

MLflow: A Platform for Productionizing Machine Learning

Matei Zaharia
Databricks and Stanford University

Outline

- Challenges using ML in industry
- ML Platforms as an emerging abstraction
- MLflow overview and interesting use cases

About Databricks

Data and ML cloud platform used by >7000 customers

- Millions of VMs and 100,000s of users

Some of our ML customers:



T-Mobile®

H&M

REGENERON

Adobe®



HSBC

VIACOM



National Institutes
of Health



Nationwide®

ML is Being Adopted for Critical Applications



Price insurance policies based on data and ML
(core business for >90 years!)



Manage inventory and supply chains
(affects company's entire cost/revenue)



Fraud detection & personalization on 170 PB data
(directly impacts profitability & compliance)

But ML is Different from Traditional Software

Traditional Software

Goal: meet a functional specification

Quality depends only on application code

Pick one software stack

Machine Learning

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

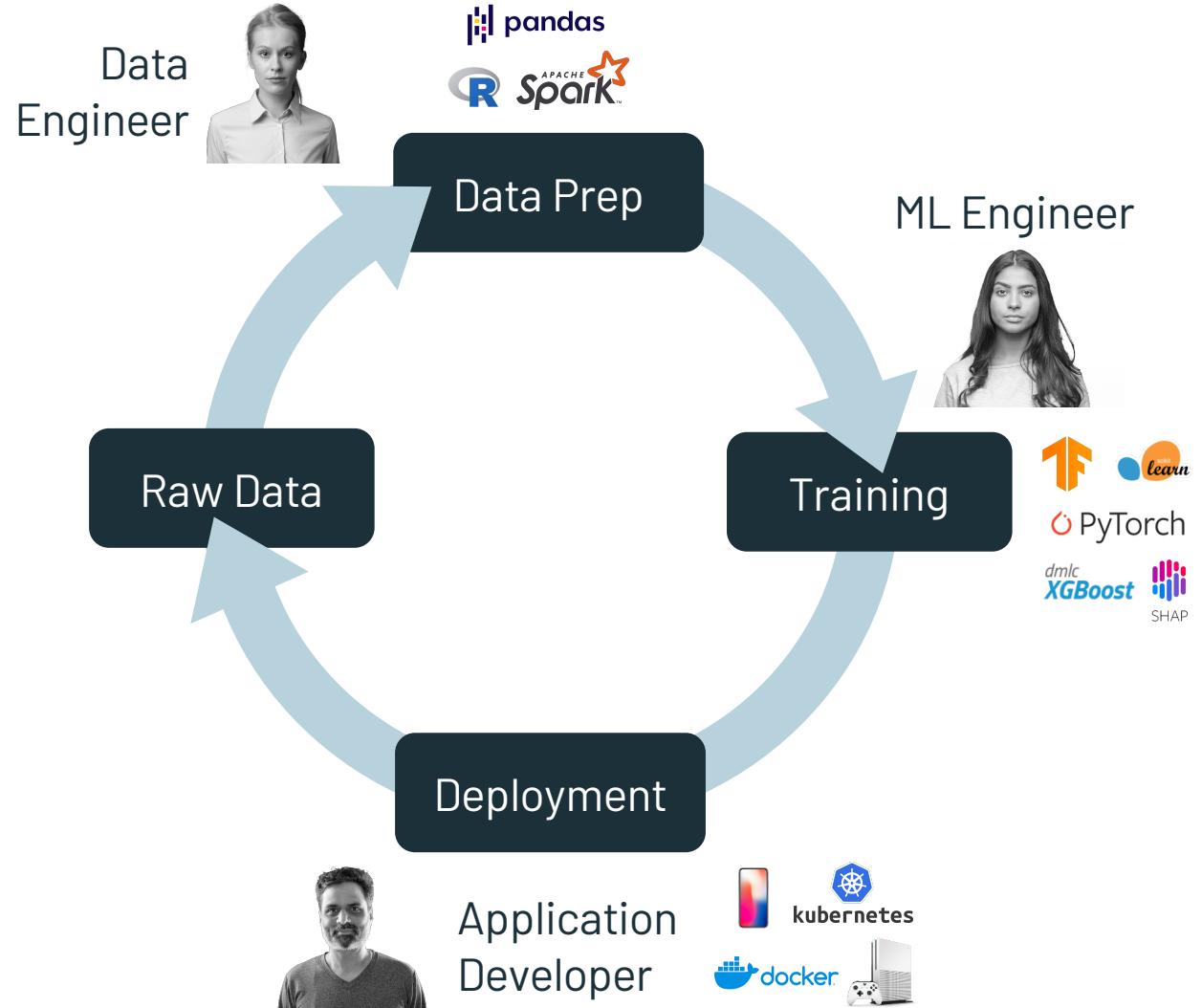
Quality depends on input data and tuning parameters

