

Model Drift Überwachung mit Evidently ML Monitoring: Eine Anwendung am Beispiel des Census-Einkommensdatensatzes

Nhut Hoa Huynh

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

Zusammenfassung. Dieses Paper präsentiert ein Experiment mit dem Census-Einkommensdatensatz, das die Funktionalität von “Evidently ML Monitoring” demonstriert. Ein Python-Skript integriert ein neues ML-Modell, erfasst Metriken wie Datendrift und fehlende Werte, visualisiert sie auf dem Monitoring-Dashboard und generiert tägliche Reports und Tests im Evidently UI. Die Ergebnisse ermöglichen eine kontinuierliche Überwachung von Modellen und Metriken. “Evidently ML Monitoring” bietet somit eine effektive Lösung zur frühzeitigen Identifizierung von Drifts und zur proaktiven Sicherung der Modellqualität.

Schlüsselwörter: Modelldrift · Evidently Reports · Evidently ML Monitoring · Monitoring-Dashboard · Datenqualität · Modellleistung · Echtzeit-Drifts · Visualisierung · Workflow-Management · Kontinuierliche Überwachung.

1 Einleitung

1.1 Ziele dieser Arbeit

Das Paper “Modelldrift” aus dem Modul “Forschungswerkstatt 1” widmet sich der eingehenden Untersuchung verschiedener Modelldrift-Arten und deren potenziellen Auslöser. Außerdem werden die wesentlichen Methoden zur Erkennung von Modelldrift sowie die erforderlichen Maßnahmen zum Umgang mit Modelldrift beleuchtet. Die Anwendung von “Evidently Reports” wurde für die Durchführung von vier Fallbeispielen genutzt, um Echtzeit-Drifts zu veranschaulichen. Dieses Paper erweitert die Analyse auf das “Evidently ML Monitoring”, bei dem ein lokal gehostetes Dashboard vorgestellt wird, das die Visualisierung der Datenqualität und der Modellleistung über die Zeit ermöglicht.

1.2 Gliederung

Die Arbeit gliedert sich in folgende Kapitel: Kapitel 2 erläutert die Grundkonzepte von “Evidently Reports” im Vergleich zu “Evidently ML Monitoring”, hebt ihre Unterschiede hervor und beschreibt ihre jeweiligen Anwendungen. Kapitel 3 widmet sich der Darstellung des mit dem ML Monitoring durchgeführten Experiments, während Kapitel 4 die Schlussfolgerungen zusammenfasst.

2 Grundkonzepte von “Evidently Reports” und “Evidently ML Monitoring”

Evidently verfolgt einen modularen Ansatz mit drei Komponenten: Reports, Test Suites und ein Monitoring-Dashboard. Diese Komponenten decken unterschiedliche Anwendungsszenarien ab, von ad-hoc-Analysen (Reports) bis hin zu automatisierten Pipeline-Tests (Test Suites) und kontinuierlichem Monitoring (Monitoring-Dashboard).

2.1 Evidently Reports

Reports ermitteln vielfältige Metriken und bieten umfassende interaktive Visualisierungen. Sie sind besonders nützlich für explorative Analysen, da sie bei der visuellen Bewertung von Daten oder der Modellleistung unterstützen. Beispielsweise sind sie hilfreich bei der explorativen Datenanalyse, der Bewertung von Modellen auf dem Trainingsdatensatz, der Fehlerbehebung bei Qualitätsverlusten des Modells und dem Vergleich verschiedener Modelle. Darüber hinaus können Reports zur Dokumentation dienen, beispielsweise zur Erstellung visueller HTML-Reports.

