

Integrationsmöglichkeiten von Evidently: Implementierung einer ML-Monitoring-Architektur

Nhut Hoa Huynh

Department Informatik, Fakultät Technik und Informatik, Hochschule für
Angewandte Wissenschaften Hamburg, Germany
nhuthoa.huynh@haw-hamburg.de

Zusammenfassung. Das Paper untersucht die Integration von "Evidently", einer Python-Bibliothek zur Überwachung der Modellleistung, in den Lebenszyklus des maschinellen Lernens. Diese Integration erstreckt sich über verschiedene Umgebungen und Werkzeuge, darunter Notebook-Umgebungen, Workflow-Management-Tools sowie Tracking-Tools. Des Weiteren präsentiert das Paper eine ML-Monitoring-Architektur, die auf Evidently, Prefect, PostgreSQL und Grafana basiert. Diese Architektur ermöglicht die Überwachung von Datenqualität, Modelldrift und Modellqualität in Batch. Die Implementierung dieser Architektur bietet zahlreiche Vorteile, darunter die Einsatzfähigkeit sowohl für Batch- als auch für Echtzeit-ML-Systeme, die asynchrone Berechnung von Metriken und die Möglichkeit, Komponenten je nach Bedarf auszutauschen. Jedoch könnte die Komplexität des Projekts eine Herausforderung darstellen, insbesondere für Anwender, die nicht bereits mit den genannten Tools vertraut sind.

Schlüsselwörter: Evidently · Modellüberwachung · Maschinelles Lernen · Integration · Notebook-Umgebungen · Workflow-Management · Tracking-Tools · ML-Monitoring-Architektur · Prefect · PostgreSQL · Grafana · Modelldrift · Batch-Verarbeitung · Echtzeit-Systeme

1 Einleitung

1.1 Ziele dieser Arbeit

Die Ziele dieser Arbeit umfassen ein im Vergleich zum Grundprojekt noch tiefer gehendes Verständnis für die Python-Bibliothek Evidently zu erlangen, um ihre Nutzung effektiver zu gestalten. Ebenso soll die Integration von Evidently in den maschinellen Lernlebenszyklus erlernt werden, um sicherzustellen, dass sie erfolgreich in realen Projekten eingesetzt werden kann. Dariüber hinaus strebt die Arbeit danach, die Fähigkeit zu entwickeln, eine ML-Monitoring-Architektur mit Evidently, Prefect, PostgreSQL und Grafana umzusetzen, um eine robuste Workflow-Struktur aufzubauen.

1.2 Gliederung

Die Gliederung dieser Arbeit umfasst zwei Hauptkapitel. Zunächst werden die Integrationsmöglichkeiten von "Evidently" untersucht. Dies schließt die Nutzung von Evidently in der Notebook-Umgebung für die Erstellung visueller Reports, die Integration mit Workflow-Management-Tools sowie die Verwendung von Tracking-Tools ein. Anschließend wird eine ML-Monitoring-Architektur vorgestellt, die auf Evidently, Prefect, PostgreSQL und Grafana basiert. Hierbei wird zuerst auf die Einrichtung des Experiments mit 4 Schritten eingegangen. Danach wird das Experiment in 3 Schritten durchgeführt. Darüber hinaus werden die Vorteile sowie die mögliche Herausforderungen und Empfehlungen für die Verwendung der Architektur diskutiert.

2 Integrationsmöglichkeiten von "Evidently"

Die Python-Bibliothek Evidently weist eine hohe Integrationsfähigkeit mit anderen Werkzeugen auf, wodurch eine nahtlose Integration in den Lebenszyklus des maschinellen Lernens gewährleistet wird.

2.1 Notebook-Umgebung

Die Python-Bibliothek Evidently kann unmittelbar in der Notebook-Umgebung genutzt werden, um visuelle HTML-Reports, JSON usw. zu erstellen und zu speichern. Evidently wurde für den Einsatz in Jupyter-Notebooks und Google Colab getestet. Es kann jedoch auch in anderen gehosteten Notebooks wie Kaggle Kernel, Databricks oder Deepnote Notebooks verwendet werden.

