Развертывание ML-моделей в продакшн

Ирина Степановна Трубчик

https://t.me/+PsC-JDrwrvsxNmVi

Лекция 7

Цели занятия

1. Понять жизненный цикл модели от обучения до продакшена
2. Изучить паттерны развертывания (batch, online, streaming)
3. Освоить инструменты: Docker, Kubernetes, облачные платформы
4. Научиться мониторить и обновлять модели в production

Сталкивались ли вы с развертыванием моделей?
Какие проблемы возникали?

Что такое deployment ML-модели

Model Deployment — процесс интеграции обученной ML-модели в production-окружение, где она будет делать предсказания на реальных данных.

ключевые компоненты:

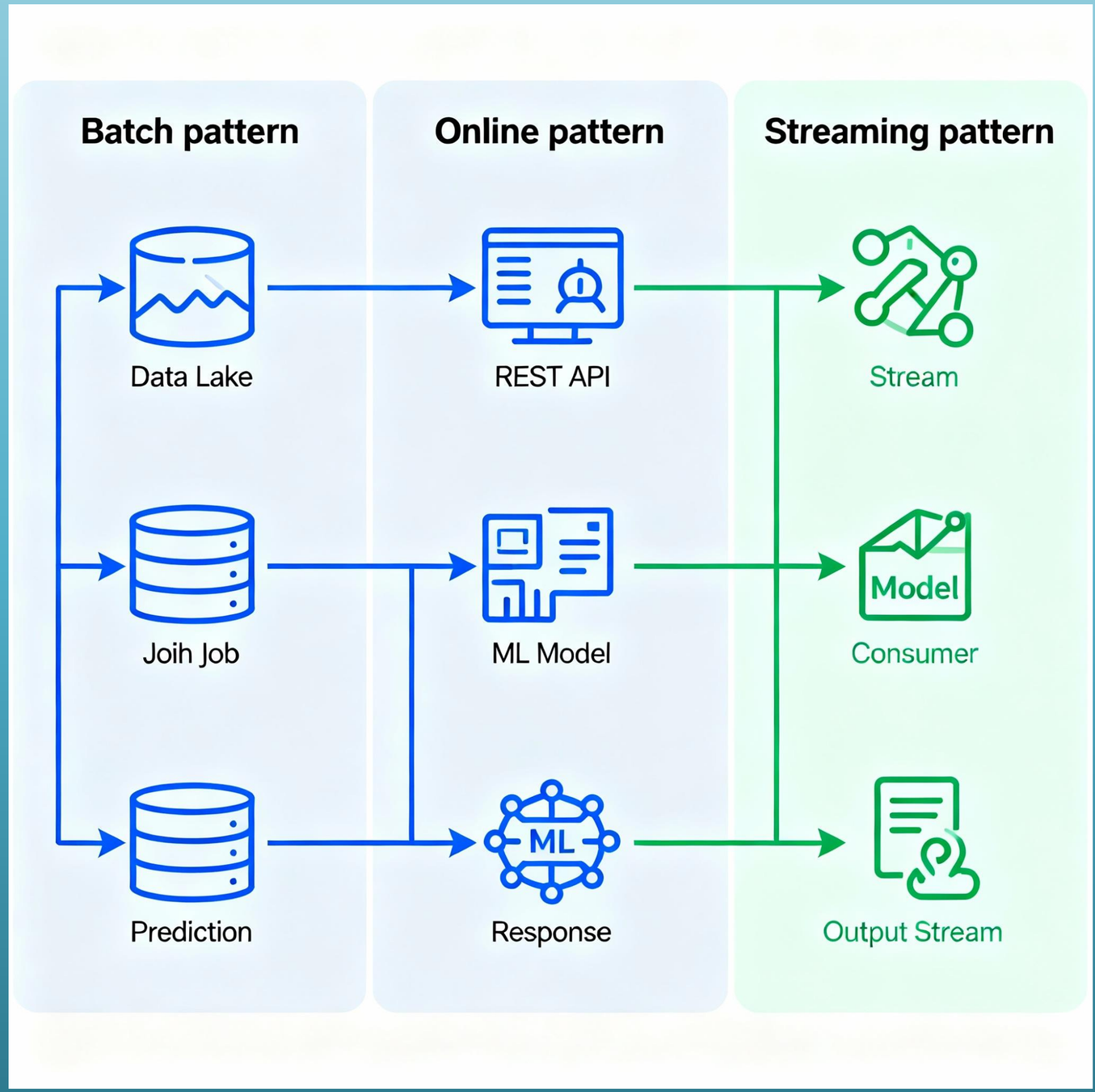
- Model artifacts — сериализованная модель (pickle, joblib, ONNX)
- Inference service — сервис, обрабатывающий запросы
- Dependencies — библиотеки и окружение
- Infrastructure — серверы, контейнеры, оркестраторы

