

# The Model Review

*improving transparency, reproducibility & knowledge sharing with MLflow*



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

# Hi, Jes Ford here 🙌

- Snowboarder 1st 
- 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?

***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
- 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

- 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"



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

image source: [databricks blog](#)

```
$ pip install mlflow
```

# MLflow Tracking

Easily log almost anything you want to keep track of:

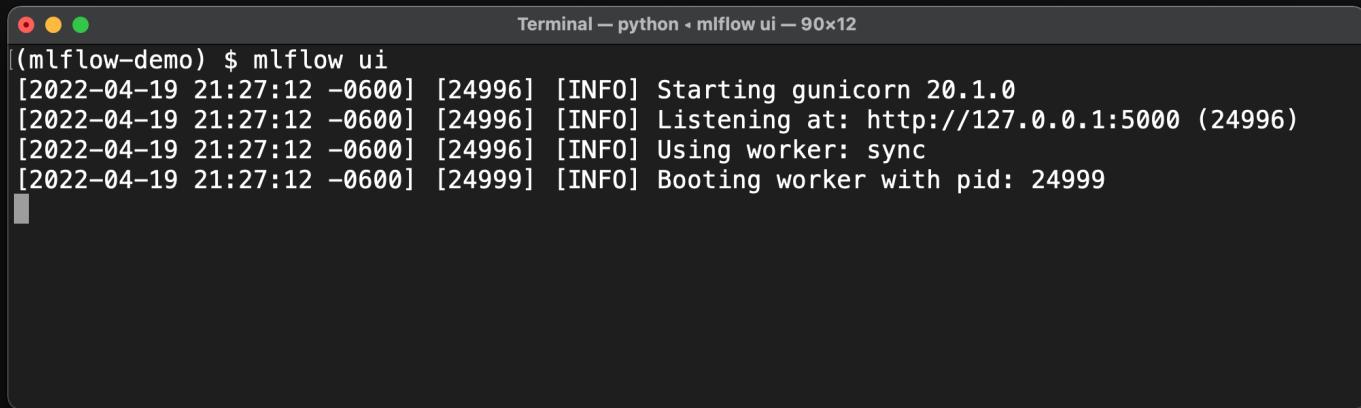
- parameters
- metrics
- arbitrary files ("artifacts" in mlflow)
  - plots, text files, Jupyter notebooks...
- code version
- training data

# First Example

```
In [1]: import mlflow

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

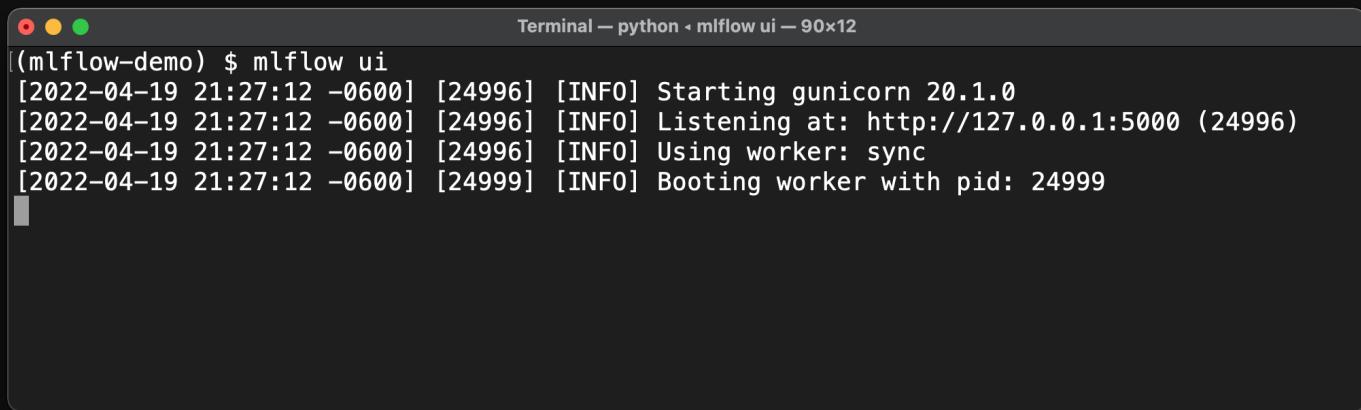
# MLflow Tracking UI



A screenshot of a terminal window titled "Terminal — python - mlflow ui — 90x12". The window contains the following text:

