Real World MLOps with MLflow

Stavros Niafas
Fosscomm 2021

mlflow™

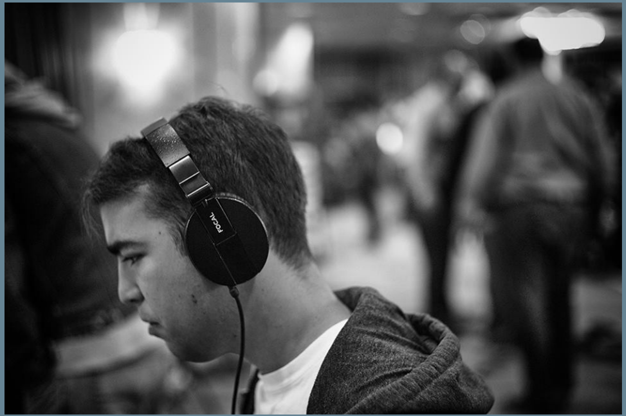
About Me

ML Engineer in convert group

2nd year speaking in Fosscomm

General Interest in

- Machine Learning
- Data-Centric AI
- FLOSS & GNU/Linux



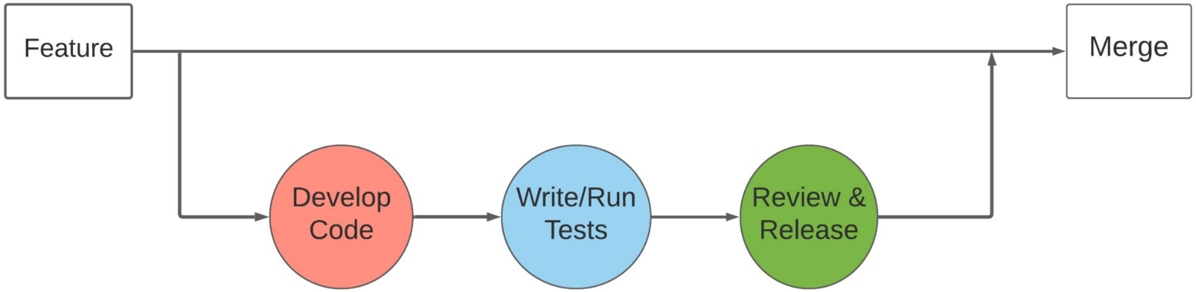
Outline

- Overview in ML development challenges
- Machine Learning Lifecycle
- Motivation and Challenges
- MLOps
- MLflow
- Q & A

Traditional Software vs Machine Learning

Traditional Software

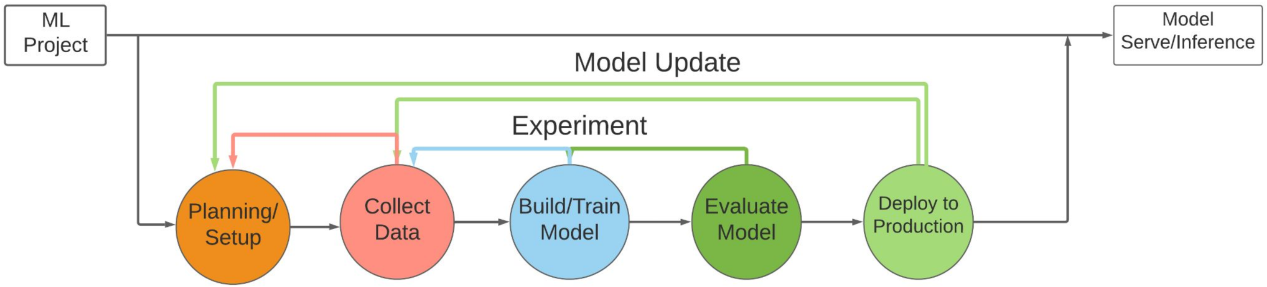
- Goal: Meet a function requirement
- Quality depends on code and testing (Unit/Integration)
- Typical pick software stack, fewer libraries and tools
- Limited deployment environments



