Analyse von Verkehrsdaten mit Zeitreihenmodellen

Patrick Reckeweg and Alfred Oshana

Hochschule Darmstadt, Haardtring 100, 64295 Darmstadt
stpareck@stud.h-da.de, alfredashur.oshana@stud.h-da.de

Abstract. Die folgende Technologiestudie setzt sich mit der Zeitreihenanalyse von Verkehrsdaten mithilfe von geeigneten statistischen und ML-Modellen auseinander. Als Library wurde AutoTS verwendet, welches speziell für Zeitreihendatensätze konzipiert wurde. Hierfür wurde ein Datensatz ausgewählt, welcher das Aufkommen von Kraftwagen an 4 festen Kreuzungen in einem Zeitraum von 2015 bis 2017 in regelmäßigen stündlichen Zeitintervallen zählt.
Anhand einer Vorabanalyse und Dekomposition des Datensatzes ist ersichtlich, dass die Daten der 4 Kreuzungen sich in Umfang, Zeitverlauf und Verlauf zu sehr unterscheiden, woraufhin im Rahmen dieser Studie entschieden wurde alle Modelle auf jede Kreuzung einzeln zu trainieren. Die Trainingsumgebung wurde auf einer Cloud-Infrastruktur mit geeigneter Hardware, Treibern, Images und Software durchgeführt, welche reproduzierbar zur Verfügung gestellt wird. Bei der Auswertung der Ergebnisse wird insbesondere der Vergleich der Modelle untereinander betrachtet und schließlich wird die Prognose auf Basis des trainierten Modells direkt mit dem echten Verlauf der Daten verglichen.

Keywords: Zeitreihendaten · Zeitreihenvorhersage · AutoML.

1 Einleitung

Für das Szenario des Big Picture "Street Smarts", ein digitales Straßensystems, bei dem mit Hilfe von IOT-Technologien Verkehrsverläufe dynamisch an die jeweilige Straßenauslastung angepasst werden sollen, werden eine Reihe von Studien durchgeführt, welche einzelne Komponenten des Systems auf Anwendbarkeit prüfen. In dieser Studie werden ML-Modelle verwendet um mithilfe von Daten Prognosen auf zukünftige Verhältnisse zu liefern. Durch diese Prognosen kann man den Verkehrsfluss im Voraus so effizient wie möglich anpassen. Die Modelle, die in dieser Studie verwendet werden, sind durch die AutoTS-Library bereitgestellt. Diese Library liefert eine End-to-End Pipeline vom Eintragen des Datensatzes bis hin zum Anwenden des trainierten Modells auf potentiell neue Datensätze. Bei den Daten, die genutzt werden handelt, es sich speziell um Zeitreihendaten, welche am geeignetsten sind um Attribute in gleichmäßig wiederkehrenden Zeitintervallen darzustellen.

1.1 Zielsetzung

Das Ziel dieser Studie ist es mithilfe von Zeitreihendaten zu prüfen, welche Modelle sich am besten eignen um zuverlässige Prognosen auf den weiteren Verlauf zu liefern. Zu untersuchen sind vor allem regressive Modelle, Modelle mit neuronalen Netzen, aber auch Modelle mit naiveren Ansätzen wie z.B. Durchschnittsberechnungen.

2 Grundlagen

Für den in der Einleitung beschriebenen Anwendungsfall werden im Verlauf dieser Arbeit verschiedene Algorithmen aus dem Bereich der Zeitreihenanalyse getestet. In diesem Abschnitt werden die Grundlagen der Zeitreihenanalyse vorgestellt.

2.1 Zeitreihendaten

Zeitreihendaten geben den Wert einer oder mehrerer Variablen in Abhängigkeit zur Zeit an. Bilden die Daten eine Variable abhängig von der Zeit ab, handelt es sich um univariate Zeitreihendaten. Bei mehreren Variablen ist es eine multivariate Zeitreihe. Zeitreihendaten können kontinuierlich oder diskret sein. Im Fall der kontinuierlichen Zeitreihendaten werden in einem konkreten Zeitraum kontinuierlich Daten aufgenommen. Bei diskreten Zeitreihendaten werden Daten zu spezifischen Zeitpunkten aufgezeichnet. Diese spezifischen Zeitpunkte sind meistens in einem gleichbleibenden zeitlichen Abstand zueinander angeordnet. Eine Möglichkeit ist zum Beispiel Daten mit einer Abtastrate von einer Stunde zu sammeln.

Durch die Abhängigkeit von der Zeit ist der Wert x_t automatisch abhängig vom Wert x_{t-1} und den weiter in der Zeit zurückliegenden Werten. [9] [8] [4] [2] [1]

2.2 Zeitreihenanalyse

Mit Hilfe der Analyse von Zeitreihen wird versucht den der Zeitreihe zu Grunde liegenden Prozess zu modellieren. Dazu werden statistische Eigenschaften der Zeitreihe untersucht. Mit Hilfe von mathematischen Modellen, die jeweils verschiedene statistische Eigenschaften der Zeitreihe voraussetzen, wird dieser Prozess angenähert. Falls die untersuchte Zeitreihe einige statistische Eigenschaften nicht aufweist, kann sie durch Vorbearbeitung der Daten so verändert werden, dass sie diese doch erfüllt. Auf diese Weise kann ein Modell auf eine Zeitreihe angewendet werden, die sich eigentlich nicht für dieses Modell eignet. Die an die Zeitreihe angepassten Modelle können anschließend für verschiedene Aufgaben wie unter anderem Vorhersage, Klassifikation, Simulation oder Anomaliedetektion genutzt werden. [7] [1] [8]

2.3 Modelltypen

Die für die Zeitreihenanalyse geeigneten Modelle lassen sich in vier Kategorien einordnen:

- Lineare Modelle
- Nicht-Lineare Modelle
- Künstliche Neuronale Netze
- Support Vektor Maschienen

Lineare Modelle setzen voraus, dass die Zeitreihe generell einen linearen Verlauf hat oder ausreichend durch einen linearen Verlauf angenähert werden kann. Der Wert x_t wird dann durch eine Linearkombination der letzten p Werte angenähert. Eine formale Formulierung eines linearen Modells hat folgende Form:

$$x_t = \alpha_0 + \alpha_1 x_{t-1} + \ldots + \alpha_p x_{t-p} + \epsilon_t \tag{1}$$

Die Faktoren α dieses linearen Modells können nun z.B. durch lineare Regression an die Zeitreihendaten angepasst werden. [10]

Im Gegensatz zu den linearen Modellen weisen nicht-lineare Modelle keinen generell linearen Verlauf auf. In der Realität weisen die wenigsten Zeitreihen einen linearen Verlauf auf, aber oft reicht die Annäherung durch ein lineares Modell aus. Sollte dies der Fall sein, werden nicht-lineare Modelle verwendet. Diese Modelle sind nicht mehr als Linearkombinationen darstellbar. Die Formel für das multiplikative Modell lautet:

$$x_t = \alpha t^{\beta} u \tag{2}$$

Bei diesem Modell müssen die Faktoren α, β und u an die Zeitreihendaten angepasst werden. [3]