```
(mlflow-demo) $ mlflow ui
[2022-04-19 21:27:12 -0600] [24996] [INFO] Starting gunicorn 20.1.0
[2022-04-19 21:27:12 -0600] [24996] [INFO] Listening at: http://127.0.0.1:5000 (24996)
[2022-04-19 21:27:12 -0600] [24996] [INFO] Using worker: sync
[2022-04-19 21:27:12 -0600] [24999] [INFO] Booting worker with pid: 24999
```

# MLflow Tracking UI

A screenshot of a terminal window titled "Terminal — python - mlflow ui — 90x12". The window shows the command "(mlflow-demo) \$ mlflow ui" followed by several log messages from gunicorn starting up. The log messages include: "[2022-04-19 21:27:12 -0600] [24996] [INFO] Starting gunicorn 20.1.0", "[2022-04-19 21:27:12 -0600] [24996] [INFO] Listening at: http://127.0.0.1:5000 (24996)", "[2022-04-19 21:27:12 -0600] [24996] [INFO] Using worker: sync", and "[2022-04-19 21:27:12 -0600] [24999] [INFO] Booting worker with pid: 24999".

```
(mlflow-demo) $ mlflow ui
[2022-04-19 21:27:12 -0600] [24996] [INFO] Starting gunicorn 20.1.0
[2022-04-19 21:27:12 -0600] [24996] [INFO] Listening at: http://127.0.0.1:5000 (24996)
[2022-04-19 21:27:12 -0600] [24996] [INFO] Using worker: sync
[2022-04-19 21:27:12 -0600] [24999] [INFO] Booting worker with pid: 24999
```

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

## 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 &lt; 1 and params.model = "tree"

Search

Filter

			Metrics	Parameters
	↓ Start Time	Run Name	Models	score
	🕒 1 minute ago	-	-	100
				4

Load more

Default &gt; 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')
```

Experiments + <

Search Experiments

Default 🔗 Delete

## 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
<input type="checkbox"/>	<span style="color: green;">🕒 16 seconds ago</span>	log-artifacts	-	95
<input type="checkbox"/>	<span style="color: green;">🕒 9 minutes ago</span>	-	-	100

Load more

Default &gt; 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](#)

# **... but 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.

# 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'):

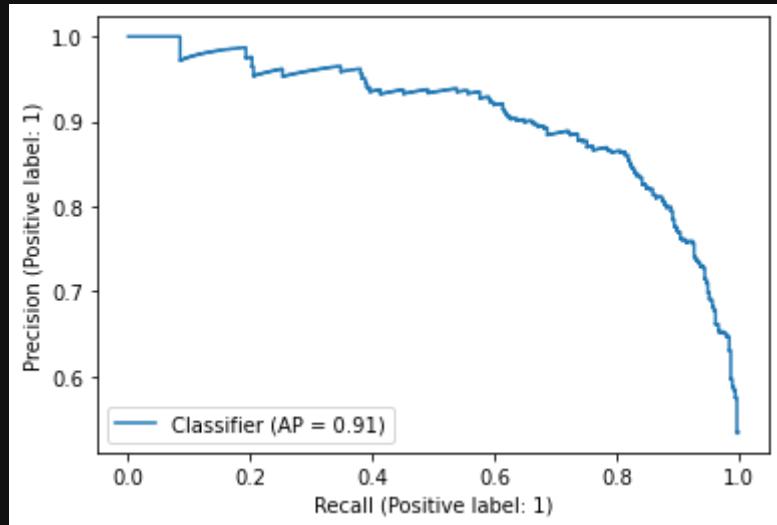
    # fit a model with certain hyperparameters
    penalty = 'l2'
    max_iter = 10
    clf = LogisticRegression(penalty=penalty, max_iter=max_iter)
    clf.fit(X_train, y_train)

