



**FOM Hochschule für Ökonomie und Management**  
Studienzentrum Köln

**Master-Thesis**

Im Studiengang Big Data und Business Analytics

zur Erlangung des Grades eines

Master of Science (M.Sc.)

über das Thema

**Evaluierung eines Algorithmus zur Vorhersage von Stromverbräuchen in  
Baden-Württemberg anhand historischer Wetter- und  
Stromverbrauchsdaten**

von

Arne Decker

Erstgutachter: Daniel Mader M.Sc.  
Matrikelnummer: 528917  
Abgabedatum: 12.02.2022

## Inhaltsverzeichnis

Inhaltsverzeichnis .....	1
Abbildungsverzeichnis .....	4
Tabellenverzeichnis .....	8
Abkürzungsverzeichnis .....	9
Formel- und Symbolverzeichnis .....	11
1 Einleitung .....	12
1.1 Problemstellung .....	12
1.2 Zielsetzung und Abgrenzung .....	13
1.3 Aufbau und Methodik .....	14
2 Stromverbrauch und Zeitreihen .....	15
2.1 Stromnetz .....	16
2.1.1 Stromerzeugung und Verbrauch .....	17
2.1.2 Stabilität im Stromnetz .....	20
2.2 Zusammensetzung von Zeitreihen .....	21
2.3 Korrelationen .....	23
2.3.1 Einfache Autokorrelation .....	24
2.3.2 Partielle Autokorrelation .....	27
2.3.3 Stationarität .....	28
2.4 Autoregressive Integrated Moving Average .....	32
2.4.1 Autoregression (AR) .....	33
2.4.2 Moving Average (MA) .....	36
2.4.3 Autoregressive Integrated Moving Average (ARIMA) .....	40
2.4.4 Seasonal ARIMA (SARIMA) .....	41
2.4.5 Parameterbestimmung gegen Über- und Unteranpassungen .....	43
2.4.6 Kontexteffekte und Kausalitäten (SARIMAX) .....	46

## Inhaltsverzeichnis

2.5 Long Short-Term Memory .....	48
2.5.1 Arbeitsweise und Architektur neuronaler Netze .....	48
2.5.2 Training neuronaler Netze .....	51
2.5.3 Über- und Unteranpassung .....	57
2.5.4 Rekurrente neuronale Netze und LSTM .....	59
2.6 Evaluation und Metriken.....	64
2.6.1 Rolling-Forecast .....	64
2.6.2 Metriken.....	68
3 Erstellung des Vorhersagemodells .....	71
3.1 Vorgehensmodell und Werkzeuge .....	71
3.1.1 Vorgehensmodell.....	71
3.1.2 Python .....	74
3.2 Business Understanding .....	74
3.3 Data Understanding .....	76
3.3.1 Analyse des Stromverbrauchs .....	79
3.3.2 Analyse des Einflusses durch die Temperatur.....	88
3.3.3 Analyse des Einflusses durch die Tagesstunden.....	93
3.3.4 Analyse des Einflusses durch die Luftfeuchtigkeit .....	99
3.3.5 Analyse der weiteren Wetterdaten.....	104
3.4 Data Preparation .....	109
3.5 Modeling (ARIMA).....	111
3.5.1 Parameterbestimmung mit auto_arima() .....	111
3.5.2 Parameterbestimmung mit ACF und PACF .....	114
3.5.3 Erweiterung durch exogene Daten .....	119
3.5.4 Verbesserungsansätze durch weitere exogene Daten .....	122
3.5.5 Zwischenfazit zu ARIMA.....	127
3.6 Modeling (LSTM).....	128

## Inhaltsverzeichnis

3.6.1	LSTM-Schichten .....	131
3.6.2	LSTM- und Dense-Schichten.....	135
3.6.3	Zwischenfazit zu LSTM .....	143
3.7	Evaluation .....	144
3.7.1	Vergleich von ARIMA und LSTM .....	145
3.7.2	Test mit Wettervorhersage .....	146
3.7.3	Modellauswahl und Restriktionen .....	149
3.8	Deployment .....	151
	Zusammenfassung .....	154
	Fazit .....	156
	Anhang .....	157
	Anhang 1: Quelltexte .....	157
	Anhang 2: Verwendete Software .....	164
	Quellenverzeichnis.....	169
	Monographien .....	169
	Aufsätze und Artikel .....	174
	Internet-Quellen.....	175
	Ehrenwörtliche Erklärung .....	188

## Abbildungsverzeichnis

Abbildung 1: Zeitreihenvorhersage .....	15
Abbildung 2: Übertragungsnetzbetreiber in Deutschland .....	16
Abbildung 3: Strommix in Deutschland (2020) .....	18
Abbildung 4: Beispielhafte Komposition einer Zeitreihe .....	22
Abbildung 5: Beispiel für einfaches Autokorrelationsdiagramm.....	25
Abbildung 6: Beispiel für partielle Autokorrelationsdiagramm.....	28
Abbildung 7: Temperatur in Luxemburg .....	29
Abbildung 8: Temperatur in Luxemburg (vor/nach Differencing) .....	31
Abbildung 9: Generierte Zeitreihe (vor/nach Differencing) .....	31
Abbildung 10: Beispiel für Autoregression .....	34
Abbildung 11: Partielles und einfaches Autokorrelationsdiagramm (AR) .....	36
Abbildung 12: Beispiel für Moving Average.....	38
Abbildung 13: Einfaches und partielle Autokorrelationsdiagramm (MA) .....	40
Abbildung 14: Beispiel für SARIMA.....	42
Abbildung 15: Beispiel für Überanpassungen .....	44
Abbildung 16: Aufbau eines neuronalen Netzes .....	48
Abbildung 17: Aufbau und Funktion eines Neurons .....	50
Abbildung 18: Aktivierungsfunktionen .....	50
Abbildung 19: Trainingsprozess neuronaler Netze.....	52
Abbildung 20: Gradientenabstiegsverfahren .....	53
Abbildung 21: Pfad in einem neuronalen Netz .....	54
Abbildung 22: Verschwindende und explodierende Gradienten.....	56
Abbildung 23: Dropout in neuronalem Netz .....	58
Abbildung 24: Neuron in rekurrenter Schicht .....	61
Abbildung 25: Rekurrentes Neuron .....	61
Abbildung 26: Neuron in LSTM-Schicht .....	63
Abbildung 27: Verarbeitungsschritte im Neuron in LSTM-Schicht.....	64
Abbildung 28: Zeitreihe für Rolling-Forecast .....	65
Abbildung 29: Rolling-Forecast (Schritt 1).....	65
Abbildung 30: Rolling-Forecast (Schritt 2).....	66
Abbildung 31: Rolling-Forecast .....	67
Abbildung 32: KDD und SEMMA .....	72
Abbildung 33: CRISP-DM .....	73

## Abbildungsverzeichnis

Abbildung 34: Baseline .....	75
Abbildung 35: Verwendete Städte für Wetterdaten .....	77
Abbildung 36: Stromverbrauch (2015-2021) .....	79
Abbildung 37: Stromverbrauch (gleitender 14-Tage Durchschnitt 2015-2021).....	80
Abbildung 38: Boxplots des Stromverbrauchs nach Monaten .....	81
Abbildung 39: Stromverbrauch 2021 .....	82
Abbildung 40: Stromverbrauch Anfang 2021 .....	83
Abbildung 41: Stromverbrauch nach Wochentagen .....	83
Abbildung 42: Stromverbrauch an Feiertagen nach Wochentagen .....	84
Abbildung 43: Regression mit Arbeitstag .....	85
Abbildung 44: Einfache Autokorrelation des Stromverbrauchs .....	86
Abbildung 45: Partielle Korrelation des Stromverbrauchs .....	87
Abbildung 46: ADF- und KPSS-Test .....	88
Abbildung 47: Korrelationen der Temperaturwerte.....	89
Abbildung 48: Stromverbrauch und Temperatur .....	90
Abbildung 49: Stromverbrauch und Temperatur (gleitender Durchschnitt).....	90
Abbildung 50: Stromverbrauch nach Temperatur (Boxplot) .....	91
Abbildung 51: Stromverbrauch nach Temperatur (Scatterplot) .....	92
Abbildung 52: Regression mit Arbeitstag und Temperatur .....	93
Abbildung 53: Stromverbrauch und Tagesstunden .....	94
Abbildung 54: Stromverbrauch und Tagesstunden (gleitender Durchschnitt) .....	94
Abbildung 55: Tagesstunden und Temperatur .....	95
Abbildung 56: Stromverbrauch nach Tagesstunden (Boxplot) .....	96
Abbildung 57: Stromverbrauch nach Temperatur und Tagesstunden .....	97
Abbildung 58: Regression mit Arbeitstag, Temperatur Tagesstunden .....	98
Abbildung 59: Stromverbrauch und Luftfeuchtigkeit.....	99
Abbildung 60: Stromverbrauch und Luftfeuchtigkeit (gleitender Durchschnitt) ....	100
Abbildung 61: Stromverbrauch nach Luftfeuchtigkeit (Boxplot) .....	101
Abbildung 62: Temperatur und Luftfeuchtigkeit.....	101
Abbildung 63: Stromverbrauch nach Luftfeuchtigkeit u. Temperatur (Scatterplot)	102
Abbildung 64: Regression mit Arbeitstag, Temperatur, Tagesstunden und Luftfeuchtigkeit.....	103
Abbildung 65: Stromverbrauch nach Wetterbedingungen (Boxplot).....	104
Abbildung 66: Wetterbedingungen nach Monaten .....	105

## Abbildungsverzeichnis

Abbildung 67: Stromverbrauch nach Regentagen.....	106
Abbildung 68: Stromverbrauch nach gerundetem Niederschlag .....	106
Abbildung 69: Stromverbrauch nach Windgeschwindigkeit (Boxplot).....	107
Abbildung 70: Stromverbrauch nach gerundeter Sichtweite.....	108
Abbildung 71: Sichtweite nach Monaten .....	108
Abbildung 72: Unskalierte Daten nach Data Preparation .....	110
Abbildung 73: Ergebnisse von auto_arima() .....	111
Abbildung 74: Ergebnisse von ARIMA(2,0,2)(2,0,2)7 .....	113
Abbildung 75: ARIMA(2,0,2)(2,0,2)7 .....	113
Abbildung 76: Einfache und partielle Autokorrelation.....	114
Abbildung 77: ARIMA([1,2,4],0,2)(2,0,2)7 .....	116
Abbildung 78: ARIMA(4,0,4)(4,0,4)7 .....	117
Abbildung 79: Residuen von ARIMA(2,0,2)(2,0,2)7 .....	118
Abbildung 80: ARIMA(2,0,2)(2,0,2) [Arbeitstag, Temperatur, Tagesstunden] .....	121
Abbildung 81: Residuen von ARIMA(2,0,2)(2,0,2) [Arbeitstag, Temperatur] .....	122
Abbildung 82: Residuen von ARIMA(2,0,2)(2,0,2)7 [Arbeitstag, Temperatur, Urlaubssaison] .....	123
Abbildung 83: Residuen von ARIMA(2,0,2)(2,0,2) [Arbeitstag, Temperatur] (neu) .....	124
Abbildung 84: Residuen um Feiertage herum .....	124
Abbildung 85: ARIMA und Baseline .....	127
Abbildung 86: Zeitfenster für Vorhersage des 15.01.2015 .....	130
Abbildung 87: Lernkurve von LSTM-1 .....	132
Abbildung 88: Lernkurve von LSTM-LSTM-3 .....	134
Abbildung 89: Lernkurve von LSTM-DENSE-2 .....	136
Abbildung 90: Lernkurve von DENSE-LSTM-3 .....	137
Abbildung 91: Lernkurve von DENSE-LSTM-5 .....	138
Abbildung 92: Verlustfunktion von DENSE-LSTM-9.....	139
Abbildung 93: Lernkurve von DENSE-LSTM-16 (fertiges Modell) .....	140
Abbildung 94: Residuen von DENSE-LSTM-16 .....	141
Abbildung 95: LSTM und Baseline .....	143
Abbildung 96: ARIMA und LSTM im Vergleich.....	145
Abbildung 97: Residuen von LSTM und ARIMA.....	146
Abbildung 98: Wettervorhersage ARIMA .....	147

## Abbildungsverzeichnis

Abbildung 99: Wettervorhersage LSTM .....	148
Abbildung 100: Oberfläche (Daten laden) .....	152
Abbildung 101: Oberfläche (Vorhersage erstellen) .....	153

## Tabellenverzeichnis

Tabelle 1: Versetzte Zeitreihen für einfache Autokorrelation .....	24
Tabelle 2: Zeitreihe für Autoregression .....	33
Tabelle 3: Zeitreihe für Moving-Average .....	37
Tabelle 4: Zeitreihe mit Zeitfenster.....	60
Tabelle 5: Merkmale in CSV-Datei mit Stromverbrauch .....	76
Tabelle 6: Städte für Wetterdaten .....	77
Tabelle 7: Merkmale der CSV-Datei für die Wetterdaten .....	78
Tabelle 8: ARIMA-Modelle mit endogenen Daten .....	115
Tabelle 9: ARIMA-Modelle mit exogenen Daten .....	120
Tabelle 10: Merkmalsrelevanz bei LSTM .....	142

## Abkürzungsverzeichnis

ACF	autocorrelation function (einfache Autokorrelation)
ADF-Test	Augmented-Dickey-Fuller-Test
AIC	Akaike Information Criterion
AR	Autoregressive
ARIMA	Autoregressive Integrated Moving Average
ARMA	Autoregressive Moving Average
BNetzA	Bundesnetzagentur
CRISP-DM	Cross Industry Standard Process for Data Mining
GmbH	Gesellschaft mit beschränkter Haftung
KDD	Knowledge Discovery in Databases
Log-L.	Log-Likelihood
LSTM	Long Short-Term Memory
MA	Moving-Average
MAE	Mean Absolute Error (mittlerer absoluter Fehler)
MAPE	Mean Absolute Percentage Error (mittlerer absoluter prozentualer Fehler)
MSE	Mean Squared Error (mittlerer quadrierter Fehler)
MWh	Megawattstunden
PACF	partial autocorrelation function (partielle Autokorrelation)
R <sup>2</sup>	Bestimmtheitsmaß
ReLU	Rectified Linear Unit
RMSE	Root Mean Squared Error (Wurzel des mittleren quadrierten Fehlers)
RNN	Rekurrentes neuronales Netz
SARIMA	Seasonal Autoregressive Integrated Moving Average
SARIMAX	Seasonal Autoregressive Integrated Moving Average eXogenous

## Abkürzungsverzeichnis

SEMMA      Sample, Explore, Modify, Model and Assess

tanh          Tangens-Hyperbolicus

TWh          Terawattstunden

**Formel- und Symbolverzeichnis**

Formel 1: Beispielhafte additive Zeitreihe .....	22
Formel 2: Regression für partielle Autokorrelation .....	27
Formel 3: Saisonales Differencing .....	30
Formel 4: Autoregressionsfunktion .....	34
Formel 5: Moving-Average-Funktion.....	38
Formel 6: ARIMA-Funktion .....	41
Formel 7: Formel für Pfad im neuronalen Netz .....	54
Formel 8: Veränderung eines Gewichts durch Gradienten und Lernrate .....	55
Formel 9: L1- und L2-Regularisierung.....	58
Formel 10: Bestimmtheitsmaß R <sup>2</sup> .....	68
Formel 11: Mittlerer absoluter Fehler MAE .....	69
Formel 12: Mittlerer quadrierter Fehler MSE .....	69
Formel 13: Wurzel des mittleren quadrierten Fehlers RMSE .....	70
Formel 14: Mittlerer absoluter prozentualer Fehler MAPE .....	70

## 1 Einleitung

Im Jahr 2020 wurden weltweit insgesamt etwa 23.100 Terrawattstunden Strom verbraucht. Das entspricht einer Steigerung um 23% seit dem Jahr 2010.<sup>1</sup> Bis 2030 wird der Stromverbrauch nach unterschiedlichen Schätzungen um 16% bis 25% auf 27.000<sup>2</sup> bis 29.000<sup>3</sup> Terrawattstunden ansteigen. Geräte, die mit elektrischem Strom betrieben werden, sind in allen Bereichen einer modernen Gesellschaft und Volkswirtschaft fest integriert. Aus diesem Grund zählt die Stromversorgung zur kritischen Infrastruktur.<sup>4</sup> Stomausfälle können das wirtschaftliche<sup>5</sup> und gesellschaftliche<sup>6</sup> Leben innerhalb weniger Stunden zum Erliegen bringen.<sup>7</sup>

### 1.1 Problemstellung

Die Stabilität des Stromnetzes ist essenziell wichtig. Strom lässt sich allerdings nur in begrenztem Maß speichern. Deshalb muss möglichst immer exakt so viel Strom ins Netz eingespeist werden, wie gleichzeitig aus dem Netz entnommen wird.<sup>8</sup>

Die Stromanbieter müssen daher den Stromverbrauch ihrer Kunden prognostizieren und gleichzeitig planen, wie der entsprechende Strom in das Netz eingespeist wird, beispielsweise aus eigenen Quellen oder durch Zukauf von anderen Anbietern. Diese Prognosen müssen dann beim Übertragungsnetzbetreiber gemeldet werden.<sup>9</sup> Wenn der Stromverbrauch die Einspeisung über- oder unterschreitet, muss der Übertragungsnetzbetreiber gegebenenfalls sogenannte Regelenergie in das Netz einspeisen. Falls die Stromeinspeisung niedriger als der Verbrauch ist, werden Reservequellen aktiviert, um Stomausfälle zu vermeiden.<sup>10</sup> Falls die Einspeisung den Stromverbrauch überschreitet, werden zusätzliche Verbrauchskapazitäten (beispielsweise in stromintensiven Industrien) aktiviert, um das Netz zu entlasten.<sup>11</sup>

Derartige Eingriffe zur Stabilisierung des Stromnetzes sind allerdings teuer und nur in begrenztem Maße möglich. Um die Stabilität und die Effizienz des Stromnetzes

---

<sup>1</sup> Vgl. *Statista*, Nettostromverbrauch weltweit, 2021.

<sup>2</sup> Vgl. *Internationale Energieagentur*, Stromverbrauchsprognose weltweit 2030, 2020.

<sup>3</sup> Vgl. *Enerdata*, Stromverbrauchsprognose weltweit 2050, 2022.

<sup>4</sup> Vgl. *Bundesamt für Bevölkerungsschutz und Katastrophenhilfe*, Kritische Infrastruktur, 2021.

<sup>5</sup> Vgl. *Shuai, M et al.*, Folgen von Stomausfällen, 2018.

<sup>6</sup> Vgl. *Bundesamt für Bevölkerungsschutz und Katastrophenhilfe*, Stromausfall, 2019.

<sup>7</sup> Vgl. *Niederhausen, H., Burkert, A.*, Elektrischer Strom, 2014, S. 24 ff., S. 110.

<sup>8</sup> Vgl. *Bundesnetzagentur*, Stromverbrauch, 2021.

<sup>9</sup> Vgl. *Bundesnetzagentur*, Ausgleichsenergie, 2022.

<sup>10</sup> Vgl. *Bundesnetzagentur*, Regelreserve, 2022.

<sup>11</sup> Vgl. *Paschotta, Dr. R.*, Regelenergie, 2021.

zu gewährleisten, sollten die entsprechenden Kapazitäten exakt geplant werden, damit so wenig wie möglich auf Regelenergie zurückgegriffen werden muss. Für diese Planung ist eine präzise Vorhersage des Stromverbrauches essenziell wichtig.<sup>12</sup>

## 1.2 Zielsetzung und Abgrenzung

Ziel der vorliegenden Arbeit ist die Entwicklung eines Vorhersagemodells, mit dem Prognosen über den täglichen Stromverbrauch des Bundeslandes Baden-Württemberg gemacht werden können. Als Grundlage dienen dafür historische Stromverbrauchsdaten. Außerdem werden historische Wetterdaten auf den Zusammenhang mit dem Stromverbrauch analysiert.

Das Vorhersagemodell soll anhand des historischen Stromverbrauchs sowie der Wettervorhersage eine Prognose über den täglichen Stromverbrauch im gesamten Bundesland abgeben. Die Prognose über den Stromverbrauch kann dann für die Planung der Stromerzeugung oder der Regelreserven verwendet werden.

Die der Arbeit zugrundeliegende Forschungsfrage lautet:

Inwieweit lässt sich der tägliche Stromverbrauch des Bundeslandes Baden-Württemberg anhand historischer Stromverbrauchs- und Wetterdaten vorhersagen?

Besonderer Fokus liegt dabei auf den folgenden Aspekten:

1. Untersuchung der Eigenschaften der Zeitreihe des Stromverbrauchs
2. Untersuchung der Abhängigkeit des Stromverbrauchs von den Wetterdaten
3. Ermittlung einer geeigneten Methodik zur Vorhersage des Stromverbrauchs
4. Evaluation der Vorhersage

Die Untersuchung und das Vorhersagemodell beziehen sich nur auf den Stromverbrauch. Es soll im Rahmen dieser Untersuchung keine Prognose über die Stromerzeugung oder den daraus resultierenden Bedarf an Regelenergie erstellt werden. Das Modell soll nur zusätzliche Informationen für derartige Prognosen liefern können. Es handelt sich um eine Analyse und Prognose des Nettostromverbrauchs<sup>13</sup>, der von Kraftwerken und anderen Stromquellen zur Stromerzeugung selbst benötigte Stromverbrauch wird also nicht betrachtet.

---

<sup>12</sup> Vgl. *Bundesnetzagentur*, Regelreserve, 2022.

<sup>13</sup> Vgl. *Bundesministerium für Wirtschaft und Energie*, Bruttostromverbrauch, 2016.

### 1.3 Aufbau und Methodik

Die Arbeit gliedert sich in zwei Teile: Im ersten Teil werden die notwendigen theoretischen Grundlagen mit Blick auf das Stromnetz erläutert. Weiterhin werden Verfahren zur Analyse, Verarbeitung und Vorhersage von Zeitreihen mittels maschinellen Lernens dargestellt. Dafür wird eine Literaturrecherche durchgeführt.

Im zweiten Teil wird die Erstellung des Vorhersagemodells gezeigt. Sie wird nach dem „Cross Industry Standard Process for Data Mining“ (CRISP-DM)<sup>14</sup> durchgeführt, daher gliedert sich der zweite Teil analog dazu in die einzelnen Phasen:

1. Festlegung und Darstellung der Zielsetzung
2. Beschaffung und Analyse der Daten (Zeitreihen-, Korrelations- und Regressionsanalyse, statistische Tests etc.)
3. Vorverarbeitung der Daten für die Modellierung
4. Modellierung und Optimierung der Modelle
5. Evaluation der Modelle und Vergleich untereinander
6. Deployment beziehungsweise Einbindung in eine Benutzeroberfläche

Die Stromverbrauchsdaten werden von der Bundesnetzagentur (BNetzA, Bonn) bezogen. Die Wetterdaten werden für die vier Städte Stuttgart, Freiburg, Mannheim und Ulm von WeatherAPI.com bezogen. Die Daten fassen den Zeitraum vom 01.01.2015 bis zum 31.12.2021.

Es handelt sich um eine quantitative Untersuchung mit einem induktiven Forschungsansatz.

---

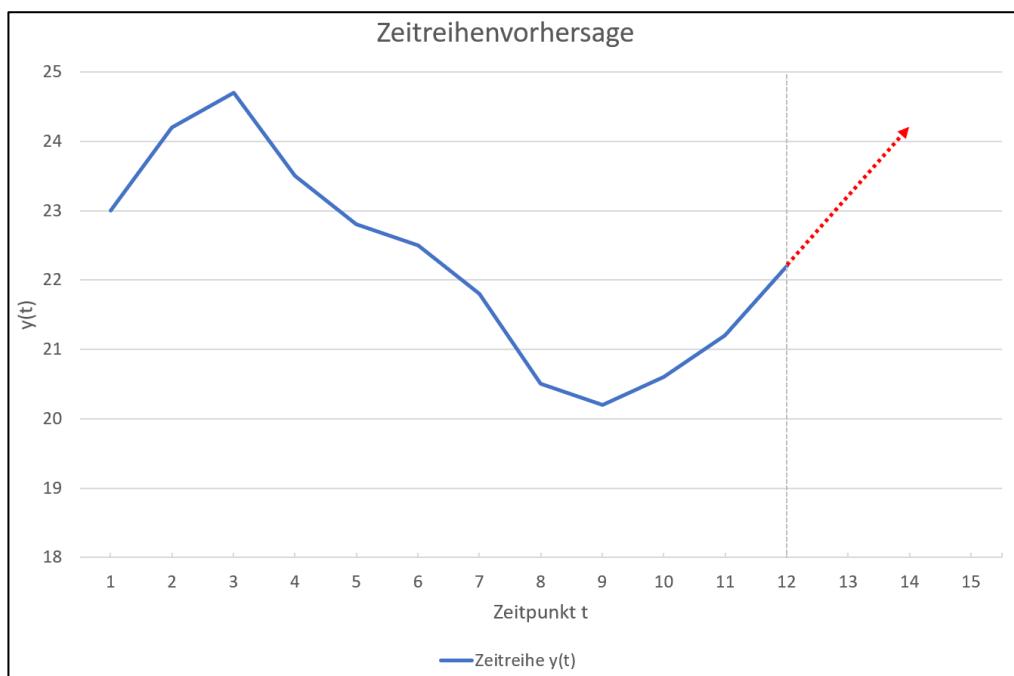
<sup>14</sup> Vgl. Nisbet, R., Yale, K., Miner, G., Statistische Analysen, 2017, S. 40 ff.

## 2 Stromverbrauch und Zeitreihen

„Time series forecasting is the process of analyzing time series data using statistics and modeling to make predictions [...]“<sup>15</sup>. So definiert das für seine gleichnamige, weit verbreitete Visualisierung- und Analysesoftware bekannte Unternehmen Tableau (Seattle, USA) das Time Series Forecasting (deutsch „Zeitreihenvorhersage“).

