The Model Review

improving transparency, reproducibility & knowledge sharing with MLflow

Jes Ford, PhD
ML Engineer at Cash App

Hi, Jes Ford here 👋

- Snowboarder 1st 🏂
- Originally from Alaska, currently living in Salt Lake City
- PhD in Physics 🔭
- Machine Learning Engineer (Modeler) at Cash App, Block
- Working on Natural Language Understanding for Customer Support

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

1. **Transparency** and record keeping of what exactly is being deployed
2. **Reproducibility** of past experiments and ease of building off of them
3. **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
2. **Reproducibility** of past experiments and ease of building off of them
3. **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/commits and also (ideally) documentation of changes, design decisions, trade-offs, etc.
- Knowledge sharing between teammates

Code vs Model Review *Similarities*

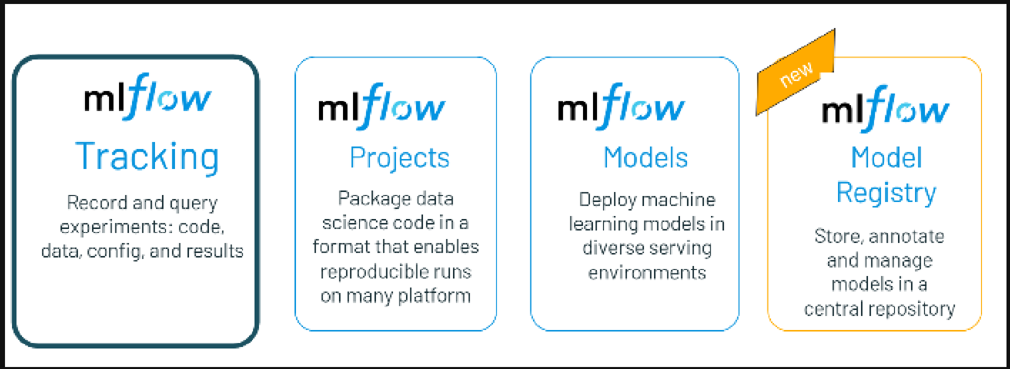
There is code involved.

Code vs Model Review *Differences*

- There is a lot of context beyond the Code Used to train a model
    - data, including any transformations
    - performance
    - entire ML process, including all your failed experiments
- We can't really review a model *just* by reading the final code
- Where to record async review comments? (GitHub not really a good fit for this)

What is MLflow?

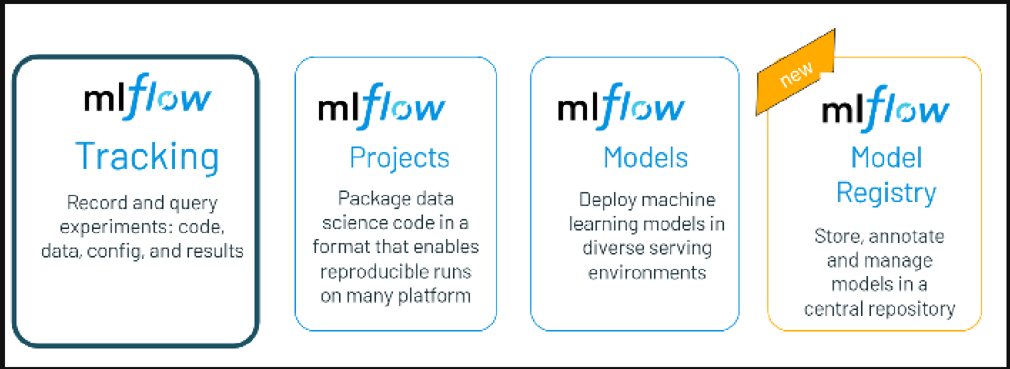
"Open source platform for managing the end-to-end machine learning lifecycle"



Language & library agnostic: includes APIs for Python, R, Java, but everything is accessible throuh a REST API & CLI, so it's very flexible for difference use cases.

What is MLflow?

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



Language & library agnostic: includes APIs for Python, R, Java, but everything is accessible throuh a REST API & CLI, so it's very flexible for difference use cases.

```
$ pip install mlflow
```

MLflow Tracking

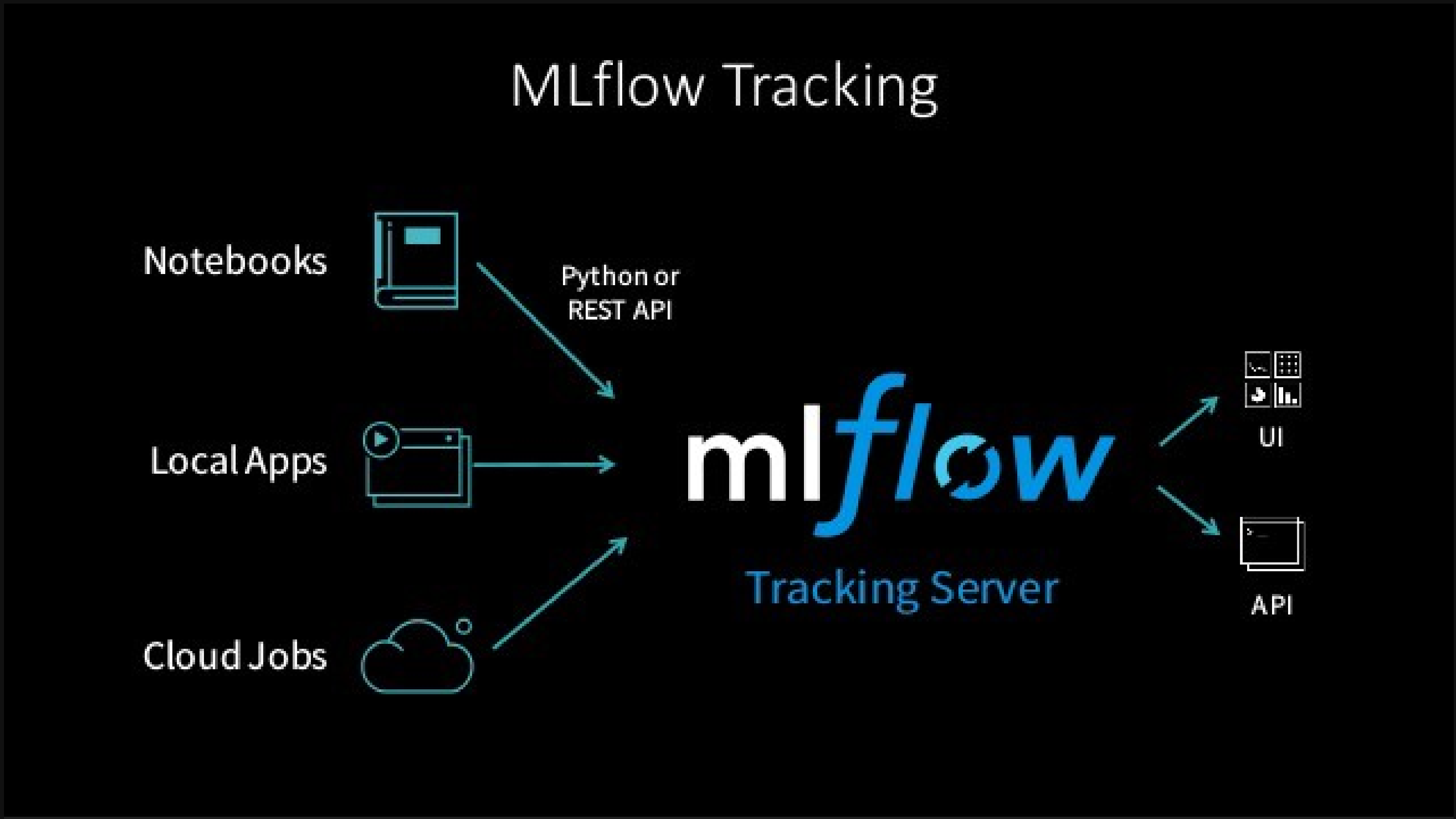
Easily log almost anything you want to keep track of:

- parameters
- metrics
- arbitrary files ("artifacts" in mlflow)
    - such as plots, output files, Jupyter notebooks...
- code version

