. Journal of Statistics Education; 19(3).**

**Complete dataset documentation:**
**http://jse.amstat.org/v19n3/decock/DataDocumentation.txt**

# Ames Housing Dataset

**2930 rows and 83 columns**

**A lot of features!**

**Selected features:**

- 'Lot Area' – size of the lot in feet
- 'Gr Liv Area' – living area above the ground level
- 'Garage Area' – garage size in square feet
- 'Bldg Type' – building type, e.g. '1Fam'

**Target: 'SalePrice' – in USD**

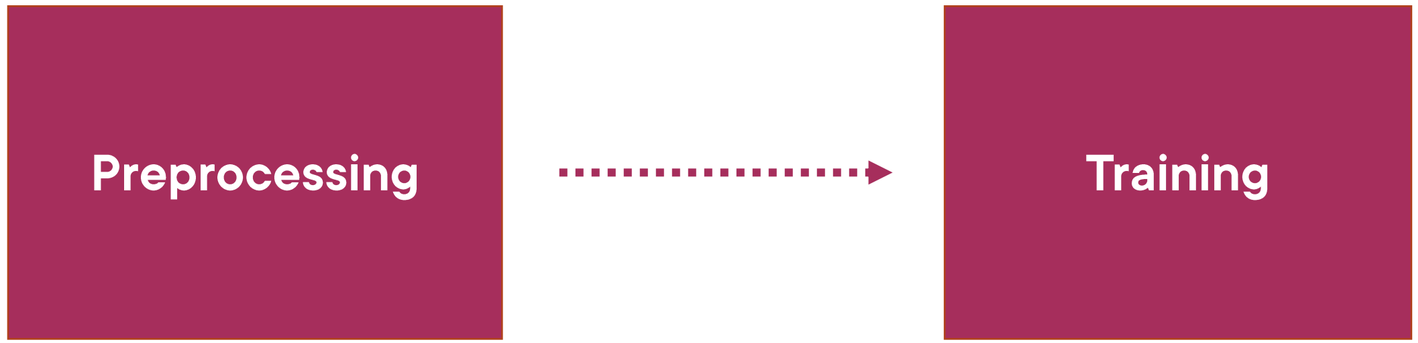
| Lot Area | Gr Living Area | Garage Area | Bldg Type | SalePrice |
|---|---|---|---|---|
| 31770 | 1656 | 528.0 | 1Fam | 215000 |
| 11622 | 896 | 730.0 | 1Fam | 105000 |
| 14267 | 1329 | 312.0 | 1Fam | 172000 |

# One-hot Encoding

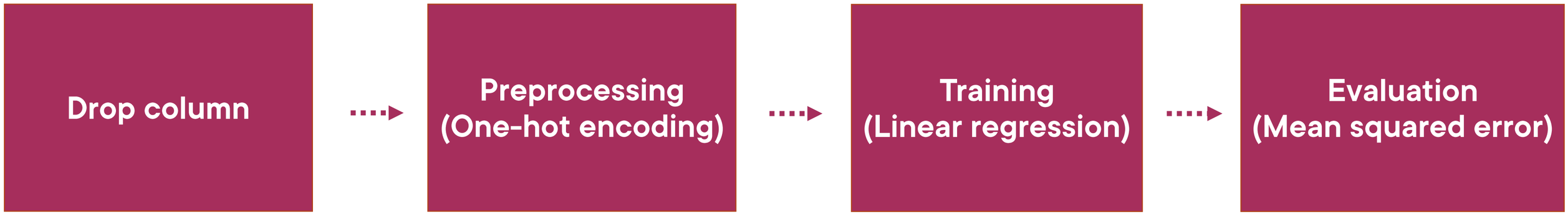
| | BldgType |
|---|---|
| 0 | 1Fam |
| 1 | Duplx |
| 2 | Twnhsl |
| 3 | Duplx |

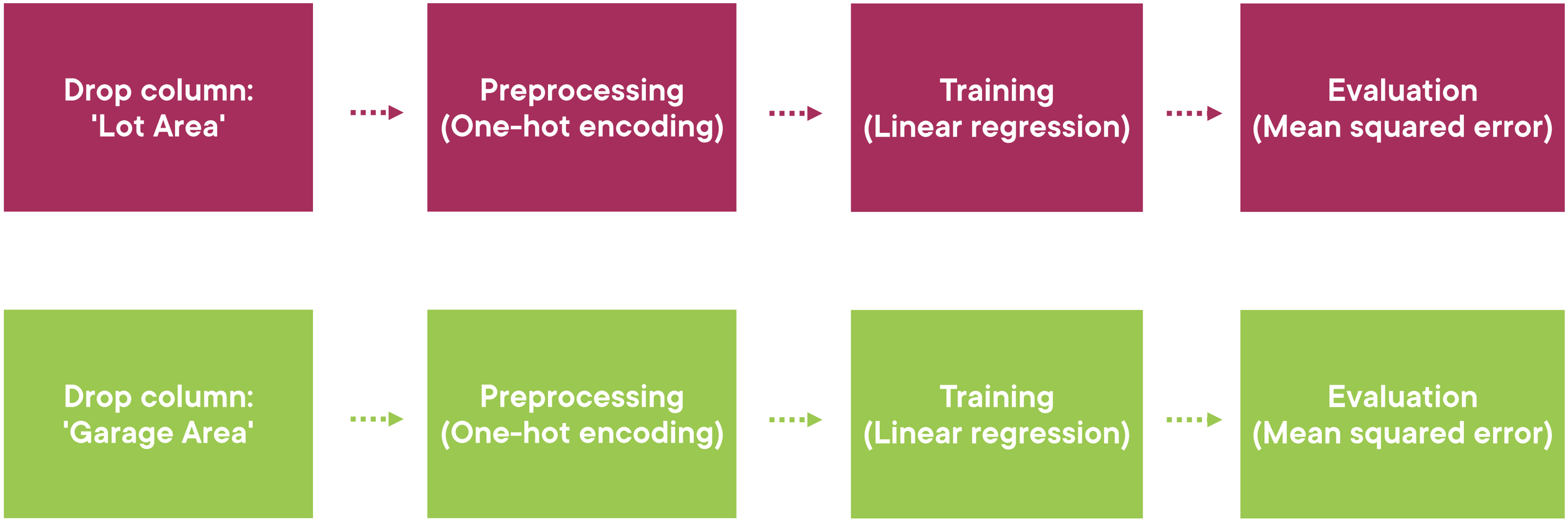
| | 1Fam | Duplx | Twnhsl |
|---|---|---|---|
| 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 2 | 0 | 0 | 1 |
| 3 | 0 | 1 | 0 |

# Experimenting with ML Models


Preprocessing
Training


Preprocessing
(One-hot encoding)
Training
(Linear regression)
Evaluation
(Mean squared error)


Drop column
Preprocessing
(One-hot encoding)
Training
(Linear regression)
Evaluation
(Mean squared error)




# Repeat for all feature columns

Initial training – error: 123.65

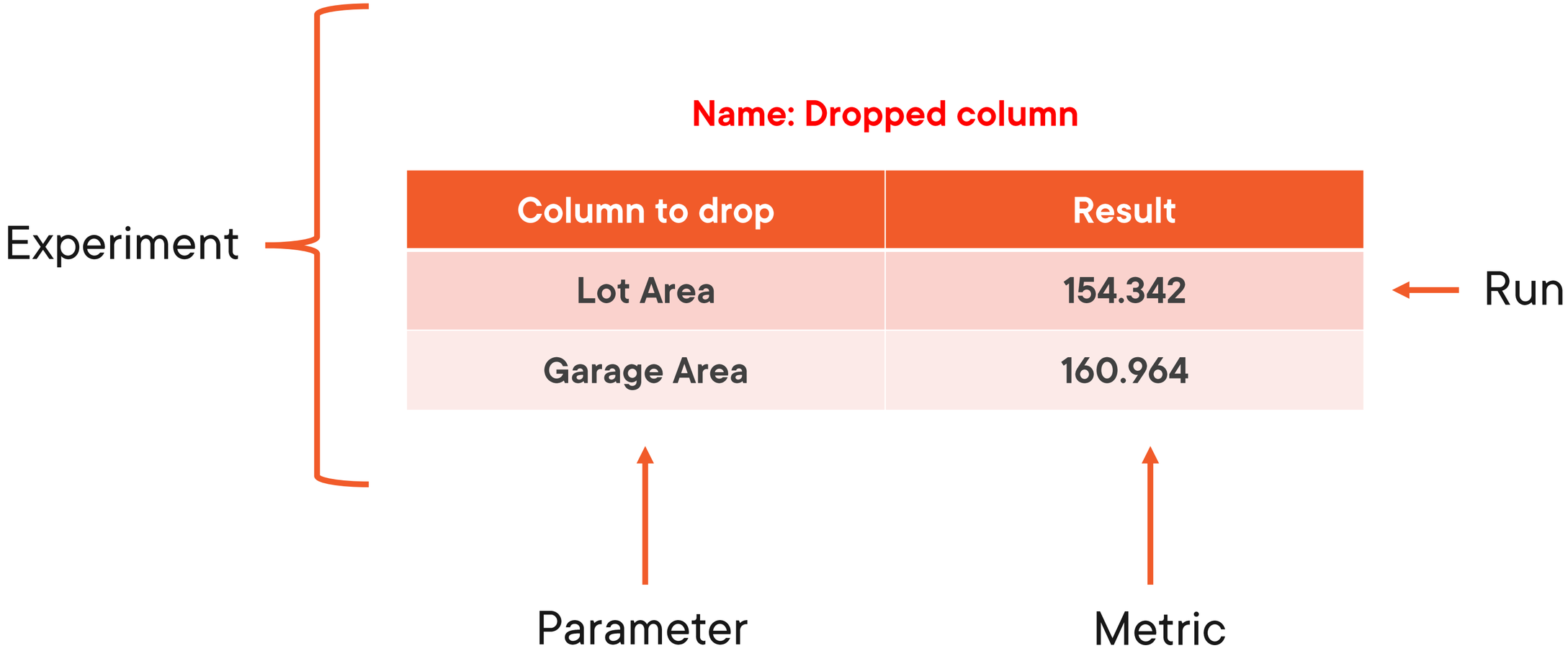
Dropped column: Lot Area – error: 154.34