In den letzten Jahren haben vor allem Modelle basierend auf Neuronalen Netzen viel Aufmerksamkeit erhalten. Durch die Art und Weise wie diese Modelle die Zielfunktion annähern, ist es möglich lineare wie nicht-lineare Funktionen abzubilden. Zusätzlich benötigen diese Modelle weniger Feineinstellung basierend auf Domänenwissen. So ist es auch unerfahreneren und domänenfremden Anwendern möglich gute Modelle zu erstellen. Außerdem treffen solche Modelle keine Annahmen bezüglich der statistischen Eigenschaften der Daten und benötigen daher auch weniger Vorbearbeitung der Daten. Durch die große Menge anzupassender Parameter in einem Neuronalen Netz besteht bei diesen Modellen die Gefahr, dass sie sich an die Daten überanpassen. Das bedeutet, dass sie die Daten auswendig lernen und so bei neuen Daten nicht mehr gut generalisieren. [11]

Zur effizienteren Optimierung werden oft stochastische Verfahren eingesetzt, sodass diese Modelle in einem lokalen Minimum enden. [11] [4, p.42-56]

Eine ebenfalls verhältnismäßig neue Gruppe von Modellen sind die Support Vektor Maschinen (SVM). Diese Modelle sind in der Lage lineare und nichtlineare Funktionen anzunähern. Durch die Formulierung dieser Modelle besteht bei ihnen nicht die Gefahr in einem lokalen Minimum zu enden. Das bedeutet im Umkehrschluss die gefundene Lösung ist optimal. [11] [6] [8] [1]

2.4 Vorgehensweise bei der Analyse und Vorhersage

Der erste Schritt ist die Analyse des Datensatzes um dessen Eigenschaften zu untersuchen. Anhand dieser Eigenschaften wird über die Auswahl eines geeigneten Modells entschieden. Dem Anwender stehen trotz der Einschränkungen durch die Eigenschaften der Zeitreihe noch eine Vielzahl an geeigneten Modellen zur Auswahl. Oft fließt die Erfahrung des Anwenders deshalb in die Modellwahl mit ein. Nach der Wahl des Modells muss entschieden werden, ob die Daten vorbearbeitet werden und, falls es sich um ein parametrisches Modell handelt, wie die Hyperparameter des Modells eingestellt werden. Da sich alle drei Aspekte gegenseitig beeinflussen ist es ohne ausreichende Erfahrung schwer eine gut aufeinander abgestimmte Auswahl zu treffen. Alle im Abschnitt 2.3 vorgestellten Modelle müssen anschließend an den jeweiligen Datensatz angepasst werden. In dieser Trainingsphase werden die Modellparameter durch einen Optimierungsalgorithmus an die Daten der Zeitreihe angepasst. Der letzte Schritt besteht darin, die Güte des Modells bezüglich einer bestimmten Metrik auf dem Modell unbekannten Daten zu evaluieren. Danach kann iterativ versucht werden die Güte des Modells durch Anpassungen an den Hyperparamterwerten zu verbessern. Die gleiche Vorgehensweise wird ebenfalls bei Datensätzen, bei denen es sich nicht um Zeitreihendaten handelt.

2.5 AutoML

"Automated Machine Learning" (AutoML) Lösungen sollen den in Abschnitt 2.4 beschrieben Vorgang automatisieren. Wie beschrieben ist viel Erfahrung nötig um initial eine sinnvolle Auswahl an Vorbearbeitungsschritten, Modellen und Hyperparameterwerten zu treffen. Die anschließende Hyperparameteroptimierung benötigt ebenfalls Erfahrung um effektiv und erfolgreich zu verlaufen. Da Maschinelles Lernen in den letzten Jahren für immer mehr Anwendungsfälle genutzt wird, ist die Nachfrage an Spezialisten größer als das Angebot. Daher bieten sich AutoML Lösungen an, um unerfahreneren Anwendern ebenfalls die Nutzung dieser Modelle zu ermöglichen.

AutoML Lösungen übernehmen daher alle Schritte auf dem Weg vom Datensatz zum trainierten und optimierten Modell. Für die kombinierte Auswahl von Vorbearbeitungsschritten, Modell und Hyperparameterwerten werden ebenfalls mathematische Optimierungsalgorithmen genutzt. Die Modellgüte wird als Funktion der Auswahl von Vorbearbeitungsschritt, Modell und Hyperparameterwerten formuliert und anschließend optimiert. Als Nutzer muss man der die AutoML Lösung implementierenden Bibliothek nur den Datensatz übergeben.

Die Rückgabe enthält das beste von der AutoML Lösung gefundene Modell sowie Informationen über die anderen getesteten Modelle. [5]

3 Versuchsaufbau

Um zu prüfen, ob Zeitreihenmodelle genutzt werden können um ausreichend gute Vorhersagen bezüglich des Verkehrsaufkommens zu erhalten, wurde ein Versuch auf einem Beispieldatensatz durchgeführt. In diesem Abschnitt wird beschrieben, welche Werkzeuge und welcher Datensatz für den Versuch genutzt wurden.

3.1 AutoTS

Um die in Abschnitt 2.3 und 2.4 beschriebenen Herausforderungen zu adressieren wurde das Python Paket "AutoTS" für den Versuch ausgewählt. Es handelt sich bei AutoTS um eine AutoML Bibliothek, die auf die Auswahl und das Training von Zeitreihenmodellen spezialisiert ist. Einige Modelle sind in der AutoTS Bibliothek selber implementiert, der Großteil der testbaren Modelle wird jedoch über andere Bibliotheken eingebunden. AutoTS bietet die Möglichkeit die in Tabelle 1 aufgelisteten Bibliotheken einzubinden.

Table 1. Diese Tabelle führt alle Bibliotheken auf, die von AutoTS integriert werden können. Zusätzlich ist vermerkt, welchem im Absatz 2.3 vorgestellten Modelltypen die Modelle dieser Bibliotheken zuzuordnen sind.

Bibliothek	Modelltyp
statsmodels	
scipy	
sklearn	Neuronale Netze, SVM
arch	
lightgbm	
tensorflow	Neuronale Netze
tensorflow_probability	
pytorch_forecasting	
prophet	
neuralprophet	
greykite	
gluonts/mxnet	Neuronale Netze

Der Nutzer muss dafür sorgen, dass alle Bibliotheken, aus denen er Modelle testen lassen will, auf dem für das Training genutzten Rechner installiert sind.