2.2 Workflow Management Tools

Zur Durchführung von Modellevaluierungen oder Datendrift-Analysen kann Evidently zusammen mit Workflow-Management-Tools wie Airflow oder Metaflow verwendet werden. In vielen Machine-Learning-Teams werden diese Art von Tools eingesetzt, um die verschiedenen Phasen des ML-Lebenszyklus zu orchestrieren, darunter Datenvorbereitung, Training, Deployment usw. In diesem Zusammenhang unterstützt Evidently durch die Berechnung einer Vielzahl von Metriken und statistischen Tests sowie die Erstellung visueller Reports.

2.3 Tracking Tools

Es kann festgestellt werden, dass die Verwendung von Tracking-Tools wie MLflow oder DVCLive in Kombination mit Evidently dazu geeignet ist, die Leistung der Modelle zu verfolgen. In diesem Fall wird Evidently verwendet, um die Metriken zu berechnen, anschließend wird die Berechnung in einem Python-Dictionary-Format erzeugt und schließlich MLflow/DVCLive verwendet, um die Ergebnisse zu protokollieren. Die erzeugten Metriken können schließlich über die entsprechende Benutzerschnittstelle abgerufen werden.

3 Eine ML-Monitoring-Architektur

3.1 Übersicht

In diesem Paper wird eine ML-Monitoring-Architektur implementiert, die auf Evidently, Prefect, PostgreSQL und Grafana basiert.

- Evidently wird, wie im Grundprojekt, verwendet, um die Datenqualität, die Modelldrift und die Modellqualität zu überprüfen.
- Prefect wird zusätzlich in diesem Projekt verwendet, um diese Überprüfungen zu orchestrieren und Metriken in die Datenbank zu schreiben.
- Die PostgreSQL-Datenbank fungiert als Speicherort für die definierten Monitoring-Metriken.
- Grafana stellt ein externes Dashboard zur Verfügung, das die Metriken in der Zeit visualisiert.

Um eine einfache Bereitstellung und Skalierbarkeit zu gewährleisten, erfolgt die Ausführung der Monitoring-Lösung in einer Docker-Umgebung.

3.2 Einrichtung

Monitoring-Cluster starten Die Einrichtung der Monitoring-Infrastruktur erfolgt mittels Docker Compose, wie in Abbildung 1 dargestellt. Ein Cluster wird mit den erforderlichen Diensten wie PostgreSQL und Grafana gestartet. Dieser Cluster ist für die Speicherung der Überwachungsmetriken und deren Visualisierung zuständig. Eine lokale Prefect Server UI muss allerdings manuell gestartet werden, indem ein entsprechender Befehl im Terminal ausgeführt wird, wie in Abbildung 2 dargestellt.

Die Clusterkomponenten werden in der docker-compose.yaml angegeben. Die Prefect UI ist verfügbar unter localhost:4200, während PostgreSQL unter localhost:5432 und Grafana Dashboards unter localhost:3000 erreichbar sind.

PostgreSQL-Datenbank-Tabellen erstellen Zur Speicherung der Überwachungsmetriken in der PostgreSQL-Datenbank ist die Erstellung der für die Speicherung der von Evidently generierten Metriken erforderlichen Tabellen erforderlich. Diese erfolgt durch Ausführung eines Python-Skripts, welches unter dem Verzeichnis "src/scripts/create_db.py" zu finden ist.

Daten und vtrainierte Modelle herunterladen Die Daten sowie die vorab trainierten ML-Modelle für dieses Projekt werden aus dem Internet heruntergeladen, wie in Abbildung 3 dargestellt. Das Beispiel basiert auf dem NYC-Taxi-Datensatz. Für die Datenladung sowie die Vorverarbeitung der Daten und das Trainieren eines simplen maschinellen Lernmodells stehen bereits vorbereitete Skripte zur Verfügung. Die entsprechenden Skripte sind wie folgt zu finden: "src/pipelines/load_data.py" und "src/pipelines/process_data.py" sowie "src/pipelines/train.py"

