

Управление версиями моделей в MLOps

Ирина Степановна Трубчик

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

Лекция 11

Цели занятия

1

Зачем управлять версиями


2

5 компонентов версионирования


3

Семантика + примеры

Зачем управлять версиями

 Без версионирования
(хаос, потери, rollback)

vs

 С версионированием
(порядок, история, control)

Проблема без версионирования:

Вопрос	Без версионирования	С версионированием
Какая модель в продакшене?	🧑‍🔧 Неизвестно	✅ v1.2.3 от 2025-11-20
Как откатиться на старую?	❌ Невозможно	✅ На один клик
Какая точность была раньше?	❌ Потеряно	✅ Все метрики сохранены
Кто деплоил последний раз?	❌ Нет истории	✅ audit_log.json

Компоненты Model Versioning (1,2,3)

Model Registry —
центральное
хранилище моделей

- Где: MLflow Model Registry, Kubeflow, TensorFlow Serving
- Что: метаданные, версии, tags, метрики

Semantic Versioning —
система нумерации
(MAJOR.MINOR.PATCH)

- v1.0.0 → v1.0.1 (fix) → v1.1.0 (feature) → v2.0.0 (breaking)
- Пример: flight_delay_v2.1.3

Metadata & Artifacts —
что сохранять

- Модель (веса, параметры)
- Метрики (accuracy, F1, ROC-AUC)
- Hyperparameters
- Data signature (schema)

Компоненты Model Versioning (4, 5)

Promotion Pipeline —
staging → prod

- Development → Staging → Canary → Production
- Автоматизированные проверки на каждом этапе

Rollback Mechanism

- Храним историю всех версий
- 1-click откат на предыдущую версию

Пример версионирования

MODEL REGISTRY (Центральное хранилище)

v1.0.0 → v1.0.1 → v1.1.0 → v2.0.0 → v2.1.0
(Prod) (Staging) (Dev) (Rollback) (Current)

Semantic Versioning в MLOps

MAJOR.MINOR.PATCH

v1.0.0

| | └ PATCH: баги, optimization
| └── MINOR: новые фичи, улучшения
└── MAJOR: breaking changes, несовместимость

Примеры для ML моделей:

Версия	Изменение	Причина
v1.0.0 → v1.0.1	Улучшена обработка пропусков	Bug fix в preprocessing
v1.0.1 → v1.1.0	Добавлены 3 новых признака	Feature engineering
v1.1.0 → v2.0.0	Изменена target variable (classification → regression)	Breaking change
v2.0.0 → v2.0.1	Нормализация параметров	Optimization
v2.0.1 → v2.1.0	Новый алгоритм (XGBoost)	Feature

v1.0.0 (Released) → v1.0.1 (Patch) → v1.1.0 (Minor) → v2.0.0 (Major)

(Stable) (+fix) (+feature)

(Breaking)

Model Registry (MLflow Model Registry)

Что такое Model Registry?

- Центральное хранилище моделей с управлением версиями, метаданными и жизненным циклом.

Компоненты:

Registered Model — группа версий одной модели	Model Version — конкретная версия модели	Stages — жизненные циклы версии	Tags & Aliases — поиск и отслеживание
<ul style="list-style-type: none">* Имя: flight_delay_predictor* Версии: v1, v2, v3, ...	<ul style="list-style-type: none">* Беса, hyperparameters, artifacts* Метаданные (created_time, run_id)* URI: models:/flight_delay_predictor/1	<ul style="list-style-type: none">* None → Staging → Production → Archived* Может быть одна модель в Production, одна в Staging	<ul style="list-style-type: none">* Тег: "champion" → best model* Алиас: "latest" → latest version

Пример из репозитория

Registered Model: flight_delay_predictor

Version 1 (Archived)

Accuracy: 0.82, F1: 0.79

Created: 2025-10-01

↓

Version 2 (Production) ★ Champion

Accuracy: 0.85, F1: 0.83

Created: 2025-10-15, Tags: [prod, v2]

