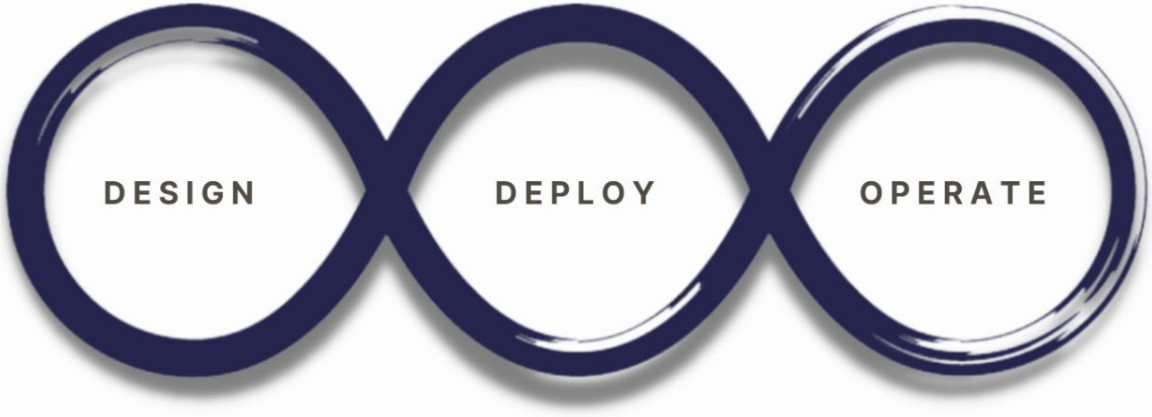
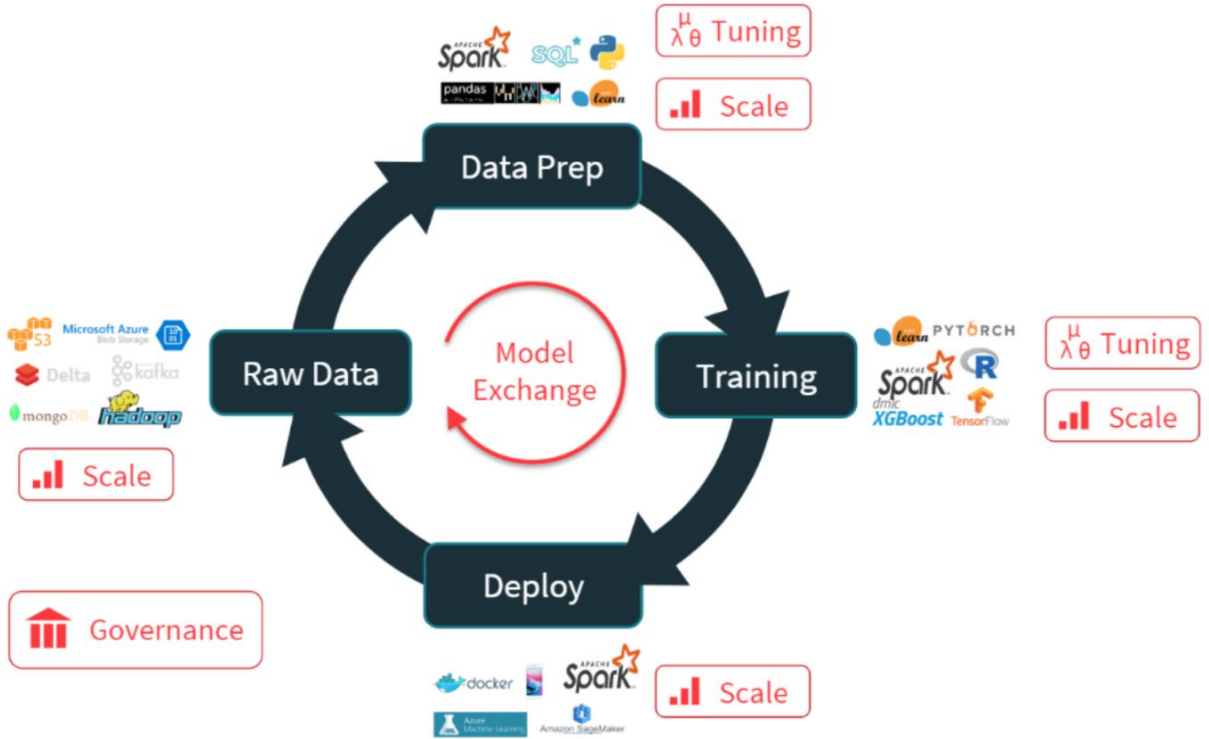