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

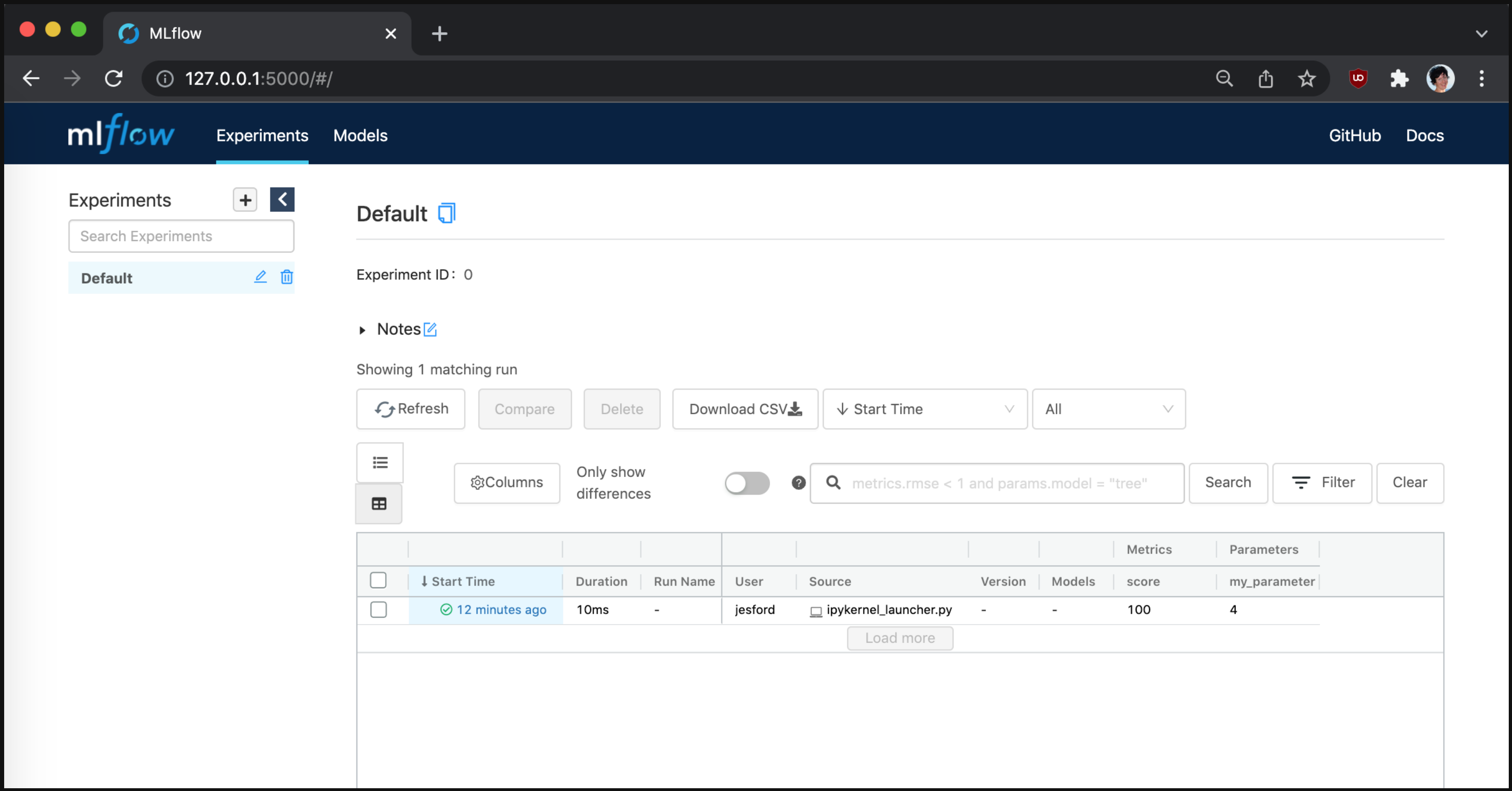
MLflow Tracking UI

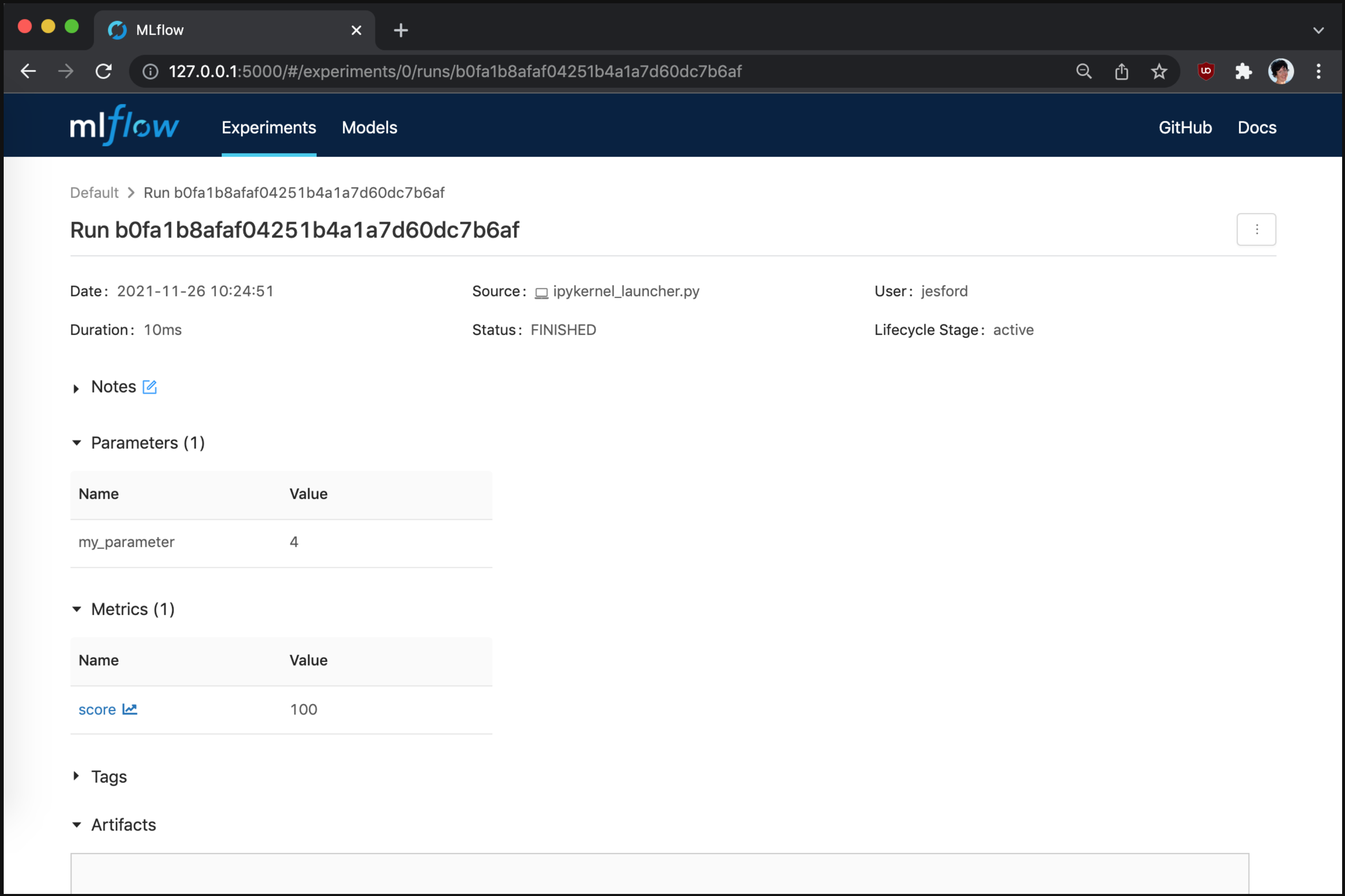
```
$ mlflow ui

[2021-11-26 10:15:48 -0700] [19408] [INFO] Starting gunicorn 20.1.0
[2021-11-26 10:15:48 -0700] [19408] [INFO] Listening at: http://127.0.0.1:5000
(19408)
[2021-11-26 10:15:48 -0700] [19408] [INFO] Using worker: sync
[2021-11-26 10:15:48 -0700] [19411] [INFO] Booting worker with pid: 19411
```

MLflow Tracking UI

```
$ mlflow ui

[2021-11-26 10:15:48 -0700] [19408] [INFO] Starting gunicorn 20.1.0
[2021-11-26 10:15:48 -0700] [19408] [INFO] Listening at: http://127.0.0.1:5000
(19408)
[2021-11-26 10:15:48 -0700] [19408] [INFO] Using worker: sync
[2021-11-26 10:15:48 -0700] [19411] [INFO] Booting worker with pid: 19411
```

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

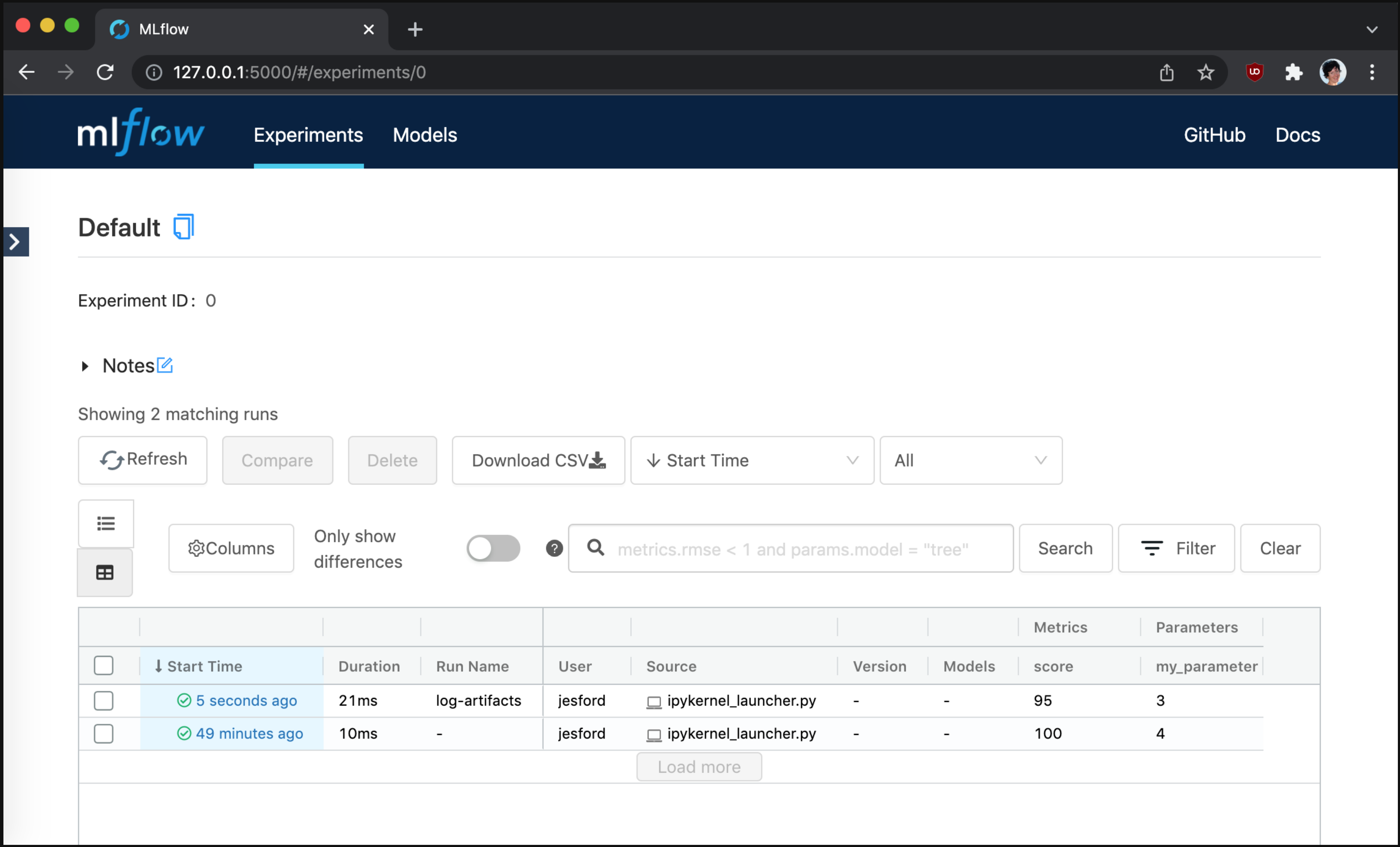

MLflow
127.0.0.1:5000/#/
mlflow
Experiments
Models
GitHub
Docs
Experiments
Search Experiments
Default
Default
Experiment ID : 0
Notes
Showing 1 matching run
Refresh
Compare
Delete
Download CSV
Start Time
All
Columns
Only show differences
metrics.rmse < 1 and params.model = "tree"
Search
Filter
Clear
Metrics
Parameters
Start Time
Duration
Run Name
User
Source
Version
Models
score
my_parameter
12 minutes ago
10ms
-
jesford
ipykernel_launcher.py
-
-
100
4
Load more

