

mlflow

Platform for Machine Learning Lifecycle

Jules S. Damji

@2twitme

Outline – Introduction to MLflow: How to Use MLflow Models – Module 3

- MLflow Models
 - Concepts and Motivations
 - Tour of the the MLflow Model API Documentation
 - How to create different model flavor
 - How to create Customized Models
 - Pyfunc Model Flavor: what and how to use it
 - Explore MLflow UI
 - Tutorials & Exercises
- Q & A

<https://github.com/dmatrix/olt-mlflow>

MLflow Components

mlflow Tracking

Record and query experiments: code, data, config, and results

mlflow Projects

Package data science code in a format that enables reproducible runs on any platform

mlflow Models

Deploy machine learning models in diverse serving environments environments

new

mlflow Model Registry

Store, annotate and manage models in a central repository

[databricks.com
/mlflow](https://databricks.com/mlflow)



mlflow.org



github.com/mlflow



twitter.com/MLflow

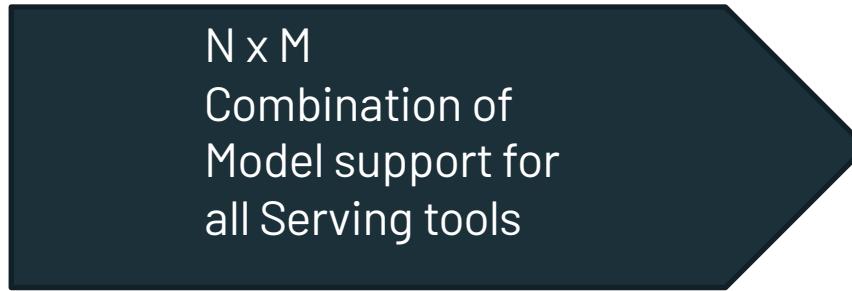
O'REILLY®

MLflow Model Motivations



dmlc
XGBoost

ML Frameworks



N x M
Combination of
Model support for
all Serving tools



Inference Code



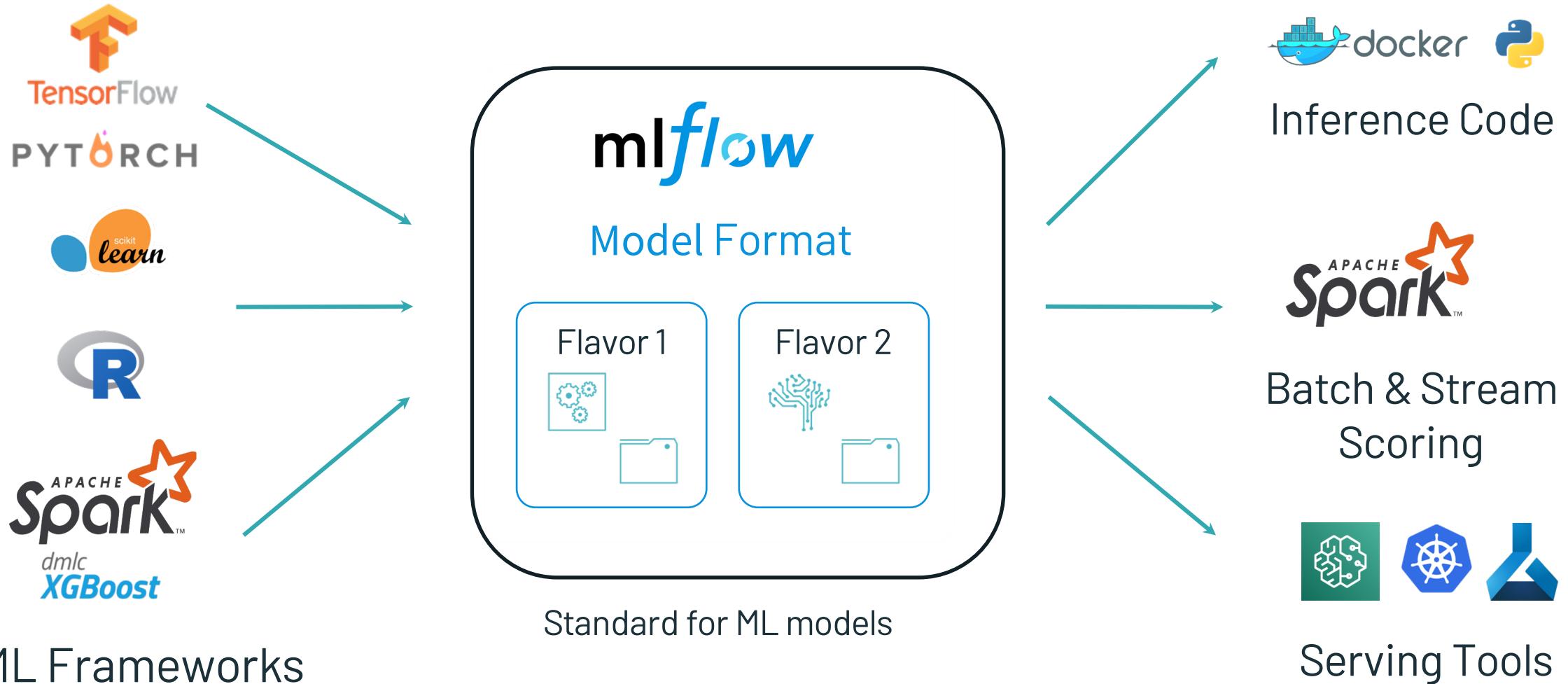
Batch & Stream Scoring



Serving Tools

O'REILLY®

MLflow Models



MLflow Model API Documentation

Model API

model flavor

model flavor generic methods

You can save and load MLflow Models in multiple ways. First, MLflow includes integrations with several common libraries. For example, `mlflow.sklearn` contains `save_model`, `log_model`, and `load_model` functions for scikit-learn models. Second, you can use the `mlflow.models.Model` class to create and write models. This class has four key functions:

- `add_flavor` to add a flavor to the model. Each flavor has a string name and a dictionary of key-value attributes, where the values can be any object that can be serialized to YAML.