Compare + combine many libraries and algorithms for the same task

Production ML is Even Harder

ML apps must be fed new data to keep working

Design, retraining & inference often done by different people



A Solution is Emerging: ML Platforms

Software to manage the ML development and deployment process, from data to experimentation to production

Examples: Google TFX, Facebook FB Learner, Uber Michelangelo

Typical concerns:

- Data management
- Experiment management
- Model management
- Deployment for inference
- Reproducibility
- Testing & monitoring

All through a
consistent interface!

ML Platforms in 2018

Each company largely designing its own platform, with limited scope

- Specific libraries (e.g. TensorFlow for TFX)
- Specific deployment environment (e.g. Kubernetes on AWS)

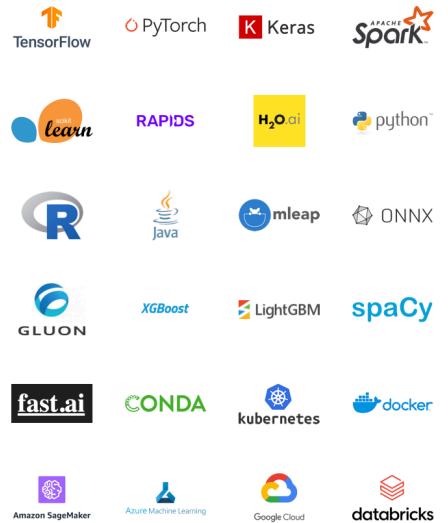
ML Platform team often becomes a bottleneck in the organization

Can we provide the same benefits with an open platform?

mlflow: An Open Source ML Platform

Based on an **open interface** design philosophy: make it easy to connect arbitrary ML code & tools into the platform

- Simple command-line and REST APIs rather than environment-specific
- Easy to add to existing software



mlflow Community

2 million downloads/month on PyPI

260 open source contributors

4x annual growth

1.5M runs/week on Databricks

Users and contributors:



