Introduction to MLflow Models

INTRODUCTION TO MLFLOW

Weston Bassler
Senior MLOps Engineer

datacamp

MLflow Models

- Simplify ML library integration
- Simplify Deployment
- Convention called "Flavors"

[1] unsplash.com

Built-In Flavors

- Python Function (`python_function`)
- R Function (`crate`)
- H_2O (`h2o`)
- Keras (`keras`)
- MLeap (`mleap`)
- PyTorch (`pytorch`)
- Scikit-learn (`sklearn`)
- Spark MLlib (`spark`)
- TensorFlow (`tensorflow`)
- ONNX (`onnx`)
- MXNet Gluon (`gluon`)
- XGBoost (`xgboost`)

[1] mlflow.org

- Write custom tools from ML libraries
- Flavors simplify the new for custom code

```
# Import flavor from mlflow module
import mlflow.FLAVOR
```

Autolog

```
# Automatically log model and metrics
mlflow.FLAVOR.autolog()
```

```
# Scikit-learn built-in flavor
mlflow.sklearn.autolog()
```

Scikit-learn Flavor

```
# Import scikit-learn
import mlflow
from sklearn.linear_model import \
    LinearRegression

# Using auto-logging
mlflow.sklearn.autolog()
```

```
# Train the model
lr = LinearRegression()
lr.fit(X, y)
```

Model will be logged automatically on `model.fit()`

Common Metrics

- Regression
  - mean squared error
  - root mean squared error
  - mean absolute error
  - r2 score
- Classification
  - precision score
  - recall score
  - f1 score
  - accuracy score

Common Parameters

```
MODEL.get_params()
```

Common parameters

```
# Train the model
lr = LinearRegression()
lr.fit(X, y)
# Get params
params = lr.get_params(deep=True)
params
```

```
{'copy_X': True, 'fit_intercept': True, 'n_jobs': None,
    'normalize': 'deprecated', 'positive': False}
```

Autolog parameters

Parameters (5)

Name	Value
copy_X	True
fit_intercept	True
n_jobs	None
normalize	deprecated
positive	False

```
# Model
lr = LinearRegression()
lr.fit(X, y)
```

Autolog metrics

Metrics (5)

Name	Value
training_mean_absolute_error	6640.5
training_mean_squared_error	89637953.1
training_r2_score	0.952
training_root_mean_squared_error	9467.7
training_score	0.952

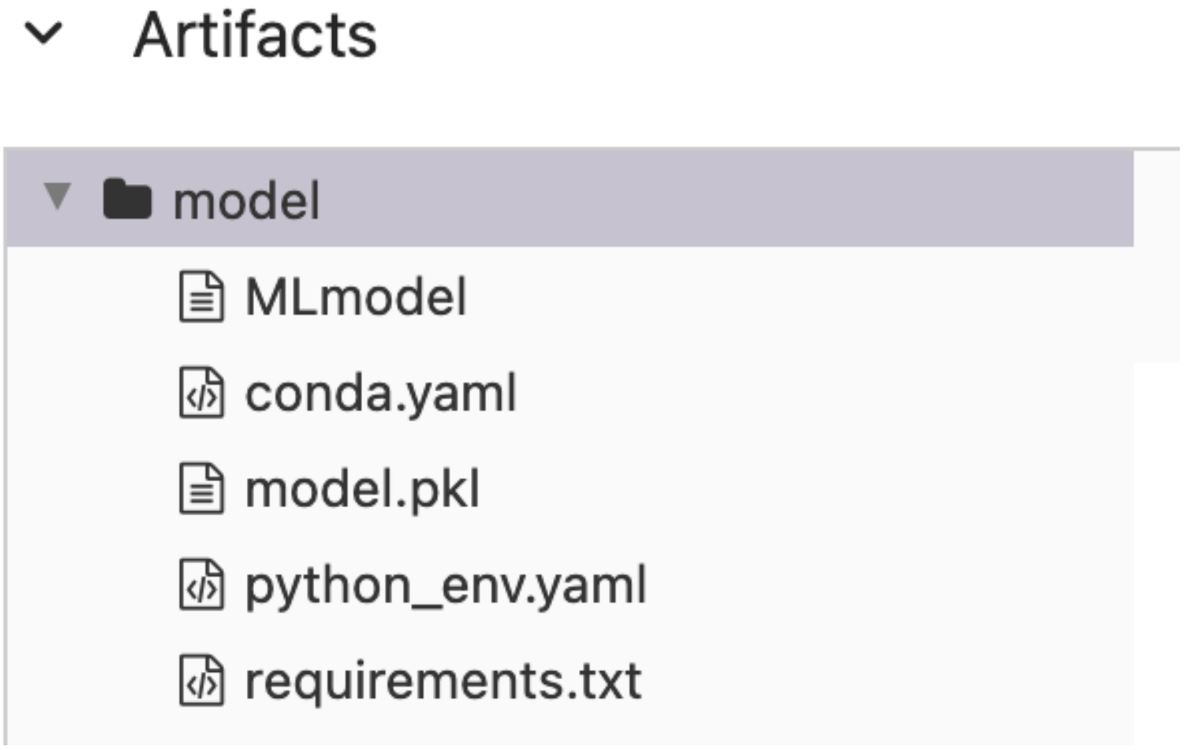
```
# Model
lr = LinearRegression()
lr.fit(X, y)
```