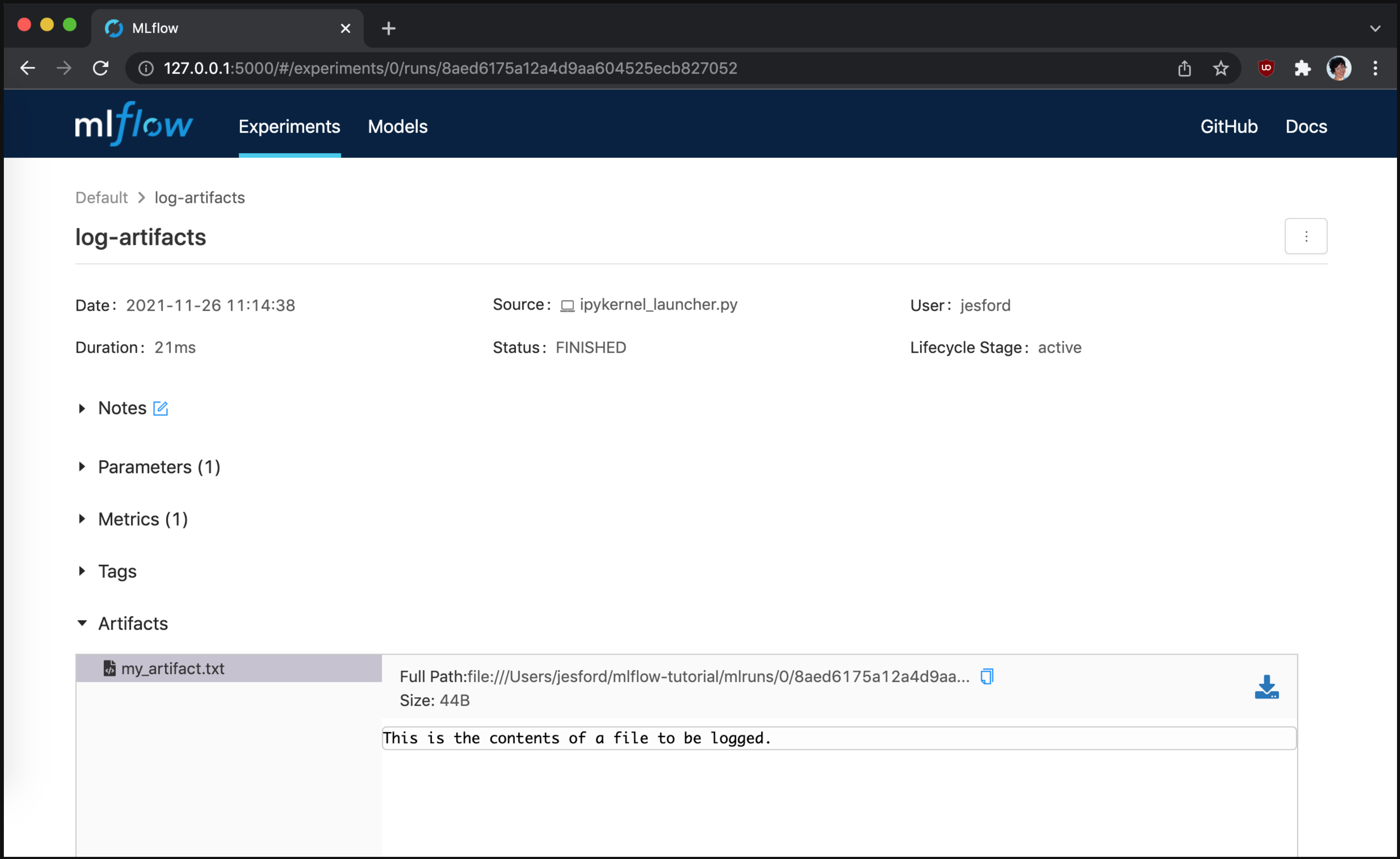

MLflow
127.0.0.1:5000/#/experiments/0/runs/b0fa1b8afaf04251b4a1a7d60dc7b6af
mlflow
Experiments
Models
GitHub
Docs
Default > Run b0fa1b8afaf04251b4a1a7d60dc7b6af
Run b0fa1b8afaf04251b4a1a7d60dc7b6af
Date: 2021-11-26 10:24:51
Source: ipykernel_launcher.py
User: jesford
Duration: 10ms
Status: FINISHED
Lifecycle 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
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.end_run()
```


MLflow
127.0.0.1:5000/#/experiments/0
mlflow
Experiments
Models
GitHub
Docs
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
Clear
Metrics
Parameters
Start Time
Duration
Run Name
User
Source
Version
Models
score
my_parameter
5 seconds ago
21ms
log-artifacts
jesford
ipykernel_launcher.py
-
-
95
3
49 minutes ago
10ms
-
jesford
ipykernel_launcher.py
-
-
100
4
Load more



MLflow
127.0.0.1:5000/#/experiments/0/runs/8aed6175a12a4d9aa604525ecb827052
mlflow
Experiments
Models
GitHub
Docs
Default > log-artifacts
log-artifacts
Date: 2021-11-26 11:14:38
Source: ipykernel_launcher.py
User: jesford
Duration: 21ms
Status: FINISHED
Lifecycle Stage: active
Notes
Parameters (1)
Metrics (1)
Tags
Artifacts
my_artifact.txt
Full Path:file:///Users/jesford/mlflow-tutorial/mlruns/0/8aed6175a12a4d9aa...
Size: 44B
This is the contents of a file to be logged.

Where MLflow data is stored

By default your runs are recorded in files in a local `mlruns/` folder that gets created in your current working directory.

Lots of other options for local and remote tracking (the latter is best for teams / sharing results) - see MLflow docs for possibilities.

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.

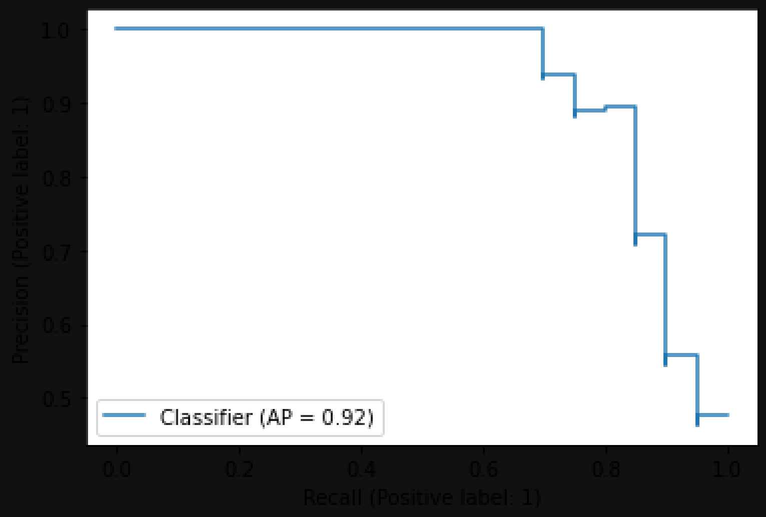
Tracking your ML Model

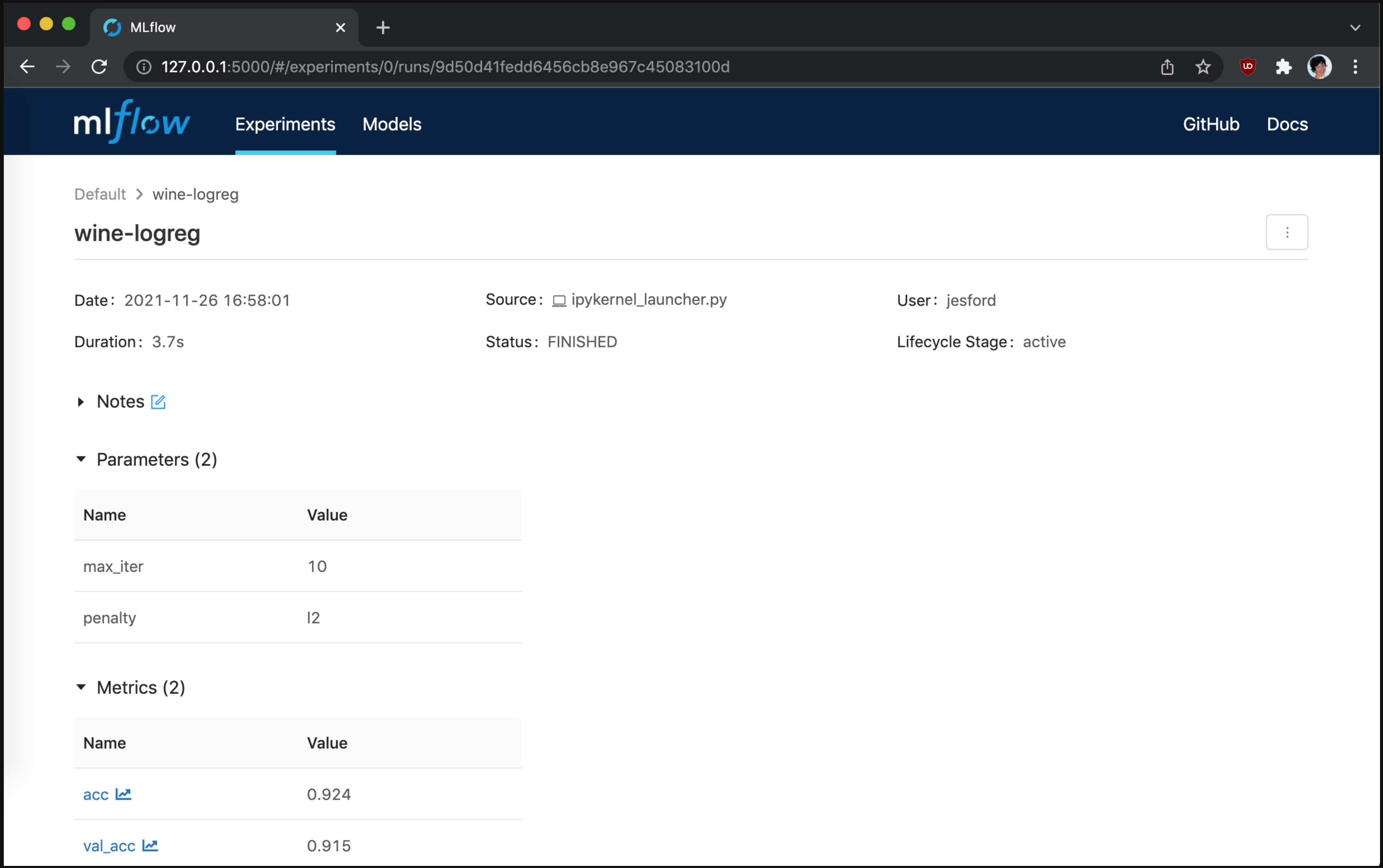
In [3]:

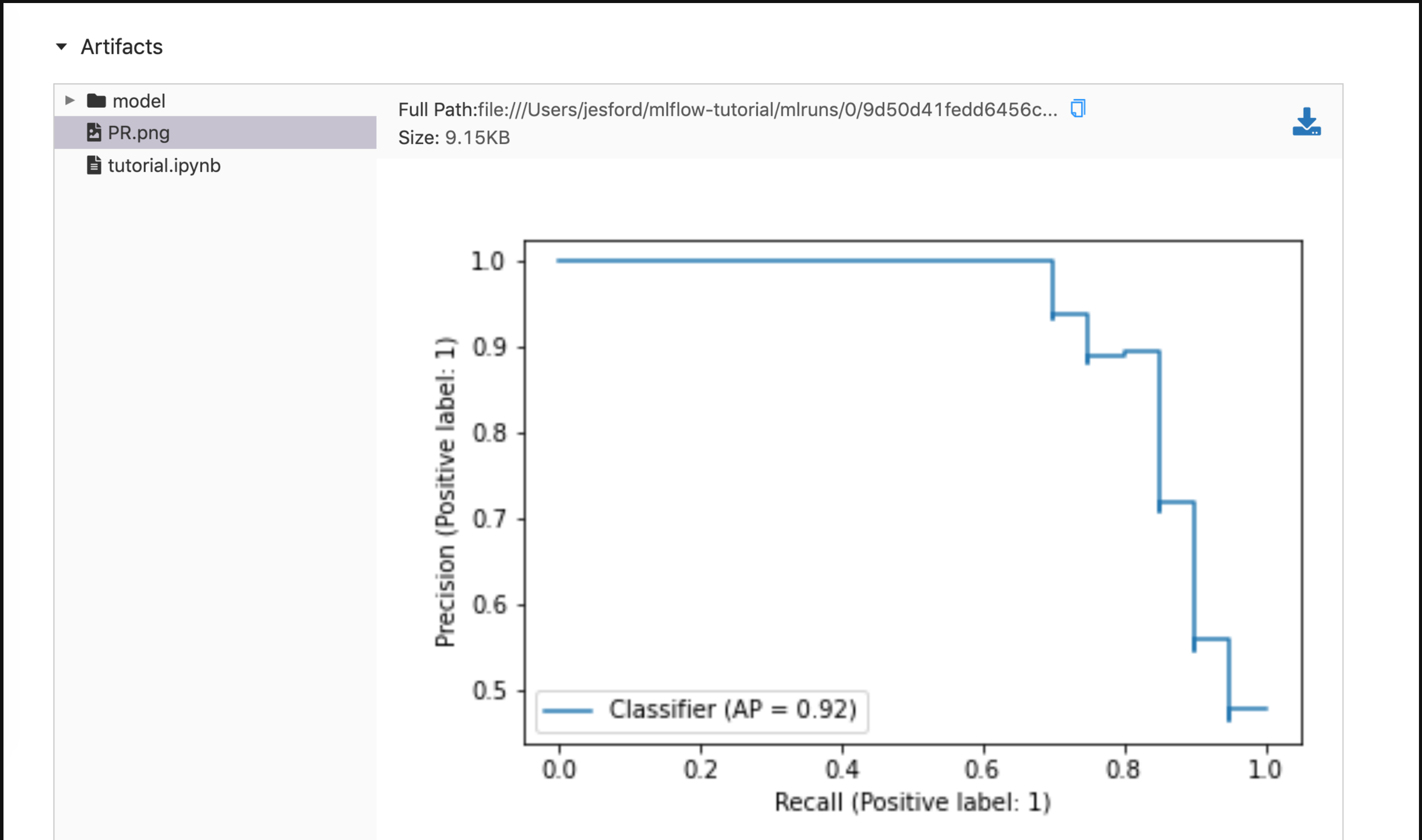
```
import mlflow
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, PrecisionRecallDisplay
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_wine

