# Feature Store и продвинутые инструменты

**Ирина Степановна Трубчик**

Лекция 13

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

# Что будет на лекции?

- Проблема: почему нужен Feature Store?
- Архитектура Feature Store
- Популярные решения (Tecton, Feast, DeltaLake)
- Feature Management: versioning и lineage
- Практика: как интегрировать в production

# 🚨 Проблема: Feature Hell

- **Дублирование:** Одну фичу пишут разные команды в разных местах
- **Несогласованность:** Feature Training = X, Feature Inference = Y (mismatch!)
- **Сложность:** 200+ строк Pandas для одной фичи, сложно поддерживать
- **Производительность:** Долгая загрузка фич для batch vs real-time

# Что такое Feature Store?

Feature Store — централизованная система управления features для ML моделей

- **Один источник истины (SSOT):** Все фичи в одном месте
- **Версионирование:** Track изменения фич
- **Reusability:** Одну фичу пишешь раз, используют все
- **Monitoring:** Отслеживай quality фич

# Архитектура Feature Store

**Batch Layer**

Исторические фичи для тренировки

**Real-Time Layer**

Свежие фичи для inference

**Storage**

DWH, Data Lake, Cache

**Serving**

Fast lookup for models

# Workflow Feature Store

1. **Define:** Опиши фичу (name, type, computation)
2. **Compute:** Вычисли для batch и real-time
3. **Store:** Сохрани в batch store (DWH) + online store (Redis)
4. **Serve:** Достань фичи для inference milliseconds

# Feast: Open-Source Feature Store

Разработана Google, Gojek. Open-source, Python-first, cloud-agnostic.

- **Feature Definition:** Python SDK для описания фич
- **Materialize:** Вычисли и сохрани в offline/online store
- **Retrieve:** Get features для training или serving
- **Monitor:** Track feature stats (mean, null %)

# 💻 Feast: Пример кода

```
from feast import Feature, FeatureView, Entity
from feast import PushSource, FeatureStore

# 1. Define Feature View
@batch_source
def get_user_features(entity_df):
return compute_features(entity_df)

user_view = FeatureView(
name="user_features",
entities=["user_id"],
source=get_user_features
)

# 2. Materialize (daily job)
fs = FeatureStore(".")
fs.materialize(features=["user_features"])

# 3. Get features for serving
features = fs.get_online_features(
features=["user_features:age"],
entity_rows=[{"user_id": 123}]
)
```

# ⚡ Tecton: Enterprise Feature Store

Коммерческая платформа. Основана бывшими инженерами Uber (авторы Michaelangelo)

- **Real-Time Features:** Millisecond latency из Redis/DynamoDB
- **SQL Interface:** Пиши features на SQL
- **Automatic Computation:** Расписание вычисления по гибкому расписанию
- **Freshness SLA:** Гарантия когда фича будет готова

# Delta Lake: Foundation для Feature Store

ACID транзакции на Data Lake. Используется как storage для features.

- **ACID:** Не половинчатые writes, полная консистентность
- **Time Travel:** Query данные в любой момент времени (версионирование)
- **Schema Enforcement:** Не можешь add random column
- **Streaming:** Append новые данные из Kafka в реальном времени

# 📌 Feature Versioning

Как отслеживать изменения фич без breaking production?

- **Version Feature:** user_age_v1, user_age_v2
- **Blue/Green Deploy:** Новая версия параллельно, потом switch
- **Canary Rollout:** 10% traffic → new feature, потом 100%
- **Backward Compat:** Старые модели должны работать со старыми фичами

# ⛓ Feature Lineage & Monitoring

Отслеживай откуда приходят данные и как меняются.

- **Data Lineage:** table_A → transform → feature_X → model_Z
- **Feature Health:** mean, std, null %, cardinality
- **Data Drift:** Распределение feature изменилось? Alert!
- **Model Impact:** Какие фичи most important для модели?

# ML Metadata (MLMD)

Google MLMD library. Отслеживай всё что производит ML pipeline.

**Artifact**

model.pkl, train.csv

**Execution**

Training run #123

**Context**

Project, experiment

# ✓ Data Quality & Validation

Great Expectations: Framework для data quality checks

- **Schema Validation:** age должен быть integer, 0-150
- **Completeness:** Нет null values в критичных полях
- **Uniqueness:** user_id должен быть уникален
- **Freshness:** Данные не старше 1 часа

# 🔧 Orchestration Tools

## Airflow

DAG scheduler. Complex workflows.

## Prefect

Modern dataflow. Python-native.

## Dagster

Asset-based. Type-safe pipelines.

