Model Review

improving transparency, reproducibility & knowledge sharing with MLflow

Jes Ford, PhD
Senior ML Engineer at Cash App (Block)

Hi, Jes Ford here 👋

- Snowboarder 1st 🏂 ⛰️
- BS in Physics at the University of Nevada, Reno
- PhD in Physics 🔭 at the University of British Columbia 🇨🇦
- Postdoc in Data Science at the University of Washington

Hi, Jes Ford here 👋

- Data Scientist at Backcountry ⛺
- Senior Data Scientist at Recursion 🧬
- Senior Machine Learning Engineer (Modeling) at Cash App 📱 (Block)
- Working on Natural Language Understanding for Customer Support

Survey

About this talk

- Why did my team decide to adopt a process for Model Review?
- What does Model Review mean?
- Our primary tool for review: MLflow
- Intro to MLflow
- How we are using MLflow to solve some of our problems

Motivation

Our team had recently doubled in size and was deploying lots of models.

But we had no good record keeping on what models exactly were in production, how they were trained, or what tricks or approaches were working well...

Motivation

What if a model needed to be retrained?

A new team member wants to build off the work that came before them?

What was the precision on that model supposed to be anyway?

Motivation

What if a model needed to be retrained?

A new team member wants to build off the work that came before them?

What was the precision on that model supposed to be anyway?

uhhh, let me see if I can find that notebook...

Goals for a new Model Review Process

Goals for a new Model Review Process

1. **Transparency** and record keeping of what exactly is being deployed

Goals for a new Model Review Process

1. **Transparency** and record keeping of what exactly is being deployed

1. **Reproducibility** of past experiments and ease of building off of them

Goals for a new Model Review Process

1. **Transparency** and record keeping of what exactly is being deployed

1. **Reproducibility** of past experiments and ease of building off of them

1. **Knowledge Sharing** so we can learn from each other and new teammembers can get up to speed

Goals for a new Model Review Process

1. **Transparency** and record keeping of what exactly is being deployed

1. **Reproducibility** of past experiments and ease of building off of them

1. **Knowledge Sharing** so we can learn from each other and new teammembers can get up to speed

ALSO: We need to **automate** as much of this as possible!

Comparison to Code Review

Why do we Review *Code*?

- More 👀 on code to spot bugs and potential issues
- Pull Requests create a record of changes and also (ideally) documentation of decisions & trade-offs
- Knowledge sharing between teammates

Code vs Model Review
Similarities

There is code involved.

Code vs Model Review *Differences*

Lots of context beyond the Code Used to train a model

Code vs Model Review *Differences*

Lots of context beyond the Code Used to train a model

- model performance

Code vs Model Review *Differences*

Lots of context beyond the Code Used to train a model

- model performance
- data, including any transformations

Code vs Model Review *Differences*

Lots of context beyond the Code Used to train a model

- model performance
- data, including any transformations
- entire ML process, including all your failed experiments

Code vs Model Review *Differences*

Lots of context beyond the Code Used to train a model

- model performance
- data, including any transformations
- entire ML process, including all your failed experiments

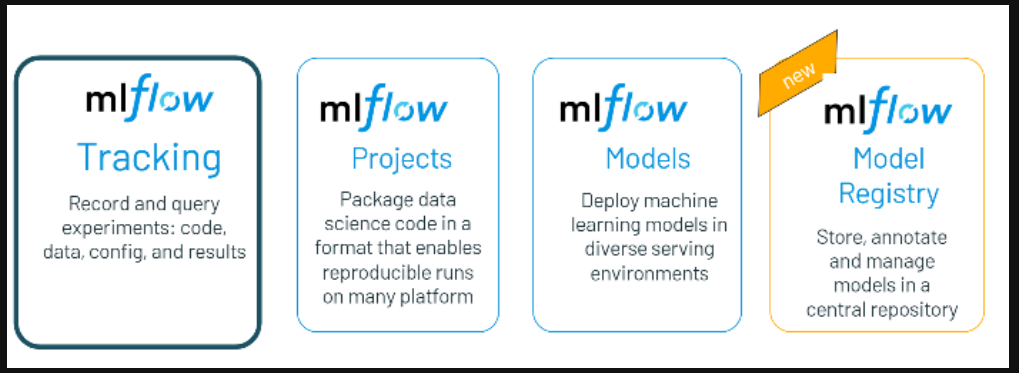
We can't really review a model *just* by reading the final code

MLflow

"Open source platform for managing the end-to-end machine learning lifecycle"

MLflow

"Open source platform for managing the end-to-end machine learning lifecycle"



Language & library agnostic:

- works with any ML library
- APIs for Python, R, Java, but everything is also accessible throuh a REST API & CLI

MLflow Tracking

Easily log almost anything you want to keep track of:

- parameters
- metrics
- arbitrary files ("artifacts" in mlflow)
    - plots, text files, Jupyter notebooks...
- code version
- training data

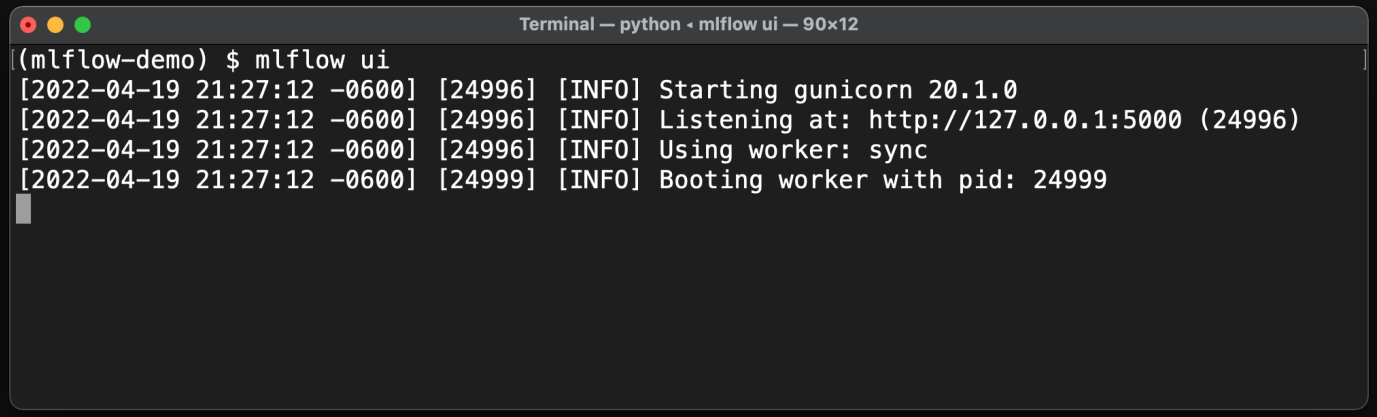
```
$ pip install mlflow
```

First Example

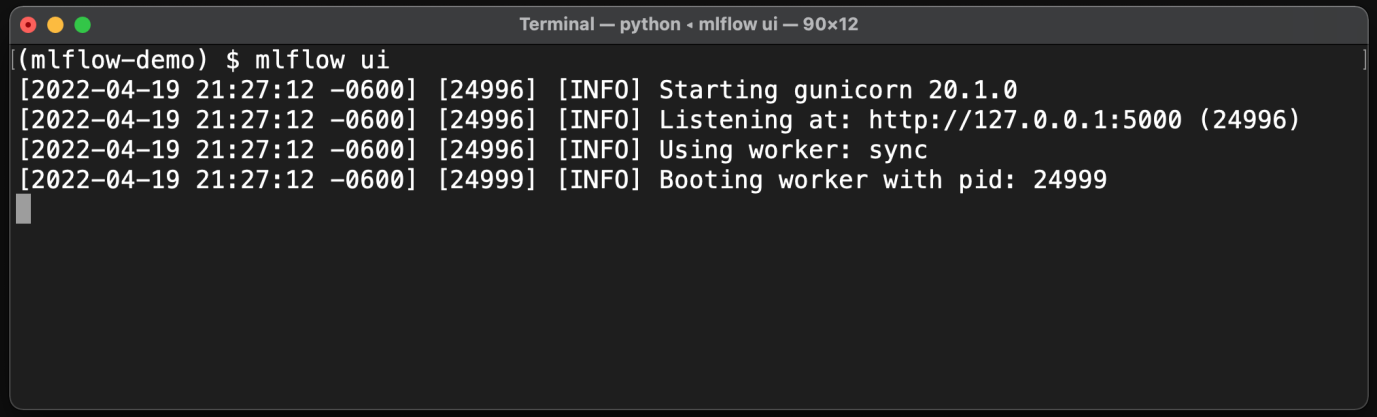
In [1]:

```
import mlflow

mlflow.log_param('my_parameter', 4)  # run starts automatically
mlflow.log_metric('score', 100)
mlflow.end_run()
```