Microsoft

R Studio

UNIVERSITY of WASHINGTON

hyperloop one

Booking.com



Zillow

SK telecom

accenture

criteo

Outreach



Telia Company

TEKION



Bering

SCRY ANALYTICS



WIX.com

ABRAXAS Intelligence



virtusa

Health 2 Sync



dL

SYSTEMATIC

despegar.com

ModelWorks



Brandfolder

ASML

fractaloo

Acerta

ENABLE AI
MASTER YOUR DATA

Linde

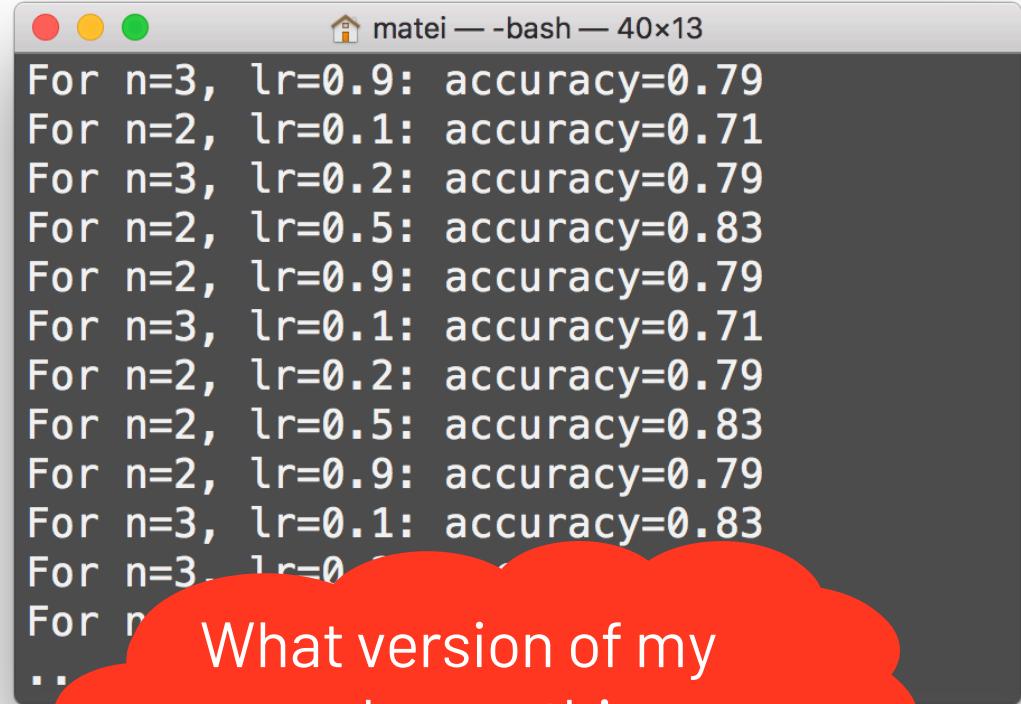
MLflow Components

MLflow Tracking

Get visibility into experiments and production runs

```
data    = load_text(file)
ngrams = extract_ngrams(data, N=n)
model   = train_model(ngrams,
                      learning_rate=lr)
score   = compute_accuracy(model)

print("For n=%d, lr=%f: accuracy=%f"
      % (n, lr, score))
```



A terminal window titled "matei — -bash — 40x13" displays a series of "For" loops. Each loop prints a configuration and its resulting accuracy. The configurations are: n=3, lr=0.9; n=2, lr=0.1; n=3, lr=0.2; n=2, lr=0.5; n=2, lr=0.9; n=3, lr=0.1; n=2, lr=0.2; n=2, lr=0.5; n=2, lr=0.9; n=3, lr=0.1. The accuracy values range from 0.71 to 0.83.

```
For n=3, lr=0.9: accuracy=0.79
For n=2, lr=0.1: accuracy=0.71
For n=3, lr=0.2: accuracy=0.79
For n=2, lr=0.5: accuracy=0.83
For n=2, lr=0.9: accuracy=0.79
For n=3, lr=0.1: accuracy=0.71
For n=2, lr=0.2: accuracy=0.79
For n=2, lr=0.5: accuracy=0.83
For n=2, lr=0.9: accuracy=0.79
For n=3, lr=0.1: accuracy=0.83
For n=3, lr=0.0: accuracy=0.79
For n=3, lr=0.1: accuracy=0.79
...
```

What version of my
code was this
result from?



MLflow Tracking

Get visibility into experiments and production runs

```
mlflow.keras.autolog()
```

```
data      = load_text(file)
ngrams   = extract_ngrams(data, N=n)
model    = train_model(ngrams,
                      learning_rate=lr)
score    = compute_accuracy(model)

# Or log custom info if desired
mlflow.log_param("country", "US")
```

The screenshot shows the MLflow UI interface. At the top, there's a header with the MLflow logo, GitHub, and Docs links. Below the header, the title 'Language Model' is displayed, along with 'Experiment ID: 0' and 'Artifact Location: /Users/matei/mlflow/mlruns/0'. There are search and filter fields for 'Search Runs' (containing 'metrics.rmse < 1 and params.model = "tree"') and 'Filter Params' (containing 'alpha, lr'). The 'Filter Metrics' field contains 'rmse, r2'. A 'Clear' button is also present. Below these controls, it says '10 matching runs' with buttons for 'Compare', 'Delete', and 'Download CSV'. The main area is a table with columns: Date, User, Source, Version, input_file, lr, n, accuracy, and f1. The data rows are:

Date	User	Source	Version	input_file	lr	n	accuracy	f1
2018-10-02 21:53:57	matei	lang_model.py	e55d56	data.txt	2.0	1	0.77	0.704
2018-10-02 21:53:55	matei	lang_model.py	e55d56	data.txt	2.0	4	0.835	0.609
2018-10-02 21:53:48	matei	lang_model.py	e55d56	data.txt	2.0	2	0.66	0.476
2018-10-02 21:53:53	matei	lang_model.py	e55d56	data.txt	1.0	1	0.663	0.468
2018-10-02 21:53:49	matei	lang_model.py	e55d56	data.txt	1.0	4	0.902	0.461