Es handelt sich dabei um einen Teilbereich des maschinellen Lernens und damit der künstlichen Intelligenz, welcher eng mit der Regression verbunden ist und dem überwachten Lernen zugeordnet wird.<sup>16</sup> Anders als bei der einfachen Regression werden einzelne Beobachtungen nicht unabhängig voneinander, sondern im Kontext beziehungsweise unter Berücksichtigung des gesamten Verlaufs der Zeitreihe betrachtet und verarbeitet.<sup>17</sup> So zeigt die Abbildung 1 eine beispielhafte **Zeitreihe  $y$**  an den Zeitpunkten  $t$ . Wenn nun der Verlauf der Zeitreihe vom letzten bekannten Punkt  $t = 12$  aus für **die nächsten drei Schritte vorhergesagt werden soll (rot markiert)**, dann ist dies wesentlich vom vorherigen Verlauf der Zeitreihe abhängig.<sup>18</sup>

Abbildung 1: Zeitreihenvorhersage



Quelle: In Anlehnung an *Box et al.*, Time Series Analysis, 2016, S. 3.

<sup>15</sup> Tableau Software, Time Series Forecasting, 2022.

<sup>16</sup> Vgl. Lazzeri, F., Machine Learning, 2021, S. 104 ff. und 14 ff.

<sup>17</sup> Vgl. *Box et al.*, Time Series Analysis, 2016, S. 1 ff.

<sup>18</sup> Beispiel in Anlehnung an *Box et al.*, Time Series Analysis, 2016, S. 1 ff.

Im folgenden Kapitel werden die theoretischen Grundlagen für die Zeitreihenvorhersage erläutert. Zunächst wird der fachliche Kontext dargestellt. Dazu gehören die grundlegende Funktionsweise des Stromnetzes mit Fokus auf Erzeugung und Verbrauch sowie Maßnahmen zur Stabilisierung des Stromnetzes. Außerdem werden Zeitreihen mit ihren wesentlichen Eigenschaften beschrieben und relevante Analyseverfahren dargestellt. Zusätzlich werden gängige Algorithmen und Methoden zur Zeitreihenvorhersage beschrieben.

## 2.1 Stromnetz

Das Stromnetz umfasst die notwendige Infrastruktur für die Aufnahme bzw. Einspeisung, den Transport und die Verteilung von elektrischem Strom. Es gliedert sich in zwei Teile: die Übertragungsnetze und die Verteilungsnetze. Übertragungsnetze werden für den Transport von Strom über große Distanzen verwendet.<sup>19</sup> Im deutschen Stromnetz gibt es vier Übertragungsnetze (auch Regelzonen genannt):

- Die TenneT GmbH (Bayreuth) betreibt das Übertragungsnetz für Schleswig-Holstein, Niedersachsen, Bremen und einen Großteil Bayerns, Hessens sowie Teile Nordrhein-Westfalens. Ihr Netz umfasst 24.000 Kilometer Hochspannungsleitungen und ist damit das größte Übertragungsnetz in Deutschland.<sup>20</sup>
- Die Amprion GmbH (Dortmund) ist für das Übertragungsnetz für Rheinland-Pfalz, das Saarland einen Großteil Nordrhein-Westfalens und einen Teil Bayerns zuständig. Mit 11.000 Kilometern Hochspannungsleitungen ist Amprion der zweitgrößte deutsche Übertragungsnetzbetreiber.<sup>21</sup>
- Die 50Hertz GmbH (Berlin) deckt das Netz für Mecklenburg-Vorpommern, Brandenburg, Berlin, Sachsen-Anhalt, Sachsen, Thüringen und Hamburg ab und ist mit 10.000 Kilometern der drittgrößte Übertragungsnetzbetreiber.<sup>22</sup>
- Die TransnetBW GmbH (Stuttgart) betreibt das Übertragungsnetz für Baden-Württemberg und ist mit einem Netz von 3.200 Kilometern Länge der kleinste deutsche Übertragungsnetzbetreiber.<sup>23</sup>

**Abbildung 2: Übertragungsnetzbetreiber in Deutschland**

---

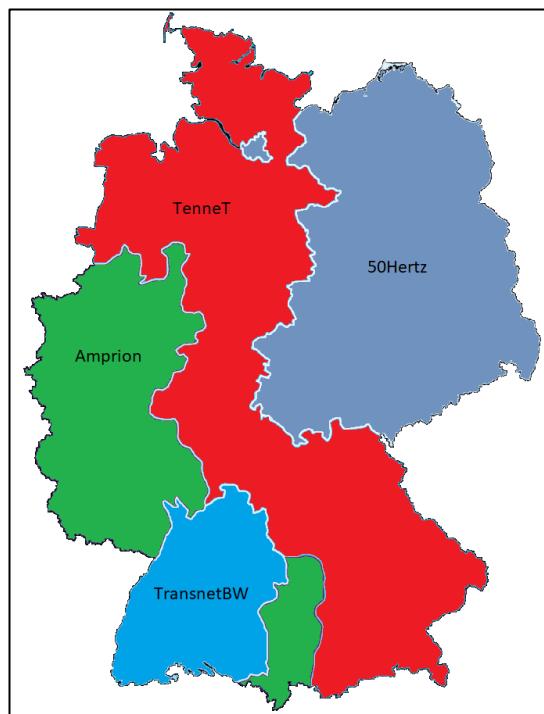
<sup>19</sup> Vgl. Bundesministerium für Wirtschaft und Energie, Stromnetz, 2022.

<sup>20</sup> Vgl. TenneT GmbH, TenneT, 2022.

<sup>21</sup> Vgl. Amprion GmbH, Amprion, 2021.

<sup>22</sup> Vgl. 50Hertz GmbH, 50Hertz, 2021.

<sup>23</sup> Vgl. TransnetBW GmbH, TransnetBW, 2021.



Quelle: In Anlehnung an Schiffer, H.-W., Energiemarkt Deutschland, 2018, S. 165.

Innerhalb der Übertragungsnetze wird die Feinverteilung des Stroms an Endverbraucher wie die Industrie oder private Haushalte über Verteilungsnetze geregelt. In Deutschland gibt es insgesamt etwa 870 Verteilungsnetzbetreiber.<sup>24</sup>

### 2.1.1 Stromerzeugung und Verbrauch

Im deutschen Stromnetz wurden im Jahr 2020 circa 488 Terrawattstunden (TWh) Strom verbraucht.<sup>25</sup> Davon entfielen 45% auf die Industrie, 27% auf Gewerbe, 26% auf Haushalte und 2% auf Verkehr.<sup>26</sup> Demgegenüber wurden 544 TWh ins Netz eingespeist<sup>27</sup> und 21 TWh ins Ausland exportiert<sup>28</sup>. Dazu kommen weiterhin Netzverluste, die bei der Übertragung vom Strom durch beispielsweise Abwärme oder Ladungsverluste entstehen.<sup>29</sup> Die Stromquellen werden anhand ihrer Energieträger in zwei Kategorien unterteilt: Konventionelle und erneuerbare Energiequellen.<sup>30</sup>

<sup>24</sup> Vgl. Statista, Anzahl Stromnetzbetreiber, 2021.

<sup>25</sup> Vgl. Statista, Nettostromverbrauch, 2022.

<sup>26</sup> Vgl. Bundesverband der Energie- und Wasserwirtschaft, Stromverbrauch, 2021.

<sup>27</sup> Vgl. Statista, Nettostromerzeugung, 2021.

<sup>28</sup> Vgl. Statista, Stromtauschsaldo, 2020.

<sup>29</sup> Vgl. TenneT GmbH, Netzverluste, 2021.

<sup>30</sup> Vgl. Bundesnetzagentur, Stromerzeugung, 2022.

Die konventionelle Energiegewinnung basiert größtenteils auf der Verbrennung fossiler Energieträger wie Kohle, Erdöl oder Erdgas. Dabei handelt es sich um eine zuverlässige und stabile Energiequelle, bei der die Erzeugung flexibel und sehr gut planbar ist. Allerdings entstehen bei der Verbrennung der Energieträger umweltschädliche Abfallprodukte wie Kohlenstoffdioxid, Stickoxide<sup>31</sup> oder im Falle der Kernenergie auch radioaktive Abläufe, die aufwändig entsorgt werden und bei denen sich die Endlagerproblematik stellt<sup>32</sup>.

Die erneuerbare Energiegewinnung nutzt Energieträger wie Wind-, Wasser- und Sonnenenergie oder Biomasse. Im Gegensatz zu den konventionellen Energiequellen entstehen bei der Erzeugung erneuerbarer Energien signifikant weniger umweltschädliche Emissionen.<sup>33</sup> Aus Gründen des Umweltschutzes tragen erneuerbare Energien im Rahmen der Energiewende einen wachsenden Anteil zum Strommix in Deutschland bei.<sup>34</sup> So ist deren Anteil am Strommix von 19,3% im Jahr 2010<sup>35</sup> auf etwa 50% im Jahr 2020<sup>36</sup> gestiegen.

**Abbildung 3: Strommix in Deutschland (2020)**

---

<sup>31</sup> Vgl. *Bundesministerium für Wirtschaft und Energie*, Konventionelle Energieträger, 2022.

<sup>32</sup> Vgl. Schiffer, H.-W., Energiemarkt Deutschland, 2018, S. 204 ff. und 213 ff.

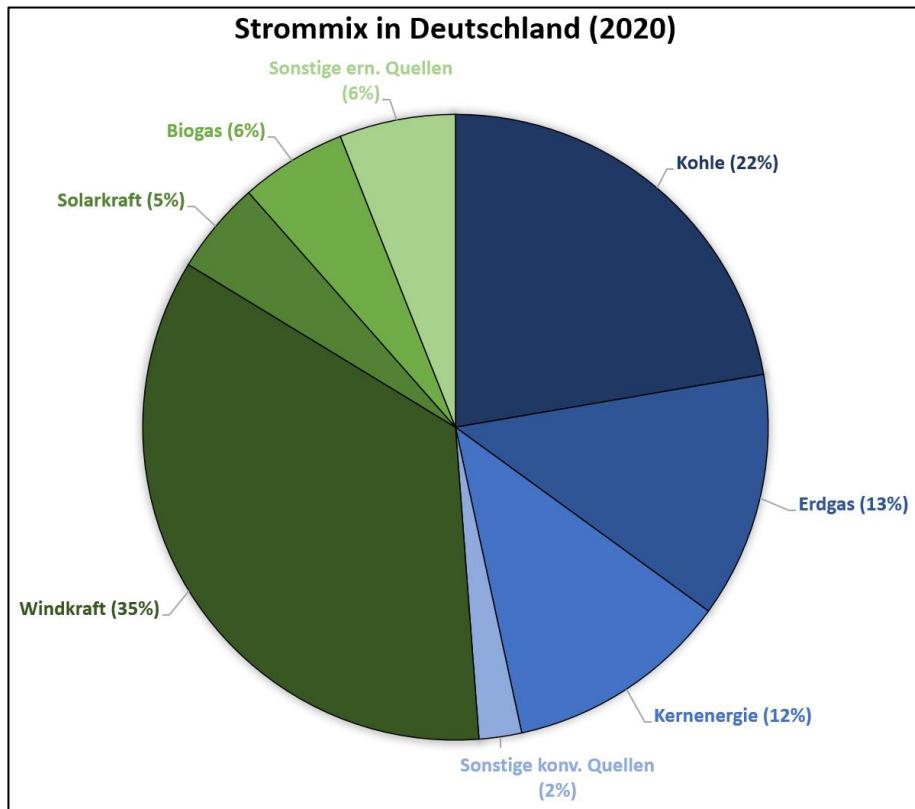
<sup>33</sup> Vgl. *Bundesministerium für Wirtschaft und Energie*, Erneuerbare Energien, 2022.

<sup>34</sup> Vgl. *Bundesnetzagentur*, Stromerzeugung, 2022.

<sup>35</sup> Vgl. *Statista*, Strommix, 2017.

<sup>36</sup> Vgl. *Fraunhofer ISE*, Strommix, 2021.

## Stromverbrauch und Zeitreihen



Quelle: In Anlehnung an *Statistisches Bundesamt (Destatis)*, Energieerzeugung, 2022.

Bis 2030 soll der Anteil erneuerbarer Energien bereits 65% sein.<sup>37</sup> Im Gegensatz zu den konventionellen Energiequellen sind viele erneuerbare Energiequellen unter anderem stark von meteorologischen Faktoren abhängig, die nicht beeinflussbar und nur begrenzt prognostizierbar sind. So hängt beispielsweise die Produktion von Strom durch Windkraft stark von der lokalen Windgeschwindigkeit am Standort des entsprechenden Windparks ab.<sup>38</sup> Die Produktion von Strom durch Solarkraft ist abhängig von der Anzahl an Sonnenstunden und der Strahlungsintensität am Tag, wohingegen nachts praktisch gar kein Strom produziert werden kann.<sup>39</sup>

Die Aspekte der schweren Planbarkeit und starken, bedingt beinflussbaren Volatilität der erneuerbaren Energiequellen stellen eine zusätzliche Herausforderung an die Stabilität des Stromnetzes dar. Bei für Solar- und Windkraft günstigen Bedingungen kann es zu Überlastungen des Netzes kommen. Umgekehrt kann es bei ungünstigen Bedingungen zu Strommangel und damit verbunden zu Stromausfällen kommen.<sup>40</sup>

<sup>37</sup> Vgl. *Die Bundesregierung*, Energiewende, 2022.

<sup>38</sup> Vgl. Schiffer, H.-W., Energiemarkt Deutschland, 2018, 304 ff.

<sup>39</sup> Vgl. *Bundesnetzagentur*, Stromerzeugung im Sommer, 2017.

<sup>40</sup> Vgl. Schiffer, H.-W., Energiemarkt Deutschland, 2018, 304 ff.

### 2.1.2 Stabilität im Stromnetz

Für die Stabilität im Stromnetz müssen Verbrauch und Einspeisung möglichst gleich sein. Zu diesem Gleichgewicht tragen verschiedene Akteure bei.<sup>41</sup>

Das Stromnetz wird auf der untersten Ebene in sogenannten Bilanzkreisen organisiert. Ein Bilanzkreis gehört zu einem Bilanzkreisverantwortlichen (beispielsweise ein Stromanbieter), der darin die ihm zugehörigen Verbraucher und Stromquellen erfasst. Grundsätzlich muss der Bilanzkreisverantwortliche exakt so viel Strom bereitstellen, wie seine Verbraucher aus dem Netz entnehmen. Die Bereitstellung kann durch Erzeugung in eigenen Stromquellen oder durch Zukauf von anderen Stromanbietern erfolgen. In jedem Fall muss der Bilanzkreisverantwortliche dem Übertragungsnetzbetreiber den erwarteten Verbrauch und die erwartete Einspeisung für den jeweils nächsten Tag melden. Beim Übertragungsnetzbetreiber werden die Prognosen dann für die jeweilige Regelzone zusammengefasst.<sup>42</sup>

Der Übertragungsnetzbetreiber wacht über die Stabilität innerhalb seiner Regelzone.<sup>43</sup> Wenn es zu Abweichungen von prognostizierten Soll- und tatsächlichen Ist-Werten innerhalb eines Bilanzkreises kommt, kann diese Abweichung auf zwei Arten abgefangen werden. Zum einen können Abweichungen einzelner Bilanzkreise im Regelzonensaldo ausgeglichen werden. Speist beispielsweise ein Bilanzkreis zu viel und ein weiterer Bilanzkreis zu wenig Strom ins Netz ein, gleichen sich die beiden Bilanzkreise gegebenenfalls aus. Dieser Stromfluss zwischen den Bilanzkreisen wird als Ausgleichsenergie bezeichnet.<sup>44</sup> Falls bei Abweichungen nicht genügend Ausgleichsenergie zur Verfügung steht und das Regelzonensaldo nicht ausgeglichen werden kann, springt der Übertragungsnetzbetreiber der jeweiligen Regelzone ein, indem positive oder negative Regelenergie in das Netz eingespeist wird.<sup>45</sup> Wenn der Stromverbrauch die Stromerzeugung überschreitet, wird positive Regelenergie eingespeist. Dabei handelt es sich um Reservequellen wie etwa speziell dafür reservierte Kapazitäten in Großkraftwerken oder Pumpspeicherwerkten, die schnell aktiviert werden können. Falls die Erzeugung hingegen den Verbrauch übersteigt, wird negative Regelenergie eingespeist. Dabei kann es sich um die Reduktion der

---

<sup>41</sup> Vgl. *Bundesnetzagentur*, Regelenergie, 2022.

<sup>42</sup> Vgl. *Bundesnetzagentur*, Ausgleichsenergie, 2022.

<sup>43</sup> Vgl. *Bundesnetzagentur*, Regelreserve, 2022.

<sup>44</sup> Vgl. *Bundesnetzagentur*, Ausgleichsenergie, 2022.

<sup>45</sup> Vgl. *Bundesnetzagentur*, Regelreserve, 2022.

Einspeisung aus flexiblen Stromquellen wie Kraftwerken handeln, alternativ kann auch kurzfristig der Verbrauch durch beispielsweise industrielle Großkunden erhöht werden, um das Netz zu entlasten.<sup>46</sup>

Der Übertragungsnetzbetreiber reserviert die entsprechenden Regelenergien. Falls ein Bilanzkreis auf Regelenergie zurückgreifen muss, werden die Kosten auf den Bilanzkreisverantwortlichen umgelegt. Die Kosten für die nicht abgerufene Regelenergie trägt der Übertragungsnetzbetreiber. Weiterhin sind die Regelreserven nur begrenzt verfügbar, daher ist jeder notwendige Zugriff darauf ein potenzielles Risiko für die Stabilität des Stromnetzes. Es ist also im Interesse der Übertragungsnetzbetreiber, möglichst präzise Prognosen des Stromverbrauchs und der Stromerzeugung zu erstellen, um die Kapazitäten und Regelreserven optimal zu planen.<sup>47</sup>

## 2.2 Zusammensetzung von Zeitreihen

Eine Zeitreihe  $y$  fasst nach Zeitpunkten  $t$  geordnete Werte einer Variable oder eines Merkmals, die als Beobachtungen  $y(t)$  bezeichnet werden. In der Regel haben die Beobachtungen den gleichen zeitlichen Abstand zwischeneinander. Eine univariate Zeitreihe fasst Beobachtungen eines Merkmals, multivariate Zeitreihen fassen Beobachtungen mehrerer Merkmale am jeweiligen Zeitpunkt.<sup>48</sup>

Eine Zeitreihe besteht aus verschiedenen Komponenten:<sup>49</sup>

1. Level: Die Beobachtungen jeder Zeitreihe bewegen sich auf einem bestimmten Niveau. Dieses Niveau wird als Level bezeichnet. Üblicherweise entspricht das Level dem Durchschnitt oder dem Median der Zeitreihe.<sup>50</sup>
2. Trend: Ein Trend ist eine langfristige und fortdauernde Änderung der Beobachtungen in eine bestimmte Richtung (z.B. aufwärts oder abwärts).<sup>51</sup>
3. Saisonalität: Bestimmte Muster oder Bewegungen, die sich über längere Zeiträume wiederholen, werden als Saisonalität bezeichnet. Die saisonalen Perioden sind gleichlang und deren Länge und ungefähre Auswirkung auf die Zeitreihe sind dabei bekannt.<sup>52</sup>

---

<sup>46</sup> Vgl. Paschotta, Dr. R., Regelenergie, 2021.

<sup>47</sup> Vgl. Bundesnetzagentur, Regelreserve, 2022.

<sup>48</sup> Vgl. Patel, A., Vishwas, B. V., Time Series Analysis Python, 2020, S. 1 ff.

<sup>49</sup> Vgl. Miles, T., Applied Time Series Analysis, 2019, S. 1 ff.

<sup>50</sup> Vgl. Hyndman, R. J., Athanasopoulos, G., Forecasting, S. 31 ff.

<sup>51</sup> Vgl. Auffarth, B., Machine Learning for Time Series, 2021, S. 4 ff. und S. 56 ff.

<sup>52</sup> Vgl. Hyndman, R. J., Athanasopoulos, G., Forecasting, S. 31 ff.

4. Residuen: Alle übrigen irregulären, zufälligen oder sonstigen Einflüsse auf die Zeitreihe, die keiner anderen Komponente zugeordnet werden können, werden als Residuen oder Rauschen bezeichnet.<sup>53</sup>
5. Zyklizität: Bei der Zyklizität handelt es sich ähnlich wie bei der Saisonalität ebenfalls um wiederkehrende Muster in den Daten. Anders als bei der Saisonalität sind Länge und Einfluss der einzelnen zyklischen Perioden unbekannt und gegebenenfalls stark unterschiedlich. Daher wird die Zyklizität meist (und auch im Folgenden) unter den Residuen zusammengefasst.<sup>54</sup>

Die Beobachtungen einer Zeitreihe setzen sich aus den Komponenten am jeweiligen Zeitpunkt zusammen. Jede Zeitreihe besteht mindestens aus einem Level und Residuen. Trend und Saisonalität sind nicht zwangsläufig in jeder Zeitreihe vorhanden.<sup>55</sup> Die Art der Zusammensetzung der Komponenten zu einer Beobachtung hängt von der Beschaffenheit der Zeitreihe ab. Beispielsweise wird bei einer additiven Zeitreihe  $y$  die Summe der Komponenten am Zeitpunkt  $t$  gebildet:

**Formel 1: Beispielhafte additive Zeitreihe**

$$y(t) = \text{Level}(t) + \text{Trend}(t) + \text{Saisonalität}(t) + \text{Residuen}(t) \quad (1)$$

Quelle: In Anlehnung an Shmueli, G., Lichtendahl, K. C., Time Series Forecasting, 2016, S. 28.

Die Abbildung 4 zeigt eine beispielhafte Komposition einer additiven Zeitreihe. Es handelt sich dabei um die monatliche Anzahl an Passagieren einer US-amerikanischen Fluglinie im Zeitraum von 1950 bis 1959. In der ersten Grafik ist die originale Zeitreihe abgebildet. Die zweite und dritte Grafik zeigen jeweils den klar erkennbaren Aufwärtstrend und die deutliche, jährliche Saisonalität. Alle übrigen Einflüsse sind in den Residuen in der letzten Grafik abgebildet:<sup>56</sup>

**Abbildung 4: Beispielhafte Komposition einer Zeitreihe**

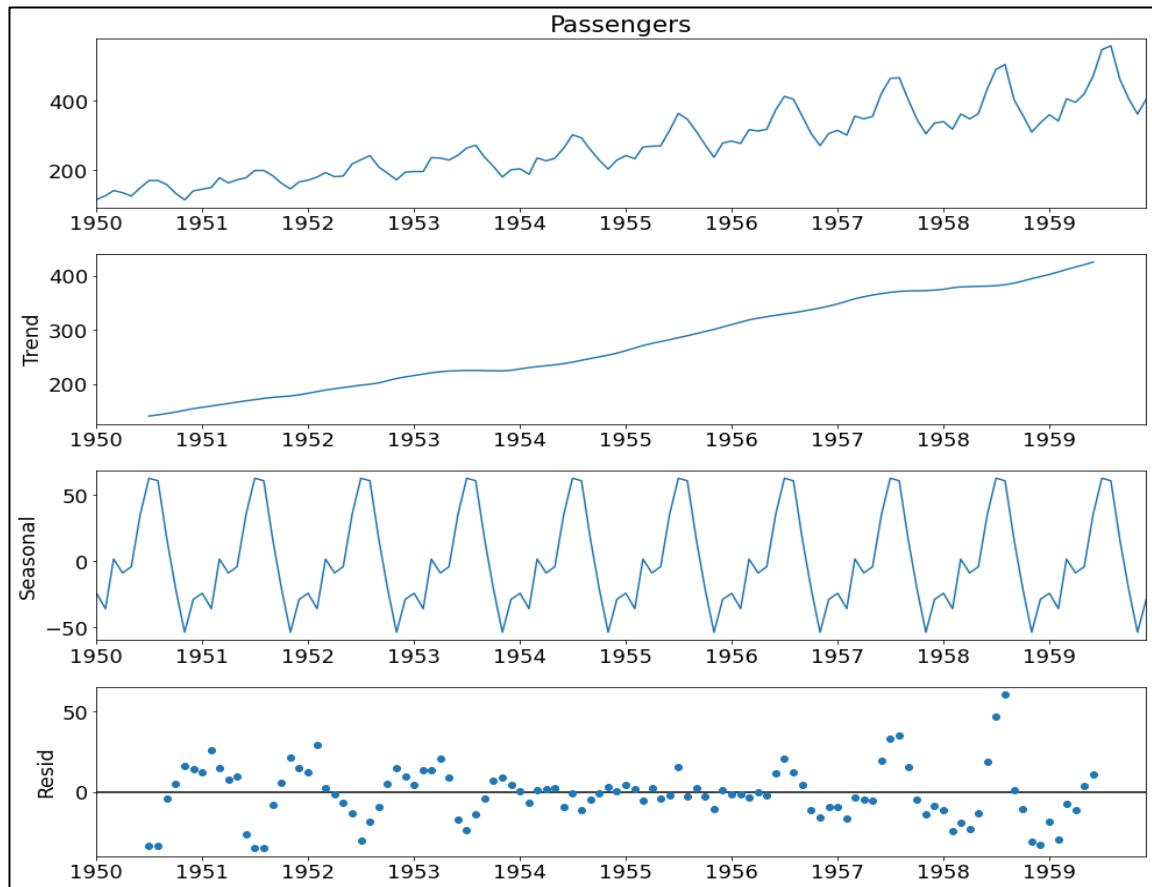
---

<sup>53</sup> Vgl. Chatfield, C., Xing, H., Time Series Analysis, 2019, S. 16 f.

<sup>54</sup> Vgl. Patel, A., Vishwas, B. V., Time Series Analysis Python, 2020, S. 16 ff.

<sup>55</sup> Vgl. Shmueli, G., Lichtendahl, K. C., Time Series Forecasting, 2016, S. 28.

<sup>56</sup> Vgl. Brownlee, J., Time Series Decomposition, 2020.



Quelle: In Anlehnung an Brownlee, J., Time Series Decomposition, 2020.

Das Level, der Trend und die Saisonalität sind sogenannte systematische Komponenten. Sie folgen einem bestimmten Muster und sind daher prognostizierbar. Systematische Komponenten können für Vorhersagemodelle verwendet werden. Im Gegensatz dazu sind die Residuen irregulär oder zufällig ausgeprägt, es handelt sich daher um eine nicht systematische Komponente, welche sich nicht oder wesentlich weniger für Vorhersagen eignet.<sup>57</sup> Zeitreihen, welche sich zu einem hohen Anteil durch systematische Komponenten erklären lassen, können besser durch entsprechende Modelle abgebildet werden und sind daher einfacher vorherzusagen.<sup>58</sup>

## 2.3 Korrelationen

Korrelationen messen die Stärke statistischer Beziehungen zweier Variablen zueinander. Wenn sich zwei Variablen in einem bestimmten Maß in die gleiche Richtung bewegen, spricht man von positiver Korrelation. Umgekehrt werden

<sup>57</sup> Vgl. Shmueli, G., Lichtendahl, K. C., Time Series Forecasting, 2016, S. 28.