    # 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')
```



Default &gt; 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

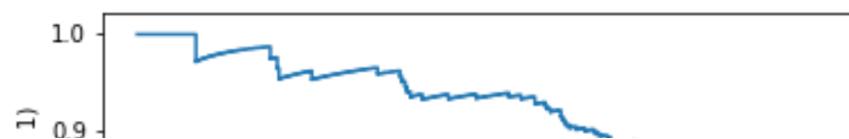
▼ Metrics (2)

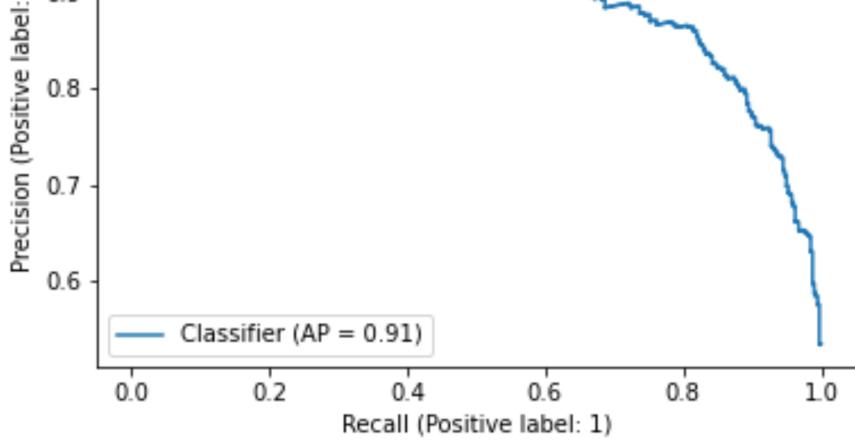
Name	Value
acc	0.887
val_acc	0.833

▶ Tags

▼ Artifacts

▶ model	Full Path:file:///Users/jesford/mlflow-tutorial/mlruns/0/e94097747f0740a5b79ec24a... 
PR.png	Size: 13.02KB
tutorial.ipynb	





▼ Artifacts

Full Path:file:///Users/jesford/mlflow-tutorial/mlruns/0/e94097747f0740a5b79ec24a... [Copy](#)

[Register Model](#)

model

- MLmodel
- conda.yaml
- model.pkl
- requirements.txt
- PR.png
- tutorial.ipynb

## 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

Name	Type
No schema. See <a href="#">MLflow docs</a> for how to include input and output schema with your model.	

**Model schema**

Input and output schema for your model. [Learn more](#)

**Make Predictions**

Predict on a Spark DataFrame:

```
import mlflow
logged_model = 'runs:/e94097747f0740a5b79ec24a7ad660f/f/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:/e94097747f0740a5b79ec24a7ad660f/f/model'

# Load model as a PyFuncModel.
loaded_model = mlflow.pyfunc.load_model(logged_model)
```



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

H

B

I

S

♂

”

↔

☒

≡

≡

≡

# Hockey vs Baseball Classifier

Details about my Logistic Regression model...

Save

Cancel

### ▼ Parameters (2)

Default &gt; 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

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

```
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...**

Default &gt; Run d2049ae6b4074af49867e89e58d9973c

## 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

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

▼ 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/d2049ae6b4074af49867e89e... ↗

Size: 9.07KB

Download

Normalized confusion matrix

True label

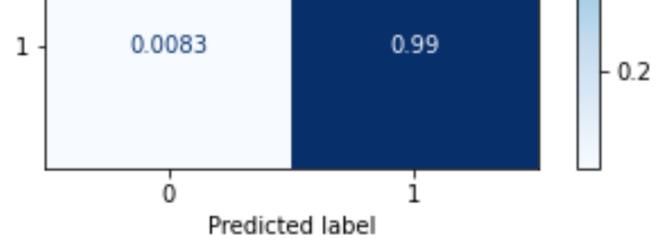
0.99

0.005

0.8

0.6

0.4



# 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{learning_rate}'):
    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\_lo  
ss: 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\_loss: 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\_loss: 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\_loss: 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\_loss: 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\_loss: 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\_loss: 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\_loss: 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\_loss: 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

Default &gt; 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

▼ model

  ▼ data

    ▼ model

      ► assets

      ► variables

      (saved\_model.pb)

      (keras\_module.txt)

      (save\_format.txt)

    MLmodel

    conda.yaml

    requirements.txt

  ► tensorboard\_logs

  (model\_summary.txt)

Full Path:file:///Users/jesford/mlflow-tutorial/mlruns/0/ba87c5548b5a473aab8818dc...