Storage format

Directory structure for a model:

```
model/
    MLmodel
    conda.yaml
    model.pkl
    python_env.yaml
    requirements.txt
```

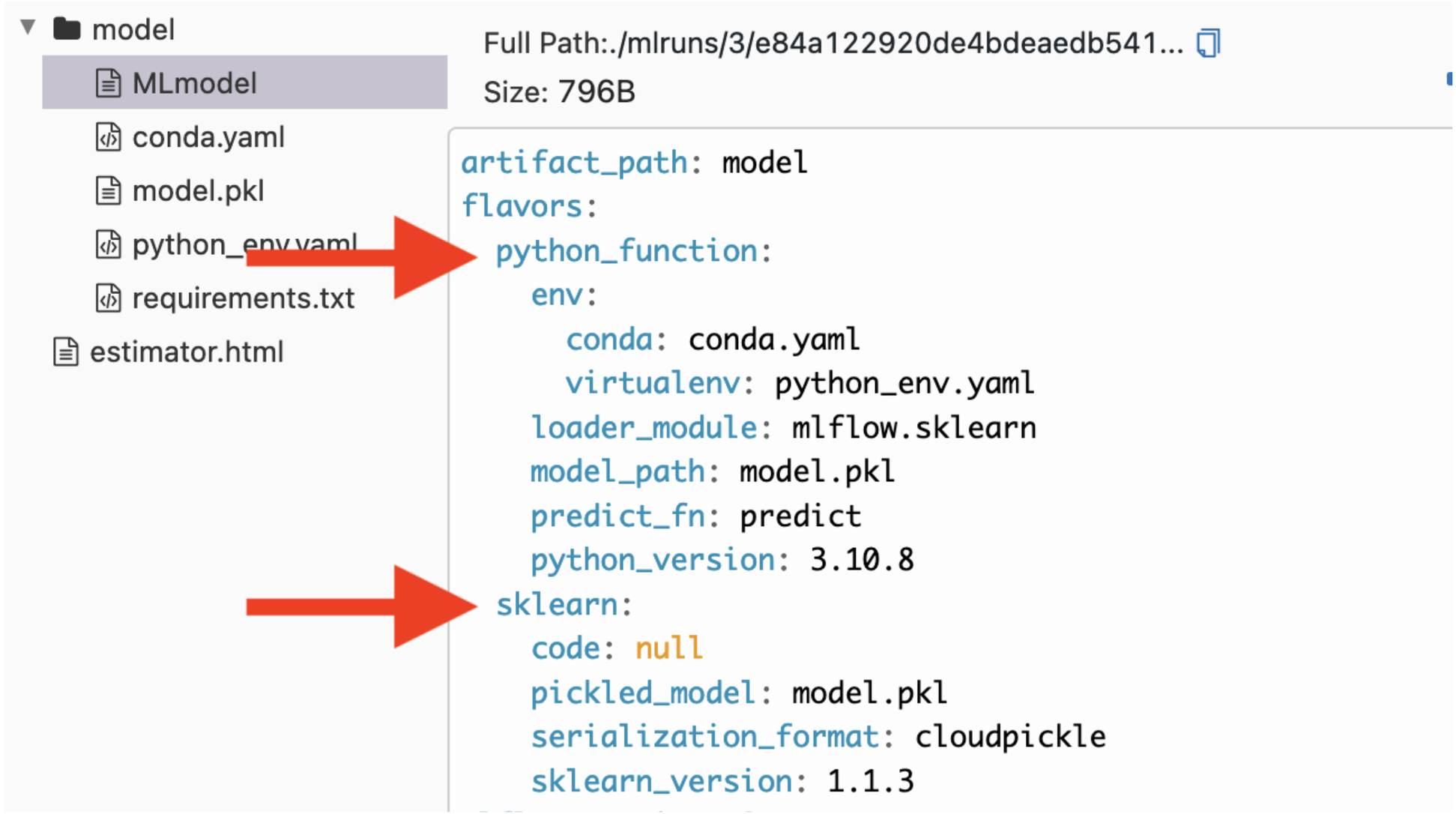
```
# Autolog
mlflow.sklearn.autolog()
```



