

# MLflow with R

Javier Luraschi

November 2018

# Overview

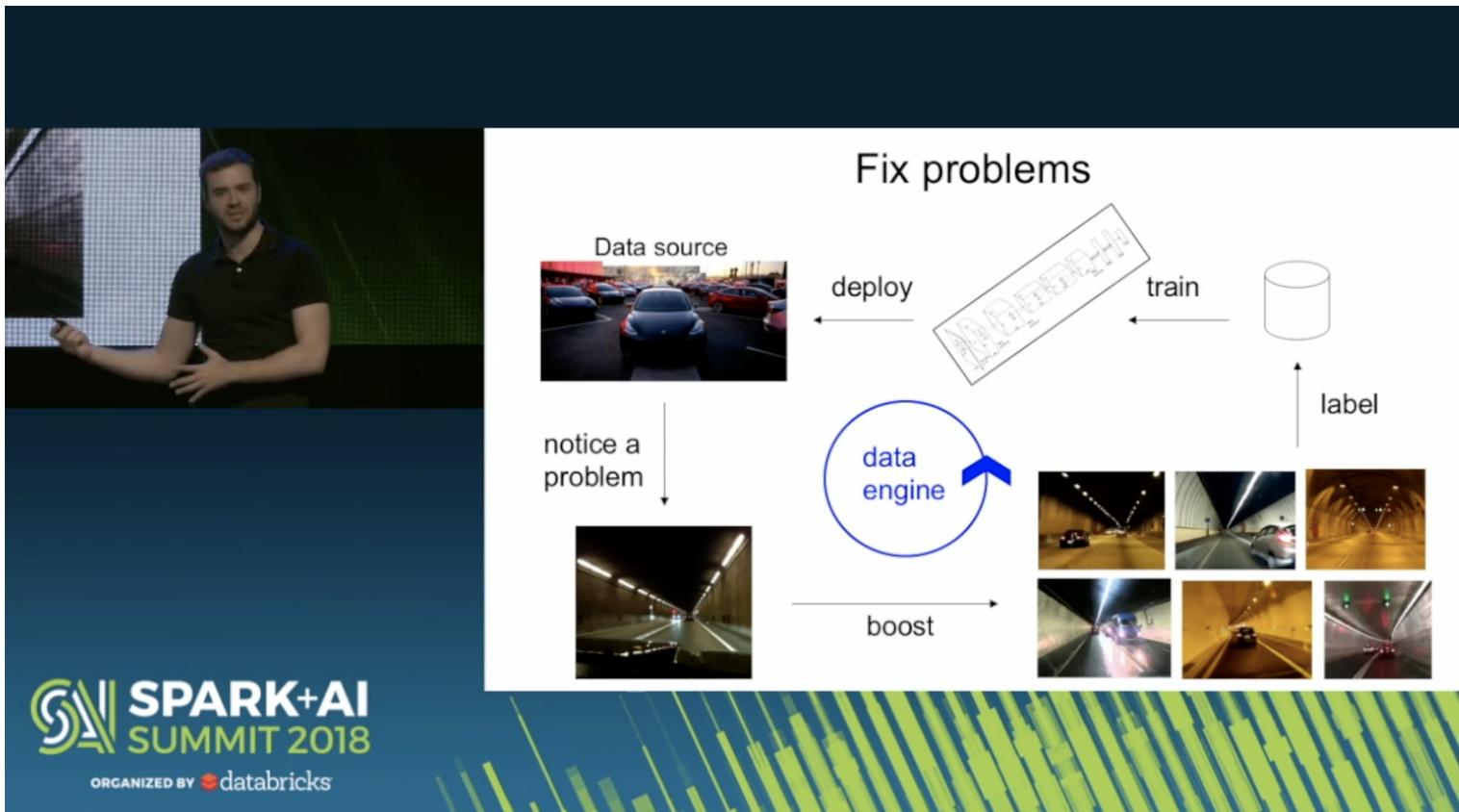
- What is MLflow?
- MLflow with R

# What is MLflow?

# Background

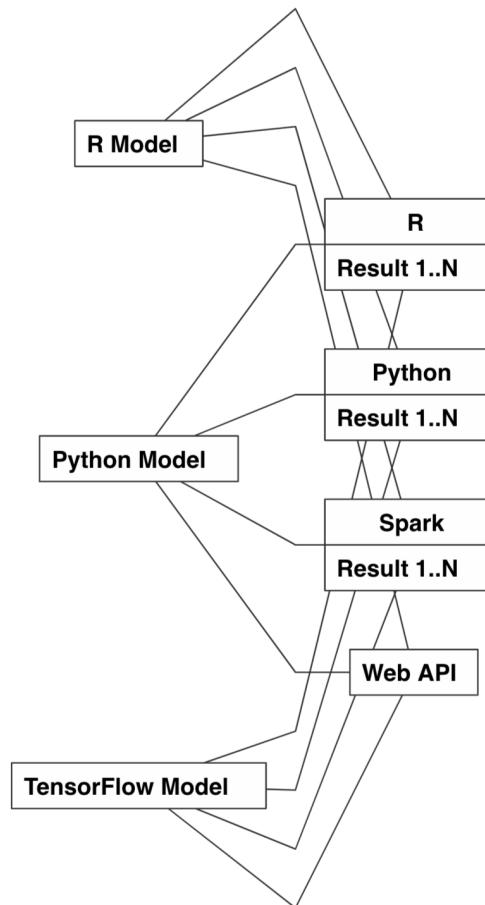
Spark Summit from Andrej Karpathy at Tesla