Size: 742B

Model: "model"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 100)]	0
dense (Dense)	(None, 32)	3232
dense_1 (Dense)	(None, 1)	33

Total params: 3,265  
Trainable params: 3,265  
Non-trainable params: 0

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

Default &gt; sports-NN-32d-lr0.1 &gt; Metrics

## Metrics

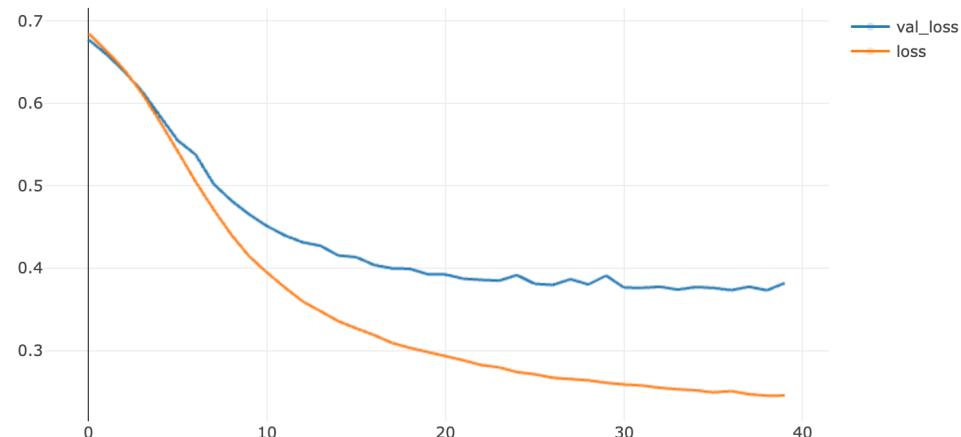
Points:  OffLine Smoothness ?

1

X-axis:

 Step Time (Wall) Time (Relative)

Y-axis:

val\_loss loss Y-axis Log Scale:  Off

# **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

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

	Start Time	Run Name	AUROC	binary_accuracy	loss	val_AUROC	↓ val_binary_accuracy	val_loss	opt_learning_rate
<input type="checkbox"/>	16 minutes ago	sports-NN-32d-lr0.1	0.965	0.894	0.245	0.917	0.838	0.382	0.1
<input type="checkbox"/>	15 minutes ago	sports-NN-32d-lr0.05	0.952	0.879	0.309	0.903	0.828	0.403	0.05
<input type="checkbox"/>	14 minutes ago	sports-NN-8d-lr0.05	0.951	0.88	0.314	0.904	0.824	0.411	0.05
<input type="checkbox"/>	14 minutes ago	sports-NN-8d-lr0.1	0.964	0.894	0.249	0.916	0.819	0.397	0.1
<input type="checkbox"/>	15 minutes ago	sports-NN-8d-lr0.01	0.85	0.78	0.625	0.796	0.724	0.639	0.01
<input type="checkbox"/>	15 minutes ago	sports-NN-32d-lr0.01	0.866	0.766	0.646	0.807	0.712	0.656	0.01

[Load more](#)

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

	Start Time	Run Name	Metrics					Parameters	
			AUROC	binary_accuracy	loss	val_AUROC	↓ val_binary_accuracy	val_loss	opt_learning_rate
<input checked="" type="checkbox"/>	16 minutes ago	sports-NN-32d-lr0.1	0.965	0.894	0.245	0.917	0.838	0.382	0.1
<input checked="" type="checkbox"/>	15 minutes ago	sports-NN-32d-lr0.05	0.952	0.879	0.309	0.903	0.828	0.403	0.05
<input checked="" type="checkbox"/>	14 minutes ago	sports-NN-8d-lr0.05	0.951	0.88	0.314	0.904	0.824	0.411	0.05
<input type="checkbox"/>	14 minutes ago	sports-NN-8d-lr0.1	0.964	0.894	0.249	0.916	0.819	0.397	0.1
<input type="checkbox"/>	15 minutes ago	sports-NN-8d-lr0.01	0.85	0.78	0.625	0.796	0.724	0.639	0.01
<input type="checkbox"/>	15 minutes ago	sports-NN-32d-lr0.01	0.866	0.766	0.646	0.807	0.712	0.656	0.01

Load more

Default &gt; Comparing 3 Runs

## Comparing 3 Runs

Run ID:	<a href="#">ba87c5548b5a473aab8818dcc957...</a>	<a href="#">84a313504e9b45d7895d8c9f77d4...</a>	<a href="#">0e11b08f73b24fe48a5129219744...</a>
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
User	opt_name	SGD	SGD
	opt_nesterov	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