Dropped column: Garage Area – error: 160.96

| Name | Column to drop | Result |
|---|---|---|
| Initial training | - | 123.654 |
| Dropped column | Lot area | 154.342 |
| Dropped column | Garage area | 160.964 |

| Name | Result |
| --- | --- |
| Initial training | 123.654 |

| Name | Column to drop | Result |
| --- | --- | --- |
| Dropped column | Lot Area | 154.342 |
| Dropped column | Garage Area | 160.964 |


Experiment
Name: Dropped column
Column to drop
Result
Lot Area
154.342
Garage Area
160.964
Run
Parameter
Metric

# Demo

**Simple ML model for house price predictions**

**Creating MLflow experiments and runs**

**Logging parameters and metrics**

**Using the MLflow UI**

# Summary

**Dataset**

**Training process**

**Concepts**

- Experiment
- Run
- Parameter
- Metric

**MLflow Python library**

**MLflow UI**

# Exporting Artifacts

**Paweł Kordek**

Software Engineer – Data and Machine Learning

@pawel_kordek

# Overview

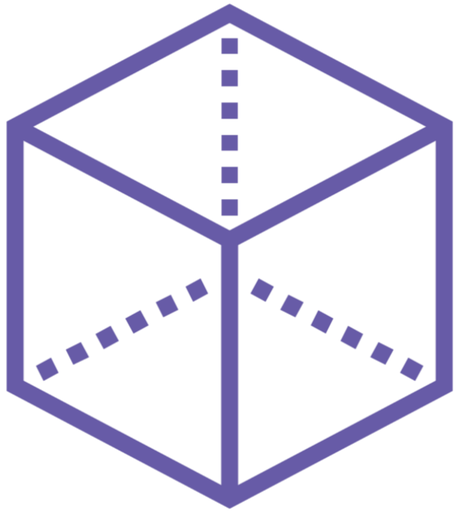
**Examples of artifacts**

**How to save an artifact**

**Demo**

# Some Examples

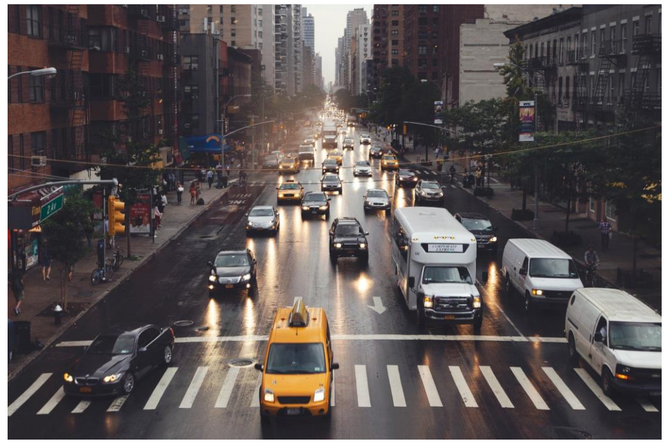
**Plots**



**Models**

**Datasets**

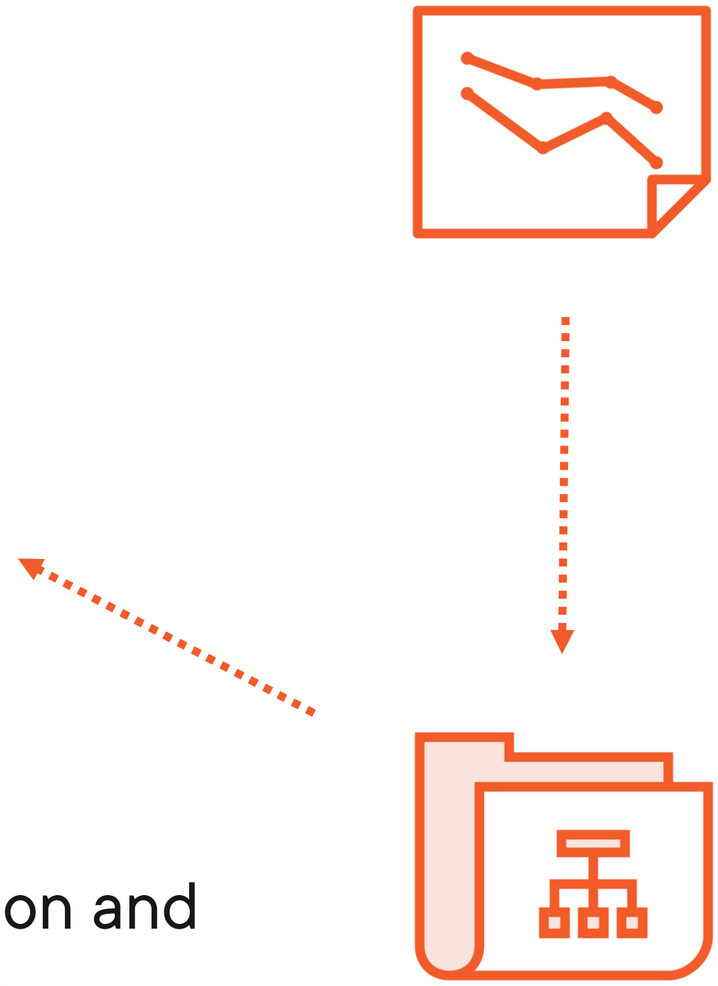
# Verifying Correctness of a Transformation



Aside from parameters and metrics, we need to store files.

# Without MLflow

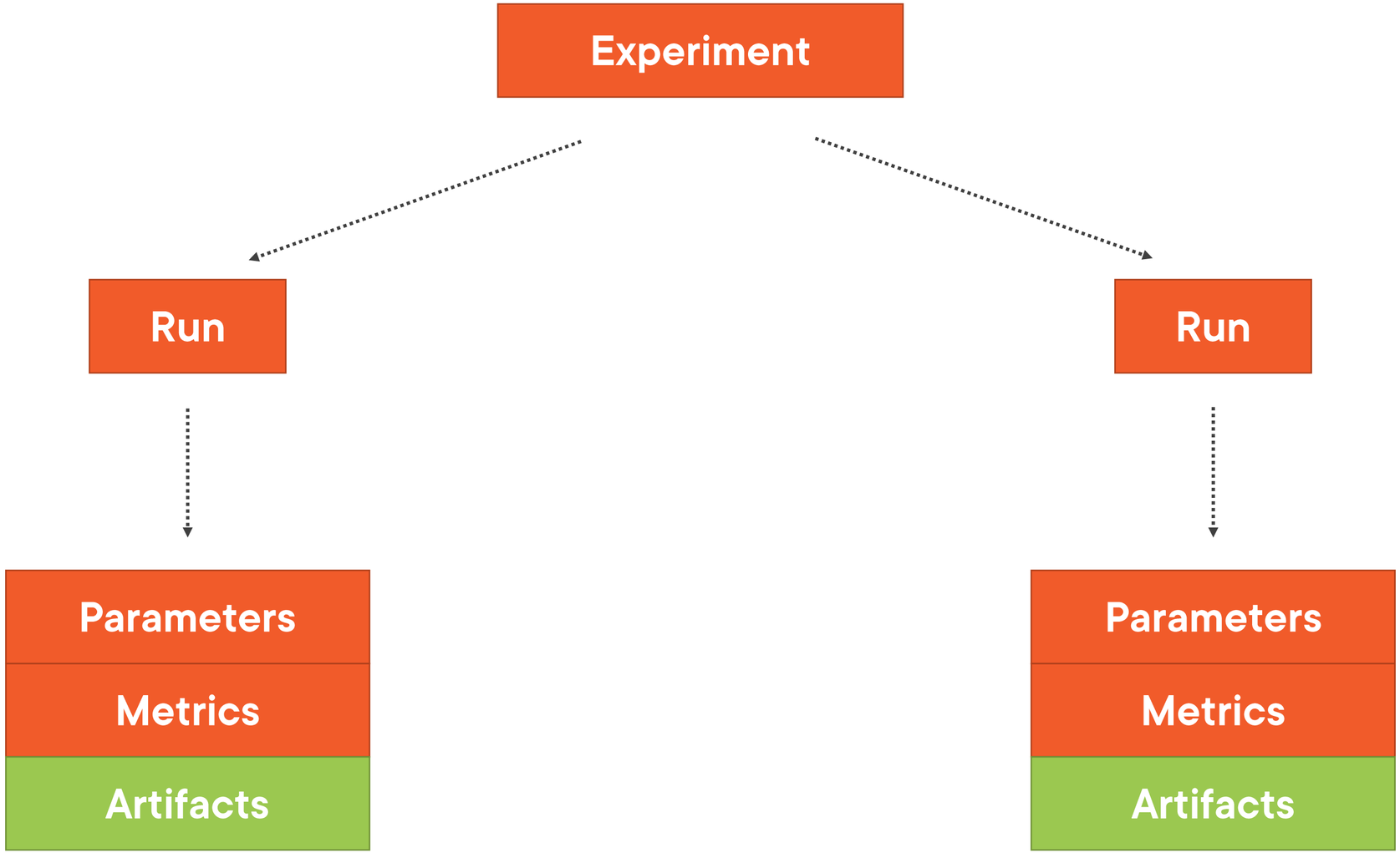
Everyone needs to agree on and adhere to the convention.