MLOPS

MACHINE LEARNING OPERATIONS

ML LIFECYCLE



SCENARIOS DURING TRAINING AND DEPLOYMENT

REPRODUCIBILITY

- Different chunks or versions of data trained with models
- Tuning various model versions based on different hyperparameters

TRACKING

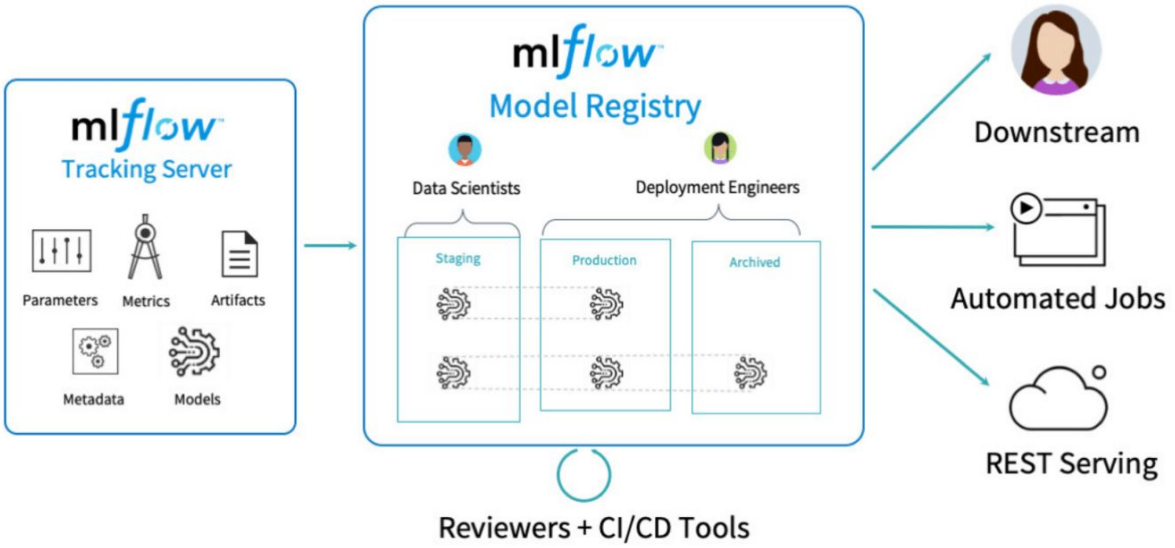
- Comparing model performance with every training
- Monitoring model performance with variable data

DEPLOYMENT

- Scaling model during the production stage
- Gathering inference from staged models

mlflow™

PROGRESS WITH EACH EXPERIMENT



MLFLOW COMPONENTS

MLflow is an open-source platform to manage the ML lifecycle, including experimentation, reproducibility, deployment, and a central model registry. MLflow currently offers four components

