NY Yellow Taxi Cab

- von Open Data zur WebApp

ML OPS SOSE 2025

NADINE BRAUN

17. MAI 2025

Agenda

- Überblick
- Vorgehen
- Learning Journey
- Herausforderungen
- Lessons Learned
- Ausblick
- Fazit

Worum geht's?

- **Daten:**
  - Historische Fahrtdaten der Yellow Cabs in NY
  - Einzelfahrterfassung
  - Informationen zur Zahlung
  - Fahrtparameter
  - monatlich veröffentlicht im Parquet-Format
- **Ziel:**
  - Prognose stündlicher Taxinachfrage als Hilfestellung für Mobilitätsdienstleister oder auch zur Trafficplanung
  - Aufbau eines durchgängigen ML-Prozesses

- **Tools:**

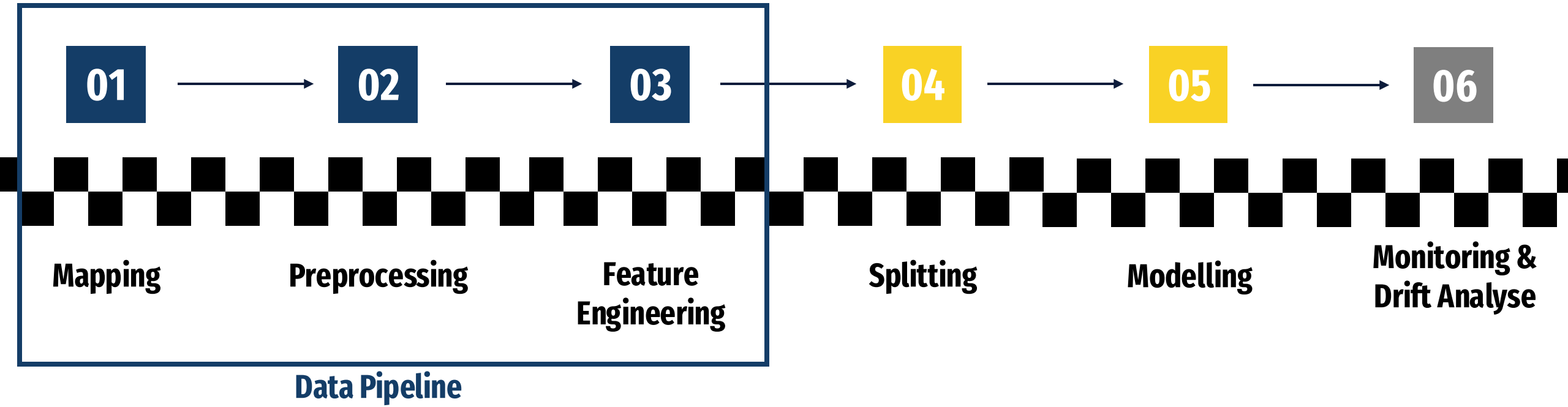
mlflow

Streamlit

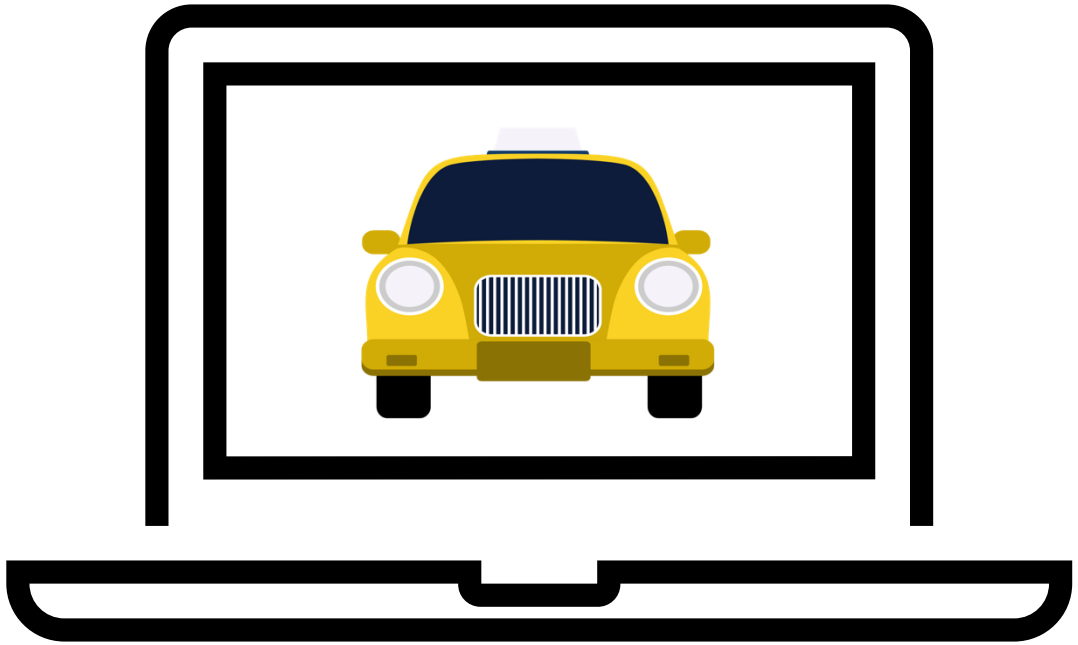
docker

HETZNER

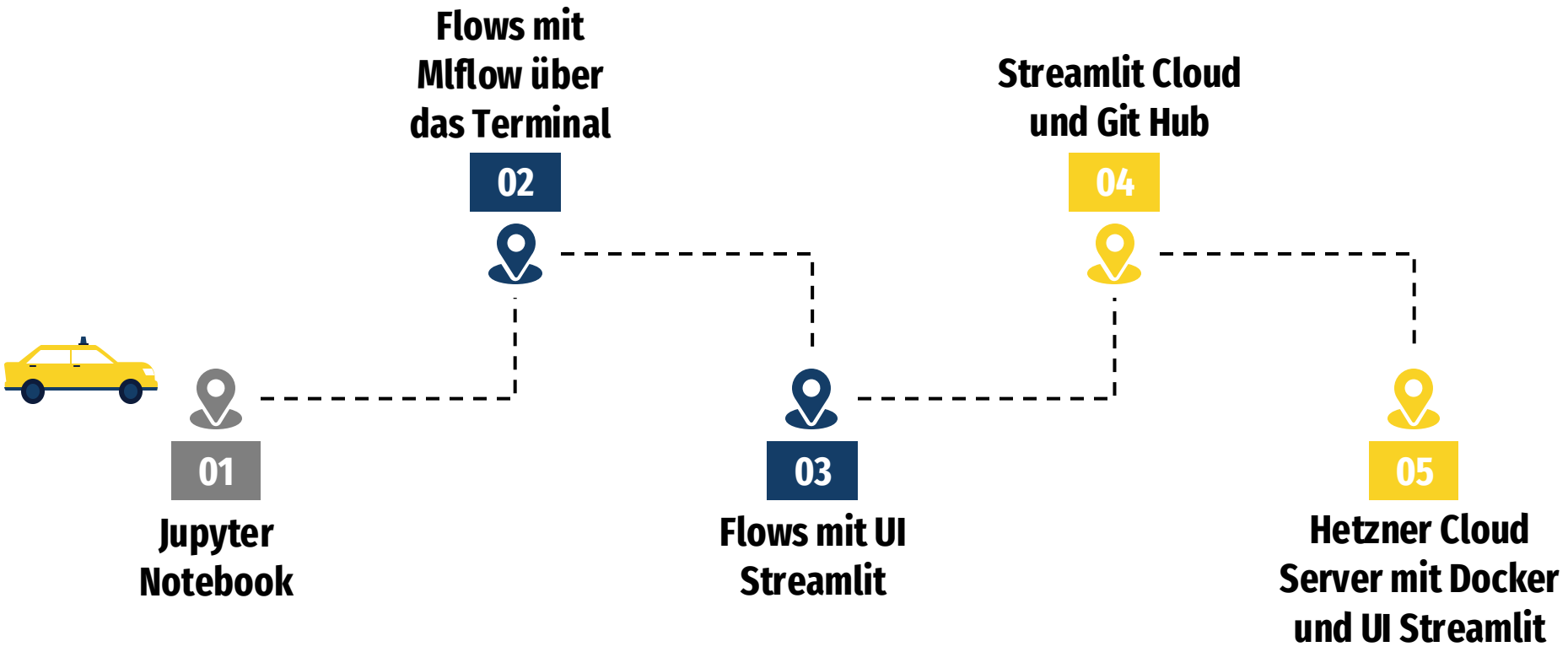
Wie das Projekt umgesetzt wurde



So sieht‘s aus



Raus aus der Comfortzone



Kein Zugriff – neue Daten, neue Komplexität

- Unternehmensdaten – nicht umsetzbar
- Entscheidung für Yellow Cab Datensatz: interessanter realer OpenData Satz, allerdings sehr umfangreich

- **Aggregation auf Stundenebene**
- **Bildung von Time-Buckets**
- **Summen der relevanten Features**

Offene Daten – aber nicht einheitlich

- Dezentrale Datenerfassung
- Unterschiedliche Spaltennamen, Formate, Kodierungen
