KARPOV.COURSES >>>
КОНСПЕКТ

> Конспект > 3 урок > Учет и трекинг моделей с ML Flow

> Оглавление

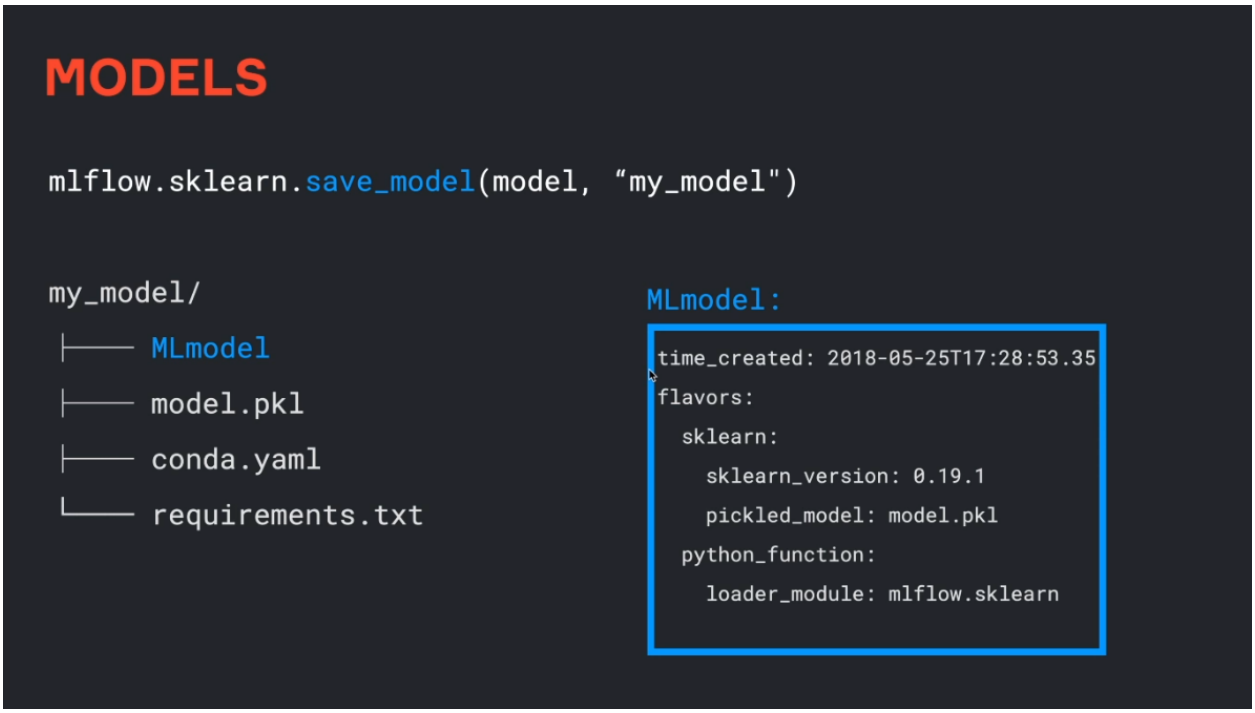
> Знакомство с ML Flow

ML Flow - это платформа с открытым исходным кодом для управления жизненным циклом машинного обучения. На каринке представлено четыре основных модуля, которые мы с вами разберем подробнее.

> Models

ML Flow Models - это стандартный формат для упаковки моделей машинного обучения. Формат определяет соглашения, которое позволяет вам сохранять модель в различных "вариантах", в дальнейшем они могут быть поняты различными последующими инструментами.

Разберем, как устроен формат хранения моделей. Представим, что есть некоторая модель на базе фреймворка `.sklearn`, в таком случае, чтобы разместить ее в ML FLow мы вызовем пакет `.sklearn` и вызовем в нем функцию `save_model`, куда укажем модель и ее имя.



ML FLow создаст директорию `my_model/` со следующими файлами:

- MLmodel (информация о создании модели, каким фреймворком, куда сохранена и python функция)
- model.pkl
- conda.yaml (которая будет описывать конфигурацию окружения, на которой она была создана)
- requirements.txt (здесь собраны фреймворки, которые нужно использовать, чтобы применять модель).

Это стандарт, поэтому такое размещение будет характерно для любого другого фреймворка, используемого совместно с ML Flow.

> Project

ML Flow Project - это соглашение об организации и описании вашего кода, позволяющее другим специалистам по данным (либо его автоматизированным инструментам) запускать его.

Каждый проект - это каталог файлов или репозиториев Git, содержащий ваш код. ML Flow может запускать некоторые проекты на основе соглашения о размещении

файлов в этом каталоге.