<sup>58</sup> Vgl. Athiyarath, S., Paul, M. Krishnaswamy, S., Forecasting Techniques, 2020.

entgegengesetzte Bewegungen als negative Korrelation bezeichnet.<sup>59</sup> Die konkrete Messung der Stärke erfolgt durch die Berechnung von Korrelationskoeffizienten. Ein Korrelationskoeffizient liegt meist zwischen -1 und +1, wobei +1 einer perfekten positiven und -1 einer perfekten negativen Korrelation entspricht. Liegt der Korrelationskoeffizient bei 0, sind die Werte unkorreliert.<sup>60</sup> Der Korrelationskoeffizient nach Pearson misst den linearen Zusammenhang zwischen zwei Variablen. Die Rangkorrelationskoeffizienten nach Spearman und Kendall können auch nicht-lineare Zusammenhänge erfassen.<sup>61</sup>

Die Autokorrelation ist eine besondere Form der Korrelation. Dabei wird nicht die statistische Beziehung zweier unterschiedlicher Variablen untersucht, sondern der Einfluss von vorherigen Beobachtungen auf Beobachtungen einer Zeitreihe. Es wird also die Korrelation einer Zeitreihe mit sich selbst untersucht.<sup>62</sup>

### 2.3.1 Einfache Autokorrelation

Bei der einfachen Autokorrelation (englisch „autocorrelation function“ bzw. ACF) wird untersucht, wie eine Beobachtung von vorherigen Beobachtungen einer Zeitreihe beeinflusst wird. Aus einer Zeitreihe  $y(t)$  lassen sich weitere, zeitversetzte Zeitreihen  $y_1(t) = y(t-1)$ ,  $y_2(t) = y(t-2)$ , ...,  $y_n(t) = y(t-n)$  bilden. Eine vorherige Beobachtung wird auch als Lag bezeichnet, so ist die unmittelbar vorherige Beobachtung  $y(t-1)$  das erste Lag,  $y(t-2)$  das zweite Lag und so weiter.<sup>63</sup>

**Tabelle 1: Versetzte Zeitreihen für einfache Autokorrelation**

t	$y(t)$	$y_1(t)$	$y_2(t)$
1	112	-	-
2	118	112	-
3	132	118	112
4	129	132	118
5	121	129	132

<sup>59</sup> Vgl. Statista, Korrelation, 2021.

<sup>60</sup> Vgl. Statista, Korrelationskoeffizient, 2021.

<sup>61</sup> Vgl. Statistics Solutions, Korrelationskoeffizienten, 2022.

<sup>62</sup> Vgl. Lazzeri, F., Machine Learning, 2021, S. 104 ff.

<sup>63</sup> Vgl. Patel, A., Vishwas, B. V., Time Series Analysis Python, 2020, S. 116 ff.

6	135	121	129
7	148	135	121
...	...	...	...

Quelle: In Anlehnung an *Brownlee, J.*, Time Series Decomposition, 2020.

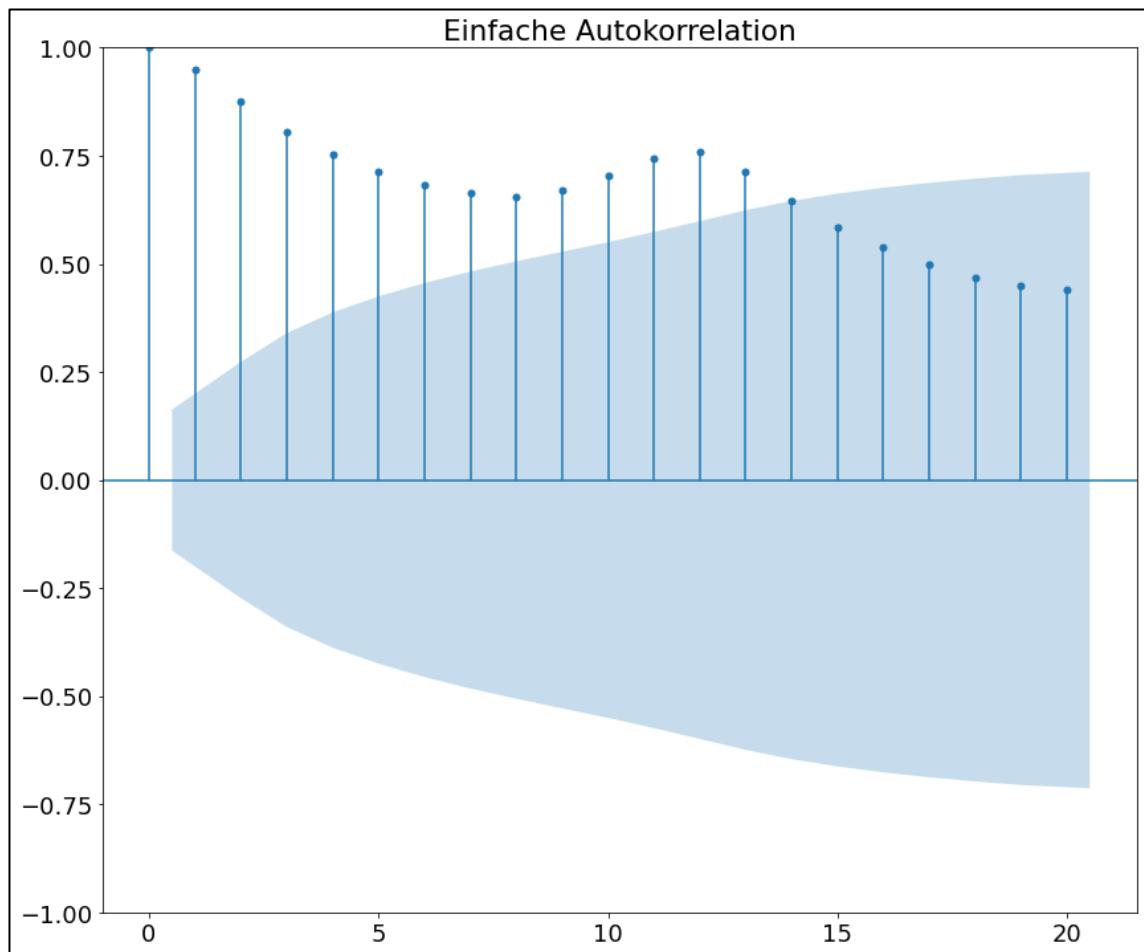
Es lässt sich nun die Korrelation der Zeitreihe  $y(t)$  mit ihren versetzten Zeitreihen  $y_1(t)$ ,  $y_2(t)$ , ... berechnen, um den Einfluss vorheriger Beobachtungen auf  $y(t)$  zu ermitteln. Diese Autokorrelationen werden häufig in einem Autokorrelationsdiagramm dargestellt. Dafür werden auf der x-Achse die Anzahl an Lags und auf der y-Achse der entsprechende Korrelationskoeffizient abgetragen. Häufig wird das Konfidenzintervall beim jeweiligen Korrelationskoeffizienten in blau eingezeichnet.<sup>64</sup>

Die Abbildung 5 zeigt ein einfaches Autokorrelationsdiagramm. Beim nullten Lag handelt es sich um die Korrelation der Zeitreihe mit sich selbst, daher ist dieser Wert stets +1. Das erste Lag korreliert mit einem Koeffizienten von +0,96 mit der Zeitreihe, die Beobachtungen  $y(t)$  korrelieren also stark positiv mit den Beobachtungen  $y(t-1)$ . Da alle Lags mit Korrelationskoeffizienten außerhalb der blau markierten Fläche als statistisch signifikant zu betrachten sind<sup>65</sup>, lässt sich eine positive Korrelation der Zeitreihe mit den Lags eins bis 14 erkennen.

**Abbildung 5: Beispiel für einfaches Autokorrelationsdiagramm**

<sup>64</sup> Vgl. *Hyndman, R. J., Athanasopoulos, G.*, Forecasting, S. 40 ff.

<sup>65</sup> Vgl. *Pal, A., Prakash, P.*, Practical Time Series Analysis, 2017, S. 37 ff.



Quelle: Eigene Darstellung

Derartige Autokorrelationen können für Vorhersagemodelle verwendet werden.<sup>66</sup> Bei der einfachen Autokorrelation gibt es eine wesentliche Einschränkung zu beachten. Wenn  $y(t)$  und  $y(t-1)$  korrelieren, dann korrelieren  $y(t-1)$  und  $y(t-2)$  ebenfalls miteinander. Eine Korrelation von  $y(t)$  und  $y(t-2)$  könnte also auch über die gemeinsame Verbindung durch Korrelationen mit  $y(t-1)$  erklärbar sein. Diese Korrelation liefert allerdings keine zusätzlichen Informationen über die Zeitreihe und wäre daher beispielsweise für Vorhersagemodelle irrelevant.<sup>67</sup> Um diese Einschränkung der einfachen Autokorrelation zu umgehen, kann die partielle Autokorrelation ermittelt werden.<sup>68</sup>

<sup>66</sup> Vgl. Shmueli, G., Lichtendahl, K. C., Time Series Forecasting, 2016, S. 147 ff.

<sup>67</sup> Vgl. Hyndman, R. J., Athanasopoulos, G., Forecasting, S. 239 ff.

<sup>68</sup> Vgl. Mills, T., Applied Time Series Analysis, 2019, S. 39 f.

### 2.3.2 Partielle Autokorrelation

Bei der partiellen Autokorrelation (englisch „partial autocorrelation“ bzw. PACF) wird ein multivariates Regressionsmodell für die Zeitreihe erstellt. Als Eingabevervariablen werden die vorherigen Beobachtungen verwendet. Die Anzahl der Lags gibt dabei an, wie viele der vorherigen Beobachtungen der Zeitreihe für die Regression verwendet werden sollen.<sup>69</sup> Eine Regression für die partielle Autokorrelation mit drei Lags könnte also folgendermaßen aussehen:

**Formel 2: Regression für partielle Autokorrelation**

$$y(t) = x_1 * y(t-1) + x_2 * y(t-2) + x_3 * y(t-3) + e \quad (1)$$

Quelle: In Anlehnung an *Box et al.*, Time Series Analysis, 2016, S. 66.

Für diese Regression werden dann die Regressionskoeffizienten  $x_1$ ,  $x_2$ ,  $x_3$  und der Fehlerterm  $e$  ermittelt. Der Regressionskoeffizient  $x_1$  gibt den Einfluss der unmittelbar vorherigen Beobachtung  $y(t-1)$  auf die Zeitreihe an. Der zweite Regressionskoeffizient  $x_2$  gibt den Einfluss der Beobachtungen  $y(t-2)$  an, der dritte Regressionskoeffizient  $x_3$  gibt den Einfluss der Beobachtungen  $y(t-3)$  an. Alle übrigen Einflüsse auf die Zeitreihe, die nicht durch die drei Regressionskoeffizienten abgebildet werden können, werden im Fehlerterm  $e$  zusammengefasst.<sup>70</sup> Die partielle Autokorrelation ergibt sich nun aus dem jeweils letzten Regressionskoeffizienten, bei drei Lags also  $x_3$ . Auf diese Weise werden Einflüsse durch andere Lags eliminiert und lediglich der Einfluss der jeweiligen versetzten Zeitreihe ermittelt.<sup>71</sup>

Auch die partiellen Autokorrelationen können in einem partiellen Autokorrelationsdiagramm dargestellt werden. Dazu werden wie bei der einfachen Autokorrelation die Koeffizienten auf der y-Achse und die Lags auf der x-Achse abgetragen sowie das Konfidenzintervall in blau eingezeichnet.<sup>72</sup> Die Abbildung 6 zeigt ein partielles Autokorrelationsdiagramm. Auch hier handelt es sich beim nullten Lag um die Korrelation der Zeitreihe mit sich selbst. Beim ersten Lag sind einfache und partielle Autokorrelation gleich, da es hier noch keine Einflüsse anderer Lags gibt, die durch die partielle Autokorrelation entfernt werden könnten.<sup>73</sup> Beim zweiten Lag zeigt sich der

<sup>69</sup> Vgl. *Montgomery, D., Jennings, C., Kulahci, M.*, Time Series, 2015, S. 348 ff.

<sup>70</sup> Vgl. *Box et al.*, Time Series Analysis, 2016, S. 64 ff.

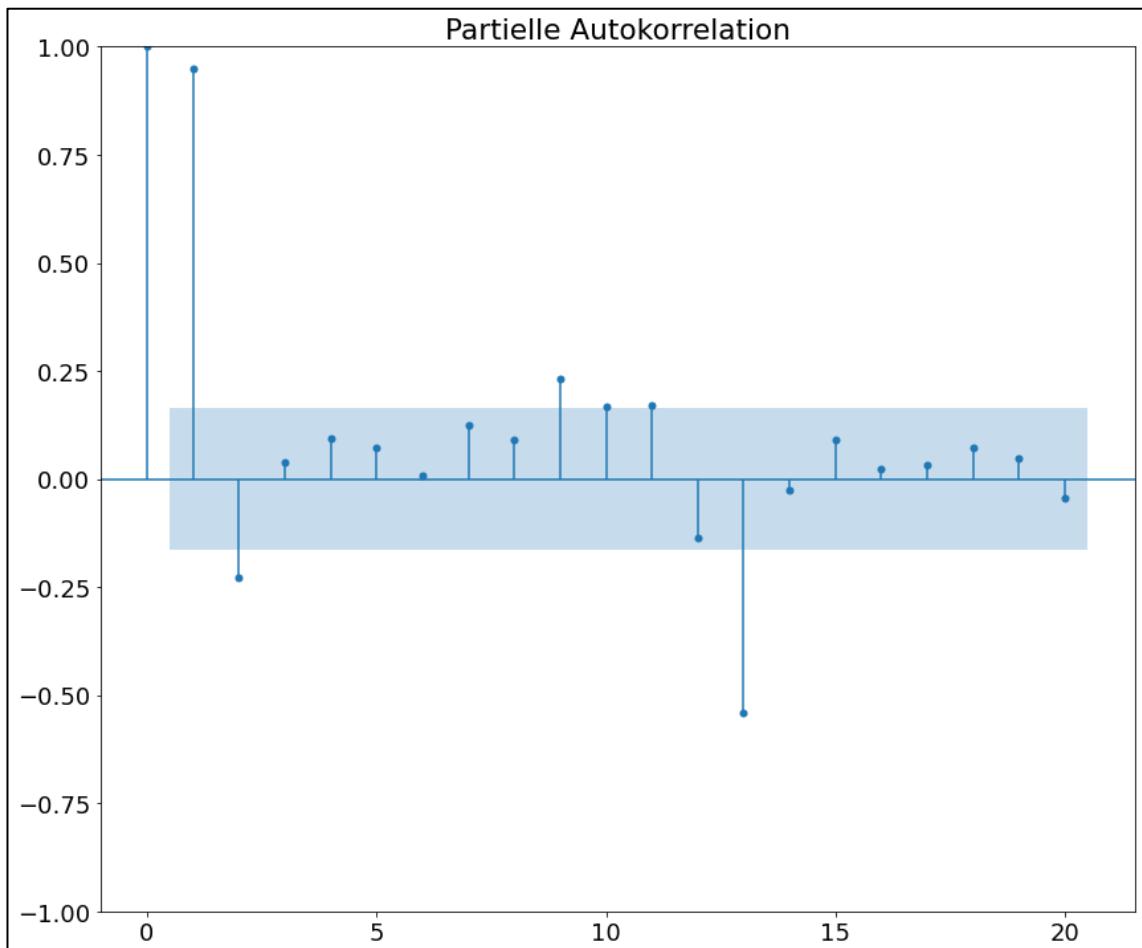
<sup>71</sup> Vgl. *Hyndman, R. J., Athanasopoulos, G.*, Forecasting, S. 239 ff.

<sup>72</sup> Vgl. *Lazzeri, F.*, Machine Learning, 2021, S. 110 ff.

<sup>73</sup> Vgl. *Hyndman, R. J., Athanasopoulos, G.*, Forecasting, S. 240.

Unterschied zur einfachen Autokorrelation, da hier mit etwa -0,25 eine leichte negative Korrelation vorliegt. Die Lags zwei bis acht sind statistisch nicht signifikant, die Lags neun bis elf sind leicht positiv korreliert und so weiter.

**Abbildung 6: Beispiel für partielle Autokorrelationsdiagramm**



Quelle: Eigene Darstellung

### 2.3.3 Stationarität

Eine Zeitreihe ist stationär, wenn statistische Eigenschaften wie beispielsweise die Varianz oder die Autokorrelationen über die gesamte Zeitreihe hinweg ohne größere Abweichungen konstant bleiben.<sup>74</sup> Viele Algorithmen nutzen die in den Daten vorhandenen Autokorrelationen für die Erstellung von Vorhersagen (hierauf wird in den Kapiteln 2.4 und 2.5 eingegangen). Wenn sich die Autokorrelationen mit der Zeit ändern, kann ein Algorithmus die Zeitreihe gegebenenfalls nicht oder nur schwer

<sup>74</sup> Vgl. Hirschle, J., Machine Learning für Zeitreihen, 2020, S. 106 ff.

abbilden. Daher sind Algorithmen für die Zeitreihenvorhersage typischerweise auf stationäre Zeitreihen mit konstanten Autokorrelationen angewiesen.<sup>75</sup> Besonders Zeitreihen mit stark ausgeprägten Trend- oder Saisonalitätskomponenten sind oft (aber nicht zwangsläufig) nicht-stationär.<sup>76</sup>

Der sogenannte Augmented-Dickey-Fuller-Test (ADF-Test) ist eine gängige Methode zum Überprüfen einer Zeitreihe auf Stationarität. Es handelt sich um einen statistischen Test, dessen Nullhypothese besagt, dass die untersuchte Zeitreihe nicht-stationär ist. Die Alternativhypothese besagt dementsprechend, dass es sich um eine stationäre Zeitreihe handelt. Als Konfidenzintervall werden üblicherweise 5% verwendet (entspricht einem p-Wert von 0,05).<sup>77</sup> Ein weiterer Test auf Stationarität ist der sogenannte Kwiatkowski–Phillips–Schmidt–Shin-Test (KPSS-Test). Es handelt sich ebenfalls um einen statistischen Test, dessen Nullhypothese allerdings von stationären Daten ausgeht.<sup>78</sup>

Nicht-stationäre Daten können durch verschiedene Methoden in stationäre Daten umgewandelt werden, was auch als Stationarisierung bezeichnet wird. Saisonalitäten und Trends werden auf unterschiedliche Arten stationarisiert.<sup>79</sup>

Die Abbildung 7 zeigt die monatliche Durchschnittstemperatur der Stadt Luxemburg von 2010 bis Mitte 2020 als Zeitreihe  $y(t)$ .<sup>80</sup> Dabei ist die jährliche Saisonalität deutlich erkennbar, es handelt sich auch nach ADF-Test um nicht-stationäre Daten.

**Abbildung 7: Temperatur in Luxemburg**

---

<sup>75</sup> Vgl. Auffarth, B., Machine Learning for Time Series, 2021, S. 56.

<sup>76</sup> Vgl. Patel, A., Vishwas, B. V., Time Series Analysis Python, 2020, S. 101.

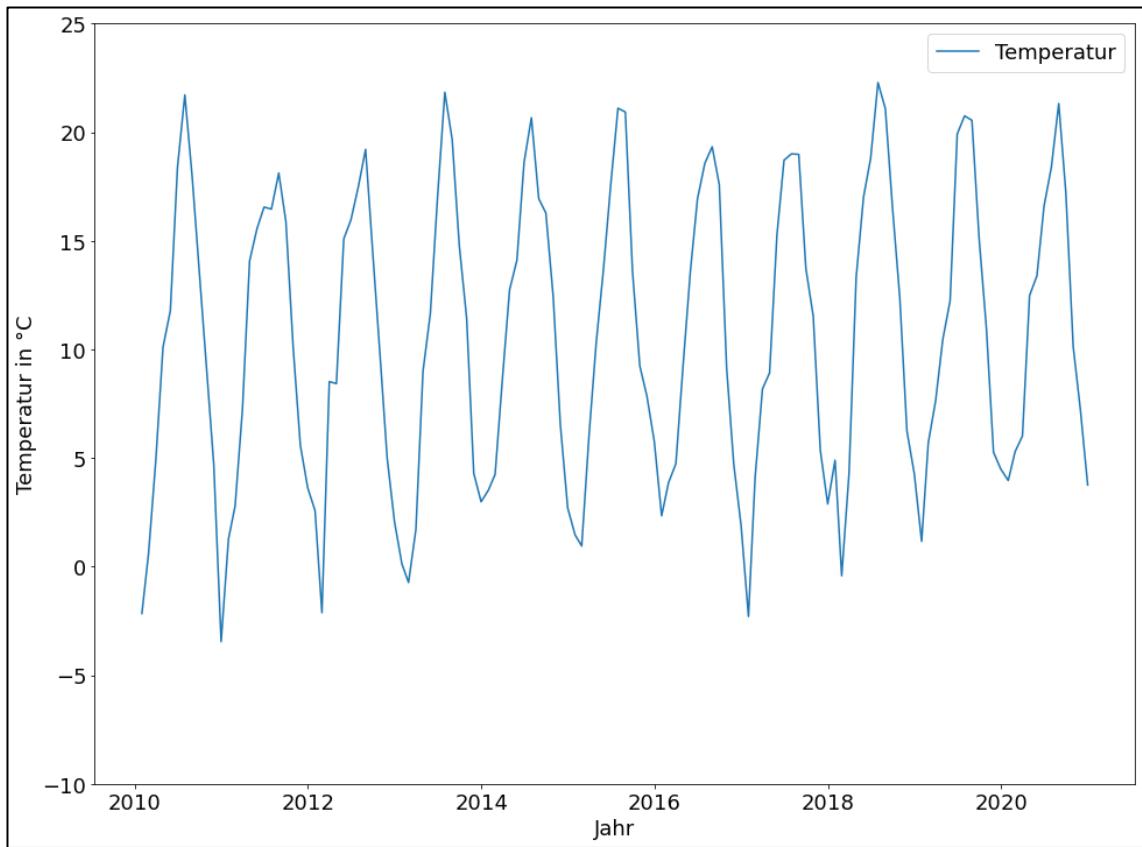
<sup>77</sup> Vgl. Hirschle, J., Machine Learning für Zeitreihen, 2020, S. 106 ff.

<sup>78</sup> Vgl. statsmodels, Stationarity, 2022.

<sup>79</sup> Vgl. Hyndman, R. J., Athanasopoulos, G., Forecasting, S. 223 ff.

<sup>80</sup> Beispiel in Anlehnung an Hirschle, J., Machine Learning für Zeitreihen, 2020, S. 109 ff.

## Stromverbrauch und Zeitreihen



Quelle: Eigene Darstellung

Um saisonale Daten zu stationarisieren, bietet sich das sogenannte saisonale Differencing an. Dabei wird die Zeitreihe  $y(t)$  in  $y'(t)$  umgewandelt, indem die korrespondierende Beobachtung der letzten Saison  $y(t-m)$  von der aktuellen Beobachtung  $y(t)$  abgezogen wird ( $m$  entspricht der Länge einer Saison). Die Länge der Saison beziehungsweise  $m$  wird dabei als Ordnung des Differencing bezeichnet.<sup>81</sup> Es gilt also folgende Formel:

### Formel 3: Saisonales Differencing

$$y'(t) = y(t) - y(t-m) \quad (1)$$

Quelle: In Anlehnung an Patel, A., Vishwas, B. V., Time Series Analysis Python, 2020, S. 108.

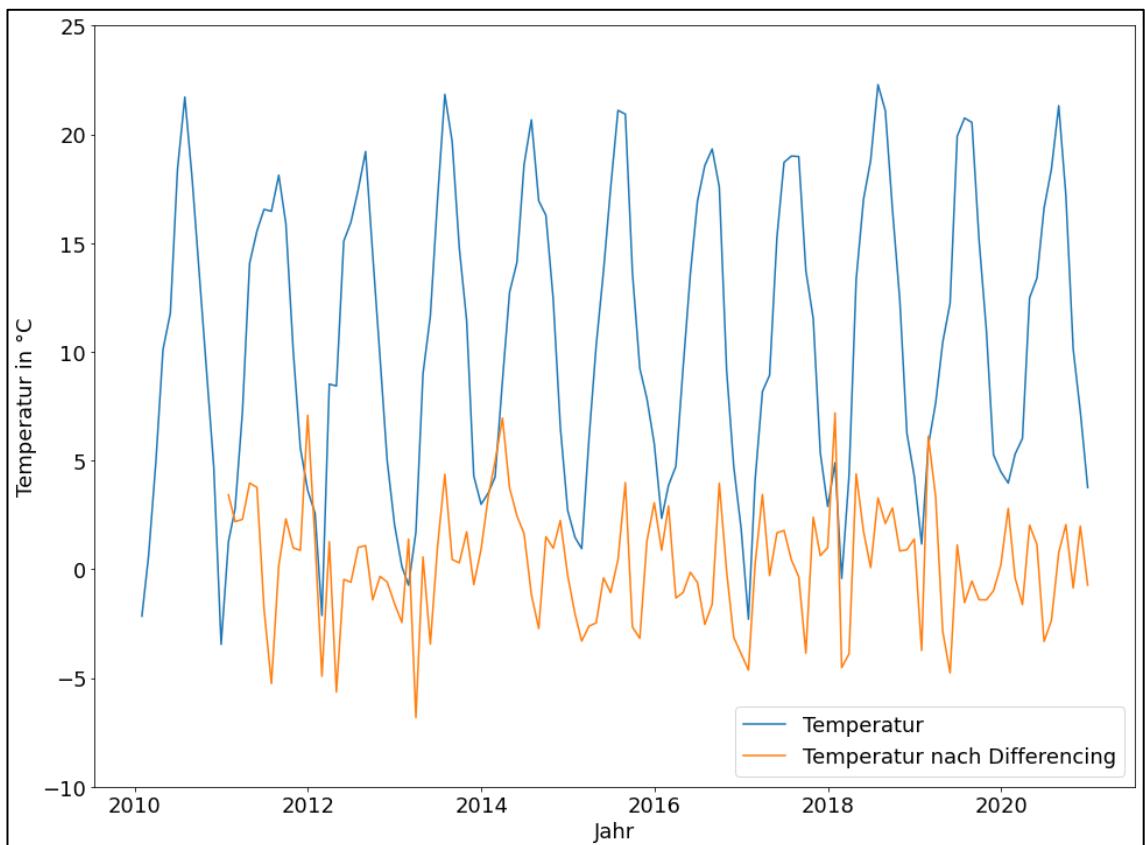
Da es sich im Beispiel um monatliche Daten handelt und sich die Saison alle zwölf Monate wiederholt, können die Daten mit Differencing der zwölften Ordnung stationarisiert werden. Die Abbildung 8 zeigt die Daten vor und nach dem Differencing. Der Verlauf der Zeitreihe nach dem Differencing hat eine wesentlich niedrigere

<sup>81</sup> Vgl. Shmueli, G., Lichtendahl, K. C., Time Series Forecasting, 2016, S. 85 ff.