(Für die Auswahl der ersten Kombinationen aus Modell & Hyperparameter wird von AutoTS im Vorbearbeitungsschritt bestimmte Templates genutzt, welche generell gute Ergebnisse zu Zeitreihendatensätze liefern.) Für die Auswahl der ersten Vorbearbeitungsschritt-Modell-Hyperparameterwert-Kombinationen nutzt AutoTS Templates von Kombinationen, die generell auf Zeitreihendatensätzen

gute Ergebnisse geliefert haben. Als Optimierungsalgorithmus zur Verbesserung der Kombinations-Auswahlen wird ein genetischer Algorithmus genutzt. [1]

3.2 Cloud

Um das Training von rechenintensiven Algorithmen wie Neuronalen Netzen oder SVM's zu ermöglichen wurden Cloudressourcen als Infrastruktur für diesen Versuch genutzt. Amazon Web Services (aws) wurde als öffentlicher Cloudanbieter ausgewählt. Damit die Reproduzierbarkeit des Versuches gewährleistet ist wurden mehrere "Infrastruktur als Code" (IaC) Werkzeuge genutzt um die Konfiguration des Betriebssystems und die Bereitstellung der Cloudressourcen zu vorzunehmen. Die Konfiguration des Betriebssystems, die Installation aller benötigten Softwarepakete und das Erzeugen eines virtuellen Machinen Images (VM-Image) dieses Zustandes wurde mit dem IaC Werkzeug "Packer" [2] durchgeführt. Die Bereitstellung aller benötigten Cloudressourcen inklusive einer VM, die das zuvor gepackte Image nutzt, wurde mit dem IaC Werkzeug "Terraform" [3] umgesetzt. Als Instanztyp für die VM für die GPU Experimente wurde der Typ "p3.2xlarge" gewählt. Die GPU beschleunigt die Trainingsgeschwindigkeit Neuronaler Netze deutlich. Als Instanztyp für die VM für die CPU Experimente wurde der Typ "m5.4xlarge" gewählt.

3.3 Datensatz

Für diesen Versuch wurde der Kaggle "Traffic Prediction" Datensatz genutzt [4].

Der Datensatz besteht aus 48120 Datenpunkten und besitzt folgende Attribute:

- ID
- DateTime
- Junction
- Vehicles

Bei dem Attribut "ID" handelt es sich um einen pro Eintrag eindeutigen Identifizierer. Daher wurde dieses Attribut am Anfang der Untersuchungen entfernt. Die Daten des Datensatzes wurden an vier verschiedenen Kreuzungen gesammelt. Das Attribut "Junction" gibt pro Datenpunkt an, an welcher der vier Kreuzungen dieser Datenpunkt aufgezeichnet wurde. Das Attribut "DateTime" enthält pro Datenpunkt den Zeitstempel der Aufzeichnung. Die Daten wurden alle 60 Minuten, also stündlich erhoben. Die Datenpunkte zu den Kreuzungen eins bis drei decken den Zeitraum vom 01.11.2015 bis zum 01.07.2017 ab. Die Aufzeichnung der Datenpunkte an Kreuzung vier startete am 01.01.2017 und

[1] https://winedarksea.github.io/AutoTS/build/html/source/tutorial.html
[2] https://www.packer.io/
[3] https://www.terraform.io/
[4] https://www.kaggle.com/datasets/fedesoriano/traffic-prediction-dataset

endete, genau wie die der anderen Kreuzungen, am 01.07.2017. Das Attribut "Vehicles" gibt an wie viele Fahrzeuge zu einem bestimmten Zeitpunkt an einer bestimmten Kreuzung vorhanden waren.

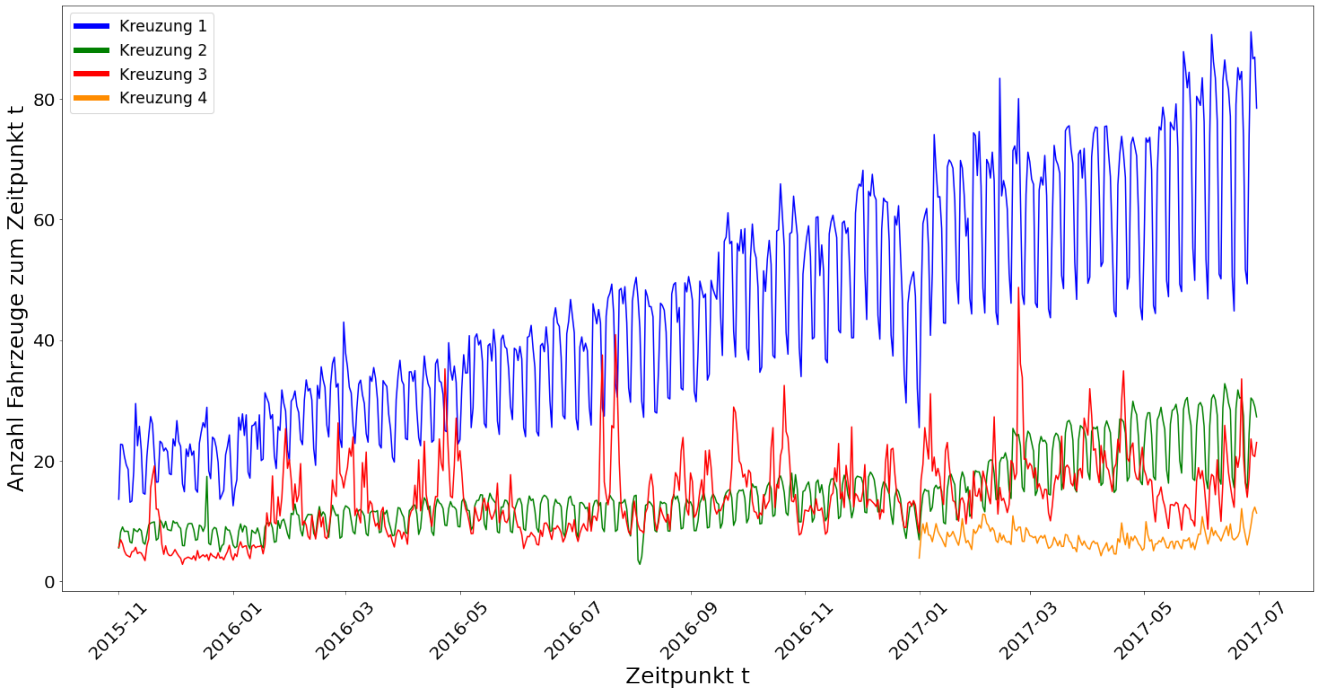

Fig. 1.

Eine Dekomposition der Zeitreihendaten in einen generellen Trend Anteil, einen Saisonalen Anteil und einen Irregulären Anteil kann so Aufschluss darüber geben, welche Art von Modellen geeignet seien könnten. Natürlich ist dies kein definitiver Beweis für die Eignung eines bestimmten Modells. Diese Dekomposition folgt entweder einem additiven oder einem multiplikativen Modell. Im Falle eines additiven Modells, ergibt sich der wahre Fahrzeuganzahlwert zum Zeitpunkt t folgendermaßen:

$$Fahrzeuganzahlwert_t = Trend_t + Saison_t + Irregulärer_t \tag{3}$$

Dementsprechend sieht der Ansatz eines multiplikativen Modells wie folgt aus:

$$Fahrzeuganzahlwert_t = Trend_t \cdot Saison_t \cdot Irregulärer_t \quad (4)$$

Die Abbildung 2 zeigt eine additive Dekomposition für alle vier Kreuzungen. Die Datenpunkte wurden wieder durch Mittelwertbildung auf eine tägliche Frequenz reduziert.