# ⚡ Real-Time Feature Serving

**Redis**
In-memory cache. <5ms latency

**DynamoDB**
AWS managed. Scalable key-value.

**Cassandra**
Distributed. High availability.

# 🚗 Case Study: Uber Michaelangelo

Feature Store для 10,000+ моделей в production

- **Challenge:** Каждая команда писала фичи по-своему. Дублирование 80%
- **Solution:** Michaelangelo Feature Store. Centralized features
- **Result:** 50% сокращение time-to-market для новых моделей
- **Impact:** Улучшенное quality, меньше ошибок

# 🛏 Case Study: Airbnb ZipLine

Real-time ML feature platform

- **Problem:** Training-serving skew. Model accuracy drop в production
- **Solution:** ZipLine. Same computation for train и inference
- **Real-time:** Features updated every minute. <50ms latency
- **Benefit:** 15% improvement в model accuracy после deployment

# Best Practices

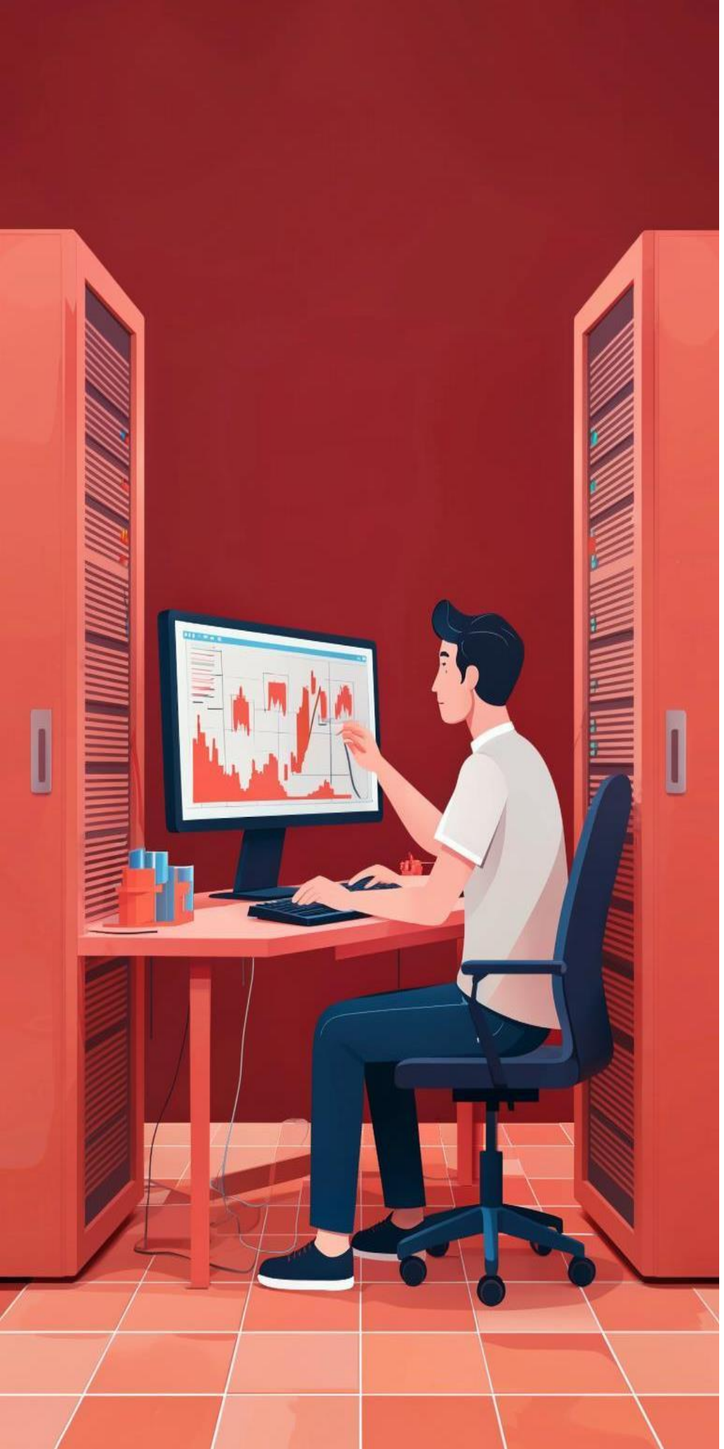
1. **Start simple:** Не все features в Feature Store. Начни с критичных
2. **Version everything:** Feature code, definitions, schemas
3. **Monitor quality:** Data drift alerts, null % tracking
4. **Document features:** Why, when, who's using, schema

# ✨ Feature Store = MLOps Success

Централизованное управление

Масштабируемые модели

Быстрая разработка



# Ключевые вопросы для самопроверки:

**Вопрос 1:** Дайте определение Feature Store и объясните, какие три основные проблемы возникают в проектах без централизованного управления фичами.

**Вопрос 2:** Нарисуйте (или опишите) типичную архитектуру Feature Store. Объясните разницу между оффлайн-слоем и онлайн-слоем, их роли, требования к latency и используемые хранилища.

**Вопрос 3:** Опишите полный жизненный цикл фичи от концепции до деприкейта. Какие проверки и валидации должны пройти на каждом этапе?

**Вопрос 4:** Объясните, как Feature Store помогает избежать training–serving skew. Приведите конкретный пример для проекта прогнозирования задержек рейсов: какую фичу можно использовать и как она должна вычисляться одинаково в оффлайн и онлайн?

**Вопрос 5:** Назовите основные отличия между Feast и облачными Feature Store (AWS, GCP, Azure). Когда использовать какой?

**Вопрос 6:** Какие метрики и алерты должны мониториться в Feature Store? Как они связаны с моделью?

# Вопросы

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

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