MLflow Tracking

Record and query experiments: code, data, config, and results

MLflow Projects

Package data science code in a format to reproduce runs on any platform

MLflow Models

Deploy machine learning models in diverse serving environments

Model Registry

Store, annotate, discover, and manage models in a central repository

TERMINOLOGIES

PARAMETERS

- Key value inputs for models, model hyperparameters

METRICS

- Performance value that is monitored

METADATA

- Tags and Notes that can be customly added to save information about the model

ARTIFACTS

- Flies, Data, Models

EXPERIMENT

- One instance of training that recorded after a RUN

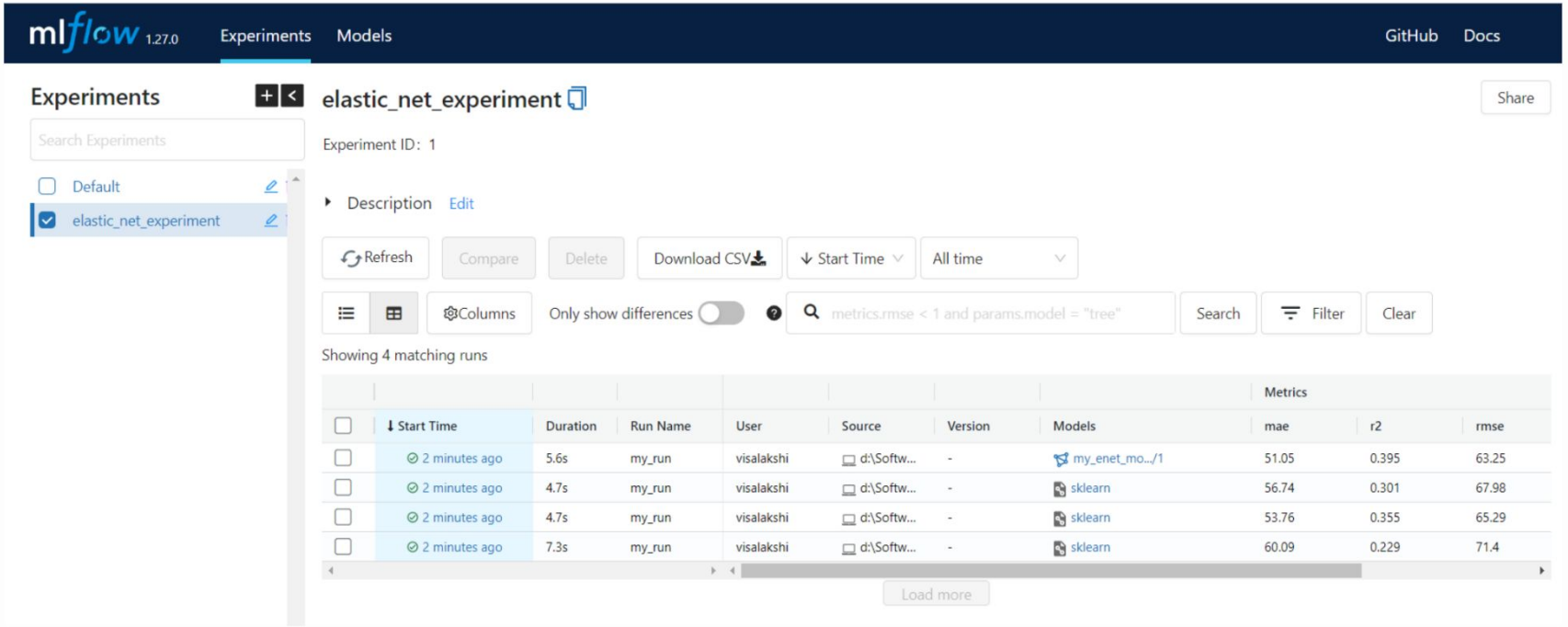
TRACKING EXPERIMENTS

There is a simple flow of creating an MLFlow Experiment. Repeating it in a FLOW gives us numerous ML Experiments that can be tracked for inference

1. Load data, set parameter,s and train
2. Start experiment with start_run()
3. Record the run parameters for a model with log_param()
4. Record metric obtained with log_metric()
5. Record the respective model trained with log_model()
6. You can also log relevant pictures with log_artifact()

```
import mlflow
data   = load_text(file)
ngrams = extract_ngrams(data, N=n)
model  = train_model(ngrams,
                     learning_rate=lr)
score  = compute_accuracy(model)
with mlflow.start_run():
  mlflow.log_param("data_file", file)
  mlflow.log_param("n", n)
  mlflow.log_param("learn_rate", lr)
  mlflow.log_metric("score", score)
  mlflow.sklearn.log_model(model)
```

MLFLOW UI


mlflow 1.27.0
Experiments
Models
GitHub
Docs
Experiments
Search Experiments
Default
elastic_net_experiment
elastic_net_experiment
Share
Experiment ID: 1
Description
Edit
Refresh
Compare
Delete
Download CSV
Start Time
All time
Columns
Only show differences
metrics.rmse < 1 and params.model = "tree"
Search
Filter
Clear
Showing 4 matching runs
Metrics
Start Time
Duration
Run Name
User
Source
Version
Models
mae
r2
rmse
2 minutes ago
5.6s
my_run
visalakshi
d:\Softw...
-
my_enet_mo.../1
51.05
0.395
63.25
2 minutes ago
4.7s
my_run
visalakshi
d:\Softw...
-
sklearn
56.74
0.301
67.98
2 minutes ago
4.7s
my_run
visalakshi
d:\Softw...
-
sklearn
53.76
0.355
65.29
2 minutes ago
7.3s
my_run
visalakshi
d:\Softw...
-
sklearn
60.09
0.229
71.4
Load more

REPRODUCIBILITY WITH PROJECTS

With great set of tools comes greater pain of reproducibility. If you have an algorithm that needs to be run in a diverse set of environment with only a few parameters changed, then migrating the set of artifacts with only relevant objects can be hectic

