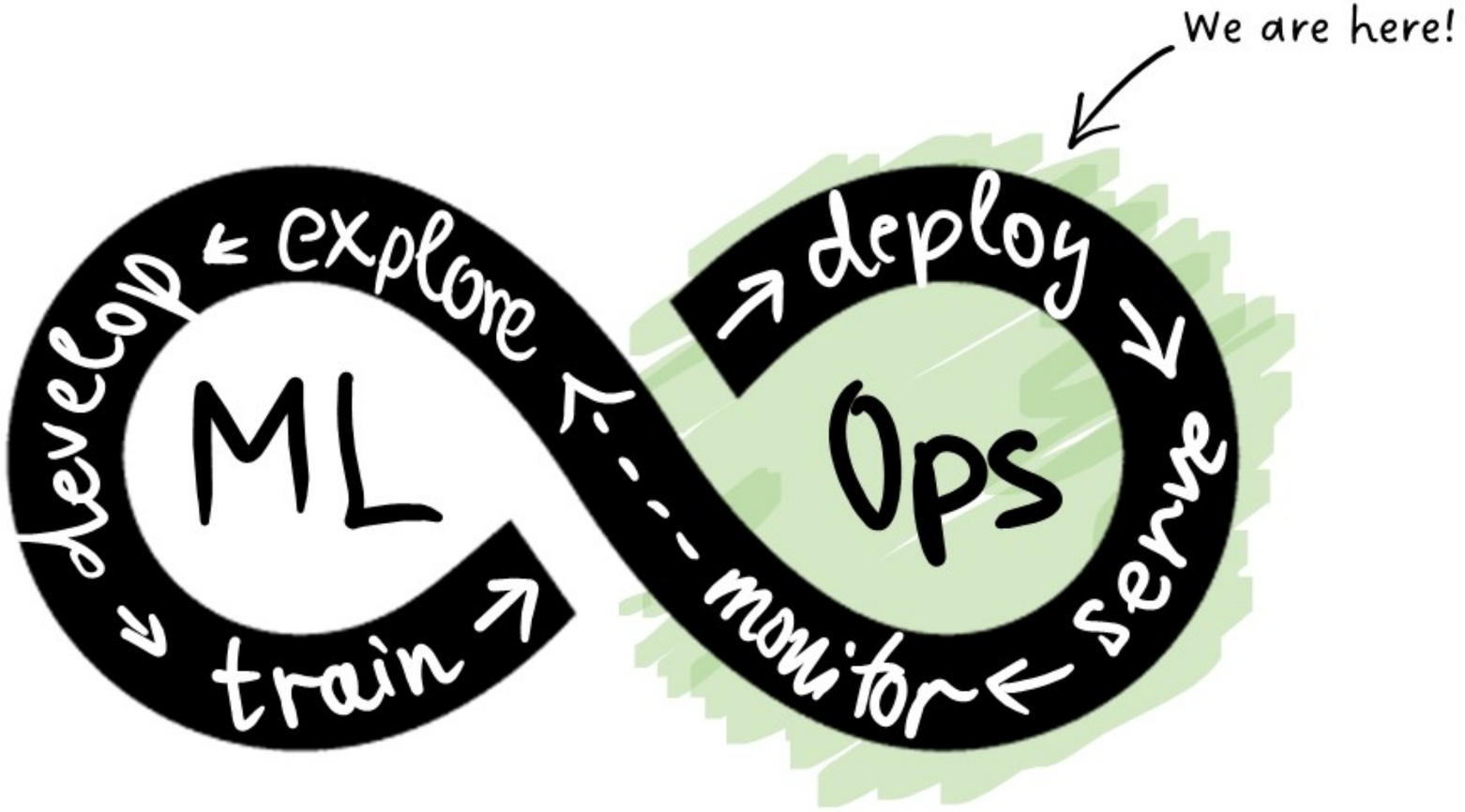
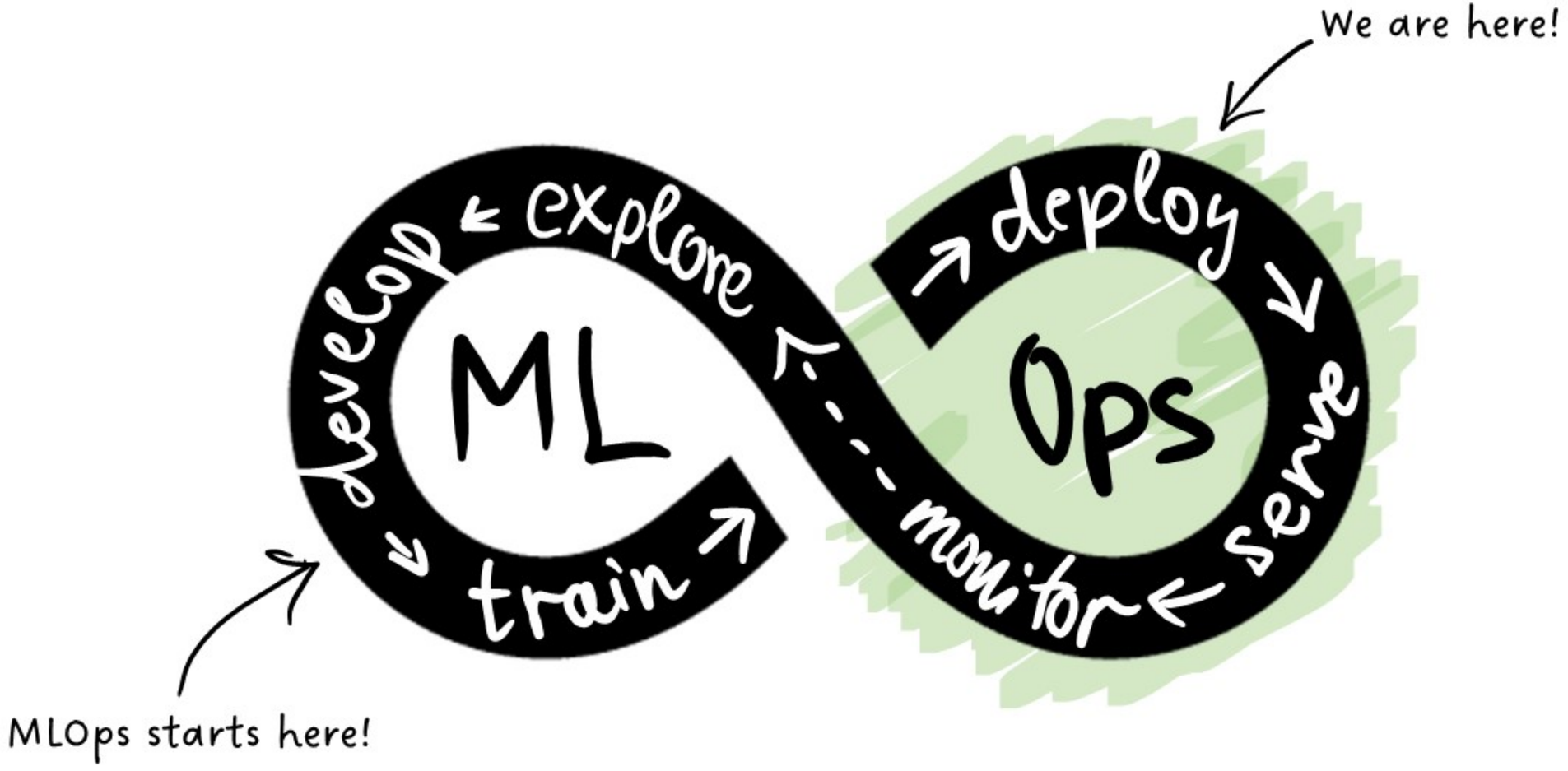
Deployment-driven development

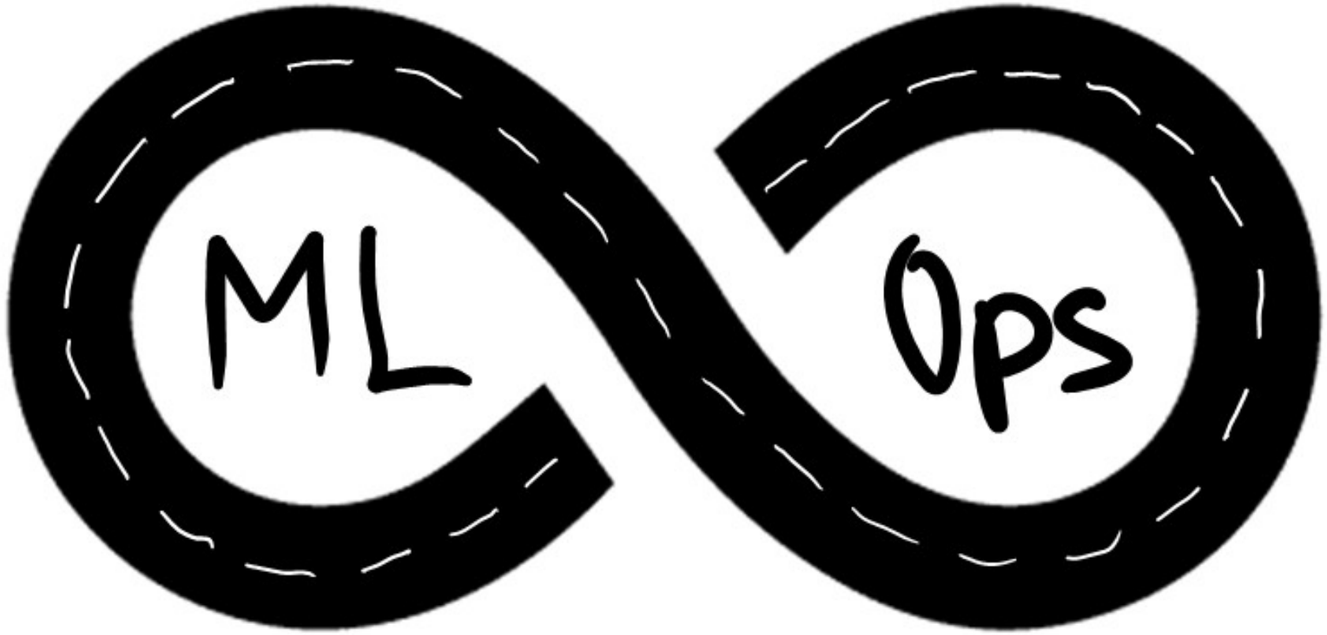
MLOPS DEPLOYMENT AND LIFE CYCLING

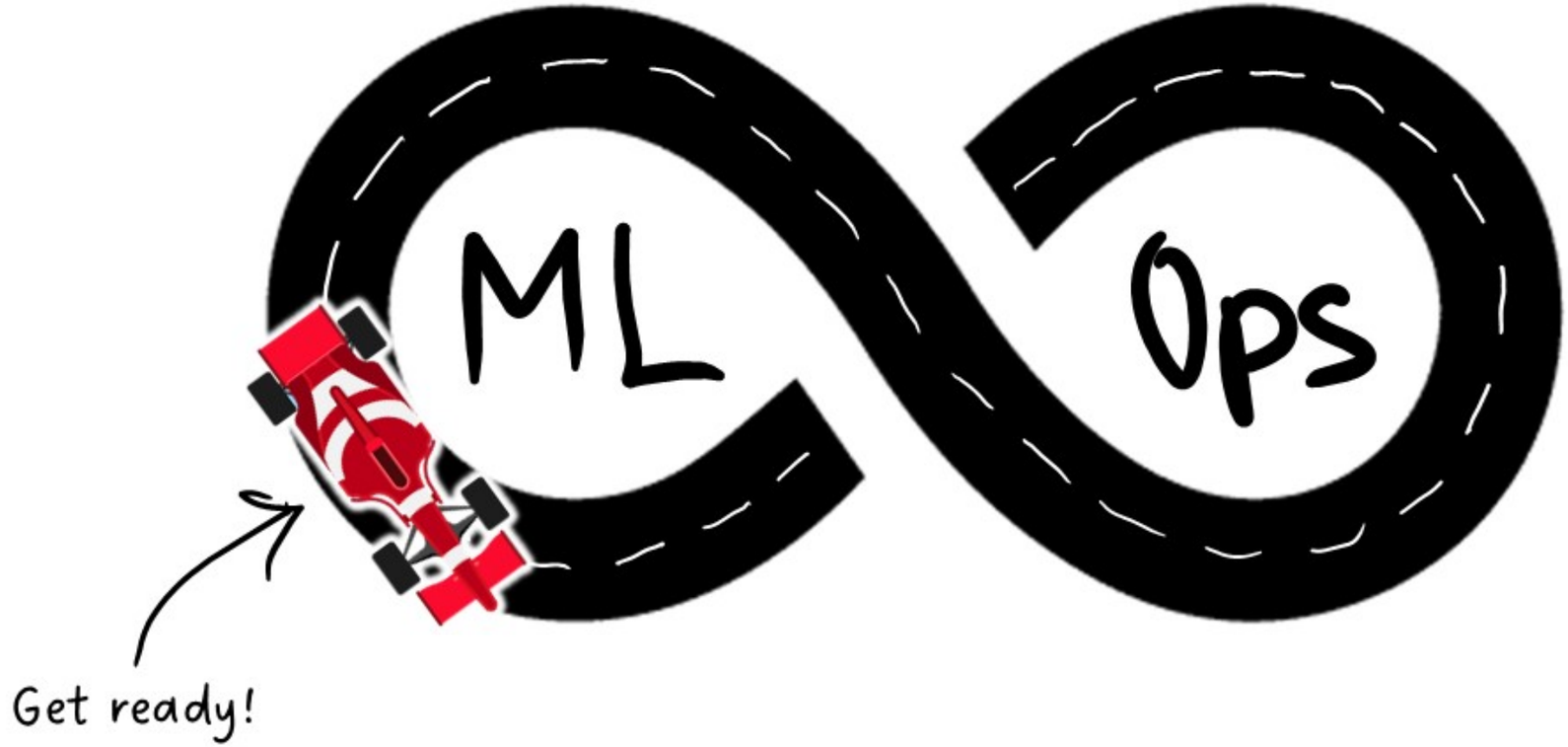
Nemanja Radojkovic
Senior Machine Learning Engineer

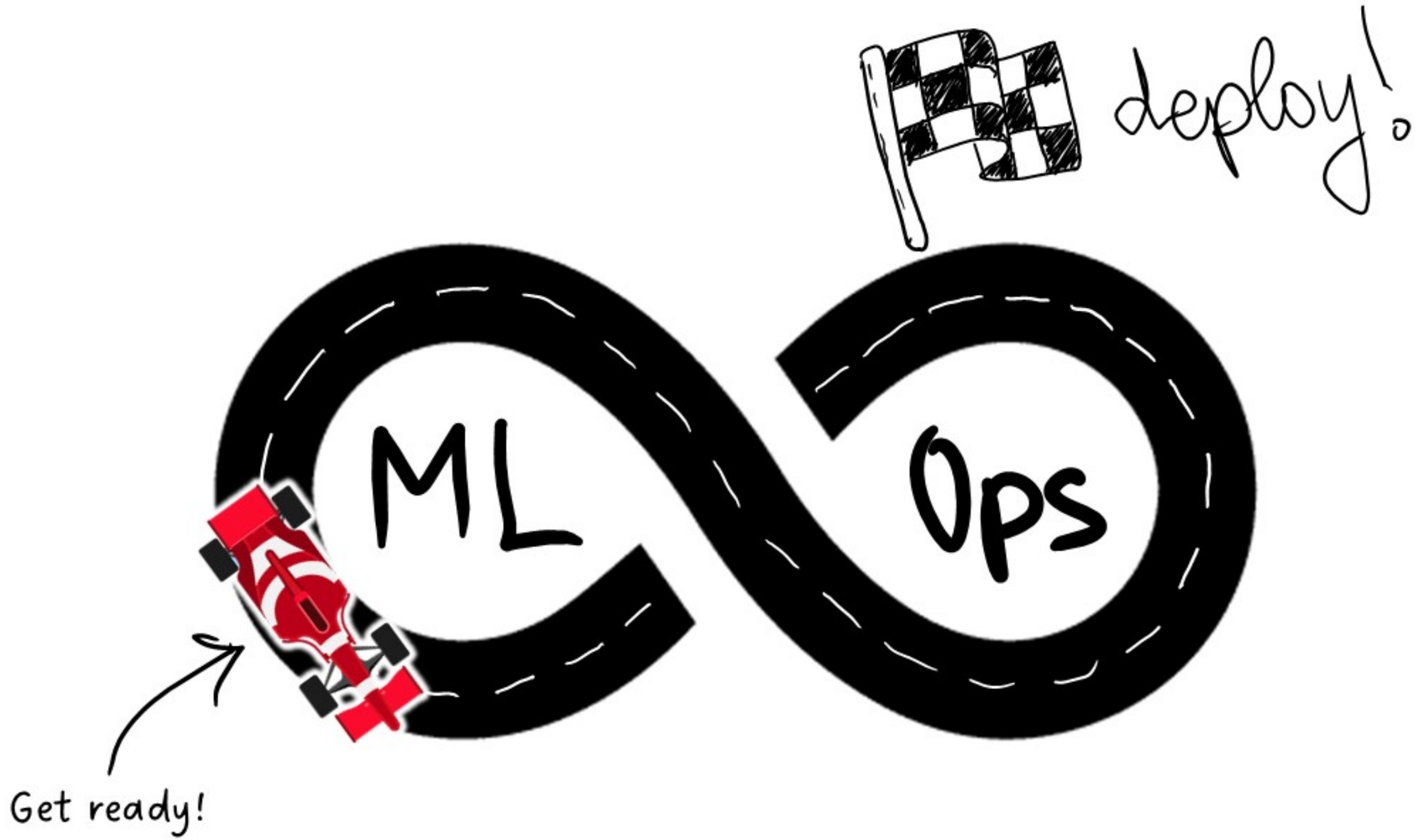
datacamp

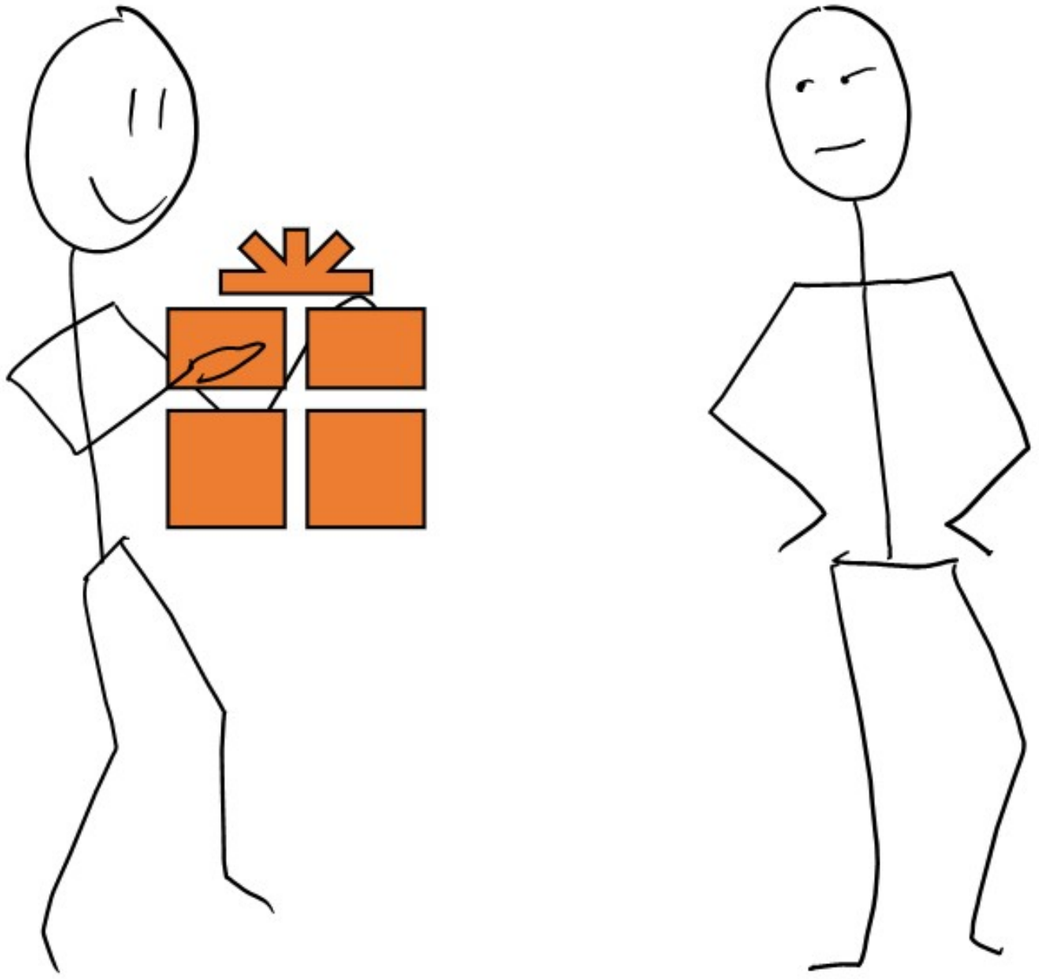

We are here!
develop
explore
deploy
ML
Ops
train
monitor
serve

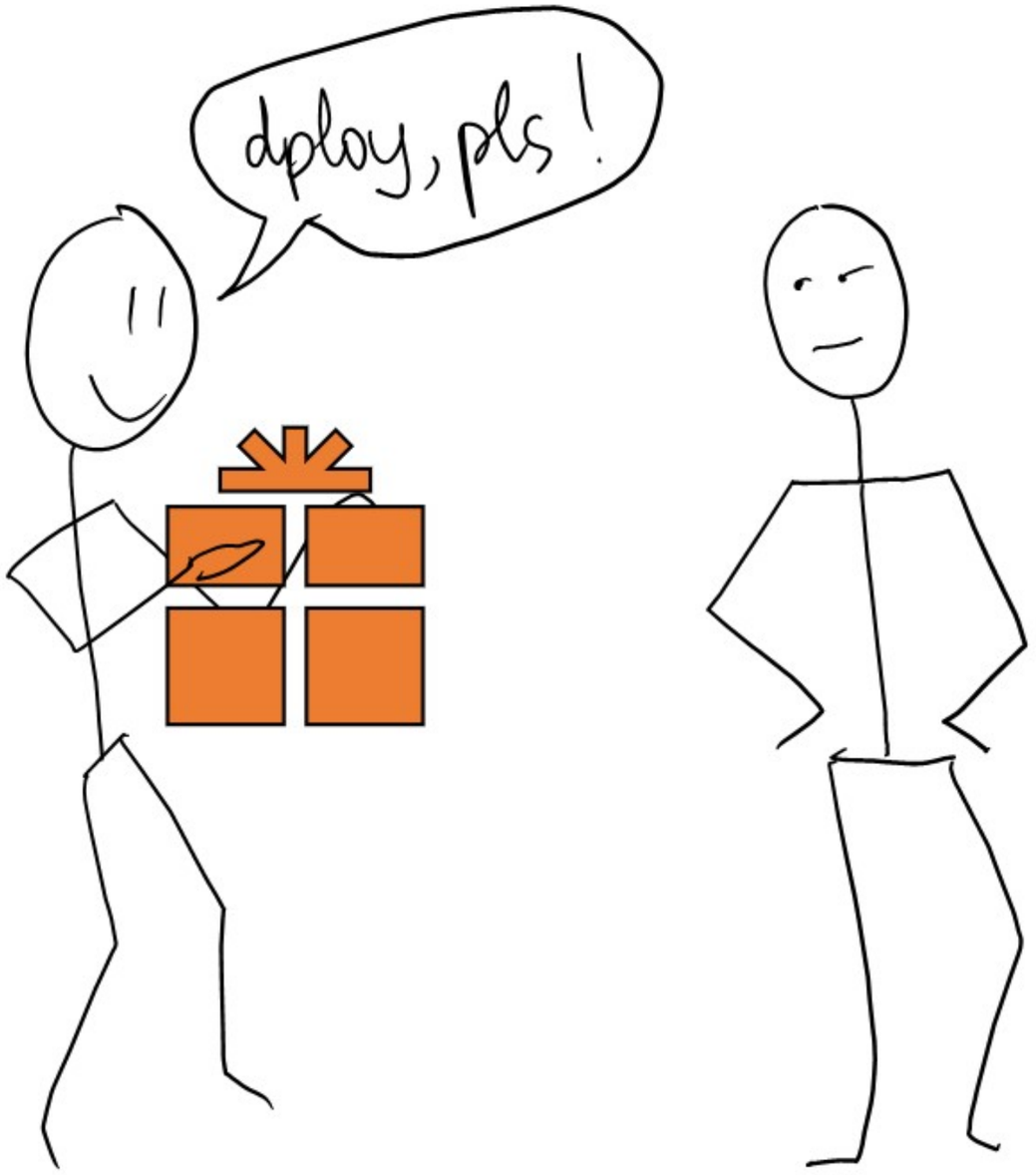

We are here!
develop
explore
deploy
ML
Ops
train
monitor
serve
MLOps starts here!