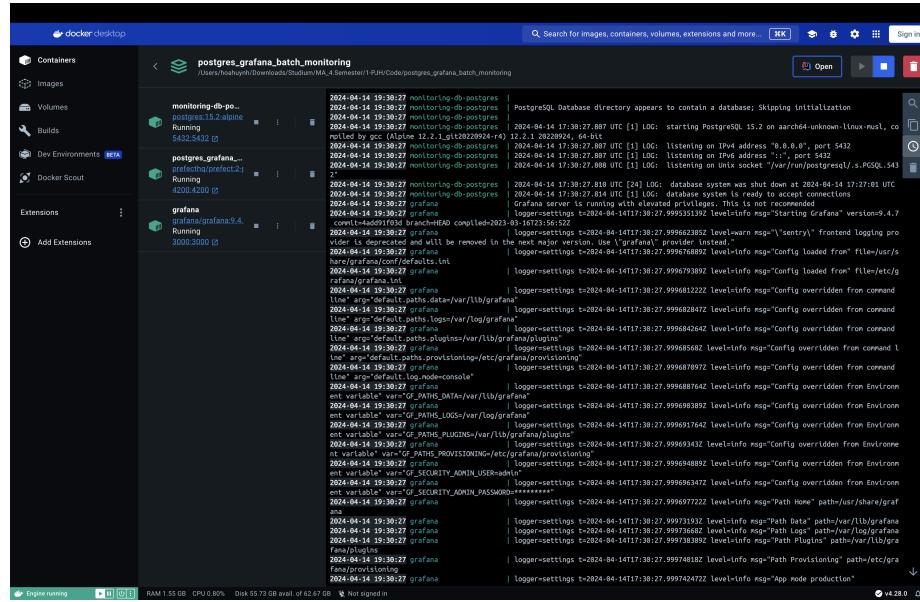


Abb. 1. Cluster mit PostgreSQL und Grafana gestartet.

```
Last login: Sun Apr 14 19:27:48 on ttys000
[(base) hoahuynh@Hoas-MBP postgres_grafana_batch_monitoring % source .venv/bin/activate
(.venv) (base) hoahuynh@Hoas-MBP postgres_grafana_batch_monitoring % prefect server start
[

Configure Prefect to communicate with the server with:

prefect config set PREFECT_API_URL=http://127.0.0.1:4200/api

View the API reference documentation at http://127.0.0.1:4200/docs

Check out the dashboard at http://127.0.0.1:4200
```

Abb. 2. Prefect Server UI gestartet.

```
Last login: Sun Apr 14 19:28:06 on ttys001
(base) hoahuynh@Hoas-MBP postgres_grafana_batch_monitoring % source .venv/bin/activate

(.venv) (base) hoahuynh@Hoas-MBP postgres_grafana_batch_monitoring % docker compose up -d
[+] Running 3/4
  ⋮ Network monitoring
  ✓ Container monitoring-db-postgres           Created      0.4s
  ✓ Container postgres_grafana_batch_monitoring-prefect-1 Started      0.3s
  ✓ Container grafana                          Started      0.3s
(.venv) (base) hoahuynh@Hoas-MBP postgres_grafana_batch_monitoring % python src/scripts/create_db.py
(.venv) (base) hoahuynh@Hoas-MBP postgres_grafana_batch_monitoring % python src/pipelines/load_data.py
[python src/pipelines/process_data.py
python src/pipelines/train.py
Download files:
green_tripdata_2021-01.parquet: 100%|██████████| 1.33M/1.33M [00:00<00:00, 2.04MB/s]
green_tripdata_2021-02.parquet: 100%|██████████| 1.15M/1.15M [00:00<00:00, 2.34MB/s]
[Download complete.
Load train data
Generate UID
Save data
Generate UID
Save data
Load train data
Train model
Get predictions for validation
Calculate validation metrics: MAE
3.321205378410284
6.021442730731626
Calculate validation metrics: MAPE
0.3904355682881315
0.5586175690766146
Save the model
(.venv) (base) hoahuynh@Hoas-MBP postgres_grafana_batch_monitoring % python src/pipelines/prepare_reference_data.py
Load data
Load model
Predictions generation
[Save reference dataset
(.venv) (base) hoahuynh@Hoas-MBP postgres_grafana_batch_monitoring % ]
```

Abb. 3. Daten und Modelle heruntergeladen.