Fig. 2. Dekomposition der Zeitreihendaten pro Kreuzung mit Hilfe eines additiven Modells.

3.4 Zielmetrik

AutoTS bietet die Möglichkeit eine gewichtete Summe verschiedener Metriken als Zielmetrik zu nutzen. Die so berechnete Metrik ist durch die Nutzung verschiedener anderer Metriken robuster und ein Model, welches bei dieser Metrik einen guten Wert erzielt generalisiert besser, als wenn es bei einer der einzelnen Metriken einen guten Wert erzielt. Auf diese Weise soll verhindert werden, dass

ein Modell ausgewählt wird, welches bezüglich einer untersuchten Metrik gute Werte liefert, bei allen anderen aber schlechte. Tabelle 2 zeigt welche Metriken mit welcher Gewichtung standardmäßig in die Zielmetrik einfließen.

Table 2. Diese Tabelle führt auf, welche Metrik mit welchem Faktor in die Berechnung der für die Experimente genutzten Metrik einfließt.

Metrik	Gewichtungsfaktor	gute Werte
Symmetric mean absolute percentage error (SMAPE)	5	klein
Mean Absolute Error (MAE)	2	klein
Root Mean Squared Error (RMSE)	2	klein
Mean Absolute Differential Error (MADE)	0.5	klein
Scaled Pinball Loss (SPL)	3	klein
Contour Loss	1	groß ()
Runtime	0.05	klein
Origin Directional Accuracy (ODA)	0.001	groß ()

3.5 Versuchsparamter

Obwohl AutoTS die Einstellung der Hyperparameter der Modelle übernimmt, müssen einige Parameter für AutoTS vom Anwender gewählt werden. Folgende Parameter wurden für die Experimente eingestellt:

- Zeithorizont: Die Anzahl an diskreten Zeitschritten, für die eine Vorhersage erstellt und evaluiert werden soll.
- Ensemble: AutoTS biete die Möglichkeit kein einzelnes Modell sondern ein Modellensemble zurückzuliefern. über dieses Attribut wird eingestellt ob diese Möglichkeit genutzt wird.
- Modell Liste: Modelle die von AutoTS genutzt werden sollen.
- Transformer Liste: Alle Vorbearbeitungsschritte, die von AutoTS genutzt werden sollen.
- Generationen: Die Anzahl an Generationen, die der genetische Optimierungsalgorithmus nutzen soll.
- Anzahl Validationen: Gibt die Anzahl an Kreuzvalidierungsdurchläufen an, die nach dem Training durchlaufen werden sollen.
- Urlaubsland: Land aus dem die Daten stammen. Manche Modelle bekommen diesen Wert übergeben und beziehen Feiertage als zusätzlich Variable mit ein.

Für die beiden Experimente wurden die Attributwerte aus Tabelle 4 genutzt. Für das CPU und GPU Experiment jeweils wurden die in Abschnitt 3.2 beschriebenen Instanztypen verwendet.

4 Ergebnisse

Bei der Durchführung beider Experimente sind Schwierigkeiten aufgetreten. Auf Grund von Inkompatibilitäten von den Python Bibliotheken, die Neuronale Netze

unterstützen, und den Grafikartentreibern konnten diese Bibliotheken die GPU nicht nutzen. Als das Experiment wegen der langen Laufzeit und aus Kostengründen abgebrochen werden musste, führte ein Fehler im Experimentcode dazu, dass für GPU Experiment keine Teilergebnisse gespeichert werden konnte. Auch die CPU Experimente mussten teilweise unterbrochen und neu gestartet werden. Dies hat dazu geführt, dass beim Training der Modelle zu Kreuzung drei das Modell ARIMA nicht trainiert und getestet wurde. Tabelle 4 ist pro Kreuzung der Metrikwert des besten Modells aufgeführt. Die Werte der Modelle zu den Kreuzungen eins bis drei bewegen sich in einem ähnlichen Wertebereich. Der Wert des Modells für Kreuzung vier weicht ab.

Table 3. Diese Tabelle zeigt den Metrikwert des jeweils besten Modells pro Kreuzung. Es handelt sich um die Validationswerte nach fünfmaliger Kreuzvalidierung.

Kreuzung	Modell	Zielmetrikwert
1	SectionalMotif	60,394647
2	UnivariateMotif	51,501498
3	UnivariateMotif	75,747905
4	DatepartRegression	116,259857

Da neben den besten Modellen ebenfalls untersucht wird, ob es eine Modellfamilie gibt, die sich besonders gut eignet, stellt Abbildung 3 dar, welche Modelle wie oft im jeweils besten bzw. schlechtesten Viertel aller trainierten Modelle vertreten sind. Zu diesem Zweck wurde pro Kreuzung das beste und schlechteste Viertel der Modelle über ihren Validationsmetrikwert bestimmt. Diese Teilmengen der trainierten Modelle sind in den Abbildungen 6 - 9 im Anhang pro Kreuzung abgebildet. Diese Teilmengen pro Kreuzung wurden zusammengeführt und bilden die Datengrundlage von Abbildung 3.

Neben Softwareproblemen hat sich beim Training der Modelle gezeigt, dass die benötigte Zeit für das Training von Modell zu Modell abweicht. Bei der Konfiguration der Metrik für dieses Experiment wird die Laufzeit nur mit einem Faktor von 0,05 einbezogen. Sie hat also keinen großen Einfluss auf den Metrikwert. Um diesen Unterschied der Modelle nicht zu vernachlässigen bildet Abbildung 4 die Verteilung der Laufzeit des Trainings pro Modell ab.