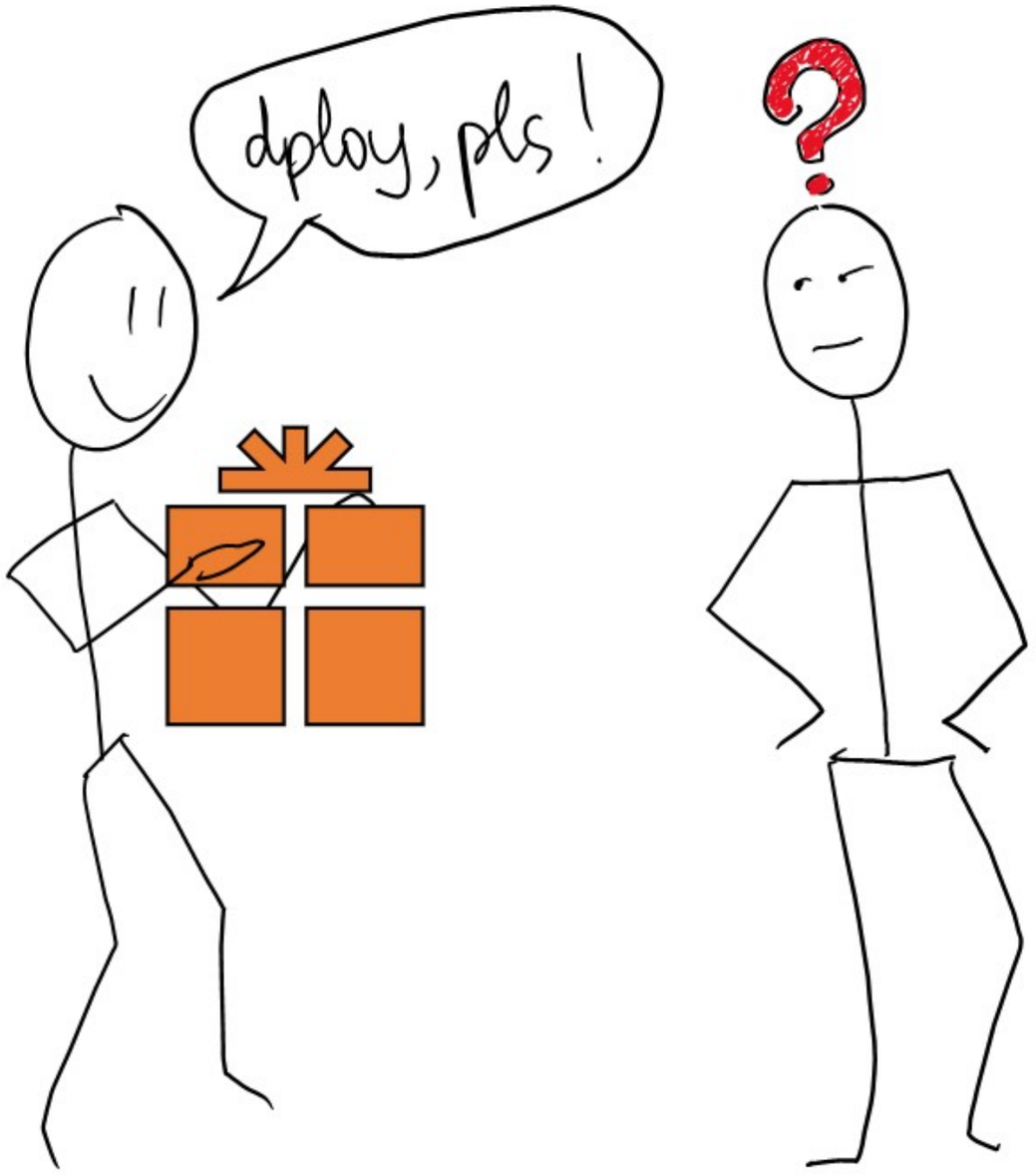

ML
Ops

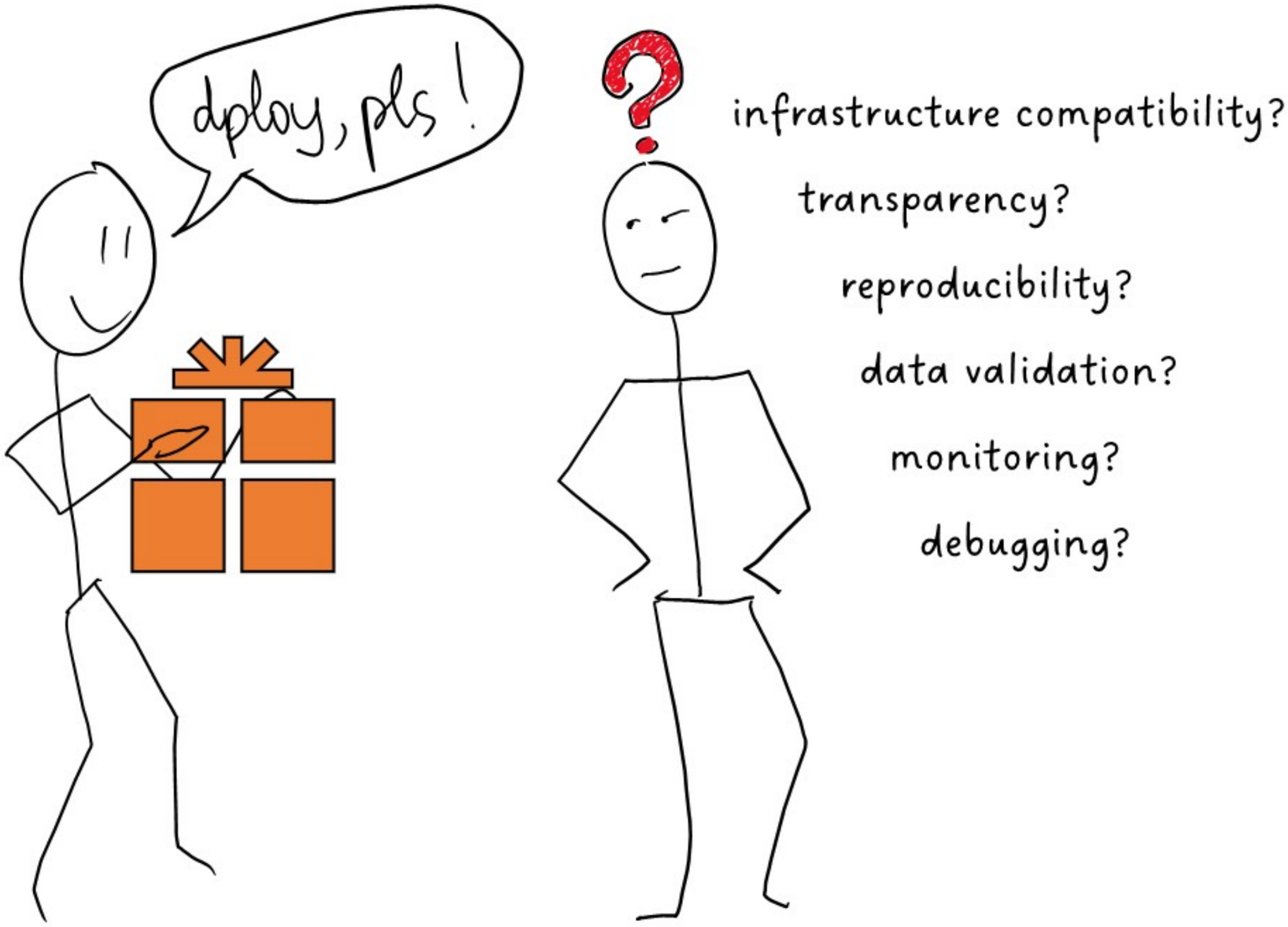

ML
Ops
Get ready!

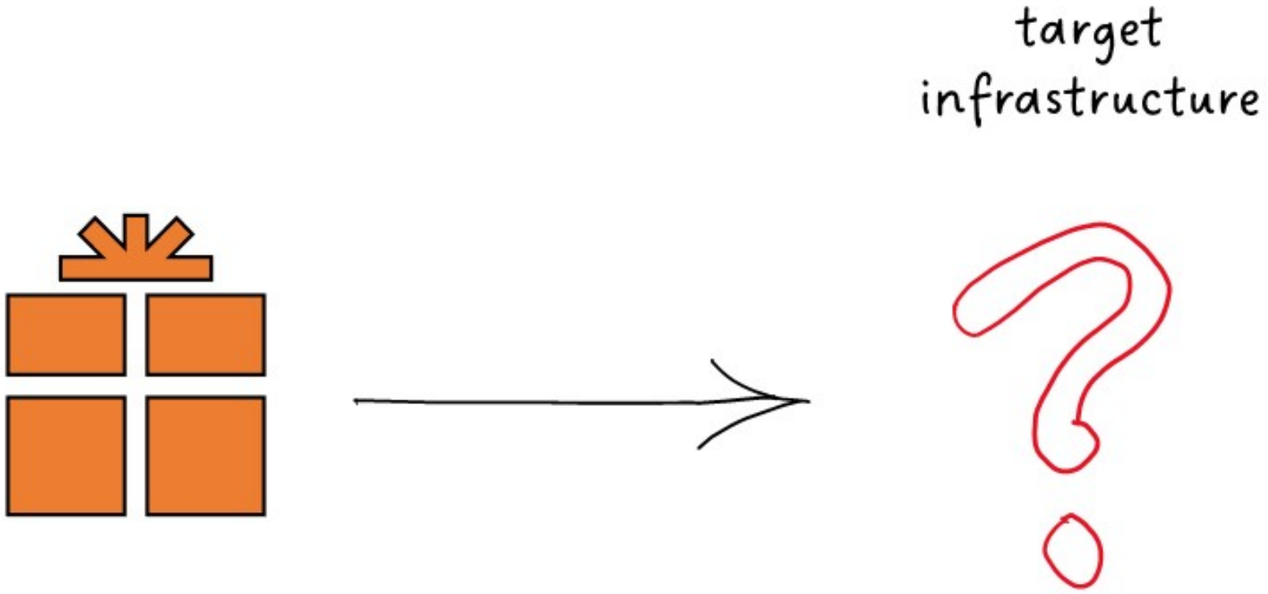

deploy!
ML
Ops
Get ready!

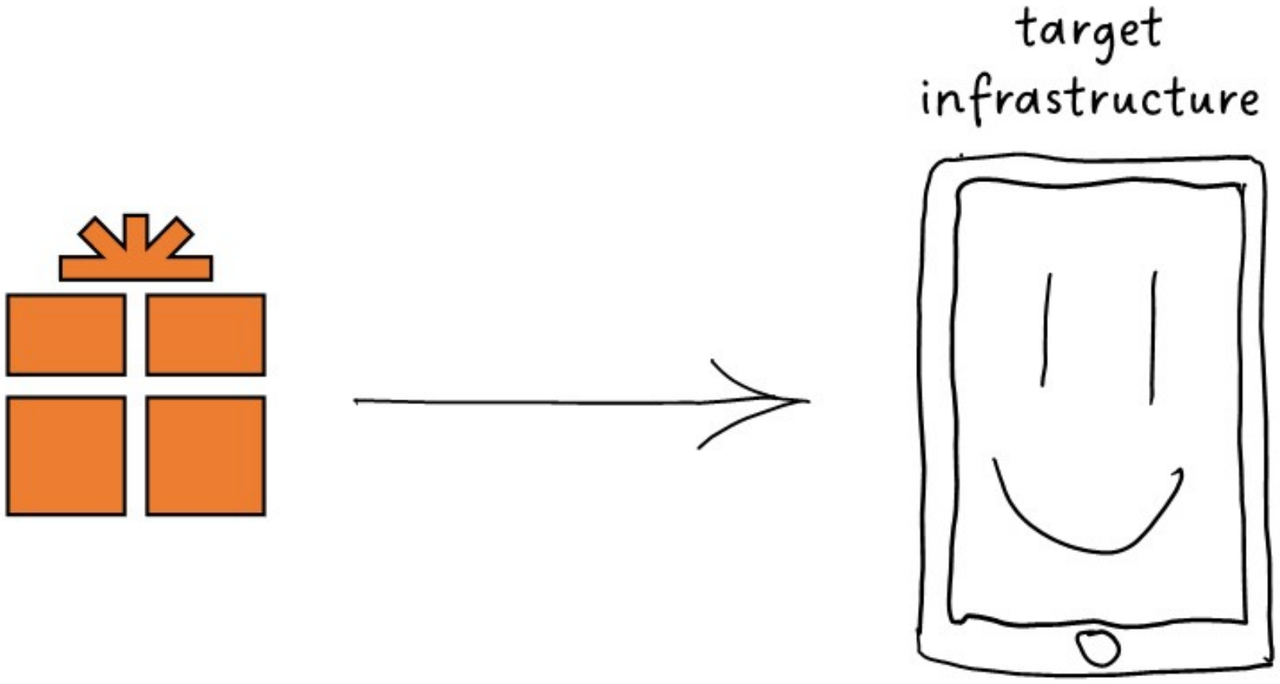


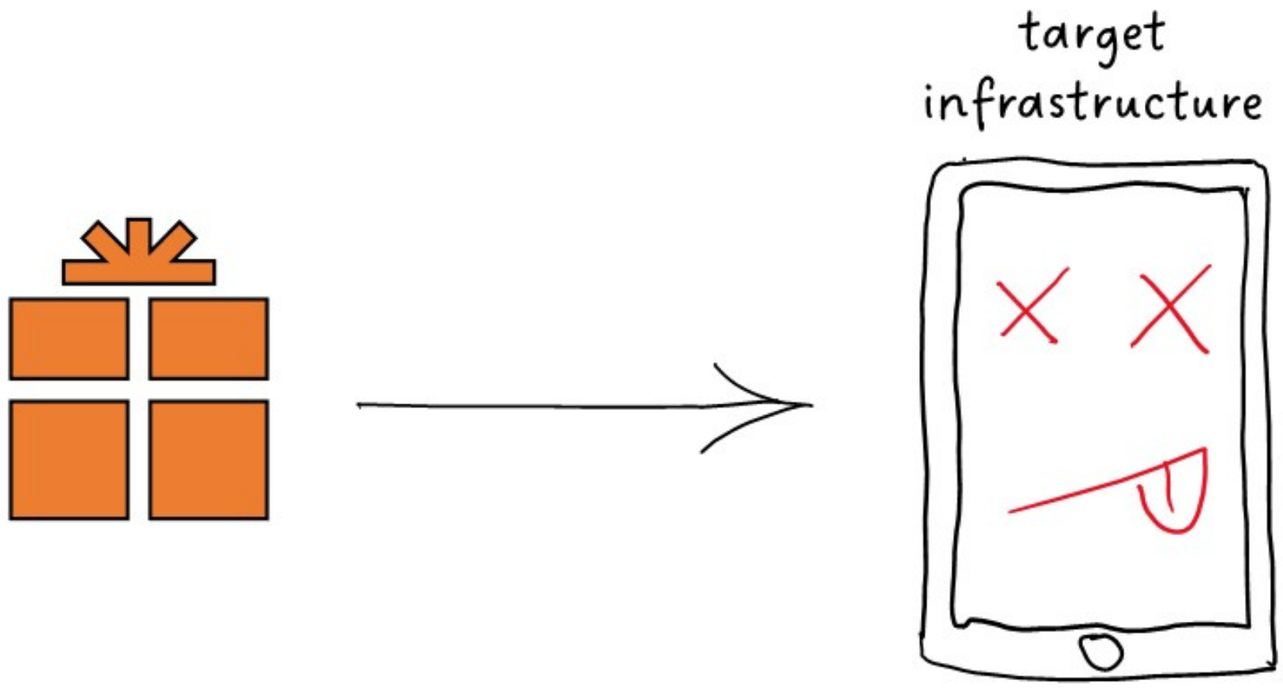

dploy, pls !


dploy, pls !

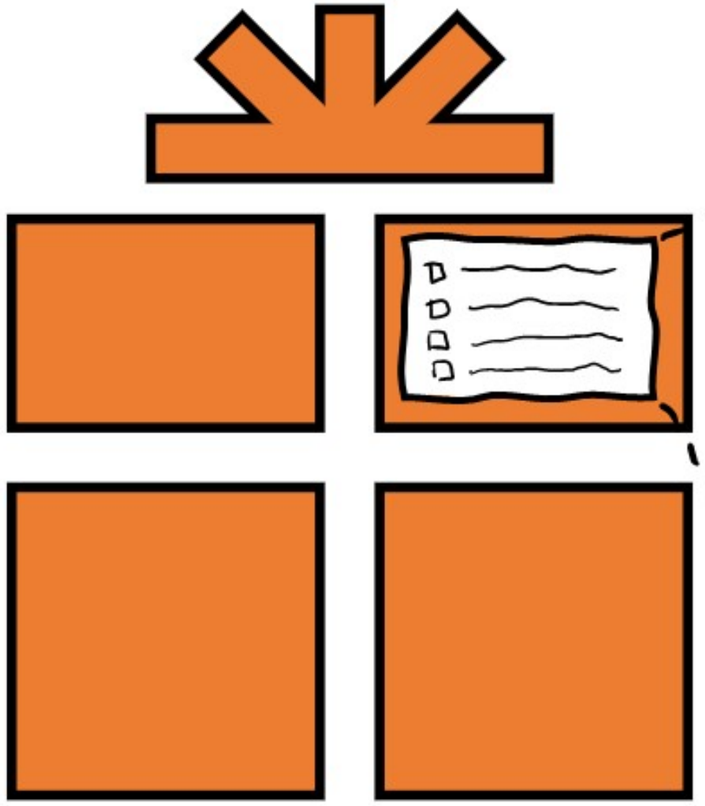

dploy, pls !
infrastructure compatibility?
transparency?
reproducibility?
data validation?
monitoring?
debugging?


target
infrastructure


target
infrastructure

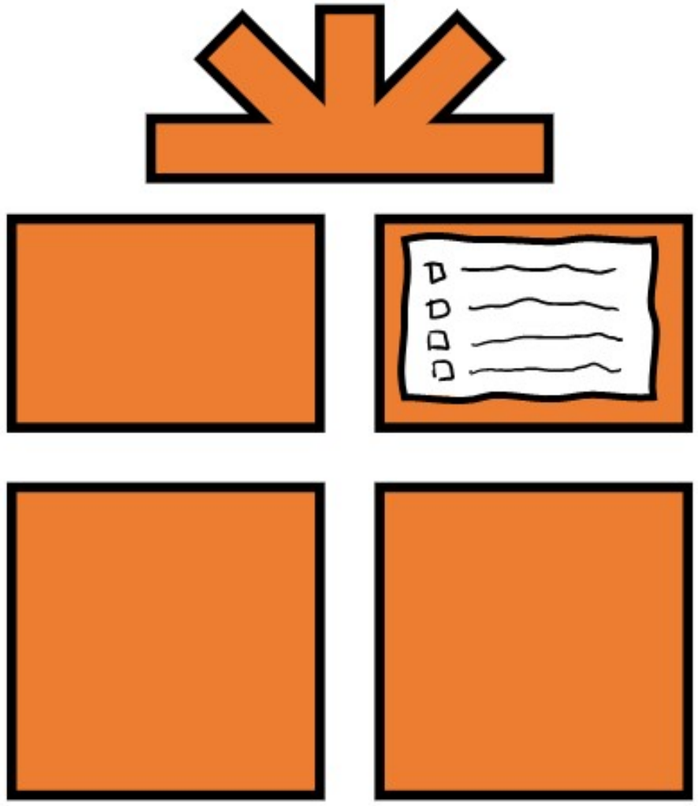

target
infrastructure

Transparency & Reproducibility



- ❑ Who trained this model?
- ❑ When?
- ❑ Using which script?
- ❑ What hyper-parameters?
- ❑ ...

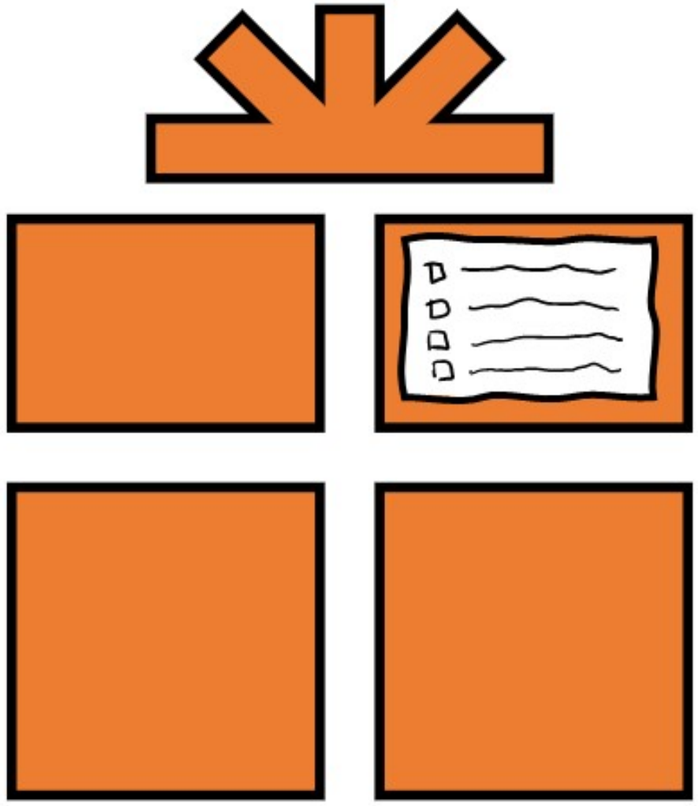
Transparency & Reproducibility



No transparency & reproducibility?

⇒ Return to sender!

Transparency & Reproducibility



No transparency & reproducibility?

⇒ Return to sender!

Versioned datasets + fully transparent pipelines?

⇒ Good to go!