# load and prepare data
data = load_wine()
binary_target = (data.target == 0).astype(int)
X_train, X_test, y_train, y_test = train_test_split(
    data.data, binary_target, test_size=0.33, shuffle=True, random_state=0,
)

with mlflow.start_run(run_name='wine-logreg'):  # use context manager instead of start/en

    # 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=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')
```

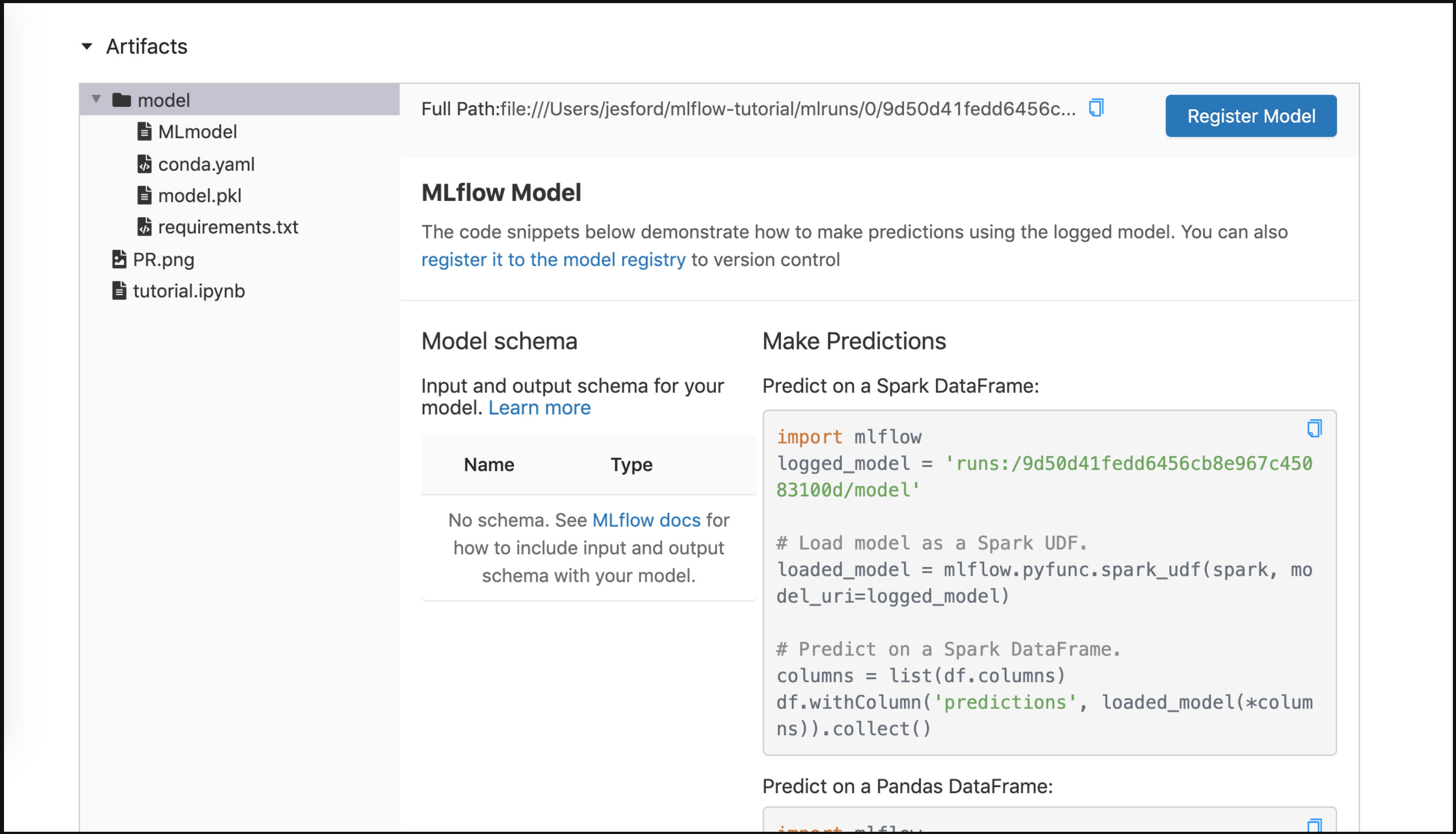
```
/Users/jesford/anaconda3/envs/mlflow-demo/lib/python3.7/site-packages/skle
arn/linear_model/_logistic.py:818: ConvergenceWarning: lbfgs failed to con
verge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```



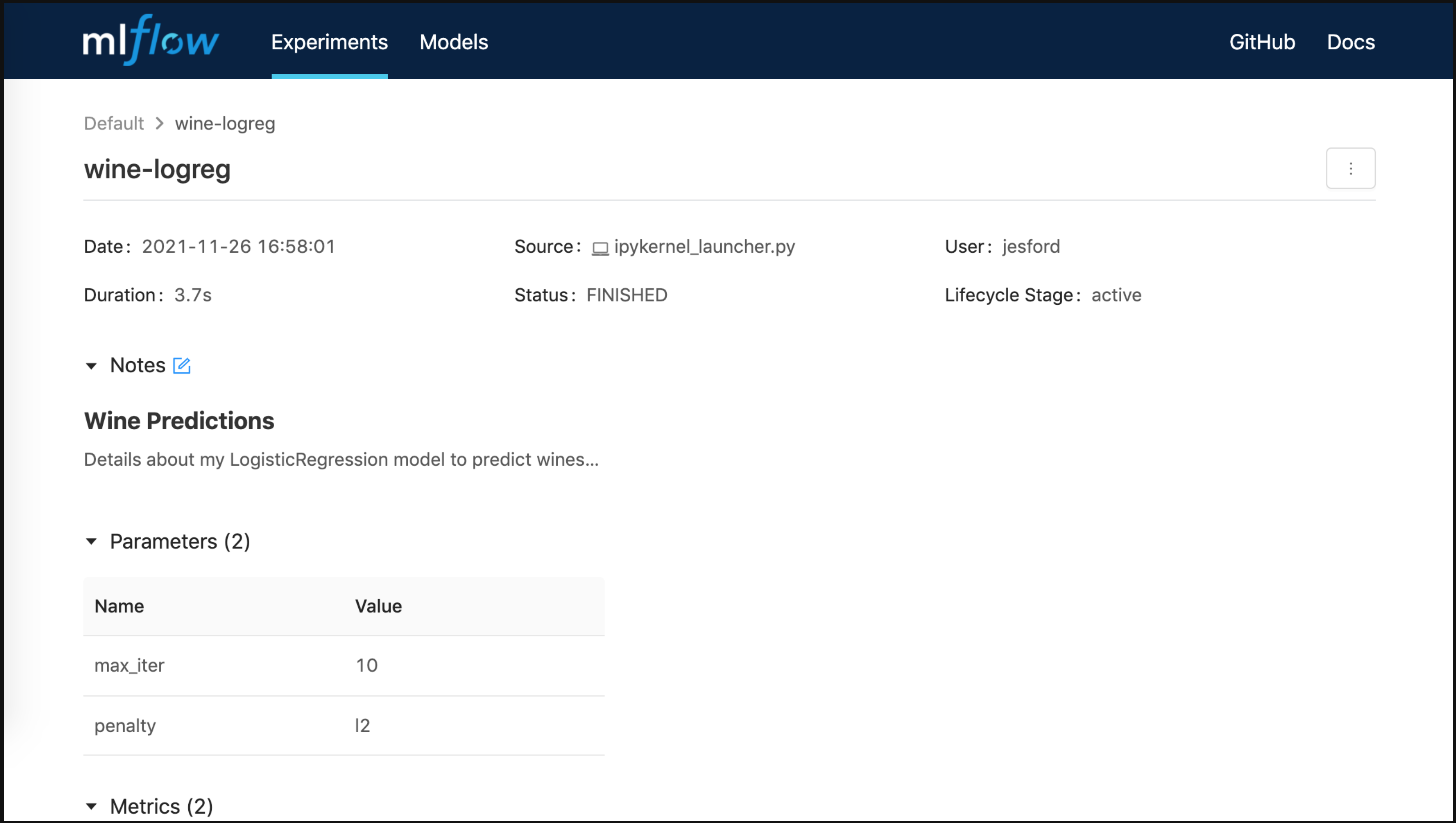

MLflow
127.0.0.1:5000/#/experiments/0/runs/9d50d41fedd6456cb8e967c45083100d
mlflow
Experiments
Models
GitHub
Docs
Default > wine-logreg
wine-logreg
Date: 2021-11-26 16:58:01
Source: ipykernel_launcher.py
User: jesford
Duration: 3.7s
Status: FINISHED
Lifecycle Stage: active
Notes
Parameters (2)
Name
Value
max_iter
10
penalty
l2
Metrics (2)
Name
Value
acc
0.924
val_acc
0.915



Artifacts
model
PR.png
tutorial.ipynb
Full Path:file:///Users/jesford/mlflow-tutorial/mlruns/0/9d50d41fedd6456c...
Size: 9.15KB
1.0
0.9
0.8
0.7
0.6
0.5
Precision (Positive label: 1)
Classifier (AP = 0.92)
0.0
0.2
0.4
0.6
0.8
1.0
Recall (Positive label: 1)



MLflow provides code snippets for loading & using the trained model via the mlflow API


mlflow
Experiments
Models
GitHub
Docs
Default > wine-logreg
wine-logreg
Date: 2021-11-26 16:58:01
Source: ipykernel_launcher.py
User: jesford
Duration: 3.7s
Status: FINISHED
Lifecycle Stage: active
Notes
Wine Predictions
Details about my LogisticRegression model to predict wines...
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.

Note: this is an "experimental" feature in MLflow!

Load & Prepare a Real World Dataset

"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

"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.