> Tracking

ML Flow Tracking - это API и пользовательский интерфейс для регистрации параметров, версий кода, показателей и моделей при запуске кода машинного обучения (экспериментов).

> Registry

ML Flow Model Registry - это централизованное хранилище моделей, набор API-интерфейсов и пользовательского интерфейса для совместного управления жизненным циклом модели ML Flow.

Поддерживаемые ML Flow фреймворки.

```
mlflow.azureml        mlflow.sagemaker
mlflow.catboost       mlflow.pyspark
mlflow.fastai         mlflow.sklearn
mlflow.h2o            mlflow.tensorflow
mlflow.keras          mlflow.xgboost
mlflow.lightgbm       mlflow.onnx
mlflow.pytorch
```

> Управление экспериментом

Для этого нам необходимо провести следующие операции:

1. импортировать пакет `mlflow`

2. законфигурировать `tracking_uri`, для того, чтобы задача по обучению нашей модели могла связаться с нашим сервисом

3. конфигурировать имя эксперимента

```
import mlflow
mlflow.set_tracking_uri("https://<mlflow-host>")
mlflow.set_experiment(experiment_name = “something-model”)

mlflow.start_run()
# Обучение модели
mlflow.end_run()
```

4. запустить функцию `.start_run()`

5. запустить обучение модели

6. завершение эксперимента функцией `.end_run()`

> Трекинг эксперимента

Логирование эксперимента происходит с помощью нескольких функций:

`.log_param`, в которой мы должны передать имя и значение параметра;

`.log_metric`, она позволяет залогировать метрику, необходимо указать имя метрики и ее значение;

`.log_model` - позволяет залогировать модель, полученную в ходе эксперимента. Так как мы модель получаем на базе определенного фреймворка, нам нужно обратиться к этому пакету, по которому модель была обучена. В нашем случае мы обучили `pipelineModel`, поэтому в ML Flow мы обращаемся к модулю spark и далее у него вызываем функцию `.log_model`, куда передаем объект нашей модели, `artifact_path` и `registered_model_name`.

При этом `registered_model_name` - это уникальное имя всех моделей, полученное в ходе определенного эксперимента, и все последующие запуски с разными параметрами и метриками будем сохраняться по тому же пути, и ML Flow будет группировать их по имени и выдавать им новые версии. Это позволит сравнивать их между собой.

`.log_artifact`, логирование сопутствующей информации или кода, используемая при обучении модели или проведения эксперимента.

> Применение и сервировка моделей

После проведения эксперимента и регистрации моделей, вы можете использовать эти модели. Их можно загружать и применять к вашим данным.

Функция `.load_model` позволяет загрузить модель, указав sap пакет используемой технологии.

Передавая уникальное имя модели, вы получите последнюю версию модели. Однако указав необходимую вам версию, вы можете загрузить именно её.

```
model = mlflow.spark.load_model("spark-model")
prediction = model.transform(test)


from pyspark.sql.functions import struct
predict_udf = mlflow.pyfunc.spark_udf(spark, “other-model”)
df.withColumn("prediction", predict_udf(struct(“name", “age"))).show()
```

Подход, представленный в ML Flow позволяет вам абстрагироваться от технологии, на которой у вас реализована модель и применять любую, удобную вам в продакшене.

> Model Serving

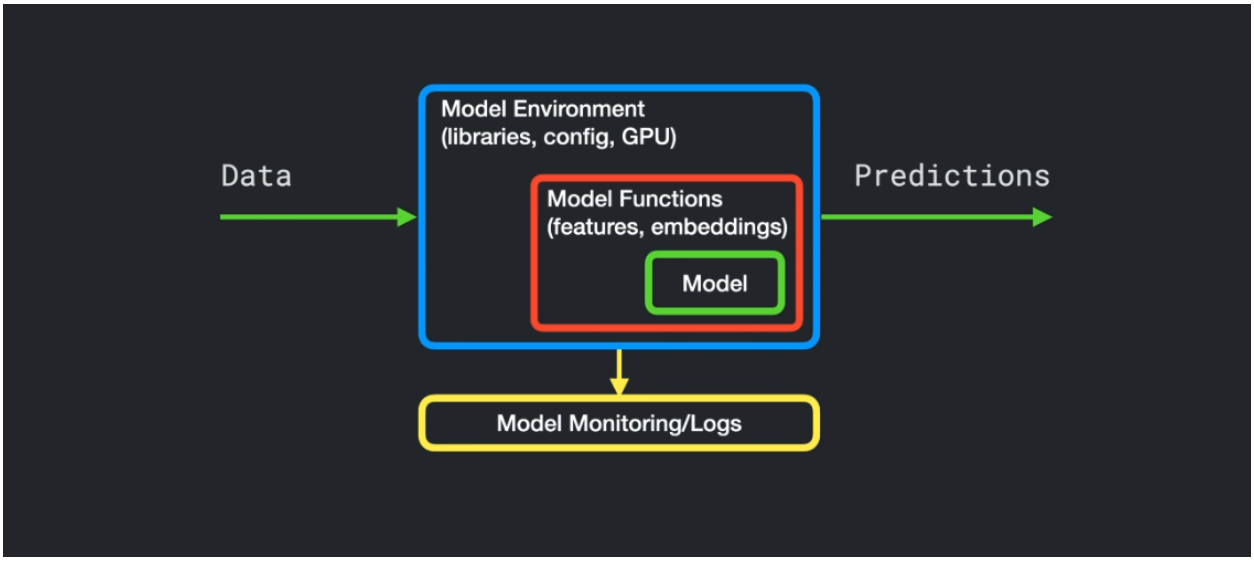
Model Serving (сервировка моделей) - процесс, при котором обученная модель становится доступной из вне для запросов на исполнение.