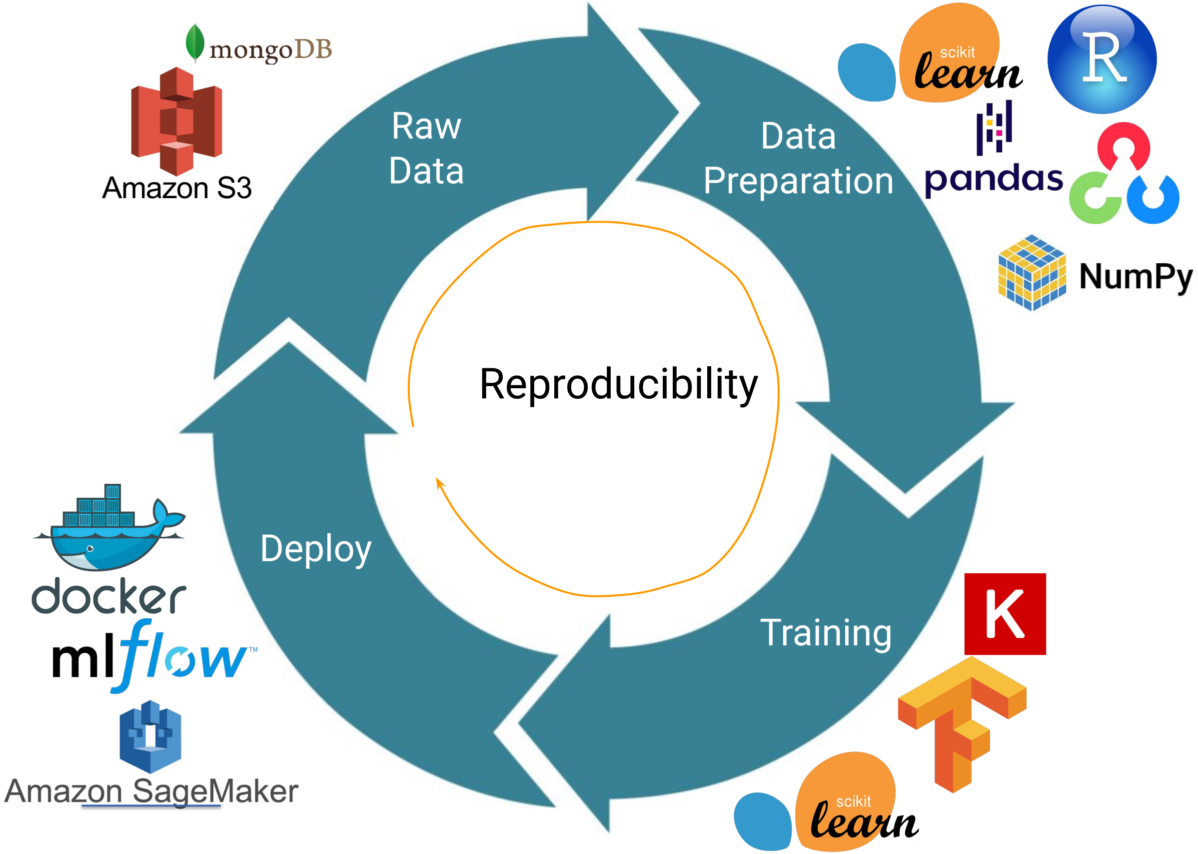
Traditional Software vs Machine Learning

Machine Learning

- Goal: Optimize a metric by experimentation
- Quality depends on code, framework's api, input data and hyperparameters
- Combine many libraries, train/evaluate models
- Diverse deployment environments (CPU/GPU, training/inference servers)