Track parameters, metrics,
output files & code version

Tracking UI: Inspecting Runs

mlflow

[GitHub](#) [Docs](#)

Language Model

Experiment ID: 0 Artifact Location: /Users/matei/mlflow/mlruns/0

Search Runs: metrics.rmse < 1 and params.model = "tree" State: Active ▾

Filter Params: alpha, lr Filter Metrics: rmse, r2

10 matching runs

<input type="checkbox"/>	Date ▾	User	Source	Version	Parameters			Metrics	
					input_file	lr	n	accuracy	f1
<input type="checkbox"/>	2018-10-02 21:53:57	matei	<input type="button" value="lang_model.py"/>	e55d56	data.txt	2.0	1	0.77	0.704
<input type="checkbox"/>	2018-10-02 21:53:56	matei	<input type="button" value="lang_model.py"/>	e55d56	data.txt	1.0	2	0.254	0.222
<input type="checkbox"/>	2018-10-02 21:53:55	matei	<input type="button" value="lang_model.py"/>	e55d56	data.txt	2.0	4	0.835	0.609
<input type="checkbox"/>	2018-10-02 21:53:53	matei	<input type="button" value="lang_model.py"/>	e55d56	data.txt	1.0	1	0.663	0.468
<input type="checkbox"/>	2018-10-02 21:53:52	matei	<input type="button" value="lang_model.py"/>	e55d56	data.txt	0.2	4	0.034	0.032
<input type="checkbox"/>	2018-10-02 21:53:51	matei	<input type="button" value="lang_model.py"/>	e55d56	data.txt	0.1	4	0.177	0.16

Tracking UI: Comparing Runs

mlflow

[GitHub](#) [Docs](#)

Language Model

Experiment ID: 0 Artifact Location: /Users/matei/mlflow/mlruns/0

Search Runs: metrics.rmse < 1 and params.model = "tree" [Filter](#) State: Active ▾ [Search](#)

Filter Params: alpha, lr [Clear](#) Filter Metrics: rmse, r2

10 matching runs [Compare](#) [Delete](#) [Download CSV](#)

	Date ▾	User	Source	Version	Parameters			Metrics	
					input_file	lr	n	accuracy	f1
<input type="checkbox"/>	2018-10-02 21:53:57	matei	lang_model.py	e55d56	data.txt	2.0	1	0.77	0.704
<input type="checkbox"/>	2018-10-02 21:53:56	matei	lang_model.py	e55d56	data.txt	1.0	2	0.254	0.222
<input type="checkbox"/>	2018-10-02 21:53:55	matei	lang_model.py	e55d56	data.txt	2.0	4	0.835	0.609
<input type="checkbox"/>	2018-10-02 21:53:53	matei	lang_model.py	e55d56	data.txt	1.0	1	0.663	0.468
<input type="checkbox"/>	2018-10-02 21:53:52	matei	lang_model.py	e55d56	data.txt	0.2	4	0.034	0.032
<input type="checkbox"/>	2018-10-02 21:53:51	matei	lang_model.py	e55d56	data.txt	0.1	4	0.177	0.16

MLflow Projects

Package code + dependencies for reusable workflows

```
my_project/  
  └── MLproject  
  
  ├── conda.yaml  
  ├── main.py  
  ├── model.py  
  └── ...
```

```
conda_env: conda.yaml  
  
entry_points:  
  main:  
    parameters:  
      training_data: path  
      lr: {type: float, default: 0.1}  
    command: python main.py {training_data} {lr}
```

```
$ mlflow run git://<my_project>  
mlflow.run("git://<my_project>", ...)
```