In [4]:

```
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import TfidfVectorizer

# 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
)
# turn the text into numbers
vectorizer = TfidfVectorizer(max_features=5000)
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.shape))
print('\nExample document:\n\n{}'.format(X_train_raw[0]))
```

```
Training set size after TF-IDF transform: (1197, 5000)

Example document:

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, April 18.
|> Does anybody know if the Tribe will be in town on those dates, 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 Toronto, and many Toronto fans make the trip to Cleveland as it is easier to get tickets in Cleveland than in Toronto. Either way, I seriously 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

Autologging Example 1: `sklearn`

In [5]:

```
from sklearn.ensemble import RandomForestClassifier

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

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

```
2021/11/27 15:57:24 WARNING mlflow.utils.autologging_utils: You are using
an unsupported version of sklearn. If you encounter errors during autologg
ing, try upgrading / downgrading sklearn to a supported version, or try up
grading MLflow.
2021/11/27 15:57:24 INFO mlflow.utils.autologging_utils: Created MLflow au
tologging run with ID '0767d058f4c34495901426d10fd4d8b0', which will track
hyperparameters, performance metrics, model artifacts, and lineage informa
tion for the current sklearn workflow
/Users/jesford/anaconda3/envs/mlflow-demo/lib/python3.7/site-packages/skle
arn/utils/validation.py:590: FutureWarning: np.matrix usage is deprecated
in 1.0 and will raise a TypeError in 1.2. Please convert to a numpy array
with np.asarray. For more information see: https://numpy.org/doc/stable/re
ference/generated/numpy.matrix.html
  FutureWarning,
/Users/jesford/anaconda3/envs/mlflow-demo/lib/python3.7/site-packages/skle
arn/utils/validation.py:590: FutureWarning: np.matrix usage is deprecated
in 1.0 and will raise a TypeError in 1.2. Please convert to a numpy array
with np.asarray. For more information see: https://numpy.org/doc/stable/re
ference/generated/numpy.matrix.html
  FutureWarning,
/Users/jesford/anaconda3/envs/mlflow-demo/lib/python3.7/site-packages/skle
```

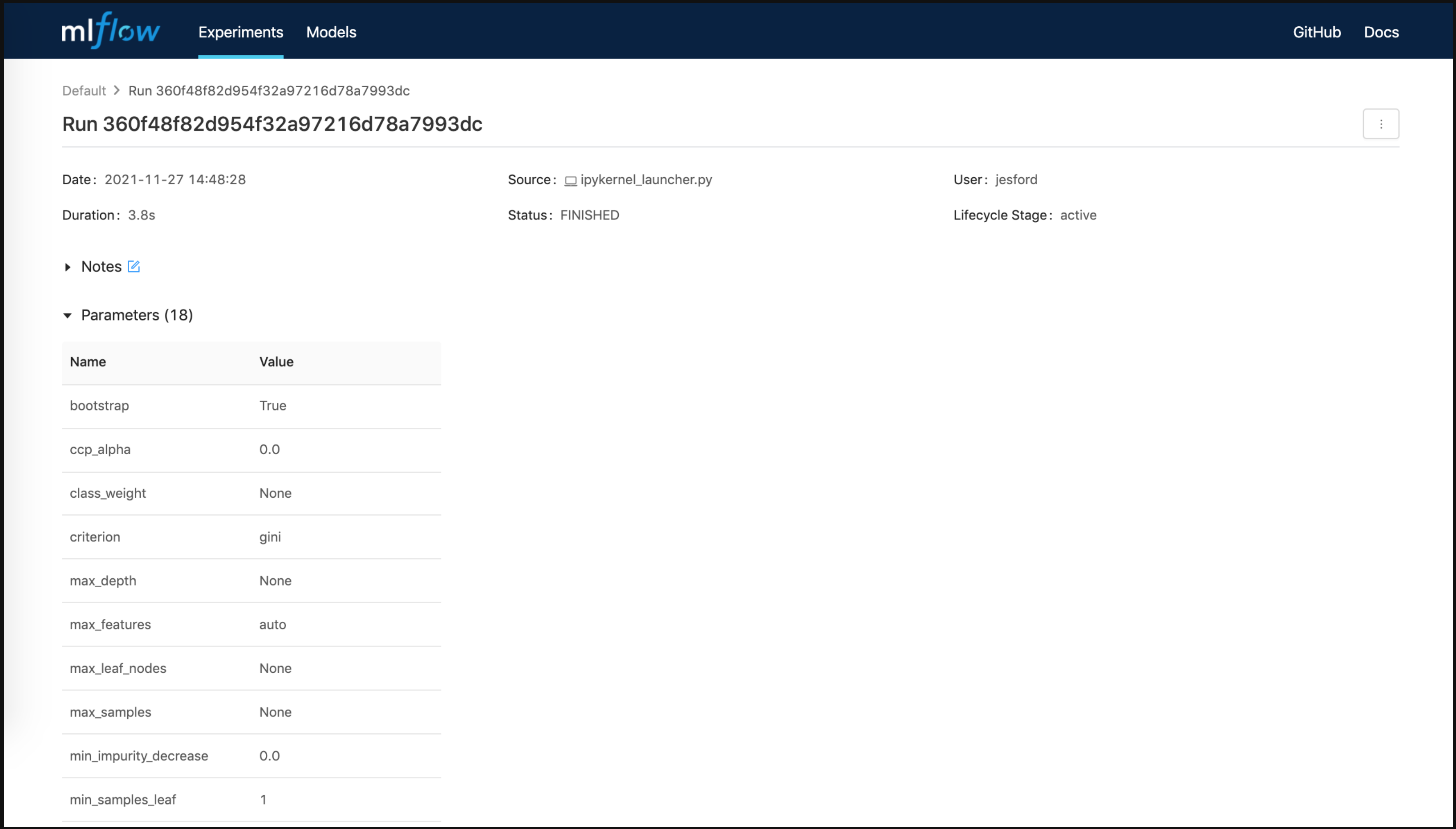
```
arn/utils/validation.py:590: FutureWarning: np.matrix usage is deprecated
in 1.0 and will raise a TypeError in 1.2. Please convert to a numpy array
with np.asarray. For more information see: https://numpy.org/doc/stable/re
ference/generated/numpy.matrix.html
  FutureWarning,
2021/11/27 15:57:25 WARNING mlflow.utils.autologging_utils: MLflow autolog
ging encountered a warning: "/Users/jesford/anaconda3/envs/mlflow-demo/li
b/python3.7/site-packages/sklearn/utils/deprecation.py:87: FutureWarning:
Function plot_confusion_matrix is deprecated; Function `plot_confusion_mat
rix` is deprecated in 1.0 and will be removed in 1.2. Use one of the class
methods: ConfusionMatrixDisplay.from_predictions or ConfusionMatrixDispla
y.from_estimator."
/Users/jesford/anaconda3/envs/mlflow-demo/lib/python3.7/site-packages/skle
arn/utils/validation.py:590: FutureWarning: np.matrix usage is deprecated
in 1.0 and will raise a TypeError in 1.2. Please convert to a numpy array
with np.asarray. For more information see: https://numpy.org/doc/stable/re
ference/generated/numpy.matrix.html
  FutureWarning,
2021/11/27 15:57:25 WARNING mlflow.utils.autologging_utils: MLflow autolog
ging encountered a warning: "/Users/jesford/anaconda3/envs/mlflow-demo/li
b/python3.7/site-packages/sklearn/utils/deprecation.py:87: FutureWarning:
Function plot_roc_curve is deprecated; Function `plot_roc_curve` is deprec
ated in 1.0 and will be removed in 1.2. Use one of the class methods: RocC
urveDisplay.from_predictions or RocCurveDisplay.from_estimator."
/Users/jesford/anaconda3/envs/mlflow-demo/lib/python3.7/site-packages/skle
arn/utils/validation.py:590: FutureWarning: np.matrix usage is deprecated
in 1.0 and will raise a TypeError in 1.2. Please convert to a numpy array
with np.asarray. For more information see: https://numpy.org/doc/stable/re
ference/generated/numpy.matrix.html
  FutureWarning,
2021/11/27 15:57:25 WARNING mlflow.utils.autologging_utils: MLflow autolog
ging encountered a warning: "/Users/jesford/anaconda3/envs/mlflow-demo/li
b/python3.7/site-packages/sklearn/utils/deprecation.py:87: FutureWarning:
```

```
Function plot_precision_recall_curve is deprecated; Function `plot_precisi
on_recall_curve` is deprecated in 1.0 and will be removed in 1.2. Use one
of the class methods: PrecisionRecallDisplay.from_predictions or Precision
RecallDisplay.from_estimator."
/Users/jesford/anaconda3/envs/mlflow-demo/lib/python3.7/site-packages/skle
arn/utils/validation.py:590: FutureWarning: np.matrix usage is deprecated
in 1.0 and will raise a TypeError in 1.2. Please convert to a numpy array
with np.asarray. For more information see: https://numpy.org/doc/stable/re
ference/generated/numpy.matrix.html
  FutureWarning,
/Users/jesford/anaconda3/envs/mlflow-demo/lib/python3.7/site-packages/skle
arn/utils/validation.py:590: FutureWarning: np.matrix usage is deprecated
in 1.0 and will raise a TypeError in 1.2. Please convert to a numpy array
with np.asarray. For more information see: https://numpy.org/doc/stable/re
ference/generated/numpy.matrix.html
  FutureWarning,
/Users/jesford/anaconda3/envs/mlflow-demo/lib/python3.7/site-packages/skle
arn/utils/validation.py:590: FutureWarning: np.matrix usage is deprecated
in 1.0 and will raise a TypeError in 1.2. Please convert to a numpy array
with np.asarray. For more information see: https://numpy.org/doc/stable/re
ference/generated/numpy.matrix.html
  FutureWarning,
```

Out[5]:

```
RandomForestClassifier(n_estimators=10)
```

Back in the MLflow UI...



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	0.997
training_f1_score	0.997
training_log_loss	0.081
training_precision_score	0.997
training_recall_score	0.997
training_roc_auc_score	1
training_score	0.997

We see all RF hyperparameters (even those we didn't set explicitly) are recorded...

... as well as some default training metrics (would have to log validation metrics manually since `.fit` doesn't know about them)

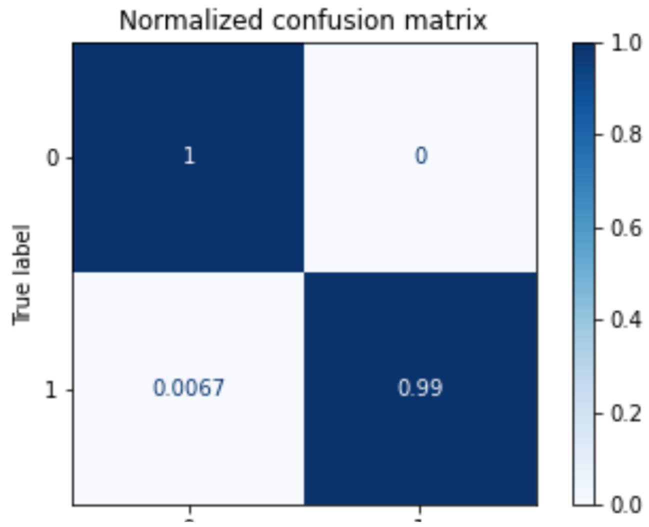
... and we get some really useful plots out of the box!