2.2 Evidently ML-Monitoring-Dashboard

Das ML-Monitoring-Dashboard ist erst ab Version 0.4.0 verfügbar. Es bietet die Möglichkeit, ein eigenes ML-Monitoring-Dashboard zu hosten, um Metriken und Testergebnisse im Verlauf der Zeit zu visualisieren. Diese Funktion baut auf Reports und Test Suites auf. Mit dem Dashboard kann man sämtliche verfügbaren Metriken in Evidently überwachen, angefangen bei der Anzahl der Null-Werte in den Daten bis hin zur Analyse der Textstimmung und zur Verfolgung von Einbettungsdrift. Der vorrangige Anwendungszweck des ML-Monitoring-Dashboards besteht in der kontinuierlichen Überwachung. Es bietet ein Echtzeit-Dashboard, um all Modelle und Metriken im Verlauf der Zeit im Blick zu behalten.

3 Experiment mit “Evidently ML Monitoring”

3.1 Datensatz

Zur Durchführung des Experiments in diesem Paper wird ein Datensatz gewählt, der häufig in früheren Forschungsarbeiten verwendet wurde: der Census- Einkommensdatensatz. Die Census-Einkommensdaten (CI) enthalten Informationen, die zur Vorhersage des Einkommens von Einzelpersonen verwendet werden. Sie umfassen 32.561 Datensätze und 12 Merkmale. Diese Merkmale heißen auf Englisch “age, work class, education, marital status, occupation, relationship, race, sex, capital gain, capital loss, hours per week, country”. Die Klassen sind niedriges Einkommen (weniger oder gleich 50.000 Dollar) oder hohes Einkommen (mehr als 50.000 Dollar).

3.2 Experiment

Der Datensatz wird zuerst mit einem Jupyter Notebook exploriert. Dann wurde er in 2 Subdatensätze aufgeteilt. Der Teil “adult-ref” wird später als Baseline für die Drifterkennung verwendet. Den Rest “adult-cur” verwenden wir, um Batch-Inferenz zu imitieren.

Dann wird ein Python-Skript erstellt, das ein neues ML-Modell zur Überwachung hinzufügt. Wir ahmen die Batch-Modellinferenz nach, um Logs zu sammeln, und wir erstellen Panels, um sie zu visualisieren.

In dem Skript wird ein Workspace definiert, der der Ordner ist, in dem Evidently Daten speichert. Er wird in dem Verzeichnis erstellt, von dem aus wir das Skript starten. Wir können auch den Namen und die Beschreibung des Projekts festlegen, die auf der Benutzeroberfläche angezeigt werden.

Ein wichtiger Schritt im Skript ist die Erfassung von Daten und Modellmetriken in Logs. Wir verwenden in diesem Schritt Evidently Reports und Test Suites. Wir definieren ein Report- und ein Test-Suite-Objekt und listen die Metriken oder Tests auf, die einbezogen werden sollen. Für die Erfassung der Daten verwenden wir dieselbe Syntax wie bei der Ausführung von Evidently Reports und Test Suites in einem Jupyter-Notebook.

```

32
33 #https://docs.evidentlyai.com/reference/all-metrics#metric-presets
34 def create_report(i: int):
35     data_drift_report = Report(
36         metrics=[
37             DatasetDriftMetric(),
38             DatasetMissingValuesMetric(),
39             ColumnDriftMetric(column_name="age", stattest="wasserstein"),
40             ColumnSummaryMetric(column_name="age"),
41             ColumnDriftMetric(column_name="education-num", stattest="wasserstein"),
42             ColumnSummaryMetric(column_name="education-num"),
43         ],
44         timestamp=datetime.datetime.now() + datetime.timedelta(days=i),
45     )
46
47     data_drift_report.run(reference_data=adult_ref, current_data=adult_cur.iloc[100 * i : 100 * (i + 1), :])
48     return data_drift_report
49
50 #https://docs.evidentlyai.com/reference/all-tests
51 def create_test_suite(i: int):
52     data_drift_test_suite = TestSuite(
53         tests=[DataDriftTestPreset()],
54         timestamp=datetime.datetime.now() + datetime.timedelta(days=i),
55     )
56
57     data_drift_test_suite.run(reference_data=adult_ref, current_data=adult_cur.iloc[100 * i : 100 * (i + 1), :])
58     return data_drift_test_suite

```

Abb. 1. Report- und Test-Suite-Objekt definieren

Wir können andere Metriken oder Tests auswählen, z.B. solche für Datenqualität, Integrität oder Datendrift, um den Inhalt des Monitorings zu definieren. Eine vollständige Liste aller Metriken und aller Tests und ihrer Parameter findet sich unter <https://docs.evidentlyai.com/reference/all-metrics> und <https://docs.evidentlyai.com/reference/all-tests>. Wir können außerdem zusätzliche Parameter an die einzelnen Tests und Metriken übergeben.