Composing Projects

```
r1 = mlflow.run("ProjectA", params)

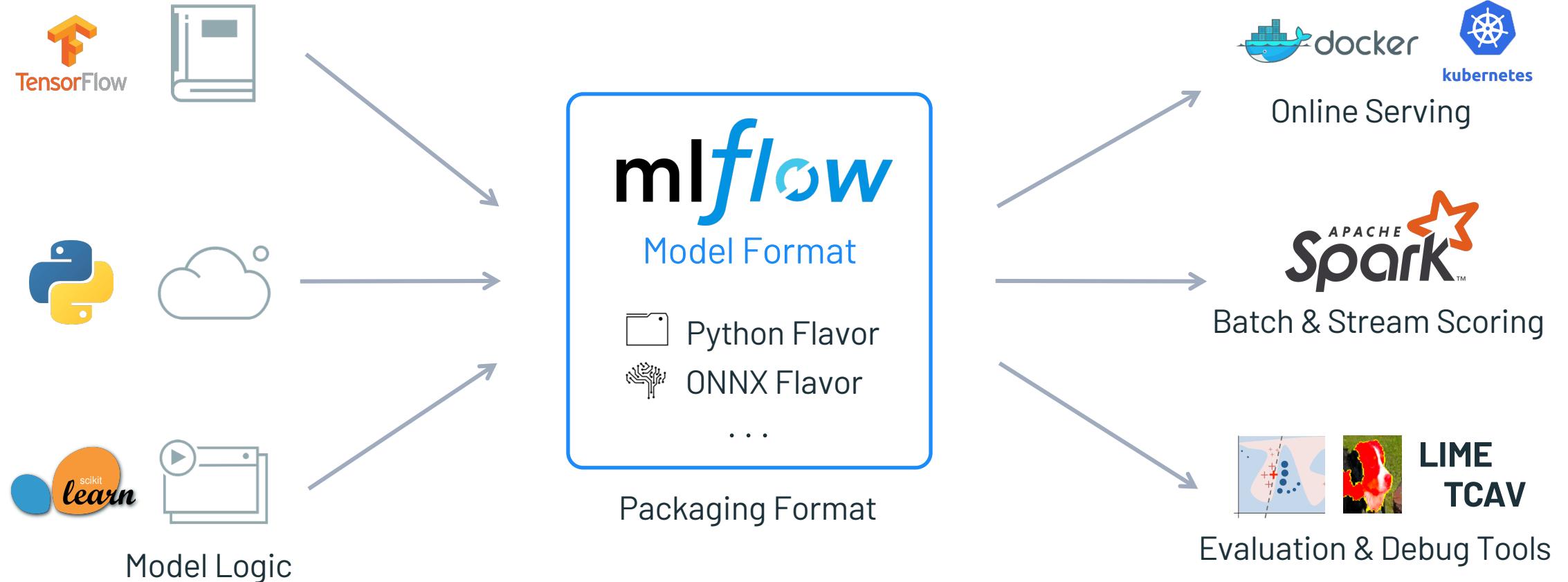
if r1 > 0:
    r2 = mlflow.run("ProjectB", ...)
else:
    r2 = mlflow.run("ProjectC", ...)

r3 = mlflow.run("ProjectD", r2)
```

Date	User	batch_size	epochs	lr	Parameters
[+ 2018-12-07 12:14:08]	matei	512		0.05	
[− 2018-12-07 12:14:05]	matei	512		0.05	
2018-12-07 12:14:05	matei		20		
2018-12-07 12:14:05	matei		40		
2018-12-07 12:14:05	matei		40		
[+] 2018-12-07 12:12:23	matei	512		0.05	

MLflow Models

Generic format to package & deploy models from any library



Example MLflow Model

```
my_model/  
└── MLmodel
```

```
run_id: 769915006efd4c4bbd662461  
time_created: 2018-06-28T12:34  
flavors:  
    tensorflow:  
        saved_model_dir: estimator  
        signature_def_key: predict  
    python_function:  
        loader_module: mlflow.tensorflow
```

