What’s Next for mlflow in 2019

Matei Zaharia, Corey Zumar, Sid Murching

February 12, 2019

databricks

Outline

MLflow overview

Feedback so far

Databricks’ development themes for 2019

Demos of upcoming features

Outline

MLflow overview

Feedback so far

Databricks’ development themes for 2019

Demos of upcoming features

ML Development is Harder than Traditional Software Development

Traditional Software

Goal: meet a functional specification

Machine Learning

Goal: optimize a metric (e.g. accuracy)

Traditional Software	Machine Learning
Goal: meet a functional specification	*Goal:* optimize a metric (e.g. accuracy)
Quality depends only on code	Quality depends on data, code & tuning → Must regularly update with fresh data

Traditional Software	Machine Learning
Goal: meet a functional specification	*Goal:* optimize a metric (e.g. accuracy)
Quality depends only on code	Quality depends on data, code & tuning → Must regularly update with fresh data
Typically one software stack	Constantly experiment w/ new libraries + models (and must productionize them!)

What is mlflow?

Open source platform to manage ML development

- Lightweight APIs & abstractions that work with any ML library
- Designed to be useful for 1 user or 1000+ person orgs
- Runs the same way anywhere (e.g. any cloud)

Key principle: “open interface” APIs that work with any existing ML library, app, deployment tool, etc

MLflow Components

mlflow Tracking

Record and query experiments: code, params, results, etc

mlflow Projects

Code packaging for reproducible runs on any platform

mlflow Models

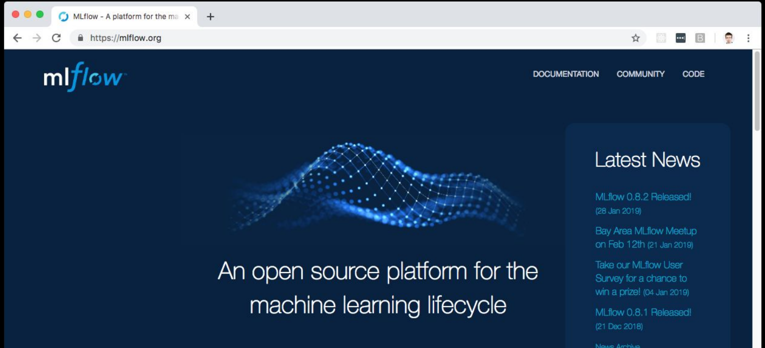
Model packaging and deployment to diverse environments

Learning mlflow

`pip install mlflow` **to get started in Python**
(APIs also available in Java and R)

Docs and tutorials at mlflow.org

- Hyperparameter tuning, REST serving, batch scoring, etc



An open source platform for the machine learning lifecycle

Latest News

MLflow 0.8.2 Released! (28 Jan 2019)

Bay Area MLflow Meetup on Feb 12th (21 Jan 2019)

Take our MLflow User Survey for a chance to win a prize! (04 Jan 2019)

MLflow 0.8.1 Released! (21 Dec 2018)