- `save` to save the model to a local directory.
- `log` to log the model as an artifact in the current run using MLflow Tracking.
- `load` to load a model from a local directory or from an artifact in a previous run.

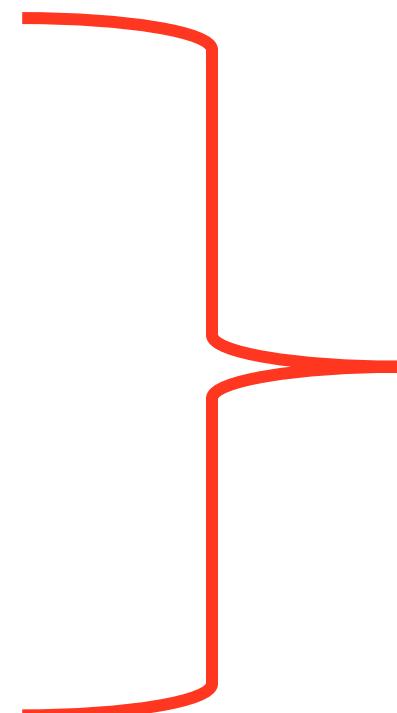
<https://mlflow.org/docs/latest/models.html#model-api>

MLflow Model API Documentation

Built-In Model Flavors

MLflow provides several standard flavors that might be useful in your applications. Specifically, many of its deployment tools support these flavors, so you can export your own model in one of these flavors to benefit from all these tools:

- [Python Function \(python_function\)](#)
- [R Function \(crate\)](#)
- [H2O \(h2o\)](#)
- [Keras \(keras\)](#)
- [MLeap \(mleap\)](#)
- [PyTorch \(pytorch\)](#)
- [Scikit-learn \(sklearn\)](#)
- [Spark MLlib \(spark\)](#)
- [TensorFlow \(tensorflow\)](#)
- [ONNX \(onnx\)](#)
- [MXNet Gluon \(gluon\)](#)
- [XGBoost \(xgboost\)](#)
- [LightGBM \(lightgbm\)](#)
- [Spacy \(spaCy\)](#)
- [Fastai \(fastai\)](#)



Model flavor has generic methods:

- `log_model()`
- `save_model()`
- `load_model()`
- `add_model_flavor()`
- `auto_log() *`

* Only some model flavors

1. Example MLflow Model

`mlflow.keras.log_model(...)`

```
model
  └── MLmodel
  └── conda.yaml
  └── data
    └── keras_module.txt
    └── model.h5
```

1 directory, 4 files

```
artifact_path: model
flavors:
  keras:
    data: data
    keras_module: keras
    keras_version: 2.3.1
    python_function:
      data: data
      env: conda.yaml
      loader_module: mlflow.keras
      python_version: 3.7.5
...
run_id: 714d418027bc43bd8adc768f0f9ecb51
utc_time_created: '2020-08-25 19:04:53.887587'
```

Usable by tools that understand
Keras model format

Usable by any tool that can run
Python (Docker, Spark, etc!)