Machine Learning Lifecycle



Motivation

INPUT

data v1

data v2

Algorithm A

Algorithm B

Param set 1

Param set 2

EXECUTION

ML Model

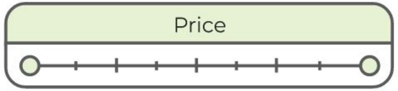
Output

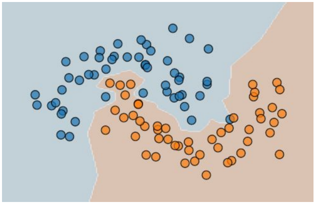
Model

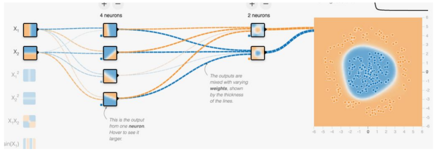
Statistics

Logs

Results







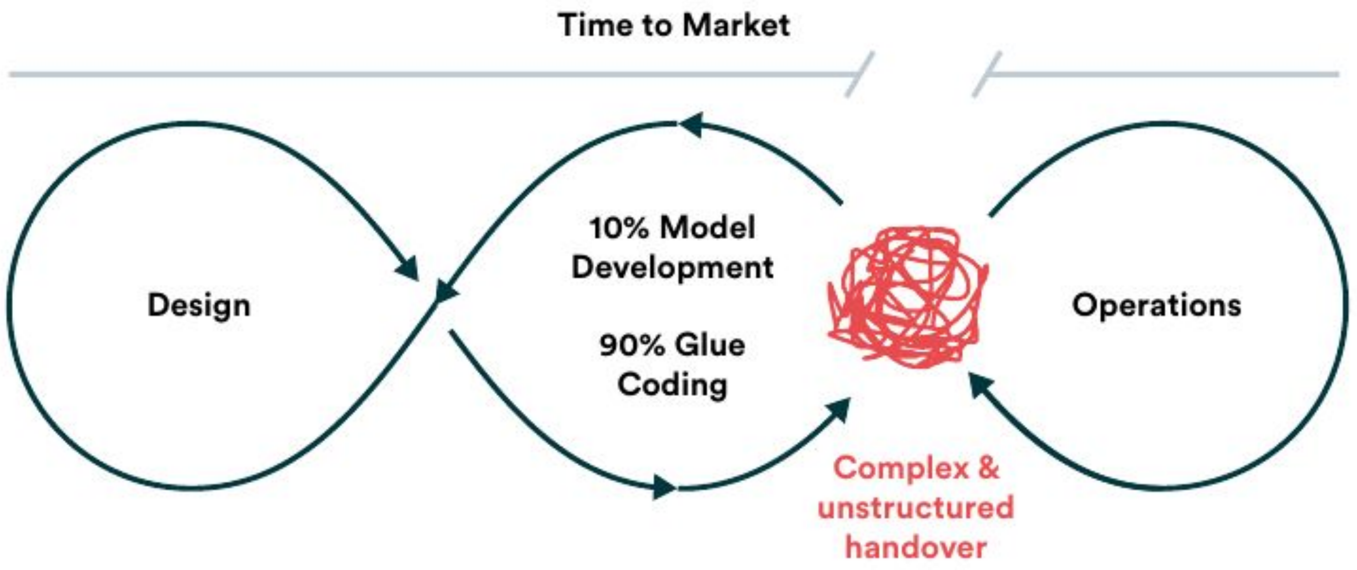
Machine Learning Challenges

Machine Learning Challenges

Venture Beat: 13% ML projects make it to production[1]

[1]:https://venturebeat.com/2019/07/19/why-do-87-of-data-science-projects-never-make-it-into-production/