Need to manually maintain a directory structure that can be matched to experiments and runs.

# With MLflow

# With MLflow


Experiment
Run
Run
Parameters
Metrics
Artifacts
Parameters
Metrics
Artifacts

# Exporting an Artifact

**Store the file in a temporary location on disk**

**Log (export) it using MLflow API**

# Exporting an Artifact

```
import mlflow
import pickle


model = ... #  e.g. LinearRegression
serialized_model = pickle.dumps(model)

with open("some_temporary_folder/model.pkl", "wb") as m:
    m.write(serialized_model)

mlflow.log_artifact("some_temporary_folder/model.pkl")
```

# Exporting an Artifact

```
import mlflow
import pickle


model = ... #  e.g. LinearRegression
serialized_model = pickle.dumps(model)

with open("some_temporary_folder/model.pkl", "wb") as m:
    m.write(serialized_model)

mlflow.log_artifacts("some_temporary_folder")
```

# Demo

**Logging the model**

**Adding more artifacts**

**Browsing files using the UI**

# Summary

The need for artifacts in MLflow

Logging (exporting) artifacts with MLflow

Using the UI to browse the artifacts

# Using MLflow in a Collaborative Scenario

**Paweł Kordek**

Software Engineer – Data and Machine Learning

@pawel_kordek

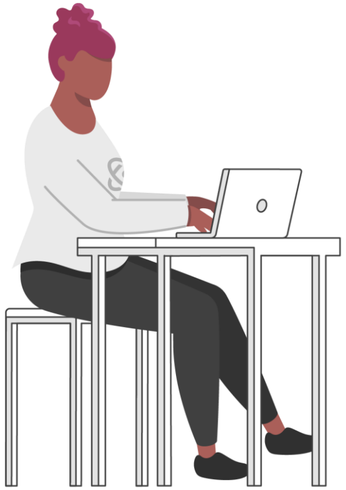
# Overview

**The need for a collaborative ML environment**

**Closer look at the current setup**

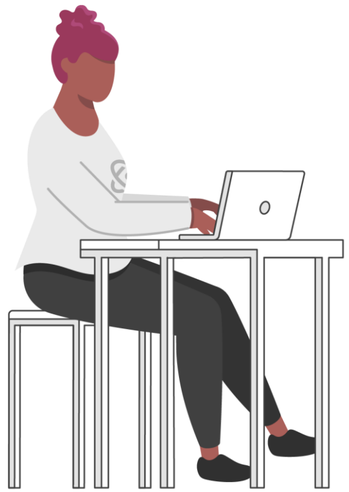
**Individual components**

**Complete MLflow deployment**



.../mlruns

.../mlruns



Network drive

custom/path

another/custom/path

# Changing MLflow Storage Location

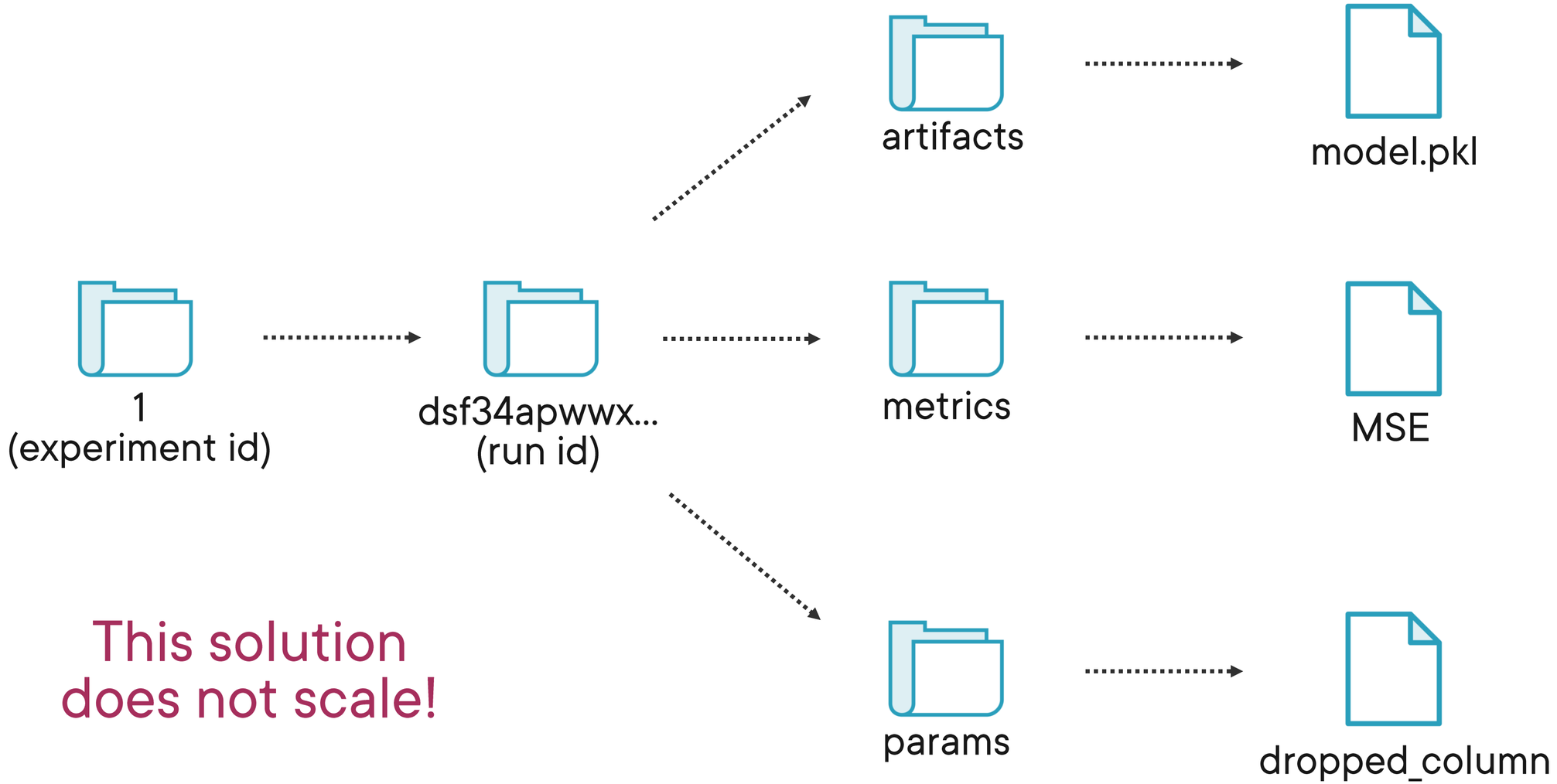
```
mlflow.set_tracking_uri(
    "file:///custom/location"
)
```

in the Python code

```
mlflow ui
--backend-store-uri file:///custom/location
```

starting the UI

# Inside 'mlruns'

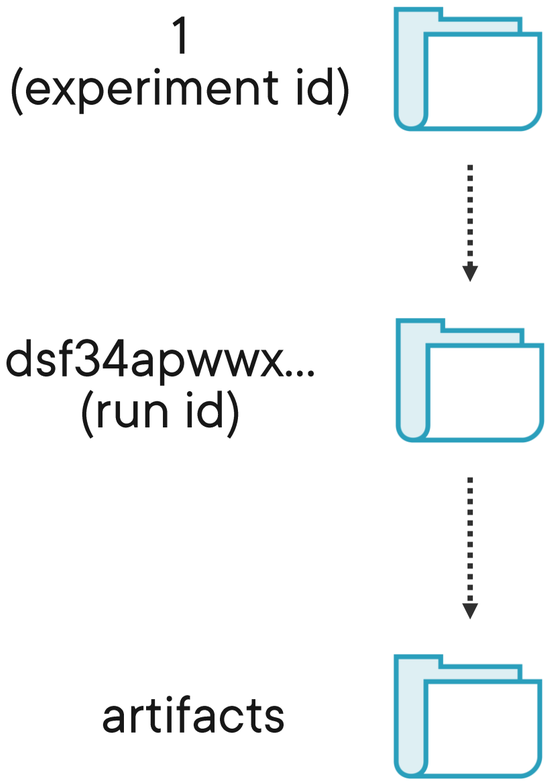



This solution
does not scale!

