h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Automatisierter Machine Learning Lifecycle mit multivariater Zeitreihenanalyse

Adrian Füller
Referent: Prof. Dr. Timo Schürg | Korreferentin: Prof. Dr. Jutta Groos
Hochschule Darmstadt – Fachbereich Mathematik und Naturwissenschaften & Informatik

Motivation

Künstliche Intelligenz wird durch viele neuen Methodiken und Modelle immer aufwändiger und die Komplexität des vorherigen Datenverarbeitungsprozesses und der Modellentwicklung steigt, weil detailliertere Schritte zu mehr manuellen Aufgaben und Abhängigkeiten führen, die den Prozess verlangsamen. Als Reaktion darauf sind neue Ansätze zur Automatisierung und Vereinfachung von Machine Learning Workflows entstanden, wodurch der ML-Lifecycle entstanden ist. Der MLLC beschreibt den gesamten Prozess von der Datenanbindung bis zur Bereitstellung der Modelle. Auch für die Zeitreihenanalyse sind neben den statistischen und naiven Methoden neue Ansätze mit MLP und Transformern entwickelt worden. Insbesondere mit Transformern gibt es vielversprechende Ideen, welche die Prognosen von Zeitreihen verbessert vorhersagen sollen.
Das Ziel dieser Thesis ist es, den MLLC so weit wie möglich zu automatisieren und die Ergebnisse reproduzierbar zu machen. Für die Vorhersage werden Datensätze mit einer täglichen und einer halbstündigen Frequenz vorhergesagt. Der Vorhersage-Horizon beträgt 48 halbe Stunden (ein Tag) bzw. 60 Tage (zwei Monate).

Konzeption

Es werden die zwei MLLC ‚Entwicklung' und ‚Produktion' aufgebaut. Für den Experimente-MLLC ist der Start-Trigger immer eine manuelle Durchführung über Skripte oder ein manuelles Ausführen eines Workflows in Airflow. Der Start-Trigger des Produktions- MLLC ist entweder basierend auf einer Codeänderung in Gitlab, was für DevOps verwendet wird, oder durch einen täglichen Aufruf über Airflow, was für den Workflow verantwortlich ist. Bei Zweiteres werden aus den externen Datenquellen die neuen Daten gefetcht und in den Data Storage MinIO gespeichert. Nach der Data Pipeline werden die gewählten Modelle aus den Bereichen Naive, Statistisch, MLP und Transformer trainiert und evaluiert und in MLFlow hinterlegt. Abschließend werden auf Basis der gewählten Metriken ein oder mehrere Modelle deployt.