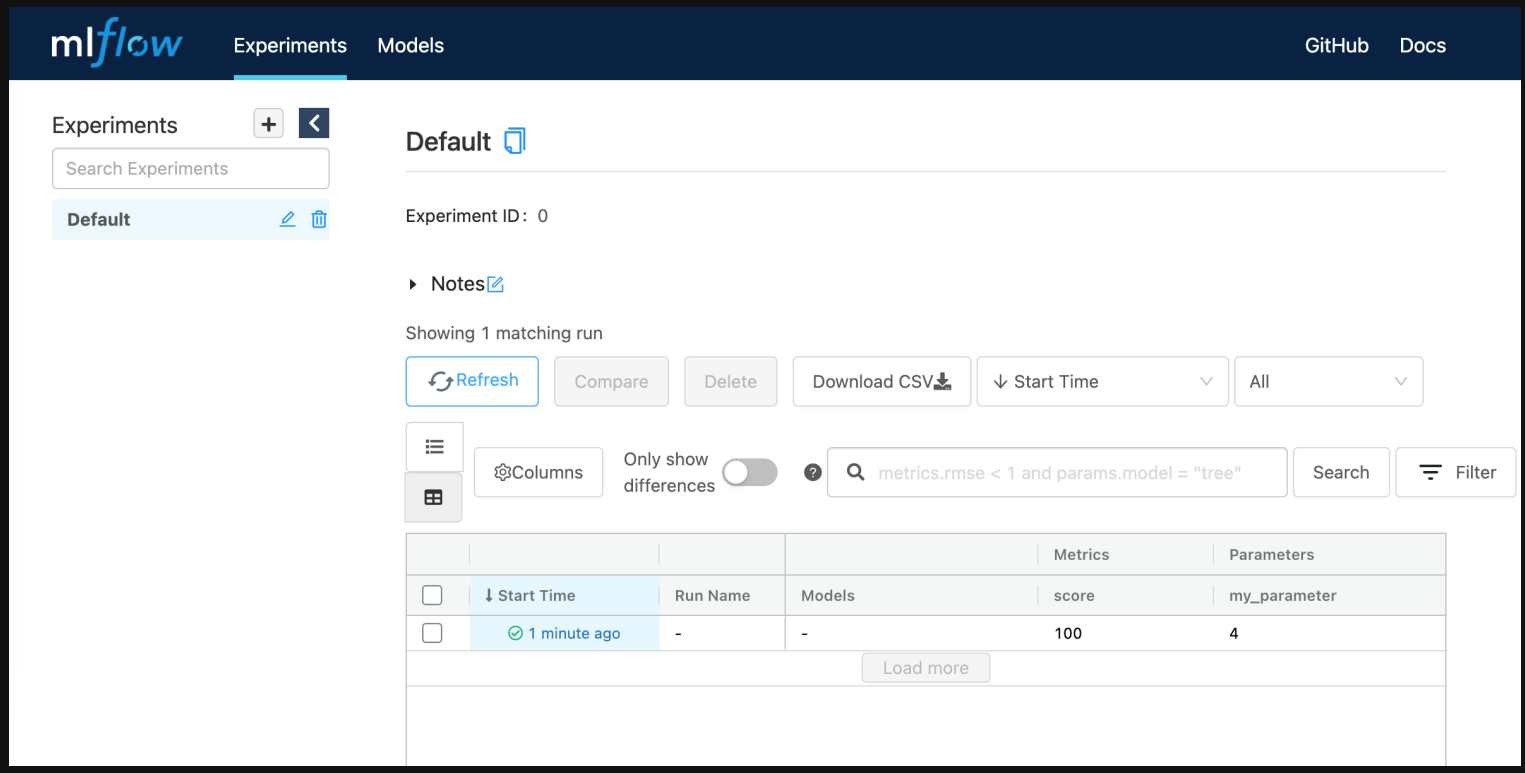
MLflow Tracking UI

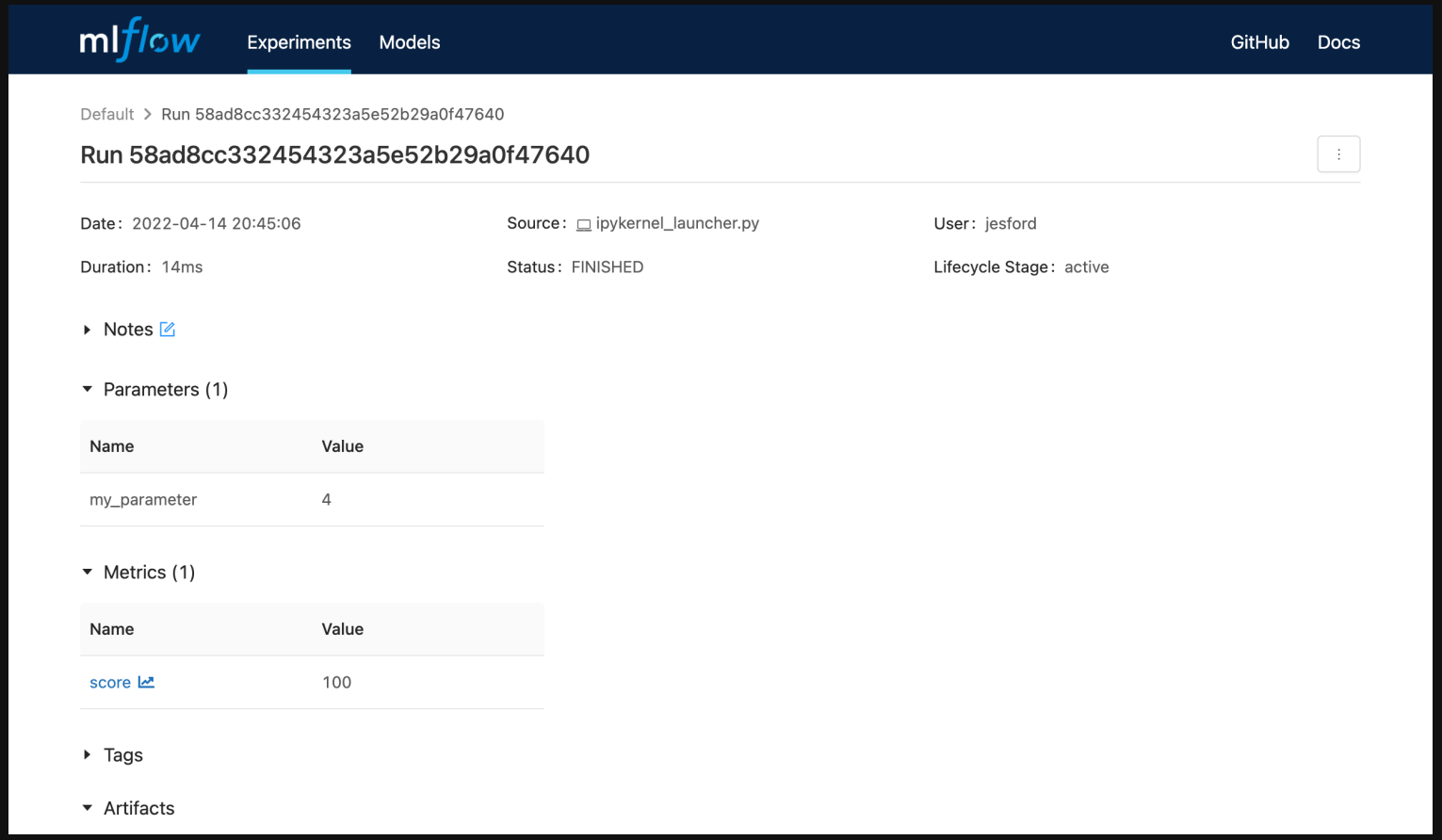


MLflow Tracking UI



Go to http://127.0.0.1:5000 in your browser...

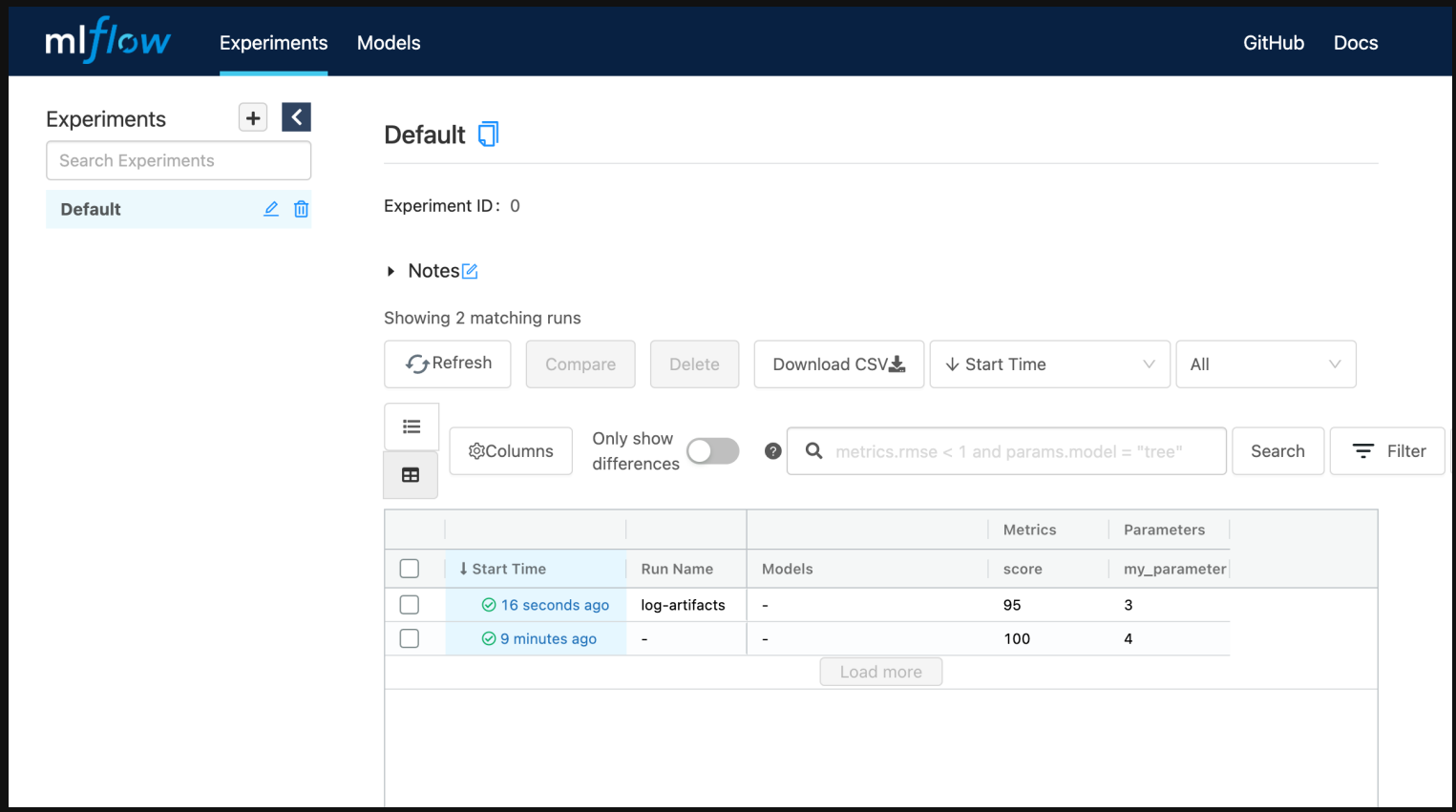

mlflow
Experiments
Models
GitHub
Docs
Experiments
Search Experiments
Default
Default
Experiment ID: 0
Notes
Showing 1 matching run
Refresh
Compare
Delete
Download CSV
Start Time
All
Columns
Only show differences
metrics.rmse < 1 and params.model = "tree"
Search
Filter
Metrics
Parameters
Start Time
Run Name
Models
score
my_parameter
1 minute ago
-
-
100
4
Load more

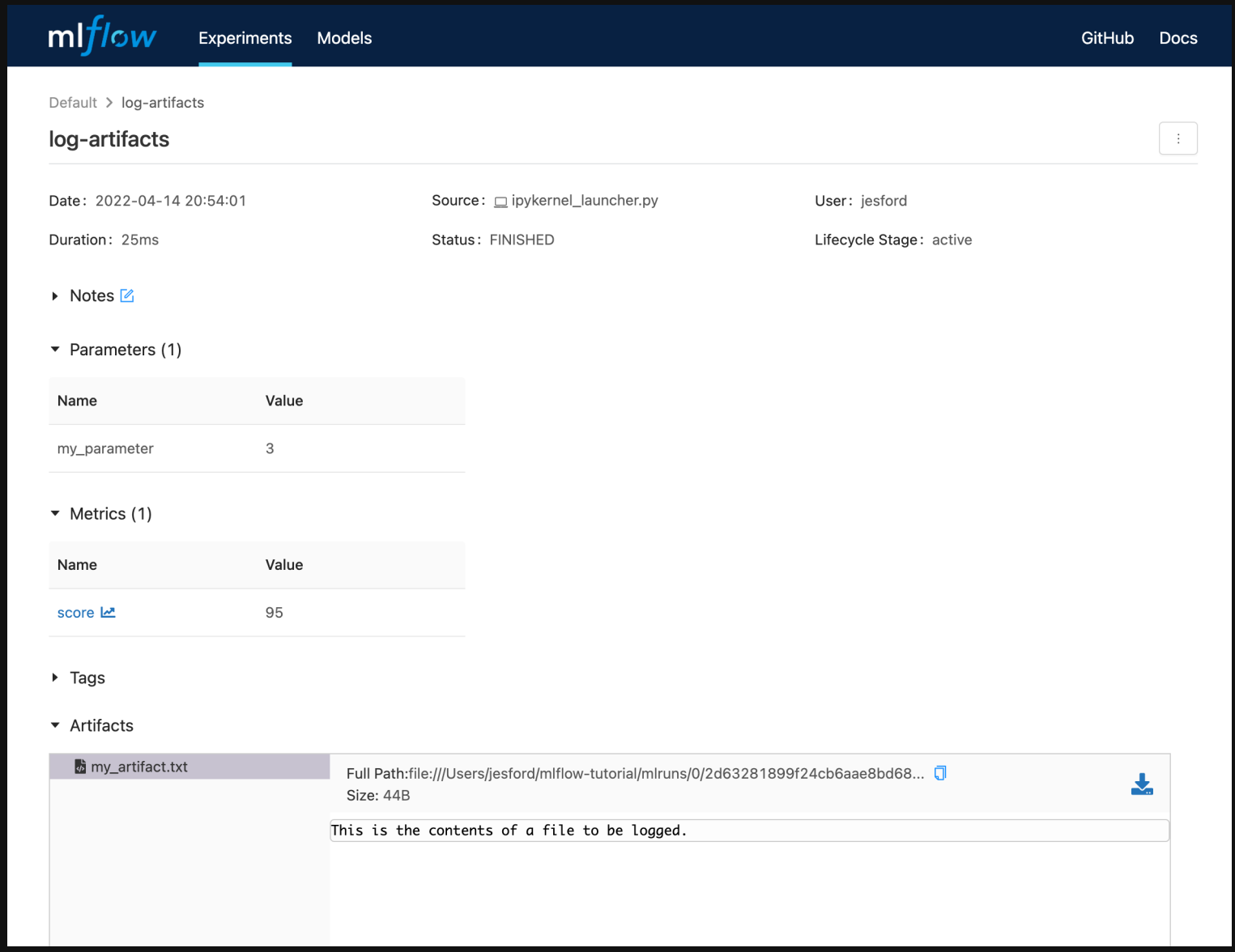

mlflow
Experiments
Models
GitHub
Docs
Default > Run 58ad8cc332454323a5e52b29a0f47640
Run 58ad8cc332454323a5e52b29a0f47640
Date: 2022-04-14 20:45:06
Source: ipykernel_launcher.py
User: jesford
Duration: 14ms
Status: FINISHED
Lifecycle Stage: active
Notes
Parameters (1)
Name
Value
my_parameter
4
Metrics (1)
Name
Value
score
100
Tags
Artifacts