Transparency & Reproducibility

No dilemma how we go from code + raw data to the final model

Transparency & Reproducibility

No dilemma how we go from code + raw data to the final model

Recreating the exact model simple and straightforward

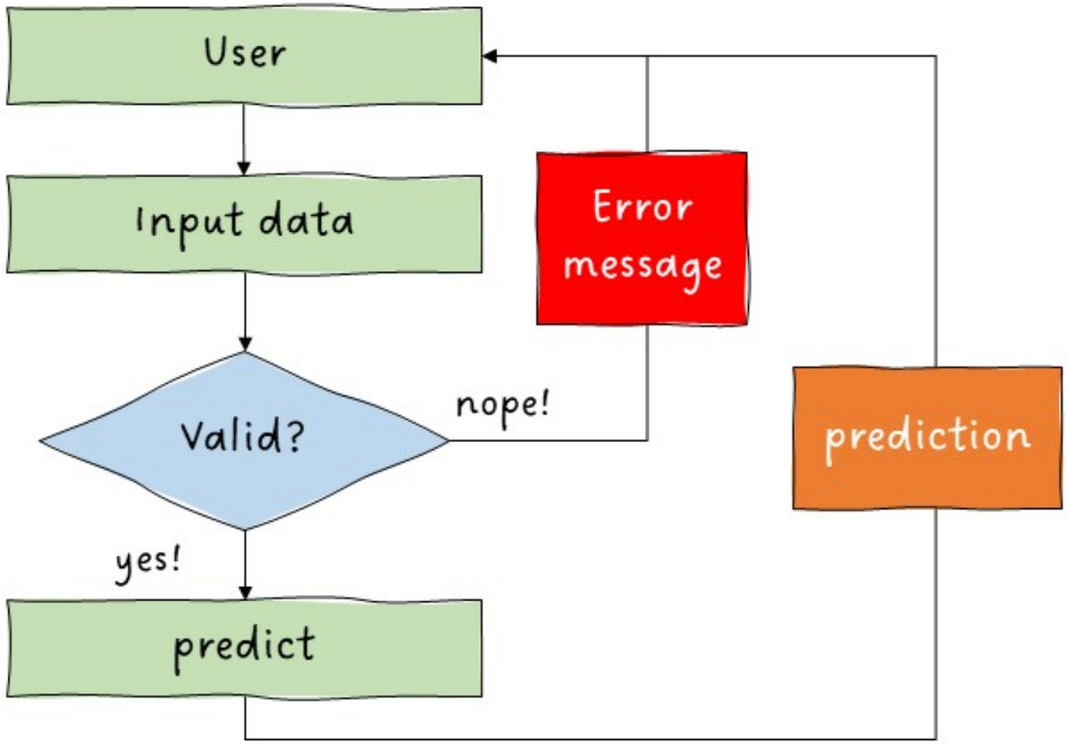
Transparency & Reproducibility

No dilemma how we go from code + raw data to the final model

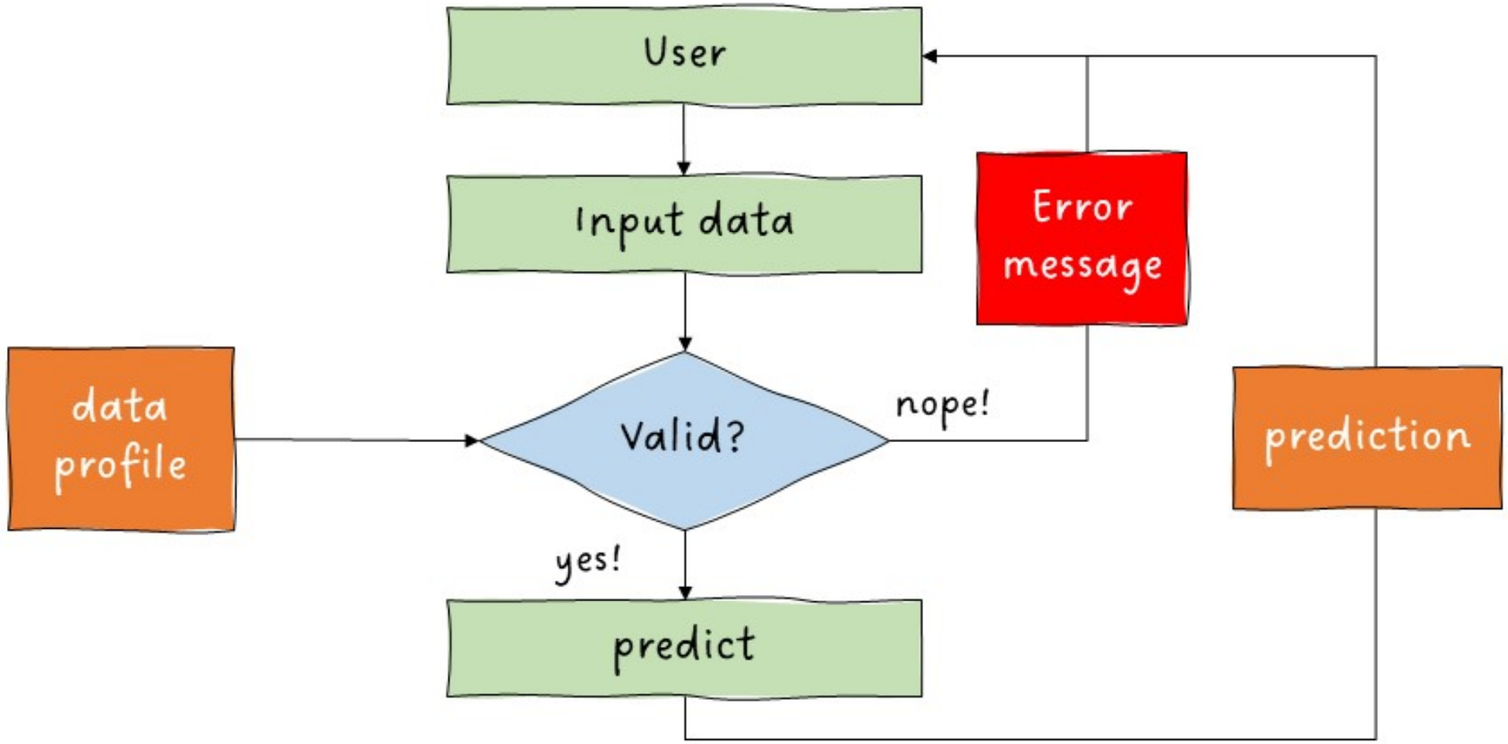
Recreating the exact model simple and straightforward