## Stromverbrauch und Zeitreihen

Amplitude, die Ausschläge nach oben und unten sind geringer und es ist keine Seasonalität mehr erkennbar. Auch der ADF-Test erkennt die Daten als stationär.

**Abbildung 8: Temperatur in Luxemburg (vor/nach Differencing)**



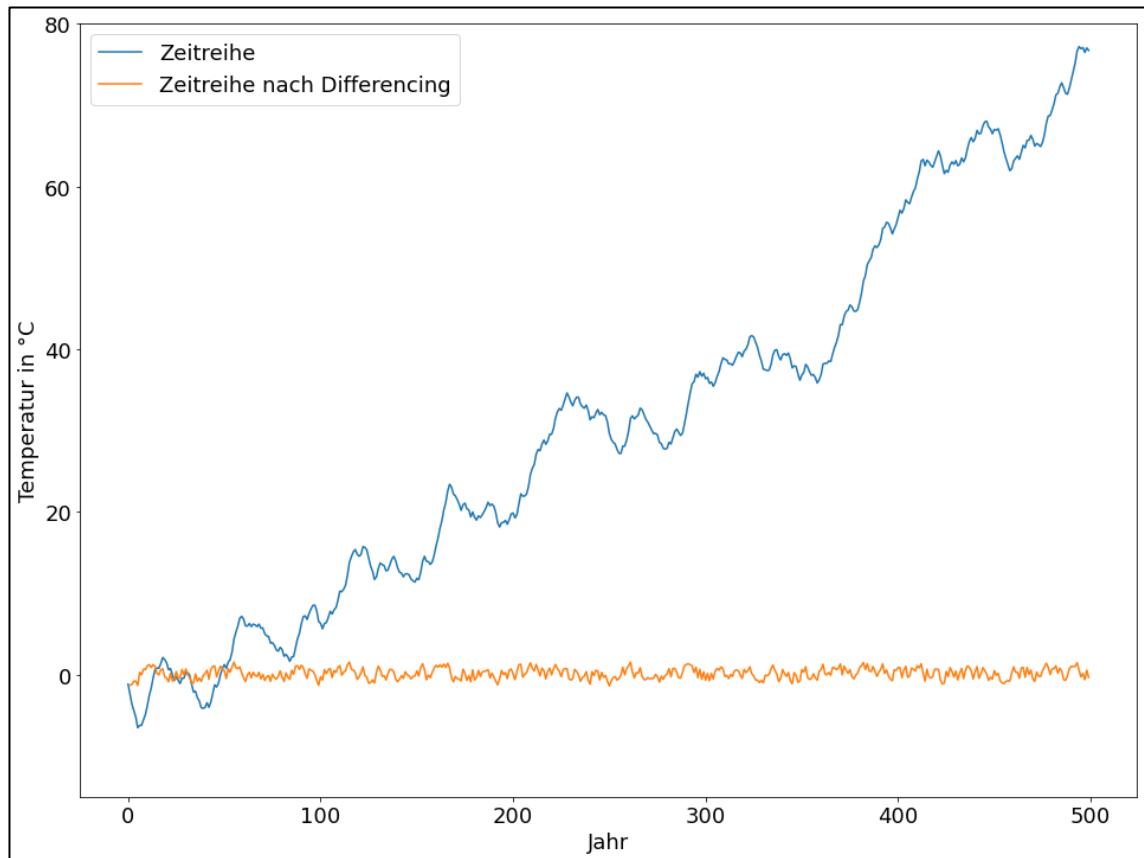
Quelle: Eigene Darstellung

Durch Differencing der ersten Ordnung lassen sich häufig auch Trends entfernen. Dabei wird dann einfach die Differenz zur vorherigen Beobachtung gebildet. Die Abbildung 9 zeigt eine zufällig generierte **Zeitreihe** (blau) mit einem deutlich erkennbaren Trend.<sup>82</sup> Durch einfaches Differencing der ersten Ordnung lässt sich der Trend jedoch entfernen (orange).<sup>83</sup>

**Abbildung 9: Generierte Zeitreihe (vor/nach Differencing)**

<sup>82</sup> Beispiel in Anlehnung an *Hirschle, J.*, Machine Learning für Zeitreihen, 2020, S. 111 ff.

<sup>83</sup> Vgl. *Hirschle, J.*, Machine Learning für Zeitreihen, 2020, S. 111 ff.



Quelle: In Anlehnung an *Hirschle, J.*, Machine Learning für Zeitreihen, 2020, S. 112.

Durch Methodiken wie Differencing werden systematische Komponenten zunächst aus den Daten entfernt. Die daraus entstandenen stationären Daten können von Algorithmen besser verarbeitet werden. Das vom Algorithmus erstellte Modell fokussiert sich dann vorrangig auf die Vorhersage des nicht-systematischen Anteils der Zeitreihe. Die systematischen Komponenten können trotzdem wieder zur Prognose hinzugezogen werden.<sup>84</sup>

## 2.4 Autoregressive Integrated Moving Average

Der ARIMA-Algorithmus ist ein häufig eingesetzter Algorithmus für die Zeitreihenvorhersage. In seiner ersten Form wurde ARIMA im Jahr 1951 vom Neuseeländer Peter Whittle vorgestellt und im Jahr 1971 von den Briten Peter Box und Gwilym Jenkins zu seiner heutigen Form weiterentwickelt.<sup>85</sup>

<sup>84</sup> Vgl. *Hirschle, J.*, Machine Learning für Zeitreihen, 2020, S. 114 f.

<sup>85</sup> Vgl. *Auffarth, B.*, Machine Learning for Time Series, 2021, S. 132 ff.

Die Abkürzung ARIMA steht für die grundlegenden Module des Algorithmus:<sup>86</sup>

- **Autoregressive (AR):** Ein Autoregressionsmodell auf einen Teil der Zeitreihe
- **Integrated (I):** Ein Modul zur Stationarisierung mittels Differencing
- **Moving Average (MA):** Ein Modell, das Fehler aus vorherigen Vorhersagen erkennt und künftige Vorhersagen dahingehend zu korrigiert versucht.

Es handelt sich bei den drei Modulen um unabhängige Algorithmen beziehungsweise Modelle, welche in einem Modell zusammengefügt werden und gegebenenfalls durch weitere Module ergänzt werden können.<sup>87</sup>

Im Folgenden werden die einzelnen Module und Erweiterungen vorgestellt.

#### 2.4.1 Autoregression (AR)

Bei der Autoregression wird ein Regressionsmodell auf einen Teil der Zeitreihe selbst gebildet. Das Regressionsmodell wird dann für die Ermittlung eines zukünftigen Wertes eingesetzt. Als Eingabedaten dienen jeweils die letzten Beobachtungen beziehungsweise Lags der Zeitreihe.<sup>88</sup> Die Ordnung der Autoregression gibt dabei an, wie viele Lags verwendet werden. Beispielsweise verwendet eine Autoregression der zweiten Ordnung die letzten zwei Lags für die Vorhersage eines in der Zukunft liegenden Wertes. Die Ordnung wird üblicherweise als Parameter  $p$  des ARIMA-Modells bezeichnet.<sup>89</sup> Die Tabelle 2 zeigt eine Zeitreihe  $y(t)$  mit den ersten zwei Lags:

**Tabelle 2: Zeitreihe für Autoregression**

t	y(t)	y(t-1)	y(t-2)
1	5	-	-
2	7	5	-
3	6	7	5
4	3,5	6	7
5	3	3,5	6
6	4	3	3,5

<sup>86</sup> Vgl. Lazzeri, F., Machine Learning, 2021, S. 101 ff. und S. 122 ff.

<sup>87</sup> Vgl. Patel, A., Vishwas, B. V., Time Series Analysis Python, 2020, S. 117 ff.

<sup>88</sup> Vgl. Shmueli, G., Lichendahl, K. C., Time Series Forecasting, 2016, S. 143 ff.

<sup>89</sup> Vgl. Lazzeri, F., Machine Learning, 2021, S. 102 ff.

7	6	4	3
8	6,5	6	4

Quelle: In Anlehnung an *Hirschle, J.*, Machine Learning für Zeitreihen, 2020, S. 103.

Für die Autoregression wird dann eine Regressionsfunktion aufgestellt. Für jedes verwendete Lag wird jeweils ein Regressionskoeffizient eingefügt ( $x_1$  und  $x_2$ ), zusätzlich wird eine **Konstante c** addiert (siehe Formel 4 Zeile 1). Die Regressionskoeffizienten und die Konstante werden dann so bestimmt, dass die Autoregressionsfunktion die Zeitreihe möglichst gut abbildet (siehe Formel 4 Zeile 2). Angenommen, die Zeitreihe ist bis zur Beobachtung  $y(7)$  bekannt und es soll die zukünftige Beobachtung  $y(8)$  vorhergesagt werden (die Vorhersage wird dann als  $\hat{y}(t)$  beziehungsweise  $\hat{y}(8)$  bezeichnet). So können die Lags in die Autoregressionsfunktion eingesetzt werden (**6 als erstes Lag** und **4 als zweites Lag** mit **5,3 als Konstante**, siehe Formel 4 Zeile 3). Es kann dann der Wert für  $\hat{y}(8)$  ermittelt werden.<sup>90</sup>

#### Formel 4: Autoregressionsfunktion

$$\hat{y}(t) = x_1 * y(t-1) + x_2 * y(t-2) + c \quad (1)$$

$$\hat{y}(t) = 0,73 * y(t-1) + -0,86 * y(t-2) + 5,3 \quad (2)$$

$$\hat{y}(8) = 0,73 * 6 + -0,86 * 4 + 5,3 \quad (3)$$

$$\hat{y}(8) = 6,24 \quad (4)$$

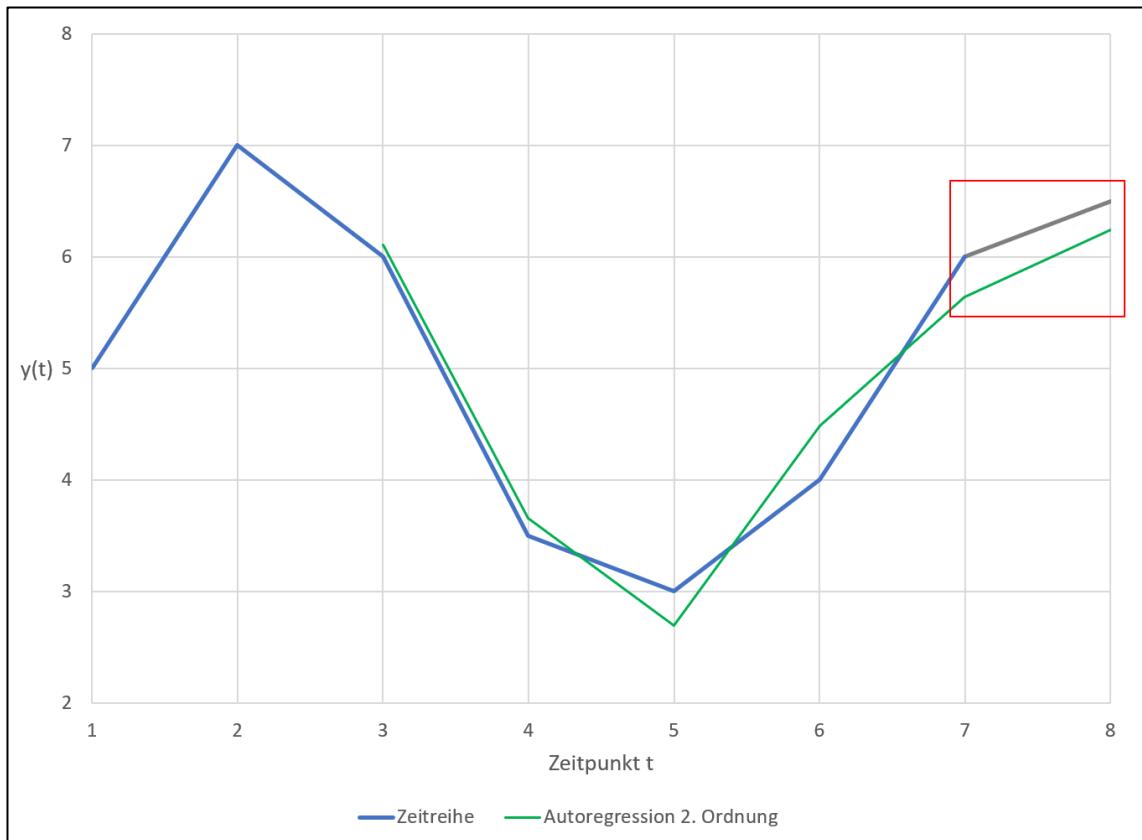
Quelle: In Anlehnung an *Shmueli, G.*, *Lichtendahl, K. C.*, Time Series Forecasting, 2016, S. 147.

Die Abbildung 10 zeigt die **Zeitreihe  $y(t)$**  in blau sowie die beispielhafte **Autoregression zweiter Ordnung** in grün. Die grüne **Autoregressionsfunktion** kann den Verlauf der Zeitreihe in etwa abbilden. Für die Vorhersage am Zeitpunkt  $t=8$  kann die Funktion dann wie oben mittels der letzten beiden Lags befüllt und eine Vorhersage errechnet werden.

Abbildung 10: Beispiel für Autoregression

<sup>90</sup> Vgl. *Hirschle, J.*, Machine Learning für Zeitreihen, 2020, S. 102 ff.

## Stromverbrauch und Zeitreihen



Quelle: Eigene Darstellung

Die Wahl der richtigen Ordnung ist entscheidend dafür, wie gut die Autoregression die Zeitreihe abbilden und damit Vorhersagen erstellen kann. Wird die Ordnung zu niedrig gewählt, werden autokorrelierte Lags mit wertvollen Informationen möglicherweise nicht in das Modell einbezogen, wodurch die Qualität der Vorhersagen beeinträchtigt werden kann.<sup>91</sup> Wird die Ordnung zu hoch gewählt, werden möglicherweise zu viele nicht-autokorrelierte Lags in das Modell integriert, die dann unter anderem zu Verzerrungen führen und die Qualität der Vorhersagen ebenfalls stark vermindern können.<sup>92</sup>

Die einfachen und partiellen Autokorrelationsdiagramme können bei der Erstellung von Autoregressionsmodellen verwendet werden. Da die Autoregression auf den Autokorrelationen der letzten verwendeten Lags basiert, macht der Einsatz der Autoregression nur dann Sinn, wenn derartige Autokorrelationen in den Daten vorhanden sind.<sup>93</sup> Für diese Analyse lässt sich das partielle Autokorrelationsdiagramm nutzen,

<sup>91</sup> Vgl. Hyndman, Rob J., Athanasopoulos, George (2018), S. 239 f.

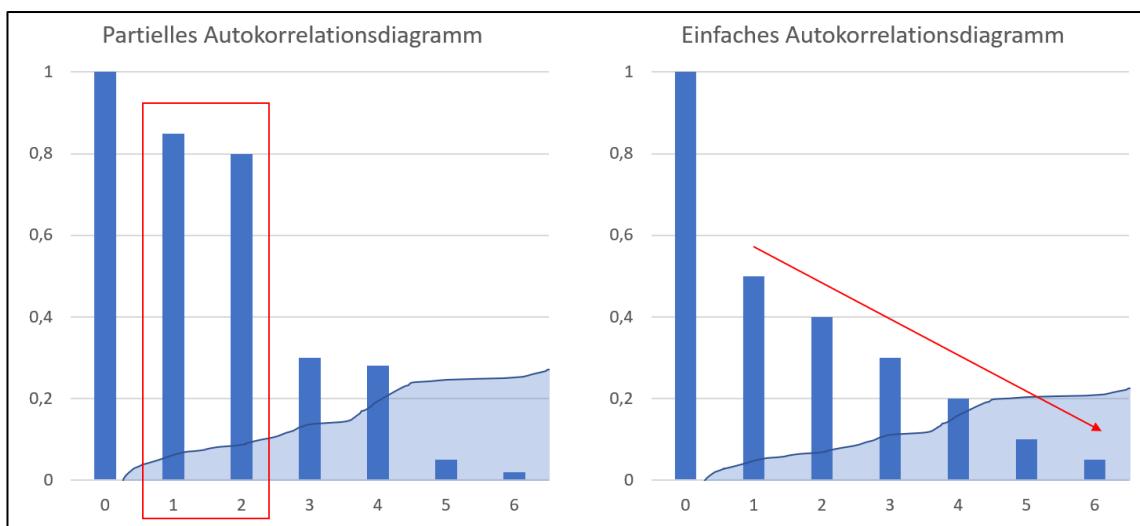
<sup>92</sup> Vgl. Jansen, S., Machine Learning for Trading, 2020, S. 266 f.

<sup>93</sup> Vgl. Shmueli, G., Lichtendahl, K. C., Time Series Forecasting, 2016, S. 147 ff.

in dem die voneinander isolierten Autokorrelationen jedes Lags einer Zeitreihe dargestellt sind.<sup>94</sup> Wie bereits oben beschrieben sollten möglichst nur stark autokorrierende Lags in das Autoregressionsmodell einbezogen werden.<sup>95</sup>

Die Abbildung 11 zeigt auf der linken Seite ein beispielhaftes partielle Autokorrelationsdiagramm. In diesem Fall böte es sich an, die ersten beiden Lags zu verwenden, da hier signifikante und starke Autokorrelationen vorliegen. Die weiteren Lags korrelieren nur schwach und teilweise nicht-signifikant und sollten daher zunächst nicht für das Modell verwendet werden.<sup>96</sup> Derartige partielle Autokorrelationen führen oft zu konstant abnehmenden einfachen Autokorrelationen (wie im rechten Diagramm dargestellt), da sich die Korrelation des ersten Lags wie bereits oben beschrieben auf alle übrigen Lags auswirkt.<sup>97</sup>

**Abbildung 11: Partielles und einfaches Autokorrelationsdiagramm (AR)**



Quelle: Eigene Darstellung

#### 2.4.2 Moving Average (MA)

Wie bereits erwähnt, verwendet ein Moving-Average-Modell die Fehler vorheriger, vorgesetzter Vorhersagen.<sup>98</sup> Als Modell zur Erstellung dieser vorgesetzten Vorhersagen kann eine Autoregression verwendet werden, es können aber auch einfachere Modelle wie beispielsweise der Zero-Rule-Algorithmus eingesetzt

<sup>94</sup> Vgl. Miles, T., Applied Time Series Analysis, 2019, S. 40.

<sup>95</sup> Vgl. Box et al., Time Series Analysis, 2016, S. 183.

<sup>96</sup> Beispiel in Anlehnung an Hirschle, J., Machine Learning für Zeitreihen, 2020, S. 115 ff.

<sup>97</sup> Vgl. Hirschle, J., Machine Learning für Zeitreihen, 2020, S. 116 f.

<sup>98</sup> Vgl. Lazzeri, F., Machine Learning, 2021, S. 119.

werden. Dabei wird für eine Zeitreihe  $y(t)$  jeweils der letzte bekannte Wert  $y(t-1)$  als Vorhersage für  $y(t)$  verwendet.<sup>99</sup> Die Abweichungen der Vorhersagen von tatsächlichen Werten (Vorhersagefehler  $e(t)$ ) werden als Eingabe für das Moving-Average-Modell verwendet.<sup>100</sup> Die Ordnung des Moving-Average-Modells gibt dabei ähnlich wie bei der Autoregression die Anzahl der verwendeten Vorhersagefehler an. So verwendet ein Moving-Average-Modell der zweiten Ordnung die letzten beiden Vorhersagefehler. Die Ordnung wird allgemein als Parameter  $q$  des ARIMA-Modells bezeichnet.<sup>101</sup> Die Tabelle 3 zeigt eine beispielhafte Zeitreihe  $y(t)$  mit der Zero-Rule-Vorhersage  $y(t-1)$  und den entsprechenden Vorhersagefehlern  $e(t)$ .

**Tabelle 3: Zeitreihe für Moving-Average**

t	$y(t)$	$y(t-1)$	$e(t) = y(t-1) - y(t)$
1	2		
2	5	2	-3
3	7	5	-2
4	4	7	3
5	3	4	1
6	5	3	-2
7	6	5	-1
8	8	6	-2

Quelle: In Anlehnung an *Hirschle, J.*, Machine Learning für Zeitreihen, 2020, S. 103 ff.

Für das Moving-Average-Modell wird dann eine Regressionsfunktion aufgestellt. Für jeden verwendeten Vorhersagefehler wird jeweils ein Regressionskoeffizient eingefügt ( $x_1$  und  $x_2$ ). Die Vorhersagefehler ( $e(t-1)$  und  $e(t-2)$ ) werden mit den Regressionskoeffizienten verrechnet und von der Zero-Rule-Vorhersage  $y(t-1)$  abgezogen, wodurch sie um die vorherigen Vorhersagefehler korrigiert wird (siehe Formel 5 Zeile 1).<sup>102</sup> Dafür müssen die optimalen Regressionskoeffizienten bei der Modellerstellung

<sup>99</sup> Vgl. *Brownlee, J.*, Baseline Predictions, 2019.

<sup>100</sup> Vgl. *Auffarth, B.*, Machine Learning for Time Series, 2021, S. 134 ff.

<sup>101</sup> Vgl. *Box et al.*, Time Series Analysis, 2016, S. 68 ff.

<sup>102</sup> Vgl. *Hirschle, J.*, Machine Learning für Zeitreihen, 2020, S. 104.

berechnet werden. Durch die unterschiedlichen Koeffizienten handelt es sich daher um einen gewichteten Durchschnitt der Vorhersagefehler.<sup>103</sup> Im Beispiel wird für beide Koeffizienten jeweils 0,5 verwendet (siehe Formel 5 Zeile 2). Angenommen, die Zeitreihe ist bis zur Beobachtung  $y(7)$  bekannt und es soll die in der Zukunft liegende Beobachtung  $y(8)$  vorhergesagt werden. Dafür werden die Vorhersagefehler (-1 für  $e(t-1)$  und -2 für  $e(t-2)$ ) sowie die Zero-Rule-Vorhersage (6 für  $y(t-1)$ ) eingesetzt (siehe Formel 5 Zeile 3). Der gewichtete Durchschnitt der Vorhersagefehler wird dann berechnet und von der Zero-Rule-Vorhersage abgezogen (siehe Formel 5 Zeile 4 bis 6), die Vorhersage  $\hat{y}(8)$  wird von 6 um 1,5 nach oben auf 7,5 korrigiert.<sup>104</sup>

#### Formel 5: Moving-Average-Funktion

$$\hat{y}(t) = y(t-1) - (x_1 * e(t-1)) + x_2 * e(t-2) \quad (1)$$

$$\hat{y}(t) = y(t-1) - (0,5 * e(t-1)) + 0,5 * e(t-2) \quad (2)$$

$$\hat{y}(8) = 6 - (0,5 * -1) + 0,5 * -2 \quad (3)$$

$$\hat{y}(8) = 6 - (-0,5) + -1 \quad (4)$$

$$\hat{y}(8) = 6 - (-1,5) \quad (5)$$

$$\hat{y}(8) = 7,5 \quad (6)$$

Quelle: In Anlehnung an Patel, A., Vishwas, B. V., Time Series Analysis Python, 2020, S. 119.

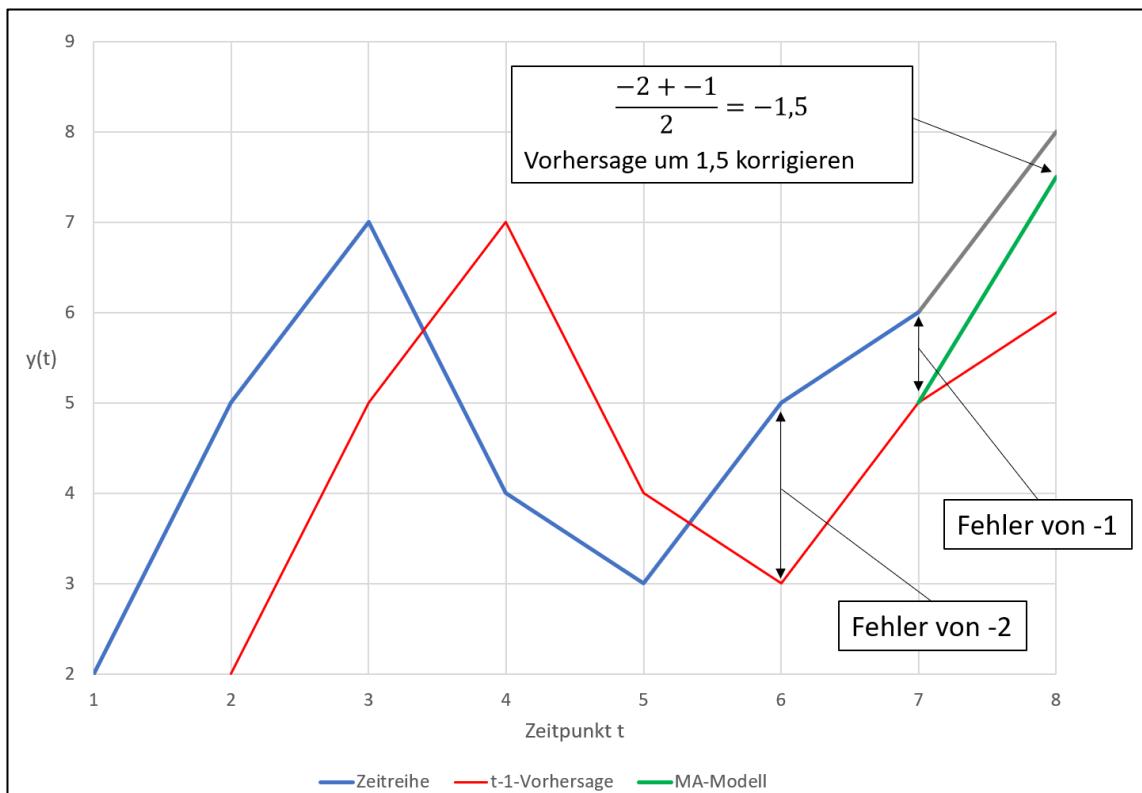
Die Abbildung 12 zeigt die Zeitreihe  $y(t)$  in blau und die Vorhersage des Zero-Rule-Modells in rot. Die Vorhersage  $\hat{y}(8)$  wird um den gewichteten Durchschnitt der vergangenen Vorhersagefehler des Zero-Rule-Modells für  $y(7)$  und  $y(6)$  korrigiert (grün), wodurch die Vorhersage wesentlich verbessert wird.