Referenzdaten für die Überwachung vorbereiten Nachdem die Daten heruntergeladen wurden, erfolgt nun eine Aufteilung in zwei Teildatensätze mittels des Skripts unter src/pipelines/prepare_reference_data.py. Für die Erstellung von Überwachungsreports mit Evidently werden in der Regel zwei Datensätze benötigt. Der Referenzdatensatz dient dabei als Basis für den Vergleich, während der aktuelle Datensatz die neueste Statistik zum Vergleich mit der Referenz enthält.

Nach Abschluss der Einrichtung ist eine funktionierende Evidently-Integration mit Prefect, PostgreSQL und Grafana vorhanden, wie in den Abbildungen 4 und 5 dargestellt.

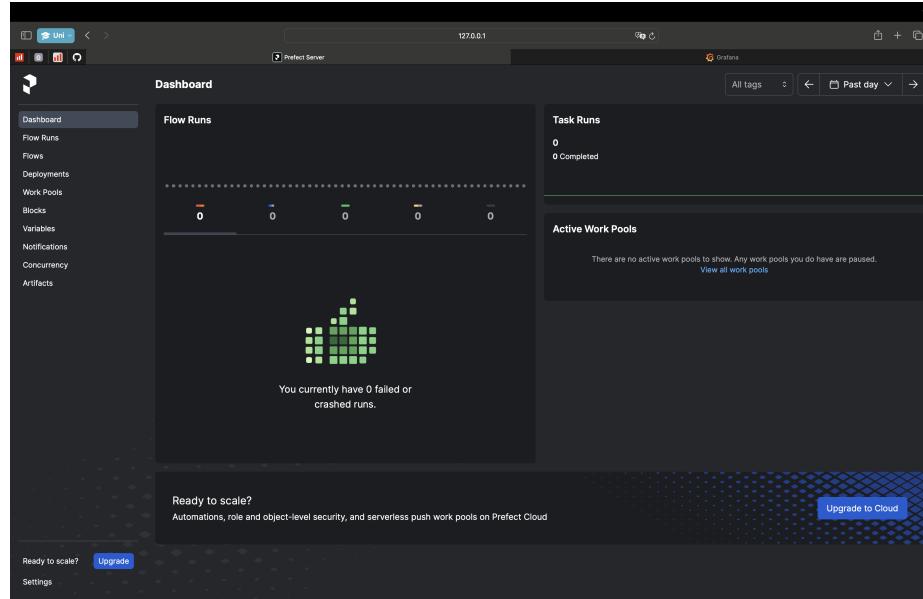


Abb. 4. Prefect UI

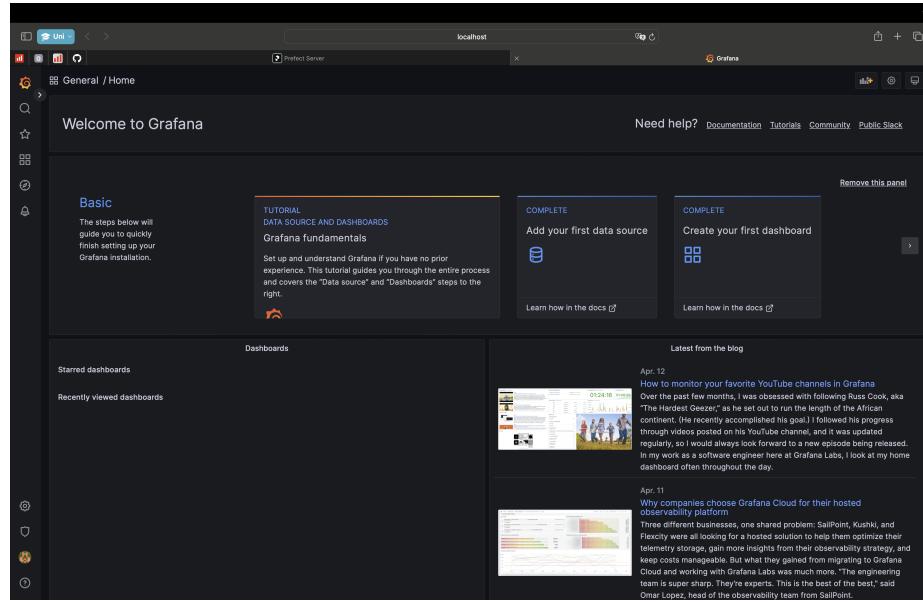


Abb. 5. Grafana UI

3.3 Experiment

Pipelines ausführen Im Folgenden werden die folgenden Pipelines ausgeführt, wie in Abbildung 6 dargestellt.