Logging Artifacts

In [2]:

```
import mlflow

# explicitly start the run to give it a nice name
with mlflow.start_run(run_name='log-artifacts'):

    mlflow.log_param('my_parameter', 3)
    mlflow.log_metric('score', 95)

    with open('my_artifact.txt', 'w') as f:
        f.write('This is the contents of a file to be logged.')

    mlflow.log_artifact('my_artifact.txt')
```


mlflow
Experiments
Models
GitHub
Docs
Experiments
Search Experiments
Default
Default
Experiment ID: 0
Notes
Showing 2 matching runs
Refresh
Compare
Delete
Download CSV
Start Time
All
Columns
Only show differences
metrics.rmse < 1 and params.model = "tree"
Search
Filter
Metrics
Parameters
Start Time
Run Name
Models
score
my_parameter
16 seconds ago
log-artifacts
-
95
3
9 minutes ago
-
-
100
4
Load more

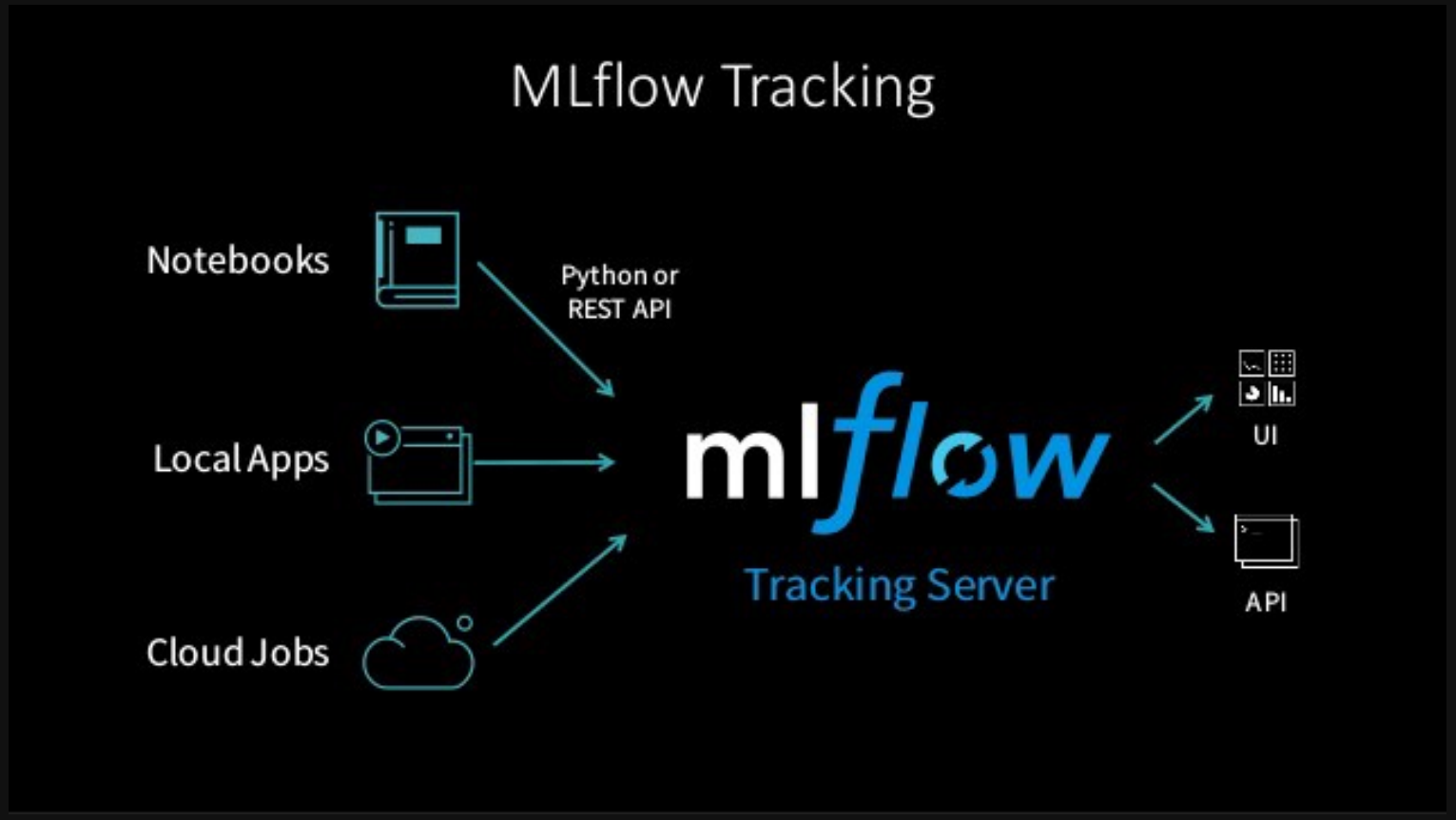

mlflow
Experiments
Models
GitHub
Docs
Default > log-artifacts
log-artifacts
Date: 2022-04-14 20:54:01
Source: ipykernel_launcher.py
User: jesford
Duration: 25ms
Status: FINISHED
Lifecycle Stage: active
Notes
Parameters (1)
Name
Value
my_parameter
3
Metrics (1)
Name
Value
score
95
Tags
Artifacts
my_artifact.txt
Full Path:file:///Users/jesford/mlflow-tutorial/mlruns/0/2d63281899f24cb6aae8bd68...
Size: 44B
This is the contents of a file to be logged.

Where is MLflow data logged?

By default to a `mlruns/` folder in your working directory. Lots of other options, including remote tracking servers for teams (see MLflow docs for possibilities).

We can *access* what we've logged using either the MLflow API or the **MLflow Tracking UI**.

image source: mlflow docs


MLflow Tracking
Notebooks
Local Apps
Cloud Jobs
Python or
REST API
mlflow
Tracking Server
UI
API

Not just for ML!

Notice that nothing we've done so far has been ML specific!

You could use MLFlow Tracking for kind of any analyses or projects where you find yourself manually recording values.

Let's do some ML

"20 Newsgroups" dataset: text documents containing discussions of 20 different topics.

We'll just use 2 of these topics and build a binary classifier to distinguish between text that is about baseball vs hockey.

Load & Prepare a Real World Dataset

In [4]:

```
from sklearn.datasets import fetch_20newsgroups

# load train and test data for 2 categories
categories_to_classify = ['rec.sport.baseball', 'rec.sport.hockey']
X_train_raw, y_train = fetch_20newsgroups(
    subset='train', categories=categories_to_classify, return_X_y=True
)
X_test_raw, y_test = fetch_20newsgroups(
    subset='test', categories=categories_to_classify, return_X_y=True
)
```

In [5]:
```
# Example document
print(X_train_raw[0])
```

```
From: dougb@comm.mot.com (Doug Bank)
Subject: Re: Info needed for Cleveland tickets
Reply-To: dougb@ecs.comm.mot.com
Organization: Motorola Land Mobile Products Sector
Distribution: usa
Nntp-Posting-Host: 145.1.146.35
Lines: 17

In article <1993Apr1.234031.4950@leland.Stanford.EDU>, bohnert@
leland.Stanford.EDU (matthew bohnert) writes:

|> I'm going to be in Cleveland Thursday, April 15 to Sunday, A
pril 18.
|> Does anybody know if the Tribe will be in town on those date
s, and
|> if so, who're they playing and if tickets are available?

The tribe will be in town from April 16 to the 19th.
There are ALWAYS tickets available! (Though they are playing To
ronto,
and many Toronto fans make the trip to Cleveland as it is easie
r to
get tickets in Cleveland than in Toronto.  Either way, I seriou
sly
doubt they will sell out until the end of the season.)

--
```

Doug Bank
dougb@ecs.comm.mot.com
dougb@nwu.edu
dougb@casbah.acns.nwu.edu

Private Systems Division
Motorola Communications Sector
Schaumburg, Illinois
708-576-8207

Transform data from raw text to numbers

In [6]:
```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(max_features=100, stop_words='english')
X_train = vectorizer.fit_transform(X_train_raw).todense()
X_test = vectorizer.transform(X_test_raw).todense()

print('Training set size after TF-IDF transform: {}'.format(X_train.sha
```

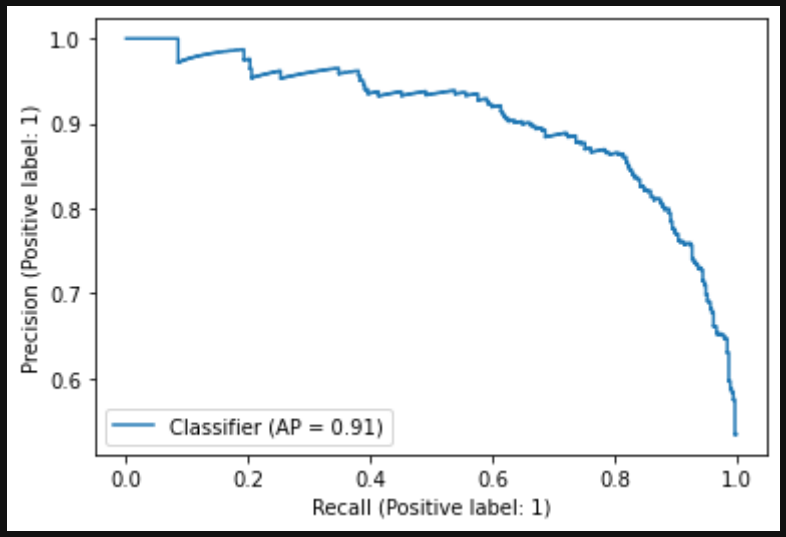
```
Training set size after TF-IDF transform: (1197, 100)
```

Tracking your ML Model

In [8]:

```
import mlflow
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, PrecisionRecallDisplay


with mlflow.start_run(run_name='hockey-vs-baseball'):

    # fit a model with certain hyperparameters
    penalty = 'l2'
    max_iter = 10
    clf = LogisticRegression(penalty=penalty, max_iter=max_iter)
    clf.fit(X_train, y_train)

    # get predictions
    y_train_pred = clf.predict(X_train)
    y_test_pred = clf.predict(X_test)
    y_test_predprob = clf.predict_proba(X_test)[:, 1]

    # make a plot
    fig, ax = plt.subplots(1, 1)
    PrecisionRecallDisplay.from_predictions(y_test, y_test_predprob, ax
    fig.savefig('PR.png')

    # track all the things
```

```
    mlflow.log_params({'penalty': penalty, 'max_iter': max_iter})
    mlflow.log_metric('acc', accuracy_score(y_train, y_train_pred))
    mlflow.log_metric('val_acc', accuracy_score(y_test, y_test_pred))
    mlflow.log_artifact('PR.png')
    mlflow.log_artifact('tutorial.ipynb')
    mlflow.sklearn.log_model(clf, 'model')
```



mlflow Experiments Models GitHub Docs

Default > hockey-vs-baseball

hockey-vs-baseball

Date: 2022-04-19 20:18:07 Source: ipykernel_launcher.py User: jesford

Duration: 4.1s Status: FINISHED Lifecycle Stage: active

▸ Notes

▾ Parameters (2)