*The toolchain for the (software) 2.0 tack does not exist.*



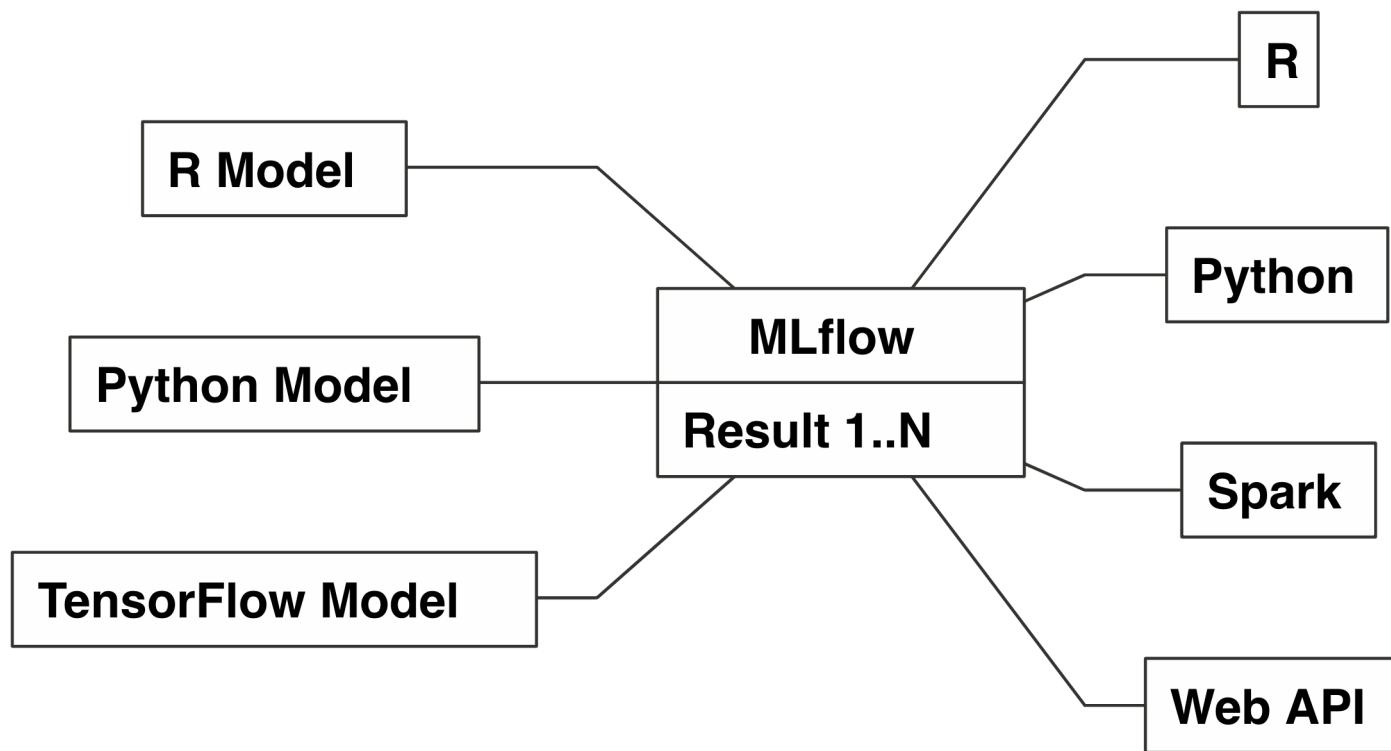
# Challenge

Different models need to be run across many runtimes each producing many results.



# Solution

MLflow can run models across runtimes tracking their results.



# MLflow

*“Helps teams manage their machine learning lifecycle.”*

- **Tracking** : Track experiments to record and compare params and results.
- **Projects** : Reuse and reproduce code to share or transfer to production.
- **Models** : Manage and deploy models from across libraries and platforms.

# MLflow with R

# Principles

- Parity with Python API.
- Designed for the R user.

# Installing

*Install Anaconda or miniconda.*

```
install.packages("mlflow")
mlflow::mlflow_install()
```

# Tracking

# Tracking - Implicit

Implicit MLflow run:

```
library(mlflow)

# Log a parameter (key-value pair)
mlflow_log_param("param1", 5)

# Log a metric; metrics can be updated throughout the run
mlflow_log_metric("foo", 1)
mlflow_log_metric("foo", 2)
mlflow_log_metric("foo", 3)

# Log an artifact (output file)
writeLines("Hello world!", "output.txt")
mlflow_log_artifact("output.txt")
```

Run terminates when the R session finishes or by running:

```
mlflow_end_run()
```

Useful when sourcing files.

# Tracking - Explicit

Explicit MLflow run:

```
library(mlflow)

with(mlflow_start_run(), {