bonus points: log experiments in metadata store

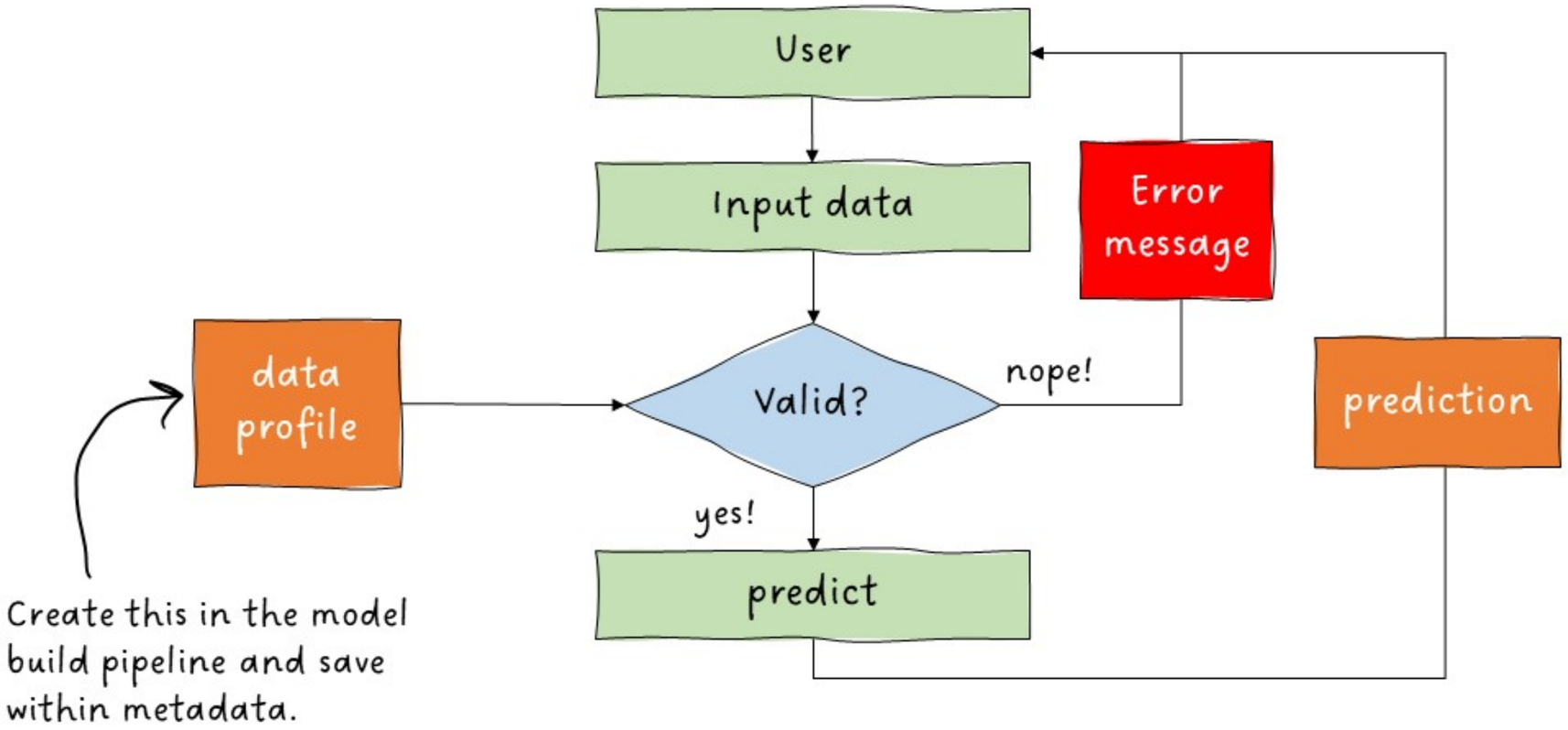
Concern: Input data validation



Concern: Input data validation

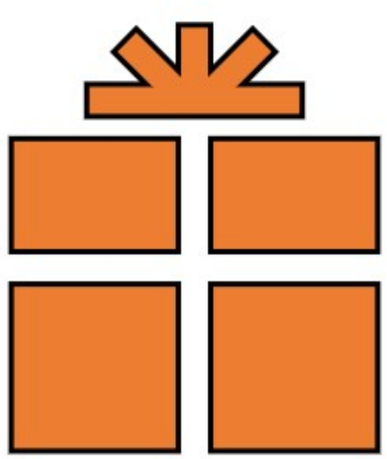


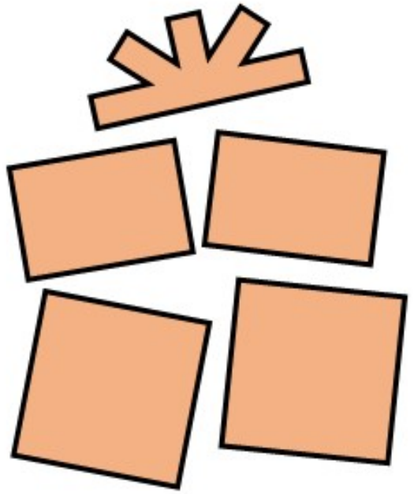
Concern: Input data validation



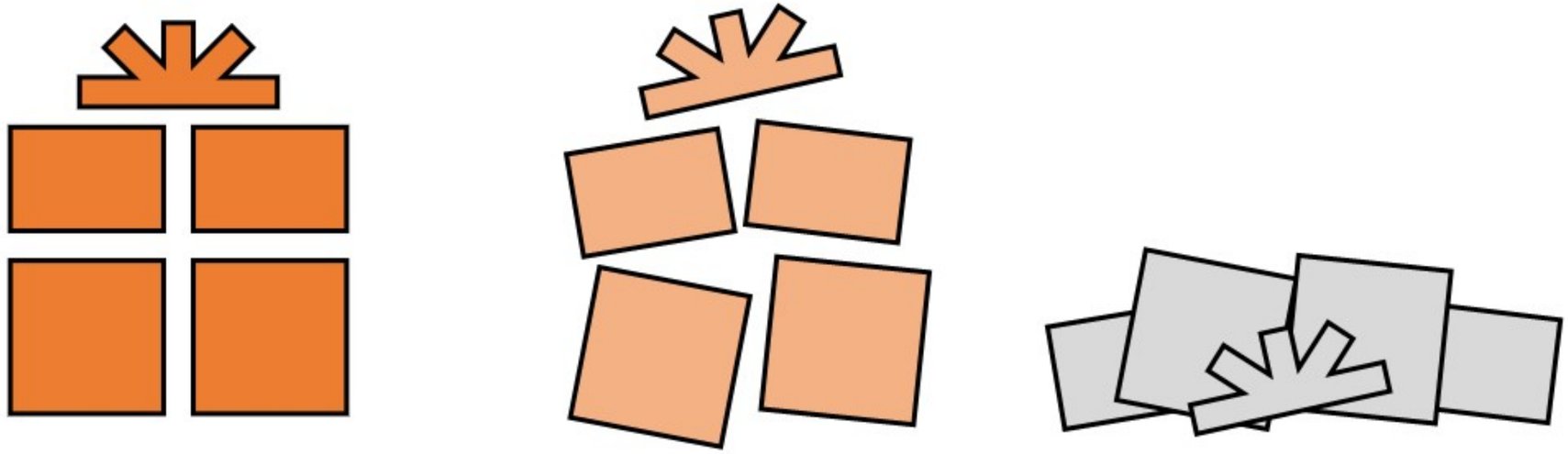
Concern: Performance deterioration

Concern: Performance deterioration

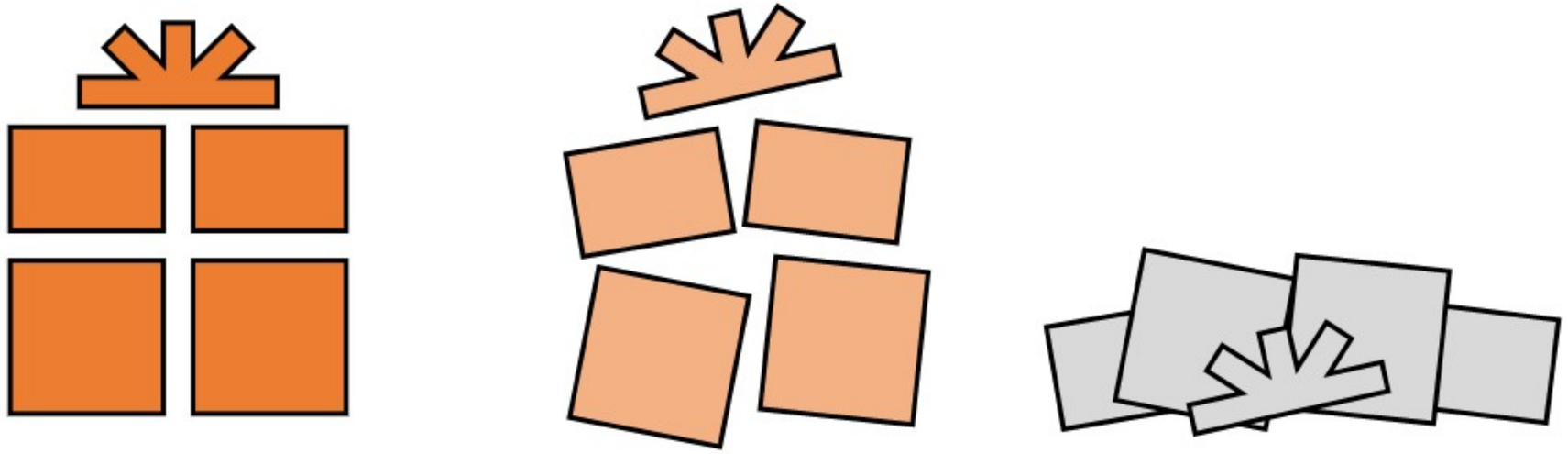




Concern: Performance deterioration

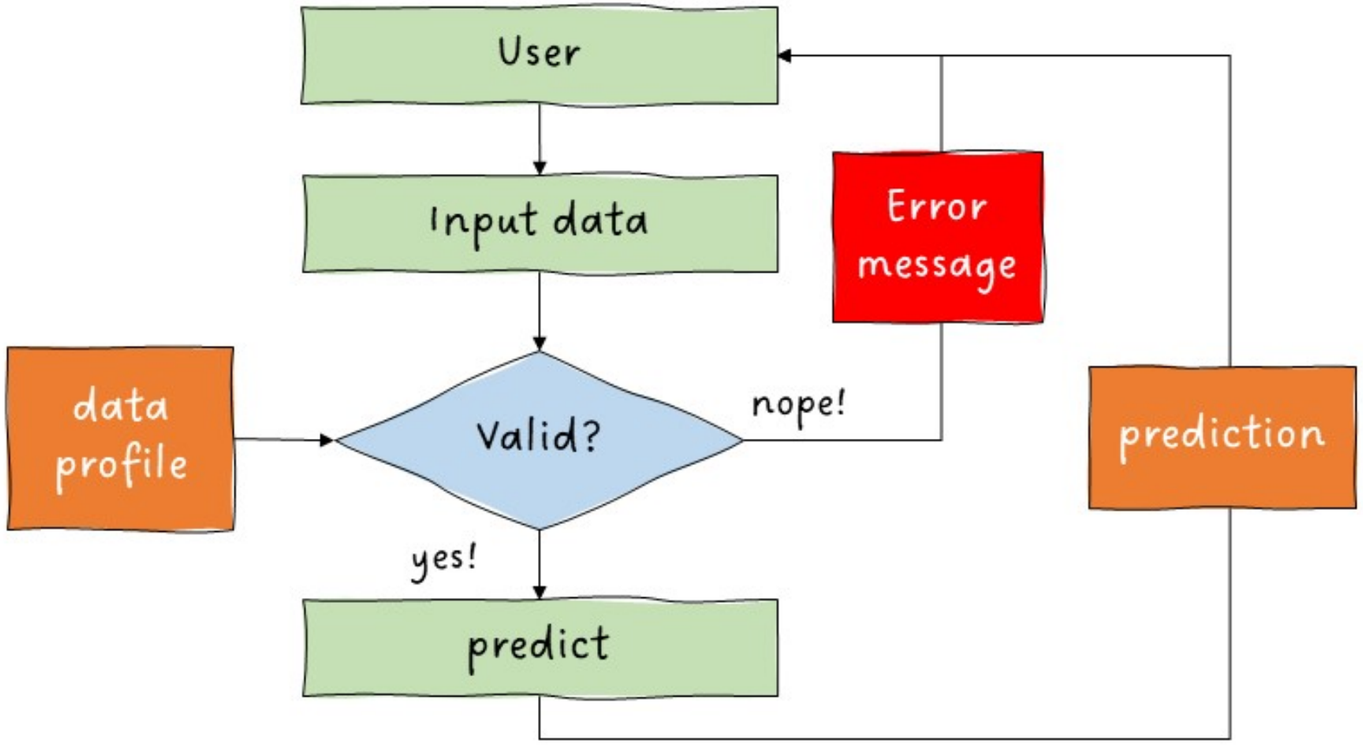


Concern: Performance deterioration

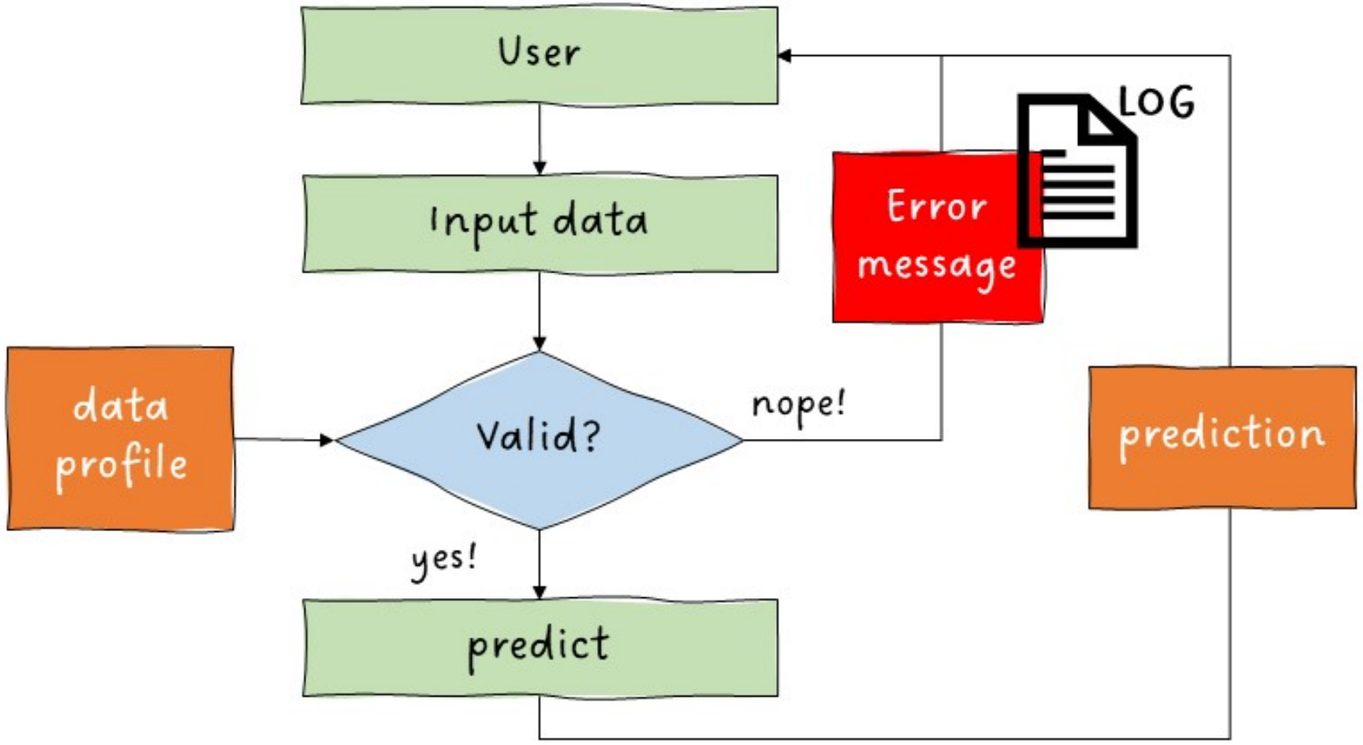


Minimum requirement: log inputs and outputs

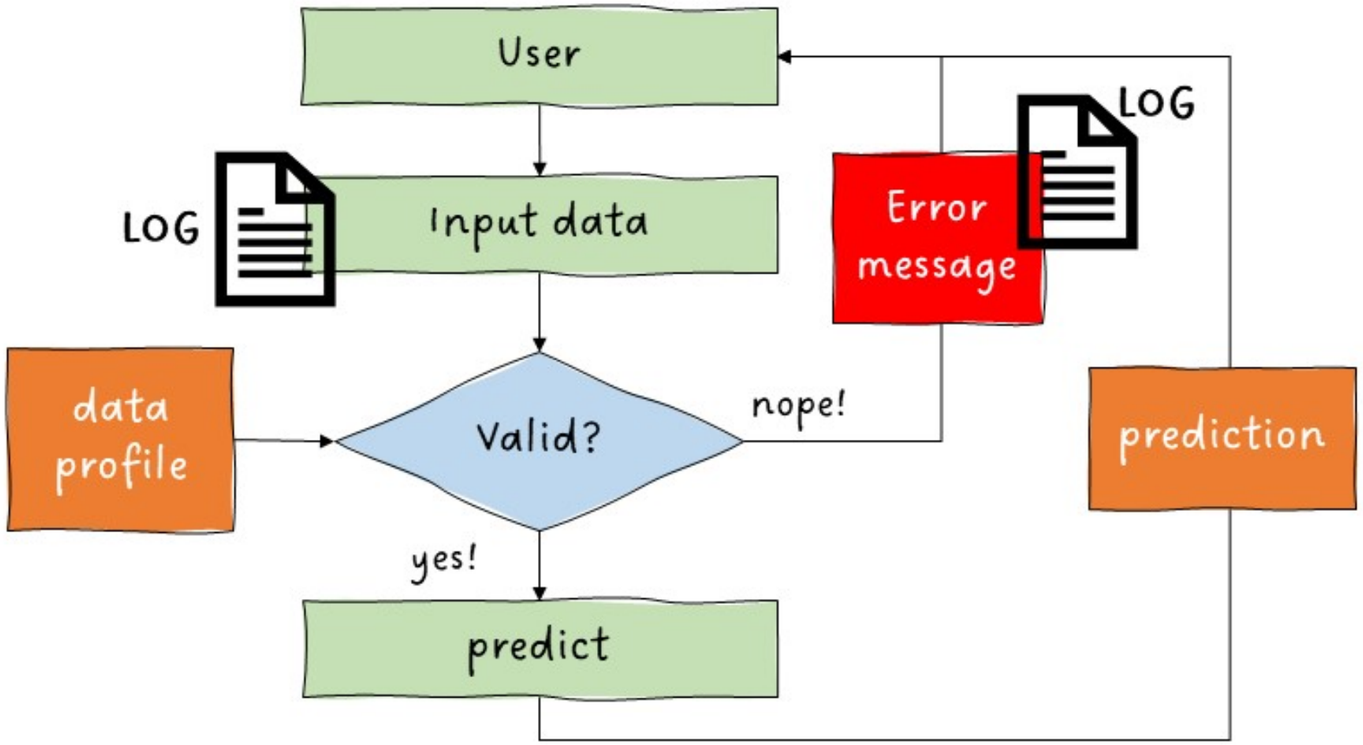
Concern: Debugging



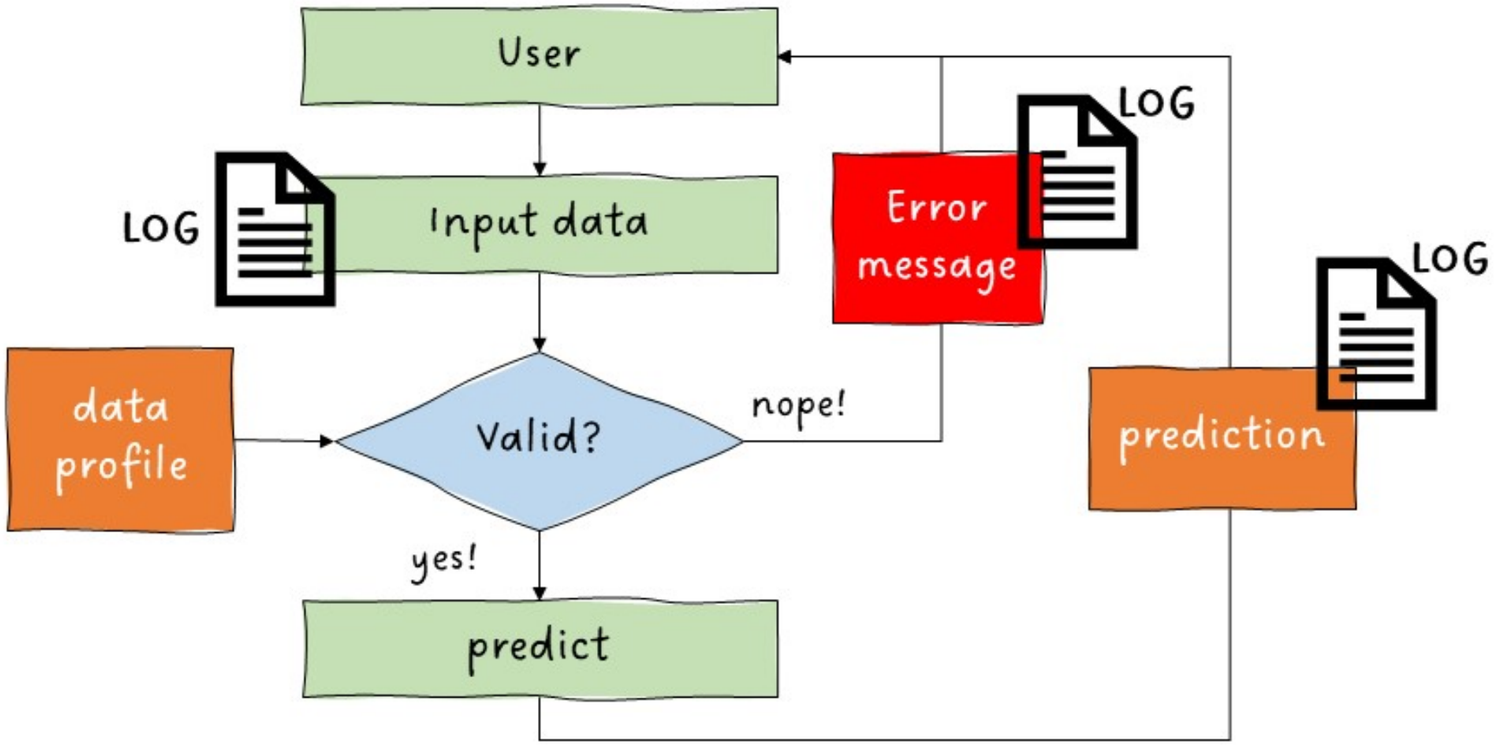
Concern: Debugging



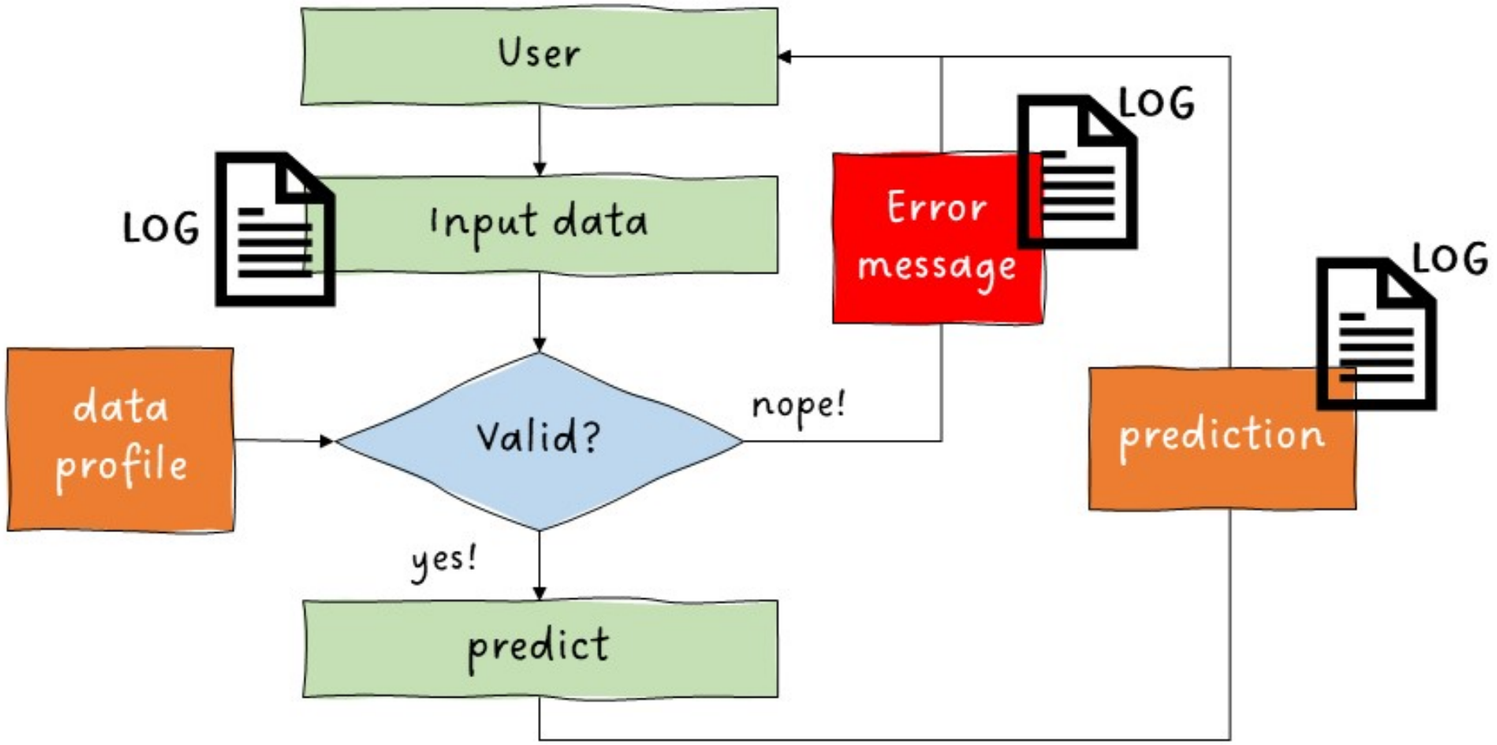
Concern: Debugging



Concern: Debugging



Concern: Debugging



Don't improvise, use purpose-built tools!

Concern: Am I comfortable making changes to this code?

Concern: Am I comfortable making changes to this code?

- unit tests
- integration tests
- load tests
- stress tests
- deployment tests


KEEP
CALM
AND
ML
OPS

Let's practice!

MLOPS DEPLOYMENT AND LIFE CYCLING

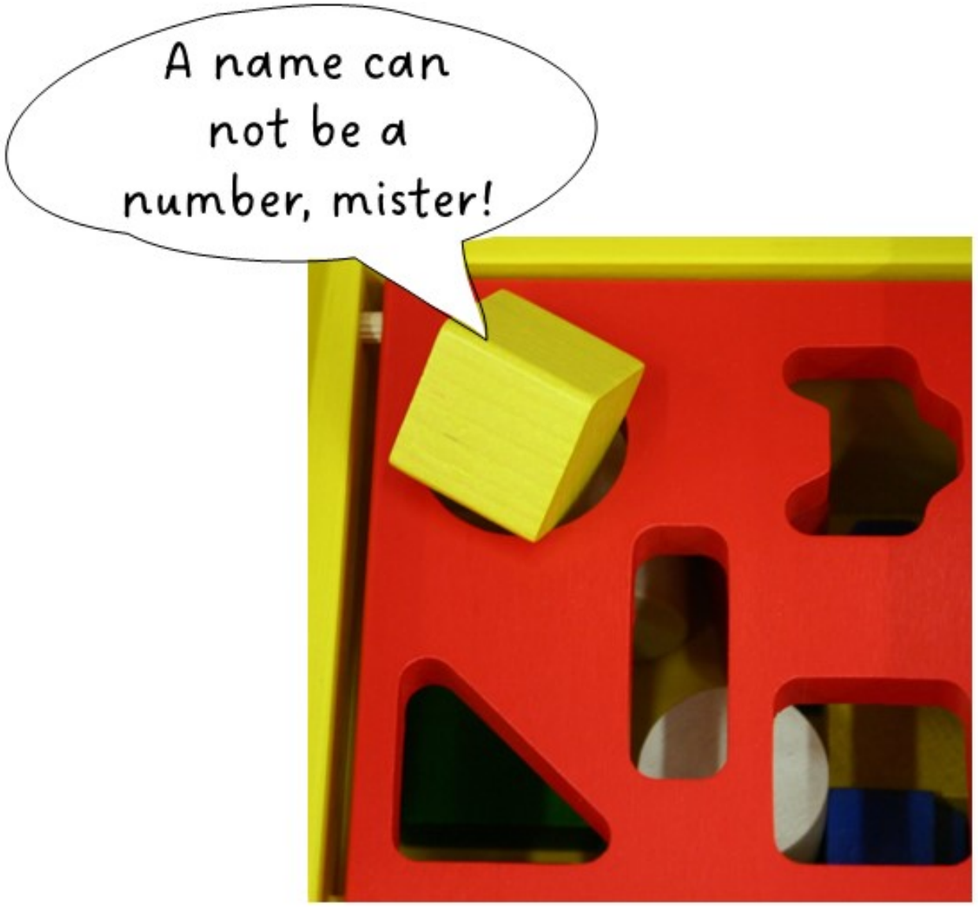
Profiling, versioning, and feature stores

MLOPS DEPLOYMENT AND LIFE CYCLING

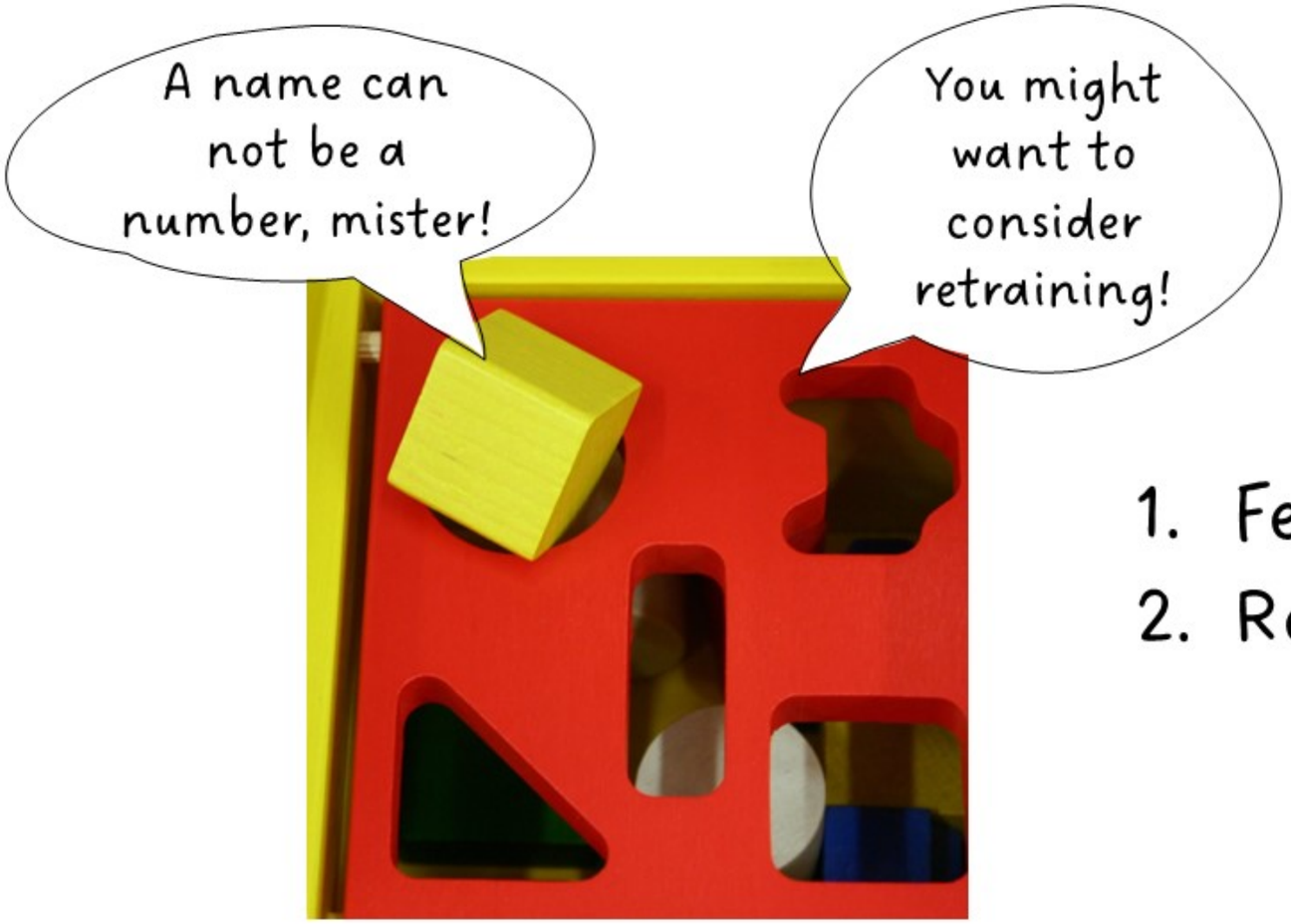
Nemanja Radojkovic
Senior Machine Learning Engineer

Data profiling

> Automated data analysis and creation of high-level summaries (a.k.a. data profiles, expectations), used for validating and monitoring data in production



1. Feedback to the user



1. Feedback to the user
2. Retraining decisions

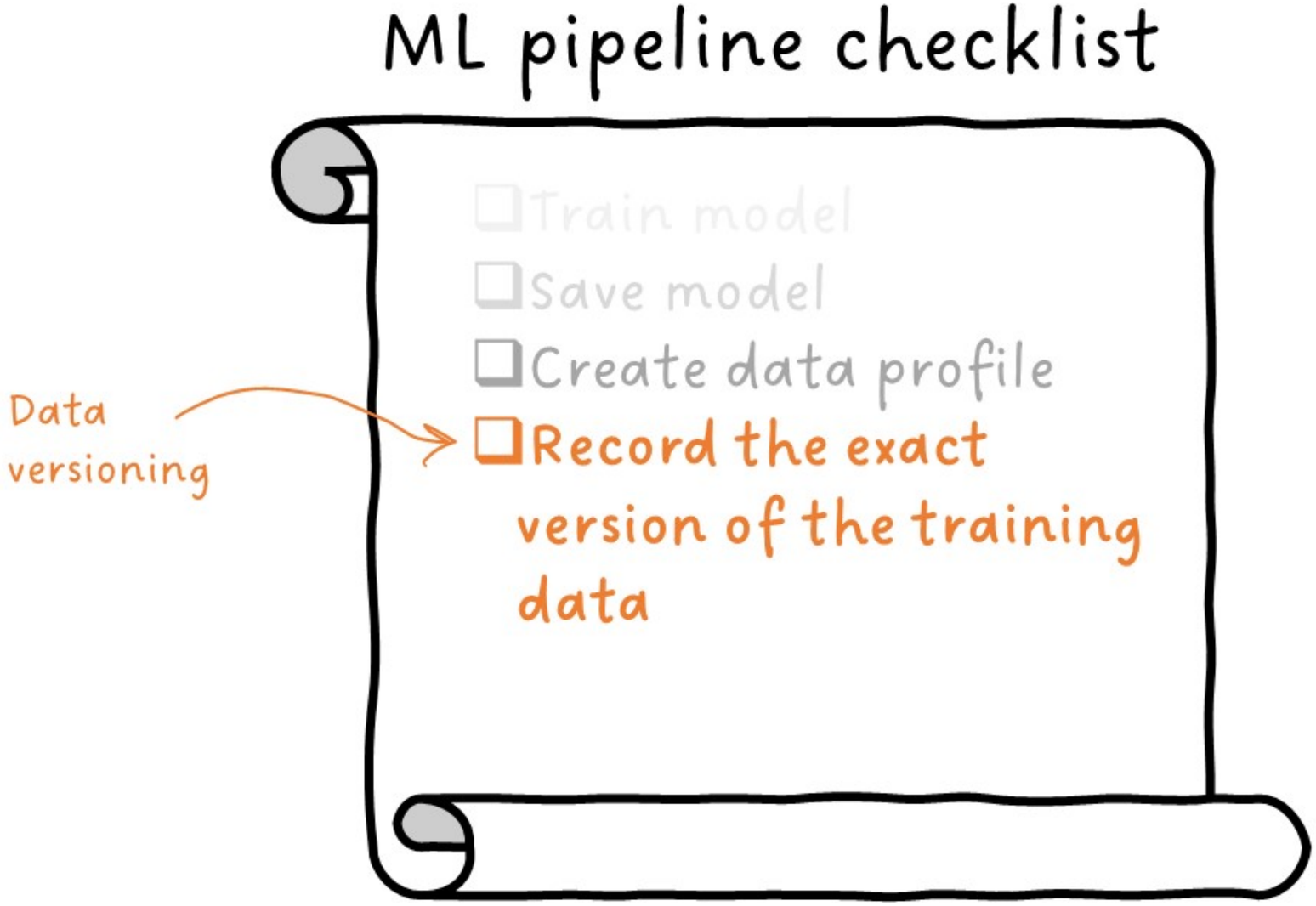
Risk of NOT using data profiles

- Clients complaining, although ***they*** submitted erroneous inputs to the model
- No way to identify that data has drifted and our model is no longer valid