Contents of MLmodel

```
artifact_path: model
flavors:
  python_function:
    env:
      conda: conda.yaml
      virtualenv: python_env.yaml
    loader_module: mlflow.sklearn
    model_path: model.pkl
    predict_fn: predict
    python_version: 3.10.8
  sklearn:
    code: null
    pickled_model: model.pkl
    serialization_format: cloudpickle
    sklearn_version: 1.1.3
```

MLmodel


model
MLmodel
conda.yaml
model.pkl
python_env.yaml
requirements.txt
estimator.html
Full Path:./mlruns/3/e84a122920de4bdeaedb541...
Size: 796B
artifact_path: model
flavors:
python_function:
env:
conda: conda.yaml
virtualenv: python_env.yaml
loader_module: mlflow.sklearn
model_path: model.pkl
predict_fn: predict
python_version: 3.10.8
sklearn:
code: null
pickled_model: model.pkl
serialization_format: cloudpickle
sklearn_version: 1.1.3

Let's practice!

INTRODUCTION TO MLFLOW

Model API

INTRODUCTION TO MLFLOW

Weston Bassler
Senior MLOps Engineer

MLflow REST API

[1] istock.com

The Model API

- Save
- Log
- Load

[1] wikipedia.org

Model API functions

```
# Save a model to the local filesystem
mlflow.sklearn.save_model(model, path)
```

```
# Log a model as an artifact to MLflow Tracking.
mlflow.sklearn.log_model(model, artifact_path)
```

```
# Load a model from local filesystem or from MLflow Tracking.
mlflow.sklearn.load_model(model_uri)
```

Load model

- Local Filesystem - `relative/path/to/local/model` or `/Users/me/path/to/local/model`
- MLflow Tracking - `runs:/<mlflow_run_id>/run-relative/path/to/model`
- S3 Support - `s3://my_bucket/path/to/model`

Save model

```
# Model
lr = LogisticRegression()
lr.fit(X, y)

# Save model locall
mlflow.sklearn.save_model(lr, "local_path")
```

```
ls local_path/
```

```
MLmodel              model.pkl         requirements.txt
conda.yaml          python_env.yaml
```