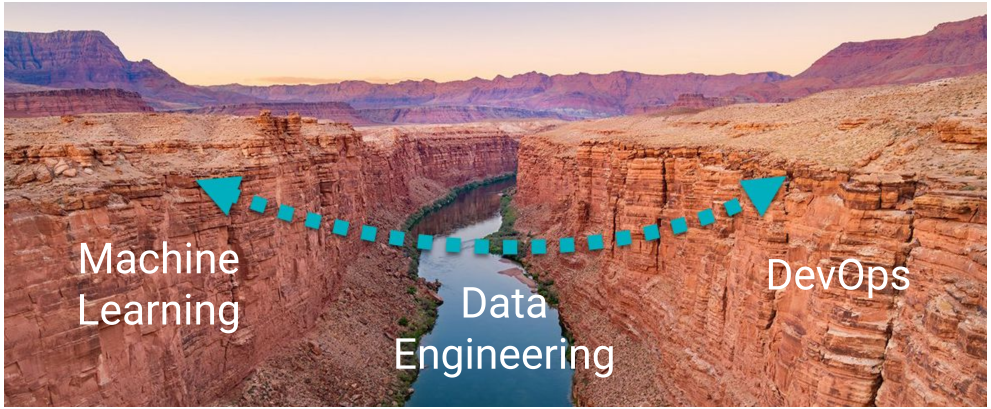
Machine Learning Challenges



[1]: Valohai, practical MLOps book

Machine Learning (Production) Challenges

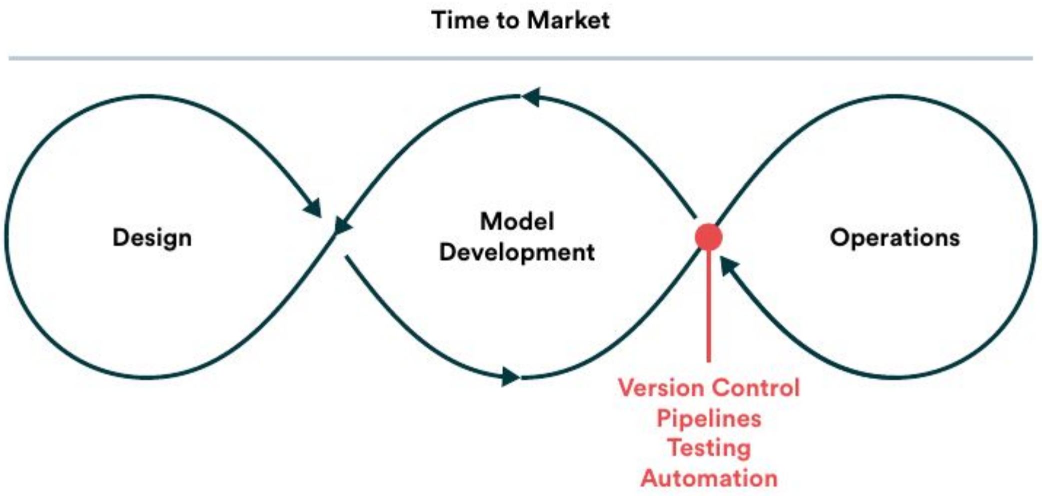
Data Science Team



Software Engineering Team

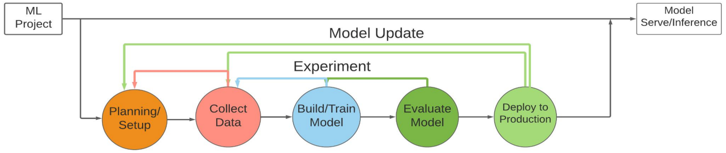
MLOps

MLOps


Time to Market
Design
Model
Development
Operations
Version Control
Pipelines
Testing
Automation

MLOps

ML action point

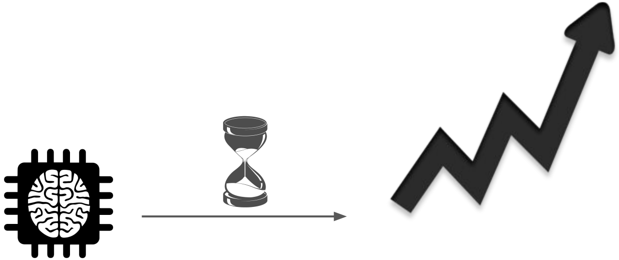


Reduce the risk involved

- Loss of knowledge
- Failures in production
- Regulatory and ethical

MLOps

Operational action point



Reduce time to market

- Model Deployment
- Automate w/ CI/CD
- Agnostic model format/software/artifact types

Introducing mlflow™

Open machine learning platform for ML lifecycle

Works w/ popular ML library & language

Runs the same way anywhere (cloud/locally)

Designed for 1 or 1000+ person organizations

Simple. Modular. Easy-to-use

Friendly learning curve

mlflow™ Design

API-First

- Track runs, log artifacts, models, metrics, data
- Agnostic model format
- Multi ML framework support (TF, Keras, Pytorch, sklearn, etc)
- Deploy anywhere (integration w/ AWS sagemaker, databricks, local)

Programmatic API clients, REST APIs, & CLI support

mlflow™ Design

Modular

- Allow different components individually (e.g. use it for tracking but not for deploying)
- Distinctive and Selective

Multi Components: Tracking/Projects/Models/Registry

MLFlow components

mlflow

Tracking

Record experiments

- Code
- HParams
- Environment
- Configuration
- Results

mlflow

Projects

Package data in format that enables reproducible runs on many platforms

mlflow

Models

Deploy machine learning models in diverse serving environments using MLflow built REST api

mlflow

Model Registry

Store, annotate, transition and manage models in a central repository. Store artifacts local or remote (e.g. s3)