# Using SQLite

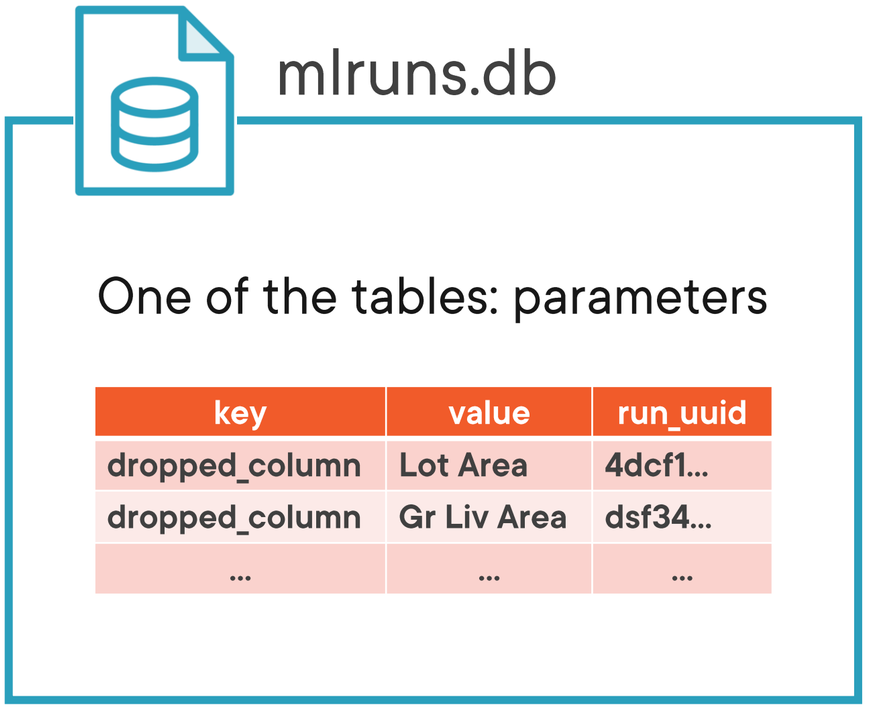
**Artifacts:**
**Same as before**

**Everything else:**
**Single SQLite file**




Artifact store

Backend store




| key | value | run_uuid |
|---|---|---|
| dropped_column | Lot Area | 4dcf1... |
| dropped_column | Gr Liv Area | dsf34... |
| ... | ... | ... |

# Using SQLite Backend Store

```
mlflow.set_tracking_uri(
    "sqlite:///custom/location/mlruns.db"
)
```

in the Python code

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

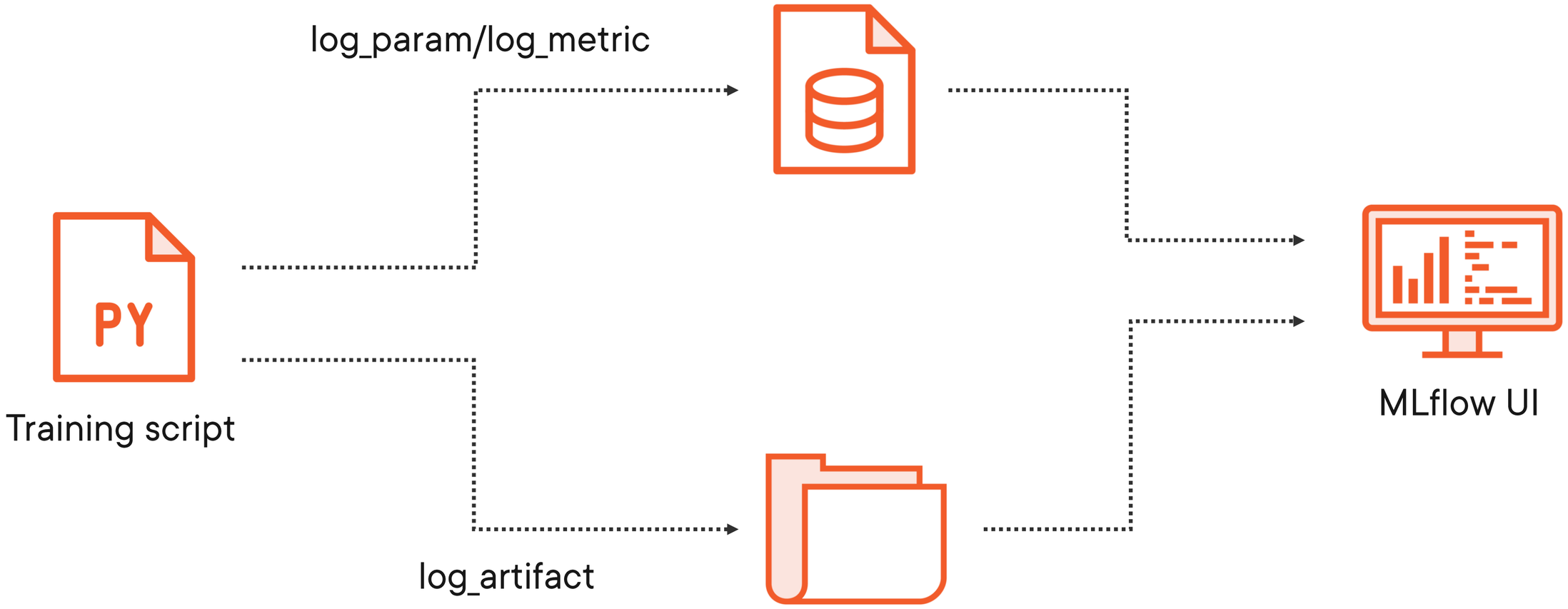
starting the UI

**Artifact store cannot be customized in this setup.**

# Up Next: MLflow Server

# MLflow Server

# Without MLflow Server



# Using MLflow Server

```
mlflow server --host 0.0.0.0
    --backend-store-uri
    sqlite:///data/mlflow/mlruns.db
    --default-artifact-root
    file:///data/mlflow/artifacts
```