News Archive

Outline

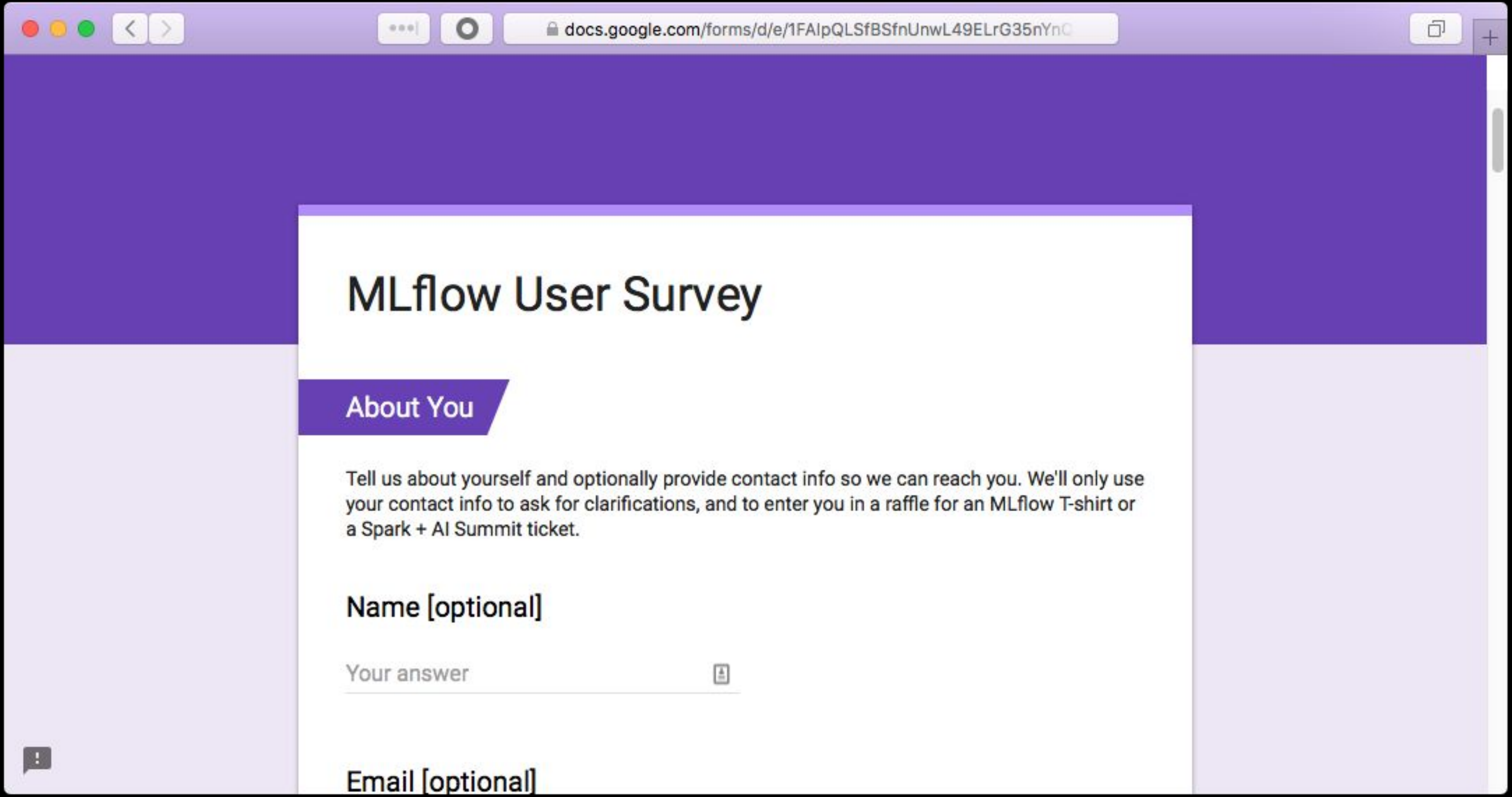
MLflow overview

Feedback so far

Databricks’ development themes for 2019

Demos of upcoming features

Running a user survey at mlflow.org (fill it in if you haven't!)