MLflow Tracking

Experiments: experiment names, run names

Parameters: (hyper)parameters inputs of code/model

Metrics: numeric values accuracy, loss, etc (updated over time)

Artifacts: files, data, logs and models

Source: deployed code executed in inference

Configuration: deployment environment yaml, dependency libraries

Version: Code version, Model version, model stage

Tags & Notes: Auxiliary information and description about a run

Model development w/out MLflow Tracking

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor

# models params
params = {"n_estimators": 1, "random_state": 42}

# split dataset
data = read_data(file)
x_train, y_train, x_test, y_test = data_split(data, split=0.7)

# train model
rfr = RandomForestRegressor(**params)
rfr.fit(x_train, y_train)

# inference
predictions = rfr.predict(x_test)
score = rfr.evaluate(predictions, y_test)

pickle.dump(model, "regressor_model.pkl")
```

n_est:1, state:42, split:0.7, model: rfr, r2_score: 0.3
n_est:2, state:42, split:0.7, model: rfr, r2_score: 0.5
n_est:3, state:42, split:0.7, model: rfr, r2_score: 0.35
n_est:1, state:42, split:0.8, model: rfr, r2_score: 0.2
n_est:1, state:42, split:0.8, model: rfr, r2_score: 0.7
n_est:1, state:42, split:0.8, model: gbr, r2_score: 0.9
n_est:2, state:42, split:0.8, model: gbr, r2_score: 0.46
n_est:3, state:42, split:0.8, model: gbr, r2_score: 0.67

What if I tune model parameters?

Model development w/out MLflow Tracking

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor

# models params
params = {"n_estimators": 1, "random_state": 42}

# split dataset
data = read_data(file)
x_train, y_train, x_test, y_test = data_split(data, split=0.7)

# train model
rfr = RandomForestRegressor(**params)
rfr.fit(x_train, y_train)

# inference
predictions = rfr.predict(x_test)
score = rfr.evaluate(predictions, y_test)

pickle.dump(model, "regressor_model.pkl")
```

```
n_est:1, state:42, split:0.7, model: rfr, r2_score: 0.3
n_est:2, state:42, split:0.7, model: rfr, r2_score: 0.5
n_est:3, state:42, split:0.7, model: rfr, r2_score: 0.35
n_est:1, state:42, split:0.8, model: rfr, r2_score: 0.2
n_est:1, state:42, split:0.8, model: rfr, r2_score: 0.7
n_est:1, state:42, split:0.8, model: gbr, r2_score: 0.9
n_est:2, state:42, split:0.8, model: gbr, r2_score: 0.46
n_est:3, state:42, split:0.8, model: gbr, r2_score: 0.67
```

What if I try another dataset split?