REPRODUCIBILITY WITH PROJECTS

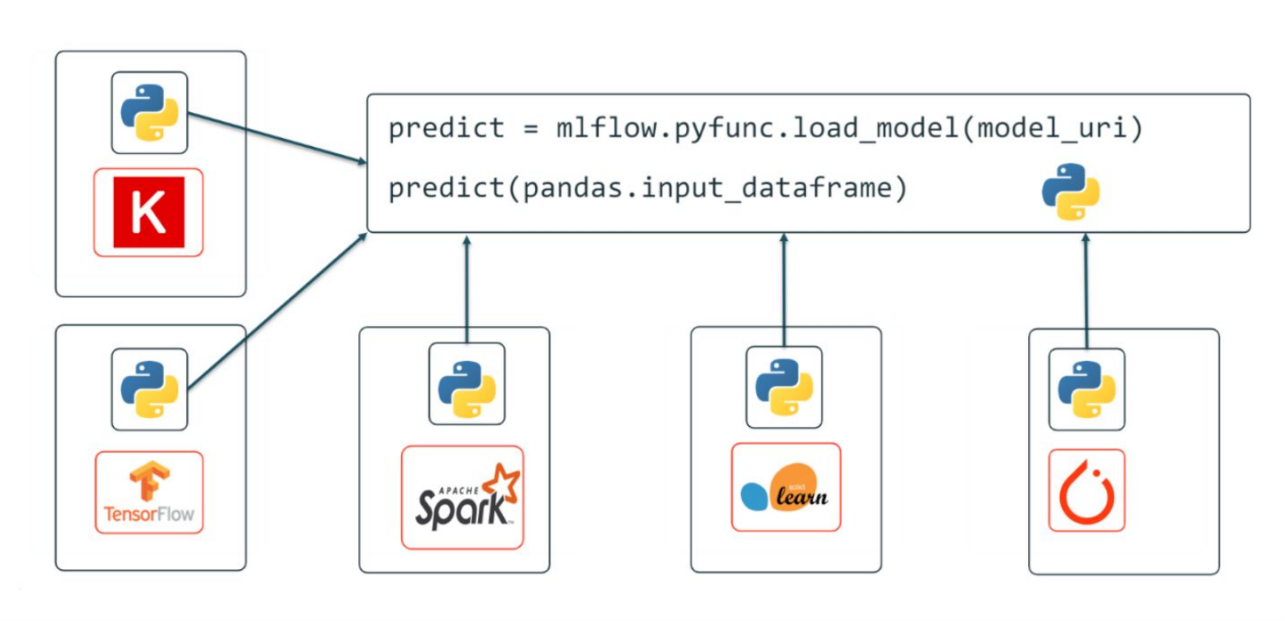
MLFlow Project saves the specific set of files relevant to each experiment so that it is easily reproducible.

With every run, the model along with it's environment config is saved and can be migrated to any suitable environment to test and deploy.

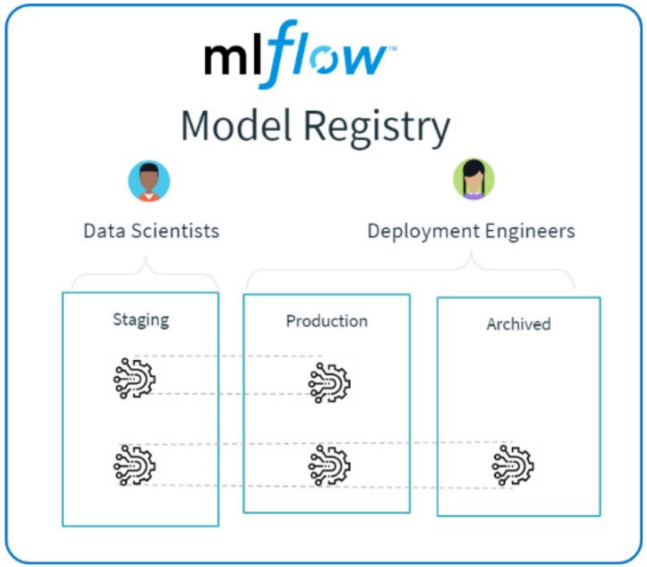
This abstraction is called Project Spec with MLFLow

REPRODUCIBILITY WITH PROJECTS

These diversified models are called FLAVORS. They are similar algorithms that run on various framework to experiment and compare results. Every flavor is also stored as a part of the project.



PRODUCTIONIZING



When the models are getting released, they can also be productionized in sequence with different arbitrators managing those models.

Every model goes through Staging, Production and at last Archive.

These functionalities are achieved with Model Registry where version control management of models as well as monitoring is performed.