- Die Pipeline „src/pipelines/predict.py“ führt die Inferenz durch, indem sie das trainierte Modell auf die Eingabedaten anwendet.
 - Die Pipeline „src/pipelines/monitor_data.py“ überwacht die Qualität der Eingabedaten, indem sie sie mit den Referenzdaten vergleicht.
 - Die Pipeline „src/pipelines/monitor_model.py“ evaluiert die Qualität der Modellvorhersagen. Diese Pipeline benötigt Ground-Truth-Daten.

Es wird angenommen, dass die Labels mit einer Verzögerung eintreffen und für den vorherigen Zeitraum verfügbar sind. Im Rahmen dieses Projekts dienen die Daten des Monats Januar 2021 als Referenz. Die genannten Daten dienen als Grundlage für die Erstellung von Überwachungsberichten für aufeinanderfolgende Zeiträume.

Abb. 6. Pipelines ausgeführt.

Pipelines in der Prefect UI anzeigen Die Prefect UI kann nun, wie in Abbildung 7 dargestellt, über localhost:4200 in einem Webbrowser aufgerufen werden, um die ausgeführten Pipelines und ihren aktuellen Status anzuzeigen.

Überwachungsmetriken in der Grafana untersuchen Die Öffnung der mehreren Grafana-Monitoring-Dashboards kann nun, wie in Abbildung 8 dargestellt, durch Aufrufen von „localhost:3000“ in einem Webbrower erfolgen. Das Login-Konto ist wie in der Grafana-Dokumentation mit dem Kontonamen „admin“ und dem Passwort „admin“ definiert. Die Dashboards umfassen nun Metriken zu den Aspekten Datenqualität, Zieldrift und Modellleistung.