Model development w/out MLflow Tracking

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor

# models params
params = {"n_estimators": 1, "random_state": 42}

# split dataset
data = read_data(file)
x_train, y_train, x_test, y_test = data_split(data, split=0.7)

# train model
rfr = RandomForestRegressor(**params)
rfr.fit(x_train, y_train)

# inference
predictions = rfr.predict(x_test)
score = rfr.evaluate(predictions, y_test)

pickle.dump(model, "regressor_model.pkl")
```

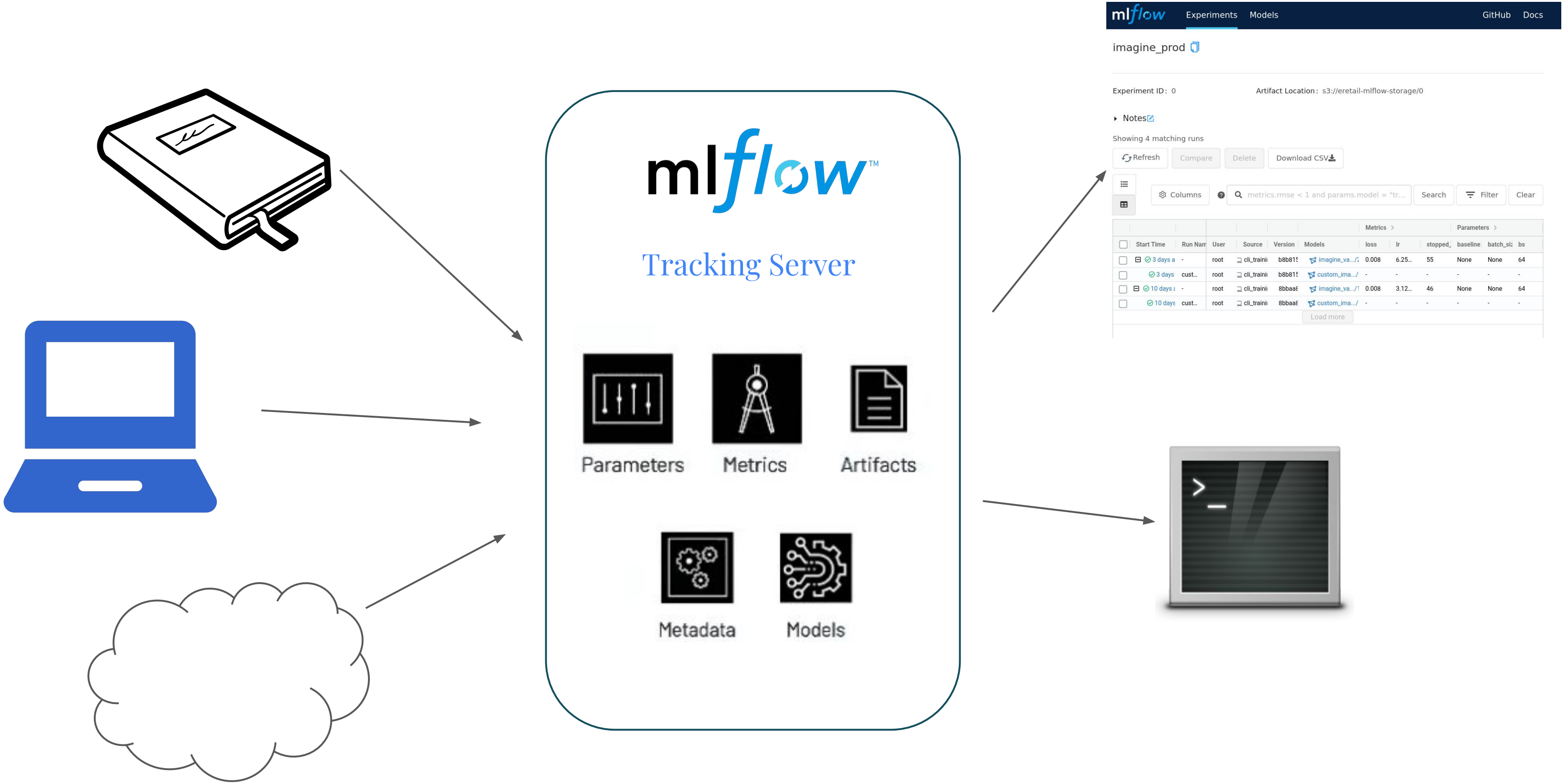