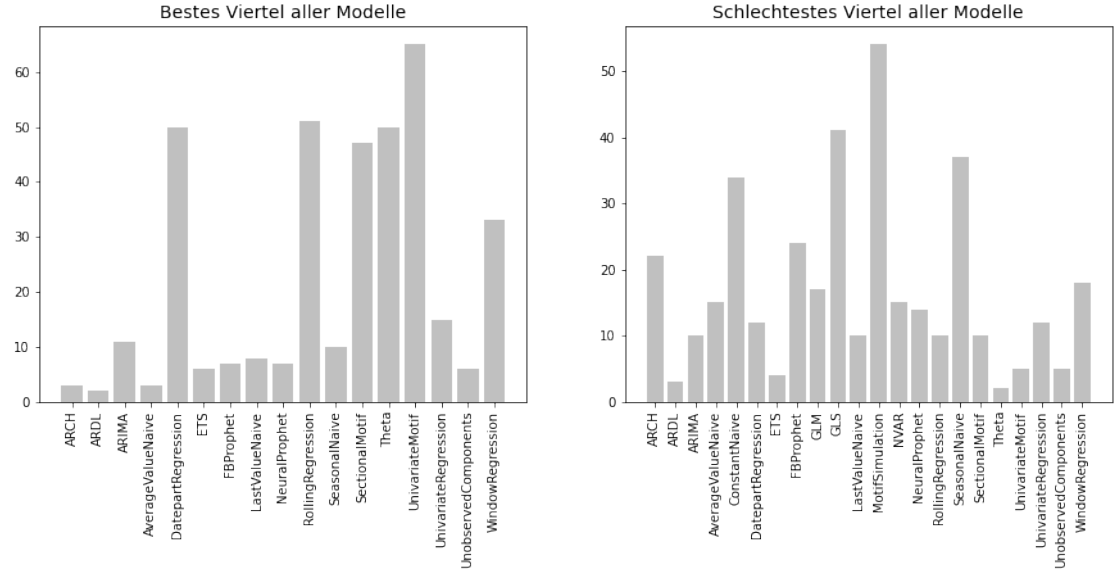

Fig. 3. Dieses Diagramm bildet die Häufigkeit ab, mit der bestimmte Modelle in den jeweils besten und schlechtesten 25% der trainierten Modelle vertreten sind.

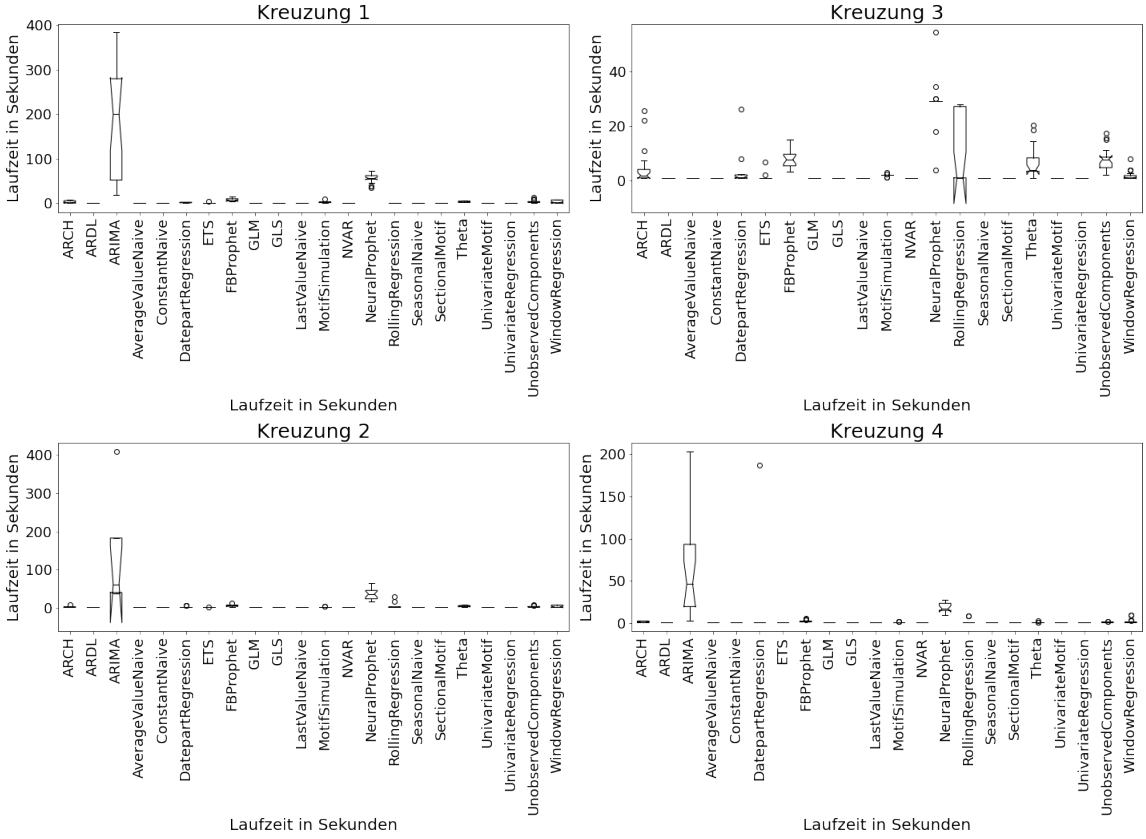

Fig. 4. Diese Abbildung zeigt die Verteilungen der Laufzeitwerte pro Modell und Kreuzung.

Abbildung 5 zeigt am Beispiel des Zeitraumes vom 01.07.2017 bis zum 31.07.2017 wie sich die Vorhersage des jeweils besten Modells pro Kreuzung im Vergleich zum wahren Wert der Zeitreihe verhält. Für diese Darstellung wurde die Konfiguration des jeweiligen Modells übernommen und das Modell neu auf den Daten trainiert. Dabei wurde jeweils die Vorhersage für den nächsten Zeithorizont aufgezeichnet. Dadurch hat das Modell keine Vorhersagen auf bereits bekannten Daten durchgeführt.