Порядок операций для Model Serving:

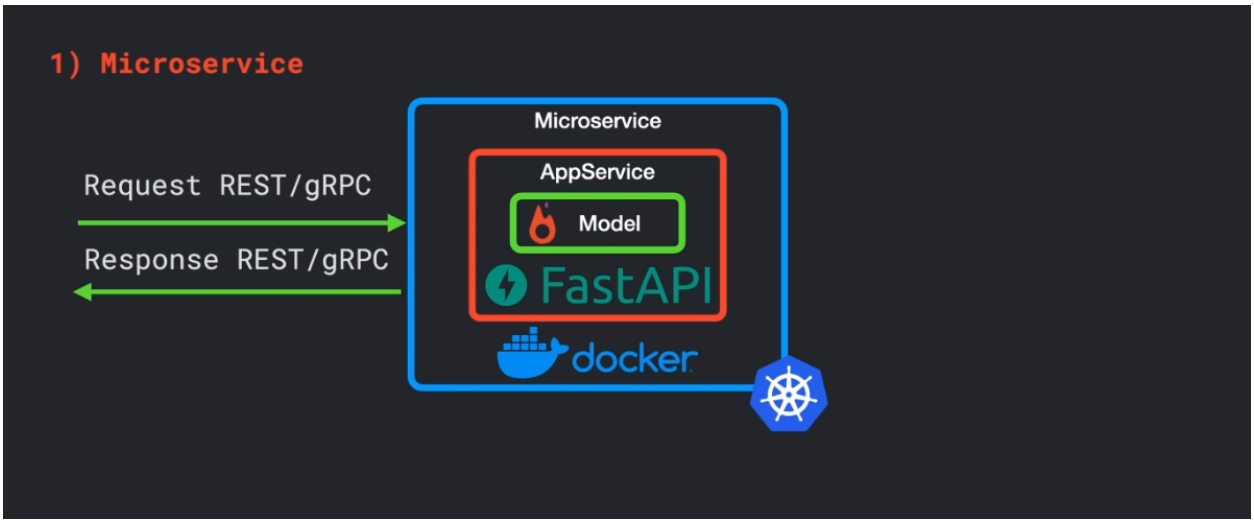
```
> mlflow models serve -m runs:/<RUN_ID>/model --port 1234

> curl --request POST http://<host:1234>/invocations
-H 'Content-Type: application/json' -d '{
    "columns": ["a", "b", "c"],
    "data": [[1, 2, 3]]
}'
```

Необходимо отметить, что этот подход позволяет протестировать и проверить модель, однако для продакшена это не очень эффективный способ.