Name	Value
max_iter	10
penalty	l2

▾ Metrics (2)

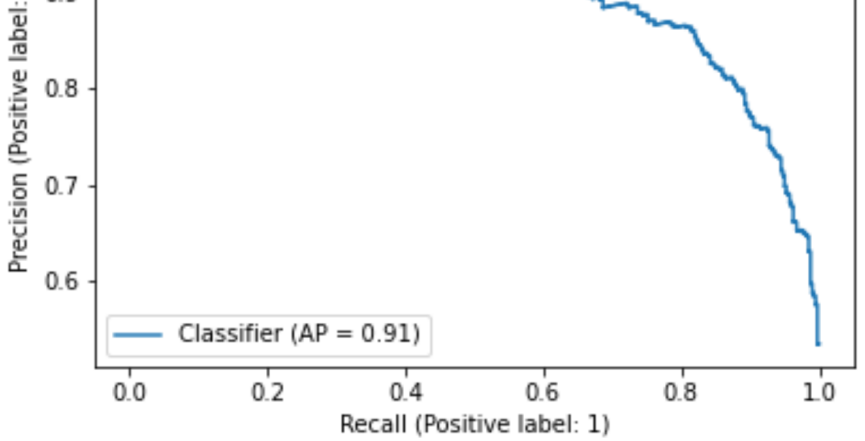
Name	Value
acc	0.887
val_acc	0.833

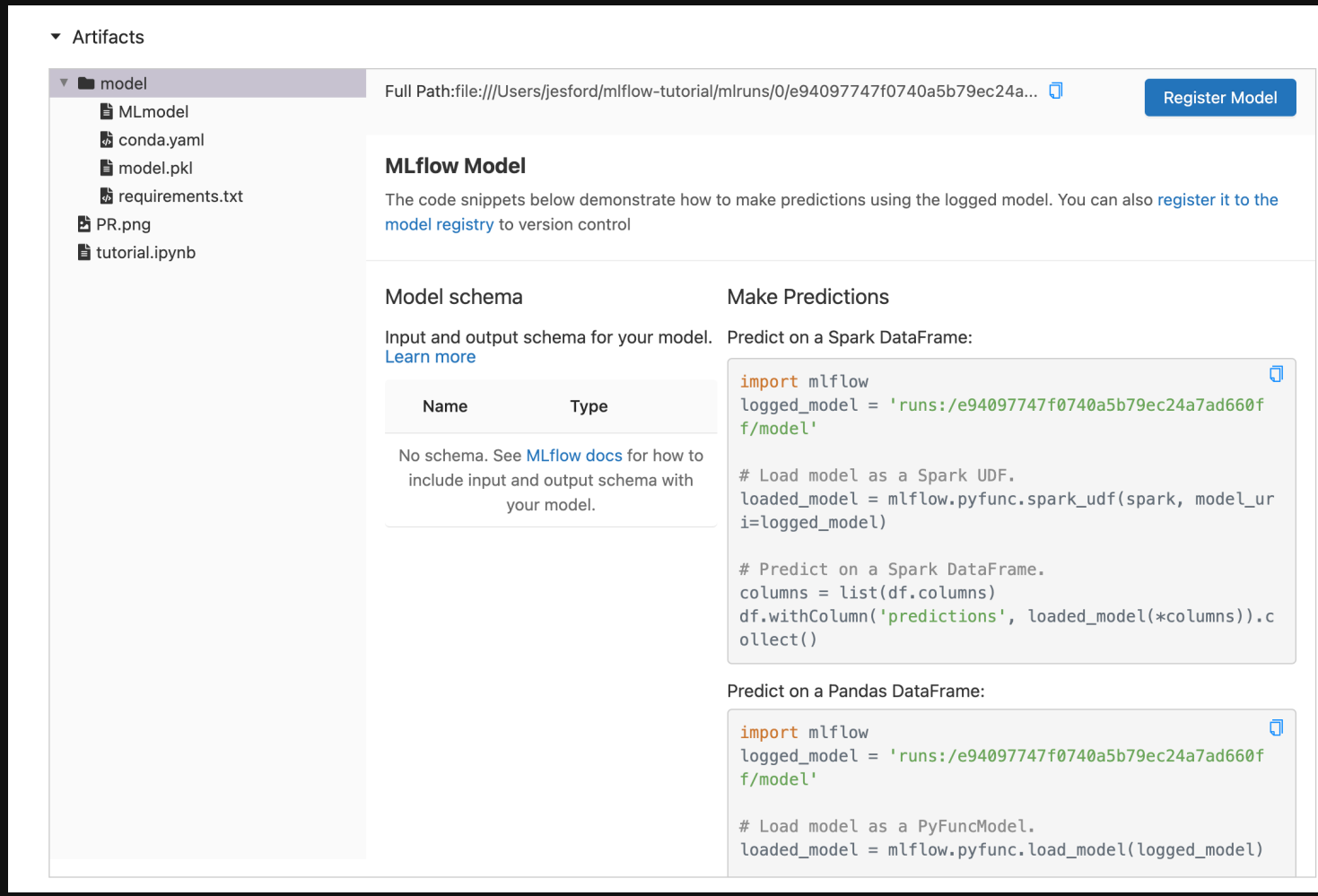
▸ Tags

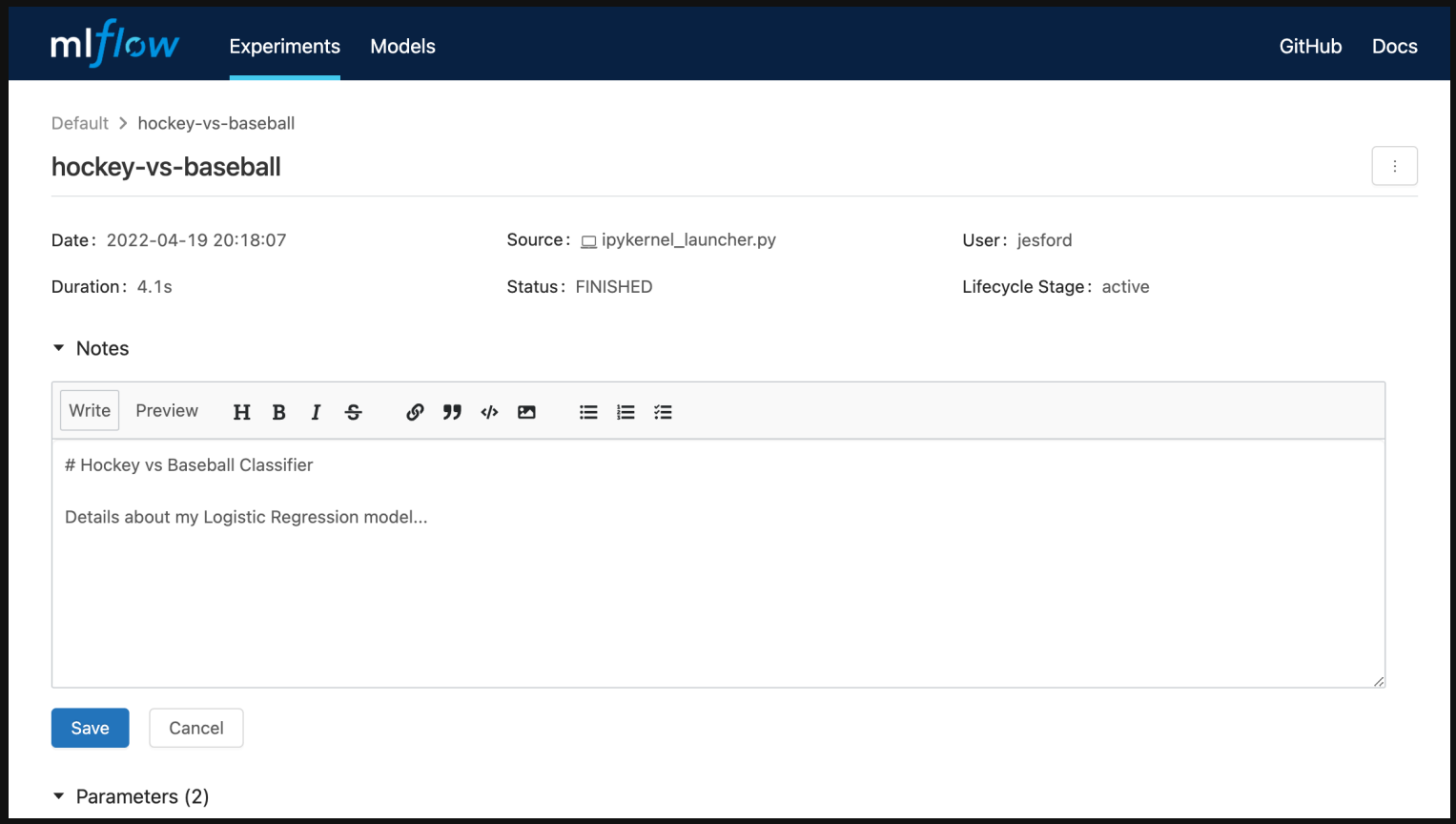
▾ Artifacts

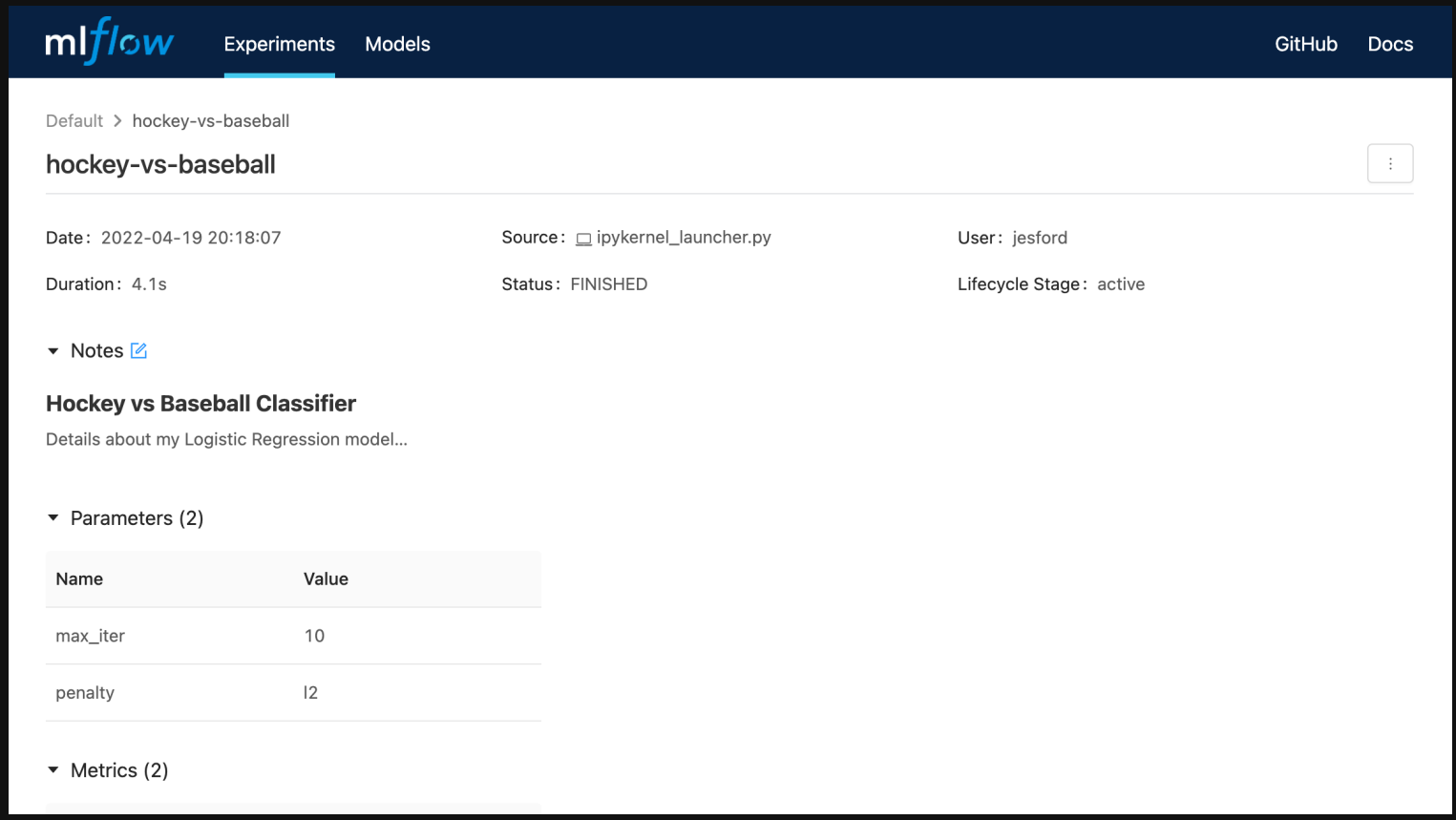
▸ model
PR.png
tutorial.ipynb

Full Path:file:///Users/jesford/mlflow-tutorial/mlruns/0/e94097747f0740a5b79ec24a...
Size: 13.02KB