n_est:1, state:42, split:0.7, model: rfr, r2_score: 0.3
n_est:2, state:42, split:0.7, model: rfr, r2_score: 0.5
n_est:3, state:42, split:0.7, model: rfr, r2_score: 0.35
n_est:1, state:42, split:0.8, model: rfr, r2_score: 0.2
n_est:1, state:42, split:0.8, model: rfr, r2_score: 0.7
n_est:1, state:42, split:0.8, model: gbr, r2_score: 0.9
n_est:2, state:42, split:0.8, model: gbr, r2_score: 0.46
n_est:3, state:42, split:0.8, model: gbr, r2_score: 0.67

What model version scores best?

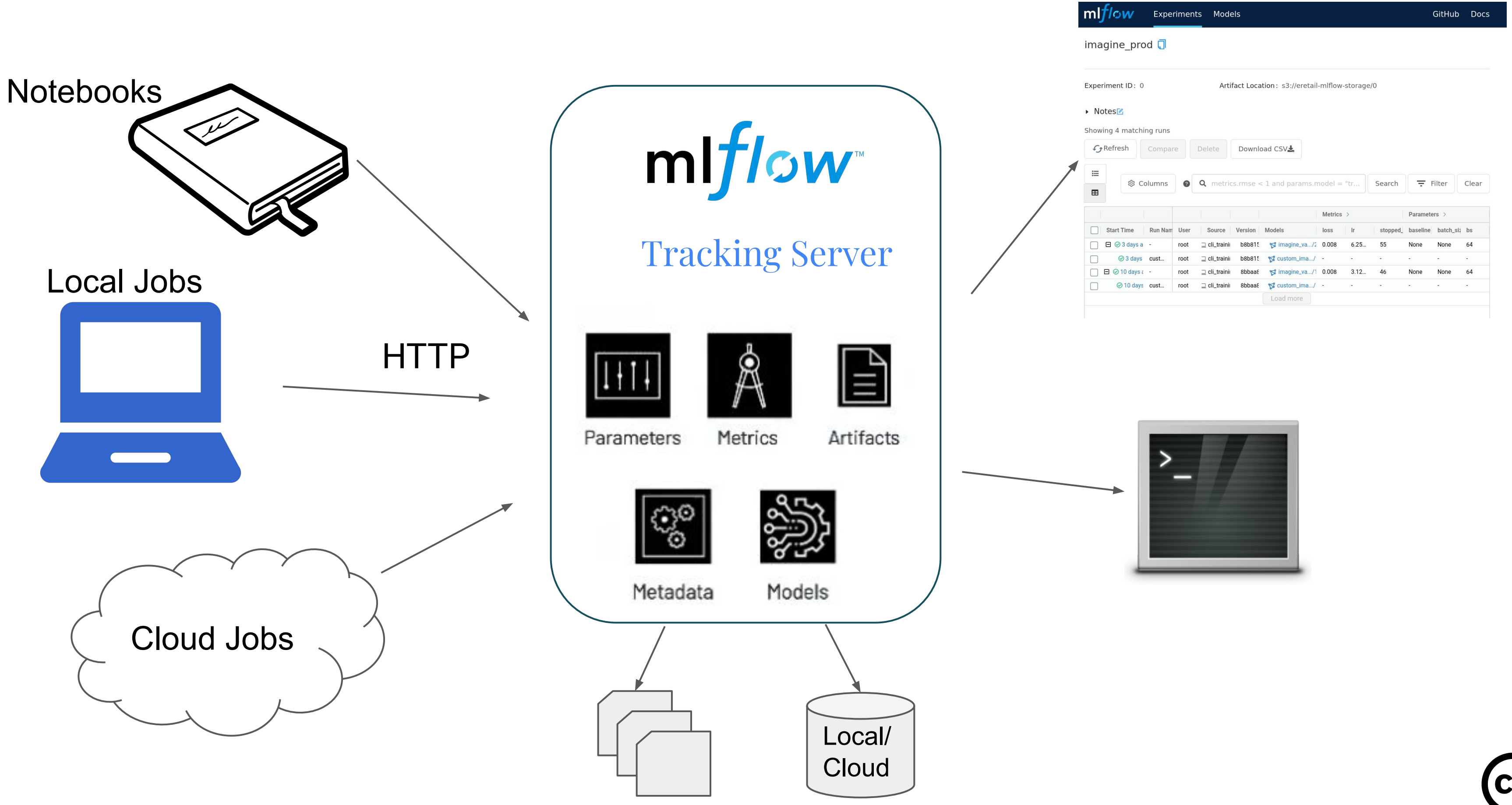
Model development with MLflow