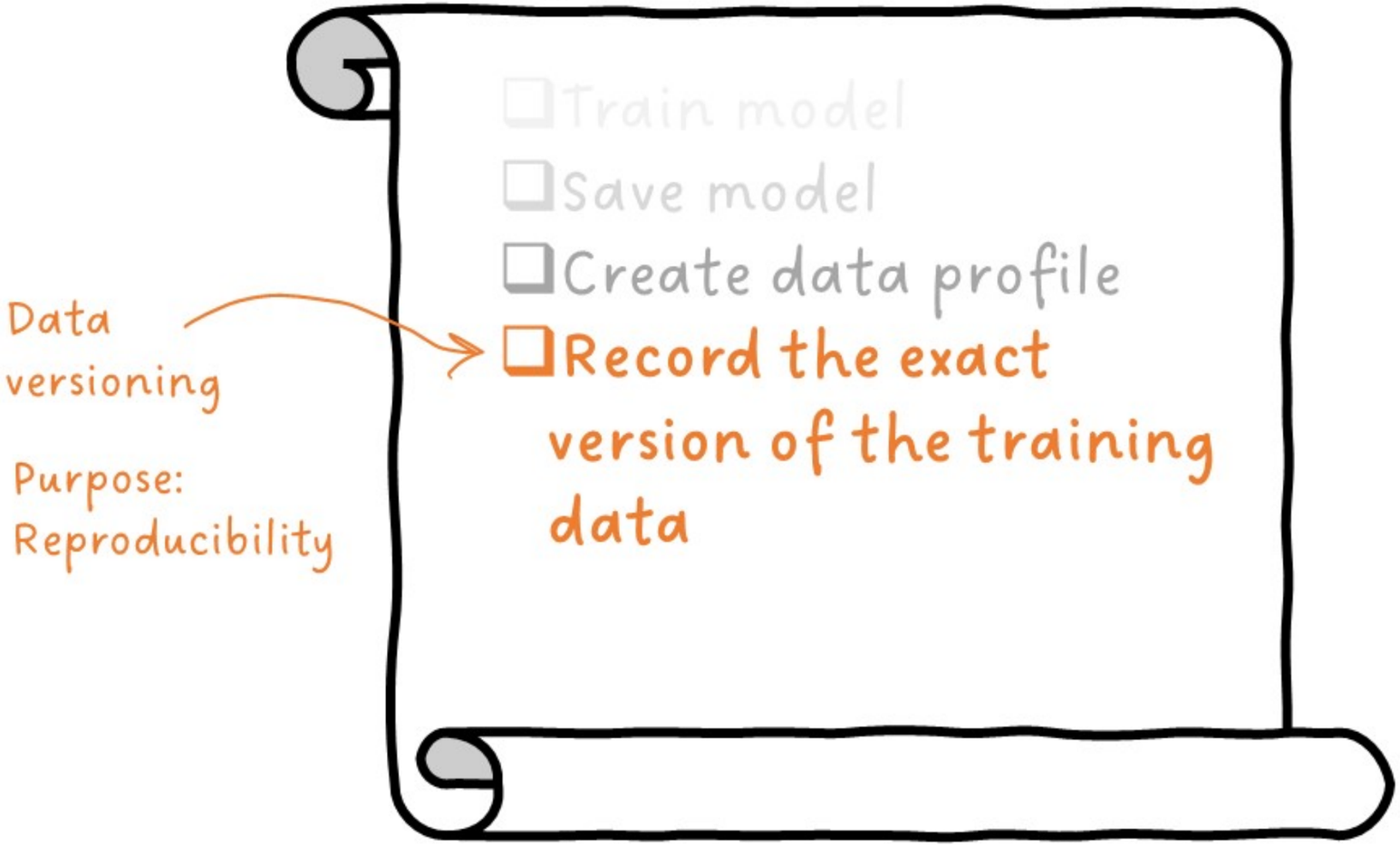
ML pipeline checklist

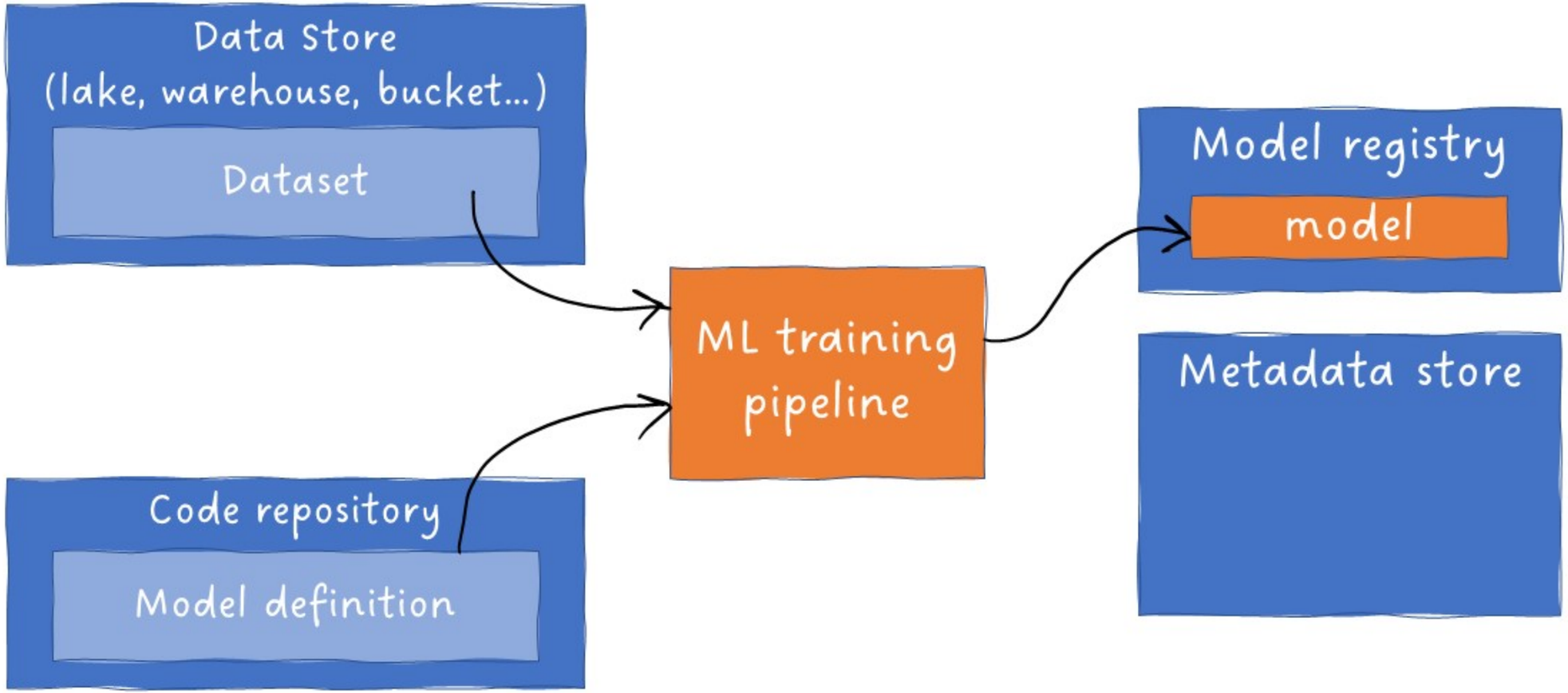
great_expectations

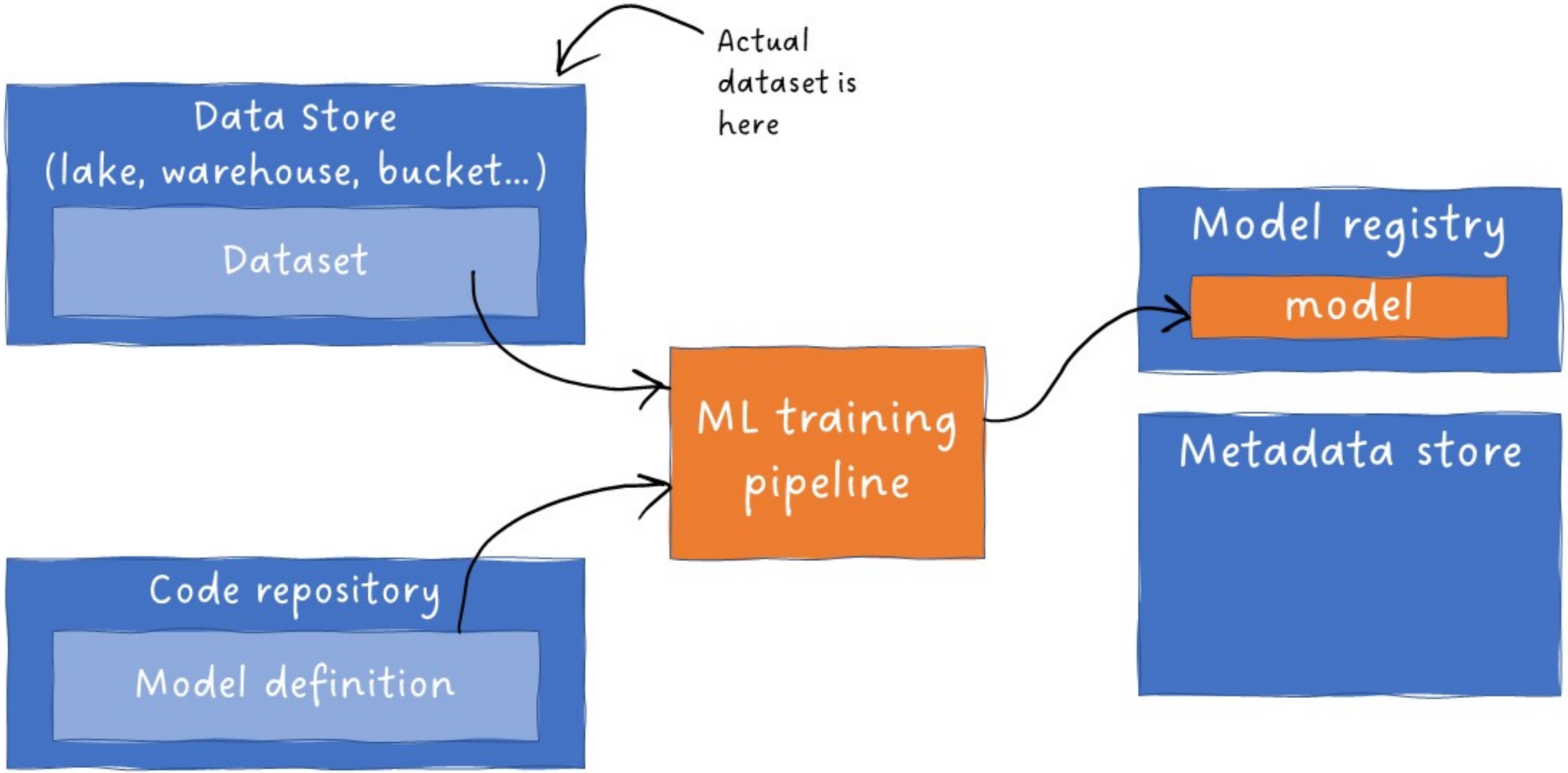
ML pipeline checklist

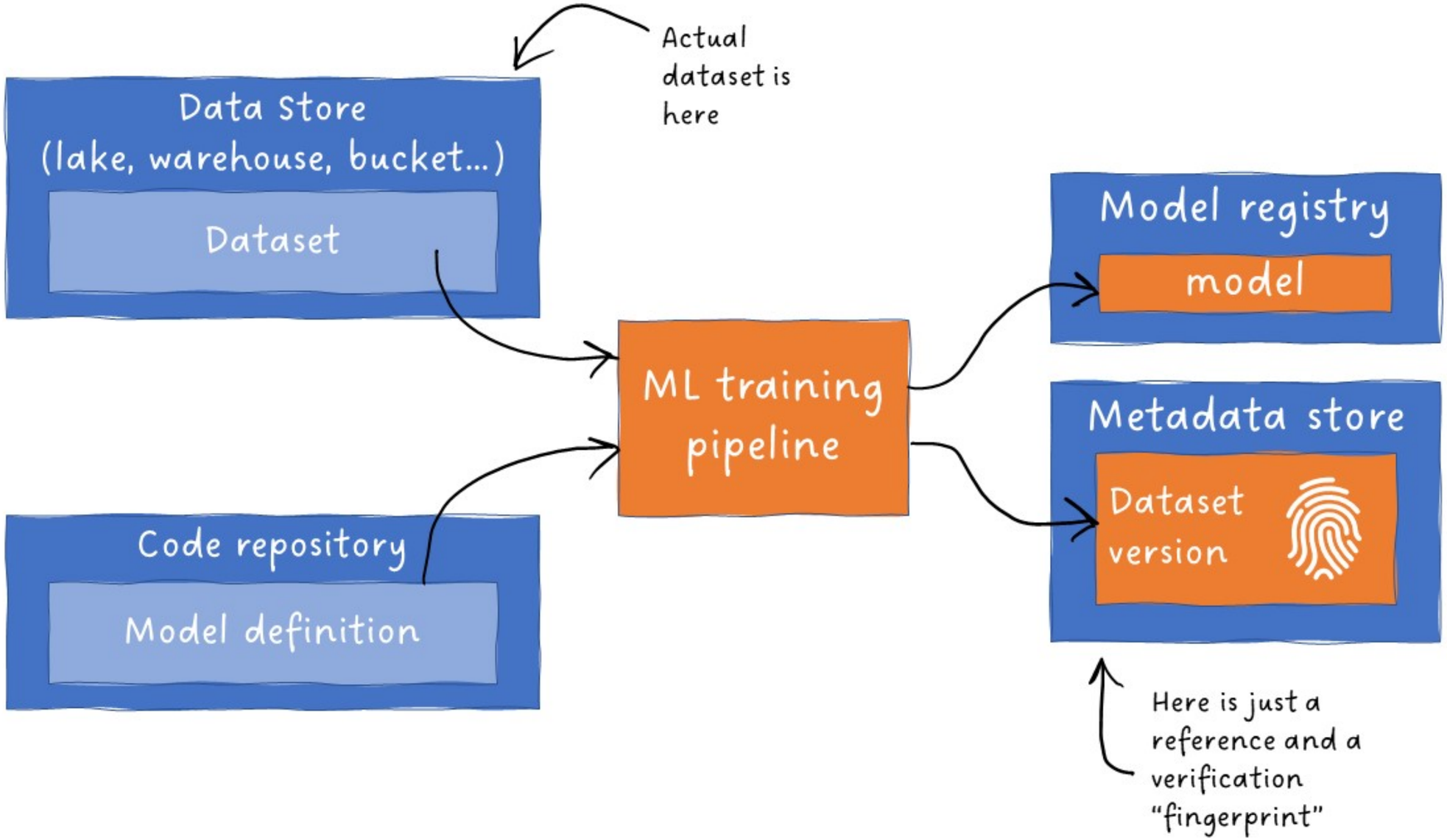

ML pipeline checklist
Train model
Save model
Create data profile
Record the exact version of the training data
Data versioning

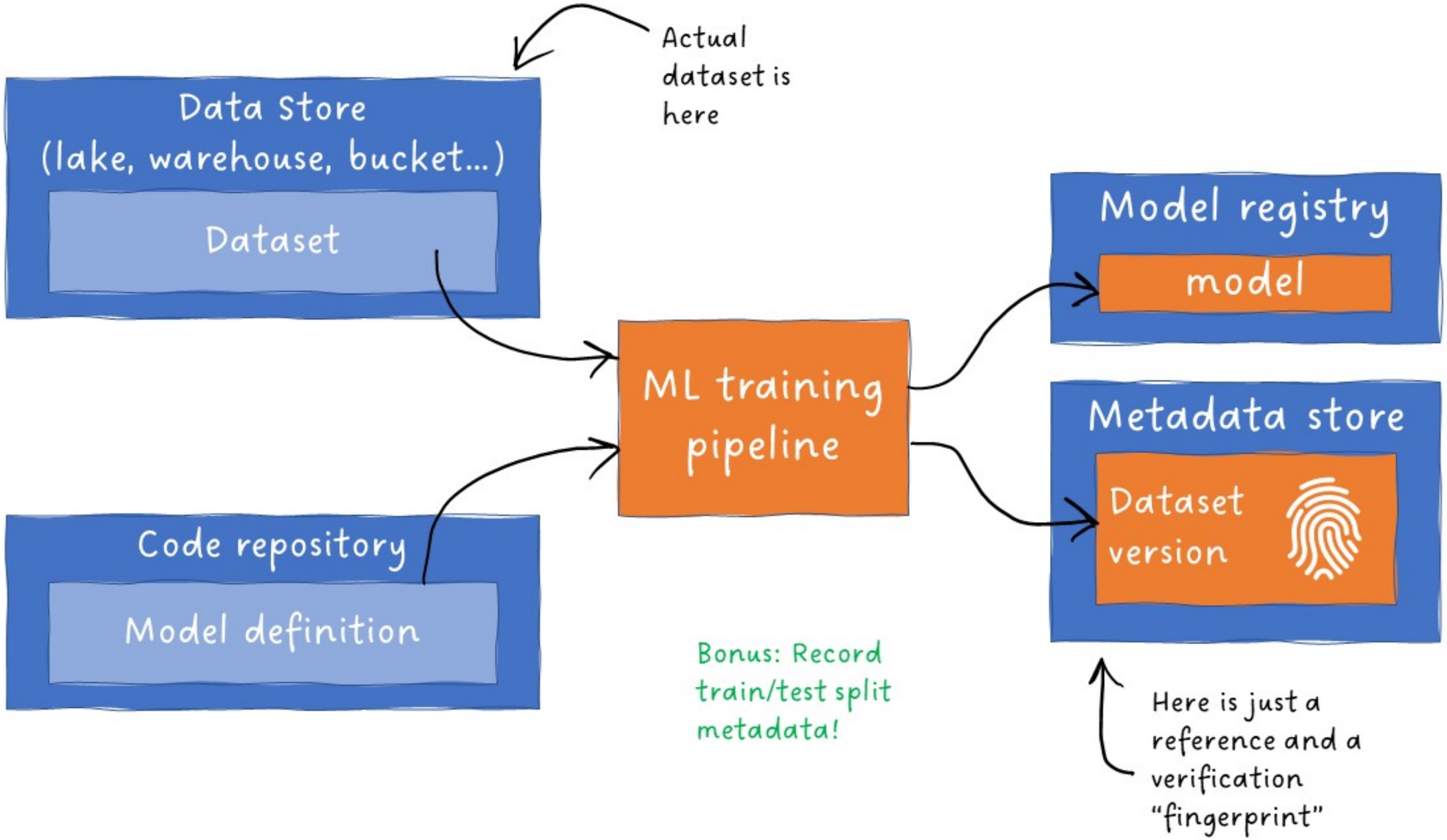
ML pipeline checklist



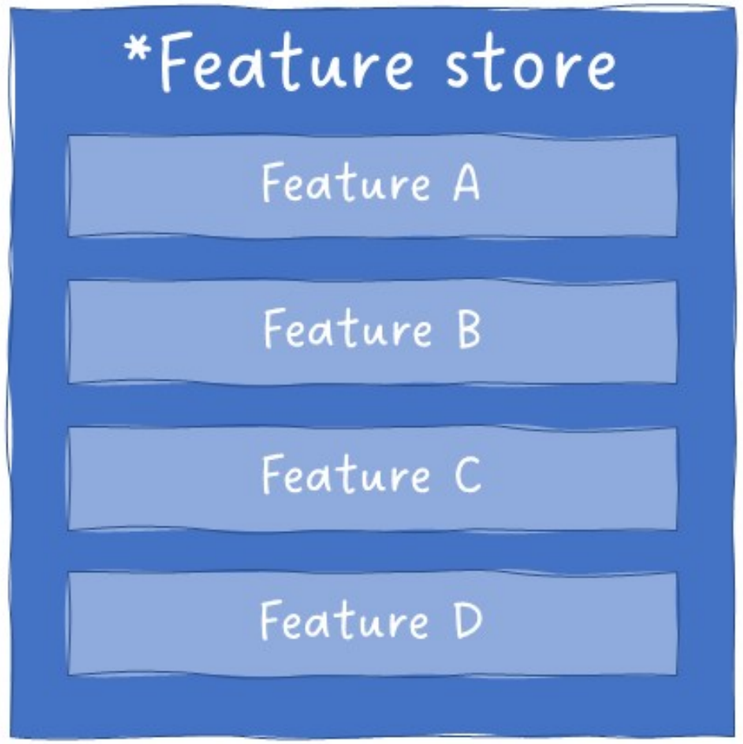

Data Store
(lake, warehouse, bucket...)
Dataset
Code repository
Model definition
ML training
pipeline
Model registry
model
Metadata store

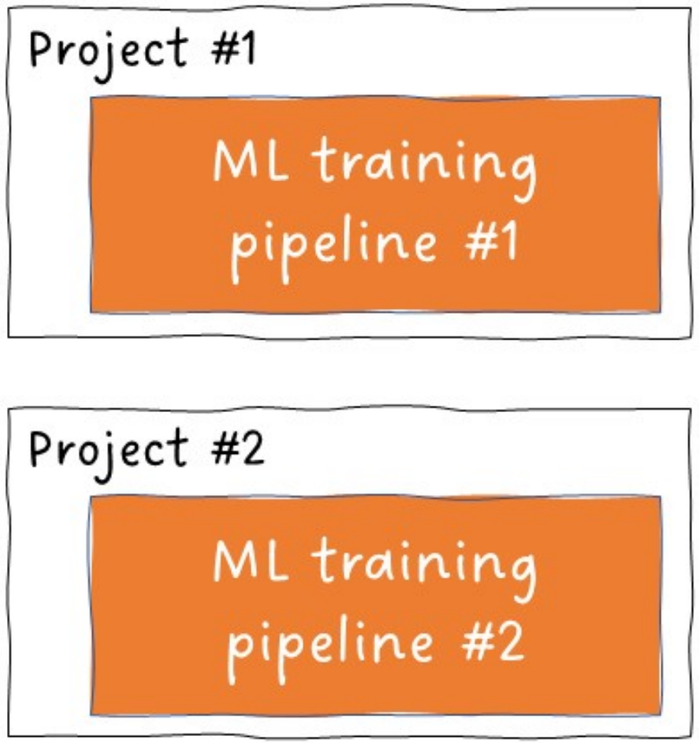

Actual dataset is here
Data Store (lake, warehouse, bucket...)
Dataset
Model registry
model
ML training pipeline
Metadata store
Code repository
Model definition

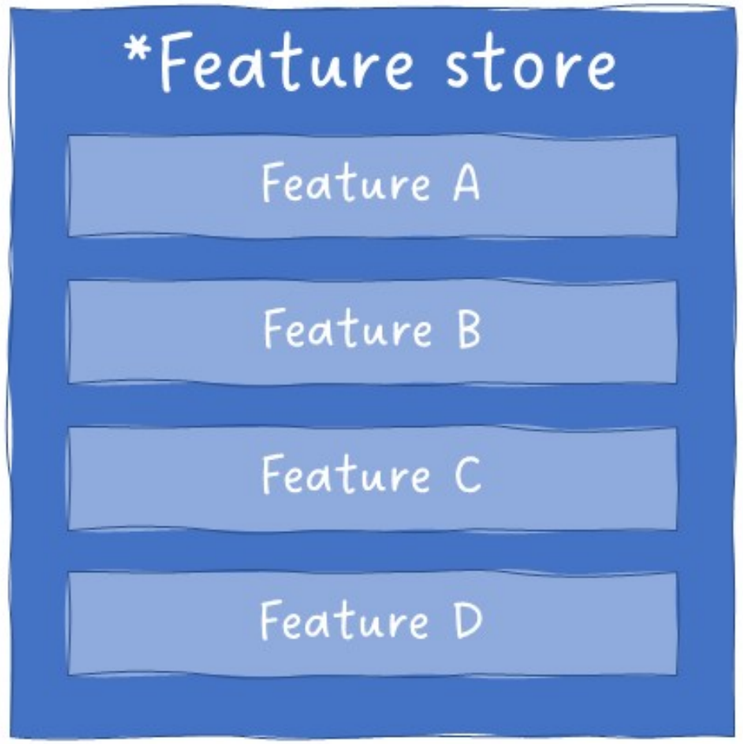

Actual dataset is here
Data Store (lake, warehouse, bucket...)
Dataset
Model registry
model
ML training pipeline
Metadata store
Dataset version
Code repository
Model definition
Here is just a reference and a verification "fingerprint"


Actual dataset is here
Data Store (lake, warehouse, bucket...)
Dataset
Model registry
model
ML training pipeline
Metadata store
Dataset version
Code repository
Model definition
Bonus: Record train/test split metadata!
Here is just a reference and a verification "fingerprint"