- Zeitstempel & Payment-Formate ändern sich über Jahre

- **Frühzeitige Formatkontrolle als zentrale Maßnahme (Überblick verschaffen)**
- **flexibles Spalten-Mapping**
- **Validierungsroutinen**

Manuelles Debugging skaliert nicht

- Millionen Datensätze
- keine Einzelfalllösungen möglich
- Verschiedenste Fälle bereits in den Testdaten

- **Regelbasiertes Filtern statt manueller Ausnahmen**
- **Reproduzierbare Pipelines als Lösung**

Was ist wirklich relevant?

- Redundanz vs. Relevanz: viele Merkmale ähnlich
- Entscheidung für Reduktion auf wenige robuste Features

- **Robustheit statt Modellkomplexität**
- **Feature Engineering als konzeptioneller, nicht technischer Prozess**

Zu viele Tools – zu wenig Klarheit

- Anfängliche Überforderung
- viele Optionen, wenig Vergleichbarkeit
- Was verwende ich wofür?

- **„Einfach mal machen!“**
- **Erster Einstieg über Mlflow**
- **Neue Tools bei Bedarf hinzunehmen**

Jede Umgebung bringt neue Anforderungen

- Code nicht 1:1 übertragbar zwischen Notebook, Terminal, Streamlit, Deployment
- Manche Logik nicht übertragbar, z. B. input() vs. Streamlit-Komponenten

- **Technisches Verständnis für Abhängigkeiten wächst mit jedem Schritt**

Deployment war nicht das Ende – sondern ein neues Kapitel