MLflow User Survey

About You

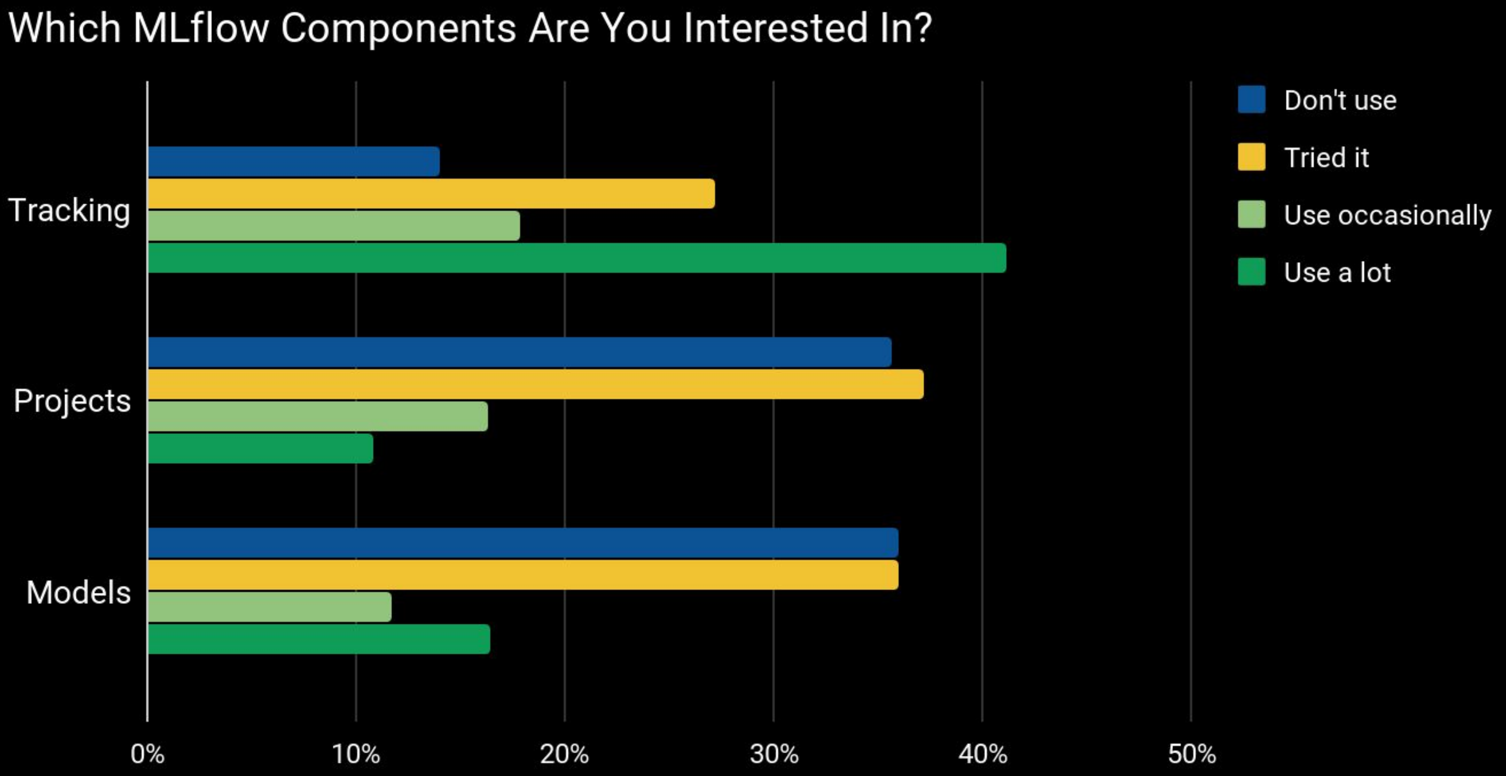
Tell us about yourself and optionally provide contact info so we can reach you. We'll only use your contact info to ask for clarifications, and to enter you in a raffle for an MLflow T-shirt or a Spark + AI Summit ticket.

Name [optional]

Your answer

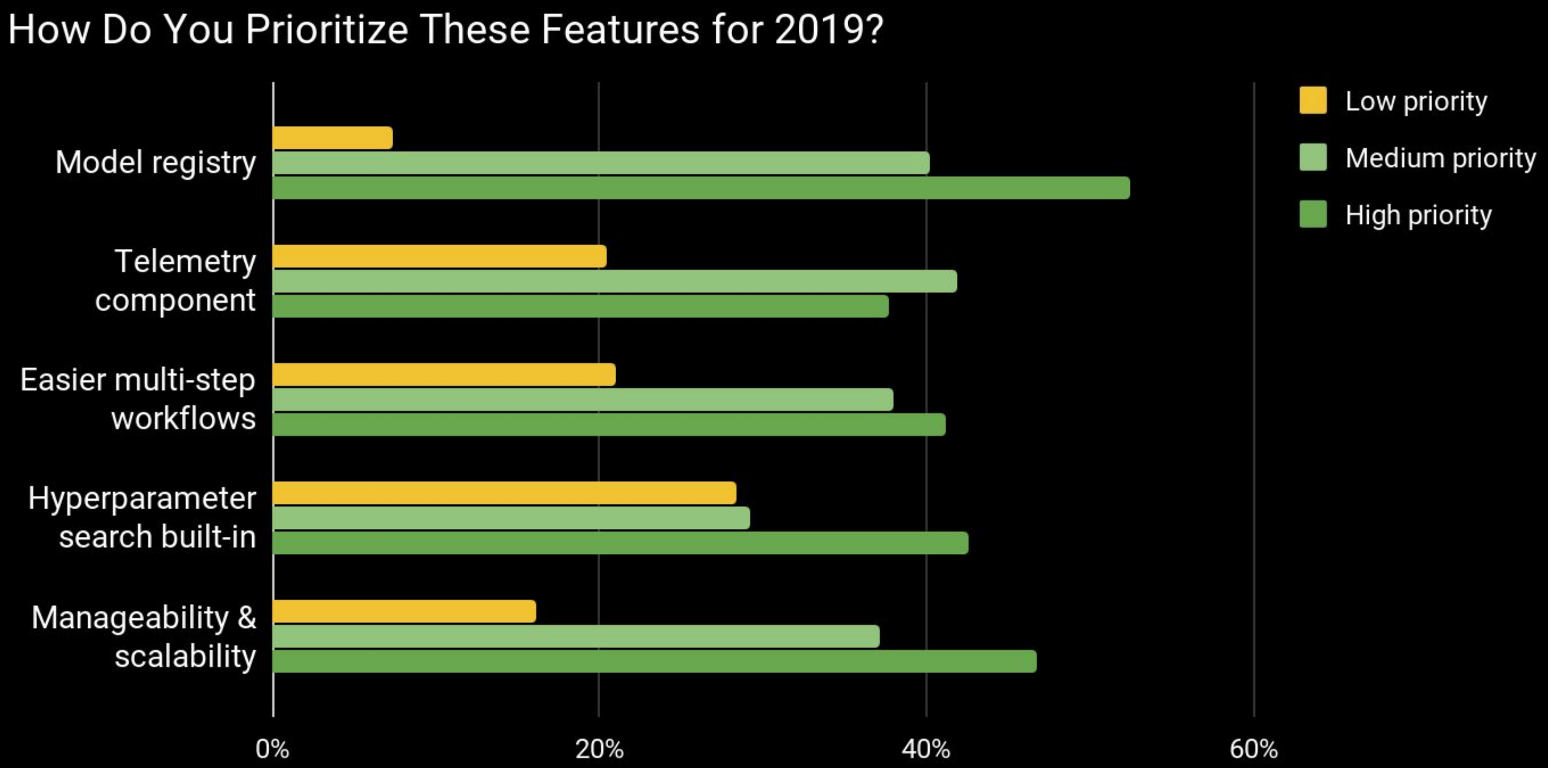
Email [optional]