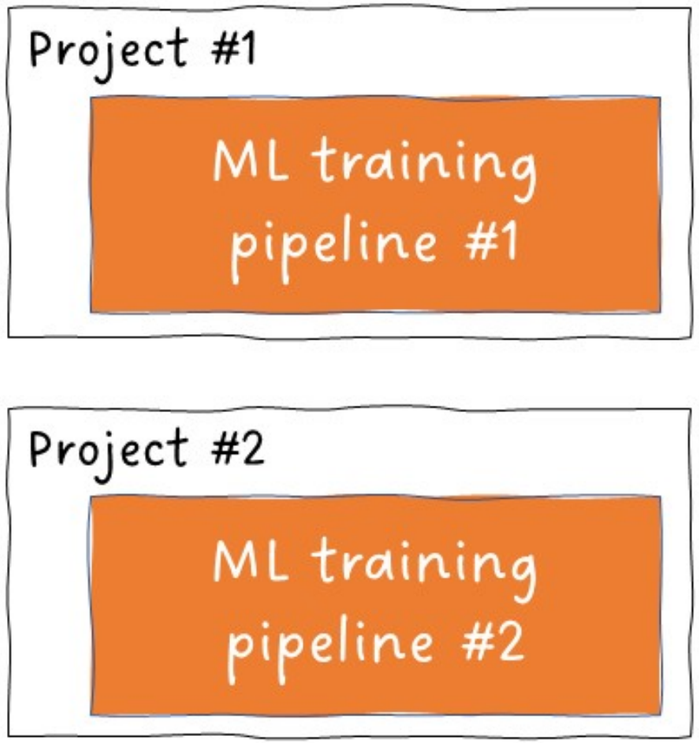
- Data
- Version
- Control

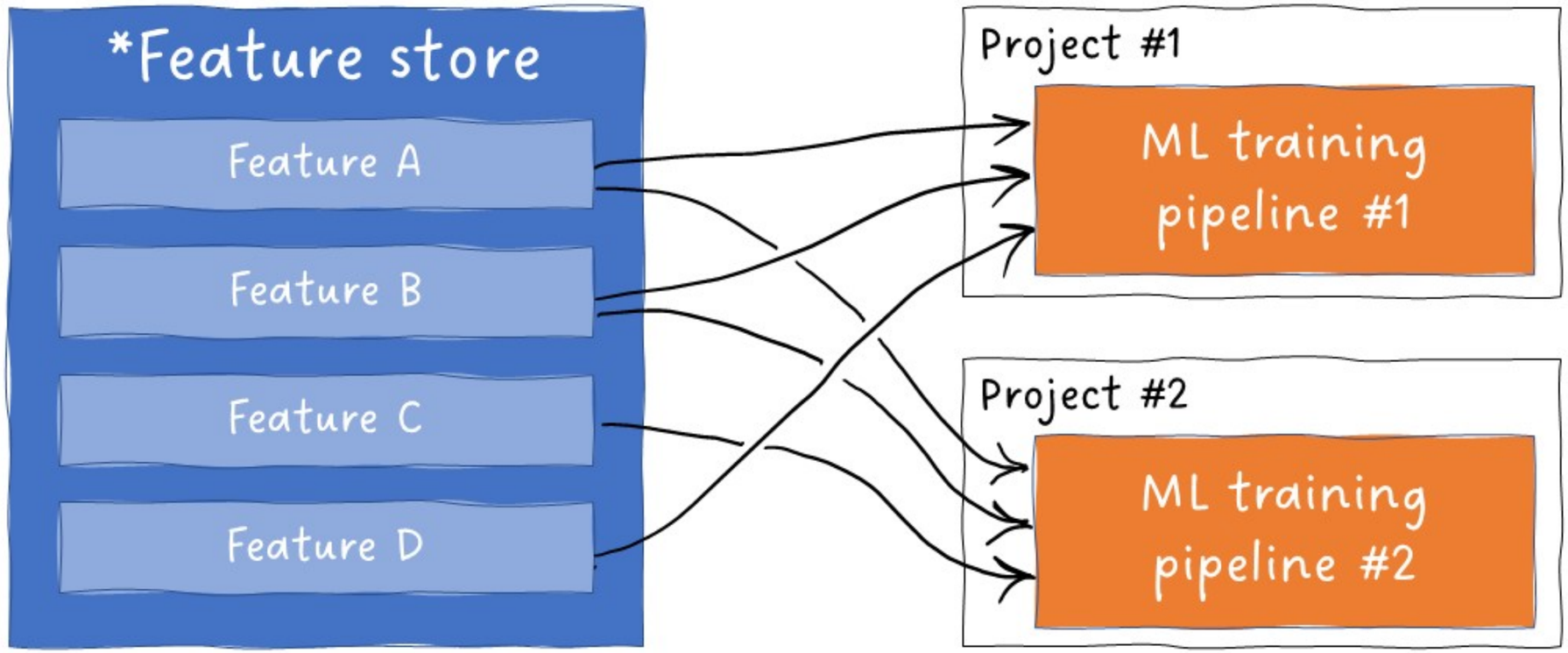

*Feature store
Feature A
Feature B
Feature C
Feature D


Project #1
ML training pipeline #1
Project #2
ML training pipeline #2

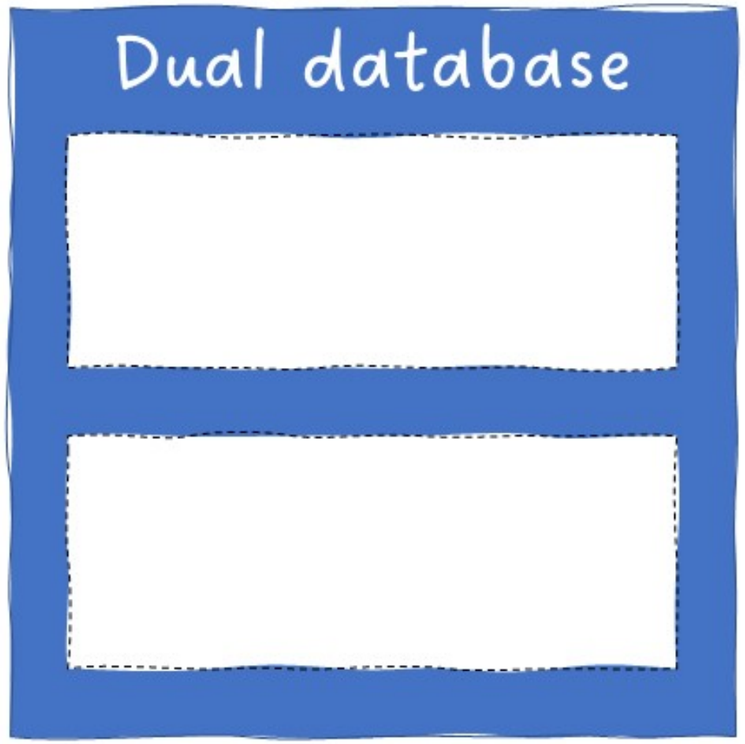


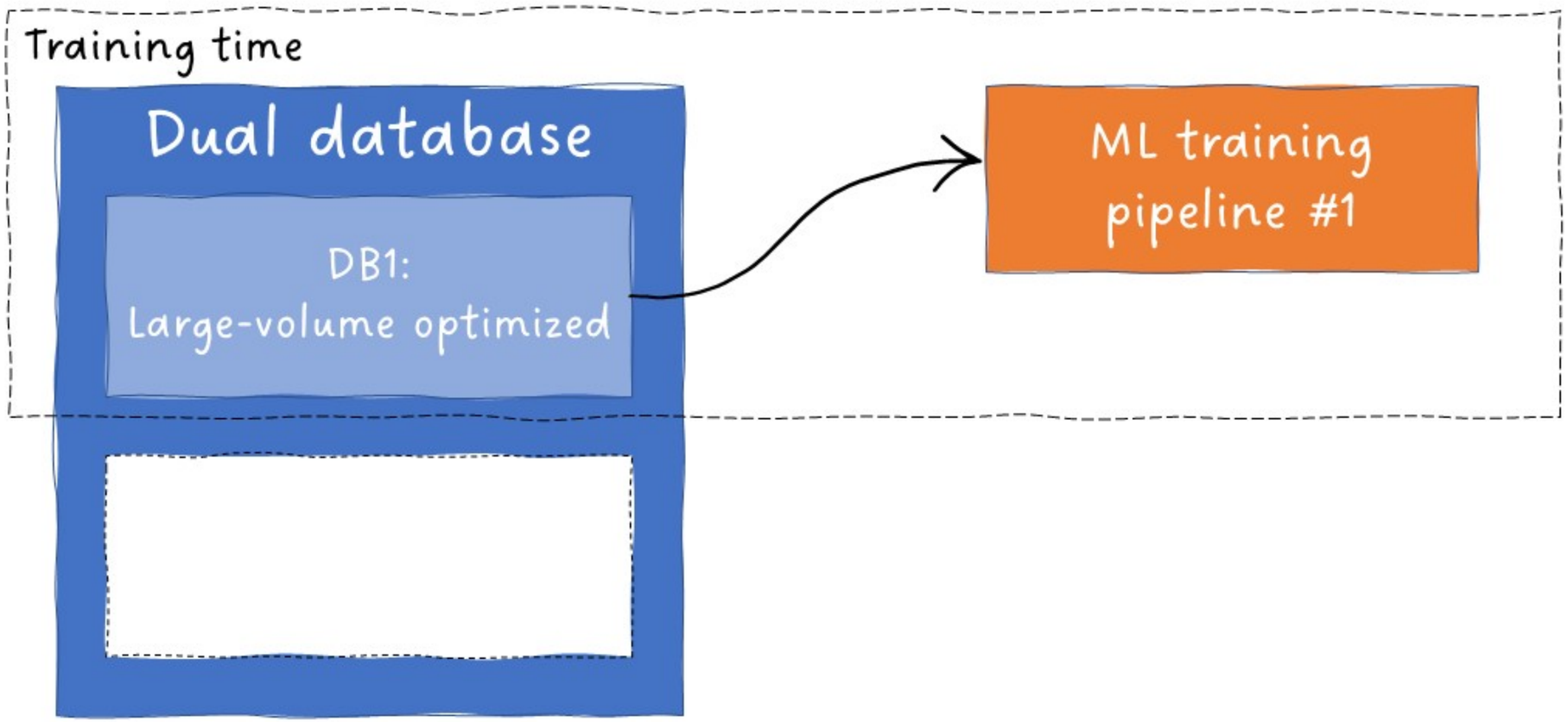
* Essentially a database

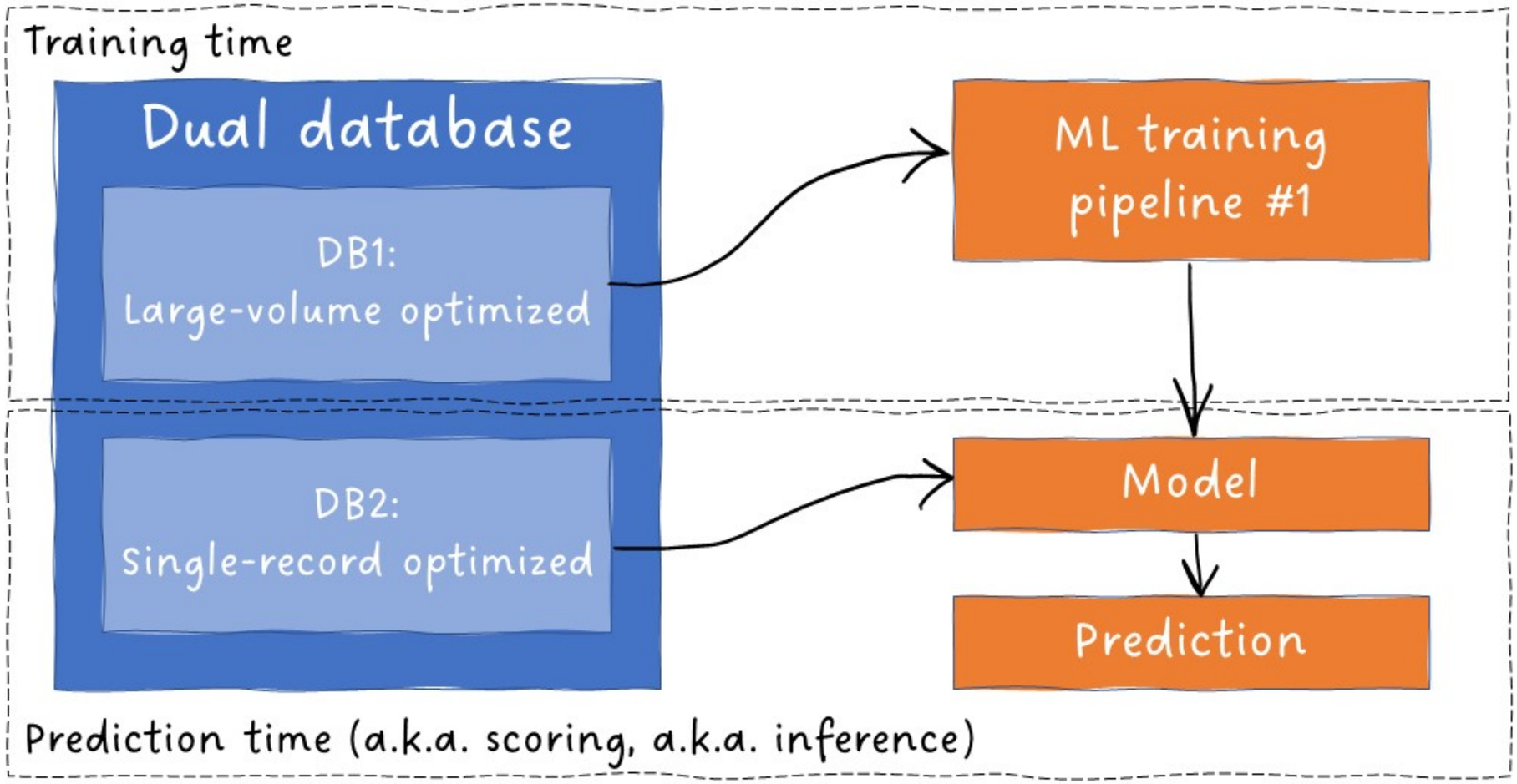




* Essentially a database


Dual database


Training time
Dual database
DB1:
Large-volume optimized
ML training
pipeline #1


Training time
Dual database
DB1:
Large-volume optimized
DB2:
Single-record optimized
ML training
pipeline #1
Model
Prediction
Prediction time (a.k.a. scoring, a.k.a. inference)

Feature store benefits

Reusability

Feature store benefits

Reusability

Consistency
(lower train/serve skew)

Feature store benefits

Reusability

Consistency
(lower train/serve skew)

Email in training

```
Lorem ipsum dolor sit amet, consectetur adi
eiusmod tempor incididunt ut labore et dolo
enim ad minim veniam, quis nostrud exercita
nisi ut aliquip ex ea commodo consequat. Du
reprehenderit in voluptate velit esse cillu
nulla pariatur. Excepteur sint occaecat cup
sunt in culpa qui officia deserunt mollit a
Sed ut perspiciatis unde omnis iste natus e
accusantium doloremque laudantium, totam re
quae ab illo inventore veritatis et quasi a
```

Email for prediction

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Title goes here</title>
  </head>
  <body>