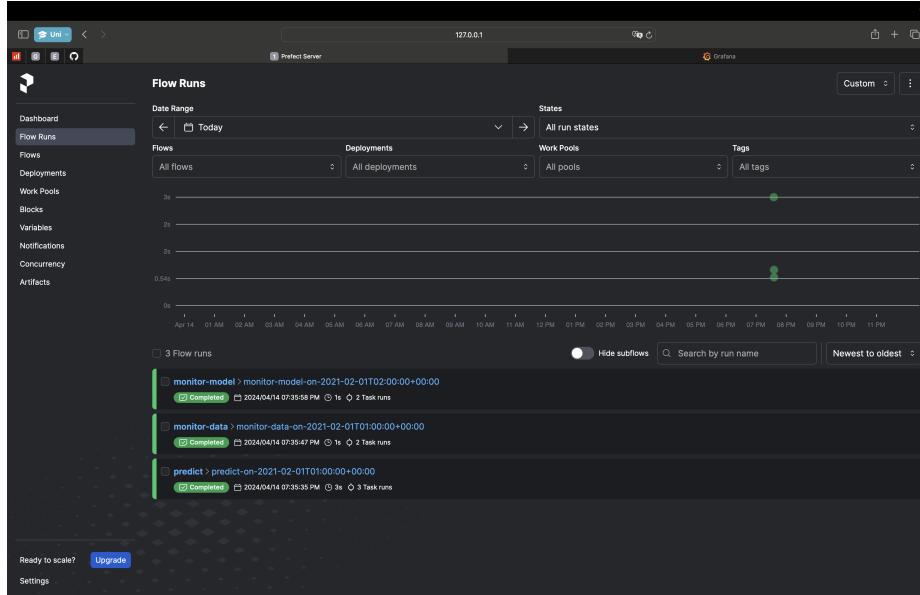


Abb. 7. Pipelines in der Prefect UI

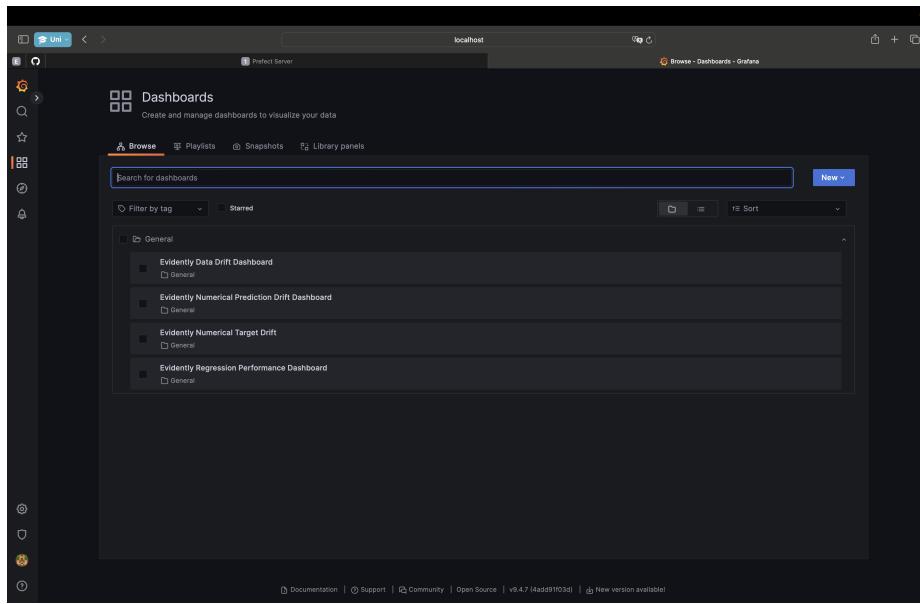


Abb. 8. Grafana Dashboards

3.4 Diskussion der Monitoring-Dashboards

Evidently Data Drift Dashboard (Abbildung 9) visualisiert die Drift der Daten. Die oberen Widgets geben Aufschluss darüber, ob für jeden Zeitraum eine Datendrift auftritt und wie lange diese andauert. Die unteren linken Widgets zeigen den Anteil der gedrifteten Spalten an der Gesamtzahl der Datensatzspalten für jeden Zeitraum. Auf diese Weise lässt sich feststellen, wann und wie stark die Datendrift ist.

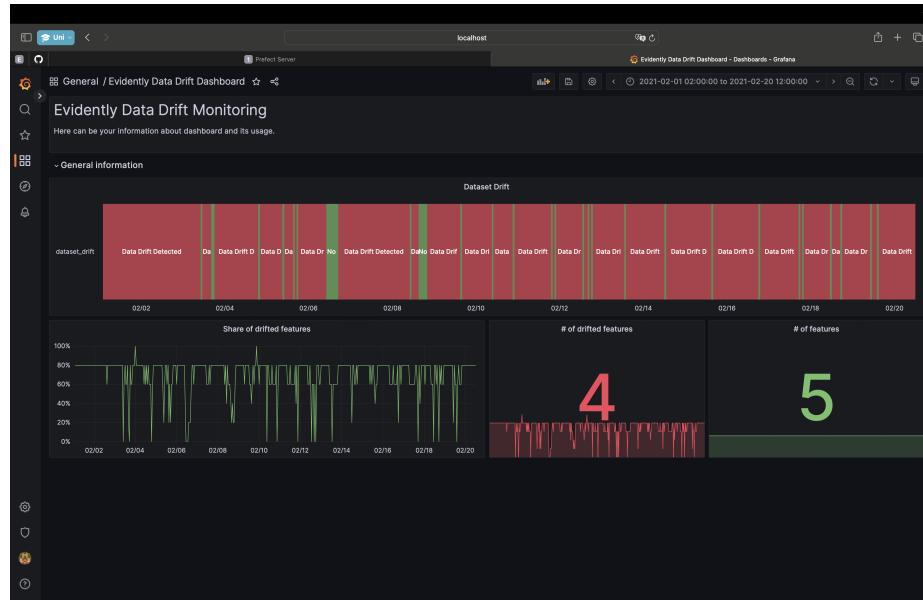


Abb. 9. Evidently Data Drift Dashboard

Evidently Numerical Prediction Drift Dashboard (Abbildung 10) bietet eine Übersicht über die Drift der Verteilung der Modellvorhersagen. Die Widgets oben links zeigen den Namen der verwendeten Drifterkennungsmethode (hier die Wasserstein-Distanz) sowie die zugehörigen Schwellenwerte an. Die beiden unteren Widgets geben jeweils die Historie der Driftprüfungen und die zugehörige Drift-Score für jeden Zeitraum wieder. Auf diese Weise lässt sich ermitteln, zu welchen Zeitpunkten und mit welcher Stärke eine Drift auftrat. Zudem kann entschieden werden, wann ein Eingriff erforderlich ist.