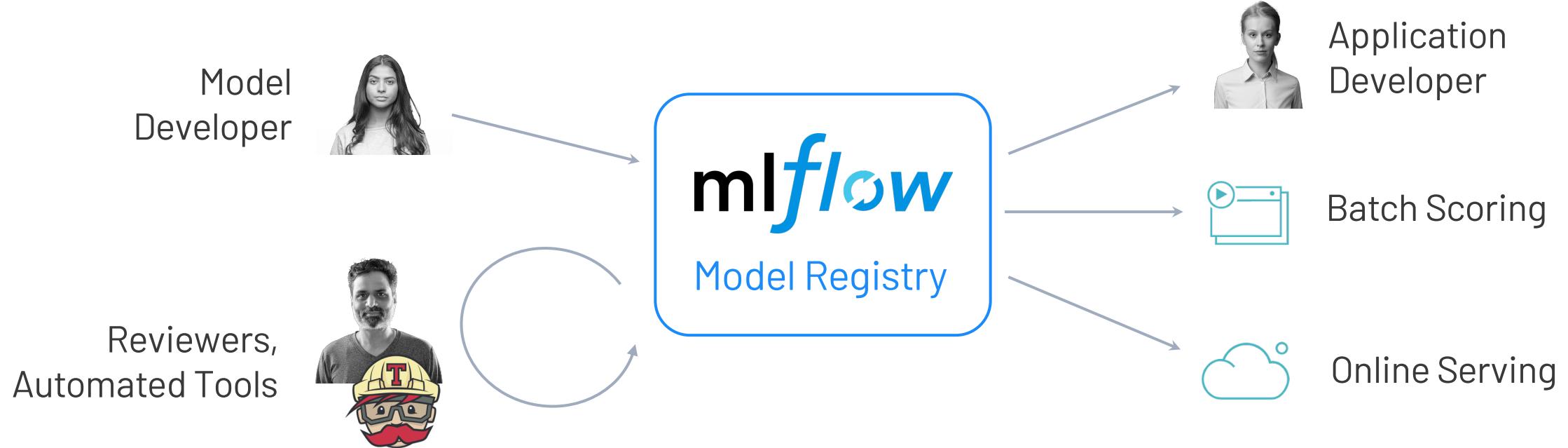
} Usable by tools that understand TensorFlow model format
} Usable by any tool that can run Python (Docker, Spark, etc)

```
estimator/  
└── saved_model.pb  
variables/  
...  
...
```

```
$ mlflow pyfunc serve -r <run_id>  
  
spark_udf = pyfunc.spark_udf(<run_id>)
```

MLflow Model Registry

GitHub-like environment for managing and reviewing models





databricks



Home



Workspace



Recents



Data



Clusters



Jobs



Models



Search



Registered Models > Airline_Delay_SparkML ▾

Created Time: 2019-10-10 15:20:29

Last Modified: 2019-10-14 12:17:04

▼ Description

Predicts airline delays (in minutes) using the best Spark RF model from the AutoML Toolkit.

▼ Versions

[All](#)[Active\(1\)](#)

Version	Registered at	Created by	Stage	Pending Requests
Version 1	2019-10-10 15:20:30	clemens@demo.com	Archived	—
Version 2	2019-10-10 21:47:29	clemens@demo.com	Archived	—
Version 3	2019-10-10 23:39:43	clemens@demo.com	Production	—
Version 4	2019-10-11 09:55:29	clemens@demo.com	None	—
Version 5	2019-10-11 12:44:44	matei@demo.com	Staging	1



databricks



Home



Workspace



Recents



Data



Clusters



Jobs



Models



Search

Registered Models > Airline_Delay_SparkML > Version 5 ▾

Registered At: 2019-10-11 12:44:44

Creator: clemens@demo.com

Stage: Staging ▾

Last Modified: 2019-10-14 12:19:32

Source Run: [Run 6151fe768a5e49d39076b07448e60d57](#)Request transition to → NoneRequest transition to → ProductionRequest transition to → ArchivedTransition to → NoneTransition to → ProductionTransition to → Archived

▼ Description

Improved the Airline delay model using a GBDT. See run for improved metrics.

▼ Pending Requests

Request	Request by	Actions
Transition to → Production	matei@demo.com	Approve Reject

▼ Activities

clemens@demo.com rejected a stage transition → None 5 minutes ago

matei@demo.com applied a stage transition None → Staging 4 minutes ago

matei@demo.com requested a stage transition Staging → Production 4 minutes ago

Tested this offline, looks good to launch!