    # Log a parameter (key-value pair)
    mlflow_log_param("param1", 5)

    # Log a metric; metrics can be updated throughout the run
    mlflow_log_metric("foo", 1)
    mlflow_log_metric("foo", 2)
    mlflow_log_metric("foo", 3)

    # Log an artifact (output file)
    writeLines("Hello world!", "output.txt")
    mlflow_log_artifact("output.txt")
})
```

# Tracking - Sources

```
mlflow_run("R/tracking.R")
```

Or adding the following to `tracking.R` in RStudio 1.2:

```
# !source mlflow::mlflow_run(entry_point = .file)
```

The screenshot shows the RStudio interface with the following details:

- Code Editor:** The main pane displays the `tracking.R` script. The code includes comments explaining the use of `mlflow_log_param`, `mlflow_log_metric`, and `mlflow_log_artifact` functions.
- Console:** Below the editor, the console output shows the command `> mlflow::mlflow_run(entry_point = "R/tracking.R")` being run, followed by logs indicating artifact logging and the creation of a temporary directory for remote URIs.
- Environment:** The right-hand sidebar shows the global environment with variables like `alpha`, `downloads...`, `idx`, `lambda`, and `mae`.
- File Explorer:** The sidebar also displays the project structure, including files like `.Rhistory`, `images`, `mlflow.Rproj`, `output.txt`, `R`, `slides.html`, `slides.Rmd`, `styles`, `mlruns`, `.gitignore`, and `rsconnect`.