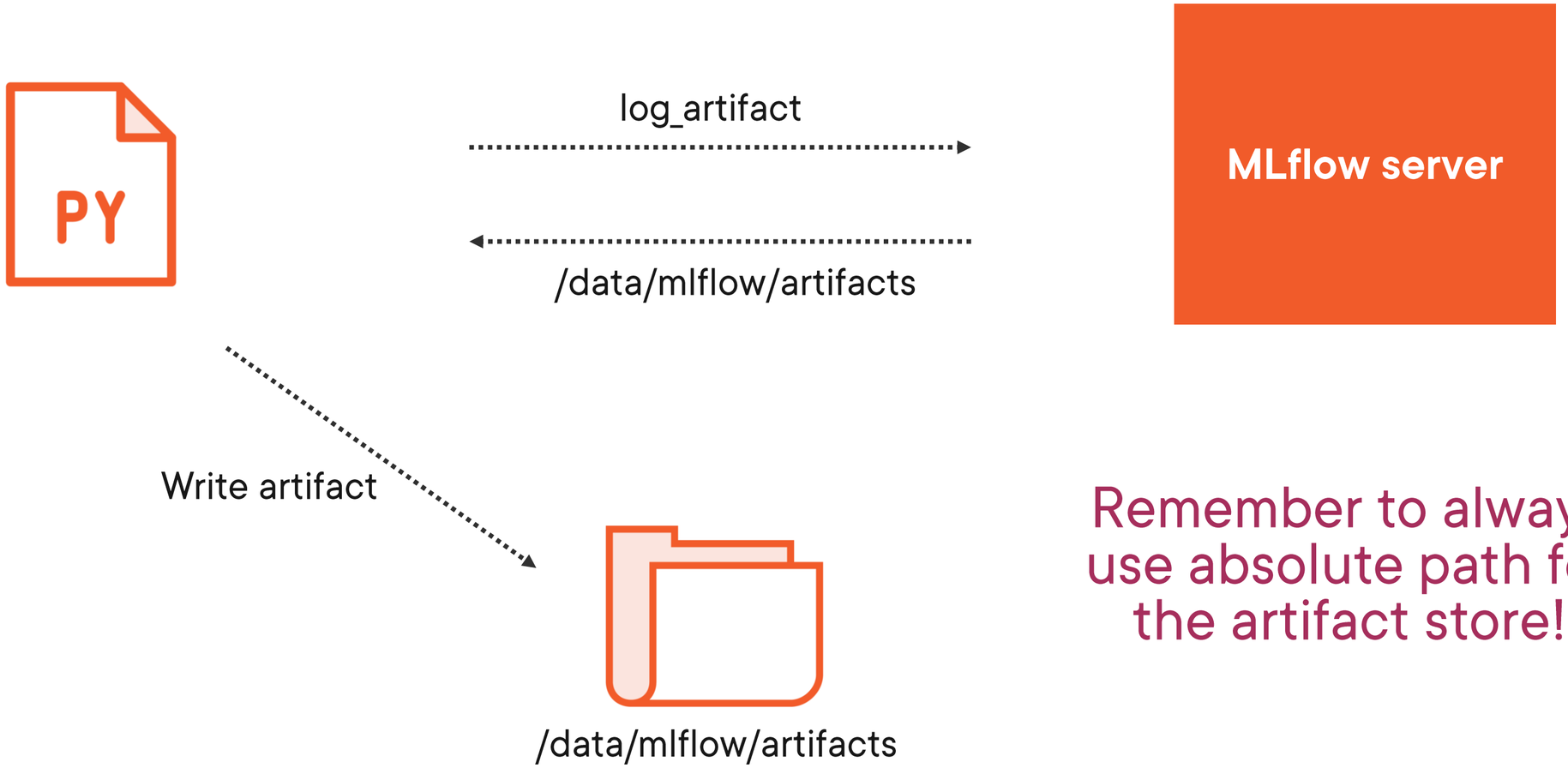
starting the server

```
mlflow.set_tracking_uri("localhost:5000")
```

connecting to the server in the Python script

# Backend Store with MLflow Server

# Artifact Store with MLflow Server




Remember to always use absolute path for the artifact store!

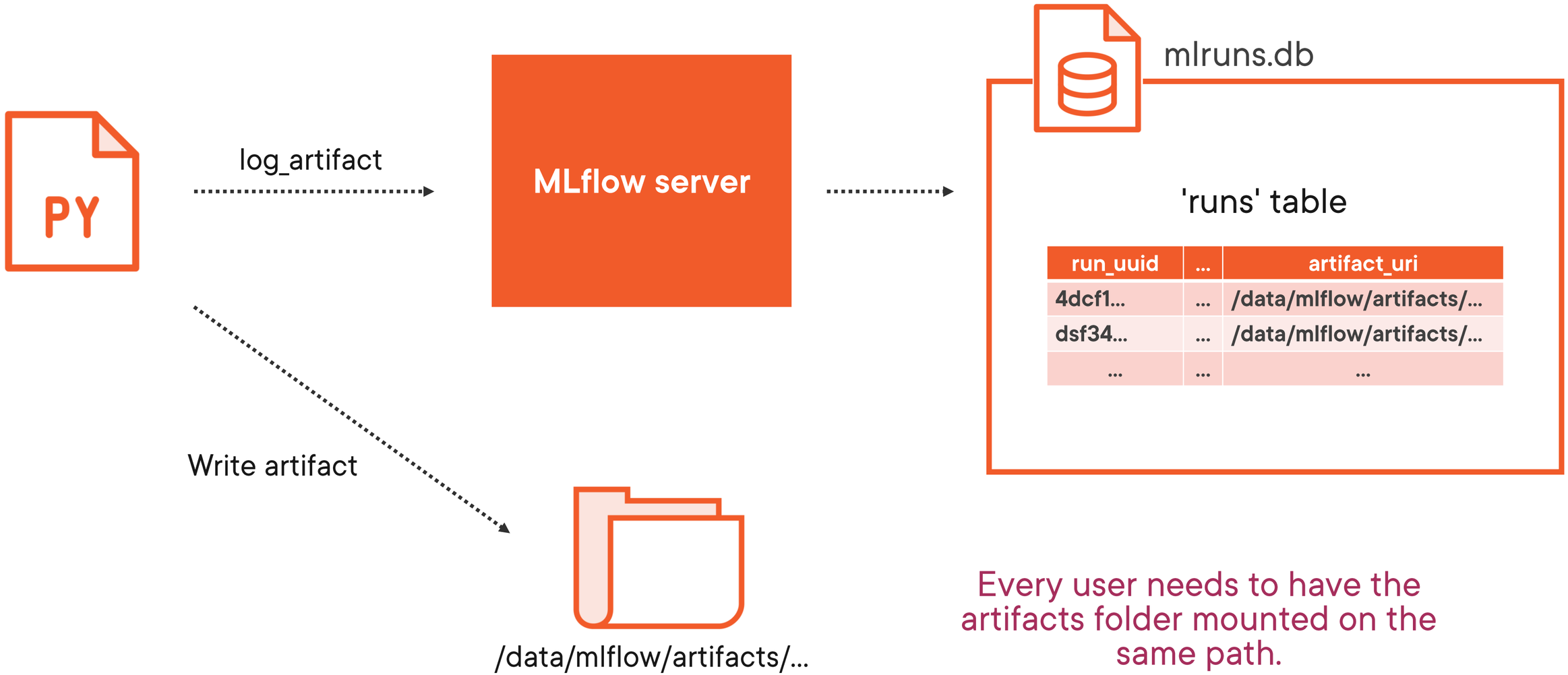
# Demo

## MLflow server

# Complete MLflow Deployment

Everyone needs to run their own MLflow server.

SQLite can be slow.




Every user needs to have the artifacts folder mounted on the same path.