Abbildung 12: Beispiel für Moving Average

<sup>103</sup> Vgl. Hirschle, J., Machine Learning für Zeitreihen, 2020, S. 104 ff.

<sup>104</sup> Vgl. Patel, A., Vishwas, B. V., Time Series Analysis Python, 2020, S. 113 ff.

## Stromverbrauch und Zeitreihen



Quelle: Eigene Darstellung

Die Wahl der Ordnung ist auch beim Moving-Average-Modell wichtig. Wenn zu wenig vorgeschaltete Vorhersagefehler beziehungsweise Lags verwendet werden, gehen dem Modell potenziell wichtige Informationen verloren. Andererseits kann der Einbezug von zu vielen Lags ohne zusätzlichen Informationsgewinn das Modell verzerrten.<sup>105</sup> Hier können ebenfalls die Autokorrelationsdiagramme verwendet werden. Wenn die einfache Autokorrelation nach einem bestimmten Lag stark abfällt, dann wirkt sich diese Autokorrelation nicht oder nur sehr schwach auf alle weiteren Lags aus. Dies wird auch als „Schock“ bezeichnet, da das jeweilige Lag keinen längerfristigen Einfluss hat. Diese Schocks lassen sich gut für das Moving-Average-Modell verwenden und können daher als Ordnung gewählt werden.<sup>106</sup> Ein derartiges Verhalten kann oft auch beim partiellen Autokorrelationsdiagramm beobachtet werden.<sup>107</sup> Dies ist allerdings nicht zwangsläufig nötig. Der Einsatz eines Moving-

<sup>105</sup> Vgl. Hyndman, R. J., Athanasopoulos, G., Forecasting, S. 234 ff.

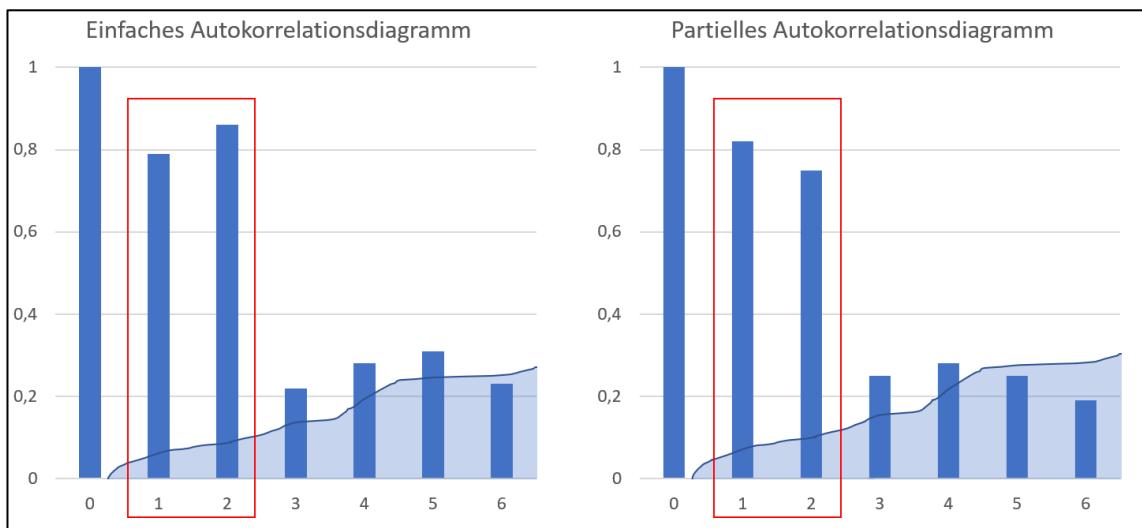
<sup>106</sup> Vgl. Hirschle, J., Machine Learning für Zeitreihen, 2020, S. 115 ff.

<sup>107</sup> Vgl. Jansen, S., Machine Learning for Trading, 2020, S. 266.

Average-Modells bietet sich auch an, wenn die partiellen Autokorrelationen nur gering ausgeprägt ist.<sup>108</sup>

Die Abbildung 13 zeigt links ein idealtypisches einfaches Autokorrelationsdiagramm für ein Moving-Average-Modell der zweiten Ordnung, da die ersten beiden Lags signifikant autokorrelieren und danach ein starker Abfall zum dritten Lag besteht. Ab dem dritten Lag autokorrelieren die Lags dann nur noch sehr schwach und teils nicht-signifikant. Auf der rechten Seite zeigt sich im partiellen Autokorrelationsdiagramm ein ähnliches Verhalten.<sup>109</sup>

**Abbildung 13: Einfaches und partielles Autokorrelationsdiagramm (MA)**



Quelle: In Anlehnung an *Hirschle, J.*, Machine Learning für Zeitreihen, 2020, S. 115 ff.

#### 2.4.3 Autoregressive Integrated Moving Average (ARIMA)

Ein Autoregressionsmodell lässt sich mit einem Moving-Average-Modell kombinieren, indem das Autoregressionsmodell zusätzlich durch ein Moving-Average-Modell um seine vergangenen Vorhersagefehler korrigiert wird.<sup>110</sup> Dies wird als ARMA-Modell bezeichnet.<sup>111</sup> Da beide Modelle auf stationäre Daten angewiesen sind, wird bei ARIMA zusätzlich ein Modul für das Differencing hinzugefügt, um Daten vor ihrer Verarbeitung durch andere Module gegebenenfalls zu stationarisieren.<sup>112</sup> Diese drei Module werden im ARIMA-Algorithmus zusammengefügt, welcher als ARIMA(p, d,

<sup>108</sup> Vgl. *Chatfield, C., Xing, H.*, Time Series Analysis, 2019, S. 90 ff.

<sup>109</sup> Beispiel in Anlehnung an *Hirschle, J.*, Machine Learning für Zeitreihen, 2020, S. 115 ff.

<sup>110</sup> Vgl. *Hyndman, R. J., Athanasopoulos, G.*, Forecasting, S. 236 f.

<sup>111</sup> Vgl. *Miles, T.*, Applied Time Series Analysis, 2019, S. 43 f.

<sup>112</sup> Vgl. *Patel, A., Vishwas, B. V.*, Time Series Analysis Python, 2020, S. 121 f.

q) notiert wird, wobei p für die Ordnung der Autoregression, q für die Ordnung des Moving-Average-Modells und d für die Ordnung des Differencing steht.<sup>113</sup>

Wenn die Module kombiniert werden, werden die Regressionsgleichungen zusammengefügt.<sup>114</sup> Wird beispielsweise das **Autoregressionsmodell aus Kapitel 2.4.1** mit dem **Moving-Average-Modell aus Kapitel 2.4.2** zu einem ARIMA(2, 0, 2)-Modell kombiniert, dann würde sich folgende Regressionsgleichung ergeben (es ist dabei zu beachten, dass die vorgeschalteten Vorhersagen des Zero-Rule-Algorithmus  $y(t-1)$  für die Berechnung des Vorhersagefehler  $e(t)$  durch die Vorhersagen des Autoregressionsmodells ersetzt werden):

#### Formel 6: ARIMA-Funktion

$$\hat{y}(t) = (x_1 * y(t-1) + x_2 * y(t-2) + c) - (x_3 * e(t-1) + x_4 * e(t-2)) \quad (1)$$

Quelle: In Anlehnung an *Hyndman, R. J., Athanasopoulos, G., Forecasting*, S. 236.

Die Module funktionieren auch unabhängig voneinander und können je nach Anwendungsfall ein oder ausgeschaltet werden, der entsprechende Parameter wird dann auf 0 gesetzt. Bei einem ARIMA(2, 0, 0)-Modell handelt es sich beispielsweise um ein Autoregressionsmodell zweiter Ordnung ohne Differencing, bei ARIMA(0, 1, 3) handelt es sich um ein Moving-Average-Modell dritter Ordnung mit Differencing erster Ordnung usw.<sup>115</sup>

#### 2.4.4 Seasonal ARIMA (SARIMA)

Der bis hierhin beschriebene ARIMA-Algorithmus eignet sich vor allem für nicht-saisonale Daten.<sup>116</sup> Wenn die Zeitreihe jedoch eine starke saisonale Komponente aufweist, kann ARIMA durch zusätzliche Module erweitert werden. Bei der Zeitreihe  $y(t)$  in Abbildung 14 lässt sich eine deutliche Saisonalität der Länge vier erkennen. Wenn die Beobachtung  $y(16)$  vorhergesagt werden soll, kann beispielsweise ein ARIMA(2, 0, 2)-Modell dies mittels der grün markierten Lags  $y(15)$  und  $y(14)$  tun.

Zusätzlich kann auch eine weitere Zeitreihe  $Y(t)$  aus den korrespondierenden Lags der vorherigen saisonalen Perioden  $y(t-4)$ ,  $y(t-8)$  und  $y(t-12)$  gebildet werden, die in

---

<sup>113</sup> Vgl. *Hirschle, J., Machine Learning für Zeitreihen*, 2020, S. 118 ff.

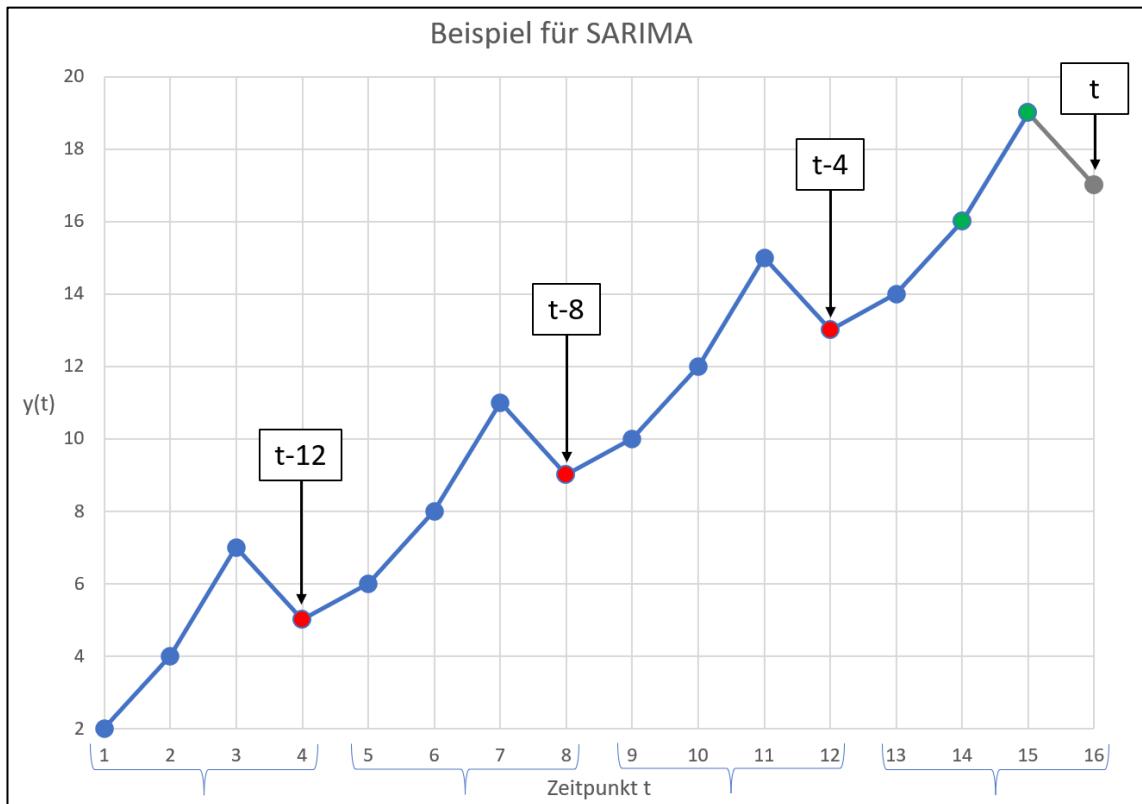
<sup>114</sup> Vgl. *Hyndman, R. J., Athanasopoulos, G., Forecasting*, S. 236.

<sup>115</sup> Vgl. *Patel, A., Vishwas, B. V., Time Series Analysis Python*, 2020, S. 120 f.

<sup>116</sup> Vgl. *Hyndman, R. J., Athanasopoulos, G., Forecasting*, S. 236.

der Abbildung rot markiert sind.<sup>117</sup> Die daraus entstandene Zeitreihe kann von einem zusätzlichen ARIMA-Modell wie bereits beschrieben verarbeitet werden, wobei es wieder jeweils ein Modul für die Autoregression, das Differencing und das Moving-Average-Modell gibt. Diese Module werden als saisonale Module bezeichnet.<sup>118</sup>

Abbildung 14: Beispiel für SARIMA



Quelle: Eigene Darstellung

Die Ordnung der saisonalen Autoregression wird mit dem Parameter P, die Ordnung des saisonalen Differencing mit D und die Ordnung des saisonalen Moving-Average-Modells mit Q angegeben.<sup>119</sup> Daraus ergibt sich folgende Notation mit den Parametern p, d und q für die nicht-saisonalen Module, den Parametern P, D und Q für die saisonalen Module und dem Parameter m für die Länge der saisonalen Perioden:

ARIMA(p, d, q)(P, D, Q)m<sup>120</sup> oder ARIMA(p, d, q)(P, D, Q, m)<sup>121</sup>

<sup>117</sup> Vgl. Hyndman, R. J., Athanasopoulos, G., Forecasting, S. 254 ff.

<sup>118</sup> Vgl. Hirschle, J., Machine Learning für Zeitreihen, 2020, S. 127 ff.

<sup>119</sup> Vgl. Patel, A., Vishwas, B. V., Time Series Analysis Python, 2020, S. 129 f.

<sup>120</sup> Vgl. Auffarth, B., Machine Learning for Time Series, 2021, S. 138.

<sup>121</sup> Vgl. *statsmodels*, SARIMAX, 2022.

Der um saisonale Module erweiterte ARIMA-Algorithmus wird auch als Seasonal ARIMA oder SARIMA bezeichnet. Wie auch beim nicht-saisonalen ARIMA kann jedes Modul des SARIMA-Algorithmus verwendet oder deaktiviert werden. Es können beispielsweise nur saisonale oder auch nur nicht-saisonale Module verwendet werden, die Module lassen sich aber auch beliebig miteinander kombinieren.<sup>122</sup>

#### 2.4.5 Parameterbestimmung gegen Über- und Unteranpassungen

Der Prozess, bei dem ein Algorithmus ein Modell für die jeweiligen Daten erstellt, wird beim maschinellen Lernen als Training bezeichnet. Während dieses Prozesses sucht der Algorithmus nach Strukturen, Mustern und Informationen, die in einem Modell verarbeitet werden können, was auch als Generalisierung bezeichnet wird. Die dafür verwendeten Daten werden als Trainingsdaten bezeichnet.<sup>123</sup> Beim ARIMA-Algorithmus werden beim Training beispielsweise die Regressionskoeffizienten auf Basis der Autokorrelationen bestimmt.<sup>124</sup>

Beim Training besteht allerdings die Gefahr, dass der Algorithmus zu wenig Informationen aus den Trainingsdaten generalisiert. Es kann vorkommen, dass der Algorithmus das Modell zu stark an die Beschaffenheit der Trainingsdaten anpasst.<sup>125</sup> Die Koeffizienten werden dann so gewählt, dass das Modell die Trainingsdaten sehr gut abbildet und ein vermeintlich sehr genaues Modell entsteht. Wenn allerdings neue, beim Training noch unbekannte Daten vorhergesagt werden sollen, nimmt die Genauigkeit der Vorhersagen stark ab, da das Modell zu wenig Informationen generalisiert und stattdessen die Trainingsdaten einfach „auswendig gelernt“ hat.<sup>126</sup> Wenn der Algorithmus das Modell zu stark an die Trainingsdaten anpasst, wird dies als Überanpassung bezeichnet.<sup>127</sup>

Die Anzahl der Koeffizienten im ARIMA-Modell wird durch die Ordnung der saisonalen und nicht-saisonalen Autoregressions- und Moving-Average-Module bestimmt. Modelle höherer Ordnung haben mehr Koeffizienten.<sup>128</sup> Dadurch können komplexere Modelle zunächst mehr Autokorrelationen abbilden und dadurch gegebenenfalls genauere Vorhersagen erzeugen. Gleichzeitig bieten komplexere Modelle einem

---

<sup>122</sup> Vgl. *Hirschle, J.*, Machine Learning für Zeitreihen, 2020, S. 127 ff.

<sup>123</sup> Vgl. *Raschka, S., Mirjalili, V.*, Machine Learning mit Python, 2021, S. 40 ff.

<sup>124</sup> Vgl. *Hirschle, J.*, Machine Learning für Zeitreihen, 2020, S. 101 ff.

<sup>125</sup> Vgl. *IBM*, Overfitting, 2021.

<sup>126</sup> Vgl. *Hirschle, J.*, Machine Learning für Zeitreihen, 2020, S. 125 f.

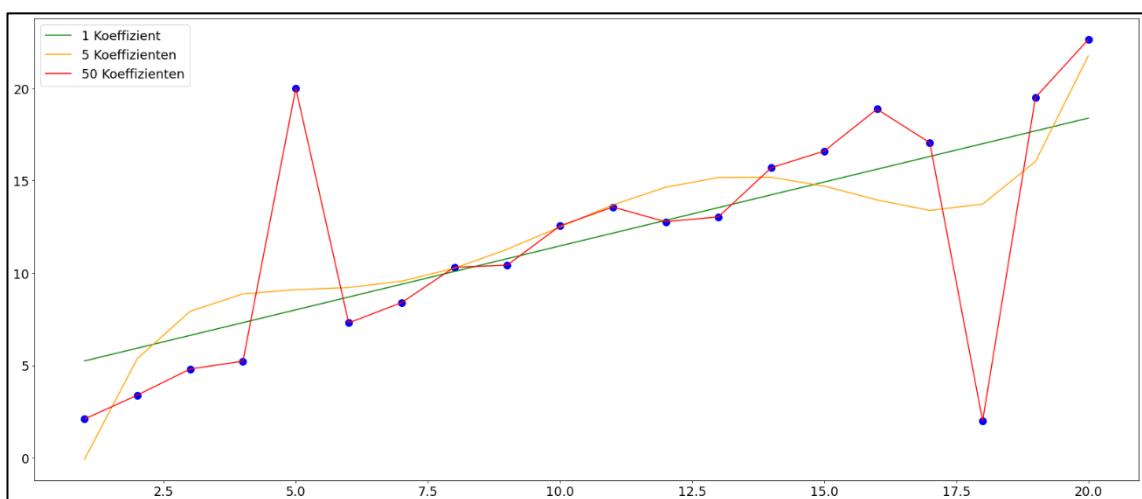
<sup>127</sup> Vgl. *Raschka, S., Mirjalili, V.*, Machine Learning mit Python, 2021, S. 101 ff.

<sup>128</sup> Vgl. *Patel, A., Vishwas, B. V.*, Time Series Analysis Python, 2020, S. 117 ff.

Algorithmus damit aber auch eher die Möglichkeit, sich stärker an die Trainingsdaten anzupassen als bei simpleren Modellen einer niedrigeren Ordnung.<sup>129</sup>

Die Abbildung 15 zeigt eine Grafik mit 20 Punkten, deren x- und y-Koordinaten leicht korrelieren, wodurch sich ein einigermaßen linearer Verlauf ergibt. Es gibt allerdings am Punkt 5 einen starken Ausreißer nach oben und am Punkt 18 einen starken Ausreißer nach unten. Die Daten sollen nun durch drei Funktionen abgebildet werden. Die grüne Funktion besteht aus lediglich einem einzigen Koeffizienten. Die Funktion muss also Informationen aus den Daten generalisieren und kann sich nicht an deren Eigenheiten anpassen. Sie passt sich daher nicht an jeden einzelnen Punkt an, sondern bildet eher den generellen Trend in den Daten ab. Die Ausreißer werden dabei größtenteils ignoriert. Die orangene Funktion besteht aus fünf Koeffizienten und kann sich daher deutlich besser an die Punkte anpassen, auch die Ausreißer fallen sehr viel stärker ins Gewicht. Die rote Funktion besteht aus 50 Koeffizienten und kann sich perfekt an jeden einzelnen Punkt inklusive der Ausreißer anpassen. Sie bildet die Trainingsdaten mit all ihren Beschaffenheiten perfekt ab und kann selbst die Ausreißer perfekt einfangen. Die zahlreichen Koeffizienten geben der Funktion die Möglichkeit, sich so stark an die Daten anzupassen, dass keine Generalisierung notwendig ist. Es wurden so gut wie keine Informationen aus den Daten generalisiert, die auf neue Punkte angewandt werden könnten.<sup>130</sup>

**Abbildung 15: Beispiel für Überanpassungen**



<sup>129</sup> Vgl. Hirschle, J., Machine Learning für Zeitreihen, 2020, S. 101 ff.

<sup>130</sup> Beispiel in Anlehnung an scikit-learn developers, Overfitting, 2021.

Quelle: In Anlehnung an *scikit-learn developers*, Overfitting, 2021.

Wird die Ordnung bei einem ARIMA-Modell zu niedrig gewählt, werden möglicherweise nützliche Autokorrelationen nicht in das Modell einbezogen und die Genauigkeit der Vorhersagen ist geringer als eigentlich möglich, was auch als Unteranpassung bezeichnet wird.<sup>131</sup> Wird die Ordnung zu hoch gewählt, besteht die Gefahr der Überanpassung an die Trainingsdaten.<sup>132</sup>

Um zu überprüfen, ob ein Modell über- oder unterangepasst ist, werden die verfügbaren Daten in der Regel in zwei Teile aufgeteilt: Trainingsdaten (meist circa 70% bis 90%) und Testdaten (dementsprechend meist 10% bis 30%).<sup>133</sup> Die Trainingsdaten werden vom Algorithmus zur Erstellung des Modells verwendet und als bekannte Daten bezeichnet. Danach wird das Modell getestet, indem es die unbekannten Testdaten vorhersagen soll.<sup>134</sup> Die Vorhersage kann mit den echten Testdaten verglichen werden. Zusätzlich kann das Modell getestet werden, indem es die Trainingsdaten „vorhersagen“ soll. Die Genauigkeit bei der Vorhersage der Testdaten und Trainingsdaten kann verglichen werden. Wenn die Genauigkeit bei den Trainingsdaten erheblich höher ist als bei den Testdaten, deutet dies auf eine Überanpassung hin.<sup>135</sup>

Das allgemeine Vorgehen zum Testen und Bewerten von Modellen zur Zeitreihenvorhersage wird im Kapitel 2.6 weiter beschrieben. Zur Bewertung von ARIMA-Modellen gibt es unter anderem zwei zusätzliche besondere Kenngrößen, mit denen sich die Modelle bewerten und untereinander vergleichen lassen: Die Log-Likelihood und das Akaike Information Criterion (AIC).<sup>136</sup>

Die Log-Likelihood bewertet, wie gut sich ein Modell an die Trainingsdaten angepasst hat. Generell gilt, je höher der Wert, desto besser die Anpassung an die Trainingsdaten.<sup>137</sup> Die Skala der Log-Likelihood ist dabei abhängig von den zugrundeliegenden Trainingsdaten. Es lassen sich zwar unterschiedliche Modelle vergleichen, allerdings muss die Trainingsdatenbasis stets exakt gleich sein.<sup>138</sup> Das AIC setzt die Log-Likelihood zusätzlich in den Kontext der Komplexität des Modells. Da

---

<sup>131</sup> Vgl. *Lazzeri, F.*, Machine Learning, 2021, S. 43.

<sup>132</sup> Vgl. *Hirschle, J.*, Machine Learning für Zeitreihen, 2020, S. 96 ff.

<sup>133</sup> Vgl. *Raschka, S., Mirjalili, V.*, Machine Learning mit Python, 2021, S. 145 ff.

<sup>134</sup> Vgl. *Lazzeri, F.*, Machine Learning, 2021, S. 42 ff.

<sup>135</sup> Vgl. *Raschka, S., Mirjalili, V.*, Machine Learning mit Python, 2021, S. 227 ff.

<sup>136</sup> Vgl. *Auffarth, B.*, Machine Learning for Time Series, 2021, S. 139 f.

<sup>137</sup> Vgl. *Box et al.*, Time Series Analysis, 2016, S. 209 ff.

<sup>138</sup> Vgl. *Hirschle, J.*, Machine Learning für Zeitreihen, 2020, S. 125 f.

komplexere Modelle wie oben beschrieben zu einer besseren Abbildung der Trainingsdaten durch Überanpassung neigen, werden simplere Modelle beim AIC bevorzugt.<sup>139</sup> Ein simples Modell wird einem komplexen Modell mit einer geringfügig höheren Log-Likelihood vorgezogen.<sup>140</sup> Wie auch die Log-Likelihood lässt sich das AIC nicht absolut interpretieren und ist nur bei Modellen mit gleicher Trainingsdatenbasis vergleichbar. Generell gilt, je niedriger das AIC, desto besser ist das Modell.<sup>141</sup>

#### 2.4.6 Kontexteffekte und Kausalitäten (SARIMAX)

Der ARIMA-Algorithmus basiert im Wesentlichen auf den Autokorrelationen der zu verarbeitenden, univariaten Zeitreihe. Diese Zeitreihe wird auch als endogen beziehungsweise als endogene Daten bezeichnet. Autokorrelationen bestehen stets nur in der Zeitreihe selbst. Eine Zeitreihe kann jedoch auch von anderen Effekten abhängig sein, welche sich nicht oder nur bedingt in den Autokorrelationen widerspiegeln. Es handelt sich dabei um Korrelationen oder sogar Kausalitäten, die auch als Kontexteffekte beziehungsweise exogene Daten bezeichnet werden.<sup>142</sup>

Wenn zwei Variablen korrelieren, dann bewegen sie sich in einem bestimmten Maße in die gleiche oder entgegengesetzte Richtung.<sup>143</sup> Betrachtet man beispielsweise einen Badestrand, so korreliert die Menge an verkaufter Eiscreme sehr wahrscheinlich positiv mit der Menge an verkaufter Sonnencreme, da sich beide Variablen in eine ähnliche Richtung bewegen.<sup>144</sup> Es lässt sich dann gegebenenfalls von der einen Variable auf die andere schließen, allerdings stehen Korrelationen nicht zwangsläufig in einem kausalen Zusammenhang<sup>145</sup>, was auch als Scheinkorrelation<sup>146</sup> bezeichnet wird. Stattdessen führen eher hohe Temperaturen dazu, dass sich mehr Badegäste am Strand einfinden, wodurch dann auch mehr Eiscreme verkauft wird. Bei der Temperatur gibt es also einen kausalen Zusammenhang<sup>147</sup>, der sich auf die anderen beiden Variablen auswirkt. Diese Information kann ebenfalls von einem Modell für die Vorhersage von Zeitreihen genutzt werden.<sup>148</sup>

---

<sup>139</sup> Vgl. Auffarth, B., Machine Learning for Time Series, 2021, S. 139 f.