  </body>
</html>
```

Let's practice!

MLOPS DEPLOYMENT AND LIFE CYCLING

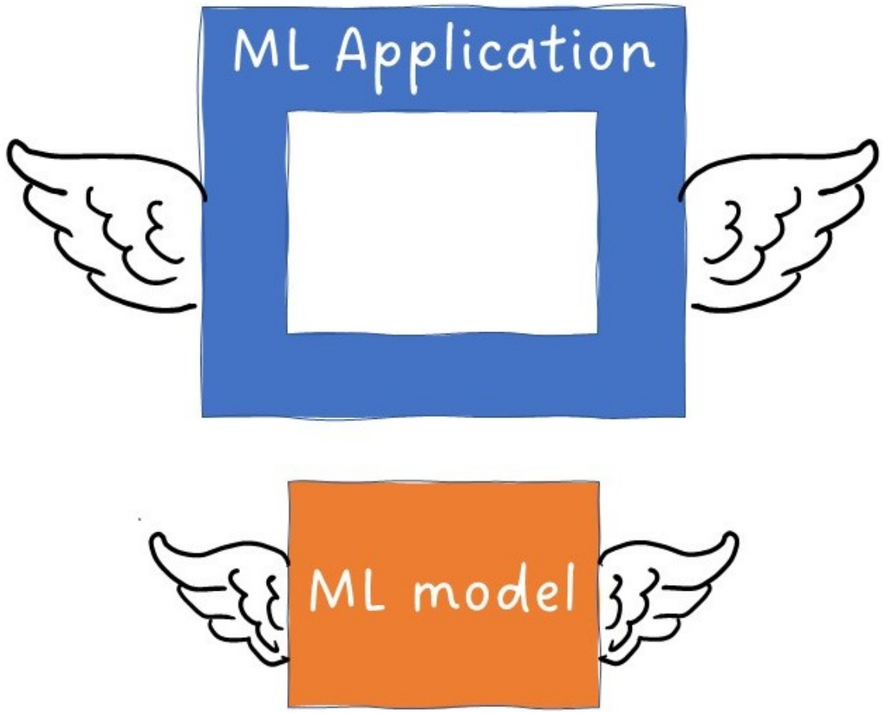
Model build pipelines in CI/CD

MLOPS DEPLOYMENT AND LIFE CYCLING

Nemanja Radojkovic
Senior Machine Learning Engineer

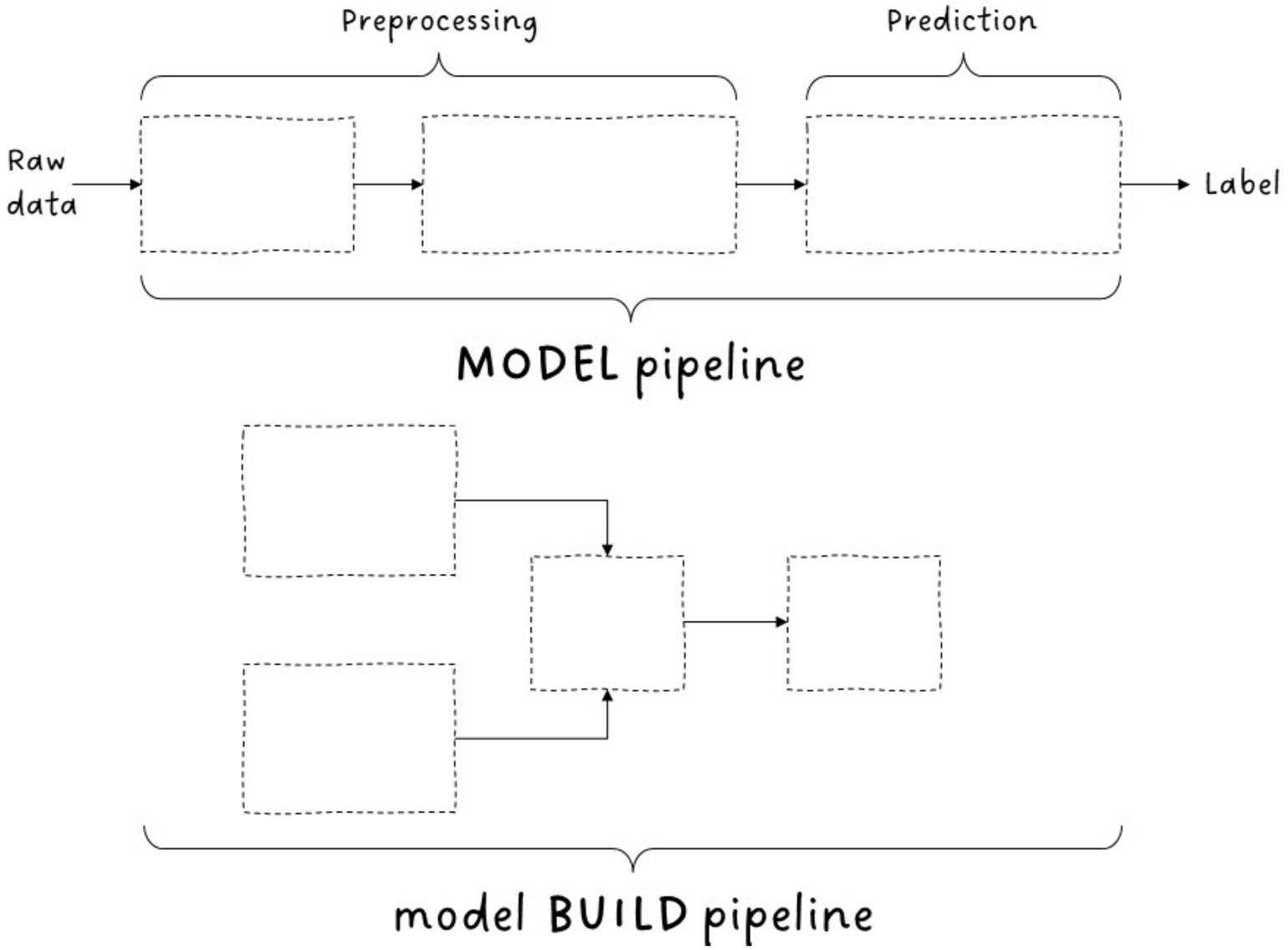
datacamp

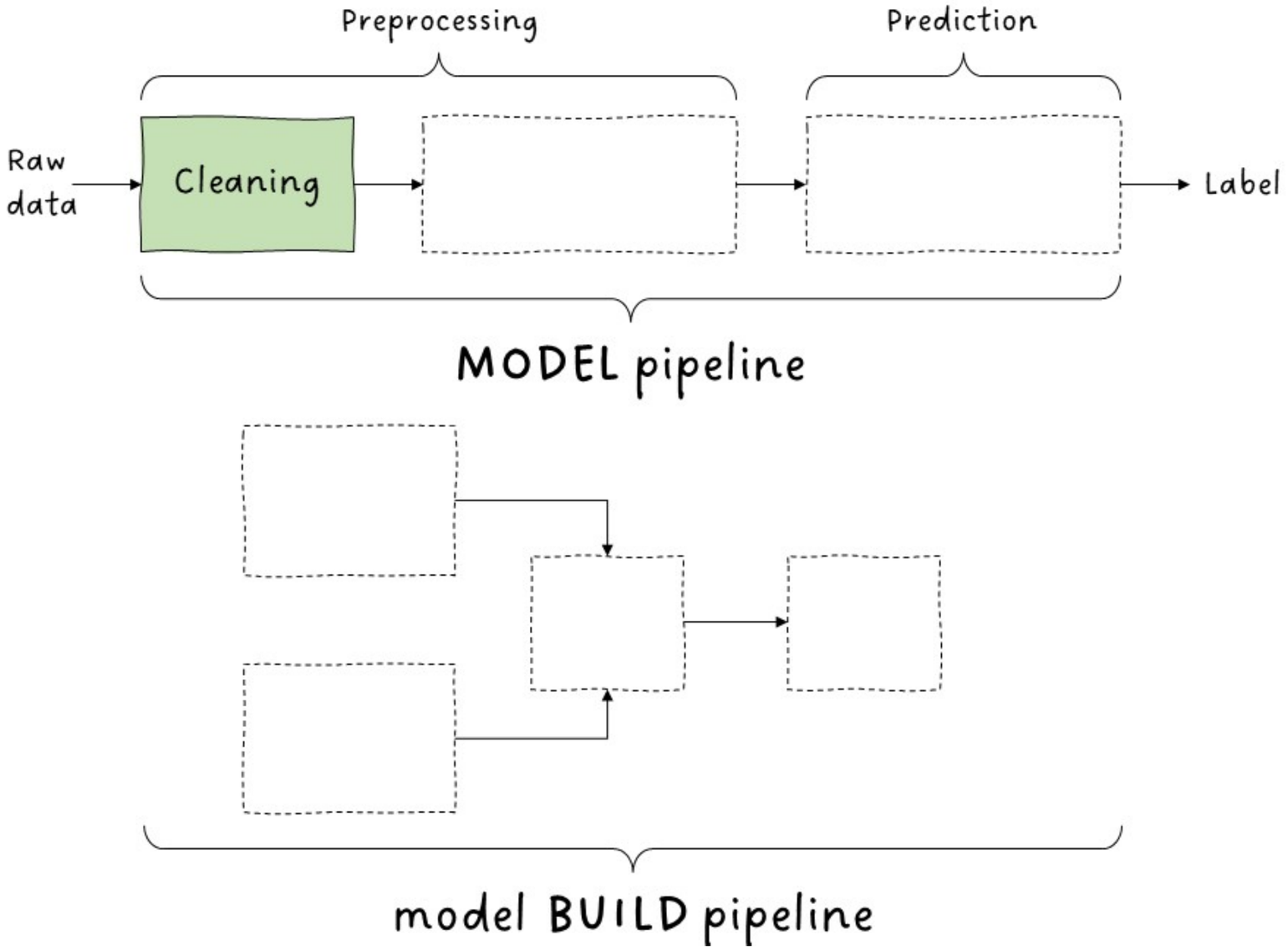
Two build pipelines in ML

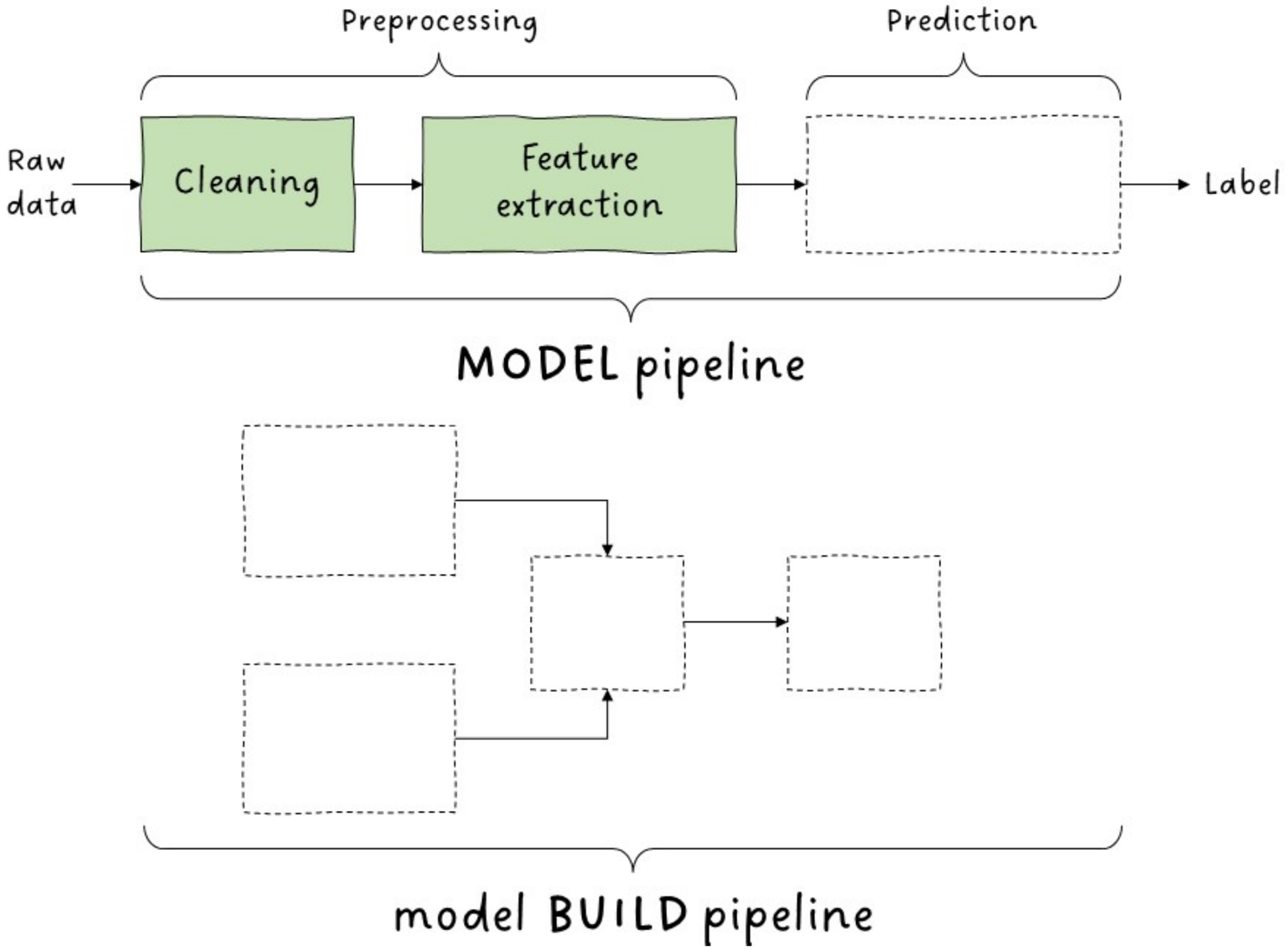


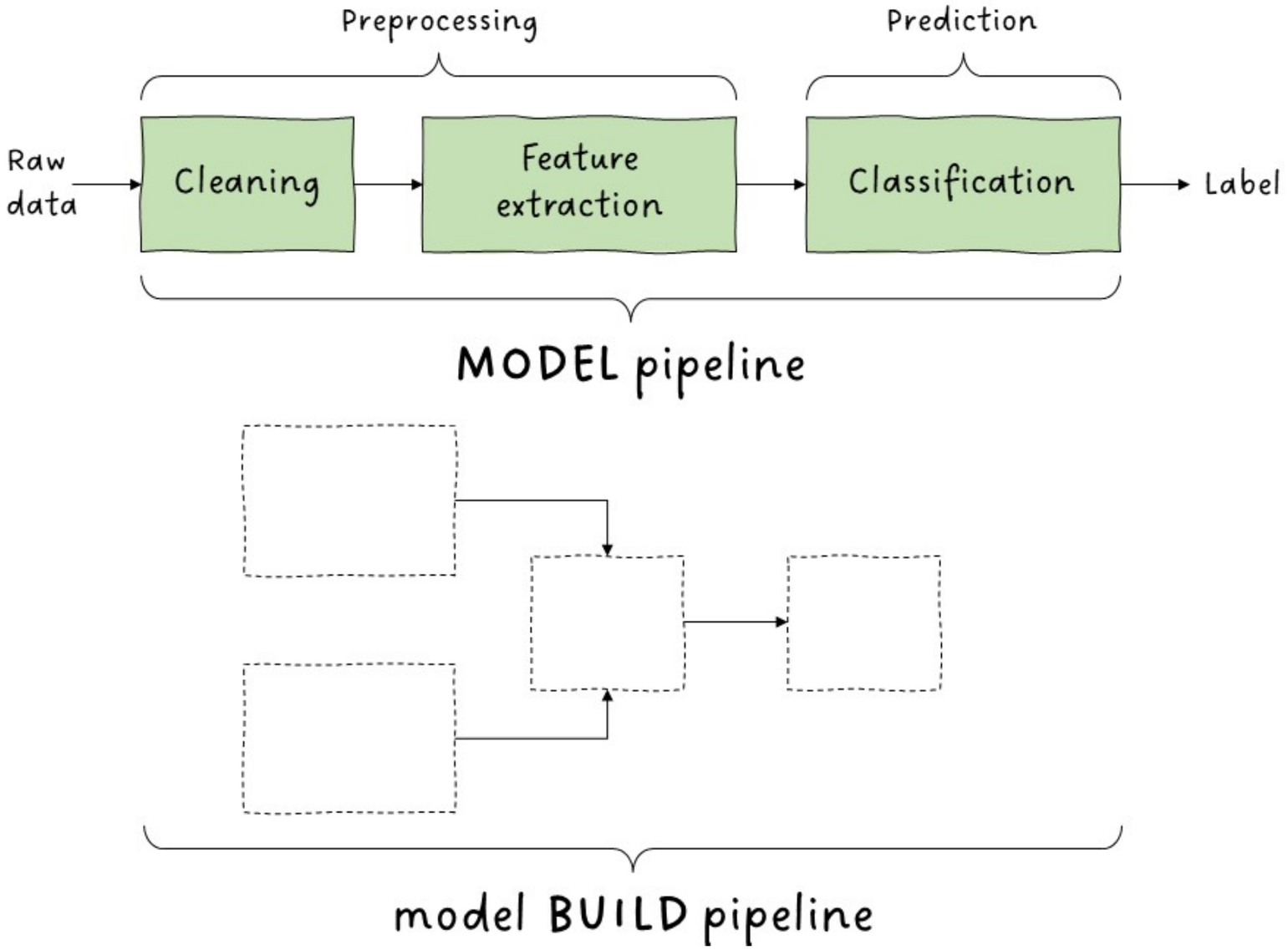
- One that builds the app that serves our model.
  - Standard software build pipeline
- One that builds (trains) the model itself
  - Central to the MLOps framework

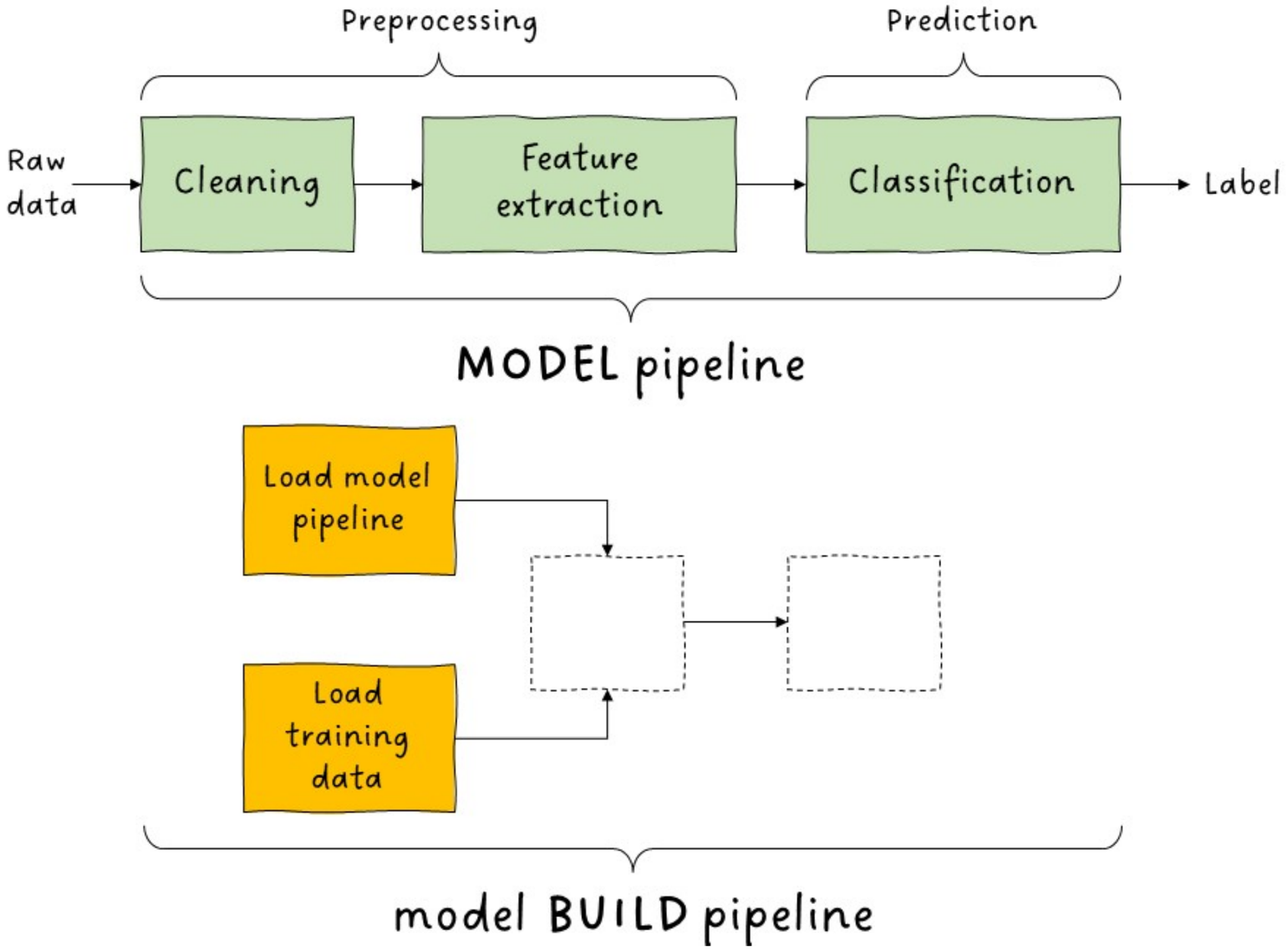
MODEL pipeline =/= model BUILD pipeline

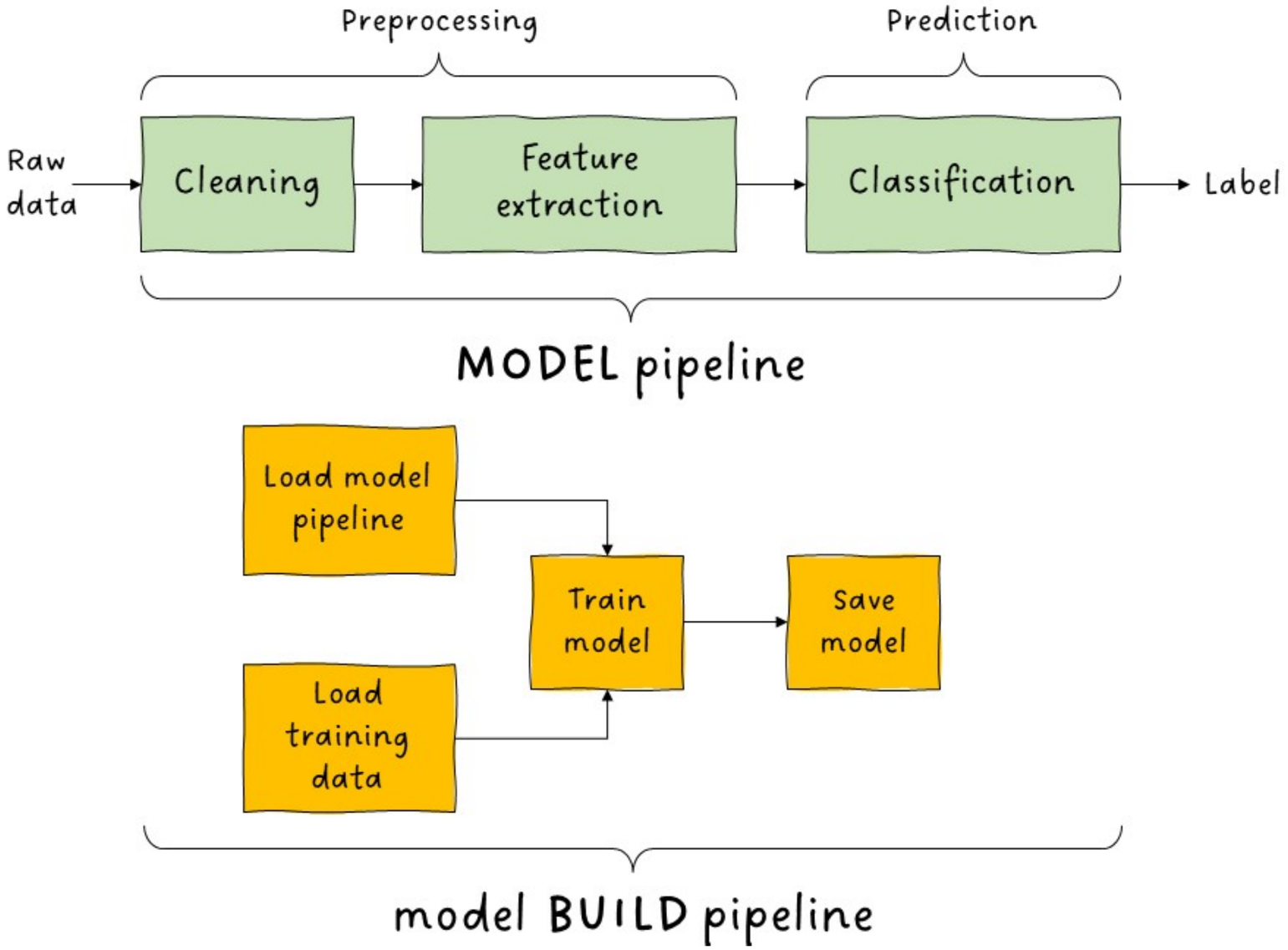

Preprocessing
Prediction
Raw data
Label
MODEL pipeline
model BUILD pipeline

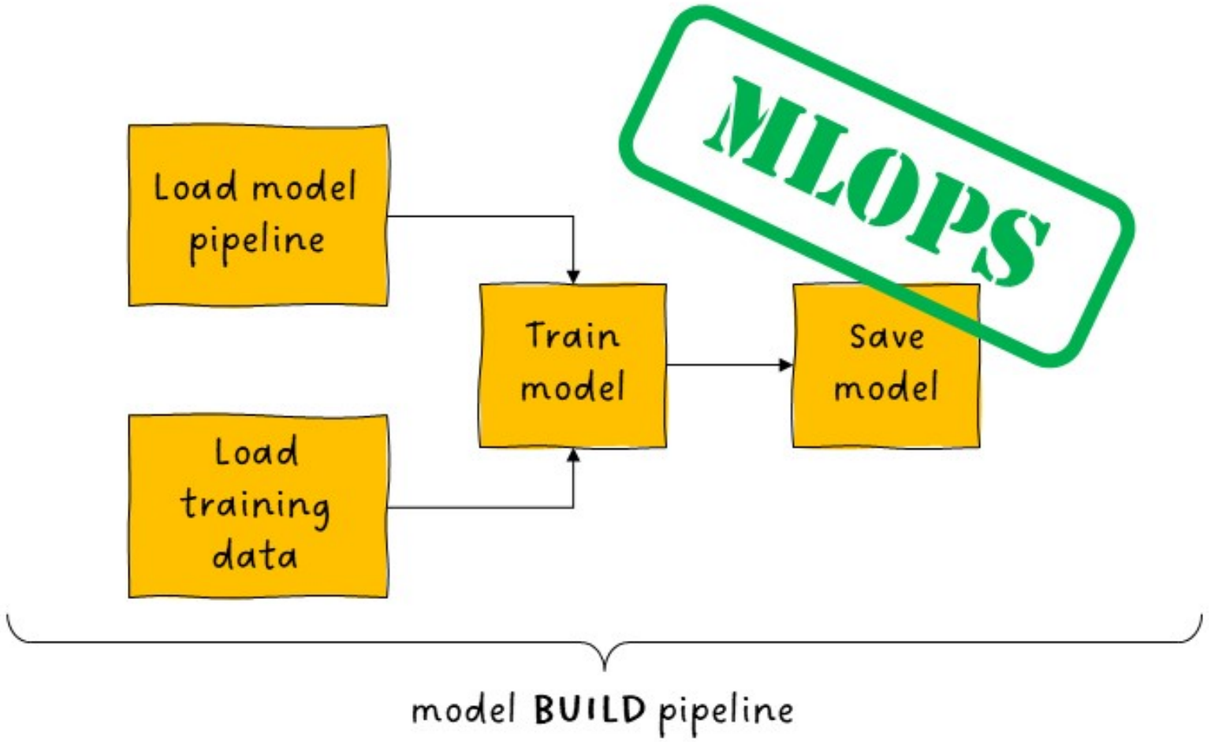

Preprocessing
Prediction
Raw data
Cleaning
Label
MODEL pipeline
model BUILD pipeline


Preprocessing
Prediction
Raw data
Cleaning
Feature extraction
Label
MODEL pipeline
model BUILD pipeline

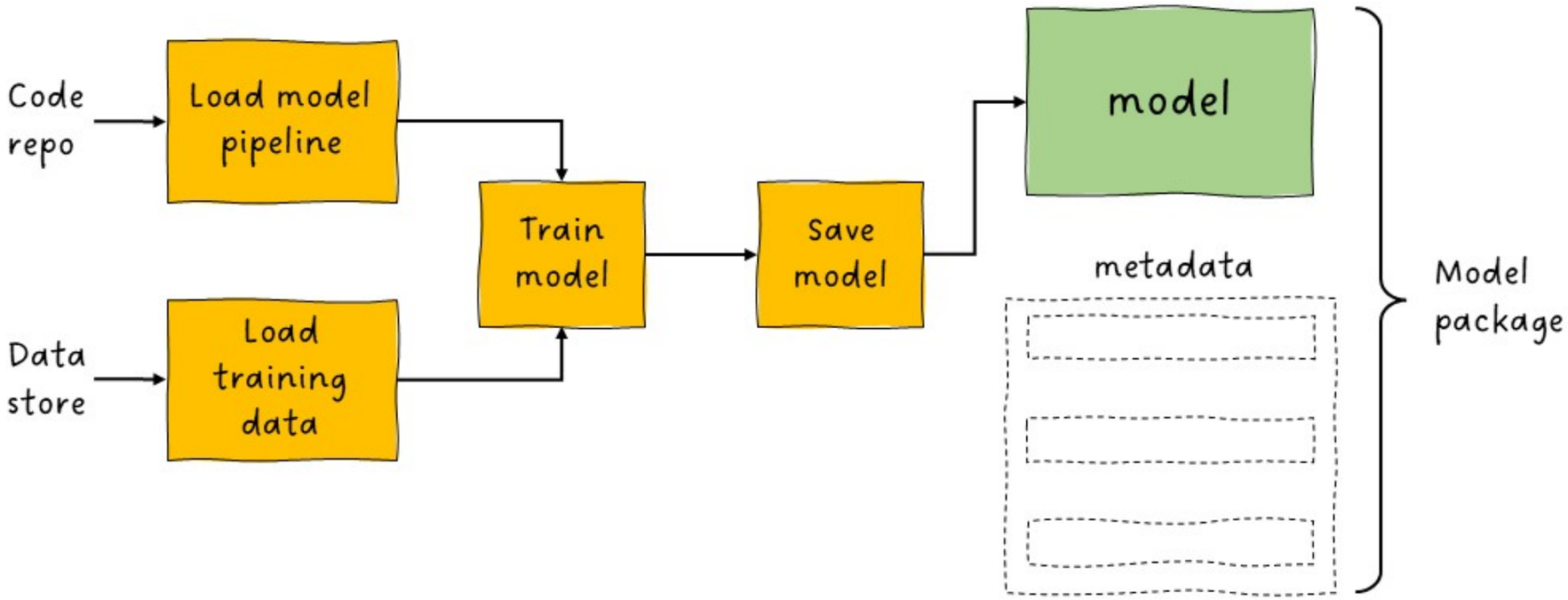

Preprocessing
Prediction
Raw data
Cleaning
Feature extraction
Classification
Label
MODEL pipeline
model BUILD pipeline

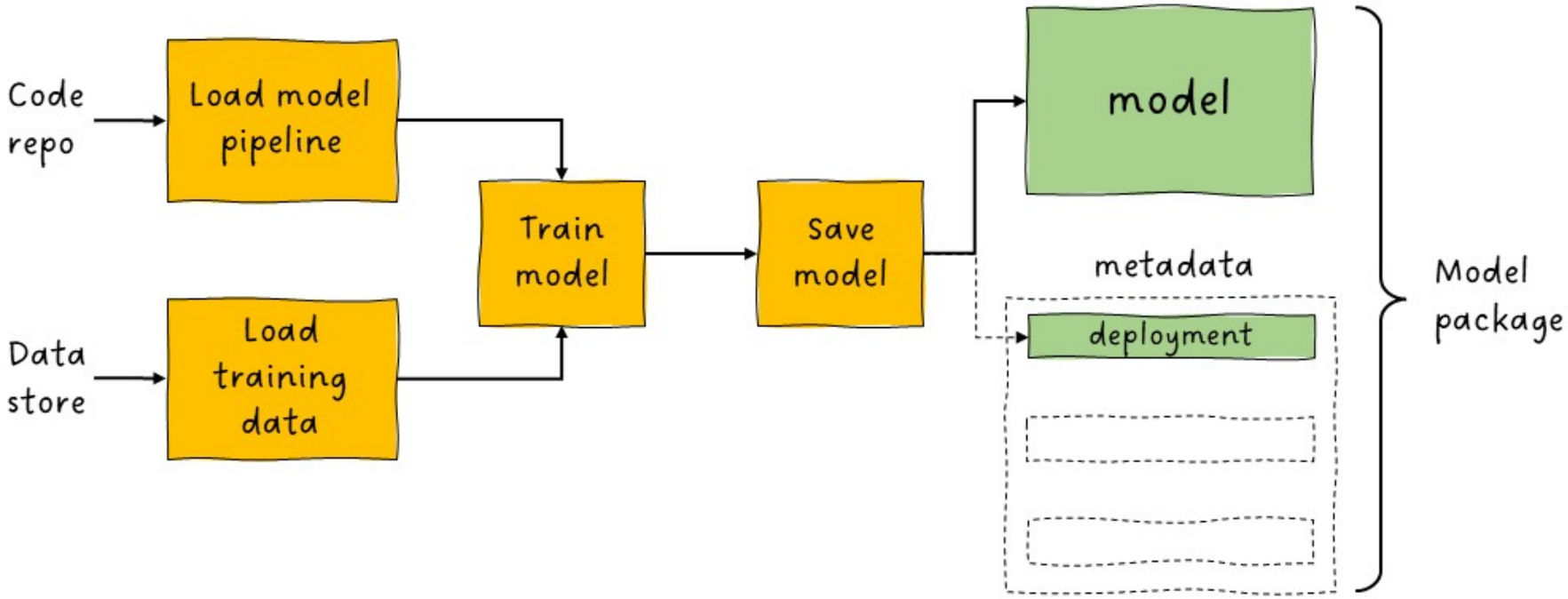

Preprocessing
Prediction
Raw data
Cleaning
Feature extraction
Classification
Label
MODEL pipeline
Load model pipeline
Load training data
model BUILD pipeline


Preprocessing
Prediction
Raw data
Cleaning
Feature extraction
Classification
Label
MODEL pipeline
Load model pipeline
Train model
Save model
Load training data
model BUILD pipeline

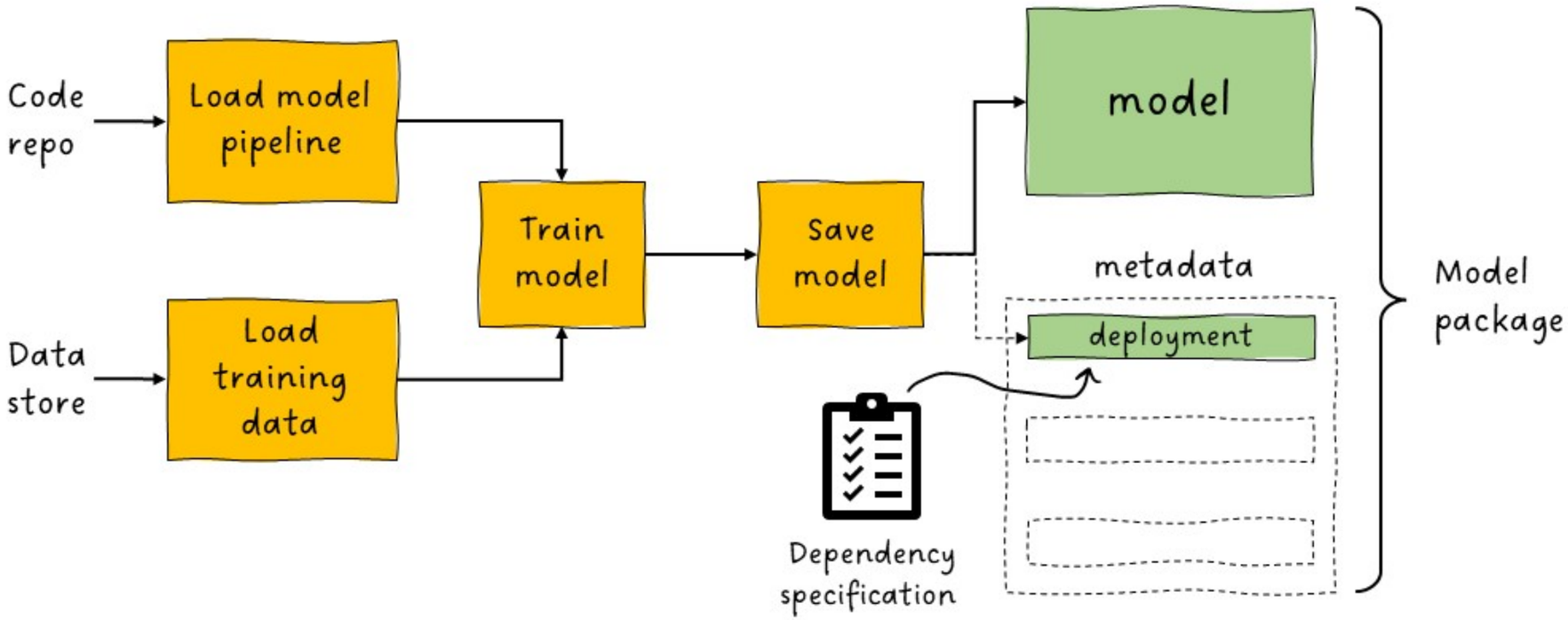


- ✓ Deployment
- ✓ Reproducibility
- ✓ Monitoring
- ✓ CI/CD integration


Code repo
Load model pipeline
Data store
Load training data
Train model
Save model
model
metadata
Model package

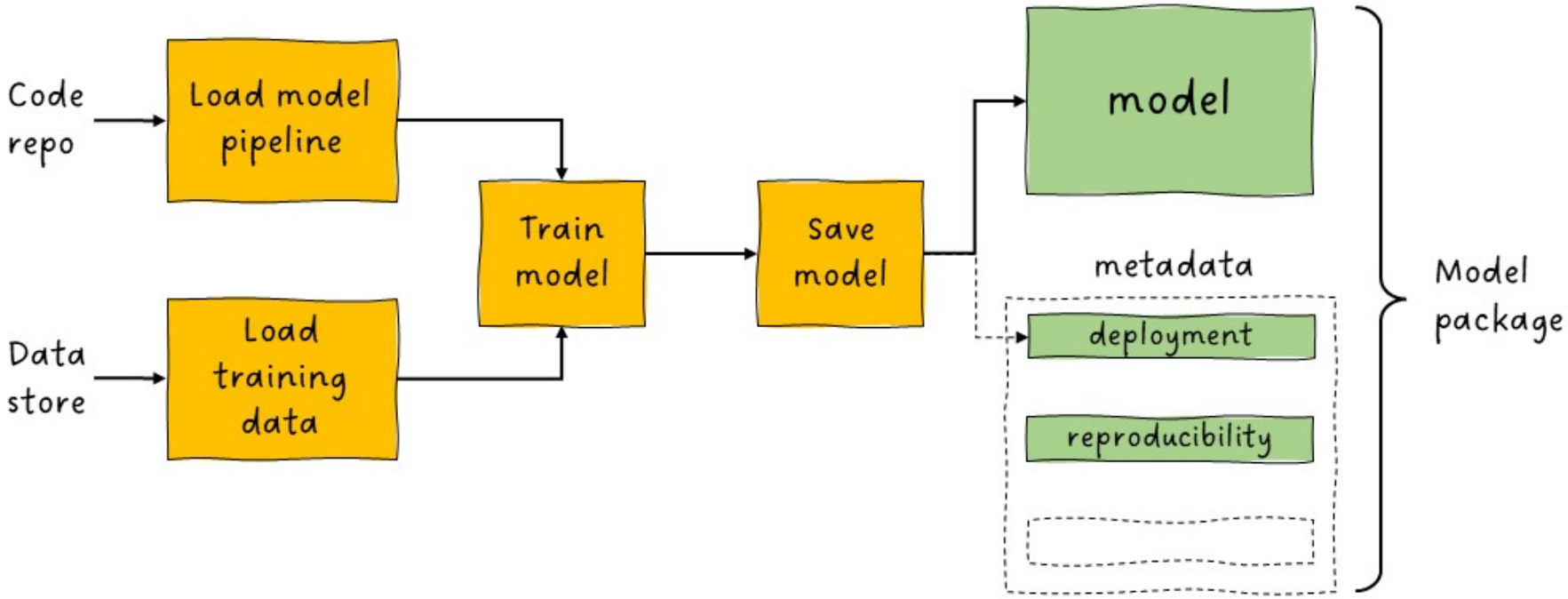


* refresher: artifacts == pipeline outputs

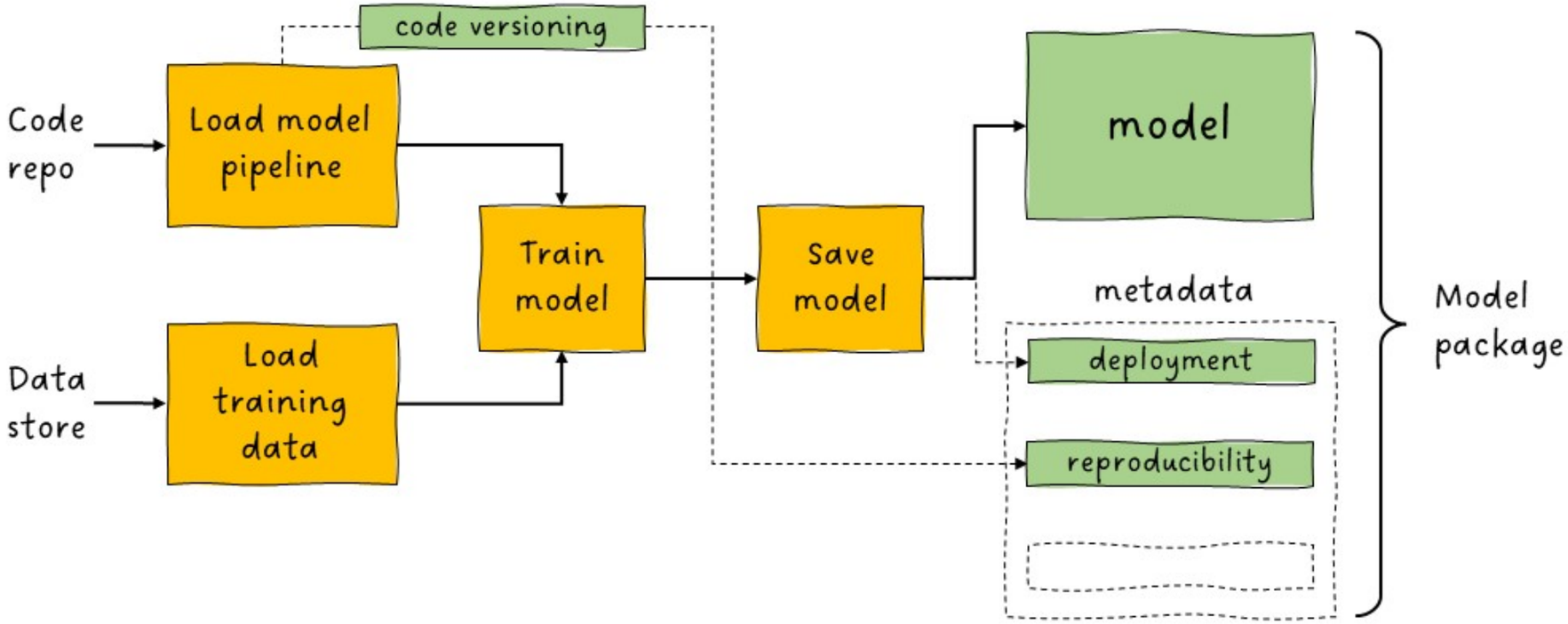


* refresher: artifacts == pipeline outputs

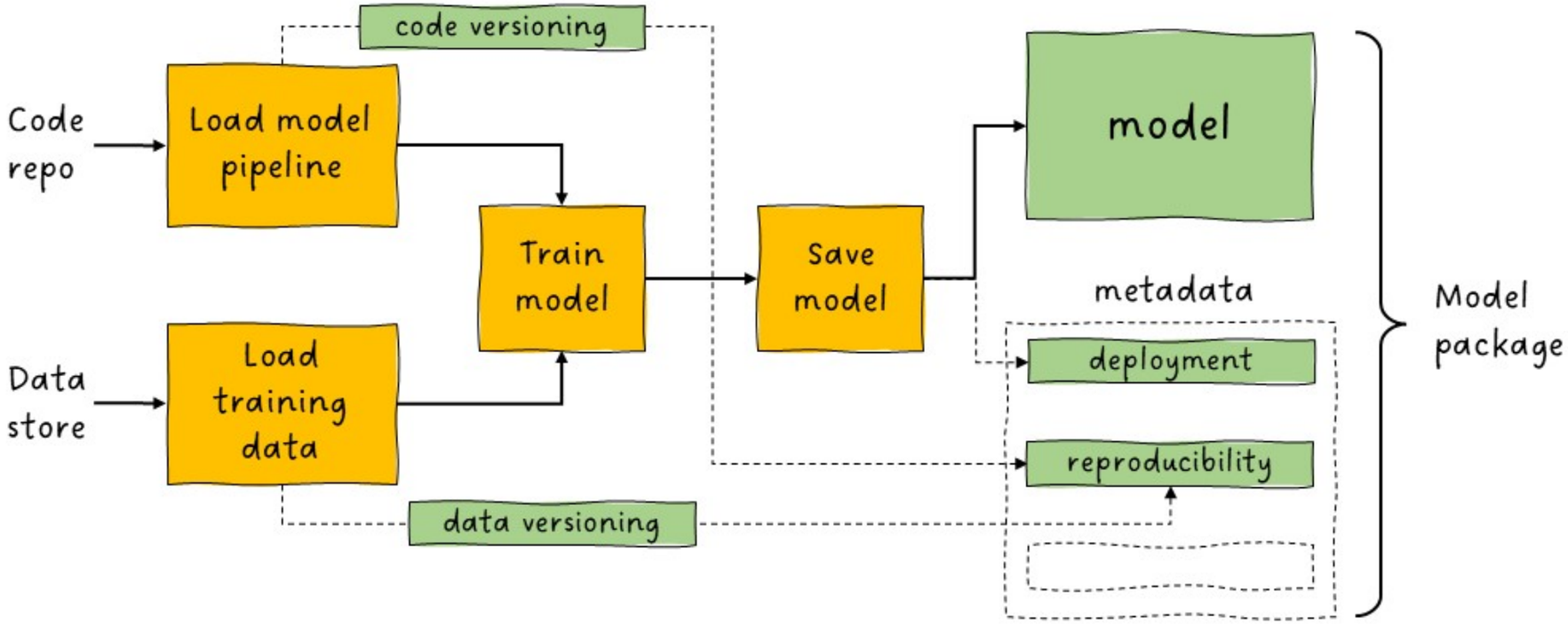
Reproducibility = Control = Trust

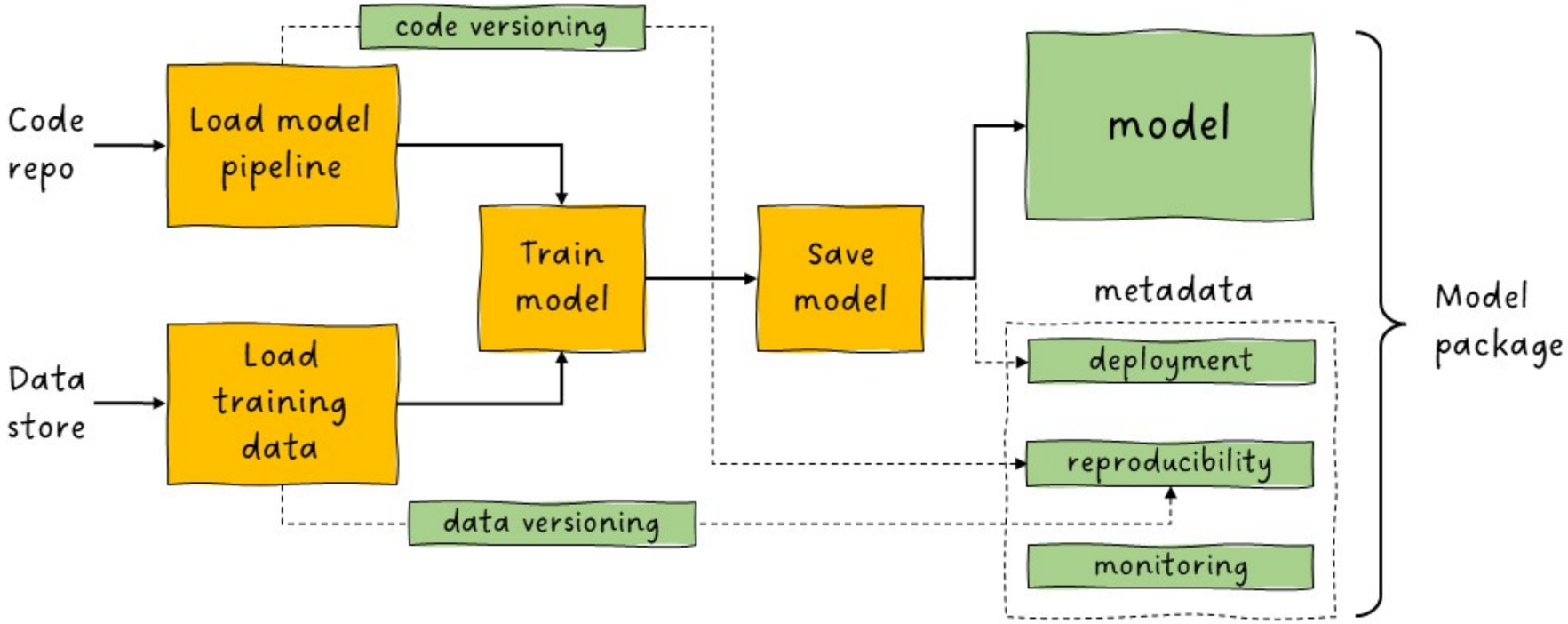


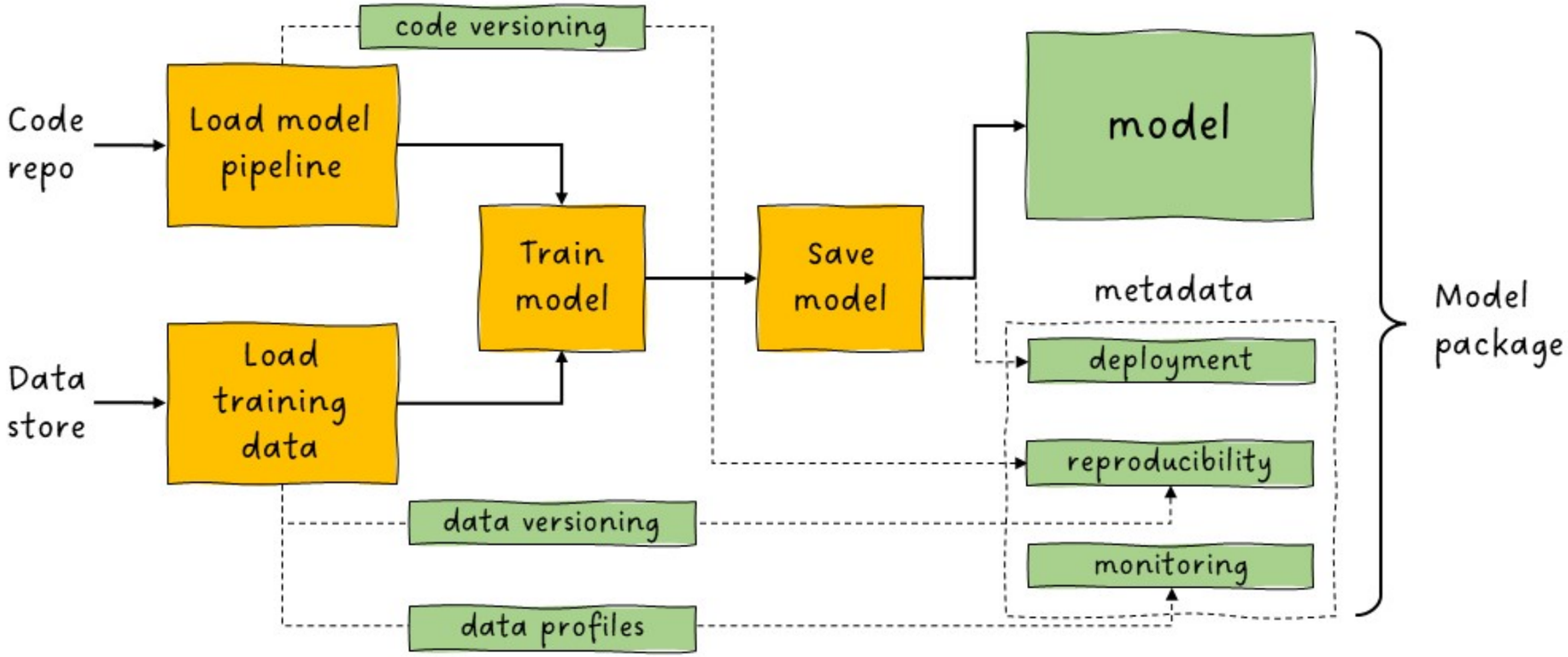
Reproducibility = Control = Trust