Load local model

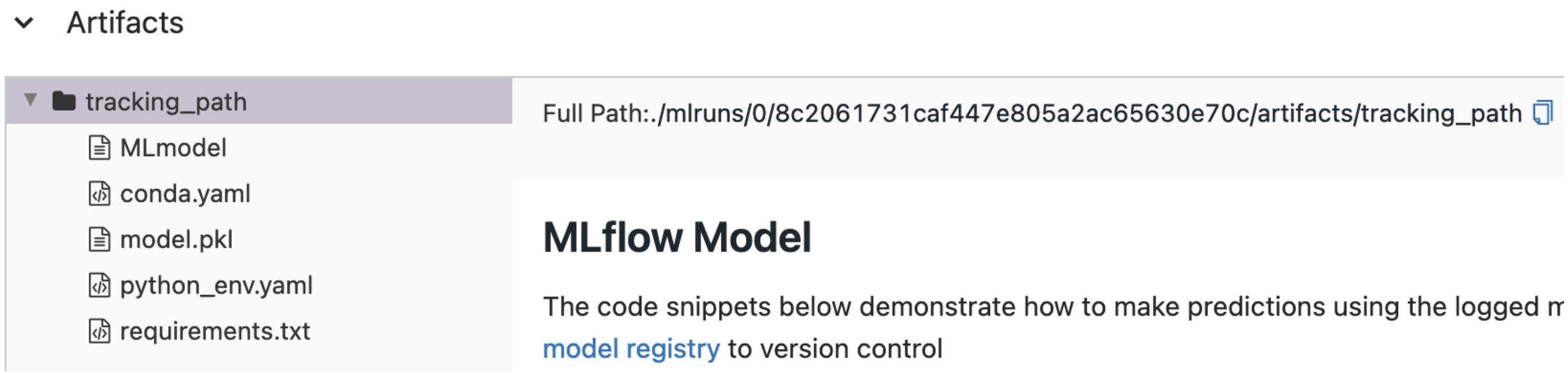
```
# Load model from local path
model = mlflow.sklearn.load_model("local_path")


# Show model
model
```

```
LogisticRegression()
```

Log model

```
# Model
lr = LogisticRegression(n_jobs=n_jobs)
lr.fit(X, y)

# Log model
mlflow.sklearn.log_model(lr, "tracking_path")
```

Tracking UI


Artifacts
tracking_path
MLmodel
conda.yaml
model.pkl
python_env.yaml
requirements.txt
Full Path:./mlruns/0/8c2061731caf447e805a2ac65630e70c/artifacts/tracking_path
MLflow Model
The code snippets below demonstrate how to make predictions using the logged m
model registry to version control

Last active run

```
# Format for runs
runs:/<mlflow_run_id>/run-relative/path/to/model
```

```
# Get last active run
run = mlflow.last_active_run()
run
```

```
<Run: data=<RunData: metrics={}, params={},
tags={'mlflow.runName': 'run_name'}>,
 info=<RunInfo: artifact_uri='uri', end_time='end_time',
 experiment_id='0', lifecycle_stage='active', run_id='run_id',
 run_name='name', run_uuid='run_uuid', start_time=start_time,
 status='FINISHED', user_id='user_id'>>
```

Last active run id

```
# Get last active run
run = mlflow.last_active_run()
```

```
# Show run_id of last run
run.info.run_id
```

```
'8c2061731caf447e805a2ac65630e70c'
```

Setting the run id

```
# Get last active run
run = mlflow.last_active_run()
# Set run_id variable
run_id = run.info.run_id
run_id
```

```
'8c2061731caf447e805a2ac65630e70c'
```

Load model from MLflow Tracking

```
# Pass run_id as f-string literal
model = mlflow.sklearn.load_model(f"runs:/{run_id}/tracking_path")
# Show model
model
```

```
LogisticRegression()
```

Let's Practice

INTRODUCTION TO MLFLOW

Custom models

INTRODUCTION TO MLFLOW

Weston Bassler
Senior MLOps Engineer

datacamp

Example use cases

- NLP - Tokenizer(s)
- Classification - Label encoder
- Pre/Post processing
- Not a built-in flavor

[1] unsplash.com

Custom Python models

- Built in Flavor - `python_function`
- `mlflow.pyfunc`
  - `save_model()`
  - `log_model()`
  - `load_model()`

Custom model class

- Custom model class
  - `MyClass(mlflow.pyfunc.PythonModel)`
- PythonModel class
  - `load_context()` - loads artifacts when `mlflow.pyfunc.load_model()` is called
  - `predict()` - takes model input and performs user defined evaluation

Python class

```
# Class
class MyPythonClass:
    # Function that prints Hello!
    def my_func():
        print("Hello!")
```

```
# Create a new Object
x = MyPythonClass()
# Excute my_func function
x.my_func
```

```
"Hello!"
```