<sup>140</sup> Vgl. Hirschle, J., Machine Learning für Zeitreihen, 2020, S. 125 f.

<sup>141</sup> Vgl. Hyndman, R. J., Athanasopoulos, G., Forecasting, S. 131 f.

<sup>142</sup> Vgl. Hirschle, J., Machine Learning für Zeitreihen, 2020, S. 142 ff.

<sup>143</sup> Vgl. Statista, Korrelation, 2021.

<sup>144</sup> Vgl. Hyndman, R. J., Athanasopoulos, G., Forecasting, S. 149 ff.

<sup>145</sup> Vgl. Frochte, J., Grundlagen des Machine Learning, 2020, S. 87 ff.

<sup>146</sup> Vgl. Statista, Scheinkorrelation, 2021.

<sup>147</sup> Vgl. Statista, Kausalität, 2021.

<sup>148</sup> Vgl. Hyndman, R. J., Athanasopoulos, G., Forecasting, S. 149 ff.

Ein ARIMA- oder SARIMA-Modell kann exogene Daten in die Vorhersage einbeziehen. Dabei wird für jedes exogene Merkmal jeweils ein weiteres Modul an das Modell beziehungsweise ein weiterer Term an die Regressionsgleichung angehängt, der dann die exogenen Daten mit eigenen Korrelationskoeffizienten durch lineare Regression in das Modell einbezieht.<sup>149</sup> Die exogenen Daten werden dann verwendet, um die Zeitreihe vorherzusagen. Daraus ergibt sich eine Herausforderung: Die exogenen Daten müssen für die Erstellung der Vorhersage im Vorhinein bekannt sein.<sup>150</sup> Wenn beispielsweise der Verkauf von Eiscreme für den nächsten Tag vorhergesagt werden soll und die Temperatur als exogenes Merkmal einbezogen wird, dann muss die Temperatur für den nächsten Tag bekannt sein. Wenn es sich bei dem exogenen Merkmal selbst um ein vorhergesagtes Merkmal handelt, dann können eventuelle Vorhersagefehler beim exogenen Merkmal die weiteren Vorhersagen verschlechtern. Wird also die Temperatur falsch vorhergesagt, wird sich die darauf basierende Vorhersage des Verkaufs von Eiscreme ebenfalls verschlechtern.<sup>151</sup>

Ein um exogene Daten erweitertes SARIMA-Modell wird als SARIMAX-Modell („SARIMA eXogenous“) bezeichnet.<sup>152</sup> Da es sich bei den Modulen für exogene Daten nur um eine einfache lineare Regression auf die entsprechenden Merkmale handelt, müssen anders als bei den anderen Modulen keine Lags oder ähnliche Parameter angegeben werden. Die exogenen Module sind daher nicht unbedingt in der Notation SARIMAX(p, d, q)(P, D, Q)m<sup>153</sup> oder SARIMAX(p, d, q)(P, D, Q, m)<sup>154</sup> enthalten. Im Folgenden werden die exogenen Variablen in [eckigen Klammern] notiert, also beispielsweise: SARIMAX(p, d, q)(P, D, Q)m [Merkmal-1, Merkmal-2, ...].

Bei ARIMA bzw. SARIMAX müssen nicht zwangsweise alle Lags von Null bis p, d, P oder Q verwendet werden. Sollen beispielsweise nur das erste, zweite und fünfte Lag für p verwendet werden, lässt sich dies durch SARIMAX([1,2,5], d, q)(P, D, Q)m notieren.<sup>155</sup>

---

<sup>149</sup> Vgl. *Hirschle, J.*, Machine Learning für Zeitreihen, 2020, S. 142 ff.

<sup>150</sup> Vgl. *Lazzeri, F.*, Machine Learning, 2021, S. 9 f.

<sup>151</sup> Vgl. *Hirschle, J.*, Machine Learning für Zeitreihen, 2020, S. 142 ff.

<sup>152</sup> Vgl. *Patel, A., Vishwas, B. V.*, Time Series Analysis Python, 2020, S. 143 ff.

<sup>153</sup> Vgl. *Jansen, S.*, Machine Learning for Trading, 2020, S. 270.

<sup>154</sup> Vgl. *statsmodels*, SARIMAX, 2022.

<sup>155</sup> Vgl. *statsmodels*, SARIMAX, 2022.

## 2.5 Long Short-Term Memory

Der Long Short-Term Memory-Algorithmus (LSTM) ist ein weit verbreiteter Algorithmus für die Verarbeitung und Vorhersage von Zeitreihen. Der LSTM-Algorithmus erstellt ein rekurrentes neuronales Netz, weshalb es sich um einen Algorithmus aus dem Bereich des sogenannten Deep Learning handelt.<sup>156</sup> LSTM wurde im Jahr 1997 von den deutschen Informatikern Sepp Hochreiter und Jürgen Schmidhuber<sup>157</sup> erstmals vorgestellt.<sup>158</sup>

Im Folgenden werden zunächst die Grundlagen neuronaler Netze dargestellt. Anschließend wird genauer auf rekurrente Netze beziehungsweise LSTM und deren Arbeitsweise eingegangen.

### 2.5.1 Arbeitsweise und Architektur neuronaler Netze

Ein neuronales Netz besteht aus Neuronen. Ein Neuron verarbeitet eine Eingabe mittels einer bestimmten Funktion und leitet das Ergebnis an andere Neuronen weiter. Neuronen, die auf der gleichen Ebene in einem Netz arbeiten, werden in einer sogenannten Schicht zusammengefasst.<sup>159</sup> Die Abbildung 16 zeigt ein beispielhaftes neuronales Netz:

Abbildung 16: Aufbau eines neuronalen Netzes

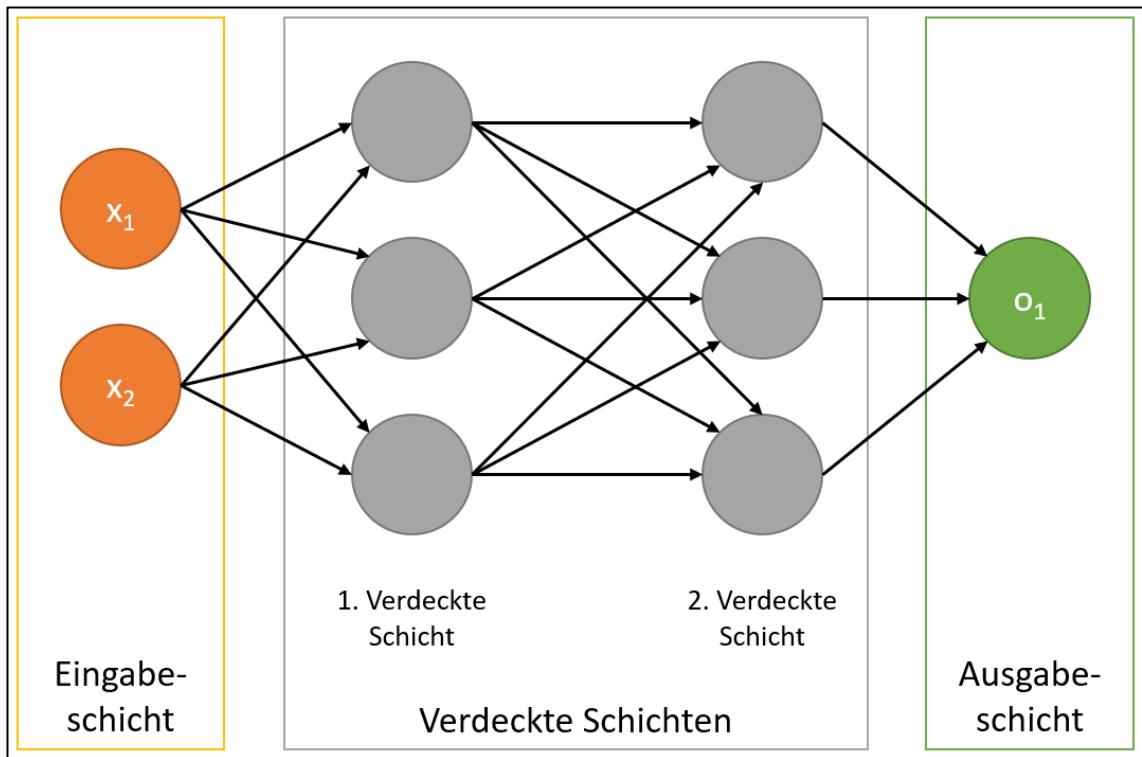
---

<sup>156</sup> Vgl. *Lazzeri, F.*, Machine Learning, 2021, S. 137 ff.

<sup>157</sup> Vgl. *Hochreiter, S., Schmidhuber, J.*, LSTM, 1997.

<sup>158</sup> Vgl. *Auffarth, B.*, Machine Learning for Time Series, 2021, S. 265.

<sup>159</sup> Vgl. *Raschka, S., Mirjalili, V.*, Machine Learning mit Python, 2021, S. 409 ff.



Quelle: In Anlehnung an IBM, Neurale Netzwerke, 2020.

Es gibt zunächst die Eingabeschicht, durch die die zu verarbeitenden Daten (beispielsweise vergangene Beobachtungen, Merkmale etc.) in das Netz geladen werden. Danach werden die Eingaben durch die Neuronen in den sogenannten verdeckten Schichten verarbeitet. Am Ende wandeln die Neuronen in der Ausgabeschicht die Ergebnisse der verdeckten Schichten um (beispielsweise in einen vorhergesagten Wert einer Zeitreihe).<sup>160</sup> Den Kern eines Neuronalen Netzes bilden die Neuronen. Die Abbildung 17 zeigt ein beispielhaftes Neuron. Wie bereits beschrieben, erhält ein Neuron zunächst eine Eingabe. Dabei handelt es sich in der Regel um mehrere Werte ( $x_1$  und  $x_2$ ). Jeder Eingabewert wird durch einen Gewichtungskoeffizienten ( $w_1$  und  $w_2$ ) gewichtet und aufsummiert. Oft wird noch eine Konstante addiert, welche als Bias ( $b$ ) bezeichnet wird und mit dem y-Achsenabschnitt (Intercept) einer Regression vergleichbar ist.<sup>161</sup> Im Neuron wird dann eine mathematische Formel, die sogenannte Aktivierungsfunktion ( $f$ ), auf die Summe der gewichteten Eingabewerte angewandt. Das Ergebnis wird anschließend als Ausgabe an die Neuronen in der nächsten Schicht weitergeleitet.<sup>162</sup> Die Anzahl an Schichten wird als Tiefe des

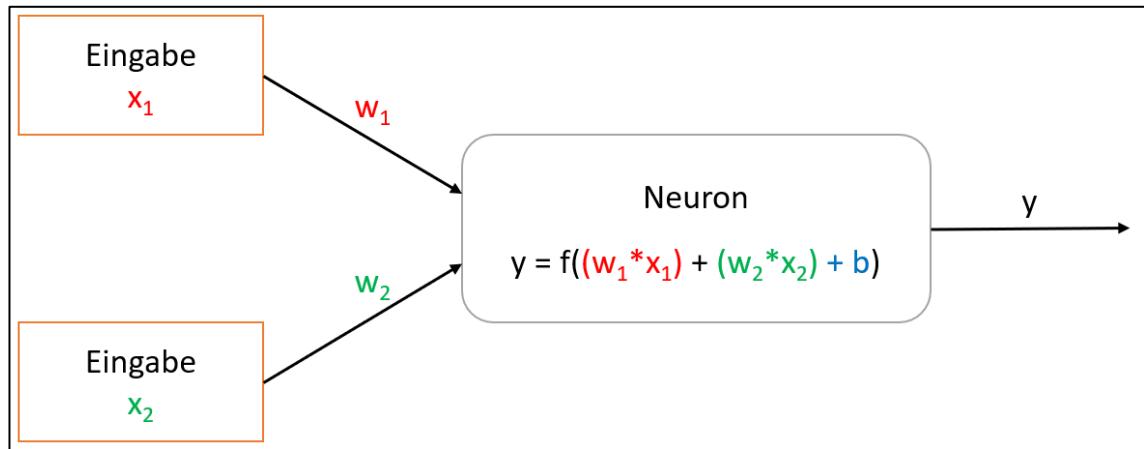
<sup>160</sup> Vgl. Hirschle, J., Machine Learning für Zeitreihen, 2020, S. 150 ff.

<sup>161</sup> Vgl. Babcock, J., Bali, R., Generative AI, 2021, S. 69 ff. und S. 73.

<sup>162</sup> Vgl. Lederer, J., Aktivierungsfunktionen, 2021.

Netzes bezeichnet, die Anzahl der Neuronen in einer Schicht wird als deren Bereit bezeichnet.<sup>163</sup>

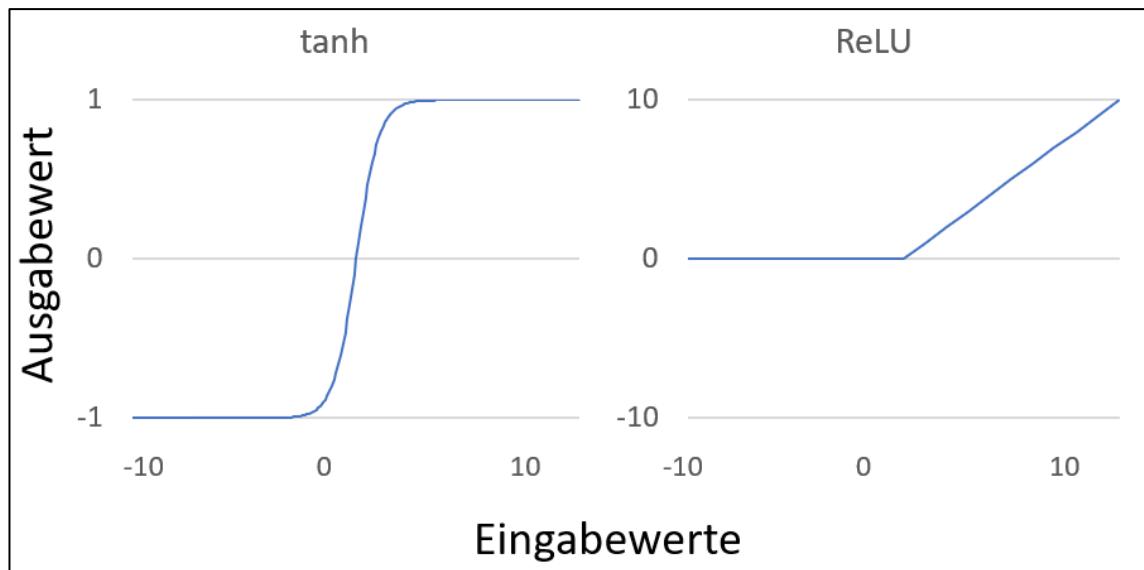
Abbildung 17: Aufbau und Funktion eines Neurons



Quelle: In Anlehnung an Babcock, J., Bali, R., Generative AI, 2021, S. 70.

Die Aktivierungsfunktion spielt eine entscheidende Rolle für das Verhalten des neuronalen Netzes. In der Praxis gibt es verschiedene gängige Aktivierungsfunktionen. Die Abbildung 18 zeigt zwei typische und häufig für Zeitreihenvorhersage verwendete Funktionen: ReLU und tanh.<sup>164</sup>

Abbildung 18: Aktivierungsfunktionen



<sup>163</sup> Vgl. Hirschle, J., Machine Learning für Zeitreihen, 2020, S. 155 f.

<sup>164</sup> Vgl. Hirschle, J., Machine Learning für Zeitreihen, 2020, S. 150 ff.

Quelle: In Anlehnung an *Korstanje, J.*, Advanced Forecasting, 2021, S. 211.

Auf der linken Seite ist die Tangens-Hyperbolicus-Funktion ( $\tanh$ ) abgebildet. Hierbei handelt es sich um eine nicht-lineare Funktion, welche einen Wert von -1 bis +1 erzeugt.<sup>165</sup> Auf der rechten Seite ist die „Rectified Linear Unit“-Funktion (ReLU) abgebildet, wobei es sich um eine teilweise lineare Funktion handelt. Positive Eingabewerte werden unverändert weitergegeben, allerdings werden negative Werte stets in 0 umgewandelt.<sup>166</sup>

Die Anzahl an Schichten und darin enthaltenen Neuronen sowie deren Aktivierungsfunktion bilden die Architektur des neuronalen Netzes, welche vor dem Training festgelegt wird.<sup>167</sup> Daneben spielen die Gewichtungskoeffizienten des neuronalen Netzes eine entscheidende Rolle. Sie werden beim Training beziehungsweise der Erstellung des Netzes bestimmt.<sup>168</sup> Das Verhalten unterschiedlicher Architekturen und besonders die Vor- und Nachteile der Aktivierungsfunktionen ergeben sich aus der Art und Weise, wie neuronale Netze trainiert werden und werden daher im weiteren Verlauf näher dargestellt.

### 2.5.2 Training neuronaler Netze

Während des Trainings werden die Gewichtungskoeffizienten bestimmt. Bei der Initialisierung werden die Gewichtungskoeffizienten zunächst mit zufälligen Werten besetzt.<sup>169</sup> Dann werden die Merkmale beziehungsweise Eingangsvariablen (beispielsweise Lags einer Zeitreihe) an die Eingangsschicht übergeben und so durch die Schichten und Neuronen geleitet (daher werden derartige Netze auch als „Feed-Forward-Netze“<sup>170</sup> bezeichnet). Am Ende werden die von den Neuronen verarbeiteten Daten durch die Ausgabeschicht entsprechend in Zielwerte umgewandelt (beispielsweise die Vorhersage für die nächste Beobachtung einer Zeitreihe). Dieser Prozess wird als Forward-Propagation bezeichnet.<sup>171</sup> Über eine Verlustfunktion wird dann ermittelt, wie weit die vom Netz vorhergesagten Werte von den wahren Werten abweichen. Diese Abweichungen werden von einer Optimierungsfunktion bewertet und die Gewichtungskoeffizienten so angepasst, dass die durch die Verlustfunktion

---

<sup>165</sup> Vgl. *Patel, A., Vishwas, B. V.*, Time Series Analysis Python, 2020, S. 192.

<sup>166</sup> Vgl. *Babcock, J., Bali, R.*, Generative AI, 2021, S. 69 ff. und S. 87.

<sup>167</sup> Vgl. *Lazzeri, F.*, Machine Learning, 2021, S. 143 ff.

<sup>168</sup> Vgl. *Korstanje, J.*, Advanced Forecasting, 2021, S. 211 f.

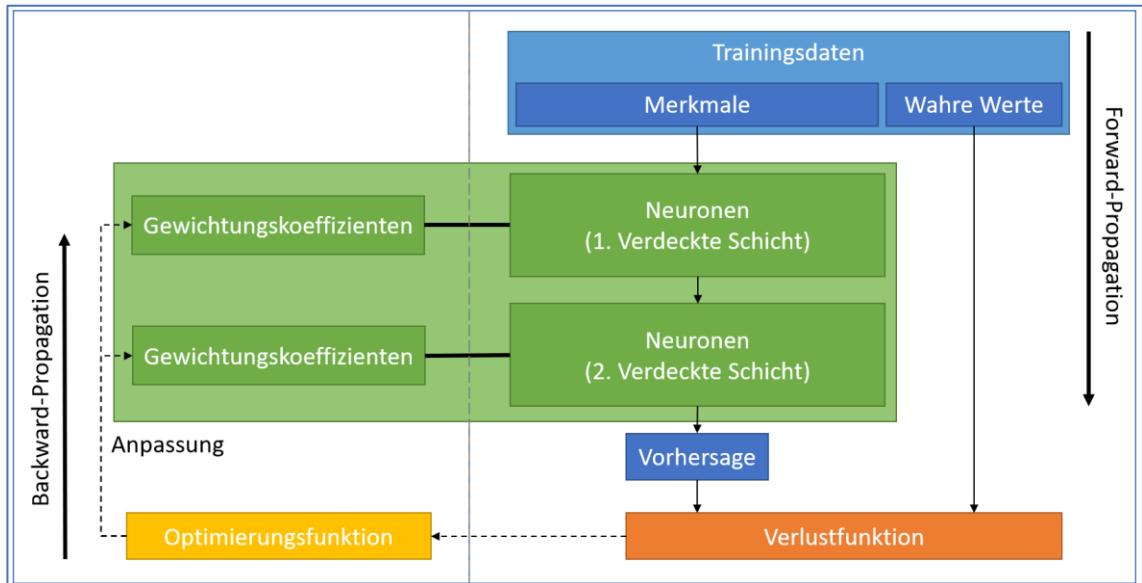
<sup>169</sup> Vgl. *Hirschle, J.*, Machine Learning für Zeitreihen, 2020, S. 170 f.

<sup>170</sup> Vgl. *Géron, A.*, Machine Learning, 2020, S. 290.

<sup>171</sup> Vgl. *Auffarth, B.*, Machine Learning for Time Series, 2021, S. 262 ff.

ermittelten Abweichungen geringer werden. Dieser Prozess wird dann als Backward-Propagation bezeichnet.<sup>172</sup> Die Abbildung 19 zeigt den Trainingsprozess eines neuronalen Netzes mit zwei verdeckten Schichten.

Abbildung 19: Trainingsprozess neuronaler Netze



Quelle: In Anlehnung an Hirschle, J., Machine Learning für Zeitreihen, 2020, S. 155.

Als Verlustfunktion (auch als „Loss“ bezeichnet) bieten sich im Falle der Zeitreihenvorhersage die typischen Metriken der Regression wie beispielsweise der durchschnittliche absolute Fehler an.<sup>173</sup> Die Metriken aus dem Bereich der Regression werden in Kapitel 2.6 näher beschrieben.

Die Optimierungsfunktion bewertet, wie sich die Gewichtungskoeffizienten ändern müssen, um die Ergebnisse der Verlustfunktion zu minimieren.<sup>174</sup> Um zu bestimmten, in welche Richtung ein Gewichtungskoeffizient geändert werden muss, wird das sogenannte Gradientenabstiegsverfahren verwendet<sup>175</sup>, welches in Abbildung 20 dargestellt ist. Während jedes Durchlaufs durch die Forward- und Backward-Propagation werden die von der Verlustfunktion berechneten Abweichungen (blau) mit dem jeweils verwendeten Gewichtungskoeffizienten (coef) in das Diagramm eingetragen, beispielsweise -4 und +3. Man geht davon aus, dass es einen optimalen Wert

<sup>172</sup> Vgl. Patel, A., Vishwas, B. V., Time Series Analysis Python, 2020, S. 196 ff.

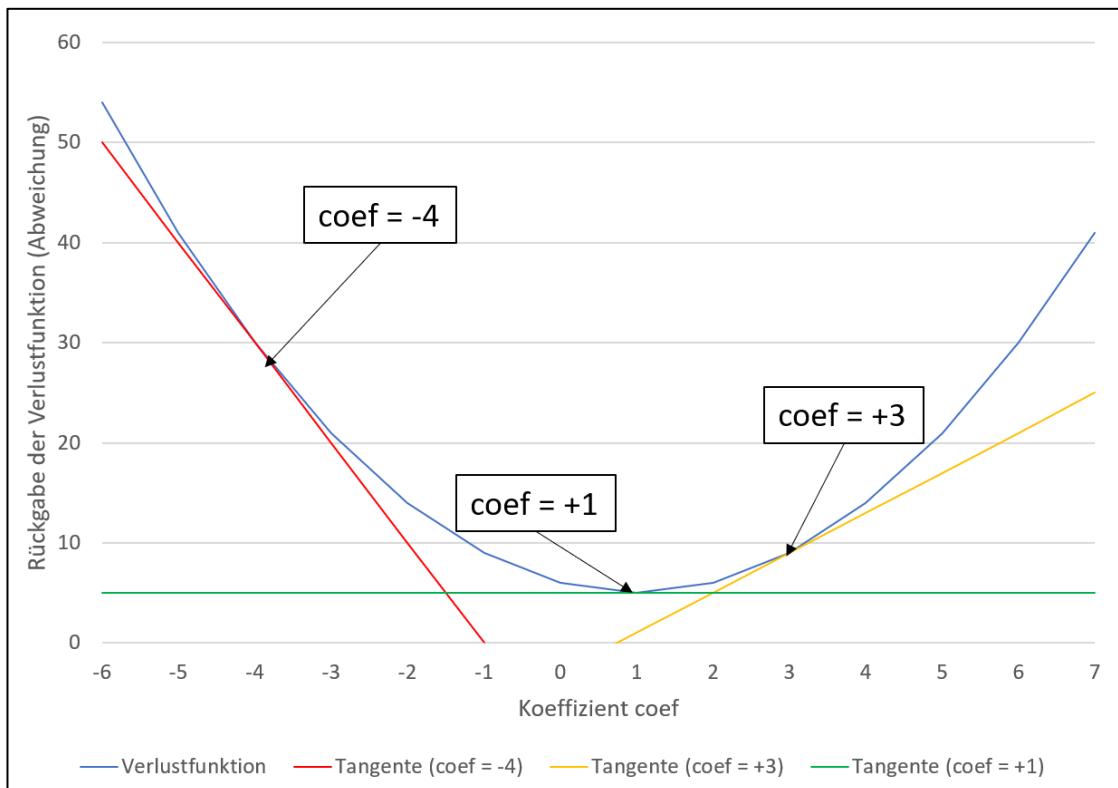
<sup>173</sup> Vgl. Le Guen, V., Thome, N., Loss Functions, 2019.