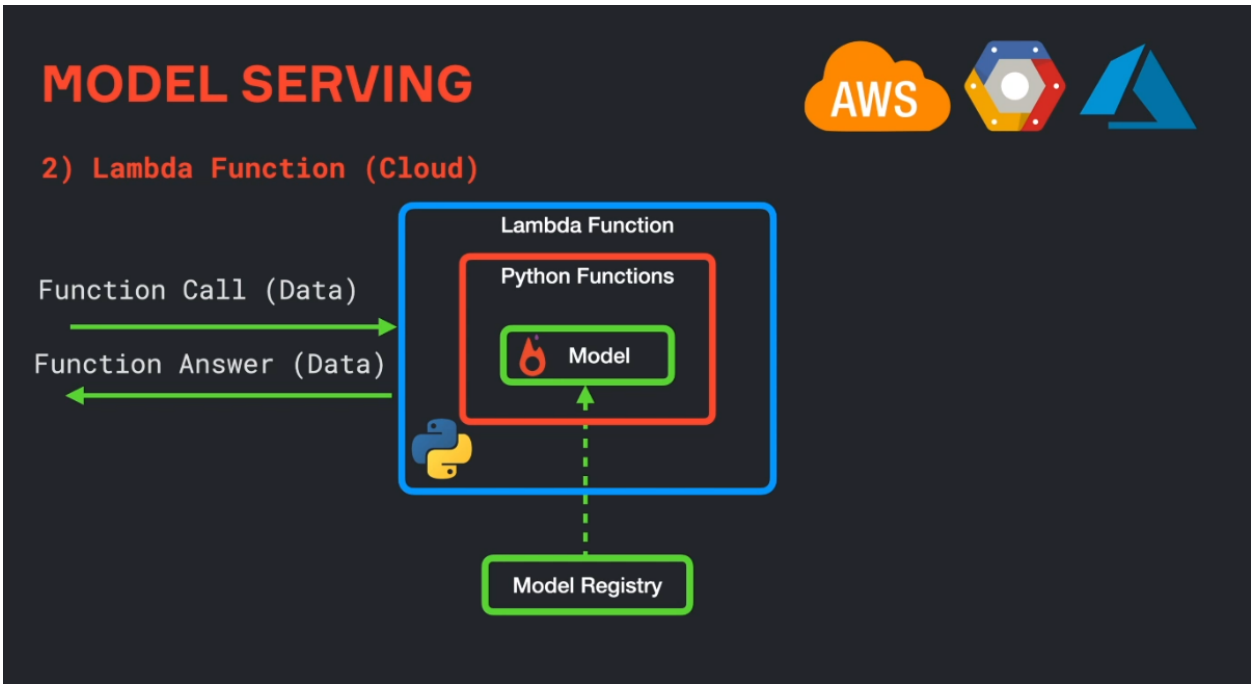
Весь процесс Model Serving можно представить в виде такой матрешки.



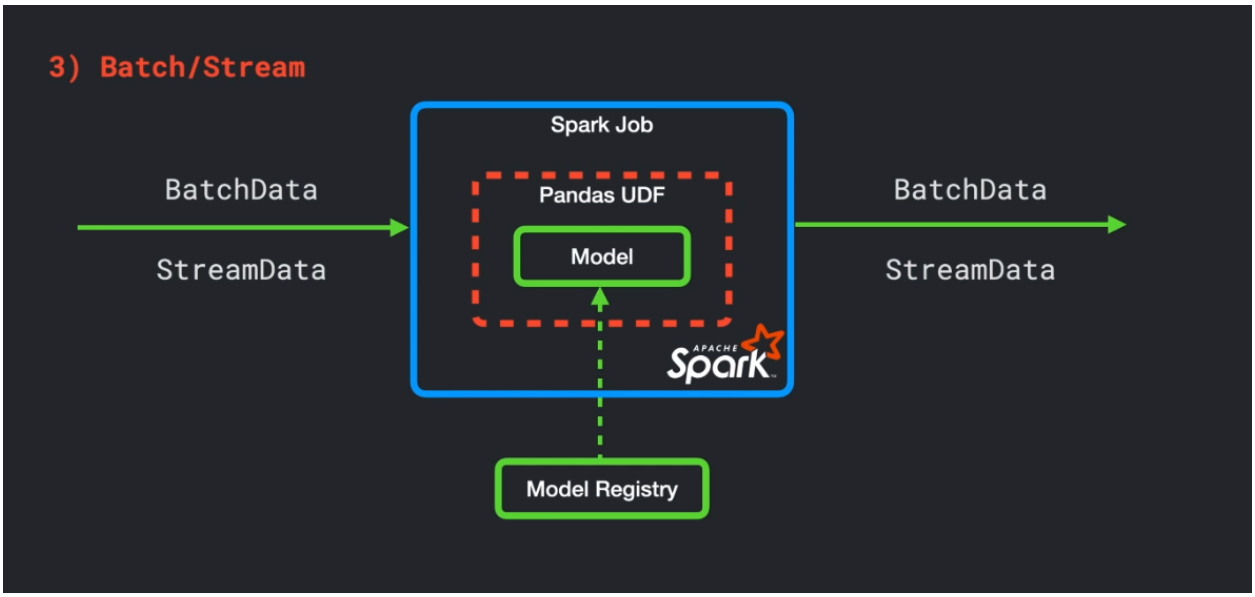
Первый подход в сервировке моделей - сервировка через микросервисы. Это популярный подход для широкого класса задач, он масштабируется и выдерживает высокие нагрузки.



Еще один подход - Lambda Function (cloud).



Последний вариант сервировки моделей - сервировка через spark pipeline. Это хороший вариант, когда вам нужно применить вашу модель к массивному объем данных


3) Batch/Stream
Spark Job
BatchData
StreamData
Pandas UDF
Model
BatchData
StreamData
Model Registry