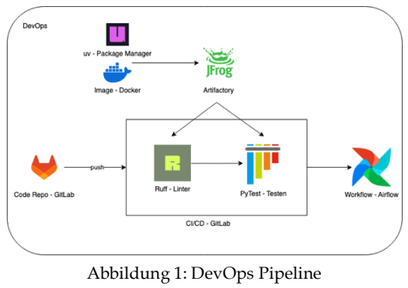

Abbildung 1: DevOps Pipeline

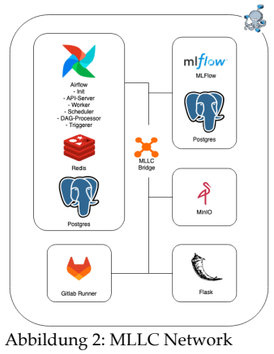

Abbildung 2: MLLC Network

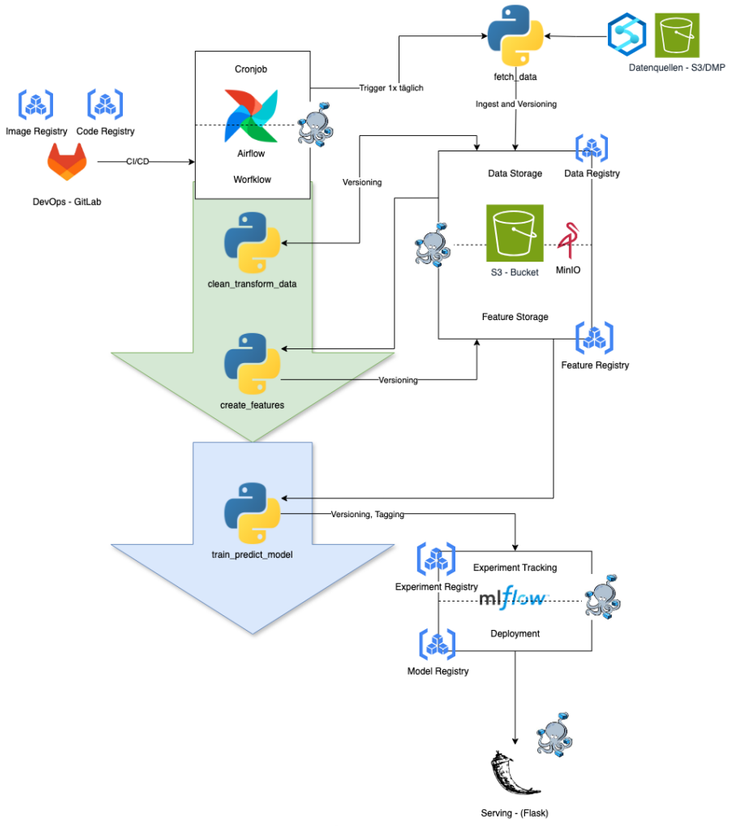

Abbildung 3: Machine Learning Lifecycle

Evaluation

Bei den Modellen zeigen abhängig von der Vorhersage-Art (täglich oder halbstündig) unterschiedliche Modelle bessere Ergebnisse. Für die tägliche Vorhersage ist Prophet das beste Modell, während für die halbstündige Vorhersage die Modelle Chronos, die besten Ergebnisse liefert. Sie sind nicht bei allen Reisezentren die besten Modelle, weswegen sich ein einzelnes Modell für alle Reisezentren nicht empfiehlt. Zwischen den Modell-Typen (MLP, Transformer, usw.) gibt es keinen klaren Trend, dass ein Modell-Typ besser ist als die anderen. Beispielsweise bei den TimeSeries Foundation Models zeigt Chronos überwiegend gute Ergebnisse während MOIRAI mittelmäßige bis schlechte Ergebnisse erzielt. Die Vorhersagen weichen ca. 10-60% abhängig vom Modell von den tatsächlichen Werten ab und streuen relativ zu ihren Werten ähnlich stark. Auch exogene Variablen bieten einen Mehrwert für die Vorhersage des Besucheraufkommens.

Model	MBE	RMSE	MASE	MAPE	QL
ARIMAX	-6.301	12.195	2.111	1.343	3.416
Chronos	-0.877	5.414	**0.81**	0.605	**1.478**
LGBMRegressor	-3.29	8.299	1.308	0.71	2.191
LSTM	-1.286	6.057	0.963	0.875	1.732
MOIRAI	0.872	8.644	1.774	3.471	3.545
Naive	-8.318	13.323	2.108	**0.397**	3.327
NeuralProphet	1.808	6.206	1.122	1.59	2.229
Prophet	0.106	5.413	1.097	1.668	1.987
SeasonalNaive	-0.221	6.432	0.982	0.891	1.844
TSMixerx	-1.169	5.614	1.03	1.353	1.846
TimeXer	-1.37	5.604	0.952	1.04	1.705
MeanNaive	-0.221	10.521	2.414	4.621	4.625
iTransformer	-1.124	**5.302**	0.894	0.947	1.607

Abbildung 4: halbstündige Ergebnisse

Model	MBE	RMSE	MASE	MAPE	QL
ARIMAX	20.069	144.439	1.142	1.959	59.54
Chronos	9.036	127.245	**0.948**	1.777	47.914
LGBMRegressor	-85.129	173.05	1.329	**1.575**	64.007
LSTM	4.097	133.545	1.046	1.78	52.808
MOIRAI	37.226	148.671	1.178	2.016	62.474
Naive	96.848	177.602	1.493	2.533	82.641
NeuralProphet	69.717	160.662	1.314	2.028	71.369
Prophet	-12.585	**123.358**	0.981	1.682	**47.636**
SeasonalNaive	-18.154	197.698	1.445	1.835	74.355
TSMixerx	-25.975	140.066	1.093	1.637	53.085
TimeXer	15.611	143.046	1.107	1.788	57.604
MeanNaive	-14.714	171.428	1.346	1.986	69.406
iTransformer	8.354	142.715	1.099	1.768	57.072

Abbildung 5: tägliche Ergebnisse

Erkenntnisse

Welches Zeitreihen-Modell ist auf Basis von vordefinierten Metriken am besten geeignet, um die Besucherzahlen vorherzusagen?
Bei den halbstündigen Vorhersagen sind auf Basis der Ergebnisse die besten Modelle Chronos, iTransformer und LSTM. Bei den täglichen Vorhersagen sind die besten Modelle Prophet, Chronos und LSTM.

In welchem Umfang kann der MLLC automatisiert werden?
Mithilfe der Tools Airflow und MLFlow ist der MLLC durchweg automatisiert. Basierend auf den Maturity Levels von Google und Microsoft befindet sich der Automatisierungs-Grad auf dem Level 2/2 bzw. 3.5/4. Sowohl die DevOps-Pipeline, als auch die Data-Pipeline und ML-Pipeline ist vollständig automatisiert. Für Stufe 4 fehlen A/B-Tests und eine Monitoring- Umgebung.

Lässt sich eine containerisierte Architektur für den MLLC nutzen, um den Prozess zu vereinfachen und plattformunabhängig zu gestalten?
Die Architektur ist mithilfe den Tools Docker und Docker Compose vollständig containerisiert. Dadurch ist der MLLC plattformunabhängig und kann auf verschiedenen Systemen ausgeführt werden. Die Container sind so konfiguriert, dass sie mithilfe des Docker Networks untereinander kommunizieren können.

Wie können durchgängig reproduzierbare Ergebnisse über den gesamten MLLC hinweg sichergestellt werden?
Alle durchgeführten Tests und Trainingsläufe sind protokolliert und können mithilfe von Versionierung und Seeding nachvollzogen und reproduziert werden. Dank der genutzten Registries und Tools besteht die Möglichkeit während der Entwicklung Pipeline-Durchläufe der Datenverarbeitung oder der Modell-Trainingsläufe zu testen und an Zwischenstellen anzustarten ohne den kompletten Workflow neu zu starten.