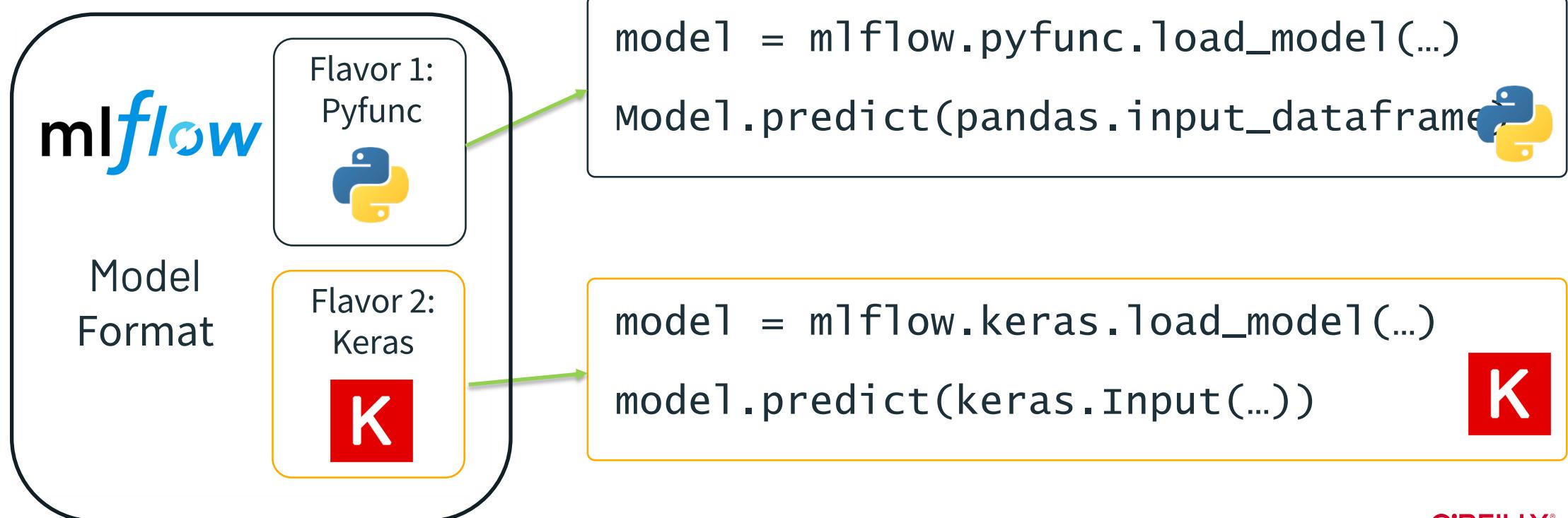
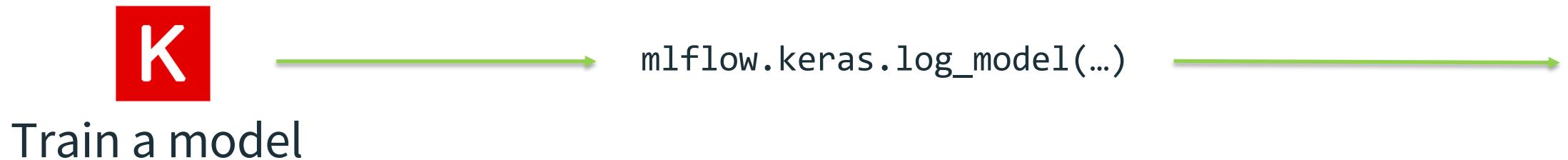
2. Example Conda.yaml

```
model
└── MLmodel
└── conda.yaml
└── data
    └── keras_module.txt
    └── model.h5
```