↓

Version 3 (Staging)

Accuracy: 0.86, F1: 0.84

Created: 2025-11-20

MLflow Model Registry:

практика - *Регистрация модели в MLflow:*

python

```
# 1. Логирование модели в experiment
mlflow.sklearn.log_model(model, "model")

# 2. Регистрация в Model Registry
mlflow.register_model(
    model_uri="runs:/ABC123/model",
    name="flight_delay_predictor"
)

# 3. Добавление в Staging
client = mlflow.tracking.MlflowClient()
client.transition_model_version_stage(
    name="flight_delay_predictor",
    version=2,
    stage="Staging"
)
```

MLflow Model Registry:

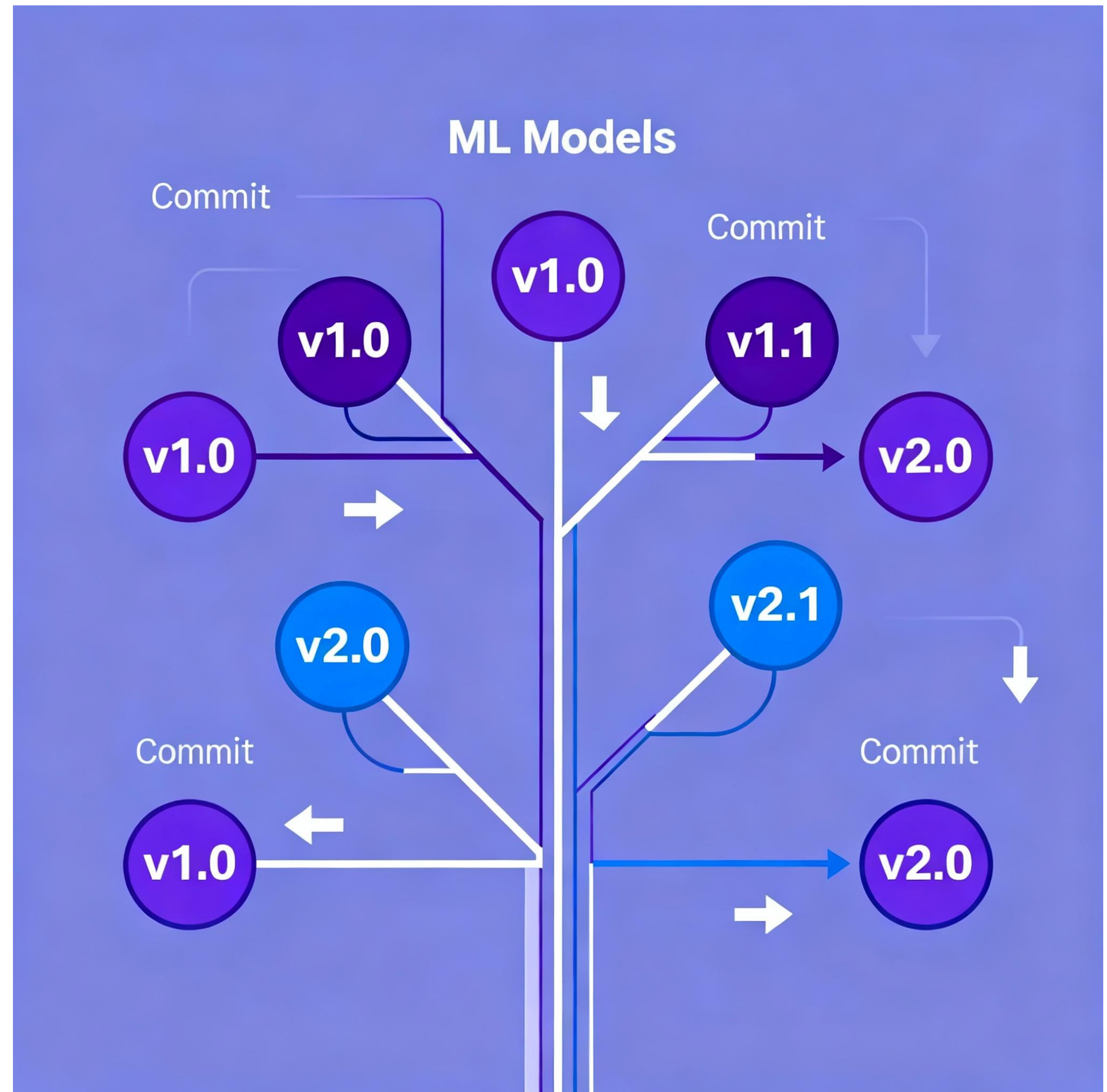
практика - *Регистрация модели в MLflow:*

```
# 4. Промоция в Production
client.transition_model_version_stage(
    name="flight_delay_predictor",
    version=2,
    stage="Production"
)

# 5. Получение Production модели
production_model = mlflow.pyfunc.load_model(
    "models:/flight_delay_predictor/Production"
)
```