Evidently Numerical Target Drift (Abbildung 11) weist eine ähnliche Struktur wie das Numerical Prediction Drift Dashboard auf. Im Gegensatz zu diesem bietet es jedoch Einblicke in die Drift der Verteilung des Modellziels. Die Nutzung dieses Dashboards ist daher ähnlich wie die des vorherigen Dashboards.

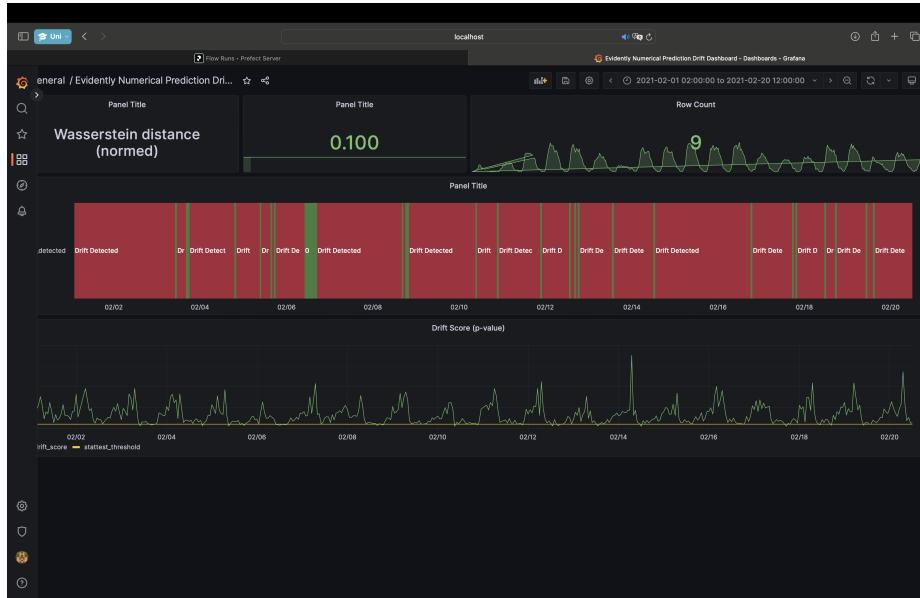


Abb. 10. Evidently Numerical Prediction Drift Dashboard

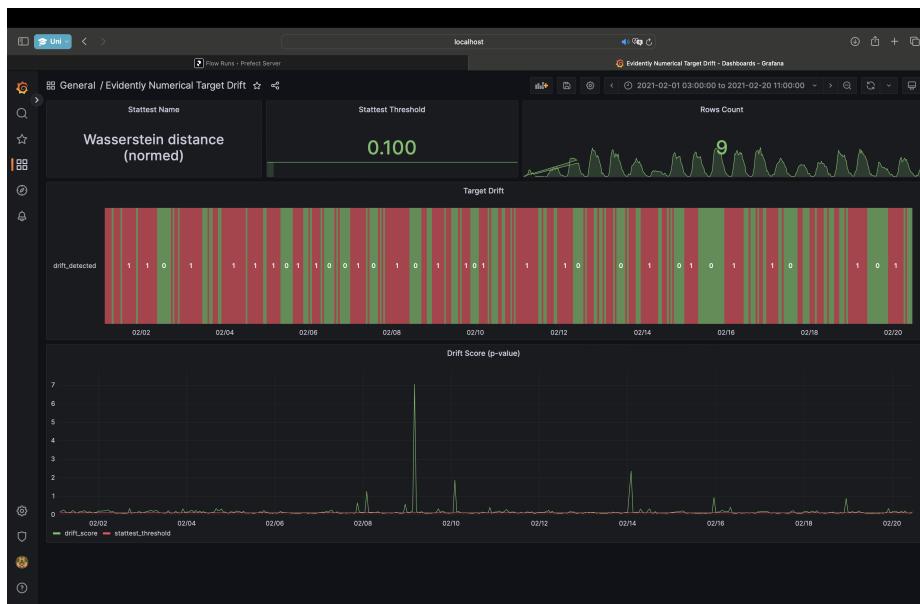


Abb. 11. Evidently Numerical Target Drift

Evidently Regression Performance Dashboard (Abbildung 12) visualisiert die Drift der Modellleistung. Die drei Widgets oben zeigen die durchschnittlichen Fehler (in diesem Fall ME, MAE und MAPE). Die beiden unteren Widgets zeigen die Aggregation dieser Fehler für jeden Zeitraum. Dies veranschaulicht die Performance des Modells zu einem bestimmten Zeitpunkt.

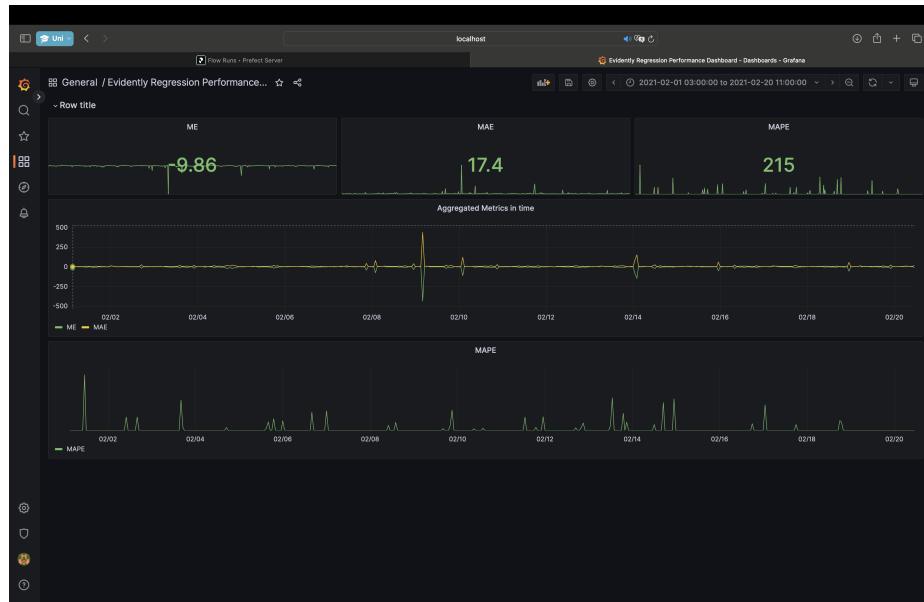


Abb. 12. Evidently Regression Performance Dashboard

3.5 Bewertung

Vorteile

- Die hier beschriebene Überwachungsarchitektur eignet sich sowohl für Batch- als auch für Echtzeit-ML-Systeme. Bei der Batch-Inferenz kann das beschriebene Experiment direkt durchgeführt werden. Bei Echtzeitsystemen besteht die Möglichkeit, Modelleingaben und Vorhersagen in der Datenbank zu protokollieren und dann die Vorhersageprotokolle in einem bestimmten Intervall aus der Datenbank zu lesen.
- Asynchrone Berechnung von Metriken: Die Berechnung von Metriken erfolgt separat von der Modellbereitstellung, sodass sich diese nicht auf die Latenzzeit bei der Modellbereitstellung auswirkt.
- Anpassungsfähig: Es besteht die Möglichkeit, bestimmte Komponenten auszutauschen. Dies kann beispielsweise durch den Einsatz eines alternativen Workflow-Orchestrator wie Airflow oder durch den Ersatz von Grafana durch ein anderes BI-Dashboard erfolgen.

Nachteile Es besteht die Möglichkeit, dass das Projekt zu anspruchsvoll ist. Die Verwendung dieser oder einer ähnlichen Architektur wird empfohlen, sofern bereits eines oder zwei der genannten Tools als Teil des Workflows verwendet werden.