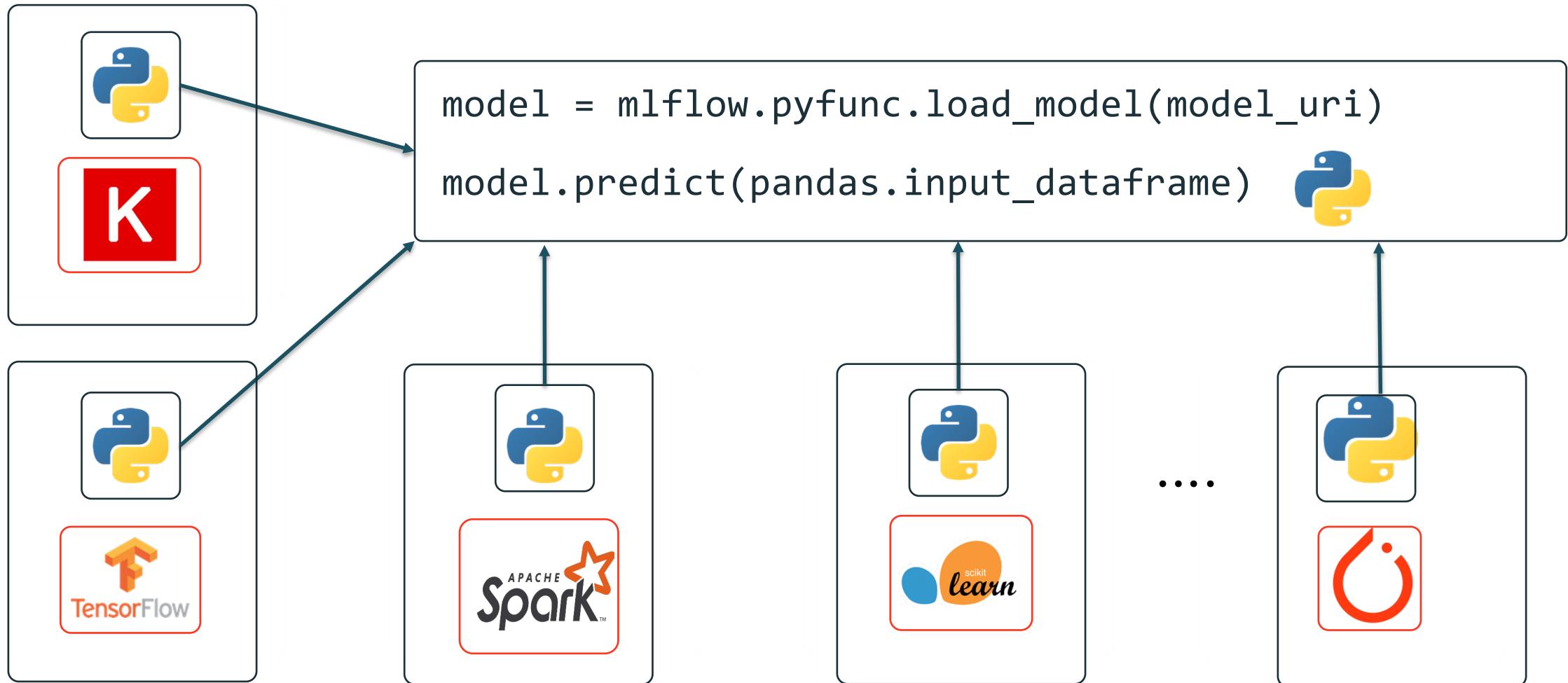
1 directory, 4 files

```
channels:
- defaults
- conda-forge
dependencies:
- python=3.7.5
- pip
- pip:
  - mlflow
  - keras==2.3.1
  - tensorflow==2.0.0
name: mlflow-env
```