In diesem Experiment werden folgende Metriken ausgewählt (siehe Abb. 1):

- “DatasetDriftMetric” zur Berechnung der Anzahl und des Anteils der drifted Features.
- “DatasetMissingValuesMetric” zur Berechnung der Anzahl und des Anteils fehlender Werte pro Spalte im Datensatz.
- “ColumnDriftMetric” zur Berechnung der Datendrift für eine bestimmte Spalte und zur Visualisierung ihrer Verteilung.
- “ColumnSummaryMetric” zur Berechnung verschiedener deskriptiver Statistiken für die Spalte.

Die ausgewählten Spalten für die letzten beiden Metriken sind “age” und “education”. Der ausgewählte Test im Experiment ist der Standard “DataDriftTest-Preset”.

Anschließend imitieren wir die Batch-Inferenz, d. h. wir führen Berechnungen durch, als ob wir Daten für i Tage (hier setzen wir sie auf 5) erfasst hätten, wobei jedes Mal 100 Beobachtungen gemacht werden. Wir übergeben “adult-ref” auch den Evidently Report, damit dieser als Vergleichsbasis für die Erkennung von Verteilungsdrift verwendet werden kann.

Nachdem die Metriken in Reports und Test Suites erfasst wurden, können wir sie nun mit Hilfe von Panels im Zeitverlauf visualisieren. Jedes Projekt kann mehrere Panels enthalten, die auf dem Monitoring-Dashboard angezeigt werden. In einem Panel können Metriken als Zähler, Zeitseriendiagramme, Balkendiagramme und Streudiagramme dargestellt werden. Eine vollständige Anleitung für die Gestaltung dieser Panels findet sich unter <https://docs.evidentlyai.com/user-guide/monitoring/design-dashboard>. In diesem Experiment werden 6 Beispiele von Panels untersucht, die jeweils einer der zuvor definierten Metriken entsprechen. Einige Beispiele sind in Abb. 2 dargestellt.

```
project.dashboard.add_panel(
    DashboardPanelCounter(
        title="Fehlende Werte im Datensatz",
        filter=ReportFilter(metadata_values={}, tag_values=[]),
        value=PanelValue(
            metric_id="DatasetMissingValuesMetric",
            field_path=DatasetMissingValuesMetric.fields.current.number_of_rows,
            legend="count",
        ),
        text="count",
        agg=CounterAgg.SUM,
        size=1,
    )
)
project.dashboard.add_panel(
    DashboardPanelCounter(
        title="Anteil der Drifted Features",
        filter=ReportFilter(metadata_values={}, tag_values=[]),
        value=PanelValue(
            metric_id="DatasetDriftMetric",
            field_path="share_of_drifted_columns",
            legend="share",
        ),
        text="share",
        agg=CounterAgg.LAST,
        size=1,
    )
)
```