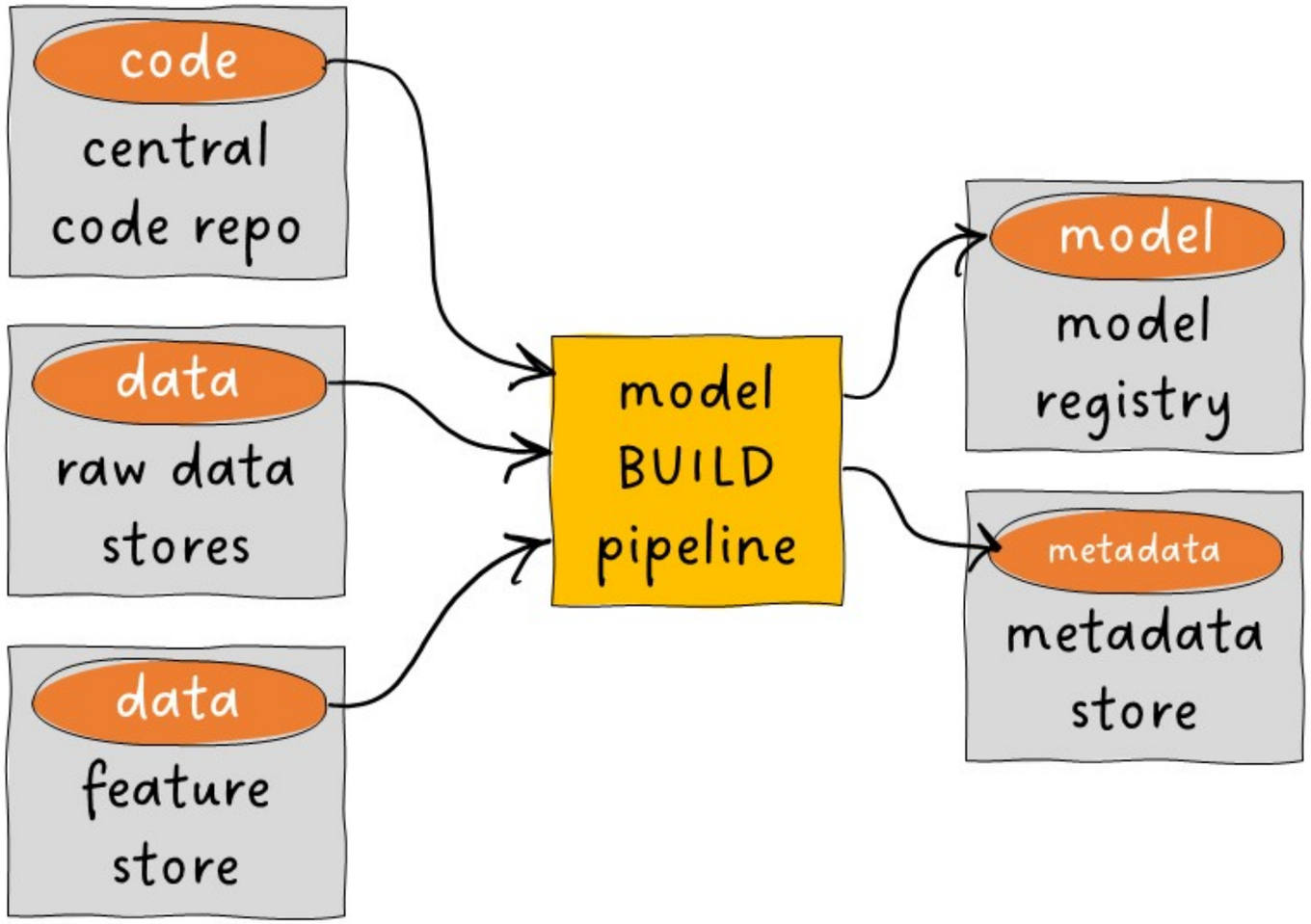


Reproducibility = Control = Trust




code versioning
Code repo
Load model pipeline
Train model
Save model
model
Data store
Load training data
metadata
deployment
reproducibility
monitoring
data versioning
Model package


code versioning
Code repo
Load model pipeline
Train model
Save model
model
Data store
Load training data
metadata
deployment
reproducibility
monitoring
Model package
data versioning
data profiles


code
central
code repo
data
raw data
stores
data
feature
store
model
BUILD
pipeline
model
model
registry
metadata
metadata
store

Let's practice!

MLOPS DEPLOYMENT AND LIFE CYCLING

Model packaging

MLOPS DEPLOYMENT AND LIFE CYCLING

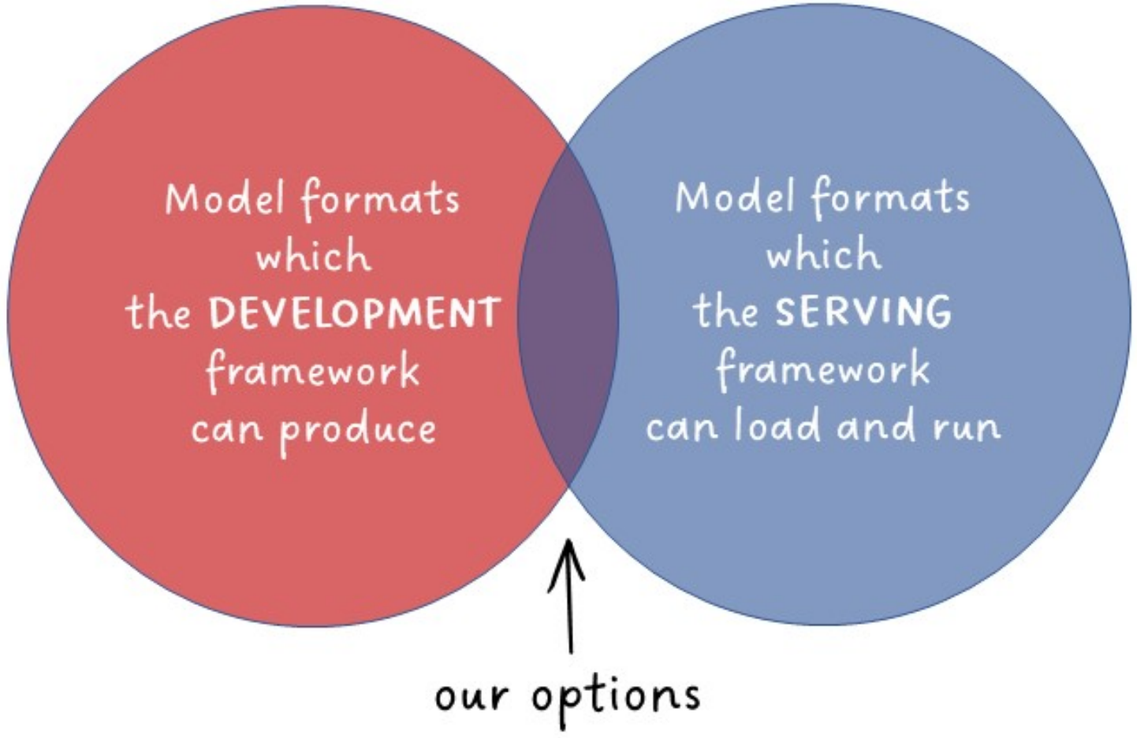
Nemanja Radojkovic
Senior Machine Learning Engineer


deployment


deployment
reproducibility


deployment
reproducibility
monitoring

Model storage options



Examples of model storage formats

python

pickle

Examples of model storage formats

python

pickle

PROs

- Universal
- Train using one programming language, serve using a completely different one.

CONs

- Difficult to customize

Examples of model storage formats

PROs

- Universal
- Train using one programming language, serve using a completely different one.

CONs

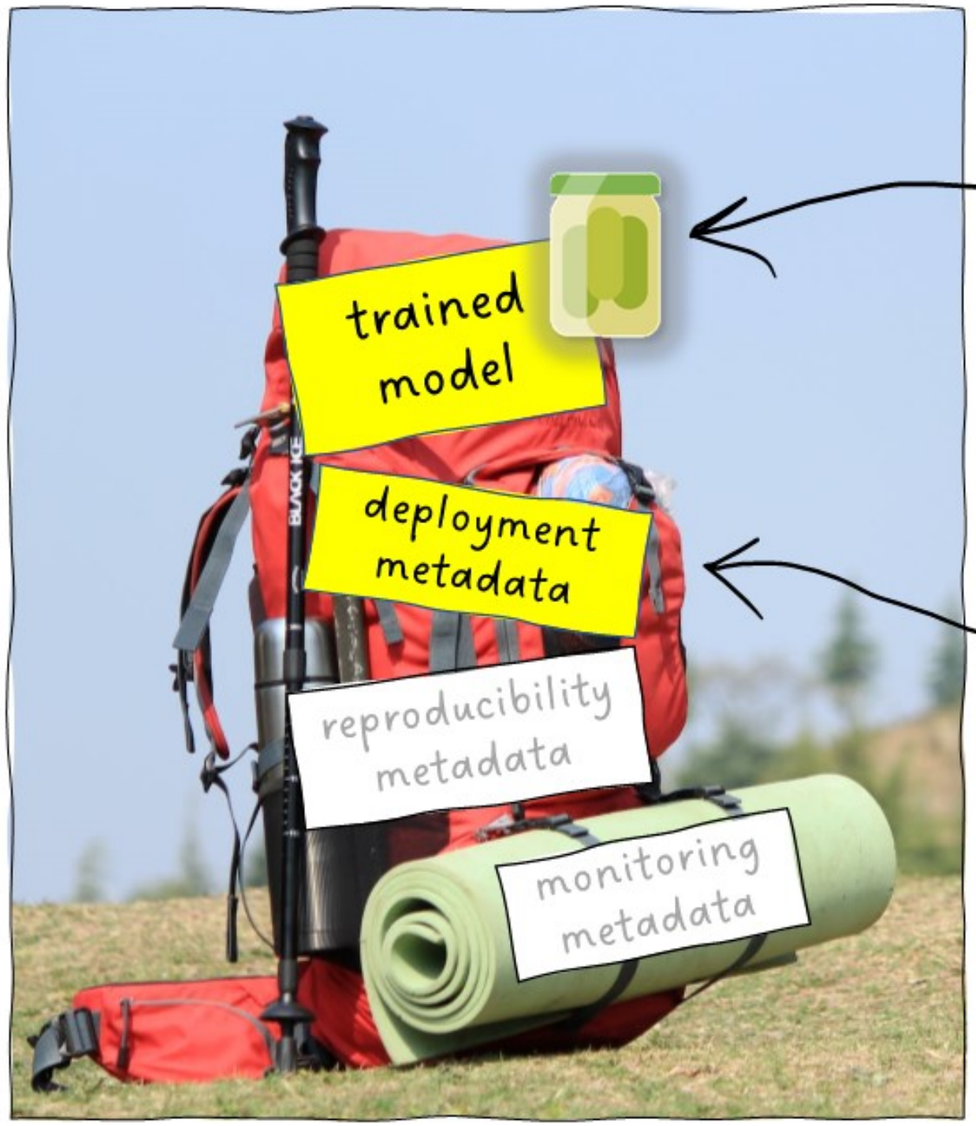
- Difficult to customize

PROs

- Python ecosystem
- Not ML specific, can store anything

CONs

- No cross-platform compatibility. Training and serving environments must be identical.



IF model_format == .pickle
THEN:

store list of model's dependencies within model package and verify server compatibility when loading

Reproducibility

Automated model re-creation

Reproducibility == Control == Trust

Reproducibility checklist

- ❑ Pointer to the exact version of the model build pipeline code.
- ❑ Pointer to the exact versions of the datasets used during the training
  - ❑ including the train/splits during performance evaluation.
- ❑ Record of the performance achieved on the test set.

Monitoring checklist

- ❑ Input data profile
- ❑ Output data profile


trained model
deployment metadata
reproducibility metadata
monitoring metadata

Lock 'n' load!

Let's practice!

MLOPS DEPLOYMENT AND LIFE CYCLING