MLflow Model Registry

```
experiment
  ↓ log_model
mlflow_ui (Artifacts)
  ↓ register_model
MLflow Model Registry
  ├── Staging (тестирование)
  ├── Production (пользователи)
  └── Archived (история)
```



Жизненный цикл модели

- **Experimentation (Development):** Обучаем разные модели; Логируем в MLflow experiments; Сравниваем метрики
- **Staging (Testing):** Деплоим кандидата на staging; A/B тесты; Валидация на реальных данных; Проверка latency
- **Production Deployment:** Когда staging прошел тесты; Canary deployment (5% трафика → 50% → 100%); Мониторинг производительности
- **Monitoring (Maintenance):** Отслеживание accuracy; Детекция data drift; Алерты при деградации; A/B тесты новых версий
- **Rollback / Archiving:** Если accuracy упал → rollback на v-1; Если модель устарела → archive; Храним историю навсегда

Model Lifecycle – стратегия развертывания

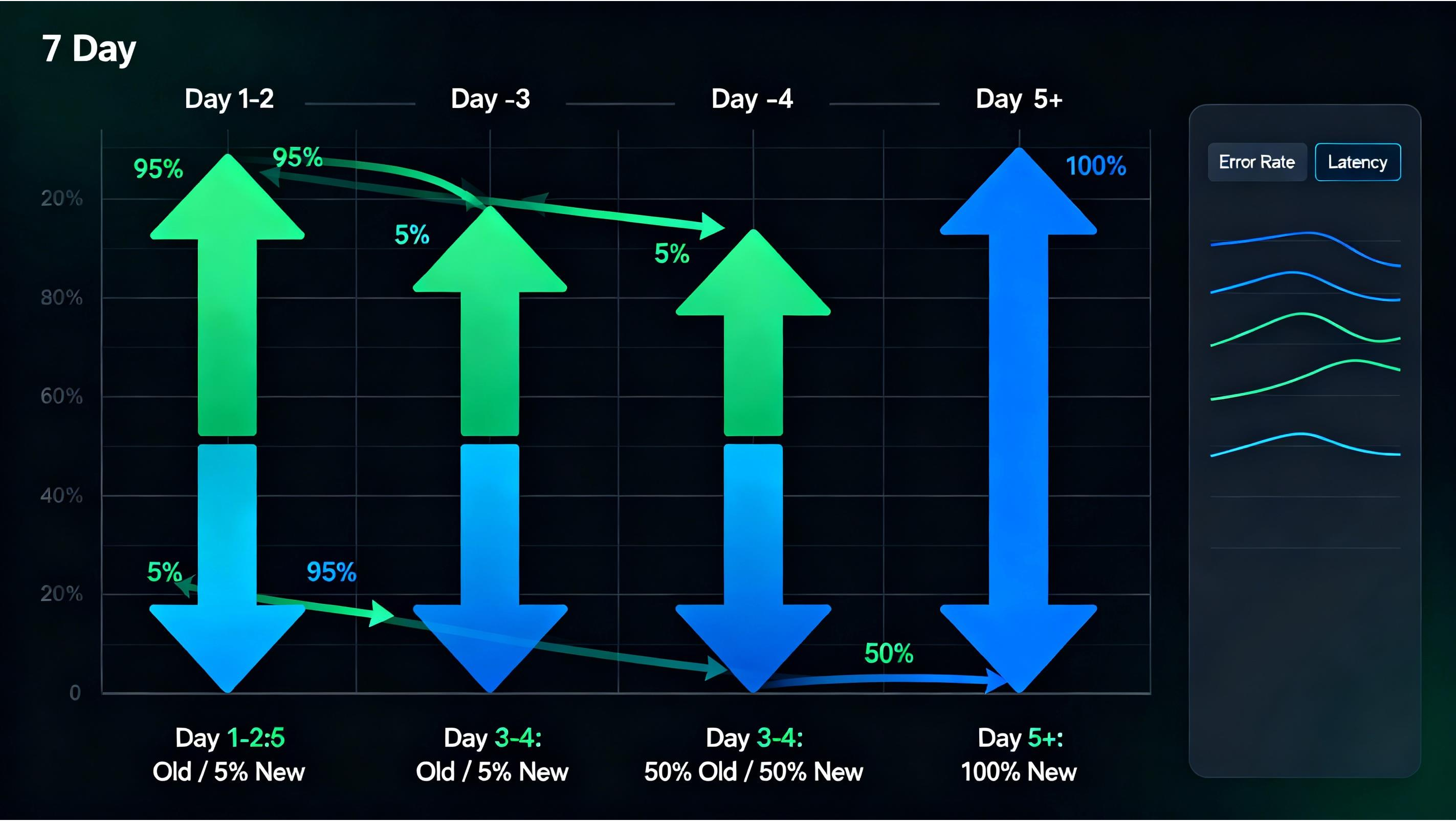
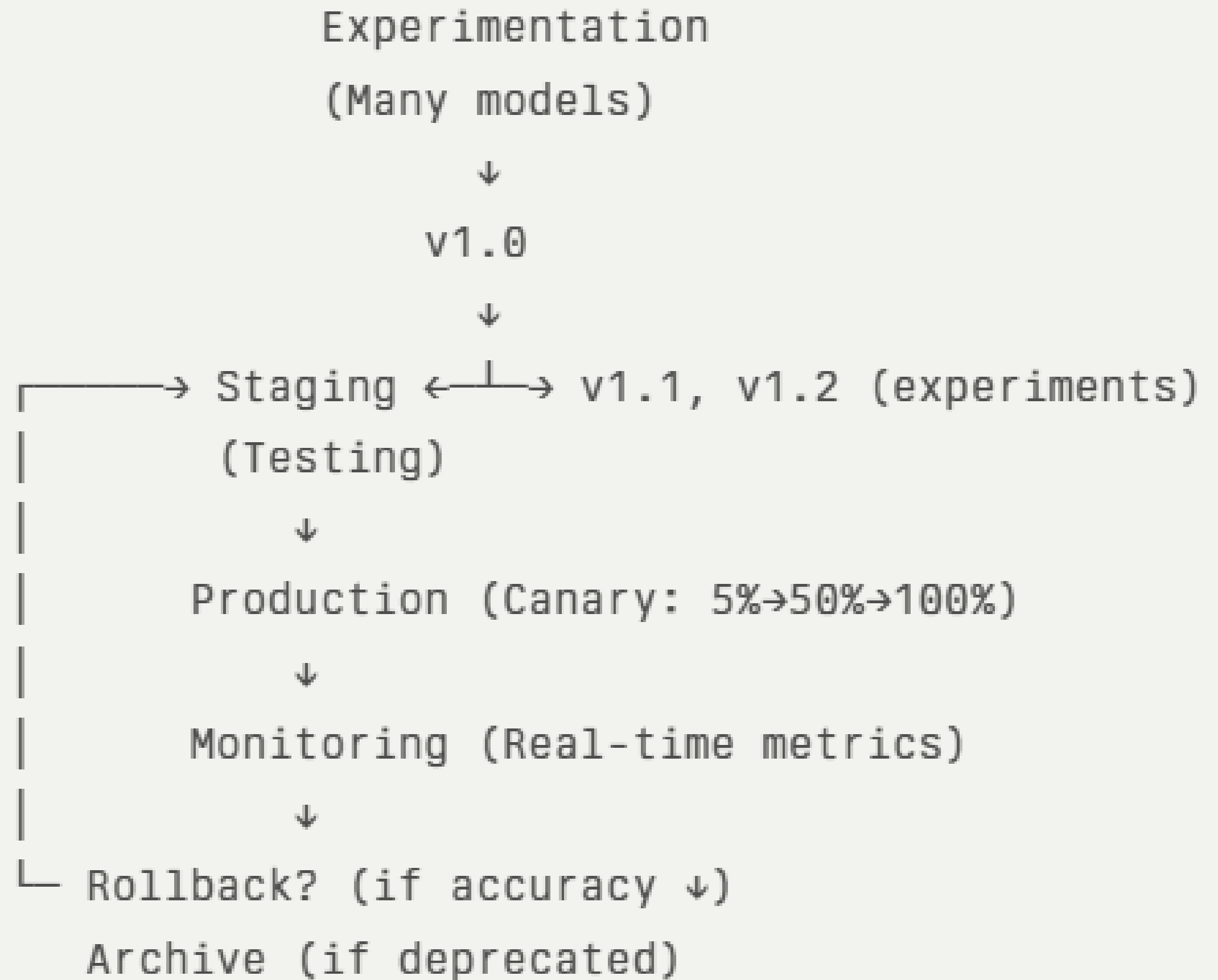