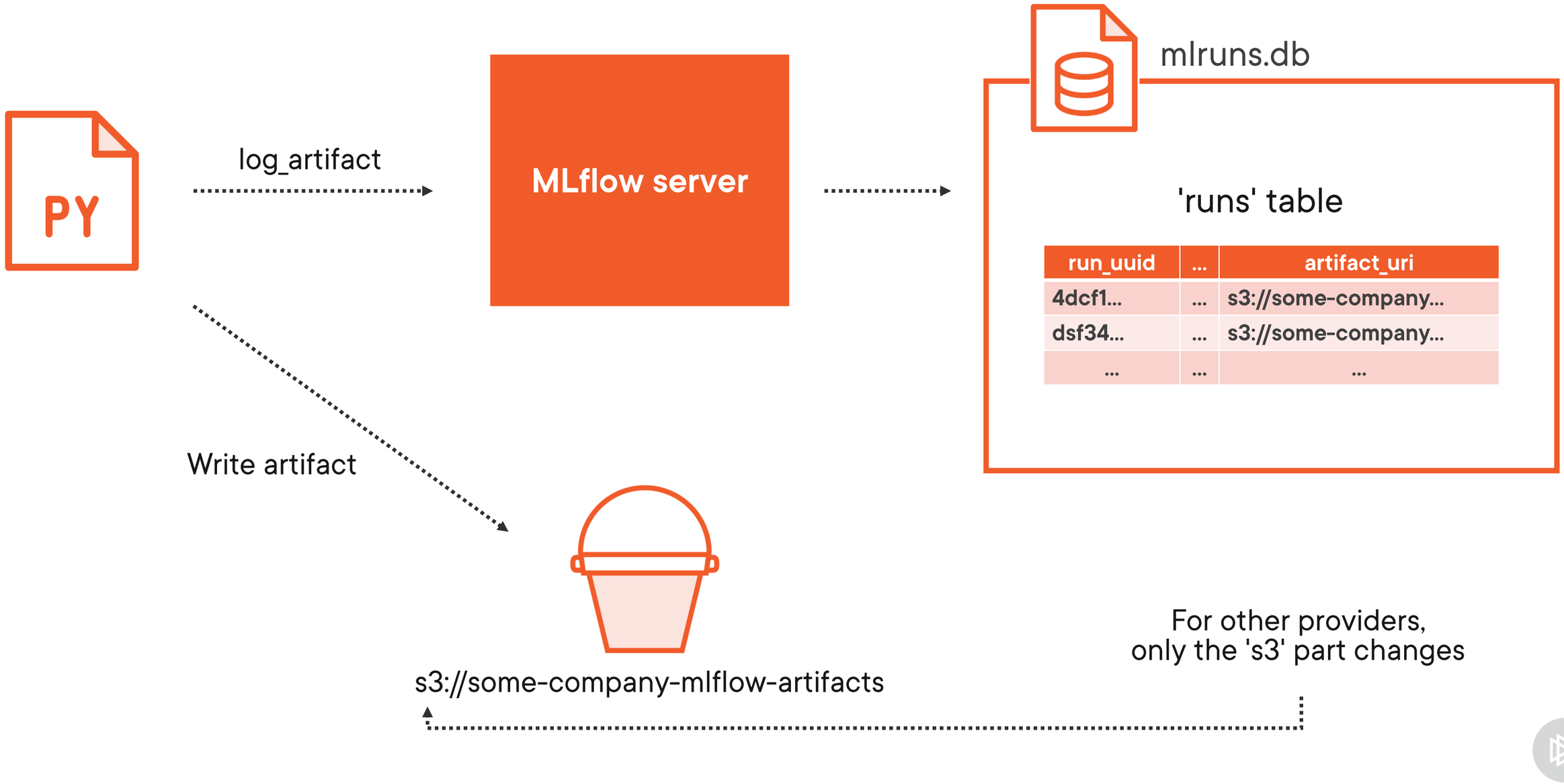
# Some Artifact Store Alternatives

# Some Artifact Store Alternatives

**Amazon S3**

**Azure Blob Storage**

**Google Cloud Storage**


PY
log_artifact
MLflow server
mlruns.db
'runs' table
run_uuid
...
artifact_uri
4dcf1...
...
s3://some-company...
dsf34...
...
s3://some-company...
...
...
...
Write artifact
s3://some-company-mlflow-artifacts
For other providers,
only the 's3' part changes

```
mlflow server --host 0.0.0.0
    --backend-store-uri
    sqlite:///data/mlflow/mlruns.db
    --default-artifact-root
    s3://some-company-mlflow-artifacts
```

# Cloud Storage for Artifacts

**Amazon S3**

# Other Artifact Store Options

**(S)FTP**

**HDFS**

# Remote Server


```
mlflow.set_tracking_uri(
    "mlflow.companyname.net"
)
```

mlflow.companyname.net

Backend store

# Summary

- Shortcomings of the default setup
- Using SQLite
- Distinction between artifact store and backend store
- MLflow server
- Architecture of a complete MLflow deployment

# Packaging and Running Models

**Paweł Kordek**

Software Engineer – Data and Machine Learning

@pawel_kordek

# Overview

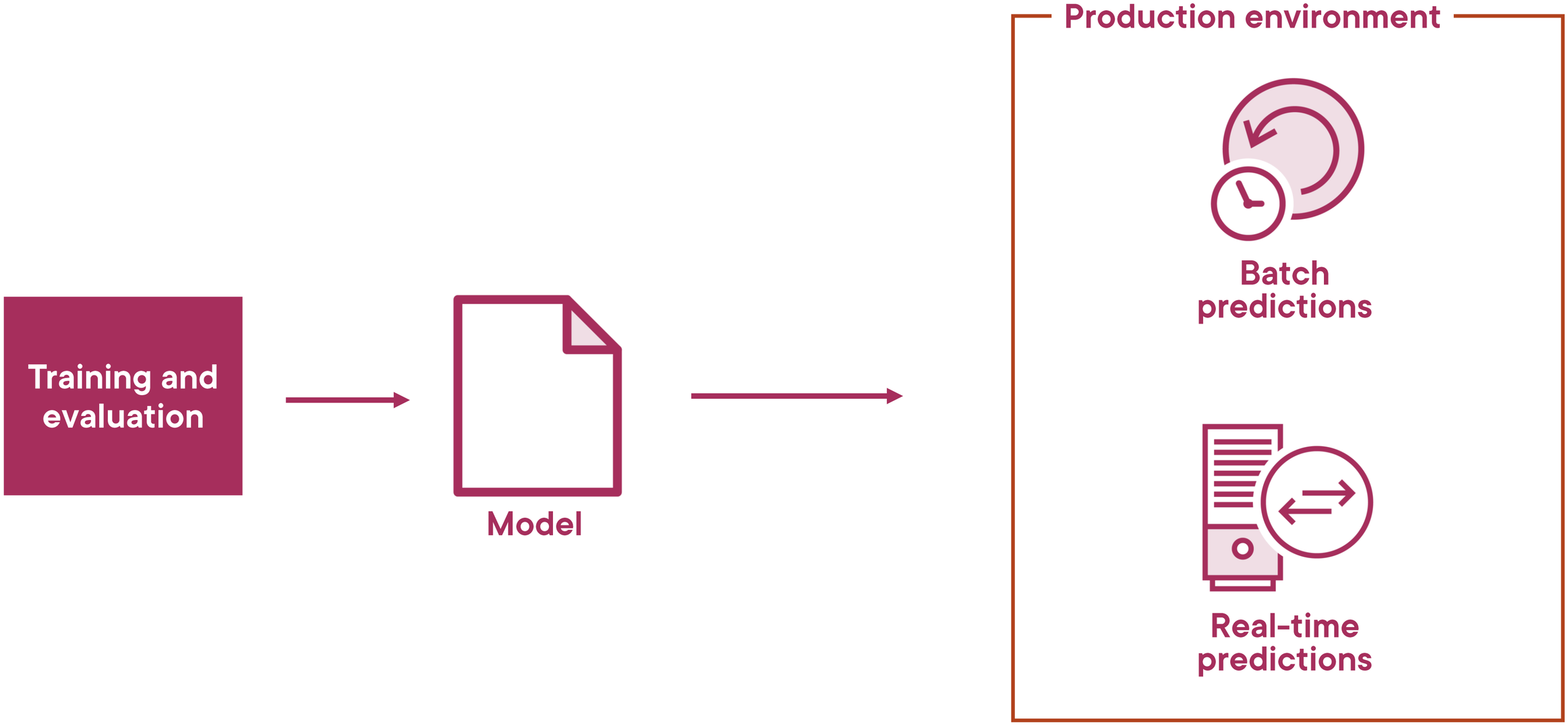
**Models as a special type of artifact**

**Models directly supported by MLflow**

**Custom models**

**Running predictions**

# Logging the Model


Training and evaluation
Model
Production environment
Batch predictions
Real-time predictions

```
lr_model = LinearRegression()
lr_model.fit(...)

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

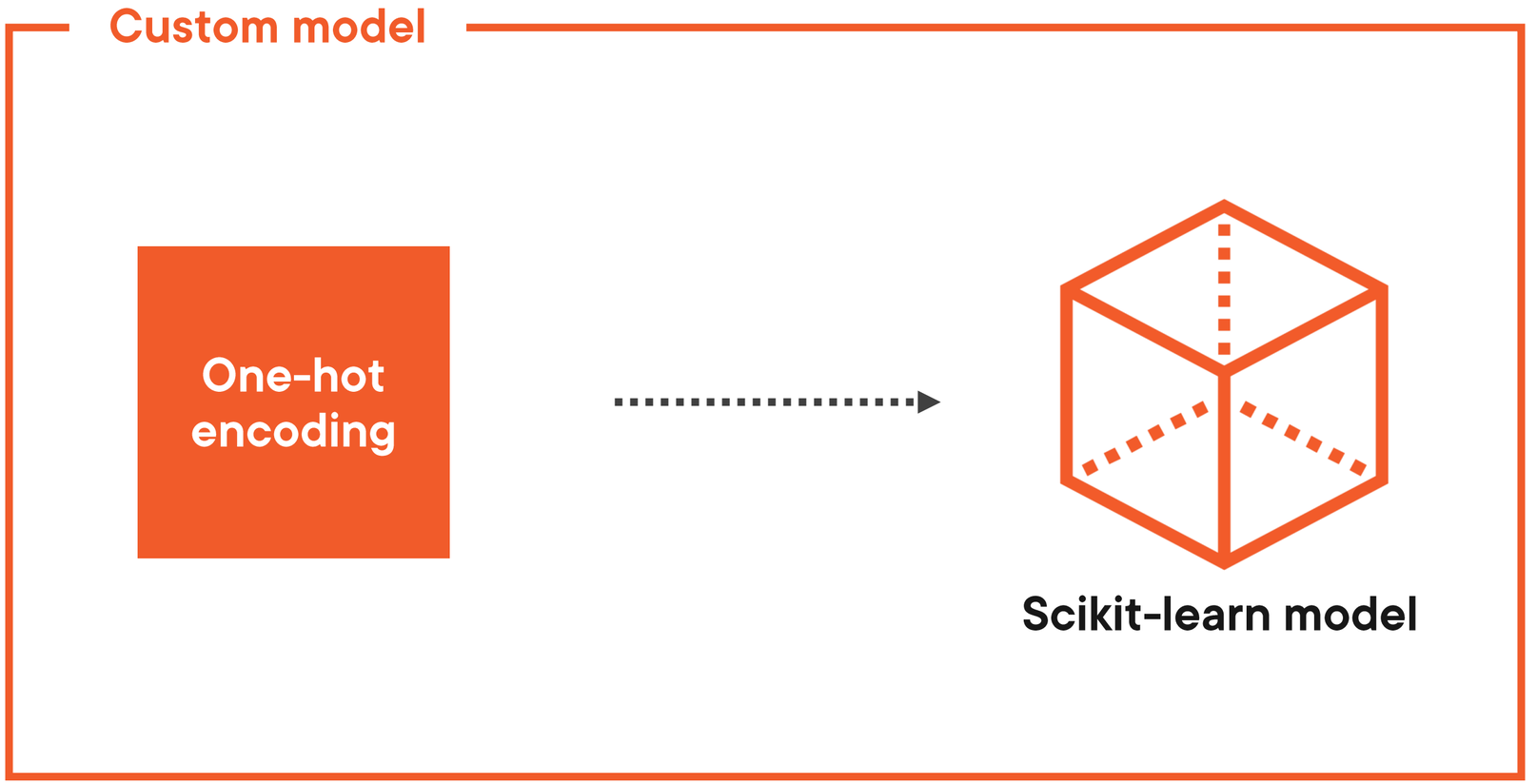
# Logging a Model

**Supported 'flavor' – e.g. scikit-learn**

# Demo

## Logging the trained model

# Customizing the Model



# Creating a Custom Model

'pyfunc' flavor