Users are using all components (but Tracking most popular)



databricks

What users want to see next



databricks

Outline

MLflow overview

Feedback so far

Databricks’ development themes for 2019

Demos of upcoming features

High-Level Themes

1) Update existing components based on feedback

2) Stabilize the APIs and dev process (MLflow 1.0)

3) Add new features for more of the ML lifecycle

Rough Development Timeline

MLflow 0.9, 0.10, etc: in the next few months

MLflow 1.0 and API stabilization: end of April
(stabilize core APIs and mark others as experimental)

After 1.0: continue releasing regularly to get features out

Updating Existing Components

MLflow Tracking

- SQL database backend for scaling the tracking server (0.9)
- UI scalability improvements (0.8, 0.9, etc)
- X-coordinate logging for metrics & batched logging (1.0)
- Fluent API for Java and Scala (1.0)

Updating Existing Components

MLflow Projects

- Docker-based project environment specification (0.9)
- X-coordinate logging for metrics & batched logging (1.0)
- Packaging projects with build steps (1.0+)

Updating Existing Components

MLflow Models

- Custom model logging in Python, R and Java (0.8, 0.9, 1.0)
- Better environment isolation when loading models (1.0)
- Logging schema of models (1.0+)

New Components in Discussion

Model registry

- A way to name and manage models, track deployments, etc
- Could be new abstraction or tags on existing runs (need feedback!)

Multi-step workflow GUI

- UI to view or even edit multi-step workflows (do you want this?)

MLflow telemetry component

- Standard API for deployed models to log metrics wherever they run
- Data collection and analytics tools downstream (need feedback!)

Outline

MLflow overview

Feedback so far

Databricks’ development themes for 2019

Demos of upcoming features

Demo: Model Customization

Motivating example: MLflow flower classification

```
f(petal_attribs) -> classification
```

```
f(petal_attribs) -> probabilities
```

Demo: Model Customization

Motivation: ML teams want to capture mathematical models **and** business logic in a single MLflow model.

mlflow.sklearn.save_model,
mlflow.pytorch.log_model,
….