```
# set experiment name
mlflow.set_experiment("Experiment_RFR")

# models params
params = {"n_estimators": 1, "random_state": 42}

# log params
mlflow.log_params(params)

# split dataset
data = read_data(file)
x_train, y_train, x_test, y_test = data_split(data, split=0.7)

# log dataset
mlflow.log_artifact(x_train)
mlflow.log_artifact(y_train)

# log experiment
with mlflow.start_run(run_name="rfr") as train_run:

  # train model
  rfr = RandomForestRegressor(**params)
  rfr.fit(x_train, y_train)

  mlflow.sklearn.log_model(rfr)

  # inference
  predictions = rfr.predict(x_test)
  score = rfr.evaluate(predictions, y_test)

  mlflow.log_metric(score)
```

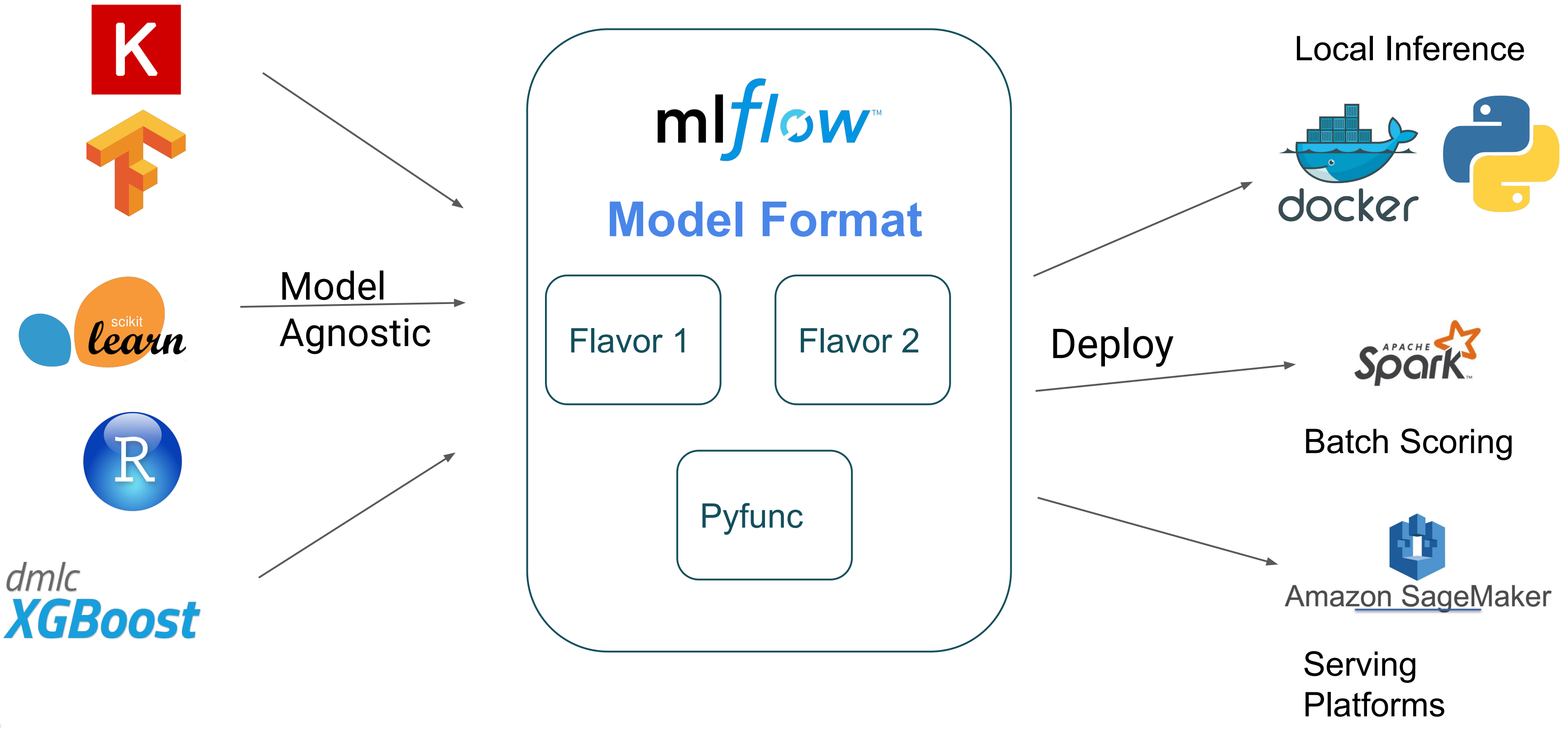
MLflow Tracking



MLflow Tracking



MLflow Models



MLflow APIs

Fluent MLflow APIs

Python

- High-level operations for runs
- Model flavor APIs

R, Java

- Experiments, runs, etc

MLflowClient

Python

- Low-level CRUD interface to experiments and runs
- Manage, select model for inference/deployment
- Mange metrics, params
- Manage experiments

MLflow REST API

- Make REST calls to Tracking Server
- REST calls to MLflow Serve API
- curl https:myservice.org:5000/endpoint

Covered Topics

- ML development lifecycle diverses from traditional software development
- Need for fill the gap and reduce technical friction from R&D -> Production
- MLflow: an open and modular library to simplify ML lifecycle
- Easy to install
- Develop or/and deploy
- Local/remote
- Rich support in model flavors

Thank you

sniafas sniafas sniafas.eu