```
from mlflow.pyfunc import PythonModel


class CustomModel(PythonModel):

    def predict(self, ..., model_input):
        return np.array(...)


mlflow.pyfunc.log_model("model", python_model=CustomModel())
```

# Storing the Original Model

```
sklearn_model = LinearRegression()
...  # Train the model

with open("tmp/model.pkl", "wb") as m:
    m.write(sklearn_model)

artifacts = { "sklearn_model": "tmp/model.pkl" }

mlflow.pyfunc.log_model("model", python_model=CustomModel(), artifacts=artifacts)
```

# Loading the Original Model

```
from mlflow.pyfunc import PythonModel


class CustomModel(PythonModel):

    def load_context(self, context):
        with open(context.artifacts["sklearn_model"]) as m:
            self.sklearn_model = pickle.load(m)

    def predict(self, ..., model_input):
        return self.sklearn_model.predict(model_input)
```

# Demo

**Custom model class**

**One-hot encoding**

# Running Predictions

# Batch Predictions

```
mlflow models predict
    -m <MODEL_PATH>
    -i <INPUT_DATASET_FILE>
    --no-conda
```

# Batch Predictions

```
mlflow models predict
    -m runs:/de223jj8sd…/model
    -i <INPUT_DATASET_FILE>
    --no-conda
```

Run ID

Artifacts subdirectory for the model

# Batch Predictions

```
$Env:MLFLOW_TRACKING_URI = "http://localhost:5000"
export MLFLOW_TRACKING_URI="http://localhost:5000"

mlflow models predict
    -m runs:/de223jj8sd…/model
    -i <INPUT_DATASET_FILE>
    --no-conda
```

Windows

Linux/macOS

# Batch Predictions

```
$Env:MLFLOW_TRACKING_URI = "http://localhost:5000"
export MLFLOW_TRACKING_URI="http://localhost:5000"

mlflow models predict
    -m runs:/de223jj8sd…/model
    -i <INPUT_DATASET_FILE>
    --no-conda
```

CSV or JSON – Pandas compatible

# Batch Predictions

```
$Env:MLFLOW_TRACKING_URI = "http://localhost:5000"
export MLFLOW_TRACKING_URI="http://localhost:5000"

mlflow models predict
    -m runs:/de223jj8sd…/model
    -i <INPUT_DATASET_FILE>
    --no-conda
    -t csv
```

# Batch Predictions

```
$Env:MLFLOW_TRACKING_URI = "http://localhost:5000"
export MLFLOW_TRACKING_URI="http://localhost:5000"