# Tracking - UI

mlflow\_ui()

The screenshot shows the mlflow UI within an RStudio Viewer window. The title bar says "RStudio Viewer". The main header has "mlflow" on the left and "GitHub Docs" on the right. Below the header, there's a breadcrumb navigation "Experiments < Default". A sub-header "Default" is highlighted with a light blue background. To the right of the sub-header, it says "Experiment ID: 0" and "Artifact Location: /Users/javierluraschi/RStudio/talks/mlflow/mlruns/0". There are search and filter fields: "Search Runs: metrics.rmse < 1 and params.model = "tree"" with a "Search" button, and "Filter Params: alpha, lr" and "Filter Metrics: rmse, r2" with a "Clear" button. Below these, it says "1 matching run" with buttons for "Compare Selected" and "Download CSV". A table follows, with columns: Date, User, Source, Version, Parameters, and Metrics. One row is shown, corresponding to the experiment run: Date is "2018-09-19 17:12:02", User is "javierluraschi", Source is "talks#mlflow:R/tracking.R", Version is "a26eb2", Parameters is "param1", and Metrics is "foo".

# Projects

# Projects - Snapshots

Create dependencies snapshot:

```
mlflow_snapshot()
```

Then restore snapshot:

```
mlflow_restore_snapshot()
```



# Projects - Consuming R from R

```
mlflow_run(  
    "train.R",  
    "https://github.com/rstudio/mlflow-example",  
    param_list = list(alpha = 0.2)  
)
```

```
Elasticnet model (alpha=0.2, lambda=0.5):  
RMSE: 0.827574750159859  
MAE: 0.632070002076146  
R2: 0.227227498131926
```

# Projects - Consuming Python from R

```
mlflow_run(  
    uri = "https://github.com/mlflow/mlflow-example",  
    param_list = list(alpha = 0.2)  
)
```

```
Elasticnet model (alpha=0.200000, l1_ratio=0.100000):  
RMSE: 0.7836984021909766  
MAE: 0.6142020452688988  
R2: 0.20673590971167466
```

# Projects - Consuming R from Bash

Or from bash:

```
mlflow run --entry-point train.R https://github.com/rstudio/mlflow-example
```

```
Elasticnet model (alpha=0.5, lambda=0.5):
RMSE: 0.828684594922867
MAE: 0.627503954965052
R2: 0.19208126758775
```

# Models

# R Models - Saving from R

```
mlflow_save_model(model)
```

Generic functions can be serialized with `carrier :: crate`:

```
model <- lm(mpg ~ . -mpg, mtcars)

mlflow_save_model(
    crate(~ stats::predict(model, .x), model)
)
```

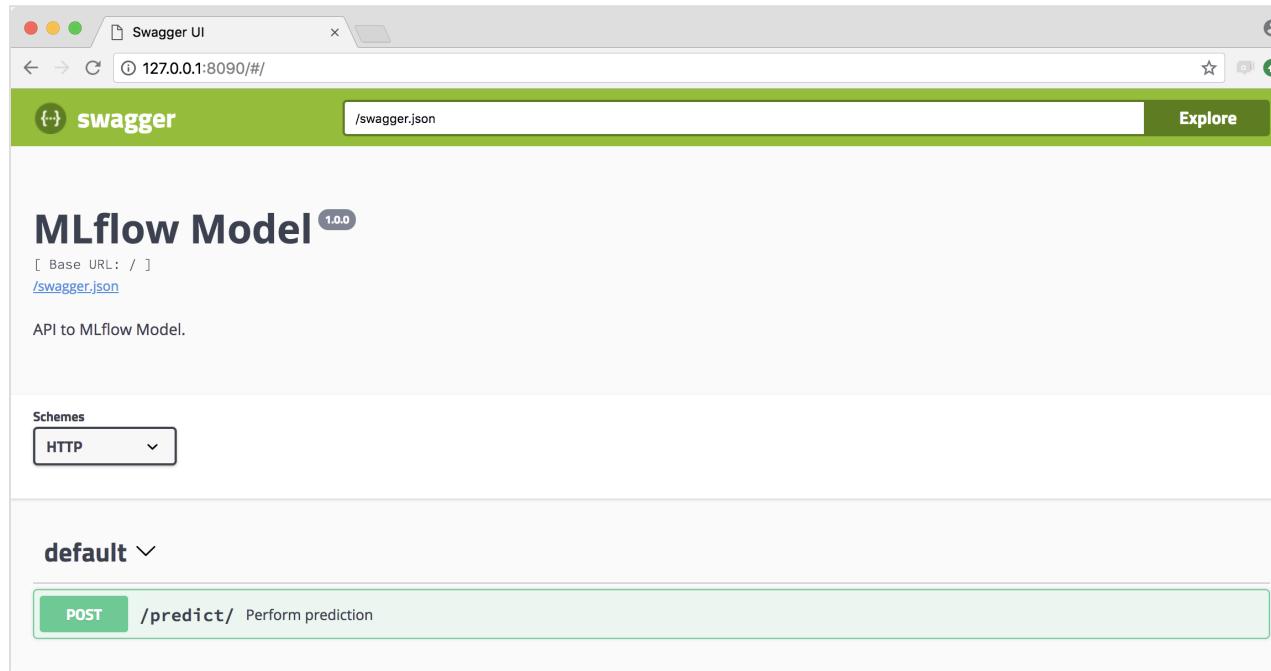
# Models - Predictions from R

```
mlflow_rfunc_predict(  
    "model",  
    data = data.frame(cyl = 2, disp = 160, hp = 110, drat = 3.9, wt = 2.62,  
                      qsec = 16.46, vs = 0, am = 1, gear = 4, carb = 4)  
)
```

```
1  
23.04527
```