- Viele Fehlerquellen schwer nachvollziehbar
  (z. B. alte Images, Caching-Probleme)
- Lange Ladezeiten
- Änderungen wirkten manchmal nicht
  – Ursache lag nicht (immer) im Code

- **Neuaufsetzen hilft oft**
- **systematisches Testen**
- **Geduld!**

Was ich jenseits des Codes gelernt habe

- Unsicherheiten gehören dazu –
  Entscheidungen müssen trotzdem getroffen werden
- Früh mit
  (Teil)Ergebnissen arbeiten hilft, Blockaden zu überwinden
- Visualisierung unterstützt nicht nur andere,
  sondern auch eigenes Denken
- Dokumentation ist kein Extra –
  sondern Voraussetzung für Verständnis
- Pragmatismus ist oft wirkungsvoller als Perfektionismus

Was noch möglich wäre

Datenintegration & Import

- Einbindung per Datenimport

Monitoring & Automatisierung

- Automatisierte Drift-Erkennung
- Retraining
- Alerts

Technische Skalierung & Steuerung

- Cloudfähige Architektur
- Zentrale Steuerung
- Modulare Containerstruktur

Was bleibt – im Code und im Kopf

01

Aus technischer Sicht

- Erster (einfacher) MLOps-Prozess umgesetzt
- (erster) Umgang mit neuen Tools
- Ein sauberes Konzept ist die halbe Miete

02

Aus persönlicher Sicht

- Ein Schritt zurück, kann auch einer nach vorn sein
- Ein lachendes und ein weinendes Auge