Схема версионирования



Data Drift & Model Decay

➤ Почему модели деградируют?

▮ Data Drift — распределение данных изменилось

Пример: рейсы после COVID-19 летают в других часы

Модель обучена на старом распределении

Результат: accuracy ↓ 0.85 → 0.71

➤ 🎯 Concept Drift — целевая переменная изменилась

Пример: определение "задержки" изменилось (было 15 мин, теперь 30 мин)

Модель предсказывает старое определение

Результат: неверные predictions

➤ сезонные паттерны

Новогодние праздники, отпуска, забастовки

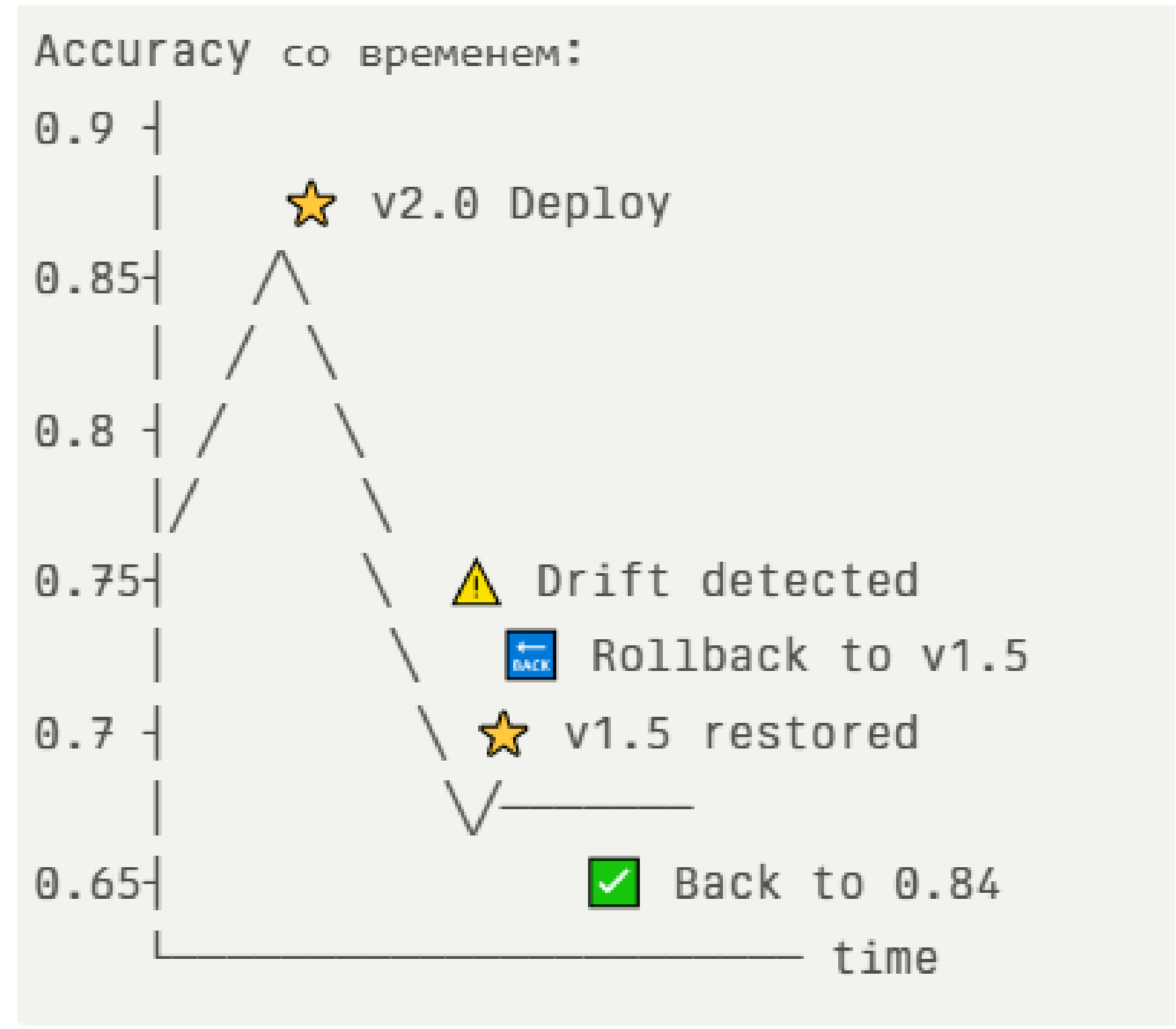
Модель не видела эти паттерны

Результат: anomalies в predictions

Data Drift & Model Decay

Решения:

- ✓ Периодическое переобучение (раз в неделю/месяц)
- ✓ Статистические тесты на drift (KS test, Wasserstein distance)
- ✓ Автоматический rollback при деградации
- ✓ Version control для быстрого отката



Сравнение инструментов версионирования

Инструмент	Особенность	Для чего	Минусы
MLflow	Python-first, easy, UI	Экспериментирование → deployment	Не масштабируется для 1000+ моделей
DVC	Git-like, pipeline versioning	Data versioning + model versioning	Требует настройки
Model Registry (custom)	Полный контроль	Специфичные требования	Много кода
Kubernetes	Container versioning	Production scale	Сложность
BentoML	Model packaging	Model serving + versioning	Специализирован
Hugging Face	Community models	NLP/CV transfer learning	Не для custom models