▾ Artifacts

- model
  - MLmodel
  - conda.yaml
  - model.pkl
  - requirements.txt
- training_confusion_matrix.png
- training_precision_recall_curve.png
- training_roc_curve.png

Full Path:file:///Users/jesford/mlflow-tutorial/mlruns/0/360f48f82d954f32a97216d78a7993dc/artifacts/training_confusion_matrix.png

Size: 8.6KB



Autologging Example 2: `tensorflow.keras`

In [6]:

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

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

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

    # parameters to vary
    hidden_layer_size = 16
    learning_rate = 0.1

    # 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_data=(X_test, y_test)
```

```
2021-11-27 15:57:29.940299: I tensorflow/core/platform/cpu_feature_guard.c
c:145] This TensorFlow binary is optimized with Intel(R) MKL-DNN to use th
```

```
e following CPU instructions in performance critical operations:  SSE4.1 S
SE4.2 AVX AVX2 FMA
To enable them in non-MKL-DNN operations, rebuild TensorFlow with the appr
opriate compiler flags.
2021-11-27 15:57:29.940539: I tensorflow/core/common_runtime/process_util.
cc:115] Creating new thread pool with default inter op setting: 4. Tune us
ing inter_op_parallelism_threads for best performance.
```

```
Train on 1197 samples, validate on 796 samples
Epoch 1/40
 864/1197 [====================>.........] - ETA: 1s - loss: 0.6932 - bina
ry_accuracy: 0.5000 - AUROC: 0.4967
```

```
2021-11-27 15:57:32.775138: I tensorflow/core/profiler/lib/profiler_sessio
n.cc:184] Profiler session started.
```

```
1197/1197 [==============================] - 3s 3ms/sample - loss: 0.6927
- binary_accuracy: 0.5138 - AUROC: 0.5178 - val_loss: 0.6923 - val_binary_
accuracy: 0.5038 - val_AUROC: 0.5648
Epoch 2/40
1197/1197 [==============================] - 0s 407us/sample - loss: 0.688
5 - binary_accuracy: 0.5556 - AUROC: 0.6514 - val_loss: 0.6881 - val_binar
y_accuracy: 0.6583 - val_AUROC: 0.7370
Epoch 3/40
1197/1197 [==============================] - 1s 508us/sample - loss: 0.682
3 - binary_accuracy: 0.6942 - AUROC: 0.8035 - val_loss: 0.6834 - val_binar
y_accuracy: 0.6118 - val_AUROC: 0.8632
Epoch 4/40
1197/1197 [==============================] - 0s 372us/sample - loss: 0.674
2 - binary_accuracy: 0.7853 - AUROC: 0.8988 - val_loss: 0.6772 - val_binar
y_accuracy: 0.6093 - val_AUROC: 0.9249
Epoch 5/40
1197/1197 [==============================] - 1s 477us/sample - loss: 0.664
7 - binary_accuracy: 0.8755 - AUROC: 0.9448 - val_loss: 0.6685 - val_binar
```

```
y_accuracy: 0.8894 - val_AUROC: 0.9536
Epoch 6/40
1197/1197 [==============================] - 1s 500us/sample - loss: 0.651
4 - binary_accuracy: 0.8881 - AUROC: 0.9638 - val_loss: 0.6623 - val_binar
y_accuracy: 0.5239 - val_AUROC: 0.9693
Epoch 7/40
1197/1197 [==============================] - 1s 420us/sample - loss: 0.638
1 - binary_accuracy: 0.8688 - AUROC: 0.9742 - val_loss: 0.6469 - val_binar
y_accuracy: 0.9146 - val_AUROC: 0.9752
Epoch 8/40
1197/1197 [==============================] - 0s 333us/sample - loss: 0.618
5 - binary_accuracy: 0.9206 - AUROC: 0.9831 - val_loss: 0.6327 - val_binar
y_accuracy: 0.9058 - val_AUROC: 0.9774
Epoch 9/40
1197/1197 [==============================] - 1s 492us/sample - loss: 0.596
7 - binary_accuracy: 0.9532 - AUROC: 0.9926 - val_loss: 0.6161 - val_binar
y_accuracy: 0.8555 - val_AUROC: 0.9809
Epoch 10/40
1197/1197 [==============================] - 1s 440us/sample - loss: 0.570
7 - binary_accuracy: 0.9674 - AUROC: 0.9948 - val_loss: 0.5978 - val_binar
y_accuracy: 0.8028 - val_AUROC: 0.9823
Epoch 11/40
1197/1197 [==============================] - 0s 341us/sample - loss: 0.541
2 - binary_accuracy: 0.9524 - AUROC: 0.9939 - val_loss: 0.5704 - val_binar
y_accuracy: 0.9384 - val_AUROC: 0.9830
Epoch 12/40
1197/1197 [==============================] - 0s 388us/sample - loss: 0.507
6 - binary_accuracy: 0.9657 - AUROC: 0.9961 - val_loss: 0.5457 - val_binar
y_accuracy: 0.9196 - val_AUROC: 0.9838
Epoch 13/40
1197/1197 [==============================] - 1s 698us/sample - loss: 0.471
1 - binary_accuracy: 0.9716 - AUROC: 0.9959 - val_loss: 0.5161 - val_binar
y_accuracy: 0.9121 - val_AUROC: 0.9851
```

```
Epoch 14/40
1197/1197 [==============================] - 1s 433us/sample - loss: 0.432
3 - binary_accuracy: 0.9758 - AUROC: 0.9974 - val_loss: 0.4851 - val_binar
y_accuracy: 0.9460 - val_AUROC: 0.9856
Epoch 15/40
1197/1197 [==============================] - 0s 410us/sample - loss: 0.394
7 - binary_accuracy: 0.9733 - AUROC: 0.9979 - val_loss: 0.4630 - val_binar
y_accuracy: 0.9146 - val_AUROC: 0.9860
Epoch 16/40
1197/1197 [==============================] - 0s 363us/sample - loss: 0.358
8 - binary_accuracy: 0.9816 - AUROC: 0.9980 - val_loss: 0.4260 - val_binar
y_accuracy: 0.9472 - val_AUROC: 0.9869
Epoch 17/40
1197/1197 [==============================] - 1s 446us/sample - loss: 0.323
0 - binary_accuracy: 0.9766 - AUROC: 0.9985 - val_loss: 0.4081 - val_binar
y_accuracy: 0.9008 - val_AUROC: 0.9875
Epoch 18/40
1197/1197 [==============================] - 0s 387us/sample - loss: 0.293
3 - binary_accuracy: 0.9749 - AUROC: 0.9985 - val_loss: 0.3729 - val_binar
y_accuracy: 0.9485 - val_AUROC: 0.9880
Epoch 19/40
1197/1197 [==============================] - 1s 463us/sample - loss: 0.264
2 - binary_accuracy: 0.9841 - AUROC: 0.9990 - val_loss: 0.3501 - val_binar
y_accuracy: 0.9497 - val_AUROC: 0.9886
Epoch 20/40
1197/1197 [==============================] - 1s 477us/sample - loss: 0.239
7 - binary_accuracy: 0.9891 - AUROC: 0.9993 - val_loss: 0.3304 - val_binar
y_accuracy: 0.9485 - val_AUROC: 0.9891
Epoch 21/40
1197/1197 [==============================] - 0s 352us/sample - loss: 0.216
5 - binary_accuracy: 0.9891 - AUROC: 0.9991 - val_loss: 0.3101 - val_binar
y_accuracy: 0.9447 - val_AUROC: 0.9896
Epoch 22/40
```

```
1197/1197 [==============================] - 0s 358us/sample - loss: 0.196
8 - binary_accuracy: 0.9866 - AUROC: 0.9994 - val_loss: 0.3046 - val_binar
y_accuracy: 0.9234 - val_AUROC: 0.9900
Epoch 23/40
1197/1197 [==============================] - 0s 340us/sample - loss: 0.178
8 - binary_accuracy: 0.9908 - AUROC: 0.9995 - val_loss: 0.2832 - val_binar
y_accuracy: 0.9560 - val_AUROC: 0.9903
Epoch 24/40
1197/1197 [==============================] - 0s 336us/sample - loss: 0.164
9 - binary_accuracy: 0.9925 - AUROC: 0.9997 - val_loss: 0.2640 - val_binar
y_accuracy: 0.9523 - val_AUROC: 0.9906
Epoch 25/40
1197/1197 [==============================] - 0s 345us/sample - loss: 0.151
6 - binary_accuracy: 0.9925 - AUROC: 0.9998 - val_loss: 0.2530 - val_binar
y_accuracy: 0.9447 - val_AUROC: 0.9908
Epoch 26/40
1197/1197 [==============================] - 0s 345us/sample - loss: 0.140
5 - binary_accuracy: 0.9925 - AUROC: 0.9997 - val_loss: 0.2417 - val_binar
y_accuracy: 0.9485 - val_AUROC: 0.9911
Epoch 27/40
1197/1197 [==============================] - 0s 338us/sample - loss: 0.129
3 - binary_accuracy: 0.9958 - AUROC: 0.9998 - val_loss: 0.2332 - val_binar
y_accuracy: 0.9598 - val_AUROC: 0.9914
Epoch 28/40
1197/1197 [==============================] - 0s 343us/sample - loss: 0.120
8 - binary_accuracy: 0.9958 - AUROC: 0.9999 - val_loss: 0.2232 - val_binar
y_accuracy: 0.9485 - val_AUROC: 0.9917
Epoch 29/40
1197/1197 [==============================] - 0s 348us/sample - loss: 0.112
2 - binary_accuracy: 0.9958 - AUROC: 0.9999 - val_loss: 0.2139 - val_binar
y_accuracy: 0.9548 - val_AUROC: 0.9919
Epoch 30/40
1197/1197 [==============================] - 1s 455us/sample - loss: 0.104
```

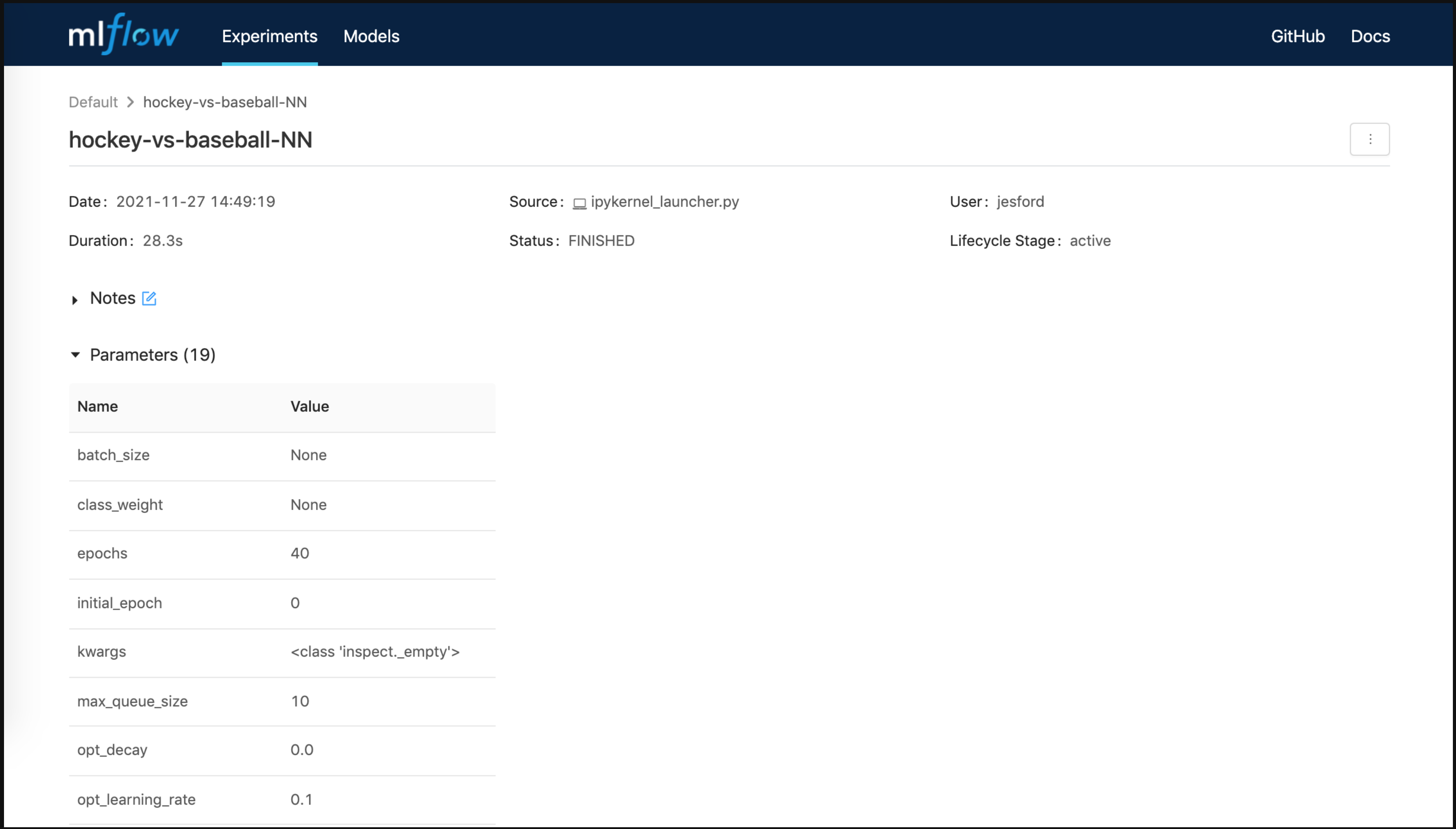
```
7 - binary_accuracy: 0.9950 - AUROC: 0.9999 - val_loss: 0.2087 - val_binar
y_accuracy: 0.9472 - val_AUROC: 0.9921
Epoch 31/40
1197/1197 [==============================] - 0s 350us/sample - loss: 0.097
8 - binary_accuracy: 0.9958 - AUROC: 0.9999 - val_loss: 0.2000 - val_binar
y_accuracy: 0.9560 - val_AUROC: 0.9923
Epoch 32/40
1197/1197 [==============================] - 1s 545us/sample - loss: 0.092
3 - binary_accuracy: 0.9975 - AUROC: 0.9999 - val_loss: 0.1935 - val_binar
y_accuracy: 0.9560 - val_AUROC: 0.9926
Epoch 33/40
1197/1197 [==============================] - 0s 398us/sample - loss: 0.086
7 - binary_accuracy: 0.9958 - AUROC: 1.0000 - val_loss: 0.1883 - val_binar
y_accuracy: 0.9560 - val_AUROC: 0.9928
Epoch 34/40
1197/1197 [==============================] - 1s 483us/sample - loss: 0.081
6 - binary_accuracy: 0.9983 - AUROC: 1.0000 - val_loss: 0.1827 - val_binar
y_accuracy: 0.9585 - val_AUROC: 0.9930
Epoch 35/40
1197/1197 [==============================] - 1s 510us/sample - loss: 0.076
8 - binary_accuracy: 0.9992 - AUROC: 1.0000 - val_loss: 0.1781 - val_binar
y_accuracy: 0.9548 - val_AUROC: 0.9931
Epoch 36/40
1197/1197 [==============================] - 1s 483us/sample - loss: 0.072
9 - binary_accuracy: 0.9992 - AUROC: 1.0000 - val_loss: 0.1735 - val_binar
y_accuracy: 0.9585 - val_AUROC: 0.9933
Epoch 37/40
1197/1197 [==============================] - 1s 493us/sample - loss: 0.068
9 - binary_accuracy: 0.9992 - AUROC: 1.0000 - val_loss: 0.1700 - val_binar
y_accuracy: 0.9648 - val_AUROC: 0.9934
Epoch 38/40
1197/1197 [==============================] - 1s 484us/sample - loss: 0.065
9 - binary_accuracy: 0.9992 - AUROC: 1.0000 - val_loss: 0.1662 - val_binar
```