Abb. 2. Panels erstellen

Schließlich können wir nun das Projekt und den Workspace erstellen (Abb. 3) und die JSON-Snapshots generieren. Der Snapshot ist eine JSON-Version des Reports und der Test-Suite, die zuvor definiert wurden. Er enthält Zusammenfassungen der erfassten Daten und Modellmetriken. Wir müssen die Snapshots in einem Verzeichnis unter dem entsprechenden Workspace-Namen speichern. Auf diese Weise ist Evidently UI in der Lage, die Metriken zu analysieren und sie in den Monitoring-Panels zu visualisieren.

```

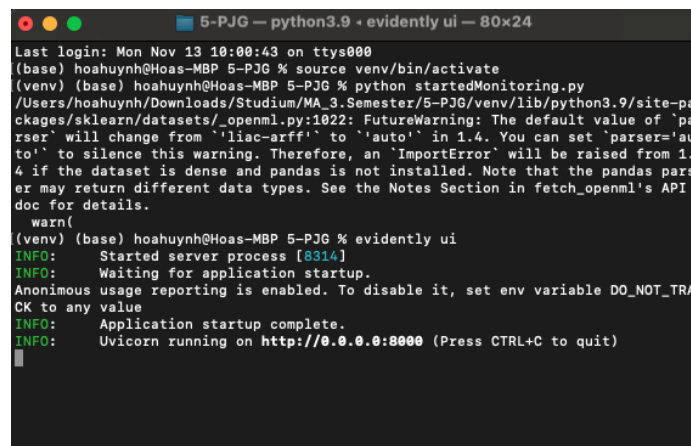
1 def create_demo_project(workspace: str):
2     ws = Workspace.create(workspace)
3     project = create_project(ws)
4
5     for i in range(0, 5):
6         report = create_report(i=i)
7         ws.add_report(project.id, report)
8
9         test_suite = create_test_suite(i=i)
10        ws.add_test_suite(project.id, test_suite)
11
12 if __name__ == "__main__":
13     create_demo_project(WORKSPACE)

```

Abb. 3. Projekt und Workspace erstellen

Um das gesamte System in Betrieb zu nehmen, müssen folgende Schritte durchgeführt werden (Abb. 4):

- Eine virtuelle Umgebung erstellen und diese aktivieren.
- Den Befehl zum Erstellen des neuen Beispielprojekts ausführen, wie im obigen Skript definiert.
- Starten der Benutzeroberfläche, die das definierte Projekt enthalten wird.



```

5-PJG — python3.9 • evidently ui — 80x24
Last login: Mon Nov 13 10:00:43 on ttys000
(base) hoahuynh@Hoas-MBP 5-PJG % source venv/bin/activate
(venv) (base) hoahuynh@Hoas-MBP 5-PJG % python startedMonitoring.py
/Users/hoahuynh/Downloads/Studium/MA_3.Semester/5-PJG/venv/lib/python3.9/site-packages/sklearn/datasets/_openml.py:1022: FutureWarning: The default value of 'parser' will change from 'liac-arff' to 'auto' in 1.4. You can set 'parser='auto' to silence this warning. Therefore, an 'ImportError' will be raised from 1.4 if the dataset is dense and pandas is not installed. Note that the pandas parser may return different data types. See the Notes Section in fetch_openml's API doc for details.
  warn(
(venv) (base) hoahuynh@Hoas-MBP 5-PJG % evidently ui
INFO: Started server process [8314]
INFO: Waiting for application startup.
Anonymous usage reporting is enabled. To disable it, set env variable DO_NOT_TRACK to any value
INFO: Application startup complete.
INFO: Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)

```

Abb. 4. Skript und Evidently UI Service ausführen

Jetzt können wir das Projekt durch Zugriff auf den Evidently UI-Dienst im Browser unter localhost:8000 anzeigen (Abb. 5).

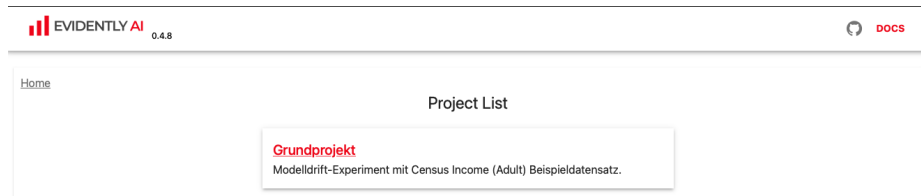


Abb. 5. Neues Projekt anzeigen

3.3 Ergebnisse

Abb. 6 zeigt das Monitoring-Dashboard mit sechs Panels, die wir im Skript entworfen haben.