Automation & Promotion Pipeline

```
Code Push → Train → Register → Stage → Production
  ↓           ↓           ↓           ↓           ↓
  git        Retrain      MLflow      Tests      API
            new model    v2.1.0      (A/B,      |
                                   load,      ↳ Requests
                                   latency)
```

best practices версионирования

- ✓ Semantic Versioning — MAJOR.MINOR.PATCH всегда
- ✓ Metadata богата — сохраняй данные об обучении (dataset version, hyperparameters)
- ✓ Уникальные tags — champion, baseline, experiment-123
- ✓ Audit trail — who, when, what changed
- ✓ Staging before Production — никогда напрямую в prod
- ✓ Canary deployment — риск снижается постепенно (5% → 50% → 100%)
- ✓ A/B тесты — сравни на реальных данных
- ✓ Automatic rollback — не нужно ждать, когда someone заметит проблему
- ✓ Archive old versions — но храни их навсегда
- ✓ Document everything — why версия создана, что изменилось



Ключевые вопросы для самопроверки:

Вопрос 1: Компоненты Model Versioning (5 баллов)

Назовите 5 компонентов

Объясните каждый

Примеры систем

Вопрос 2: Semantic Versioning & Breaking Changes (5 баллов)

Сценарий: Модель требует новый признак (weather)

Выбрать версию: v1.3.1 vs v1.4.0 vs v2.0.0?

Объяснение почему не v1.x

Вопрос 3: MLflow Model Registry Stages (5 баллов)

Жизненный цикл в MLflow

Все stages: None, Staging, Production, Archived

Вопрос 4: Data Drift & Automatic Rollback (5 баллов)

Сценарий: Accuracy упал 0.85 → 0.71 (-16%)

Диагностика (Data Drift vs Concept Drift)

Вопрос 5: Выбор инструмента (5 баллов)

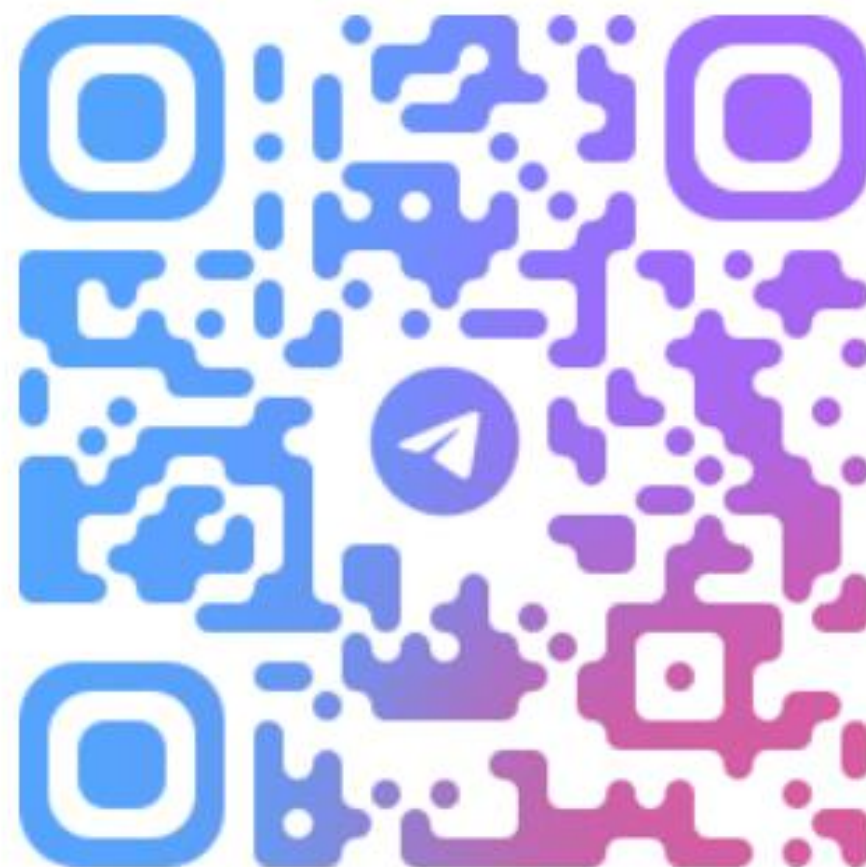
Startup с 8 людьми

Выбрать между MLflow и DVC

Вопросы



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



СКИФ

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