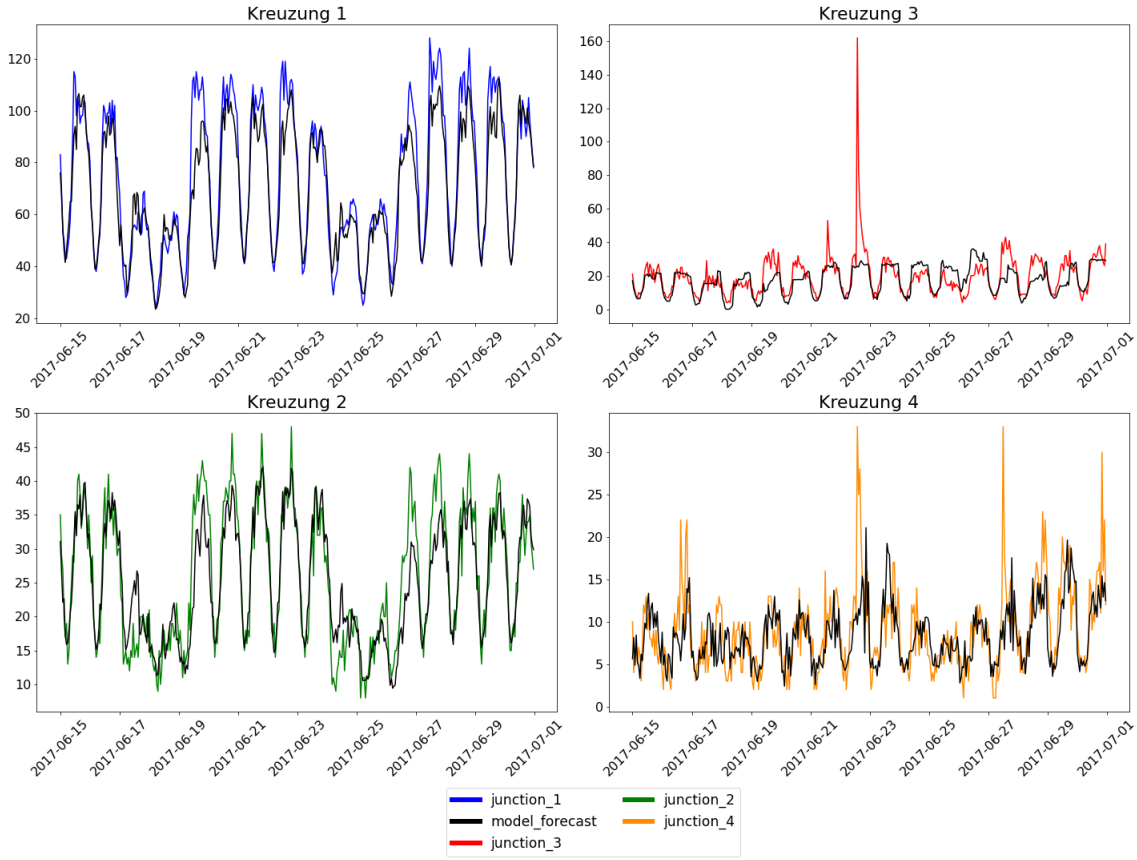

Fig. 5. Diese Abbildung zeigt die Vorhersagen der besten Modelle pro Kreuzung verglichen mit den originalen zeitreihendaten.

5 Evaluation

In Anbetracht der verwendeten Metriken liefern alle Modelle keinen guten Wert. Dies könnte an der Anzahl von Generationen liegen. Es wurde eine Anzahl von 10 Generationen gewählt, um eine sehr lange Laufzeit und hohe Kosten

zu vermeiden. Die Metrikwerte pro Kreuzung aus Tabelle 4 zeigen auch Unterschiede zwischen den Kreuzungen. Am meisten weicht das Ergebnis des Modells zu Kreuzung vier von den anderen ab. Dies könnte daran liegen, dass zu den anderen Kreuzungen fünfmal so viele Daten vorhanden sind. Dieses Modell konnte daher weniger trainiert werden. Vergleicht man die Abbildung 2 mit der Tabelle 4 fällt auf, dass die Kreuzungen eins und drei schlechtere Ergebnisse aufweisen als Kreuzung zwei. Im Dekompositionsdiagramm fallen diese beiden Kreuzungen dadurch auf, dass ihr Irregulärer, also zufälliger Anteil höher ist als der von Kreuzung zwei. Vergleicht man in diesem Kontext die Abbildungen 6 - 8 zeigt sich, dass im besten Viertel der trainierten Modelle zu Kreuzung eins und zwei ähnliche Modelle vertreten sind. Bezüglich dieser beiden Kreuzungen sind Modelle und das Auftreten vergleichbar. Anders als erwartet zeigt Abbildung 8, dass bei den besten Modellen zu Kreuzung drei auch sehr simple Modelle vertreten sind. Bei dem verhältnismäßig hohen Anteil an Zufall in der Zeitreihe zu dieser Kreuzung wäre eher zu erwarten gewesen, dass vor allem komplizierte Modelle erfolgreich waren.

Im Bezug auf die Fragestellung, ob Zeitreihenmodelle geeignet sind um sie als Teil eines Systems zur Regelung und Optimierung von Verkehr genutzt zu werden zeigt Abbildung 5, dass die Modelle sogar nach nur zehn Durchläufen des genetischen Algorithmus von AutoTS hinreichende Ergebnisse liefern. Da es im Fall von "StreetSmarts" nicht um die Steuerung von Autos sondern das Sperren bzw. Freigeben von Fahrbahnspuren oder das Umleiten von Verkehr handelt, sind Fehler hinnehmbar. Sollte ein Modell ein falsches Verkehrsaufkommen prognostizieren, führt das im schlimmsten Fall zu Stau.

Ein Faktor, der während den Experimenten mehrfach deutlich wurde, ist dass AutoTS noch eine Betaversion (v0.4.2) ist. Diese Bibliothek erleichtert es Anwendern, die kein Fachwissen besitzen erfolgreich Modelle zu trainieren und auszuwählen. Durch die Unreife der Bibliothek kann es aber auch oft zu Fehlern führen. Zusätzlich hat es sich als sehr schwierig erwiesen alle unterstützten Bibliotheken gleichzeitig zu installieren. Es kann schnell zu Inkompatibilitäten führen und z.B. die Bibliotheken mit GPU Unterstützung existieren nicht alle als fertige Pakete mit Unterstützung für die gleich Version von CUDA Treibern. Es gibt natürlich die Möglichkeit diese Pakete per Hand zu kompilieren. Dies erschwert wiederum das Experimentieren.

5.1 Beschränkungen des Versuches

Eine Beschränkung dieser Studie ist, dass nicht alle möglichen Arten von Modellen getestet werden konnten. Es konnten nur wenige Regressionsmodelle die auf Neuronalen Netzen beruhen getestet werden und keine Support Vektor Maschinen. Diese Tatsache lässt keine direkten Vergleiche zwischen den in Abschnitt 2.3 beschriebenen Modelltypen zu. Zusätzlich konnten die getesteten Modelle nicht vollkommen optimiert werden, da die Hardware dafür nicht zur Verfügung stand. Eine weitere Einschränkung ist der Fakt, dass der genutzte Datensatz nur eine univariate Zeitreihe bereitstellt. Gerade durch den Bezug zu Iot sollte davon ausgegangen werden, dass viel mehr Sensoren deutlich mehr Messwerte

sammeln. Alle diese Daten können die Modellgüte unterstützen und bisher als Zufall gewertete Anteile der Varianz der Daten erklären. Auch eine zeitliche Frequenz von 60 Minuten ist für einen Anwendungsfall in der echten Welt ggf. zu groß. Zusätzlich sind bei dieser Studie die Laufzeit der Algorithmen sowie ihre Hardwareanforderungen größtenteils ignoriert worden. Im IOT Umfeld kann das normalerweise alleinig über die Eignung eines Modells entscheiden, denn wenn wiederholtes lokales trainieren notwendig ist, muss an jeder Kreuzung die entsprechende benötigte Hardware vorhanden sein.

6 Fazit und Ausblick

Abschließend hat diese Studie gezeigt, das Zeitreihenmodelle für den Anwendungsfall im Projekt "StreetSmarts" geeignet sind. Es existieren viele Bibliotheken in Python, die den Anwender bei der Suche und dem Training eines geeigneten Modells unterstützen. AutoTS ist gut geeignet um unerfahrene Anwender bei der Modellauswahl zu unterstützen. Sollte ein solches gefunden sein, biete es sich aber an diese Modell ohne AutoTS, vielleicht sogar in einer Sprache zu implementieren, deren Paketmanagement weniger fehleranfällig ist. Ein solches weiterführendes Hyperparametertuning und eine Re-Implementierung bieten sich daher für weitere Experimente an. Bei solchen Experimenten müssen auch die Hardwareanforderungen berücksichtigt und untersucht werden. Durch die Wahl einer entsprechenden Programmiersprache könnten diese sogar gesenkt werden.