Tag and Search APIs for Automated CI/CD

Tags to track custom metadata for a model version, e.g. test results

Search API to automate model management and MLOps actions

The screenshot shows the mlflow interface for managing model versions. The top navigation bar includes 'mlflow', 'Experiments', and 'Models'. The 'Models' tab is selected, showing 'Models > KNN > Version 12'. Below this, detailed information is provided: 'Created at: 2018-12-04 17:11:06', 'User: test@example.com', 'Stage: Staging', 'Last Modified: 2018-12-04 17:11:06', and 'Source: Run 123'. A sidebar on the left lists 'Notes', 'Pending Requests', and 'Tags'. The 'Tags' section is expanded, showing two entries in a table:

Name	Value	Actions
passed-gdpr-review	true	
passed-performance-test	true	

A red arrow points to the delete icon for the second tag entry ('passed-performance-test'). At the bottom of the tags section is a blue 'Add Tag' button. The 'Activity' section below shows two recent events: 'Alice requested a stage transition' (None → Staging) and 'Carol approved a stage transition' (None → Staging), both occurring 10 hours ago.

How are Companies Using MLflow?

"Expected" Use Cases

- Track experiments during model design
- Track performance of continuous training & deployment pipelines
- Deploy the same model for batch and real-time scoring
- Run pipelines deterministically in different environments
- CI/CD using Model Registry stages and APIs

Example Use Cases

••T••Mobile•

Manage the models for ad fraud detection, including monitoring for drift in over 200 metrics.



Let data scientists spend 70%-90% of their time on model development instead of tuning and monitoring.



ABN·AMRO

Automated and consistent deployment of 100+ models, from fraud detection, to marketing, to logistics

Interesting Use Case: Massive # of Models

Company wants to train a separate model for each {facility, customer, chemical processing machine, ...}

- Avoid interference across these entities
- Preserve privacy & compliance

Examples:



Predictive maintenance



Energy grid

Enterprise Software
Company

Per-customer models

Interesting Use Case: Massive # of Models

Company wants to train a separate model for each {facility, customer, chemical processing machine, ...}

- Avoid interference across these entities
- Preserve privacy & compliance

Solution: “hands-free” ML with large-scale analytics

- Train millions of models in parallel using an AutoML library on each entity
- Query experiment metrics using analytics tools (e.g. MLflow -> Pandas API)
- Run online or batch inference with the models

Interesting Use Case: Experiments Beyond ML

Systems like MLflow can add structure to other experimentation tasks



Develop, review & publish visualizations for COVID-19 data using the MLflow Model Registry



Tune hundreds of Hyperloop engineering design parameters in simulation to optimize efficiency

Interesting Use Case: Reproducibility & Explainability

Government regulators and highly-regulated companies want:

- Documentation of every piece of data and code that went into a result (many have built their own lineage systems for this)
- Explanation of models
- AutoML to demonstrate they employed best practices

Environment Reproducibility on Databricks

The screenshot shows the Databricks interface with a sidebar containing icons for Home, Workspace, Recent, Data, Cluster, Jobs, Model, and Search. The main area displays a 'Default Experiment > Run cb55fd06262448ad99b08cb5e9dd2e98' page. A 'Clone Run' dialog box is open, titled 'Clone Run'. It contains instructions: 'You can clone a run to recreate and iterate on the original notebook, cluster, and environment used to execute the run.' Below this, there are three options with checkboxes:

- Clone notebook: 'Clone Risk Model Training' to '/users/myfolder' (with 'Edit Folder' button)
- Attach to the cluster: 'Shared autoscaling' (with 'View Spec' button)
- Inject the environment set-up into the notebook (with 'View Environment' button)

At the bottom of the dialog are 'Cancel' and 'Clone' buttons. The background shows experiment parameters like 'Name: run_origin', 'Max_iter: 100', 'param: 0.01', and 'Tags: Name: version.sklearn, Value: 0.20.2'.

Recreate exact configuration of an experiment run

Recently Added Features

mlflow Tracking for Data Versioning

Spark and Delta Lake auto-logging: track data sources read and data versions in Delta Lake

```
with mlflow.start_run(run_name='keras'):
    # log model and datasource
    mlflow.keras.autolog()
    mlflow.spark.autolog()
```

Tags

Name	Value	Actions
sparkDatasourceInfo	path=dbfs:/delta/clemens_windfarm,version=2,format=delta	 



To reload data version:

```
df = spark.read.format("delta") \
    .option("versionAsOf", 2) \
    .load("/delta/clemens_windfarm")
```

mlflow Tracking for Model Schemas

Record what fields are consumed & produced by the model to prevent data mismatches

```
with mlflow.start_run(run_name='keras'):
    # log model and datasource
    mlflow.keras.autolog()
    mlflow.spark.autolog()

    sig = infer_signature(X_train, y_train)
```