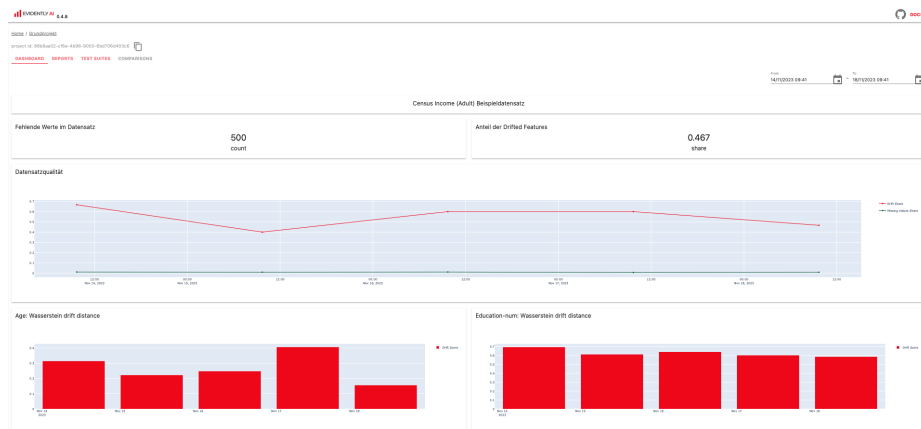


Abb. 6. Dashboard

Auf dem Reiter “Reports” (siehe Abb. 7) finden wir alle täglichen Reports, die die im Skript ausgewählten Metriken anzeigen (siehe Abb. 8, Abb. 9, Abb. 10).

EVIDENTLY ML 0.4.0 DOCS

Home / Dashboard / Reports

Project id: 808ba32-476a-4098-9050-d0c7084030d6

DASHBOARD **REPORTS** TEST SUITES COMPARISONS

Filter by Tags

Report ID	Tags	Timestamp	Actions
9398990-3d70-41e5-9a5d-08f3a6a084e8		2023-11-16T09:41	VIEW DOWNLOAD
4383f6c8-6a86-4f70-880a-3276c34a886a		2023-11-16T09:41	VIEW DOWNLOAD
992829e-9a29-41a3-ef1b-62a77634830a		2023-11-16T09:41	VIEW DOWNLOAD
471c47c7-41ef-49aa-84cc-6e0d4a2f8751		2023-11-17T09:41	VIEW DOWNLOAD
95860735-d0f9-4269-91a2-6583370c0a5c		2023-11-18T09:41	VIEW DOWNLOAD

Abb. 7. Reports

EVIDENTLY ML 0.4.0 DOCS

Home / Dashboard / Reports / 9398990-3d70-41e5-9a5d-08f3a6a084e8

Project id: 808ba32-476a-4098-9050-d0c7084030d6

DASHBOARD **REPORTS** TEST SUITES COMPARISONS

Dataset Drift
Dataset Drift is detected. Dataset drift detection threshold is 0.5

15 Columns	9 Drifted Columns	0.6 Share of Drifted Columns
16 (1067%) Missing values (Current data)	2092 (0.985%) Missing values (Reference data)	

Current dataset
Table HISTOGRAM

Value	Count
workplaces	2
workplaces	2
native-country	2
sex	0
fringe	0
education	0
education-num	0
marital-status	0
relationship	0
race	0
sex	0
capital-gain	0
capital-loss	0
hours-per-week	0
status	0

Reference dataset
Table HISTOGRAM

Value	Count
workplaces	902
workplaces	898
native-country	294
sex	0
fringe	0
education	0
education-num	0
marital-status	0
relationship	0
race	0

Abb. 8. Ein Report (1)