Precision (Positive label:
0.8
0.7
0.6
Classifier (AP = 0.91)
0.0
0.2
0.4
0.6
0.8
1.0
Recall (Positive label: 1)


Artifacts
model
MLmodel
conda.yaml
model.pkl
requirements.txt
PR.png
tutorial.ipynb
Full Path:file:///Users/jesford/mlflow-tutorial/mlruns/0/e94097747f0740a5b79ec24a...
Register Model
MLflow Model
The code snippets below demonstrate how to make predictions using the logged model. You can also register it to the model registry to version control
Model schema
Input and output schema for your model. Learn more
Name
Type
No schema. See MLflow docs for how to include input and output schema with your model.
Make Predictions
Predict on a Spark DataFrame:
import mlflow
logged_model = 'runs:/e94097747f0740a5b79ec24a7ad660ff/model'
# Load model as a Spark UDF.
loaded_model = mlflow.pyfunc.spark_udf(spark, model_uri=logged_model)
# Predict on a Spark DataFrame.
columns = list(df.columns)
df.withColumn('predictions', loaded_model(*columns)).collect()
Predict on a Pandas DataFrame:
import mlflow
logged_model = 'runs:/e94097747f0740a5b79ec24a7ad660ff/model'
# Load model as a PyFuncModel.
loaded_model = mlflow.pyfunc.load_model(logged_model)


mlflow
Experiments
Models
GitHub
Docs
Default > hockey-vs-baseball
hockey-vs-baseball
Date: 2022-04-19 20:18:07
Source: ipykernel_launcher.py
User: jesford
Duration: 4.1s
Status: FINISHED
Lifecycle Stage: active
Notes
Write
Preview
Hockey vs Baseball Classifier
Details about my Logistic Regression model...
Save
Cancel
Parameters (2)


mlflow
Experiments
Models
GitHub
Docs
Default > hockey-vs-baseball
hockey-vs-baseball
Date: 2022-04-19 20:18:07
Source: ipykernel_launcher.py
User: jesford
Duration: 4.1s
Status: FINISHED
Lifecycle Stage: active
Notes
Hockey vs Baseball Classifier
Details about my Logistic Regression model...
Parameters (2)
Name
Value
max_iter
10
penalty
l2
Metrics (2)

Autologging Model Training

Autologging Model Training

A single line of code will automatically tracks lots of useful things about your model, metrics, plots.

Supported for `scikit-learn`, `tensorflow` & `keras`, `pytorch`, `xgboost`, and more.

Autologging Example 1: `sklearn`

Autologging Example 1: `sklearn`

In [9]:
```
from sklearn.ensemble import RandomForestClassifier

mlflow.sklearn.autolog()  # just one line added!

rfc = RandomForestClassifier(n_estimators=10)
rfc.fit(X_train, y_train)
```

```
2022/04/19 20:37:41 WARNING mlflow.utils.autologging_utils: You
are using an unsupported version of sklearn. If you encounter e
rrors during autologging, try upgrading / downgrading sklearn t
o a supported version, or try upgrading MLflow.
2022/04/19 20:37:41 INFO mlflow.utils.autologging_utils: Create
d MLflow autologging run with ID 'd2049ae6b4074af49867e89e58d99
73c', which will track hyperparameters, performance metrics, mo
del artifacts, and lineage information for the current sklearn
workflow
```

Out[9]:
```
RandomForestClassifier(n_estimators=10)
```

Back in the MLflow UI...

Run d2049ae6b4074af49867e89e58d9973c

Date: 2022-04-19 20:37:41 Source: ipykernel_launcher.py User: jesford

Duration: 3.5s Status: FINISHED Lifecycle Stage: active

▸ Notes

▾ Parameters (18)

Name	Value
bootstrap	True
ccp_alpha	0.0
class_weight	None
criterion	gini
max_depth	None
max_features	auto
max_leaf_nodes	None
max_samples	None
min_impurity_decrease	0.0
min_samples_leaf	1
min_samples_split	2
min_weight_fraction_leaf	0.0
n_estimators	10
n_jobs	None

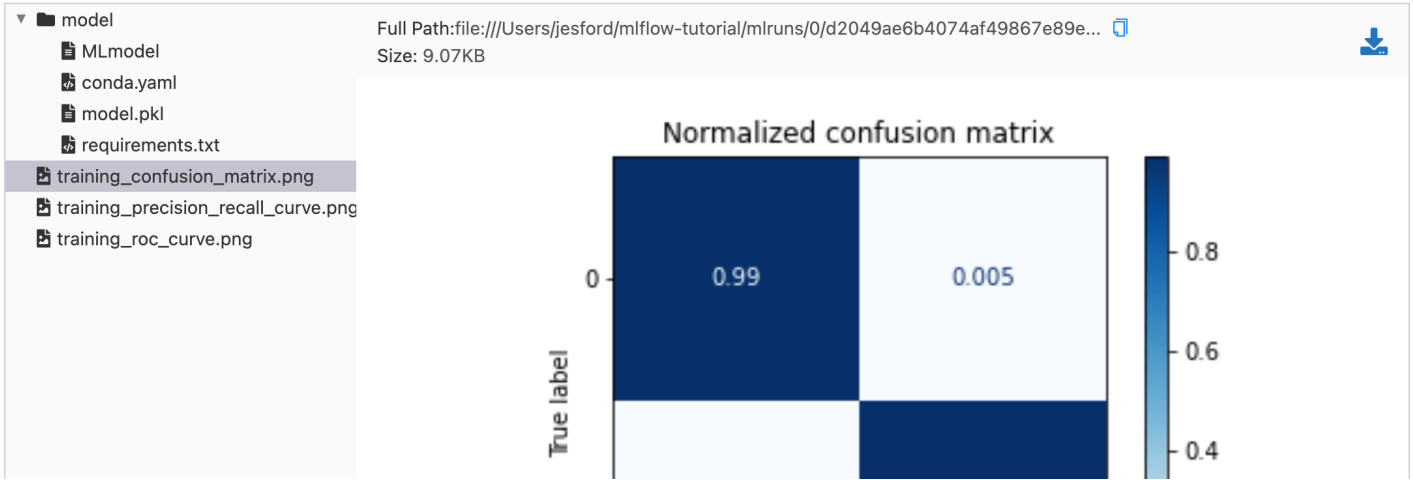
oob_score	False
random_state	None
verbose	0
warm_start	False

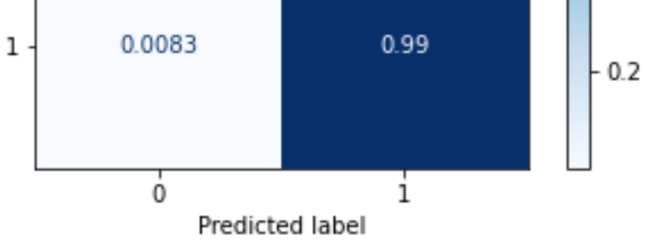
▼ Metrics (7)

Name	Value
training_accuracy_score	0.993
training_f1_score	0.993
training_log_loss	0.102
training_precision_score	0.993
training_recall_score	0.993
training_roc_auc_score	1
training_score	0.993

▶ Tags (2)

▼ Artifacts




0.0083
0.99
1
0
1
0.2
Predicted label

Autologging Example 2:

`tensorflow.keras`

Autologging Example 2: `tensorflow.keras`

In [16]:

```
import tensorflow as tf
from tensorflow.keras import Input, Model, layers

# parameters to vary
hidden_layer_size = 32
learning_rate = 0.01

with mlflow.start_run(run_name=f'sports-NN-{hidden_layer_size}d-lr{lear

    mlflow.tensorflow.autolog()  # just one line added!

    # a simple NN with one hidden layer
    inputs = Input(shape=(X_train.shape[1],))
    x = layers.Dense(hidden_layer_size, activation='relu')(inputs)
    outputs = layers.Dense(1, activation='sigmoid')(x)
    model = Model(inputs, outputs)

    loss = 'binary_crossentropy'
    optimizer = tf.optimizers.SGD(learning_rate=learning_rate)
    metrics = [
        tf.metrics.BinaryAccuracy(),
        tf.metrics.AUC(curve='ROC', name='AUROC'),
```

```
    ]
    model.compile(optimizer=optimizer, loss=loss, metrics=metrics)

    model.fit(X_train, y_train, epochs=40, shuffle=True, validation_dat
```

```
Train on 1197 samples, validate on 796 samples
Epoch 1/40
1152/1197 [===========================>..] - ETA: 0s - loss: 0.
6944 - binary_accuracy: 0.4826 - AUROC: 0.4870

2022-04-19 20:51:03.144988: I tensorflow/core/profiler/lib/prof
iler_session.cc:184] Profiler session started.

1197/1197 [==============================] - 2s 2ms/sample - lo
ss: 0.6945 - binary_accuracy: 0.4795 - AUROC: 0.4853 - val_los
s: 0.6879 - val_binary_accuracy: 0.5540 - val_AUROC: 0.5841
Epoch 2/40
1197/1197 [==============================] - 0s 252us/sample -
loss: 0.6933 - binary_accuracy: 0.4937 - AUROC: 0.5005 - val_lo
ss: 0.6870 - val_binary_accuracy: 0.5666 - val_AUROC: 0.5963
Epoch 3/40
1197/1197 [==============================] - 0s 250us/sample -
loss: 0.6920 - binary_accuracy: 0.5104 - AUROC: 0.5184 - val_lo
ss: 0.6861 - val_binary_accuracy: 0.5854 - val_AUROC: 0.6069
Epoch 4/40
1197/1197 [==============================] - 0s 267us/sample -
loss: 0.6909 - binary_accuracy: 0.5246 - AUROC: 0.5357 - val_lo
ss: 0.6852 - val_binary_accuracy: 0.5930 - val_AUROC: 0.6176
Epoch 5/40
1197/1197 [==============================] - 0s 246us/sample -
loss: 0.6896 - binary_accuracy: 0.5455 - AUROC: 0.5524 - val_lo
ss: 0.6843 - val_binary_accuracy: 0.6055 - val_AUROC: 0.6310
```

```
Epoch 6/40
1197/1197 [==============================] - 0s 255us/sample -
loss: 0.6885 - binary_accuracy: 0.5522 - AUROC: 0.5698 - val_lo
ss: 0.6834 - val_binary_accuracy: 0.6131 - val_AUROC: 0.6431
Epoch 7/40
1197/1197 [==============================] - 0s 274us/sample -
loss: 0.6873 - binary_accuracy: 0.5647 - AUROC: 0.5869 - val_lo
ss: 0.6825 - val_binary_accuracy: 0.6219 - val_AUROC: 0.6518
Epoch 8/40
1197/1197 [==============================] - 0s 301us/sample -
loss: 0.6861 - binary_accuracy: 0.5773 - AUROC: 0.6031 - val_lo
ss: 0.6817 - val_binary_accuracy: 0.6231 - val_AUROC: 0.6625
Epoch 9/40
1197/1197 [==============================] - 0s 291us/sample -
loss: 0.6849 - binary_accuracy: 0.5865 - AUROC: 0.6189 - val_lo
ss: 0.6808 - val_binary_accuracy: 0.6294 - val_AUROC: 0.6699
Epoch 10/40
1197/1197 [==============================] - 0s 284us/sample -
loss: 0.6837 - binary_accuracy: 0.5990 - AUROC: 0.6317 - val_lo
ss: 0.6799 - val_binary_accuracy: 0.6394 - val_AUROC: 0.6832
Epoch 11/40
1197/1197 [==============================] - 0s 347us/sample -
loss: 0.6825 - binary_accuracy: 0.6065 - AUROC: 0.6501 - val_lo
ss: 0.6789 - val_binary_accuracy: 0.6420 - val_AUROC: 0.6894
Epoch 12/40
1197/1197 [==============================] - 0s 352us/sample -
loss: 0.6812 - binary_accuracy: 0.6157 - AUROC: 0.6617 - val_lo
ss: 0.6780 - val_binary_accuracy: 0.6545 - val_AUROC: 0.6957
Epoch 13/40
1197/1197 [==============================] - 0s 253us/sample -
loss: 0.6800 - binary_accuracy: 0.6224 - AUROC: 0.6728 - val_lo
```

```
ss: 0.6770 - val_binary_accuracy: 0.6595 - val_AUROC: 0.7031
Epoch 14/40
1197/1197 [==============================] - 0s 253us/sample -
loss: 0.6787 - binary_accuracy: 0.6391 - AUROC: 0.6875 - val_lo
ss: 0.6761 - val_binary_accuracy: 0.6608 - val_AUROC: 0.7121
Epoch 15/40
1197/1197 [==============================] - 0s 328us/sample -
loss: 0.6774 - binary_accuracy: 0.6466 - AUROC: 0.7000 - val_lo
ss: 0.6750 - val_binary_accuracy: 0.6671 - val_AUROC: 0.7196
Epoch 16/40
1197/1197 [==============================] - 0s 334us/sample -
loss: 0.6761 - binary_accuracy: 0.6591 - AUROC: 0.7102 - val_lo
ss: 0.6740 - val_binary_accuracy: 0.6683 - val_AUROC: 0.7267
Epoch 17/40
1197/1197 [==============================] - 0s 353us/sample -
loss: 0.6748 - binary_accuracy: 0.6658 - AUROC: 0.7199 - val_lo
ss: 0.6729 - val_binary_accuracy: 0.6796 - val_AUROC: 0.7318
Epoch 18/40
1197/1197 [==============================] - 0s 310us/sample -
loss: 0.6734 - binary_accuracy: 0.6717 - AUROC: 0.7289 - val_lo
ss: 0.6719 - val_binary_accuracy: 0.6784 - val_AUROC: 0.7373
Epoch 19/40
1197/1197 [==============================] - 0s 369us/sample -
loss: 0.6720 - binary_accuracy: 0.6775 - AUROC: 0.7399 - val_lo
ss: 0.6707 - val_binary_accuracy: 0.6872 - val_AUROC: 0.7452
Epoch 20/40
1197/1197 [==============================] - 0s 374us/sample -
loss: 0.6706 - binary_accuracy: 0.6809 - AUROC: 0.7488 - val_lo
ss: 0.6696 - val_binary_accuracy: 0.6935 - val_AUROC: 0.7507
Epoch 21/40
1197/1197 [==============================] - 0s 373us/sample -
```

```
loss: 0.6691 - binary_accuracy: 0.6884 - AUROC: 0.7558 - val_lo
ss: 0.6683 - val_binary_accuracy: 0.6960 - val_AUROC: 0.7550
Epoch 22/40
1197/1197 [==============================] - 0s 350us/sample -
loss: 0.6676 - binary_accuracy: 0.6884 - AUROC: 0.7639 - val_lo
ss: 0.6671 - val_binary_accuracy: 0.6985 - val_AUROC: 0.7609
Epoch 23/40
1197/1197 [==============================] - 0s 278us/sample -
loss: 0.6661 - binary_accuracy: 0.6984 - AUROC: 0.7708 - val_lo
ss: 0.6658 - val_binary_accuracy: 0.7010 - val_AUROC: 0.7651
Epoch 24/40
1197/1197 [==============================] - 0s 349us/sample -
loss: 0.6644 - binary_accuracy: 0.6976 - AUROC: 0.7792 - val_lo
ss: 0.6645 - val_binary_accuracy: 0.7010 - val_AUROC: 0.7688
Epoch 25/40
1197/1197 [==============================] - 0s 300us/sample -
loss: 0.6628 - binary_accuracy: 0.7068 - AUROC: 0.7846 - val_lo
ss: 0.6631 - val_binary_accuracy: 0.7023 - val_AUROC: 0.7745
Epoch 26/40
1197/1197 [==============================] - 0s 363us/sample -
loss: 0.6611 - binary_accuracy: 0.7101 - AUROC: 0.7914 - val_lo
ss: 0.6617 - val_binary_accuracy: 0.7048 - val_AUROC: 0.7801
Epoch 27/40
1197/1197 [==============================] - 0s 283us/sample -
loss: 0.6593 - binary_accuracy: 0.7143 - AUROC: 0.7967 - val_lo
ss: 0.6602 - val_binary_accuracy: 0.7085 - val_AUROC: 0.7838
Epoch 28/40
1197/1197 [==============================] - 0s 254us/sample -
loss: 0.6575 - binary_accuracy: 0.7126 - AUROC: 0.8026 - val_lo
ss: 0.6587 - val_binary_accuracy: 0.7111 - val_AUROC: 0.7877
Epoch 29/40
```

```
1197/1197 [==============================] - 0s 274us/sample -
loss: 0.6557 - binary_accuracy: 0.7168 - AUROC: 0.8087 - val_lo
ss: 0.6572 - val_binary_accuracy: 0.7148 - val_AUROC: 0.7904
Epoch 30/40
1197/1197 [==============================] - 0s 320us/sample -
loss: 0.6538 - binary_accuracy: 0.7243 - AUROC: 0.8131 - val_lo
ss: 0.6556 - val_binary_accuracy: 0.7186 - val_AUROC: 0.7942
Epoch 31/40
1197/1197 [==============================] - 0s 346us/sample -
loss: 0.6518 - binary_accuracy: 0.7302 - AUROC: 0.8165 - val_lo
ss: 0.6540 - val_binary_accuracy: 0.7211 - val_AUROC: 0.7968
Epoch 32/40
1197/1197 [==============================] - 0s 363us/sample -
loss: 0.6498 - binary_accuracy: 0.7377 - AUROC: 0.8225 - val_lo
ss: 0.6523 - val_binary_accuracy: 0.7186 - val_AUROC: 0.8005
Epoch 33/40
1197/1197 [==============================] - 0s 360us/sample -
loss: 0.6477 - binary_accuracy: 0.7385 - AUROC: 0.8250 - val_lo
ss: 0.6505 - val_binary_accuracy: 0.7211 - val_AUROC: 0.8035
Epoch 34/40
1197/1197 [==============================] - 0s 307us/sample -
loss: 0.6456 - binary_accuracy: 0.7393 - AUROC: 0.8297 - val_lo
ss: 0.6487 - val_binary_accuracy: 0.7224 - val_AUROC: 0.8053
Epoch 35/40
1197/1197 [==============================] - 0s 273us/sample -
loss: 0.6433 - binary_accuracy: 0.7469 - AUROC: 0.8331 - val_lo
ss: 0.6469 - val_binary_accuracy: 0.7286 - val_AUROC: 0.8102
Epoch 36/40
1197/1197 [==============================] - 0s 242us/sample -
loss: 0.6410 - binary_accuracy: 0.7510 - AUROC: 0.8368 - val_lo
ss: 0.6450 - val_binary_accuracy: 0.7312 - val_AUROC: 0.8116
```

```
Epoch 37/40
1197/1197 [==============================] - 0s 331us/sample -
loss: 0.6387 - binary_accuracy: 0.7536 - AUROC: 0.8402 - val_lo
ss: 0.6430 - val_binary_accuracy: 0.7299 - val_AUROC: 0.8143
Epoch 38/40
1197/1197 [==============================] - 0s 269us/sample -
loss: 0.6363 - binary_accuracy: 0.7561 - AUROC: 0.8441 - val_lo
ss: 0.6411 - val_binary_accuracy: 0.7337 - val_AUROC: 0.8170
Epoch 39/40
1197/1197 [==============================] - 0s 348us/sample -
loss: 0.6337 - binary_accuracy: 0.7619 - AUROC: 0.8471 - val_lo
ss: 0.6390 - val_binary_accuracy: 0.7349 - val_AUROC: 0.8194
Epoch 40/40
1197/1197 [==============================] - 0s 360us/sample -
loss: 0.6312 - binary_accuracy: 0.7619 - AUROC: 0.8500 - val_lo
ss: 0.6369 - val_binary_accuracy: 0.7374 - val_AUROC: 0.8215
INFO:tensorflow:Assets written to: /var/folders/xv/48y52h7j27g9
jp0vs5n4s8rc0000gn/T/tmp7jcsckvn/model/data/model/assets
```

... back in MLflow UI

Again, all the parameters & metrics relevant to the run are recorded automatically

mlflow Experiments Models GitHub Docs

Default > sports-NN-32d-lr0.1

sports-NN-32d-lr0.1

Date: 2022-04-19 20:44:18 Source: ipykernel_launcher.py User: jesford

Duration: 22.2s Status: FINISHED Lifecycle Stage: active

▸ Notes

▾ Parameters (19)

Name	Value
batch_size	None
class_weight	None
epochs	40
initial_epoch	0
kwargs	<class 'inspect._empty'>
max_queue_size	10
opt_decay	0.0
opt_learning_rate	0.1
opt_momentum	0.0
opt_name	SGD
opt_nesterov	False
sample_weight	None
shuffle	True
steps_per_epoch	None

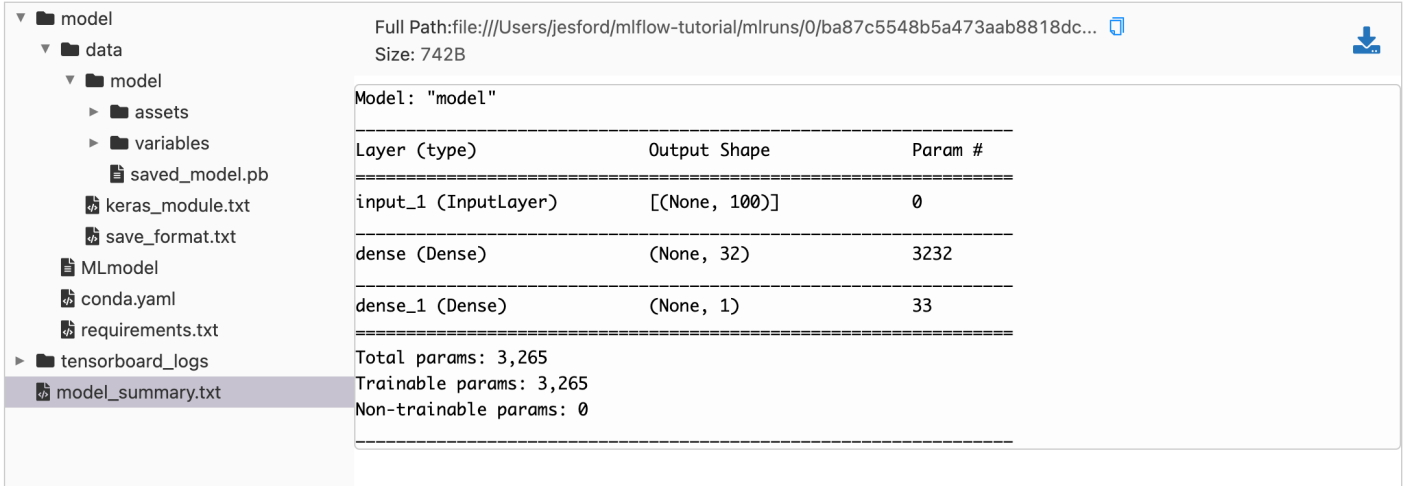
use_multiprocessing	False
validation_freq	1
validation_split	0.0
validation_steps	None
workers	1