Schema	
Name	Type
Inputs (7)	
user_purchases_7d	long
user_purchases_14d	long
user_location	string
user_last_login	string
item_inventory	long
Outputs (1)	
prediction	double

mlflow Tracking for Interpretability

SHAP library feature importances and visualizations

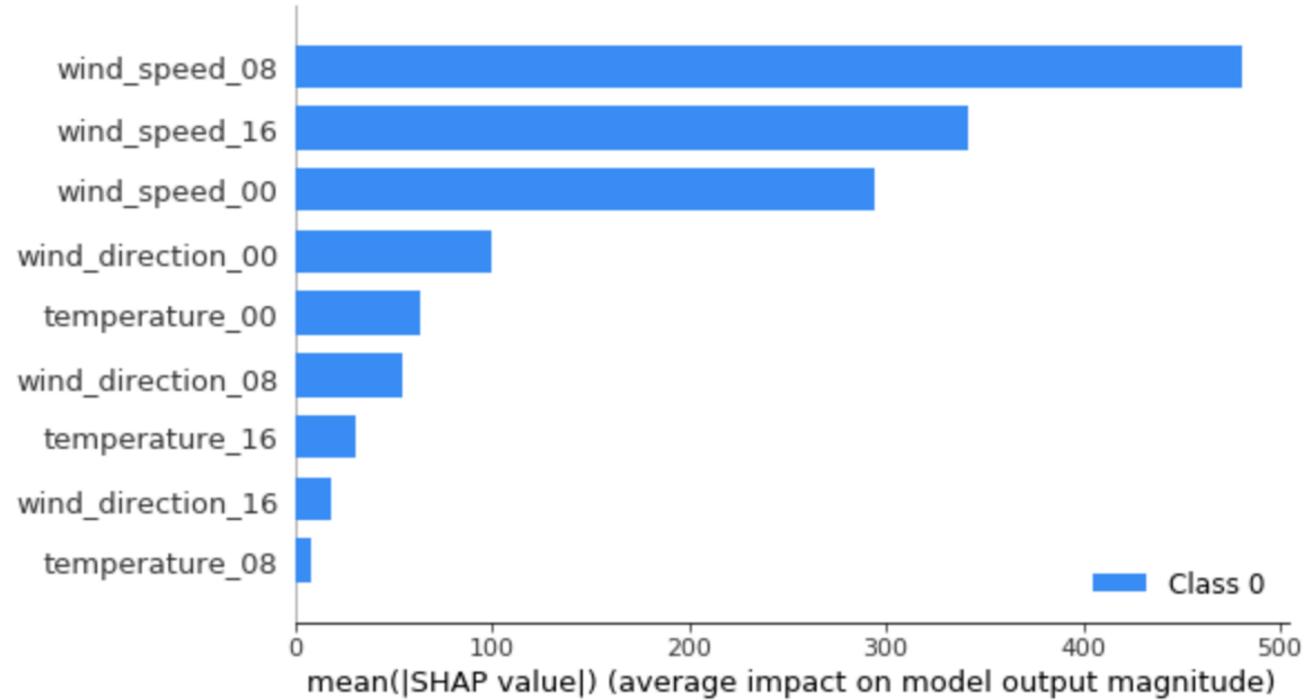
```
with mlflow.start_run(run_name='keras'):
    # log model and datasource
    mlflow.keras.autolog()
    mlflow.spark.autolog()

    sig = infer_signature(X_train, y_train)

    mlflow.shap.log_explanation(model, X_train[:100])
```



SHAP



just released

mlflow + PyTorch

Collaboration between Facebook and Databricks to bring ML platform features to PyTorch

- MLflow autologging for PyTorch Lightning
- TorchScript support for faster packaged models
- Model deployment to TorchServe

Many Open Questions in ML Platforms!

- How to design “feature stores” that can hold fast-changing data about entities to be used in training and inference?
- How to automatically detect performance degradation in models?
- What kind of information should be tracked at inference time?

Conclusion

Machine learning is being applied to critical problems in industry, but requires careful management when the stakes are high

ML Platforms are an emerging abstraction to help with this

- And we believe an “open-interface” design is very important for usability

Many open problems, especially as needs evolve