```
y_accuracy: 0.9648 - val_AUROC: 0.9935
Epoch 39/40
1197/1197 [==============================] - 1s 454us/sample - loss: 0.061
6 - binary_accuracy: 0.9992 - AUROC: 1.0000 - val_loss: 0.1638 - val_binar
y_accuracy: 0.9636 - val_AUROC: 0.9936
Epoch 40/40
1197/1197 [==============================] - 1s 483us/sample - loss: 0.059
1 - binary_accuracy: 0.9992 - AUROC: 1.0000 - val_loss: 0.1598 - val_binar
y_accuracy: 0.9661 - val_AUROC: 0.9938
WARNING:tensorflow:From /Users/jesford/anaconda3/envs/mlflow-demo/lib/pyth
on3.7/site-packages/tensorflow_core/python/ops/resource_variable_ops.py:17
81: calling BaseResourceVariable.__init__ (from tensorflow.python.ops.reso
urce_variable_ops) with constraint is deprecated and will be removed in a
future version.
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
```

```
2021-11-27 15:57:54.053267: W tensorflow/python/util/util.cc:299] Sets are
not currently considered sequences, but this may change in the future, so
consider avoiding using them.
```

```
INFO:tensorflow:Assets written to: /var/folders/xv/48y52h7j27g9jp0vs5n4s8r
c0000gn/T/tmph4l1rgnb/model/data/model/assets
```

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



mlflow Experiments Models GitHub Docs

Default > hockey-vs-baseball-NN

hockey-vs-baseball-NN

Date: 2021-11-27 14:49:19
Source: ipykernel_launcher.py
User: jesford
Duration: 28.3s
Status: FINISHED
Lifecycle Stage: active

Notes

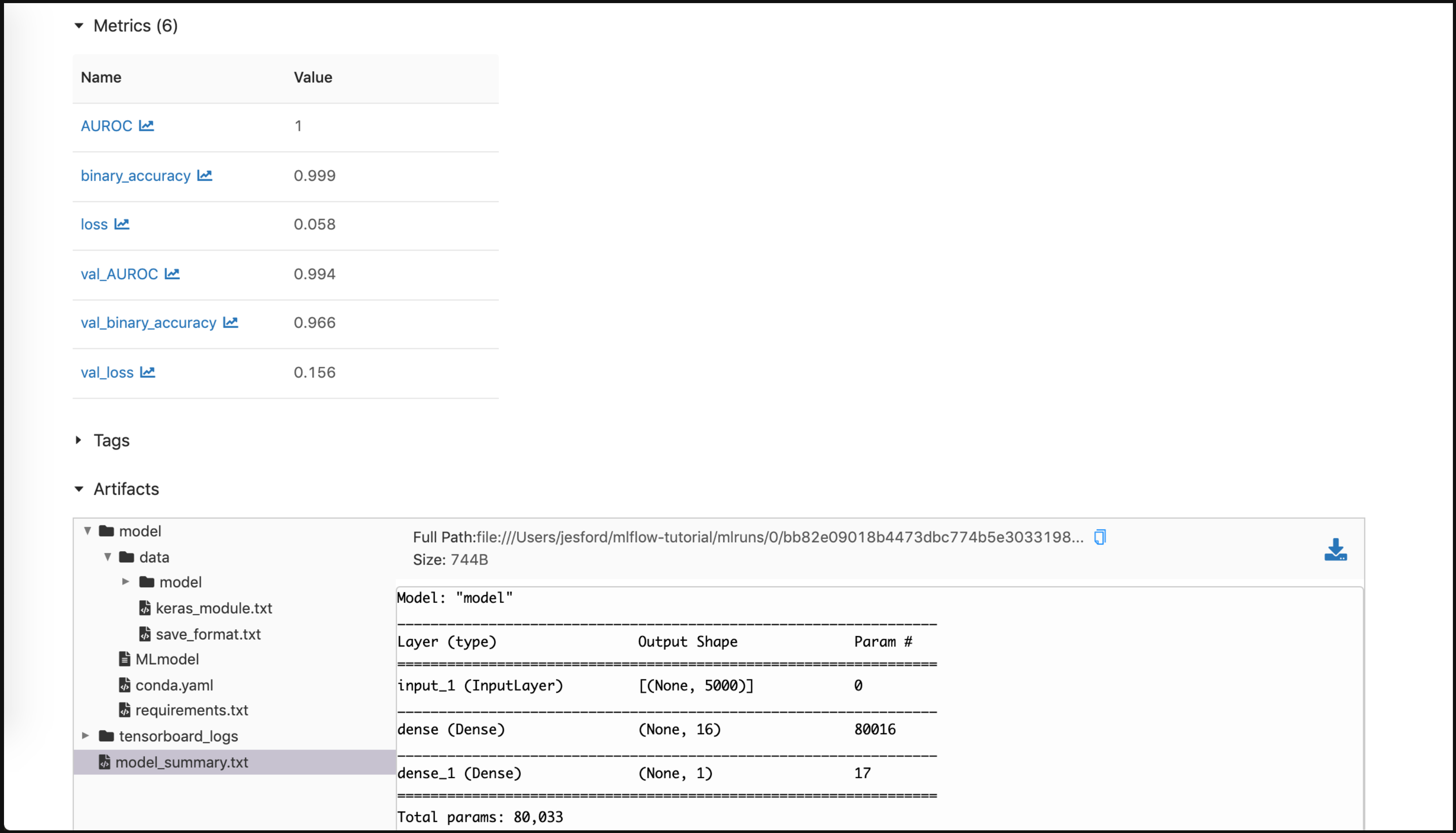
Parameters (19)

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)

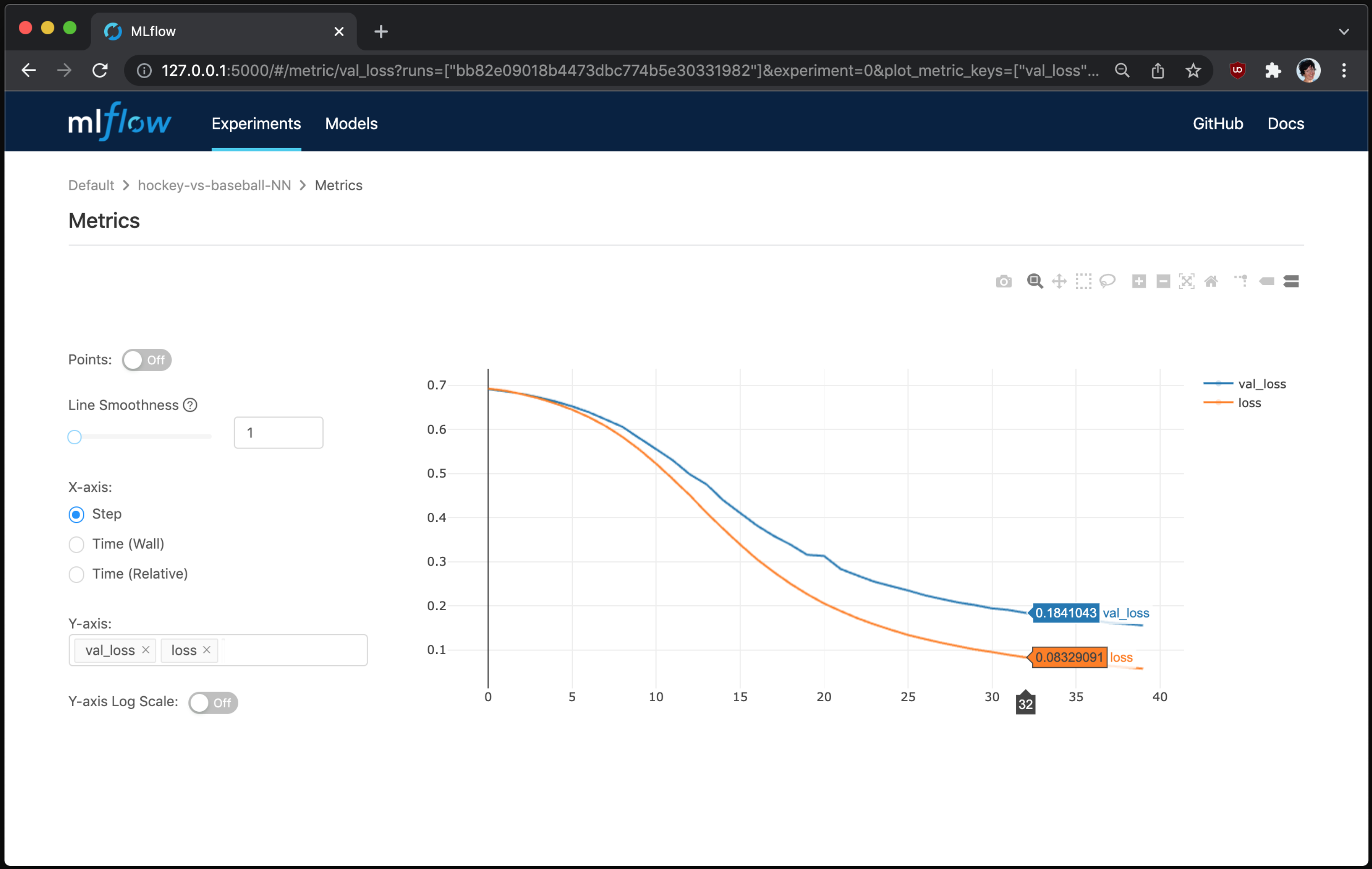
Name	Value
AUROC	1
binary_accuracy	0.999
loss	0.058



Notice model summary, tensorboard logs, and the TF saved model

Note: our team uses this saved model artifact (the inner model folder) directly for deployment, since our infrastructure isn't yet set up to deploy from MLflow directly.

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



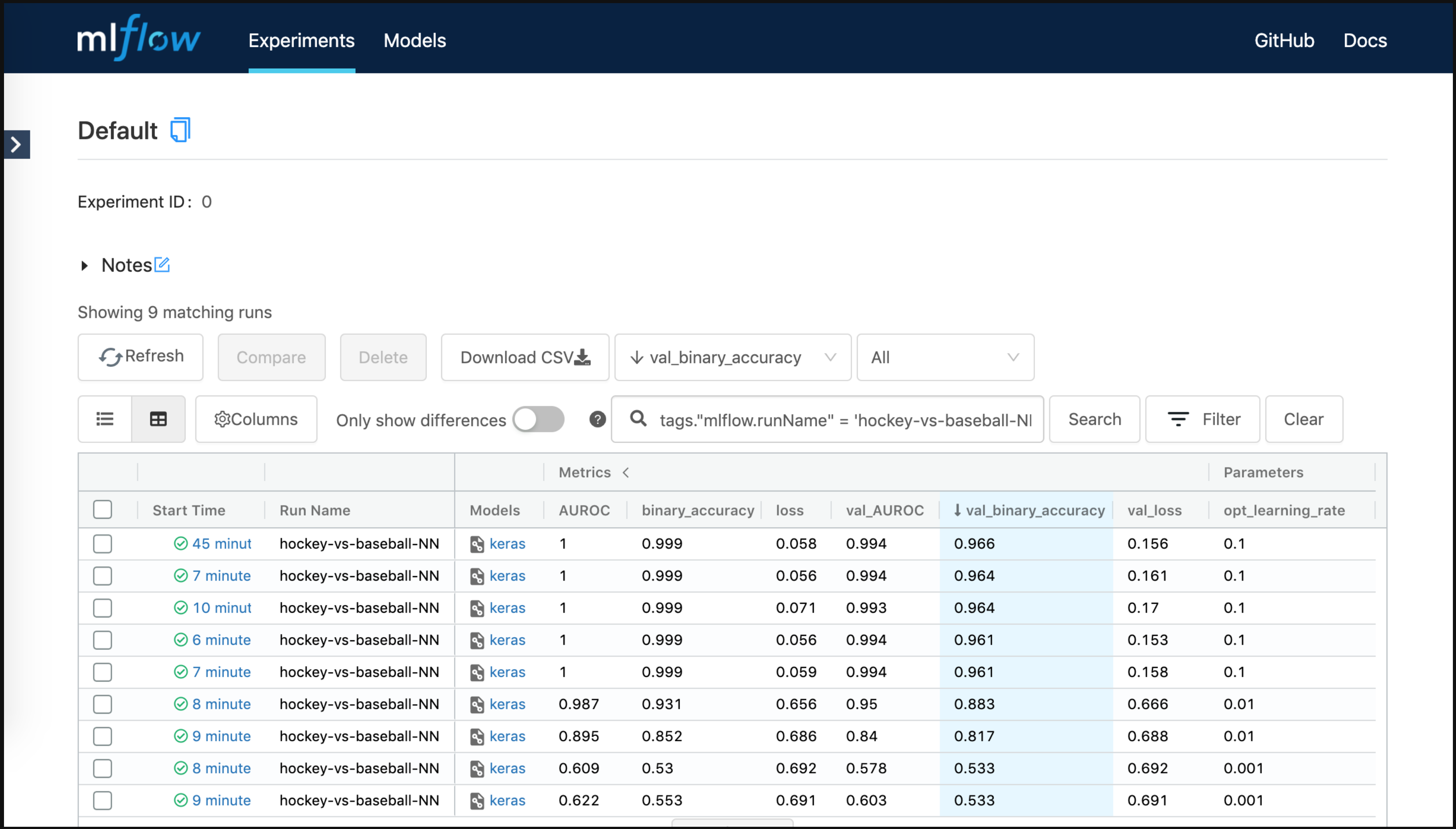
Powerful features for comparing lots of models

Suppose you had run many iterations of this small NN with different hyperparameters...

Powerful features for comparing lots of models

Suppose you had run many iterations of this small NN with different hyperparameters...

In the UI you can sort by metrics



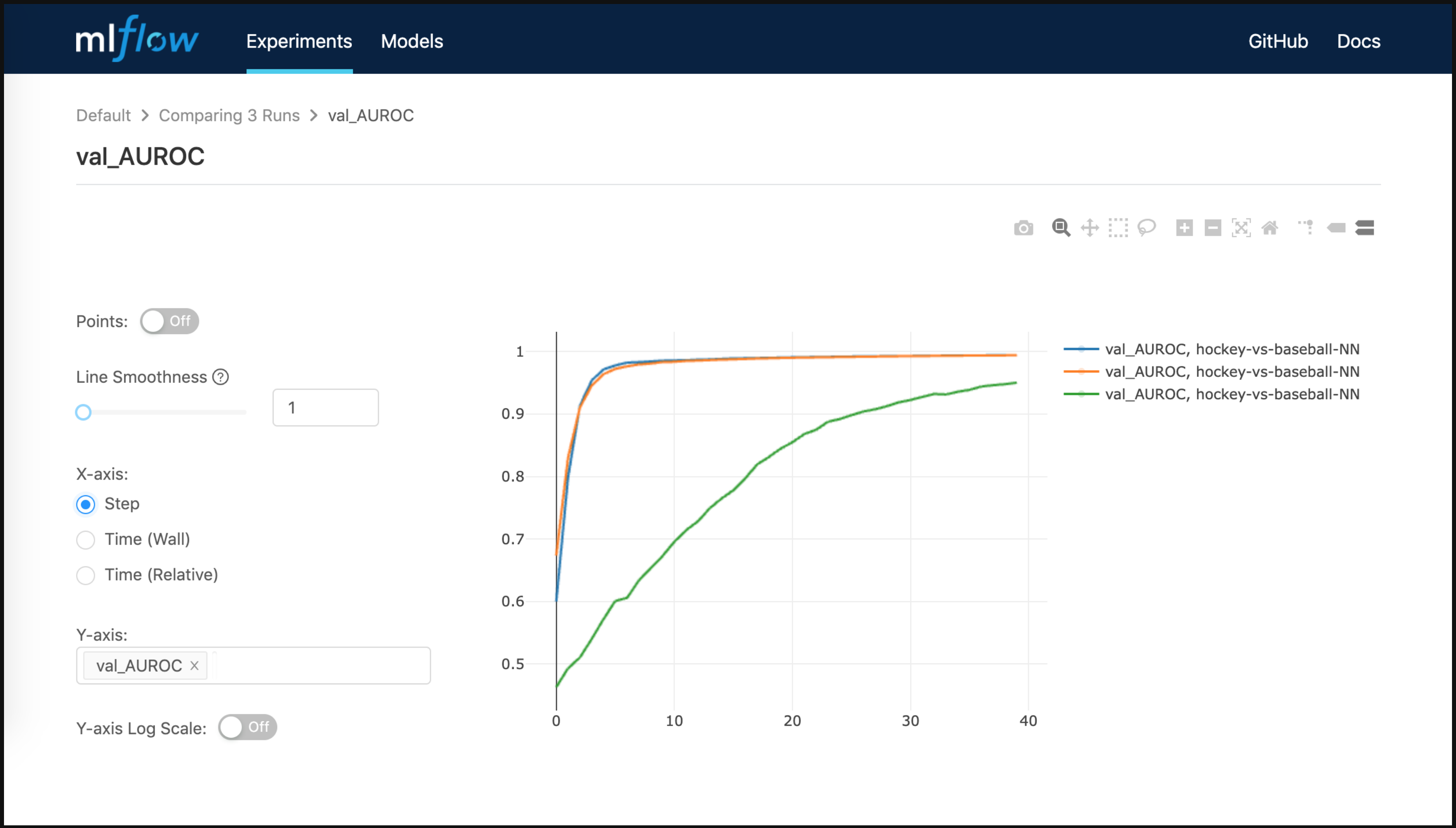
Start Time	Run Name	Models	AUROC	binary_accuracy	loss	val_AUROC	↓ val_binary_accuracy	val_loss	opt_learning_rate
45 minut	hockey-vs-baseball-NN	keras	1	0.999	0.058	0.994	0.966	0.156	0.1
7 minute	hockey-vs-baseball-NN	keras	1	0.999	0.056	0.994	0.964	0.161	0.1
10 minut	hockey-vs-baseball-NN	keras	1	0.999	0.071	0.993	0.964	0.17	0.1
6 minute	hockey-vs-baseball-NN	keras	1	0.999	0.056	0.994	0.961	0.153	0.1
7 minute	hockey-vs-baseball-NN	keras	1	0.999	0.059	0.994	0.961	0.158	0.1