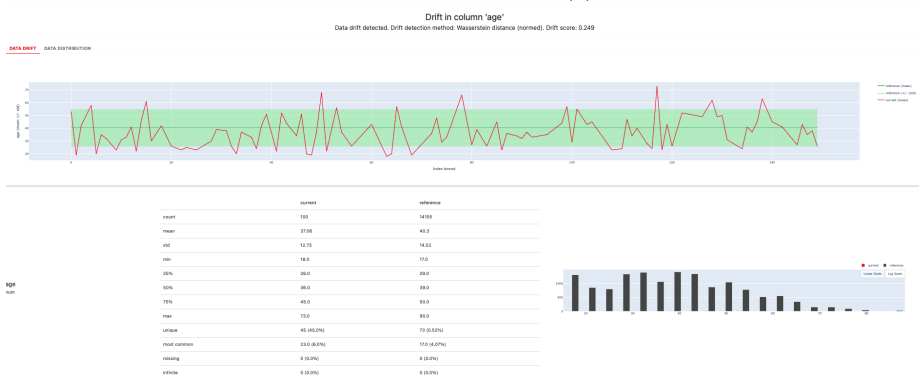


Abb. 9. Ein Report (2)

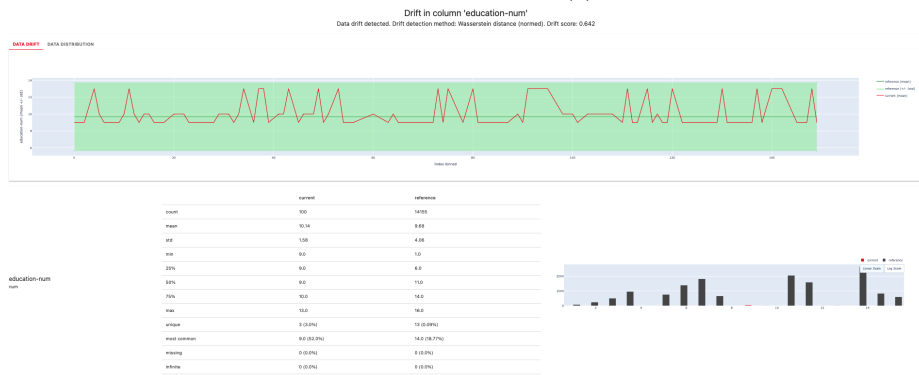


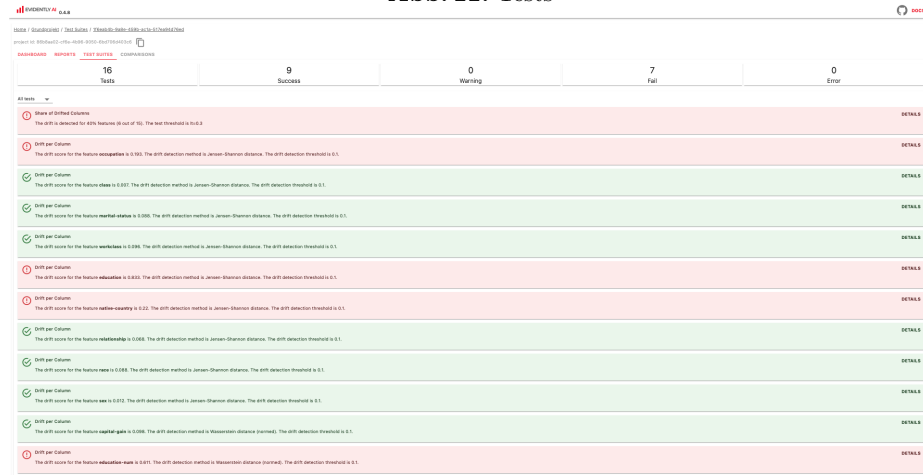
Abb. 10. Ein Report (3)

In ähnlicher Weise finden wir auf dem Reiter “Test Suites” (siehe Abb. 11) die täglichen Tests, die den vordefinierten Test enthalten (siehe Abb. 12).