7 Anhang

Table 4: Diese Tabelle listet die Parameter auf die für die jeweiligen Versuche genutzt wurden. Alle weiteren Parameter, die AutoTS bietet wurden nicht verändert. Für sie gelten die Standardeinstellungen von AutoTS

Attribut	Versuch GPU	Versuch CPU
Zeithorizont	5	5
Ensemble	Nein	Nein
Modell Liste	DatepartRegression, GluonTS, MotifSimulation, Greykite, PytorchForecasting, ARCH, NeuralProphet, NVAR, SectionalMotif, UnivariateMotif, WindowRegression, RollingRegression, ARDL, Theta, ARIMA, ETS, UnobservedComponents, GLS, GLM, ConstantNaive, LastValueNaive, AverageValueNaive, SeasonalNaive, FBProphet	DatepartRegression, MotifSimulation, ARCH, NeuralProphet, NVAR, SectionalMotif, UnivariateMotif, WindowRegression, RollingRegression, ARDL, Theta, ARIMA, ETS, UnobservedComponents, GLS, GLM, ConstantNaive, LastValueNaive, AverageValueNaive, SeasonalNaive, FBProphet

Transformer Liste	MinMaxScaler, PowerTransformer, QuantileTransformer, MaxAbsScaler, StandardScaler, RobustScaler, PCA, FastICA, Detrend, RollingMeanTransformer, RollingMean100thN, DifferencedTransformer, SinTrend, PctChangeTransformer, CumSumTransformer, PositiveShift, Log, IntermittentOccurrence, SeasonalDifference, cffilter, bkfilter, convolution_filter, HPFilter, DatepartRegression, ClipOutliers, Discretize, CenterLastValue, Round, Slice, ScipyFilter, STLFilter, EWMAFilter, MeanDifference, BTCD, Cointegration	MinMaxScaler, PowerTransformer, QuantileTransformer, MaxAbsScaler, StandardScaler, RobustScaler, PCA, FastICA, Detrend, RollingMeanTransformer, RollingMean100thN, DifferencedTransformer, SinTrend, PctChangeTransformer, CumSumTransformer, PositiveShift, Log, IntermittentOccurrence, SeasonalDifference, cffilter, bkfilter, convolution_filter, HPFilter, DatepartRegression, ClipOutliers, Discretize, CenterLastValue, Round, Slice, ScipyFilter, STLFilter, EWMAFilter, MeanDifference, BTCD, Cointegration
Generationen	10	10
Anzahl Validationen	5	5
Urlaubsland	US	US

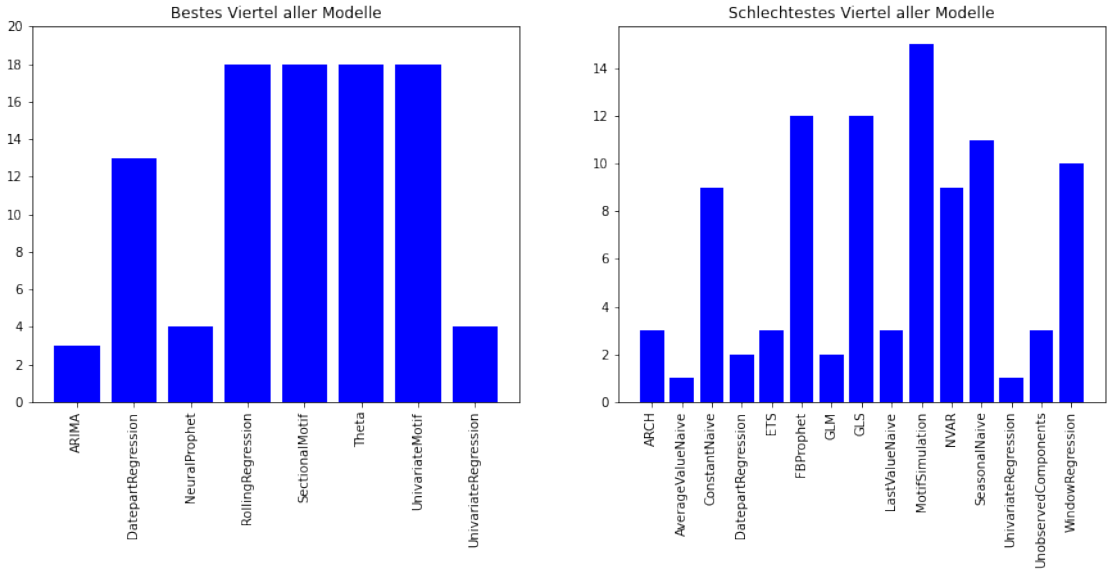

Fig. 6. Dieses Diagramm bildet die Häufigkeit ab, mit der bestimmte Modelle in den jeweils besten und schlechtesten 25% der trainierten Modelle für Kreuzung 1 vertreten sind.

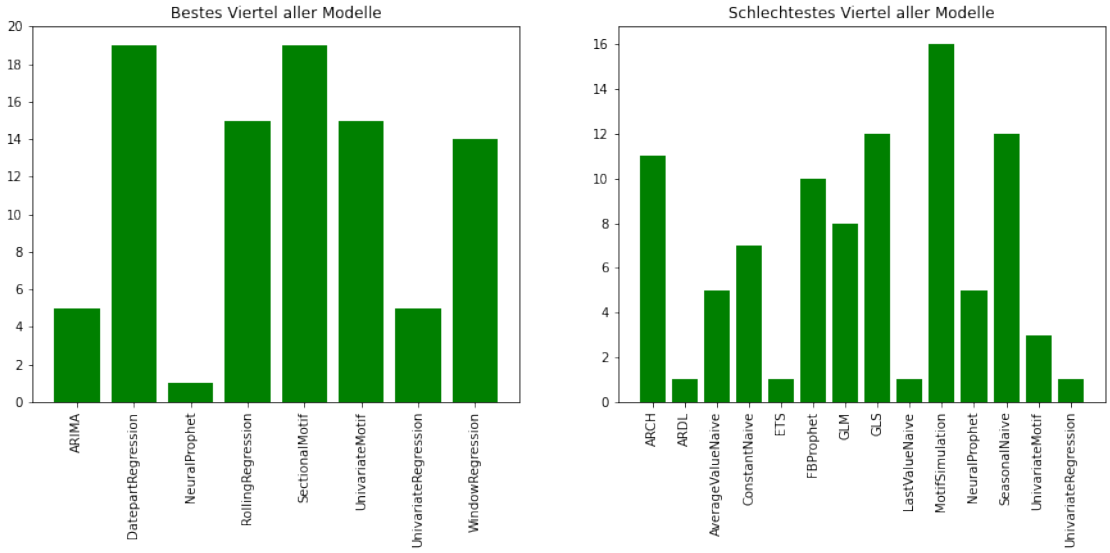

Fig. 7. Dieses Diagramm bildet die Häufigkeit ab, mit der bestimmte Modelle in den jeweils besten und schlechtesten 25% der trainierten Modelle für Kreuzung 2 vertreten sind.