▼ Metrics (6)

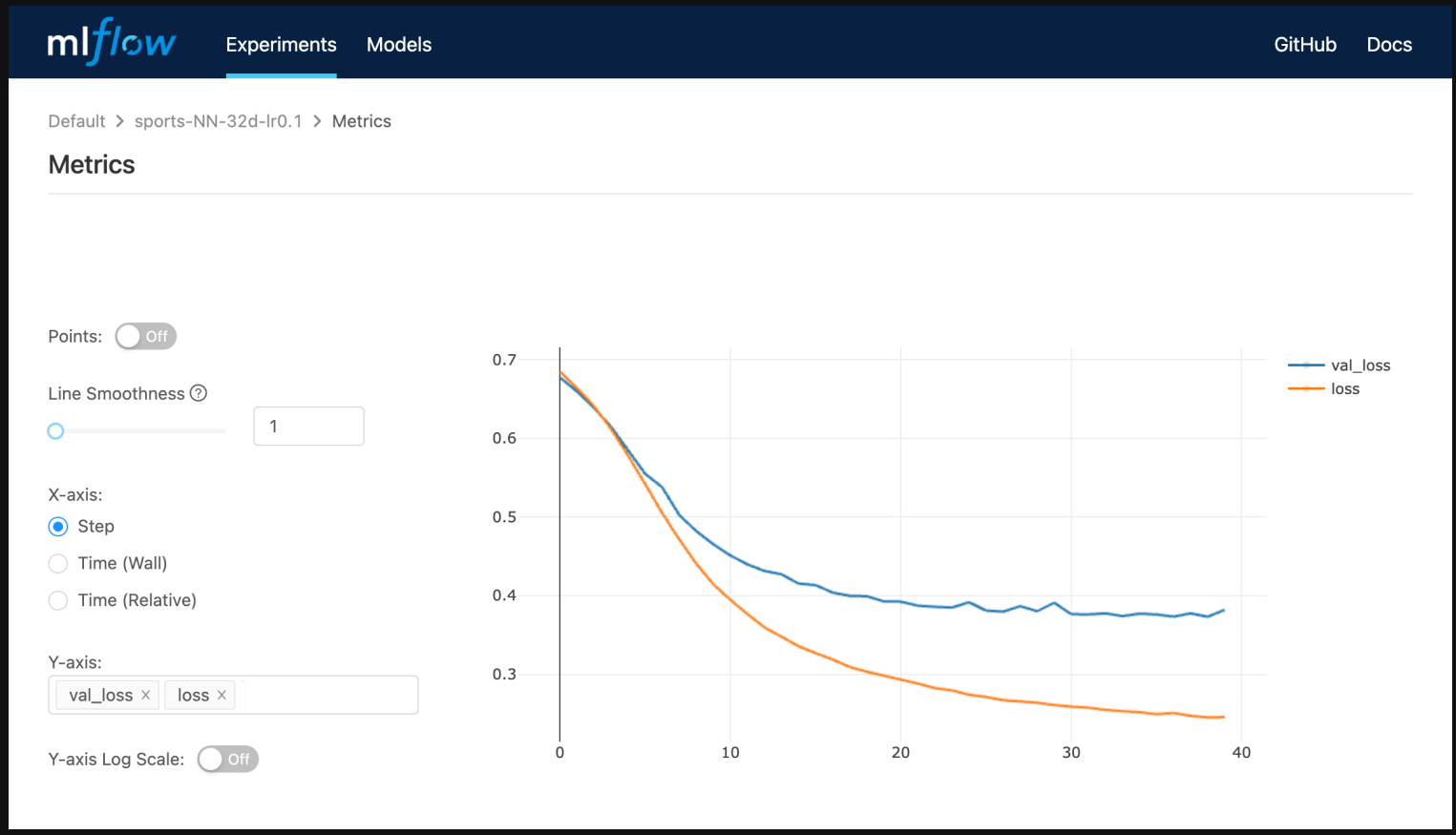
Name	Value
AUROC	0.965
binary_accuracy	0.894
loss	0.245
val_AUROC	0.917
val_binary_accuracy	0.838
val_loss	0.382

▶ Tags

▼ Artifacts

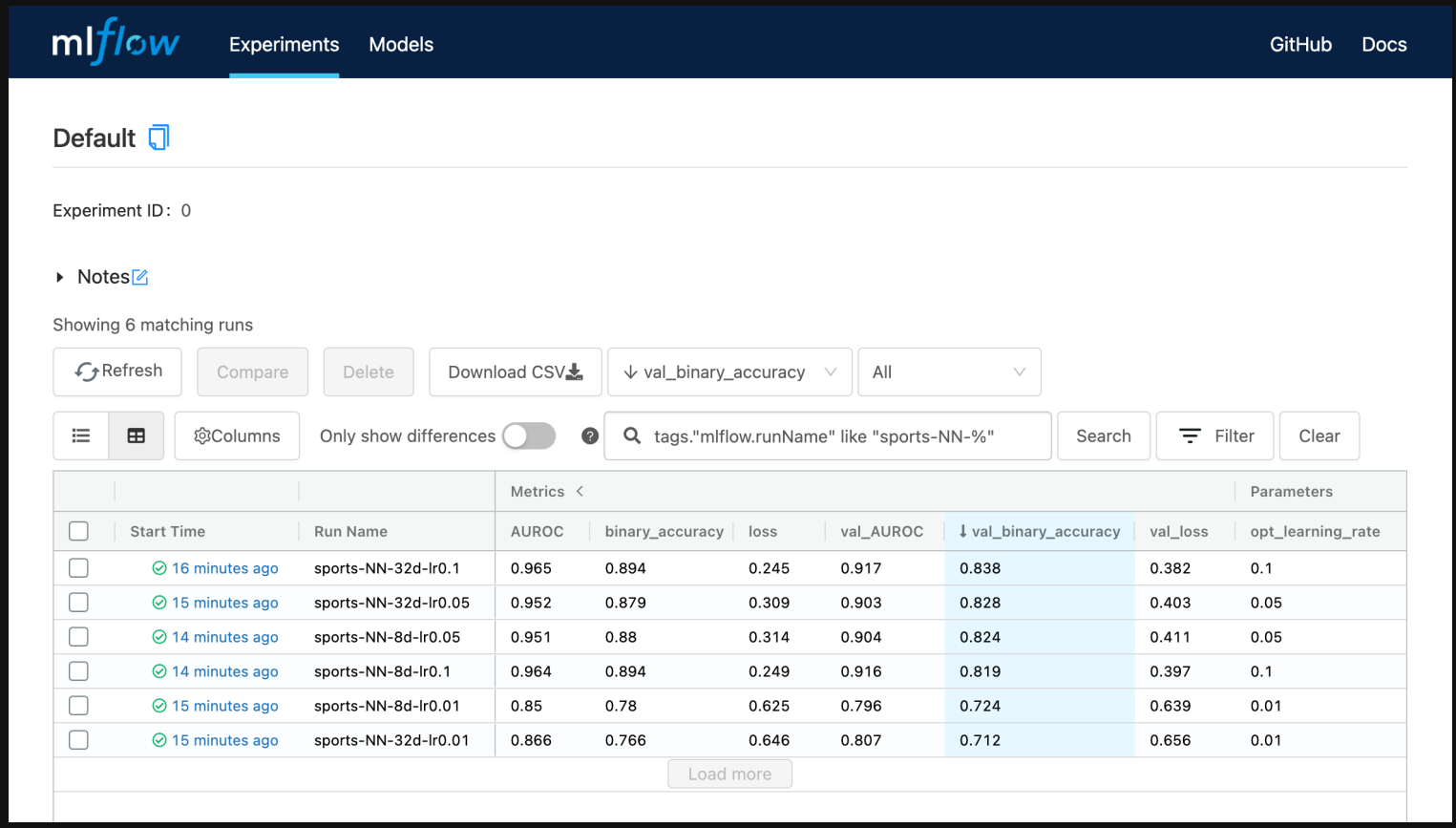


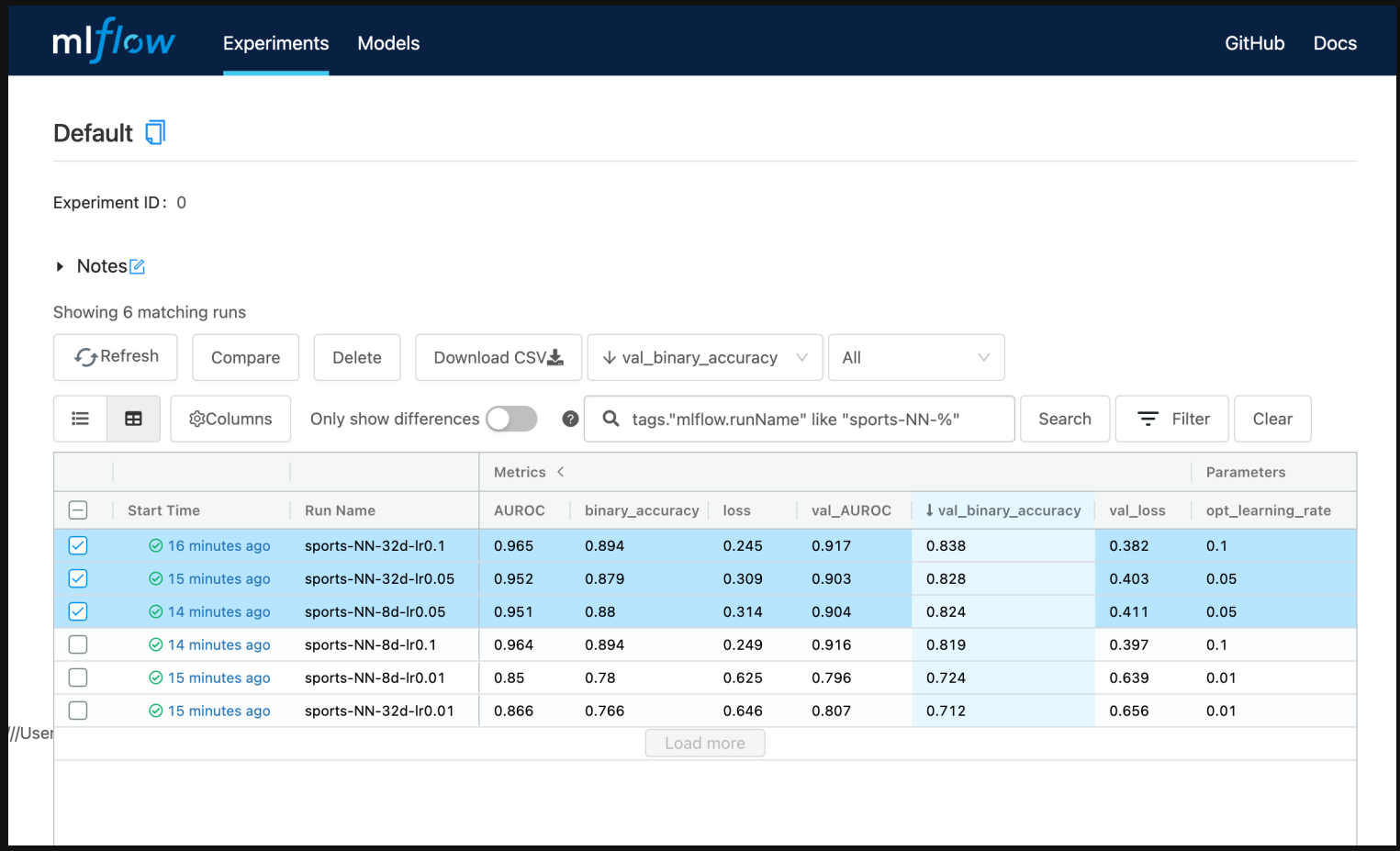
Clicking into one of the metrics, we can view training curves


mlflow
Experiments
Models
GitHub
Docs
Default > sports-NN-32d-lr0.1 > Metrics
Metrics
Points: Off
Line Smoothness
1
X-axis:
Step
Time (Wall)
Time (Relative)
Y-axis:
val_loss x
loss x
Y-axis Log Scale: Off
0.7
0.6
0.5
0.4
0.3
0
10
20
30
40
val_loss
loss