## Metrics

AUROC <a href="#">🔗</a>	0.965	0.952	0.951
binary_accuracy <a href="#">🔗</a>	0.894	0.879	0.88
loss <a href="#">🔗</a>	0.245	0.309	0.314
val_AUROC <a href="#">🔗</a>	0.917	0.903	0.904
val_binary_accuracy <a href="#">🔗</a>	0.838	0.828	0.824
val_loss <a href="#">🔗</a>	0.382	0.403	0.411

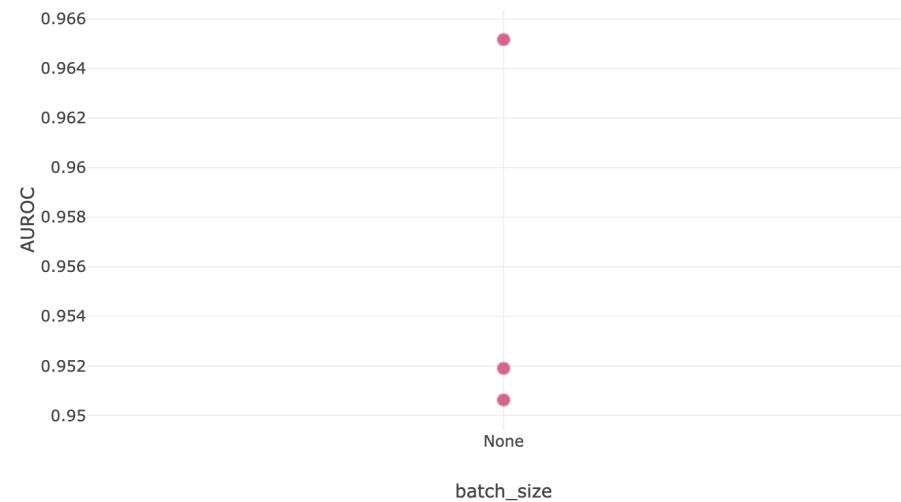
Scatter Plot

Contour Plot

Parallel Coordinates Plot

X-axis:

Y-axis:



Default &gt; Comparing 3 Runs &gt; val\_AUROC

## val\_AUROC

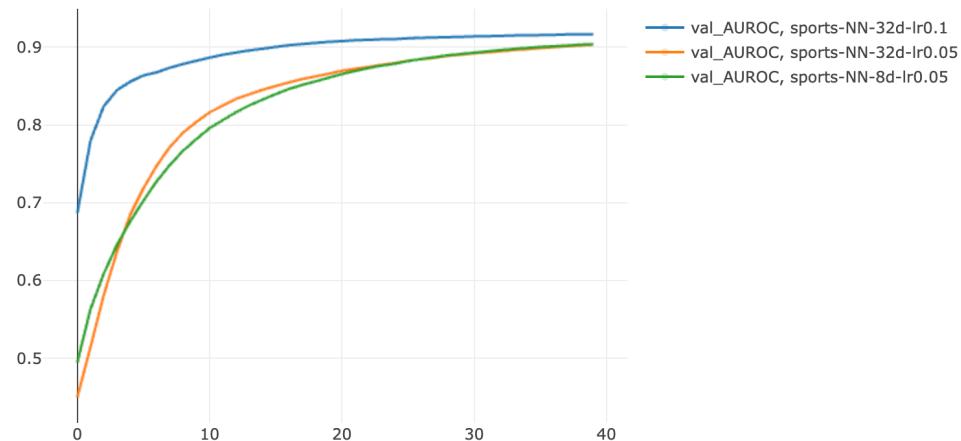
Points: Line Smoothness ?

1

X-axis:

 Step Time (Wall) Time (Relative)

Y-axis:

val\_AUROC Y-axis Log Scale: 

# 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
- 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)
- My team uses MLflow tracking in our training infrastructure to improve our model deployment process & record keeping
- Our review "process" is a WIP and evolving!

# Code

This presentation is a notebook on GitHub: [github.com/jesford/model-review](https://github.com/jesford/model-review)

## Want to work with me?

Cash App is hiring for all kinds of positions: [cash.app/careers](https://cash.app/careers)

## Questions? 🤷‍♀️🤷‍♂️🤷‍♀️🤷‍♂️🤷‍♀️🤷‍♂️🤷‍♀️🤷‍♂️