<sup>174</sup> Vgl. Babcock, J., Bali, R., Generative AI, 2021, S. 93 ff.

<sup>175</sup> Vgl. Lazzeri, F., Machine Learning, 2021, S. 146.

für den jeweiligen Gewichtungskoeffizienten gibt, bei dem die Verlustfunktion minimal ist. Liegt der Gewichtungskoeffizient darüber oder darunter, erhöht sich die Verlustfunktion wieder. Dementsprechend wird mittels Regression eine nicht-lineare Funktion gebildet, die die bekannten Punkte (-4 und +3) möglichst gut abbildet.<sup>176</sup> Nun kann per Differentialrechnung eine Tangente an die jeweiligen Punkte angelegt werden. Ist die Steigung wie bei  $\text{coef} = -4$  negativ (rot), kann man davon ausgehen, dass ein höherer Koeffizient die Verlustfunktion senkt. Ist die Steigung wie bei  $\text{coef} = +3$  positiv (gelb), kann man umgekehrt davon ausgehen, dass ein niedrigerer Koeffizient die Verlustfunktion senkt.<sup>177</sup> Bei  $\text{coef} = +1$  ist die Steigung neutral (grün), man kann also davon ausgehen, dass das Minimum der Verlustfunktion erreicht wurde und weitere Veränderungen am Gewichtungskoeffizienten keine Verbesserungen mehr herbeiführen. Auf diese Weise kann der optimale Wert für den jeweiligen Gewichtungskoeffizienten näherungsweise bestimmt werden. Das Gradientenabstiegsverfahren wird auf jedes Gewicht angewandt.<sup>178</sup>

Abbildung 20: Gradientenabstiegsverfahren



<sup>176</sup> Vgl. Hirschle, J., Machine Learning für Zeitreihen, 2020, S. 42 ff.

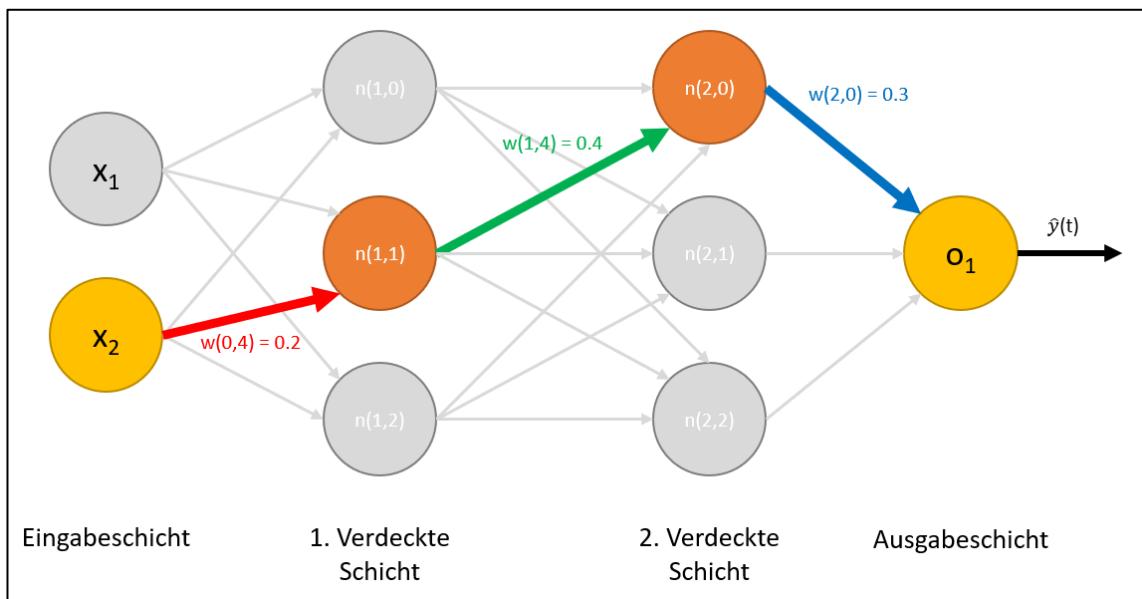
<sup>177</sup> Vgl. Géron, A., Machine Learning, 2020, S. 121 ff.

<sup>178</sup> Vgl. Hirschle, J., Machine Learning für Zeitreihen, 2020, S. 42 ff.

Quelle: In Anlehnung an *Hirschle, J.*, Machine Learning für Zeitreihen, 2020, S. 43.

Durch das oben dargestellte Gradientenabstiegsverfahren lässt sich zunächst ermitteln, ob und in welche Richtung sich ein Gewichtungskoeffizient ändern muss. Um zu bestimmten, wie stark sich das entsprechende Gewicht ändern muss, wird dessen Einfluss auf das Endergebnis berechnet.<sup>179</sup> Die Abbildung 21 zeigt einen Pfad durch ein neuronales Netz mit zwei verdeckten Schichten (der Einfachheit halber wird von einer linearen Aktivierungsfunktion ohne Bias ausgegangen, welche immer den Eingangswert zurückgibt und daher im Weiteren ignoriert werden kann).

**Abbildung 21: Pfad in einem neuronalen Netz**



Quelle: In Anlehnung an *Aggarwal, C.*, Neural Networks, 2018, S. 111.

Ein Eingangswert wird im eingezeichneten Pfad jeweils von den Gewichten  $w(0,4)$ ,  $w(1,4)$  und  $w(2,0)$  gewichtet, um den Ausgangswert des Pfades zu bestimmten, der dann mit dem tatsächlichen Wert verglichen werden kann, um mittels Verlustfunktion die Abweichung zu bestimmten.<sup>180</sup>

#### Formel 7: Formel für Pfad im neuronalen Netz

$$\text{Pfad}(x_2) = x_2 * \quad \textcolor{red}{w(0,4)} * \quad \textcolor{green}{w(1,4)} * \quad \textcolor{blue}{w(2,0)} \quad (1)$$

$$\text{Pfad}(x_2) = x_2 * \quad \textcolor{red}{0,2} \quad * \quad \textcolor{green}{0,4} \quad * \quad \textcolor{blue}{0,3} \quad (2)$$

<sup>179</sup> Vgl. *Babcock, J., Bali, R.*, Generative AI, 2021, S. 75 ff.

<sup>180</sup> Vgl. *Aggarwal, C.*, Neural Networks, 2018, S. 21 ff.

Quelle: In Anlehnung an Aggarwal, C., Neural Networks, 2018, S. 21 ff.

Durch die Kettenregel aus der Infinitesimalrechnung<sup>181</sup> lässt sich der Einfluss jedes Gewichts auf den Ausgangswert und damit auf die Abweichung des gesamten Netzes ermitteln, was auch als Gradient des Gewichts bezeichnet wird.<sup>182</sup> Letztendlich bestimmt dann die Lernrate<sup>183</sup>, wie stark das jeweilige Gewicht angepasst wird, indem der **Gradient ( $\nabla$ )** mit der **Lernrate ( $\eta$ )** multipliziert und das Produkt auf das Gewicht addiert oder davon subtrahiert wird.<sup>184</sup>

#### Formel 8: Veränderung eines Gewichts durch Gradienten und Lernrate

$$W_{\text{angepasst}} = W - \eta * \nabla \quad (1)$$

Quelle: In Anlehnung an Aunkofer, B., Deep Learning, 2019.

Je tiefer ein Netz, desto mehr Schichten sind enthalten. Dementsprechend sind die Pfade im Netz länger und haben wesentlich mehr Gewichte. Aus diesem Grund tritt bei tieferen Netzen häufig das Phänomen der sogenannten „verschwindenden Gradienten“ auf. Wenn in einem Pfad besonders viele Gewichtungskoeffizienten Werte unter eins annehmen, dann werden die Gradienten (vor allem der frühen Schichten) besonders klein.<sup>185</sup> In der Abbildung 21 entspräche der Gradient des ersten Gewichts im Pfad  $w(0,4)$  gerade einmal  $0,024$  ( $0,2 * 0,4 * 0,3$ ). Je kleiner die Gradienten werden, umso geringer wird das dazugehörige Gewicht angepasst. Dies kann dazu führen, dass sich die Gewichte der frühen Schichten gegebenenfalls kaum bis gar nicht verändern und daher nur stark eingeschränkt optimiert werden können.<sup>186</sup> Im umgekehrten Fall, wenn viele Gewichte Werte über Eins annehmen, werden die Gradienten und damit die Veränderungen der Gewichte besonders groß, was auch als explodierende Gradienten bezeichnet wird.<sup>187</sup>

Wenn sich die Gewichte aufgrund verschwindender Gradienten oder einer zu klein gewählten Lernrate nur sehr langsam verändern, dann benötigt es gegebenenfalls sehr viele Iterationen, um die Gewichte zu optimieren. Außerdem besteht die Gefahr,

<sup>181</sup> Vgl. Claster, W., Mathematik, 2020, S. 199 ff.

<sup>182</sup> Vgl. Babcock, J., Bali, R., Generative AI, 2021, S. 75 ff.

<sup>183</sup> Vgl. Raschka, S., Mirjalili, V., Machine Learning mit Python, 2021, S. 48 ff.

<sup>184</sup> Vgl. Aunkofer, B., Deep Learning, 2019.

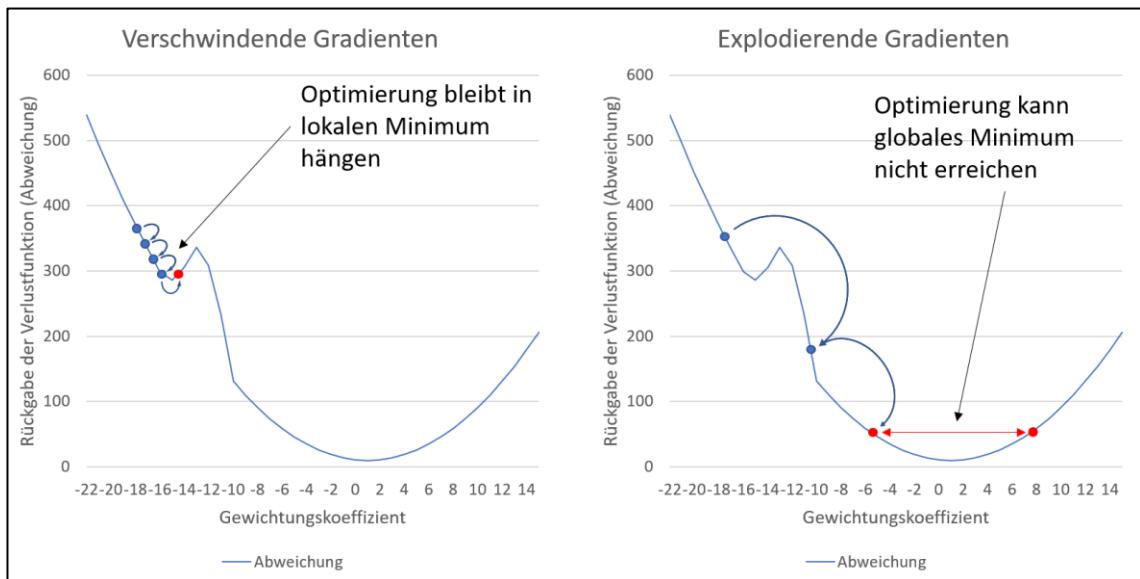
<sup>185</sup> Vgl. Babcock, J., Bali, R., Generative AI, 2021, S. 82 ff.

<sup>186</sup> Vgl. Aunkofer, B., Deep Learning, 2019.

<sup>187</sup> Vgl. Aggarwal, C., Neural Networks, 2018, S. 129 ff.

dass die Optimierung in einem lokalen Minimum der Verlustfunktion hängen bleibt.<sup>188</sup> Dieses Verhalten wird in Abbildung 22 auf der linken Seite dargestellt. Andererseits führen große Veränderungen der Gewichte auf einmal durch explodierenden Gradienten oder eine zu hoch gewählte Lernrate dazu, dass das Minimum Verlustfunktion gegebenenfalls nicht erreicht werden kann.<sup>189</sup>

**Abbildung 22: Verschwindende und explodierende Gradienten**



Quelle: In Anlehnung an Aunkofer, B., Deep Learning, 2019.

Neben den Gewichten und der Lernrate haben auch die bereits angesprochenen Aktivierungsfunktionen in den Neuronen (Abbildung 18) einen Einfluss auf das Training und das Problem der verschwindenden oder explodierenden Gradienten.<sup>190</sup> Die tanh-Funktion beispielsweise wandelt Eingangswerte in einen Wert zwischen -1 und +1 um. Die Funktion kann negative und positive Werte gut verarbeiten, allerdings tritt beim Einsatz eventuell das Problem der verschwindenden Gradienten auf.<sup>191</sup> Im Gegensatz tritt das Problem der verschwindenden Gradienten bei der ReLU-Funktion sehr selten auf. Allerdings können hierbei explodierende Gradienten auftreten. Zudem führen negative Eingangswerte dazu, dass das Neuron quasi deaktiviert wird und die Backward-Propagation gegebenenfalls nicht mehr richtig funktioniert.<sup>192</sup>

<sup>188</sup> Vgl. Géron, A., Machine Learning, 2020, S. 121 ff.

<sup>189</sup> Vgl. Patel, A., Vishwas, B. V., Time Series Analysis Python, 2020, S. 204 ff.

<sup>190</sup> Vgl. Raschka, S., Mirjalili, V., Machine Learning mit Python, 2021, S. 491 ff.

<sup>191</sup> Vgl. Aggarwal, C., Neural Networks, 2018, S. 12 f.

<sup>192</sup> Vgl. Patel, A., Vishwas, B. V., Time Series Analysis Python, 2020, S. 213 ff.

Der in Abbildung 19 dargestellte Trainingsprozess aus Forward- und Backward-Propagation wird iterativ wiederholt, um das Netz möglichst gut zu optimieren. Die Trainingsdaten werden dabei in Batches aufgeteilt. Nachdem alle Daten eines Batches den Forward-Propagation-Prozess durchlaufen haben, wird die Verlustfunktion berechnet und der Backward-Propagation-Prozess beginnt.<sup>193</sup> Der Durchlauf aller Batches wird als Epoche bezeichnet. Je mehr Epochen durchlaufen werden, desto mehr kann das Netz an die Trainingsdaten angepasst werden.<sup>194</sup>

### 2.5.3 Über- und Unteranpassung

Ähnlich wie beim ARIMA-Algorithmus kann es auch bei neuronalen Netzen zu Über- und Unteranpassungen an die Trainingsdaten kommen. Das Konzept der Über- und Unteranpassung wurde bereits in Kapitel 2.4.5 dargestellt. Bei Überanpassungen generalisiert der Algorithmus zu wenig Informationen aus den Daten und passt das Modell (in diesem Fall das neuronale Netz) sehr stark an die konkrete Beschaffenheit der Trainingsdaten an. Unbekannte Daten, die in ihrer Beschaffenheit von den Trainingsdaten abweichen, können dann nur schlecht vom Modell verarbeitet werden.<sup>195</sup>

Je breiter und tiefer ein neuronales Netz ist, desto mehr Gewichte sind darin enthalten. Dementsprechend hat das Netz zwar mehr Potenzial, um bestimmte Zusammenhänge und Komplexitäten abzubilden.<sup>196</sup> Ähnlich wie bei einem ARIMA-Modell mit sehr vielen Koeffizienten besteht aber auch bei einem Netz mit sehr vielen Gewichten die Gefahr, dass sich das Netz zu stark an die Trainingsdaten anpasst.<sup>197</sup> Tiefere neuronale Netze neigen daher nicht nur zu den verschwindenden oder explodierenden Gradienten, sondern auch zu Überanpassungen.<sup>198</sup> Es gibt aber zwei Möglichkeiten, um auch bei tiefen Netzen Überanpassungen vorzubeugen: Regularisierung und Dropouts.

Neuronale Netze neigen im Rahmen der Überanpassung dazu, an bestimmten Stellen besonders hohe Gewichte zu verwenden, um beispielsweise Ausreißer einzufangen.<sup>199</sup> Die Grundidee der Regularisierung ist es daher, die Verwendung von

---

<sup>193</sup> Vgl. Korstanje, J., Advanced Forecasting, 2021, S. 211 ff.

<sup>194</sup> Vgl. Auffarth, B., Machine Learning for Time Series, 2021, S. 315 ff.

<sup>195</sup> Vgl. Raschka, S., Mirjalili, V., Machine Learning mit Python, 2021, S. 101 ff.

<sup>196</sup> Vgl. Aggarwal, C., Neural Networks, 2018, S. 34.

<sup>197</sup> Vgl. Berk, R., Statistisches Lernen, 2020, S. 380.

<sup>198</sup> Vgl. Aggarwal, C., Neural Networks, 2018, S. 27 ff.

<sup>199</sup> Vgl. Hirschle, J., Machine Learning für Zeitreihen, 2020, S. 177 ff.

besonders hohen Gewichten zusätzlich zu bestrafen.<sup>200</sup> Dafür wird die Verlustfunktion um einen sogenannten Strafterm erweitert. Bei der Lasso- oder L1-Regularisierung werden die absoluten Werte aller Gewichte ( $\theta_i$ ) aufsummiert und mit der Regularisierungsstärke ( $\lambda$ ) multipliziert. Bei der Ridge- oder L2-Regularisierung wird die Summe der quadrierten Gewichte ( $\theta_i^2$ ) mit der Regularisierungsstärke ( $\lambda$ ) multipliziert. In beiden Fällen wird der Strafterm dann zur **Verlustfunktion** hinzuaddiert.<sup>201</sup>

#### Formel 9: L1- und L2-Regularisierung

$$\text{L1-Regularisierung: } \text{verlustfunktion} + \lambda * \sum_{i=1}^n |\theta_i| \quad (1)$$

$$\text{L2-Regularisierung: } \text{verlustfunktion} + \lambda * \sum_{i=1}^n \theta_i^2 \quad (2)$$

Quelle: In Anlehnung an *Hirschle, J.*, Machine Learning für Zeitreihen, 2020, S. 179.

Wenn also besonders hohe Gewichtungskoeffizienten verwendet werden, um die **Verlustfunktion** zu senken, dann wird gleichzeitig der Strafterm erhöht. Der Einsatz hoher Gewichtungskoeffizienten ist damit nicht generell verboten, allerdings lohnt es sich nur noch bei einer signifikanten Verbesserung der Verlustfunktion.<sup>202</sup> Die Regularisierungsstärke ( $\lambda$ ) bestimmt darüber, wie stark oder schwach höhere Gewichte beim Trainingsprozess bestraft werden.<sup>203</sup>

Beim Dropout wird der Anlernprozess des Netzes bewusst gestört, um Überanpassungen zu vermeiden.<sup>204</sup> Neuronale Netze neigen dazu, nicht alle Neuronen gleichermassen zu trainieren und in das Netz einzubeziehen. Stattdessen bilden die Neuronen oft bestimmte Cluster, in denen dann jeweils nur ein Teil der Neuronen aktiv ist oder sich auf bestimmte Aufgaben wie die Verarbeitung von Ausreißern spezialisiert.<sup>205</sup> Beim Dropout wird in jeder Epoche ein bestimmter Teil der Neuronen zufällig ausgewählt und deaktiviert, um die Bildung solcher Cluster zu vermeiden. Stattdessen müssen alle Neuronen (möglichst) gleichmäßig in den Anlernprozess einbezogen werden.<sup>206</sup>

**Abbildung 23: Dropout in neuronalem Netz**

<sup>200</sup> Vgl. *Raschka, S., Mirjalili, V.*, Machine Learning mit Python, 2021, S. 101 ff.

<sup>201</sup> Vgl. *Babcock, J., Bali, R.*, Generative AI, 2021, S. 219.

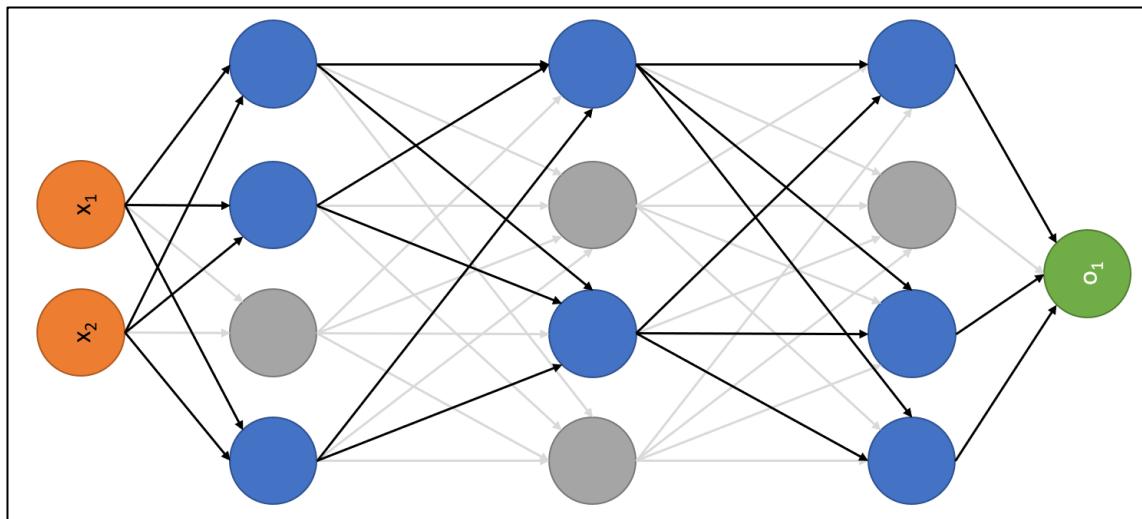
<sup>202</sup> Vgl. *Géron, A.*, Machine Learning, 2020, S. 367 ff.

<sup>203</sup> Vgl. *Raschka, S., Mirjalili, V.*, Machine Learning mit Python, 2021, S. 101 ff.

<sup>204</sup> Vgl. *Aggarwal, C.*, Neural Networks, 2018, S. 34 ff.

<sup>205</sup> Vgl. *Hirschle, J.*, Machine Learning für Zeitreihen, 2020, S. 182 ff.

<sup>206</sup> Vgl. *Raschka, S., Mirjalili, V.*, Machine Learning mit Python, 2021, S. 566 ff.



Quelle: In Anlehnung an Babcock, J., Bali, R., Generative AI, 2021, S. 88.

Umgekehrt können sich neuronale Netze auch unteranpassen. In diesem Fall gelingt es dem Modell wie bereits beschrieben nicht, die Zusammenhänge zwischen den Eingabedaten und den vorherzusagenden Ergebnissen zu erlernen.<sup>207</sup> Eine zu hoch gewählte Regularisierungsstärke kann beispielsweise dazu führen, dass Gewichte nicht mehr ausreichend angepasst werden.<sup>208</sup> Ein zu hoher Dropout kann den Lernprozess zu stark stören und gegebenenfalls verhindern, dass das Netz bestimmte Aufgaben erfüllen kann.<sup>209</sup> Auch die Länge des Trainingsprozesses ist entscheidend. Je mehr Epochen ein Netz trainiert wird, desto häufiger werden die Gewichte angepasst. Falls das Training zu lang ist, wird das Netz zu stark an die Trainingsdaten angepasst.<sup>210</sup> Falls das Training jedoch zu kurz ist, können eventuell aussagekräftige Zusammenhänge möglicherweise nicht erkannt werden.<sup>211</sup>

#### 2.5.4 Rekurrente neuronale Netze und LSTM

Die Einflüsse vergangener Beobachtungen sind wie bereits erwähnt essenziell für die Zeitreihenvorhersage (siehe Kapitel 2.2 und 2.4). Beim ARIMA-Algorithmus wird jede vergangene Beobachtung, die für die Erstellung einer Vorhersage einbezogen wird, als Eingangsparameter an das Modell übergeben und mit einem eigenen Koeffizienten verarbeitet (siehe Kapitel 2.4).

<sup>207</sup> Vgl. IBM, Underfitting, 2021.

<sup>208</sup> Vgl. Raschka, S., Mirjalili, V., Machine Learning mit Python, 2021, S. 101 ff.

<sup>209</sup> Vgl. Aggarwal, C., Neural Networks, 2018, S. 188 ff.

<sup>210</sup> Vgl. Deshpande, A., Kumar, M., AI for Big Data, 2018, S. 116 ff.

<sup>211</sup> Vgl. IBM, Underfitting, 2021.

Bei neuronalen Netzen hat sich allerdings (unter anderem) ein anderes Verfahren durchgesetzt, bei dem die Architektur durch Rückkoppelungen zu sogenannten rekurrenten neuronalen Netzen (RNN)<sup>212</sup> erweitert wird.<sup>213</sup> Dabei werden die Beobachtungen eines Merkmals nicht als einzelne Werte unabhängig voneinander verarbeitet, sondern im Kontext der vorangegangenen Beobachtungen.<sup>214</sup> Rekurrente Netze verarbeiten die Eingangsdaten mittels sogenannter Zeitfenster. Die Länge eines Zeitfensters kann beliebig festgelegt werden.<sup>215</sup> Die Tabelle 4 zeigt eine beispielhafte Zeitreihe  $y(t)$ . Fasst das Zeitfenster beispielsweise drei Beobachtungen, dann bilden die Beobachtungen für  $t = 1, t = 2$  und  $t = 3$  das erste Zeitfenster, die Beobachtungen für  $t = 2, t = 3$  und  $t = 4$  das zweite und so weiter, bis zum letzten Zeitfenster aus den Beobachtungen für  $t = 4, t = 5$  und  $t = 6$ . Für eine Vorhersage von  $y(7)$  wird dann das rot markierte Zeitfenster aus  $y(6), y(5)$  und  $y(4)$  verwendet.<sup>216</sup>

**Tabelle 4: Zeitreihe mit Zeitfenster**

t	y(t)
1	15
2	17
3	21
4	20
5	18
6	15

Quelle: Eigene Darstellung

In einem rekurrenten Netz sind die Neuronen rückgekoppelt. Es gibt darin nicht für jede Beobachtung eines Merkmals jeweils ein eigenes (Eingangs-)Neuron mit eigenen Gewichten, stattdessen verarbeitet ein Neuron jede Beobachtung des jeweiligen

---

<sup>212</sup> Vgl. Patel, A., Vishwas, B. V., Time Series Analysis Python, 2020, S. 202 ff.;  
Vgl. IBM, Recurrent Neural Networks, 2020.

<sup>213</sup> Vgl. TensorFlow.org, Time Series Forecasting, 2022;  
Vgl. IBM, Recurrent Neural Networks, 2021;  
Vgl. Luber, S., Litzel, N., rekurrente neuronale Netze, 2019.