8 minute	hockey-vs-baseball-NN	keras	0.987	0.931	0.656	0.95	0.883	0.666	0.01
9 minute	hockey-vs-baseball-NN	keras	0.895	0.852	0.686	0.84	0.817	0.688	0.01
8 minute	hockey-vs-baseball-NN	keras	0.609	0.53	0.692	0.578	0.533	0.692	0.001
9 minute	hockey-vs-baseball-NN	keras	0.622	0.553	0.691	0.603	0.533	0.691	0.001

... you also view run details side-by-side

By selecting a few checkboxes and clicking the "Compare" button. Notice parameter differences are highlighted!

... and plot training curves from different runs

Similar to TensorBoard.



Incorporating MLflow into our Model Review Process

Step 1: Training Infrastructure

Create/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 & settings
- train and test metrics
- environment: the docker image, code version, the training script or notebook itself
- model (and related artifacts needed for deployment)
- common plots and results of analyses that help us understand performance

Step 3: Use Notes for Context

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

- business requirements, e.g. links to any project docs
- things we tried that didn't work
- describe any non-standard treatment of the data
- decisions or trade-offs made along the way
- if shadow deployed, any observations on live data
- anything you'd want to know if you were picking up the project from scratch

Step 4: Model Review

Prior to Live Launch of a new model, trigger a **Model Review**:

- make sure all "required" fields were tracked and fill out the Notes section
- request a primary and secondary reviewer and send them the link to the MLflow run
- reviewers expected to review the MLflow run & related code async
- schedule a 30min *synchronous* review, other team members invited as optional
- meeting is to discuss, ask questions, make suggestions (rarely to block deployment)

Back to our Goals for Model Review

1. **Transparency** is acheived by having shareable links to all the training details.
2. **Reproducibility** is improved by recording everything needed to re-create a training run.
3. **Knowledge Sharing** happens through the review process or simply by sharing MLflow examples with colleagues.

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

Summary

- We improved model deployment record keeping by integrating MLflow tracking into our shared training infrastructure.
- MLflow is a lightweight and powerful way to track your ML experiments
    - adding one line of code (autologging) can get you a long way!
    - we only talked about Tracking here (see also: MLflow Projects, Models, Model Registry)
- Our review "process" is a WIP! I'd *really* love to talk to anyone who has different processes in place for things like this!

Questions? 🤔

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