Model Keras Flavor Example

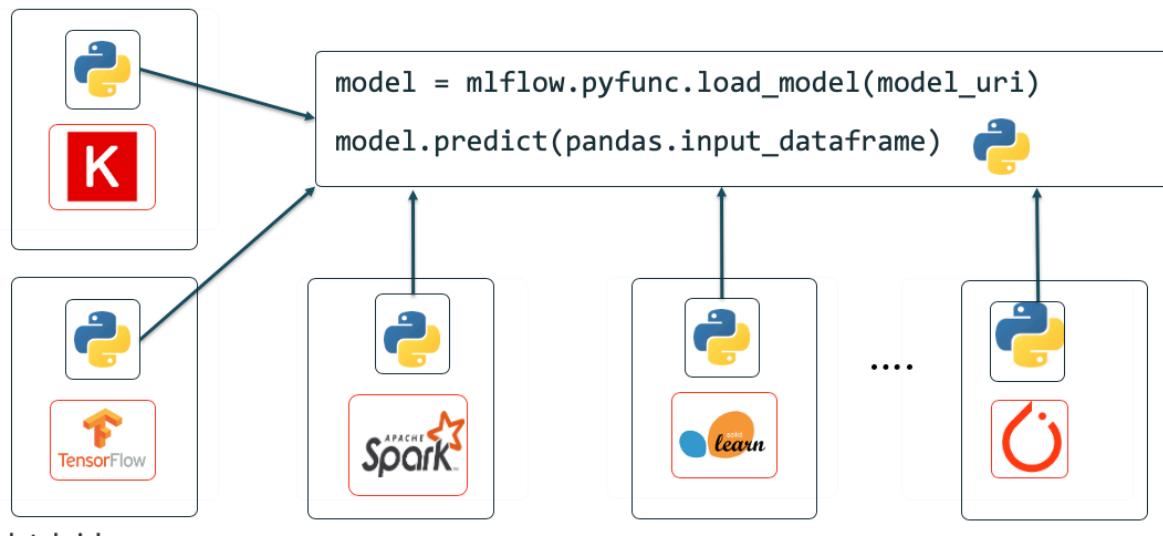


Model Flavors Example



MLflow PythonModel

- Serves as Generic Python model interface



- Use with Spark UDFs

```
predict = mlflow.pyfunc.spark_udf(spark, "runs://run_id/model")  
df.withColumn("prediction", predict("name", "age")).show()
```

MLflow PythonModel

- Use ML framework not supported
 - For example, HuggingFace, SparkNLP, VaderSentiment etc

```
16  class SocialMediaAnalyserModel(mlflow.pyfunc.PythonModel):  
17  
18      def __init__(self):  
19          self._analyser = SentimentIntensityAnalyzer()  
20  
21      def _score(self, text):  
22          scores = self._analyser.polarity_scores(text)  
23          return scores  
24  
25  ↗  def predict(self, context, model_input):  
26      model_output = model_input.apply(lambda col: self._score(col))  
27      return model_output
```

MLflow PyFuncModel

```
30 ► if __name__ == "__main__":
31     model_path = "vader"
32     vader_model = SocialMediaAnalyserModel()
33     with mlflow.start_run(run_name="Vader Sentiment Analysis") as run:
34         mlflow.pyfunc.log_model(model_path, python_model=vader_model)
35
36     # load back the model
37     model_uri = f"runs:{run.info.run_uuid}/{model_path}"
38     loaded_model = mlflow.pyfunc.load_model(model_uri)
39     mlflow.log_param("algorithm", "VADER")
40     mlflow.log_param("total_sentiments", len(INPUT_TEXTS))
41
42     # Use inference to predict output from the model
43     for i in range(len(INPUT_TEXTS)):
44         text = INPUT_TEXTS[i]['text']
45         mlflow.log_param(f"text_{i}", text)
46         model_input = pd.DataFrame([text])
47         scores = loaded_model.predict(model_input)
48         print(f"<{text}> ----- {str(scores[0])}>")
49         for index, value in scores.items():
50             [mlflow.log_metric(f"{key}_{i}", value) for key, value in value.items()]
51
```

MLflow PythonModel

- Deploy with MLflow built-in tools
 - <https://mlflow.org/docs/latest/models.html#built-in-deployment-tools>



```
mlflow azureml build-image -w <workspace-name> -m <model-path>  
-d "Wine regression model v.1"
```

```
az ml service create aci -n <deployment-name> --image-id  
<image-name>:<image-version>
```

```
mlflow sagemaker --help  
mlflow sagemaker build-and-push-container --help mlflow  
sagemaker run-local --help  
mlflow sagemaker deploy --help
```

```
pyfunc_udf = mlflow.pyfunc.spark_udf(<path-to-model>)  
df = spark_df.withColumn("prediction",  
pyfunc_udf(<features>))
```

Recap: MLflow Models

Packaging format for ML Models

- Any directory with MLmodel file

Defines dependencies for reproducibility

- Conda environment can be specified in MLmodel configuration

Model creation and loading utilities

- mlflow.<model_flavor>.save_model(...) or log_model(...) or autolog()
- mlflow.<model_flavor>.load_model(...)

Deployment APIs

- CLI / Python / R / Java
- **mlflow models [OPTIONS] COMMAND [ARGS]...**
 - mlflow models serve [OPTIONS [ARGS]]
 - mlflow models predict [OPTIONS [ARGS] ...]
 - mlflow sagemaker -help
 - mlflow azureml --help

MLflow Models Tutorials

Tutorials: <https://github.com/dmatrix/olt-mlflow>

Thank you! ☺

Q & A

jules@databricks.com
@2twitme

<https://www.linkedin.com/in/dmatrix/>