Схема жизненного цикла

Паттерны развертывания

Batch Prediction (пакетная обработка)	Online/Real-time Prediction (реалтайм)	Streaming Prediction
➢ Обработка данных по расписанию (daily, hourly) ➢ Подходит для некритичных по времени задач ➢ Пример: рекомендации email-рассылок	➢ API-сервис отвечает на запросы немедленно ➢ Низкая latency (<100ms) ➢ Пример: fraud detection, рекомендации в e-commerce	➢ Обработка потоковых данных (Kafka, Kinesis) ➢ Для непрерывных потоков событий ➢ Пример: мониторинг IoT-датчиков

Диаграмма паттернов



Serving архитектуры

Model-in-Service (embedded)

```python
# Модель загружена в памяти сервиса
from fastapi import FastAPI
model = load_model("model.pkl")

@app.post("/predict")
def predict(data):
    return model.predict(data)
```

✅ Простота, низкая задержка (latency)

❌ Сложность обновления

Model-as-Service (dedicated)

```text
[Client] → [Gateway] → [Model Service 1]
                     → [Model Service 2]
```

✅ Независимое масштабирование

❌ Дополнительная latency

Sidecar Pattern

```text
[App Container] ↔ [Model Container]
```

✅ Изоляция, легкое обновление

❌ Дополнительные ресурсы

Контейнеризация с Docker

Зачем контейнеры для ML?

- Изоляция зависимостей
- Воспроизводимость окружения
- Портабельность между средами

Минимальный Dockerfile:

```text
FROM python:3.10-slim

WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY models/ models/
COPY src/ src/

EXPOSE 8080
CMD ["uvicorn", "src.api:app", "--host", "0.0.0.0", "--port", "8080"]
```

Сборка и запуск:

```bash
docker build -t ml-model:v1 .
docker run -p 8080:8080 ml-model:v1
```

Какие еще зависимости нужно включить в образ?

Kubernetes для ML

Почему Kubernetes?

- Автоматическое масштабирование (HPA)
- Self-healing (перезапуск упавших pods)
- Обнаружение служб и балансировка нагрузки
- Управление secrets и configs

Минимальный Deployment manifest:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ml-model
spec:
  replicas: 3
  selector:
    matchLabels:
      app: ml-model
  template:
    metadata:
      labels:
        app: ml-model
    spec:
      containers:
      - name: model
        image: ml-model:v1
        ports:
        - containerPort: 8080
        resources:
          requests:
            memory: "512Mi"
            cpu: "500m"
          limits:
            memory: "1Gi"
            cpu: "1000m"
```

```yaml
---
apiVersion: v1
kind: Service
metadata:
  name: ml-model-service
spec:
  selector:
    app: ml-model
  ports:
  - port: 80
    targetPort: 8080
  type: LoadBalancer
```

Стратегии развертывания

Blue-Green Deployment	Canary Deployment	A/B тестирование
text [Blue (v1)] ← 100% traffic [Green (v2)] ← ready → Switch traffic instantly text [Blue (v1)] ← idle [Green (v2)] ← 100% traffic ✓ Мгновенный rollback ✗ Требует 2x ресурсов	text [v1] ← 90% traffic [v2] ← 10% traffic → Gradually increase text [v1] ← 50% traffic [v2] ← 50% traffic ✓ Снижение рисков ✗ Сложность мониторинга	text Group A → [Model v1] Group B → [Model v2] ✓ Бизнес-метрики ✗ Требует аналитику

Облачные платформы

AWS SageMaker

```python
from sagemaker.sklearn import SKLearnModel

model = SKLearnModel(
    model_data="s3://bucket/model.tar.gz",
    role=role,
    entry_point="inference.py"
)

predictor = model.deploy(
    instance_type="ml.m5.large",
    initial_instance_count=2
)
```

Google Vertex AI

```python
from google.cloud import aiplatform

endpoint = aiplatform.Endpoint.create(display_name="ml-model")
endpoint.deploy(
    model=model,
    machine_type="n1-standard-4",
    min_replica_count=1,
    max_replica_count=10
)
```


```python
from azureml.core.webservice import AciWebservice

deployment_config = AciWebservice.deploy_configuration(
    cpu_cores=2,
    memory_gb=4
)

service = Model.deploy(
    workspace=ws,
    name="ml-model",
    models=[model],
    deployment_config=deployment_config
)
```

Российские облачные платформы

Yandex.Cloud	VK Cloud (бывший MCS)	Ростелеком/Ростелеком Cloud
➢ Container Registry — хранилище Docker образов ➢ Managed Kubernetes — управляемый K8s сервис ➢ Полная интеграция с Russian Federation, соответствие законам о локализации данных	➢ Kubernetes as a Service ➢ Хранение данных в России (Frankfurt, Moskva regions) ➢ Интеграция с VK экосистемой	➢ На базе OpenStack + Kubernetes ➢ Для госзаказов особенно популярно