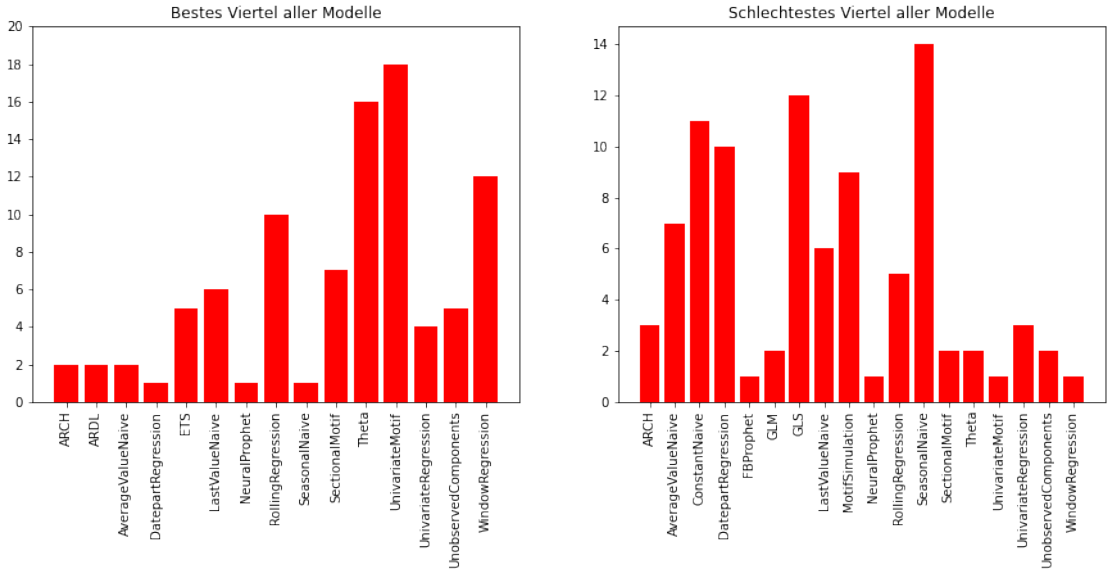

Fig. 8. Dieses Diagramm bildet die Häufigkeit ab, mit der bestimmte Modelle in den jeweils besten und schlechtesten 25% der trainierten Modelle für Kreuzung 3 vertreten sind.

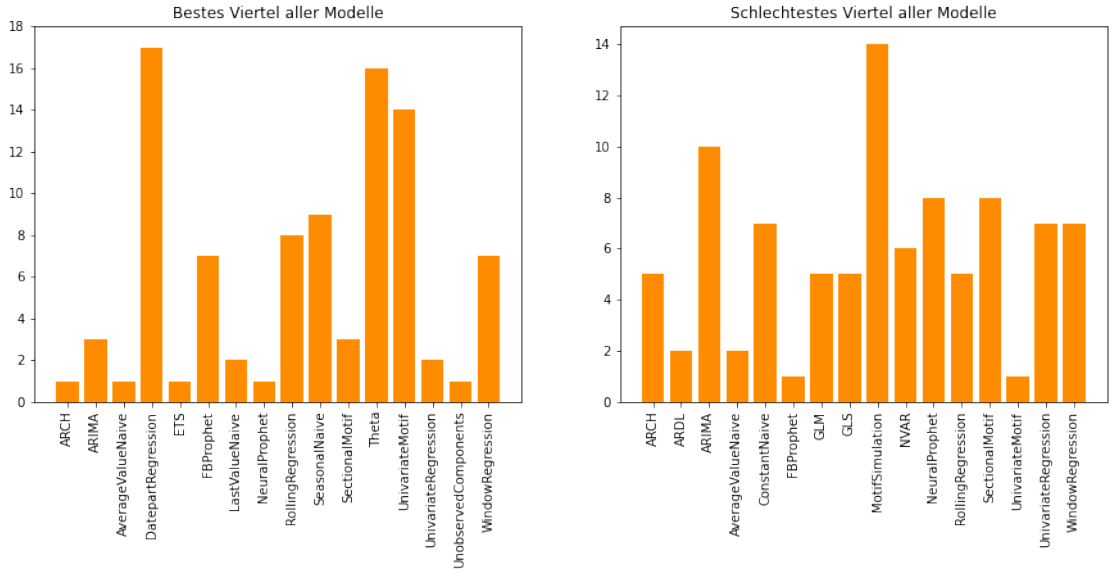

Fig. 9. Dieses Diagramm bildet die Häufigkeit ab, mit der bestimmte Modelle in den jeweils besten und schlechtesten 25% der trainierten Modelle für Kreuzung 4 vertreten sind.

References

1. Adhikari, R., Agrawal, R.K.: An introductory study on time series modeling and forecasting. arXiv preprint arXiv:1302.6613 (2013)
2. Ang, J.S., Ng, K.W., Chua, F.F.: Modeling time series data with deep learning: A review, analysis, evaluation and future trend. In: 2020 8th International Conference on Information Technology and Multimedia (ICIMU). pp. 32–37. IEEE (2020)
3. Backhaus, K., Erichson, B., Plinke, W., Weiber, R.: Zeitreihenanalyse, pp. 125–161. Springer Berlin Heidelberg, Berlin, Heidelberg (2018). https://doi.org/10.1007/978-3-662-56655-8_3
4. Dama, F., Sinoquet, C.: Time series analysis and modeling to forecast: a survey. arXiv preprint arXiv:2104.00164 (2021)
5. He, X., Zhao, K., Chu, X.: Automl: A survey of the state-of-the-art. Knowledge-Based Systems **212**, 106622 (2021)
6. Hendikawati, P., et al.: A survey of time series forecasting from stochastic method to soft computing. In: Journal of Physics: Conference Series. vol. 1613, p. 012019. IOP Publishing (2020)
7. Ivanović, M., Kurbalija, V.: Time series analysis and possible applications. In: 2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). pp. 473–479. IEEE (2016)
8. Jebb, A.T., Tay, L., Wang, W., Huang, Q.: Time series analysis for psychological research: examining and forecasting change. Frontiers in psychology **6**, 727 (2015)

9. Santos, T., Kern, R.: A literature survey of early time series classification and deep learning. Sami@ iknow (2016)
10. Schlittgen, R., Streitberg, B.H.: Zeitreihenanalyse. In: Zeitreihenanalyse, chap. 3. Oldenbourg Wissenschaftsverlag (2001)
11. Tay, F.E., Cao, L.: Application of support vector machines in financial time series forecasting. omega **29**(4), 309–317 (2001)