Powerful features for comparing models

- Previously we've looked at just one mlflow run at a time
- MLflow UI lets you sort and compare models directly


mlflow
Experiments
Models
GitHub
Docs
Default
Experiment ID: 0
Notes
Showing 6 matching runs
Refresh
Compare
Delete
Download CSV
val_binary_accuracy
All
Columns
Only show differences
tags."mlflow.runName" like "sports-NN-%"
Search
Filter
Clear
Metrics
Parameters
Start Time
Run Name
AUROC
binary_accuracy
loss
val_AUROC
val_binary_accuracy
val_loss
opt_learning_rate
16 minutes ago
sports-NN-32d-lr0.1
0.965
0.894
0.245
0.917
0.838
0.382
0.1
15 minutes ago
sports-NN-32d-lr0.05
0.952
0.879
0.309
0.903
0.828
0.403
0.05
14 minutes ago
sports-NN-8d-lr0.05
0.951
0.88
0.314
0.904
0.824
0.411
0.05
14 minutes ago
sports-NN-8d-lr0.1
0.964
0.894
0.249
0.916
0.819
0.397
0.1
15 minutes ago
sports-NN-8d-lr0.01
0.85
0.78
0.625
0.796
0.724
0.639
0.01
15 minutes ago
sports-NN-32d-lr0.01
0.866
0.766
0.646
0.807
0.712
0.656
0.01
Load more


mlflow
Experiments
Models
GitHub
Docs
Default
Experiment ID: 0
Notes
Showing 6 matching runs
Refresh
Compare
Delete
Download CSV
val_binary_accuracy
All
Columns
Only show differences
tags."mlflow.runName" like "sports-NN-%"
Search
Filter
Clear
Metrics
Parameters
Start Time
Run Name
AUROC
binary_accuracy
loss
val_AUROC
val_binary_accuracy
val_loss
opt_learning_rate
16 minutes ago
sports-NN-32d-lr0.1
0.965
0.894
0.245
0.917
0.838
0.382
0.1
15 minutes ago
sports-NN-32d-lr0.05
0.952
0.879
0.309
0.903
0.828
0.403
0.05
14 minutes ago
sports-NN-8d-lr0.05
0.951
0.88
0.314
0.904
0.824
0.411
0.05
14 minutes ago
sports-NN-8d-lr0.1
0.964
0.894
0.249
0.916
0.819
0.397
0.1
15 minutes ago
sports-NN-8d-lr0.01
0.85
0.78
0.625
0.796
0.724
0.639
0.01
15 minutes ago
sports-NN-32d-lr0.01
0.866
0.766
0.646
0.807
0.712
0.656
0.01
Load more

Comparing 3 Runs

Run ID:	ba87c5548b5a473aab8818dcc957...	84a313504e9b45d7895d8c9f77d4...	0e11b08f73b24fe48a5129219744...
Run Name:	sports-NN-32d-lr0.1	sports-NN-32d-lr0.05	sports-NN-8d-lr0.05
Start Time:	2022-04-19 20:44:18	2022-04-19 20:45:05	2022-04-19 20:46:20

Parameters

batch_size	None	None	None
class_weight	None	None	None
epochs	40	40	40
initial_epoch	0	0	0
kwargs	<class 'inspect._empty'>	<class 'inspect._empty'>	<class 'inspect._empty'>
max_queue_size	10	10	10
opt_decay	0.0	0.0	0.0
opt_learning_rate	0.1	0.05	0.05
opt_momentum	0.0	0.0	0.0
opt_name	SGD	SGD	SGD
opt_nesterov	False	False	False
sample_weight	None	None	None
shuffle	True	True	True
steps_per_epoch	None	None	None
use_multiprocessing	False	False	False
validation_freq	1	1	1
validation_split	0.0	0.0	0.0
validation_steps	None	None	None
workers	1	1	1

/User 8b5a473aab8818dc...

Metrics

AUROC	0.965	0.952	0.951
binary_accuracy	0.894	0.879	0.88
loss	0.245	0.309	0.314
val_AUROC	0.917	0.903	0.904
val_binary_accuracy	0.838	0.828	0.824
val_loss	0.382	0.403	0.411

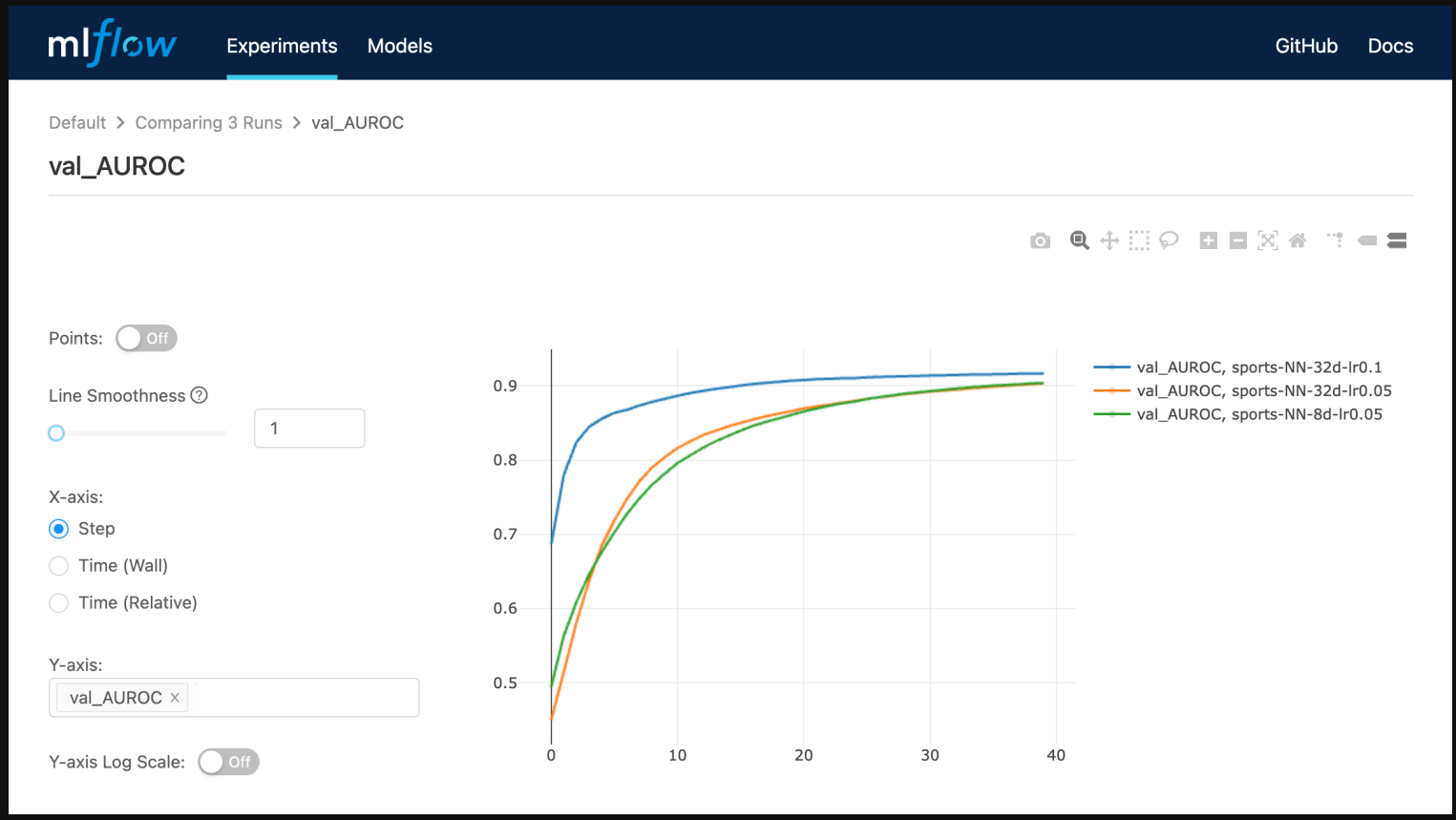
Scatter Plot | Contour Plot | Parallel Coordinates Plot

X-axis:

batch_size

Y-axis:

AUROC


mlflow
Experiments
Models
GitHub
Docs
Default > Comparing 3 Runs > val_AUROC
val_AUROC
Points: Off
Line Smoothness
1
X-axis:
Step
Time (Wall)
Time (Relative)
Y-axis:
val_AUROC
Y-axis Log Scale: Off
0.9
0.8
0.7
0.6
0.5
0
10
20
30
40
val_AUROC, sports-NN-32d-lr0.1
val_AUROC, sports-NN-32d-lr0.05
val_AUROC, sports-NN-8d-lr0.05

Incorporating MLflow into our Model Review Process

Step 1: Training Infrastructure

Maintain a reasonably-flexibly **common training infrastructure** that works for *most* of our team's problems.

Step 2: MLflow Tracking

MLflow Tracking is embedded in this infrastructure, so experiment record-keeping happens automatically:

- all training parameters
- train and test metrics
- environment
    - docker image, code version, training script or notebook
- model (and related artifacts for deployment)
- common plots & analyses

Step 3: Use Notes for Context

MLflow UI Markdown Notes section used for all the context that can't be recorded automatically.

Step 4: Model Review

Prior to Live Launch of any new model, we require a **Model Review**:

- share link to MLflow run early
- select primary & secondary reviewers
- 30 min meeting to discuss, ask questions, make suggestions

Back to our Goals for Model Review

Back to our Goals for Model Review

1. **Transparency** is achieved by having shareable links to all the training details

Back to our Goals for Model Review

1. **Transparency** is achieved by having shareable links to all the training details

1. **Reproducibility** is improved by recording everything needed to re-create a training run

Back to our Goals for Model Review

1. **Transparency** is achieved by having shareable links to all the training details

1. **Reproducibility** is improved by recording everything needed to re-create a training run

1. **Knowledge Sharing** happens through the review process & by simply sharing MLflow examples

Back to our Goals for Model Review

1. **Transparency** is achieved by having shareable links to all the training details

1. **Reproducibility** is improved by recording everything needed to re-create a training run

1. **Knowledge Sharing** happens through the review process & by simply sharing MLflow examples

MLflow is a helpful for this because we can view so many details about the model in one place.

Summary

- MLflow is a lightweight and powerful way to track your ML experiments
    - adding one line of code (autologging) can get you a long way!
    - I only covered Tracking (also check out: MLflow Projects, Models, Model Registry)

Summary

- MLflow is a lightweight and powerful way to track your ML experiments
    - adding one line of code (autologging) can get you a long way!
    - I only covered Tracking (also check out: MLflow Projects, Models, Model Registry)
- We use MLflow Tracking to automatically keep training records & to review models before deployment

Summary

- MLflow is a lightweight and powerful way to track your ML experiments
    - adding one line of code (autologging) can get you a long way!
    - I only covered Tracking (also check out: MLflow Projects, Models, Model Registry)
- We use MLflow Tracking to automatically keep training records & to review models before deployment
- Our review "process" is a WIP and evolving!

Code

This presentation is a notebook on GitHub: github.com/jesford/model-review

Want to work with me?

Cash App is hiring for all kinds of positions: cash.app/careers

Questions?