```bash
# Пример: создание K8s кластера в Yandex.Cloud
yc k8s cluster create \
  --name my-cluster \
  --region ru-central1 \
  --version 1.28
```

сравнительная таблица

Решение	Тип	Российское	Функциональность	Сложность	Best for
Kubernetes	Open-source	Нет (Google)	★★★★★	Высокая	Production, большие системы
Yandex.Cloud K8s	Managed	✅	★★★★★	Средняя	Облачные системы в России
Docker Swarm	Open-source	Нет	★★★	Низкая	Небольшие проекты, быстрый старт
Apache Mesos	Open-source	Нет	★★★★	Высокая	Дата-центры, высоконагруженные
VK Cloud K8s	Managed	✅	★★★★★	Средняя	VK экосистема
Ростелеком Cloud	Managed	✅	★★★★	Средняя	Госзаказы, on-premise

Мониторинг в production

Operational Metrics	Model Quality Metrics	Business Metrics	Инструменты:
➢ Latency (p50, p95, p99) ➢ Throughput (requests/sec) ➢ Error rate ➢ Resource utilization (CPU, RAM)	➢ Prediction distribution drift ➢ Feature drift ➢ Accuracy degradation ➢ Outlier detection	➢ Conversion rate ➢ Revenue impact ➢ User satisfaction	➢ Prometheus + Grafana ➢ Evidently AI ➢ WhyLabs ➢ Seldon Alibi

Пример Prometheus метрик:

```
from prometheus_client import Counter, Histogram

predictions_total = Counter('predictions_total', 'Total predictions')
prediction_latency = Histogram('prediction_latency_seconds', 'Prediction latency')

@app.post("/predict")
def predict(data):
    with prediction_latency.time():
        result = model.predict(data)
        predictions_total.inc()
        return result
```

Model Registry и версионирование

```
import mlflow

# Регистрация модели
mlflow.register_model(
    model_uri="runs:/abc123/model",
    name="flight-delay-predictor"
)

# Продвижение в Production
client = mlflow.tracking.MlflowClient()
client.transition_model_version_stage(
    name="flight-delay-predictor",
    version=3,
    stage="Production"
)

# Загрузка для deployment
model = mlflow.pyfunc.load_model(
    model_uri="models:/flight-delay-predictor/Production"
)
```

Stages:

- ✓ None → только что зарегистрирована
- ✓ Staging → тестируется
- ✓ Production → активно используется
- ✓ Archived → устаревшая

развертывание модели (demo)

Подготовка модели

```python
# train.py
import mlflow
import joblib
from sklearn.ensemble import RandomForestClassifier

with mlflow.start_run():
    model = RandomForestClassifier()
    model.fit(X_train, y_train)

    mlflow.sklearn.log_model(model, "model")
    mlflow.register_model("runs:/.../model", "demo-model")
```

развертывание модели (demo)

Создание API:

```python
# api.py
from fastapi import FastAPI
import mlflow.pyfunc

app = FastAPI()
model = mlflow.pyfunc.load_model("models:/demo-model/Production")

@app.post("/predict")
def predict(features: dict):
    prediction = model.predict([list(features.values())])
    return {"prediction": prediction[0]}
```

развертывание модели (demo)

Развертывание

```bash
docker build -t demo-model .
docker run -p 8080:8080 demo-model


# Тест
curl -X POST http://localhost:8080/predict \
  -H "Content-Type: application/json" \
  -d '{"feature1": 0.5, "feature2": 0.3}'
```

Чек-лист best practices

Pre-deployment:	Deployment:	Post-deployment:
❑ Модель зарегистрирована в Model Registry ❑ Все тесты пройдены (unit, integration, model quality) ❑ Документация API обновлена ❑ Определены SLA и resource limits ❑ Настроен мониторинг и алерт	❑ Используется версионирование образов ❑ Применяется canary/blue-green стратегия ❑ Настроен health check endpoint ❑ Логируются все предсказания для аудита ❑ Есть rollback plan	❑ Мониторинг latency и throughput ❑ Отслеживание data drift ❑ Сравнение business metrics с baseline ❑ Регулярный review логов ❑ План обновления модели

Ключевые вопросы для самопроверки:

1. Что такое model deployment и какие этапы он включает?

2. В чем разница между batch и online inference?

3. Зачем нужен Model Registry?

4. Что такое canary deployment?

5. Какие метрики важно мониторить в production?

Материалы и ссылки

Документация и туториалы:

1. MLflow Model Registry
2. Kubernetes ML Patterns
3. AWS SageMaker Deployment Guide
4. Seldon Core — open-source ML deployment

Книги:

1. "Building Machine Learning Powered Applications" by Emmanuel Ameisen
2. "Machine Learning Design Patterns" by Lakshmanan, Robinson, Munn

Курсы:

1. DeepLearning.AI: "MLOps Specialization"
2. DataLens: анализ и визуализация данных: https://yandex.cloud/ru/training/datalens

Материалы и ссылки

Инструменты:

1. BentoML — framework для packaging и serving
2. TorchServe — deployment для PyTorch
3. TensorFlow Serving
4. KServe (ex-KFServing)
5. https://datalens.ru/

Домашнее чтение:

Изучить документацию одной из облачных платформ для ML deployment и написать краткое сравнение.

Вопросы

Телеграм https://t.me/+PsC-JDrwrvsxNmVi

СКИФ (https://do.skif.donstu.ru/course/view.php?id=7508)