# Models - Serving from R

```
mlflow_rfunc_serve("model")
```



```
mlflow rfunc serve model-path model
```

```
curl -X POST "http://127.0.0.1:8090/predict/" -H "accept: application/json" -H "Content-Type: application/json" -d '[{"cyl":2,"disp":160,"hp":110,"drat":3.9,"wt":2.62,"qsec":16.46,"vs":0,"am":1,"gear":4,"carb":4}]'
```

# Models - Predict from Bash

Or from bash,

```
mlflow rfunc predict --model-path model --input-path data.csv
```

# Keras Models - Saving

However, `mlflow_save_model()` can be extended by packages:

```
library(keras)

cars <- scale(mtcars)
model <- keras_model_sequential() %>%
  layer_dense(units = 8, activation = "relu", input_shape = ncol(mtcars) - 1) %>%
  layer_dense(units = 4, activation = "relu") %>%
  layer_dense(units = 1, activation = "relu") %>%
  compile(
    loss = "mse",
    optimizer = optimizer_rmsprop(),
    metrics = "mean_absolute_error"
  )

fit(model, cars[, -1], cars[, 1], epochs = 10)
mlflow_save_model(model, "keras")
```

# Keras Models - Prediction

```
mlflow_rfunc_predict(  
    "keras",  
    data = data.frame(cyl = 1, disp = 1, hp = 1, drat = 1, wt = 1,  
                      qsec = 1, vs = 1, am = 1, gear = 1, carb = -1)  
)
```

```
[,1]  
[1,] 0.02378437
```

# Keras Models - Serving

```
mlflow_rf_func_serve("keras")
```

```
[{"cyl":1,"disp":1,"hp":1,"drat":1,"wt":1,"qsec":1,"vs":1,"am":1,"gear":1,"carb":-1}]
```

```
{"predictions": [[0.0238]]}
```

# Models - Prediction in Spark

```
library(sparklyr)
sc <- spark_connect(master = "local")
mtcars_tbl <- sdf_copy_to(sc, mtcars)

model <- mlflow_load_model("model")

mtcars_tbl %>% spark_apply(function(x, model) model(x), context = model)
```

```
# Source: spark<?> [?? x 1]
  result
*  <dbl>
1  22.6
2  22.1
3  26.3
4  21.2
5  17.7
```

**Thanks!**

# Resources

*mlflow.org*

*github.com/mlflow/mlflow*

*rpubs.com/jluraschi/mlflow-rseattle*

*@javierluraschi*

*javier@rstudio.com*