Example custom Class

```
import mlflow.pyfunc


# Define the model class
class CustomPredict(mlflow.pyfunc.PythonModel):
    # Load artifacts
    def load_context(self, context):
        self.model = mlflow.sklearn.load_model(context.artifacts["custom_model"])
    # Evaluate input using custom_function()
    def predict(self, context, model_input):
        prediction = self.model.predict(model_input)
        return custom_function(prediction)
```

Saving and logging a custom model

```
# Save model to local filesystem
mlflow.pyfunc.save_model(path="custom_model", python_model=CustomPredict())
```

```
# Log model to MLflow Tracking
mlflow.pyfunc.log_model(artifact_path="custom_model", python_model=CustomPredict())
```

Loading custom models

```
# Load model from local filesystem
mlflow.pyfunc.load_model("local")
```

```
# Load model from MLflow Tracking
mlflow.pyfunc.load_model("runs:/run_id/tracking_path")
```

Model Evaluation

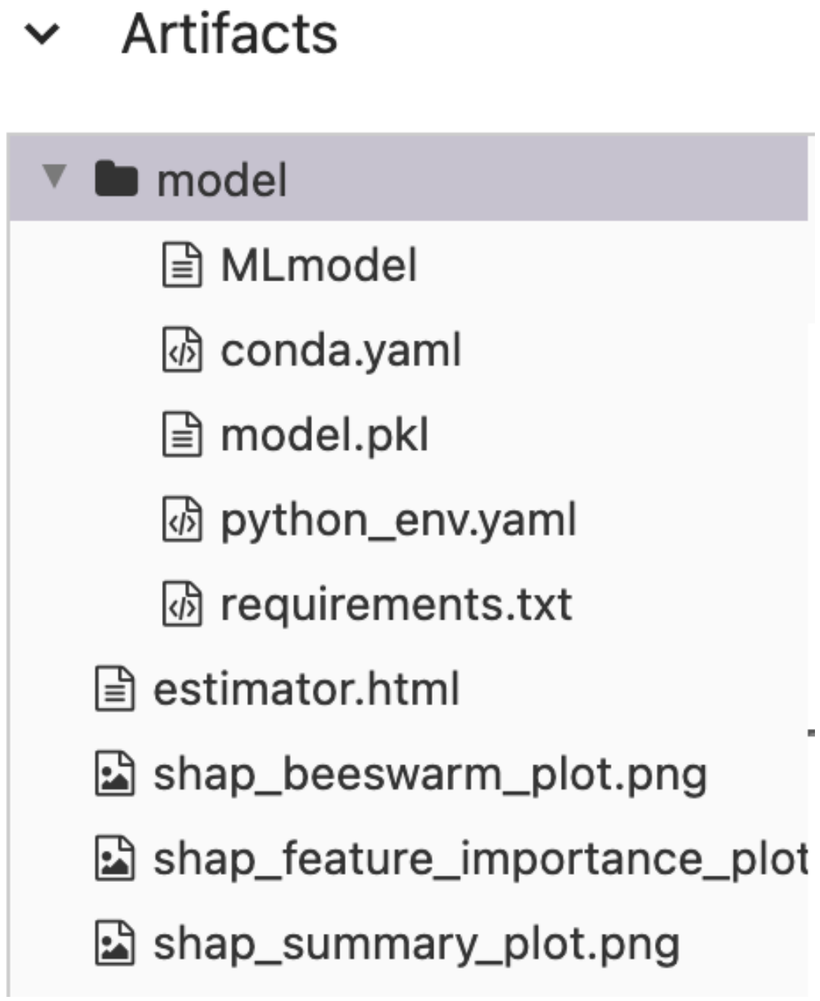
- `mlflow.evaluate()` - Performance based on a dataset
- Regression and Classification models