mlflow models predict
    -m runs:/de223jj8sd…/model
    -i <INPUT_DATASET_FILE>
    --no-conda
    -t csv
    -o results.json
```

# Demo

**Predictions using existing model**

**Static dataset**

**Real-time predictions**

# Summary

**Models as a special class of artifacts**

**Custom models**

**Running predictions**

# Sharing and Managing Models with Model Registry

**Paweł Kordek**

Software Engineer – Data and Machine Learning

@pawel_kordek

# Overview

**Problems that Model Registry solves**

**Working with Model Registry in the UI**

**Using Model Registry when training and running predictions**


Training and evaluation
Model
Production environment
Batch predictions
Real-time predictions

Production environment

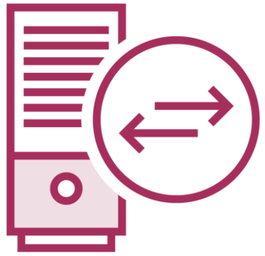
Batch predictions

House prices

Focus areas

Another task

...



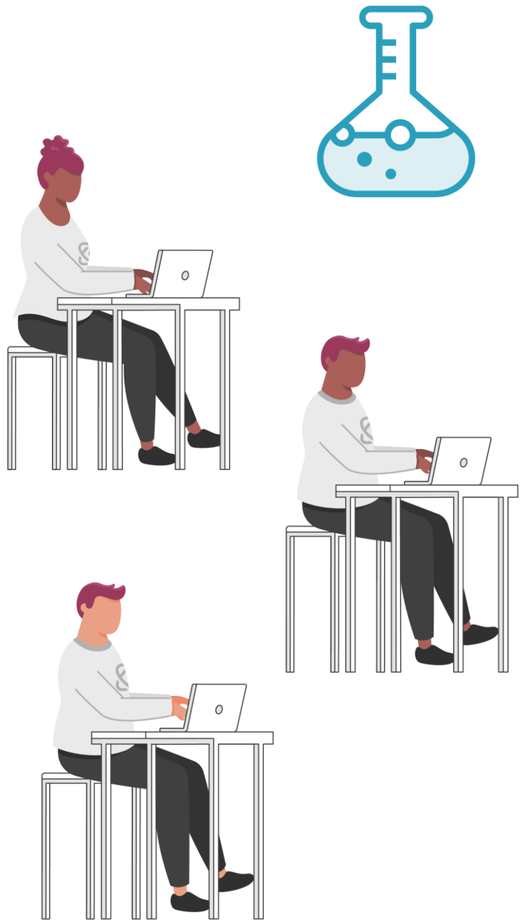
Real-time predictions

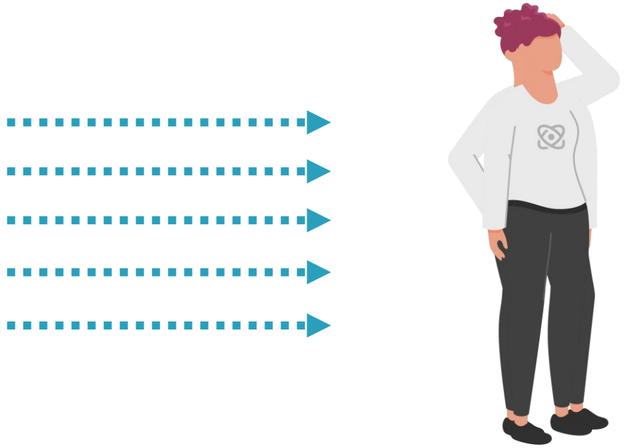
House prices

Focus areas

Another task

...










Takes care of model deployments

# Model Registry

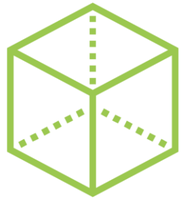
# Registered Model

**House prices**

runs:/3d7133.../model

**Version 1**

**Archived**



runs:/503685.../model

**Version 2**

**Production**

runs:/e899c.../model

**Version 3**

**Staging**

# Demo

**Creating registered models**

**Registering individual models**

**Managing stages**

**Running predictions**

# Summary

- Problems that Model Registry solves
- Registered models, versions, and stages
- Registering models from the UI
- Registering models from training scripts
- Using models from the registry

# Summary

**Paweł Kordek**

Software Engineer – Data and Machine Learning

@pawel_kordek

# Machine Learning Workflows

**Common problems**

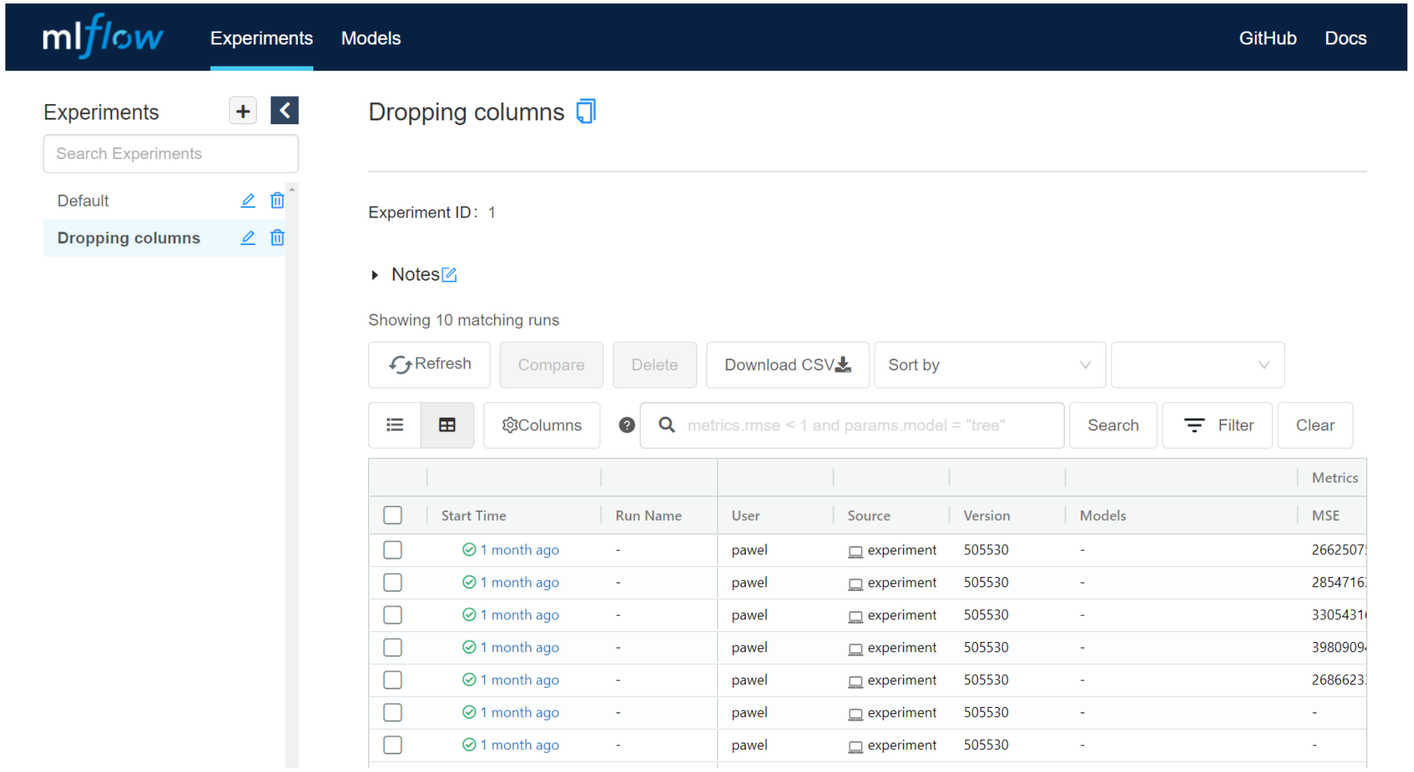
**MLOps**

**Role of MLflow**

# MLflow Tracking

## Experiments, runs, parameters, and metrics

## MLflow UI



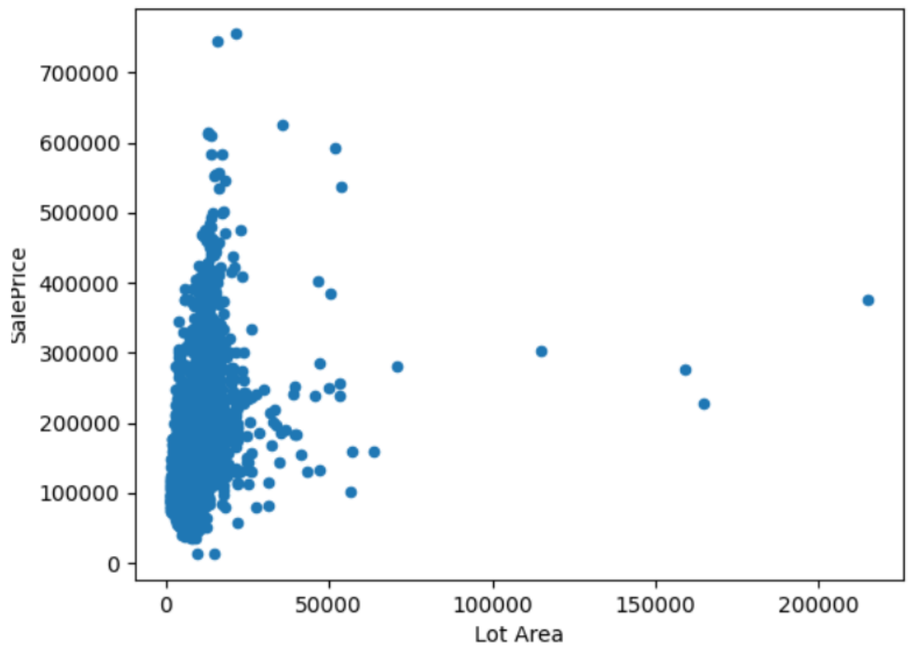
# Artifacts

Artifacts

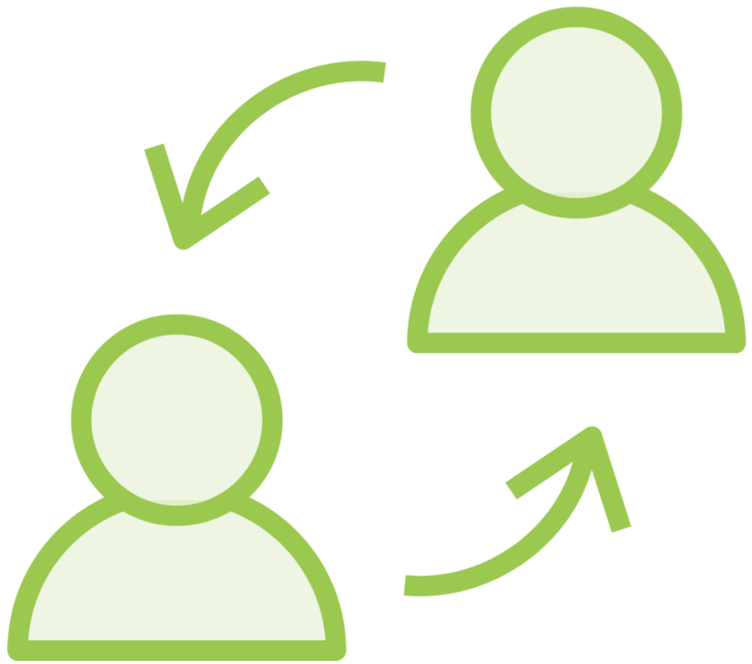
- dataset.csv
- model.pkl
- plot.png

Full Path:file:///'module4/mlruns/1/cf029ad774044220bdc64c96bb...

Size: 24.12KB



# MLflow in Teams



**MLflow architecture**

**Backend store vs. artifact store**

**MLflow server**

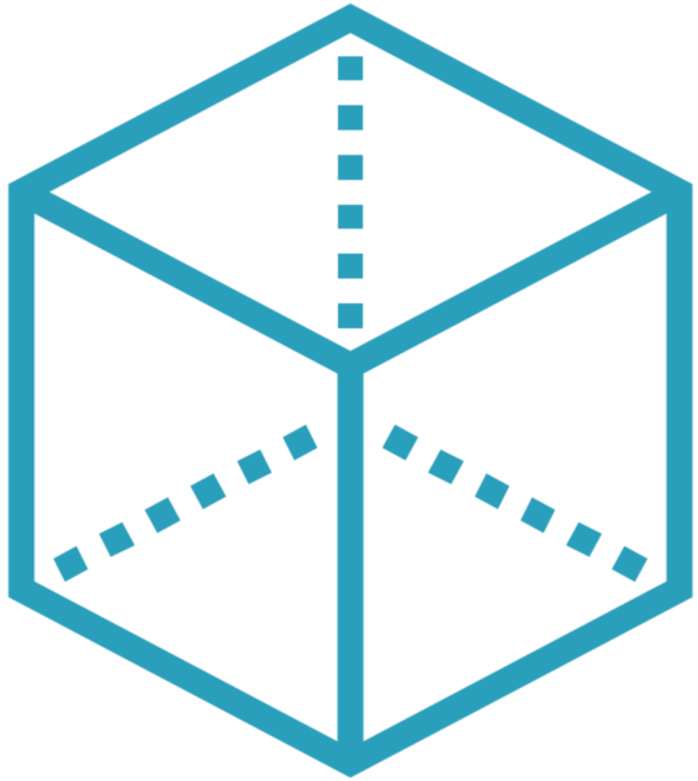
**Complete MLflow deployment**

# MLflow Models

**Supported flavors**

**'Pyfunc' flavor for custom models**

**Running predictions in batch + real-time**



# Model Registry

**Registered models**

**Versioning and stages**

**Running predictions**

# Thank You!