Test Suite ID	Test	Timestamp	Actions
1544545-4854-4434-4744444744		2022-11-10T08:45	VIEW DOWNLOAD
4444747-5434-4444-4444-4444444444		2022-11-10T08:45	VIEW DOWNLOAD
4444444-4444-4444-4444-4444444444		2022-11-10T08:45	VIEW DOWNLOAD
4444444-4444-4444-4444-4444444444		2022-11-10T08:45	VIEW DOWNLOAD
4444444-4444-4444-4444-4444444444		2022-11-10T08:45	VIEW DOWNLOAD

Abb. 11. Tests



Test Name	Status	Details
Share of Undefined Columns	Fail	The drift is detected for share of undefined columns (0.0 out of 10). The test threshold is 0.0.
DRIF per Column	Fail	The drift score for the feature occupation is 0.001. The drift detection method is Jensen-Shannon distance. The drift detection threshold is 0.1.
DRIF per Column	Pass	The drift score for the feature class is 0.000. The drift detection method is Jensen-Shannon distance. The drift detection threshold is 0.1.
DRIF per Column	Pass	The drift score for the feature marital status is 0.000. The drift detection method is Jensen-Shannon distance. The drift detection threshold is 0.1.
DRIF per Column	Pass	The drift score for the feature workclass is 0.000. The drift detection method is Jensen-Shannon distance. The drift detection threshold is 0.1.
DRIF per Column	Fail	The drift score for the feature education is 0.001. The drift detection method is Jensen-Shannon distance. The drift detection threshold is 0.1.
DRIF per Column	Fail	The drift score for the feature native country is 0.001. The drift detection method is Jensen-Shannon distance. The drift detection threshold is 0.1.
DRIF per Column	Pass	The drift score for the feature relationship is 0.000. The drift detection method is Jensen-Shannon distance. The drift detection threshold is 0.1.
DRIF per Column	Pass	The drift score for the feature race is 0.000. The drift detection method is Jensen-Shannon distance. The drift detection threshold is 0.1.
DRIF per Column	Pass	The drift score for the feature sex is 0.000. The drift detection method is Jensen-Shannon distance. The drift detection threshold is 0.1.
DRIF per Column	Pass	The drift score for the feature capital gain is 0.000. The drift detection method is Wasserstein distance (normal). The drift detection threshold is 0.1.
DRIF per Column	Fail	The drift score for the feature education num is 0.001. The drift detection method is Wasserstein distance (normal). The drift detection threshold is 0.1.

Abb. 12. Ein Test

4 Fazit und Empfehlungen für künftige Forschungen

In diesem Experiment wurde der Census-Einkommensdatensatz verwendet, um die Funktionalität von “Evidently ML Monitoring” zu demonstrieren. Ein Python-Skript wurde erstellt, um ein neues ML-Modell zur Überwachung hinzuzufügen und Batch-Inferenz zu simulieren. Das Skript nutzt die Evidently-Bibliothek, um verschiedene Metriken zu erfassen, darunter Datendrift, fehlende Werte und deskriptive Statistiken für ausgewählte Spalten. Diese Metriken wurden dann in Panels auf dem Monitoring-Dashboard visualisiert. Das Dashboard ermöglicht die kontinuierliche Überwachung von Modellen und Metriken im Zeitverlauf. Die Ergebnisse wurden in Form von täglichen Reports und Tests aufbereitet, die im Evidently UI angezeigt werden. Das Dashboard sowie die Reports und Tests bieten eine intuitive Möglichkeit, relevante Metriken zu überwachen und potenzielle Probleme bei der Modelleistung frühzeitig zu erkennen.

Insgesamt ermöglicht “Evidently ML Monitoring” eine effektive Überwachung von Modellen, wodurch Datenwissenschaftler in der Lage sind, die Leistung ihrer Modelle kontinuierlich zu überwachen, Drifts zu identifizieren und proaktiv Maßnahmen zu ergreifen, um die Modellqualität aufrechtzuerhalten.

Ein entscheidender Schritt für die Integration von “Evidently ML Monitoring” in realen Projekten besteht allerdings in der Implementierung eines effektiven Workflow-Managements. Zukünftige Forschungen könnten sich darauf konzentrieren, optimierte und anpassbare Workflows zu gestalten, um eine reibungslose Integration und regelmäßige Berechnung von Evidently-Snapshots zu ermöglichen.