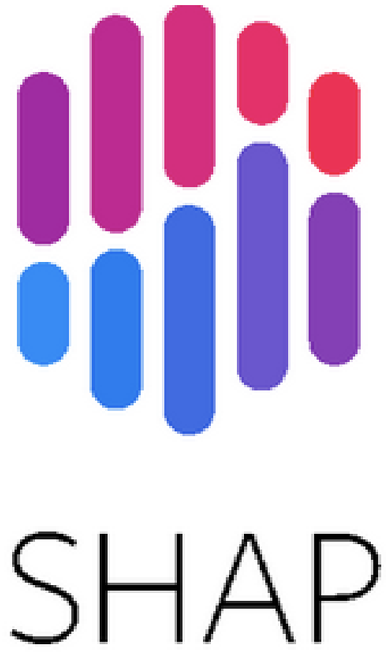
Evaluation Example

```
# Training Data
X_train, X_test, y_train, y_test = \
    train_test_split(X, y,
    train_size=0.7,random_state=0)

# Linear Regression model
lr = LinearRegression()
lr.fit(X_train, y_train)
```

```
# Dataset
eval_data = X_test
eval_data["test_label"] = y_test

# Evaluate model with Dataset
mlflow.evaluate(
    "runs:/run_id/model",
    eval_data,
    targets="test_label",
    model_type="regressor"
)
```

Tracking UI





[1] shap.readthedocs.io

Let's practice!