Demo: Model Customization

MLflow 0.9: Users can easily customize models, introducing inference logic and data dependencies

```
class PythonModel:

  def load_context(self, context):
    # The context object contains paths to
    # files (artifacts) that can be loaded here

  def predict(self, context, input_df):
    # Inference logic goes here
```

```
class ToyModel(mlflow.pyfunc.PythonModel):

  def __init__(self, return_value):
    self.return_value = return_value

  def predict(self, context, input_df):
    return self.return_value

mlflow.pyfunc.save_model(
  python_model=ToyModel(pd.DataFrame([42])),
  dst_path="toy_model")
```

```python
class ProbaModel(mlflow.pyfunc.PythonModel):

  def predict(self, context, input_df):
    sk_model = mlflow.sklearn.load_model(
      context.artifacts["sk_model"])
    return sk_model.predict_proba(input_df)

mlflow.pyfunc.save_model(
  dst_path="proba_model",
  python_model=ProbaModel(),
  artifacts={"sk_model": "s3://model/path"})
```

Demo: Model Customization

We will fit a model that identifies iris flowers based on their petals, emitting a probability distribution

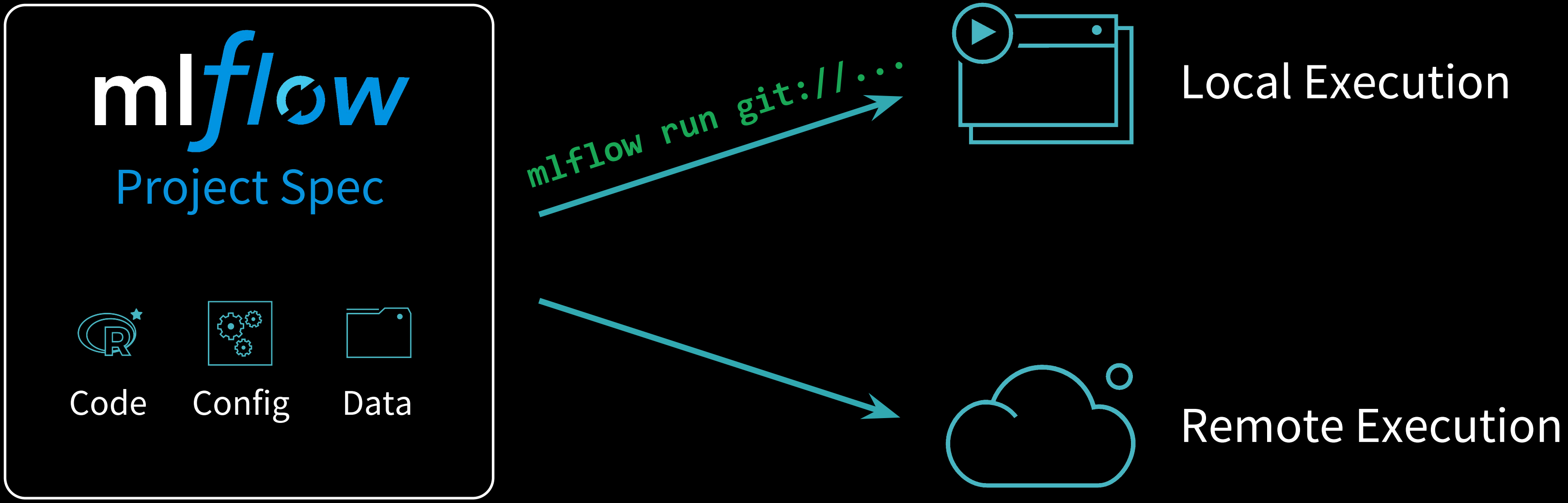
```
f(pwidth, plength) -> probabilities
```

↓

across 3 flower types

Demo

MLflow Projects



mlflow

Project Spec

Code

Config

Data

```
mlflow run git://...
```

Local Execution

Remote Execution

Demo: Docker-based Projects

MLflow 0.9: run projects in docker containers (@marcusrehm)

Package code with arbitrary dependencies (Java etc)

Run, share, track code with same MLflow APIs

Demo: Docker-based Projects

Docker handles the dependencies

```
docker_env:
  image:  continuumio/anaconda
```

MLflow provides unified interface for running code

```
$ mlflow run git://<my_project>
```

Project Structure

```
my_project/
├── MLproject
│
│
│
│
│
│
├── train.py
└── utils.py
     ...
```

```
docker_env:
  image:  continuumio/anaconda

entry_points:
  main:
    parameters:
      training_data: path
      lambda: {type: float, default: 0.1}
    command: python train.py {training_data} {lambda}
```

```
$ mlflow run git://<my_project>
```

Demo: Docker-based Projects

See example project at
github.com/mlflow/mlflow/tree/master/examples/docker

Demo

What’s next: Docker-based Projects

Remote execution (Kubernetes, Databricks) for horizontal, vertical scaleout

Ease-of-use improvements add custom Docker build steps, log to remote artifact stores

Thank You!

Get started with MLflow at mlflow.org

- Fill out our survey and join our Slack!

Spark AI Summit 15% discount: MLflowMeetup