INTRODUCTION TO MLFLOW

Model serving

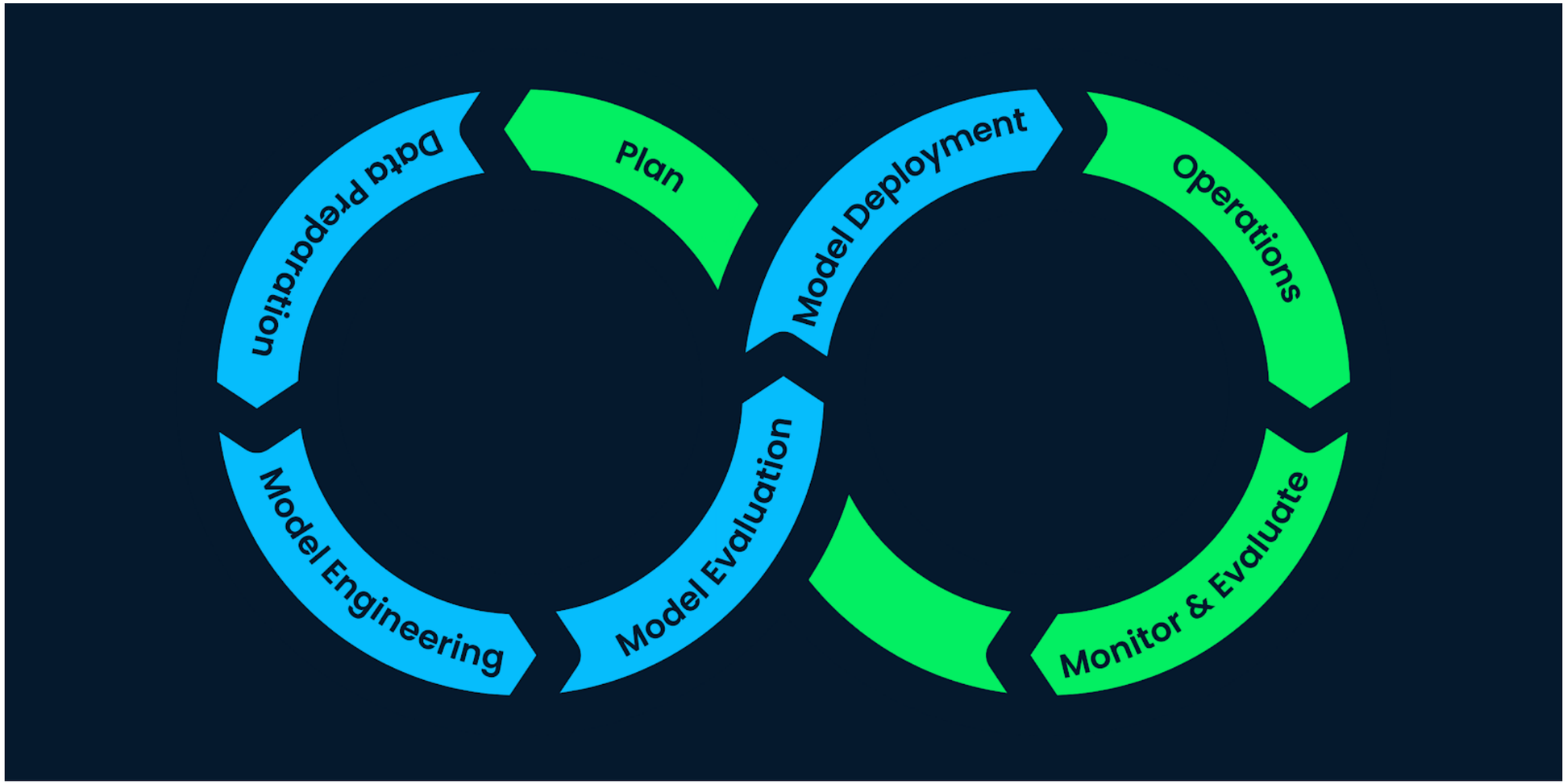
INTRODUCTION TO MLFLOW

Weston Bassler
Senior MLOps Engineer

MLflow Models

- Standardize model packaging
- Log models
- Model Evaluation

Model Deployment



[1] datacamp.com

REST API

- `/ping` - for health checks
- `/health` - for health checks
- `/version` - for getting the version of MLflow
- `/invocations` - for model scoring
- Port 5000

Invocations endpoint

`/invocations`

```
No,Name,Subject
1,Bill Johnson,English
2,Gary Valentine,Mathematics
```

`Content-Type` : `application/json` or `application/csv`

```
{
    "1": {
        "No": "1",
        "Name": "Bill Johnson",
        "Subject": "English"
    },
    "2": {
        "No": "2",
        "Name": "Gary Valentine",
        "Subject": "Mathematics"
    }
}
```

CSV format

- Pandas Dataframe
- `pandas_df.to_csv()`

JSON format

- `dataframe_split` - pandas DataFrame in split orientation
- `dataframe_records` - pandas DataFrame in records orientation

DataFrame split

```
# Dataframe split orientation
{
  "dataframe_split": {
      "columns": ["sex", "age", "weight"],
      "data": [["male", 23, 160], ["female", 33, 120]]
  }
}
```

Serving Models

```
# MLflow serve command
mlflow models serve --help
Usage: mlflow models serve [OPTIONS]
```

Serve uri

```
# Local Filesystem
mlflow models serve -m relative/path/to/local/model
```

```
# Run ID
mlflow models serve -m runs:/<mlflow_run_id>/artifacts/model
```

```
# AWS S3
mlflow models serve -m s3://my_bucket/path/to/model
```

Serve example

```
# Serve model from run
mlflow models serve -m runs:/e84a122920de4bdeaedb54146deeb429/artifacts/model
```

```
2023/03/12 16:28:28 INFO mlflow.models.flavor_backend_registry:
Selected backend for flavor 'python_function'
2023/03/12 16:28:28 INFO mlflow.pyfunc.backend: === Running command
'exec gunicorn --timeout=60 -b 127.0.0.1:5000 -w 1 ${GUNICORN_CMD_ARGS} --
mlflow.pyfunc.scoring_server.wsgi:app'
[2023-03-12 16:28:29 -0400] [48431] [INFO] Starting gunicorn 20.1.0
[2023-03-12 16:28:29 -0400] [48431] [INFO] Listening at: http://127.0.0.1:5000
(48431)
[2023-03-12 16:28:29 -0400] [48431] [INFO] Using worker: sync
[2023-03-12 16:28:29 -0400] [48432] [INFO] Booting worker with pid: 48432
```

Invocations Request

```
# Send dataframe_split orientation payload to MLflow
curl http://127.0.0.1:5000/invocations -H 'Content-Type: application/json' -d '{
  "dataframe_split": {
      "columns": ["sex", "age", "weight"],
      "data": [["male", 23, 160], ["female", 33, 120]]
  }
}'
```

```
{"predictions": [1, 0]}
```

Let's practice!

INTRODUCTION TO MLFLOW