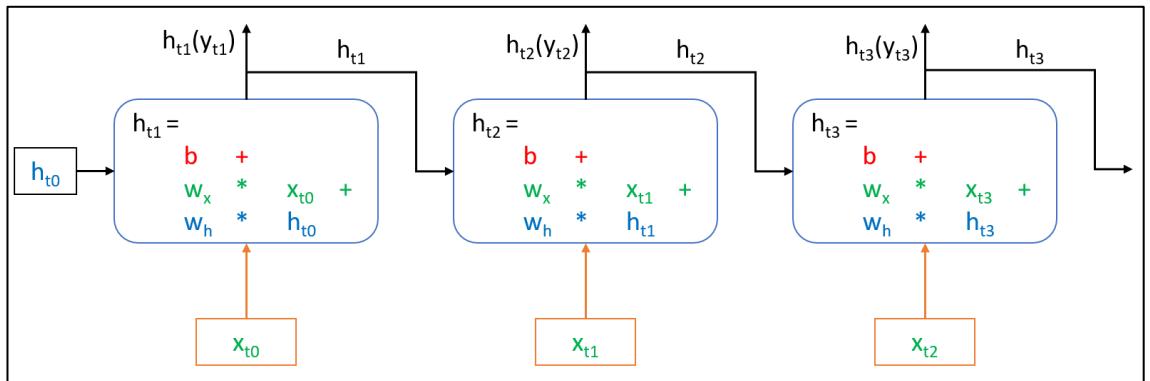
<sup>214</sup> Vgl. Aggarwal, C., Neural Networks, 2018, S. 271 ff.

<sup>215</sup> Vgl. Hirschle, J., Machine Learning für Zeitreihen, 2020, S. 186 ff.

<sup>216</sup> Beispiel in Anlehnung an Hirschle, J., Machine Learning für Zeitreihen, 2020, S. 182 ff.

Merkmals mit den selben Neuronen und Gewichten unter Berücksichtigung der vorherigen Beobachtungen.<sup>217</sup> Dieser Prozess ist in Abbildung 24 dargestellt:

Abbildung 24: Neuron in rekurrenter Schicht



Quelle: In Anlehnung an Hirschle, J., Machine Learning für Zeitreihen, 2020, S. 182 ff.

Im Neuron einer rekurrenten Schicht gibt es dann jeweils zwei Eingabewerte, den zu verarbeitenden Wert aus den Eingangsdaten  $x_t$  und den Zustand  $h_t$ . So wird beispielsweise im ersten Schritt der Wert für  $y(4)$  als  $x_t$  beziehungsweise  $x_{t0}$  übergeben. Da es keine vorherigen Zustände gibt, wird für  $h_t$  beziehungsweise  $h_{t0}$  einfach 0 übergeben. Der **Zustand**  $h_{t0}$  und die **Eingabedaten**  $x_{t0}$  werden dann jeweils mit den Gewichtungskoeffizienten  $w_h$  und  $w_x$  gewichtet, zusätzlich wird der **Bias**  $b$  addiert und gegebenenfalls aktiviert.<sup>218</sup> Das Neuron erzeugt dadurch einen neuen Zustand  $h_{t1}$  als Ausgabe. Im nächsten Schritt verarbeitet das Neuron den nächsten Wert der Eingabedaten  $y(5)$ . Es erhält dafür als Eingangswerte  $y(5)$  als  $x_{t1}$  und den vergangenen Zustand  $h_{t1}$ . Auf diese Weise können Zustände beziehungsweise Informationen aus vergangenen Beobachtungen in die Verarbeitung der aktuellen Beobachtung einbezogen werden. Dieser Prozess wird für alle Werte aus den Eingangsdaten wiederholt.<sup>219</sup> Das Netz kann entweder den letzten Zustand als finalen Ausgabewert erzeugen oder aber alle Zustände an die nächste Schicht weiterreichen.<sup>220</sup> Die Abbildung 25 zeigt die zusammengefasste Darstellung eines rekurrenten Neurons.

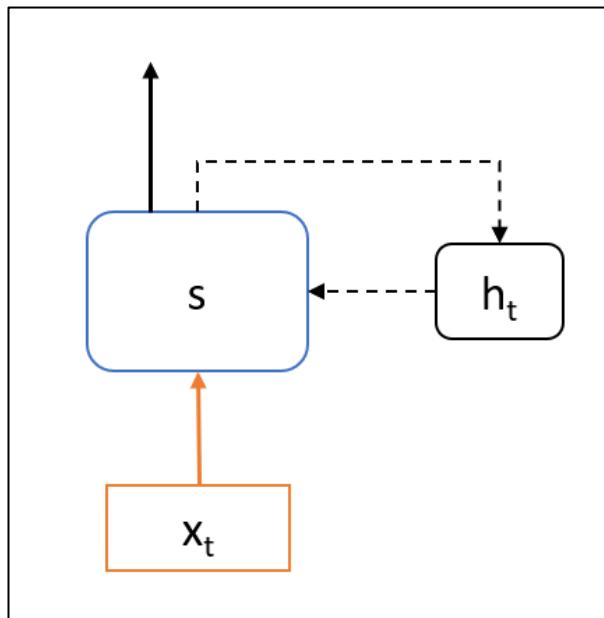
Abbildung 25: Rekurrentes Neuron

<sup>217</sup> Vgl. Géron, A., Machine Learning, 2020, S. 502 ff.

<sup>218</sup> Vgl. Patel, A., Vishwas, B. V., Time Series Analysis Python, 2020, S. 205 f.

<sup>219</sup> Vgl. Babcock, J., Bali, R., Generative AI, 2021, S. 321 ff.

<sup>220</sup> Vgl. TensorFlow.org, LSTM, 2022.



Quelle: In Anlehnung an *Lazzeri, F.*, Machine Learning, 2021, S. 145 ff.

Beim Training wird eine rekurrente Schicht wie in Abbildung 24 dargestellt in mehrere zeitliche Schichten ausgerollt. Wird das Netz beispielsweise mit einem Zeitfenster der Länge drei trainiert, ergeben sich daraus drei zeitliche Schichten, auf die dann der Backward-Propagation-Mechanismus angewandt wird (daher auch als Backward-Propagation-Through-Time bezeichnet).<sup>221</sup> Dadurch werden rekurrente Netze sehr schnell besonders tief und neigen daher zum Problem der verschwindenden oder explodierenden Gradienten. Außerdem nimmt der Einfluss früherer Beobachtungen des Zeitfensters auf spätere Zustände stark ab, was auch als „vergessen“ bezeichnet wird.<sup>222</sup> Beim LSTM-Algorithmus handelt es sich um eine Erweiterung der rekurrenten Netze zur Kompensation dieser Probleme.<sup>223</sup>

Bei einfachen rekurrenten Netzen wird nur ein Zustand  $h$  vom einem zum nächsten Verarbeitungsschritt übergeben, was als Kurzzeitgedächtnis bezeichnet wird. Beim LSTM-Algorithmus wird zusätzlich noch ein Zellzustand  $C$  übergeben, der als Langzeitgedächtnis dient und Informationen aus früheren Verarbeitungsschritten behalten kann.<sup>224</sup> Außerdem werden drei zusätzliche sogenannte Gatter hinzugefügt. Jedes Gatter verarbeitet die Eingabe mittels einer Sigmoid-Funktion ( $\sigma$ ), die Werte zwischen 0 und 1 zurückgibt und darüber entscheidet, wie viel der Eingabe durch das

<sup>221</sup> Vgl. *Lazzeri, F.*, Machine Learning, 2021, S. 145 ff.

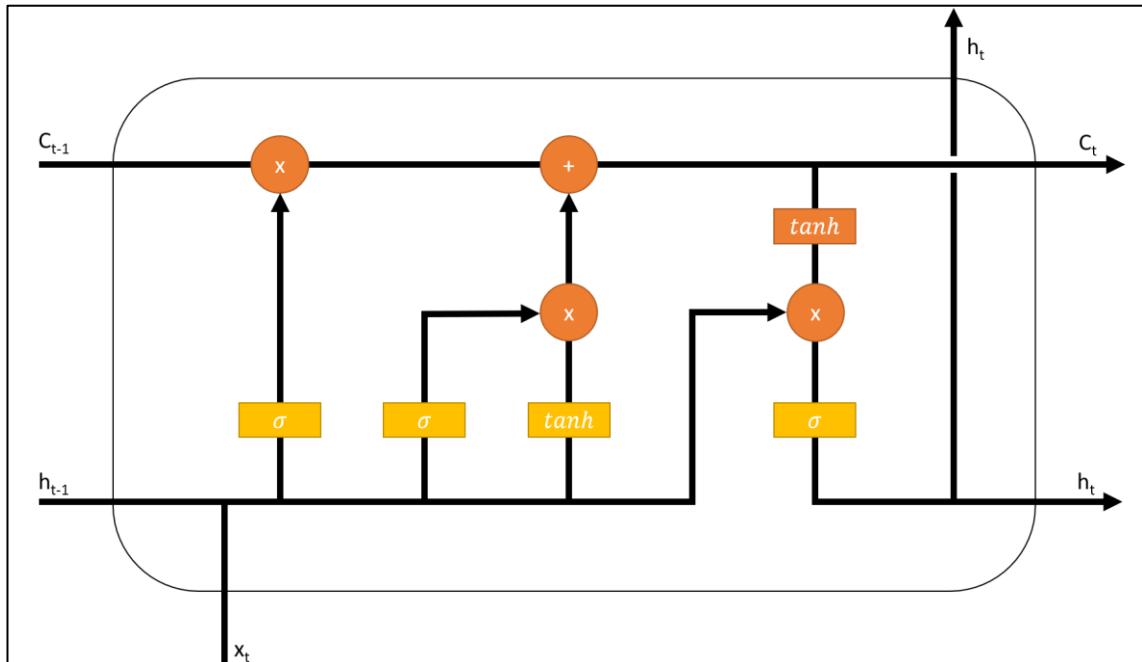
<sup>222</sup> Vgl. *Patel, A., Vishwas, B. V.*, Time Series Analysis Python, 2020, S. 206 ff.

<sup>223</sup> Vgl. *Luber, S., Litzel, N.*, LSTM, 2018.

<sup>224</sup> Vgl. *Raschka, S., Mirjalili, V.*, Machine Learning mit Python, 2021, S. 610 ff.

Gatter weitergeleitet wird. Diese Sigmoid-Funktionen werden ebenfalls mit eigenen Gewichtungskoeffizienten versehen.<sup>225</sup> Die Abbildung 26 zeigt ein Neuron in einer LSTM-Schicht.

Abbildung 26: Neuron in LSTM-Schicht



Quelle: In Anlehnung an Babcock, J., Bali, R., Generative AI, 2021, S. 92.

Die Abbildung 27 (siehe unten) zeigt die Verarbeitungsschritte im Neuron. Zunächst wird im rot markierten Teil der Zustand der vergangenen Verarbeitung  $h_{t-1}$  (Kurzzeitgedächtnis) mit der aktuellen Eingangsvariable  $x_t$  an das erste, sogenannte „Löschen-Gatter“ übergeben. Es entscheidet darüber, welche Informationen aus dem vorherigen Zustand relevant sind.<sup>226</sup> Im grün markierten, zweiten Schritt entscheidet das sogenannte „Eingabe-Gatter“, welche Informationen für die Erzeugung des neuen Zellzustand  $C_t$  (Langzeitgedächtnis) relevant sind.<sup>227</sup> Im dritten, blau markierten Schritt werden dann der alte Zellzustand  $C_{t-1}$  mit dem Zustand  $h_{t-1}$  und der Eingangsvariable  $x_t$  zum neuen Zustand  $C_t$  aktualisiert.<sup>228</sup> Im letzten, braun markierten Schritt wird dann der neue Zustand  $h_t$  aus dem Zellzustand  $C_t$ , der Eingangsvariable  $x_t$  und

<sup>225</sup> Vgl. Aggarwal, C., Neural Networks, 2018, S. 292 ff.

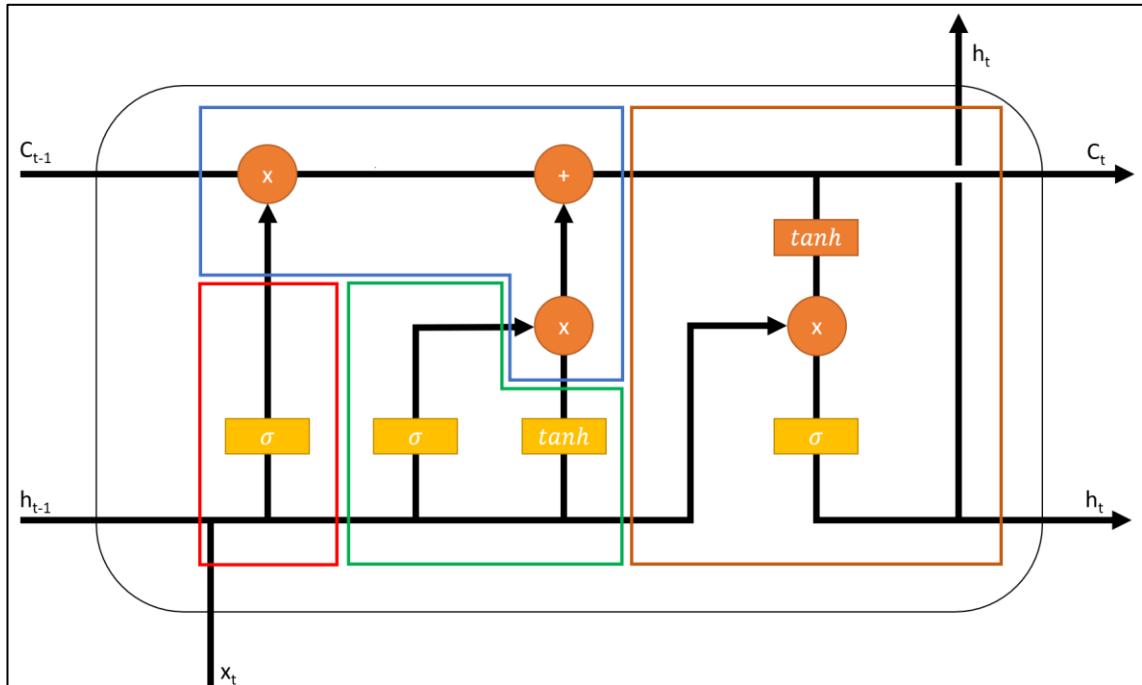
<sup>226</sup> Vgl. Aggarwal, C., Neural Networks, 2018, S. 292 ff.

<sup>227</sup> Vgl. Géron, A., Machine Learning, 2020, S. 518 ff.

<sup>228</sup> Vgl. Lazzeri, F., Machine Learning, 2021, S. 147 ff.

dem vorherigen Zustand  $h_{t-1}$  erzeugt. Hierbei entscheidet das sogenannte „Ausgabe-Gatter“, welche Informationen in den Zustand  $h_t$  einfließen sollen.<sup>229</sup>

Abbildung 27: Verarbeitungsschritte im Neuron in LSTM-Schicht



Quelle: In Anlehnung an Babcock, J., Bali, R., Generative AI, 2021, S. 92.

Der LSTM-Algorithmus hat also durch den Zustand  $h_t$  ein Kurzzeitgedächtnis und kann durch den Zellzustand  $C_t$  Informationen im Langzeitgedächtnis behalten.<sup>230</sup>

## 2.6 Evaluation und Metriken

Aufgrund der Nähe der Zeitserienvorhersage zur Regression werden häufig die gleichen Verfahren zur Evaluation eingesetzt, um die Vorhersagen mit der tatsächlichen Zeitreihe zu vergleichen und die Qualität der Vorhersagen zu bewerten.<sup>231</sup> Im folgenden Kapitel werden gängige Metriken und Verfahren zur Evaluation dargestellt.

### 2.6.1 Rolling-Forecast

Zur Bewertung von Modellen zur Zeitreihenvorhersage kann das sogenannte Rolling-Forecast-Verfahren eingesetzt werden. Dabei wird die vorherzusagende Zeitreihe (ähnlich dem bereits in Kapitel 2.4.5 beschriebenen Verfahren) in Trainings-

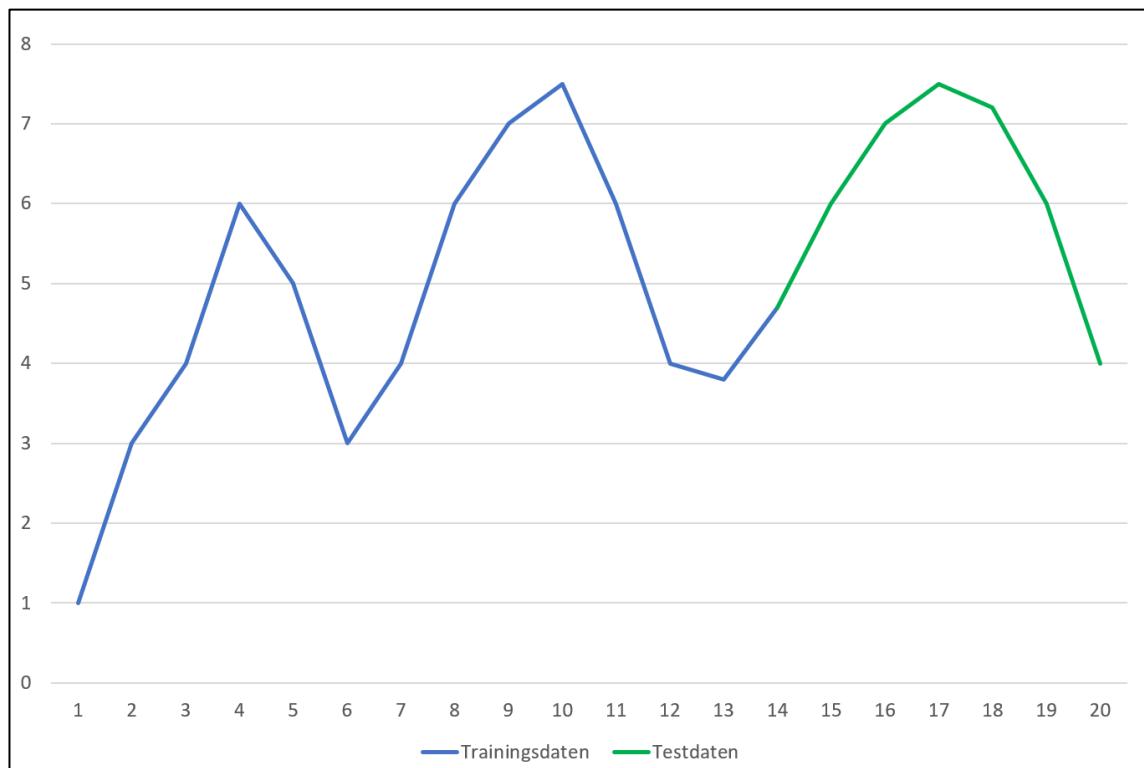
<sup>229</sup> Vgl. Babcock, J., Bali, R., Generative AI, 2021, S. 91 ff.

<sup>230</sup> Vgl. Korstanje, J., Advanced Forecasting, 2021, S. 243 ff. und S. 250.

<sup>231</sup> Vgl. Auffarth, B., Machine Learning for Time Series, 2021, S. 105 ff.

und Testdaten aufgeteilt. Die Reihenfolge der Beobachtungen darf dabei über die Trainings- und Testdaten nicht verändert werden.<sup>232</sup> Die Abbildung 28 zeigt eine beispielhafte Zeitreihe  $y(t)$  mit 20 Beobachtungen, die nach der Beobachtung  $y(14)$  in **Trainingsdaten (blau)** und **Testdaten (grün)** eingeteilt wird.

**Abbildung 28: Zeitreihe für Rolling-Forecast**



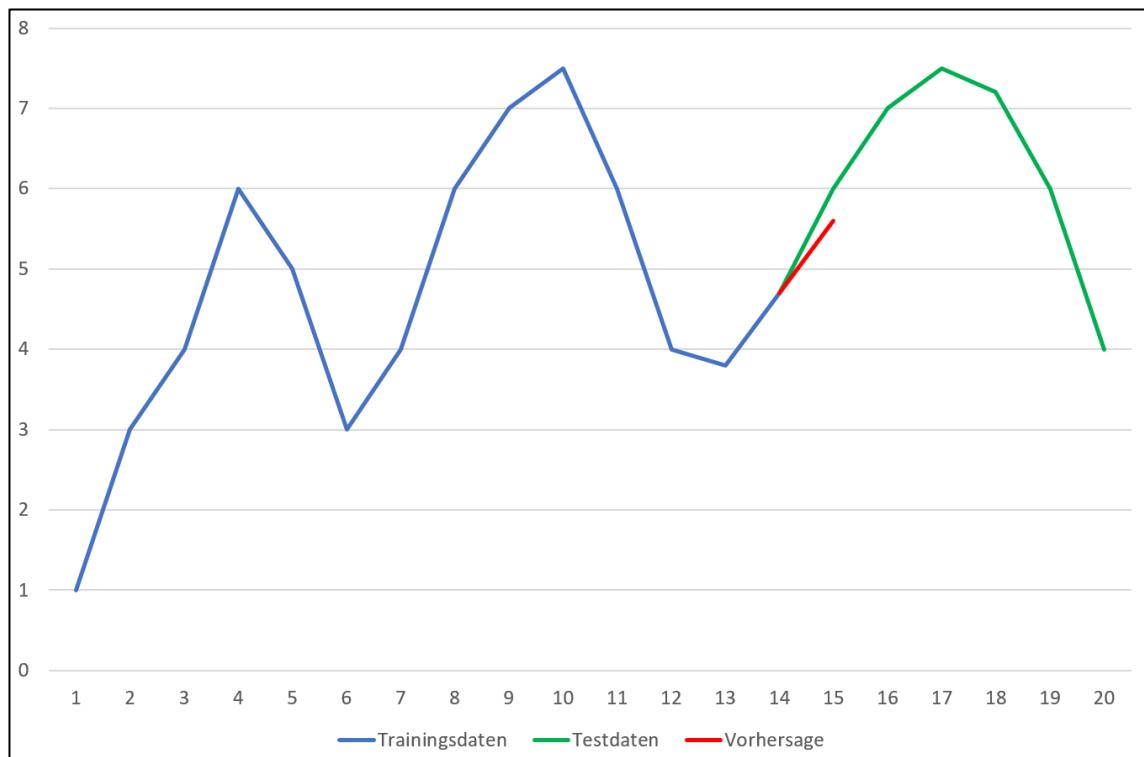
Quelle: Eigene Darstellung

Es kann dann ein Vorhersagemodell mit den **Trainingsdaten bis einschließlich  $y(14)$**  trainiert und eine **Vorhersage  $\hat{y}(15)$  für die nächste Beobachtung** erstellt werden.<sup>233</sup>

**Abbildung 29: Rolling-Forecast (Schritt 1)**

<sup>232</sup> Vgl. Hyndman, R. J., Athanasopoulos, G., Forecasting, S. 31 ff.

<sup>233</sup> Vgl. Hyndman, R. J., Athanasopoulos, G., Forecasting, S. 31 ff.

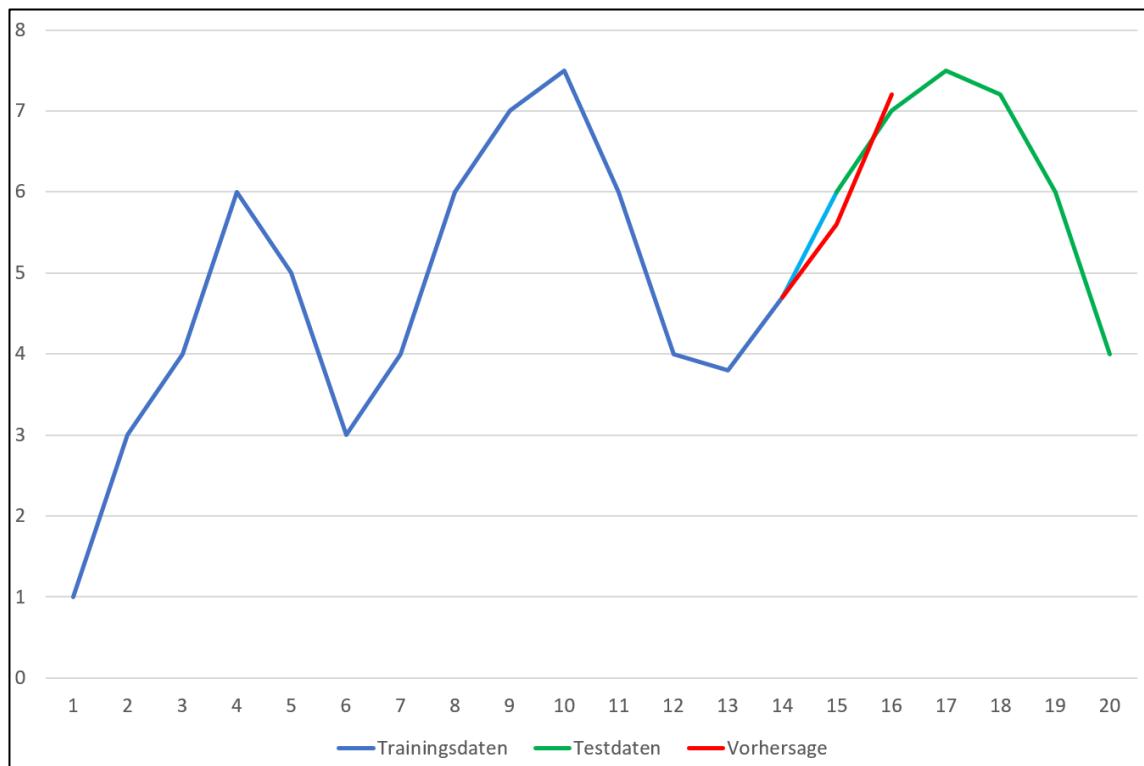


Quelle: Eigene Darstellung

Anschließend wird die erste Beobachtung der Testdaten  $y(t_{15})$  zu den Trainingsdaten hinzugefügt. Das bestehende Modell kann mit den neuen Trainingsdaten aktualisiert oder neu trainiert werden. Dann wird die Vorhersage  $\hat{y}(16)$  für die nächste Beobachtung erstellt.<sup>234</sup>

Abbildung 30: Rolling-Forecast (Schritt 2)

<sup>234</sup> Vgl. Brownlee, J., Rolling Forecast, 2017.

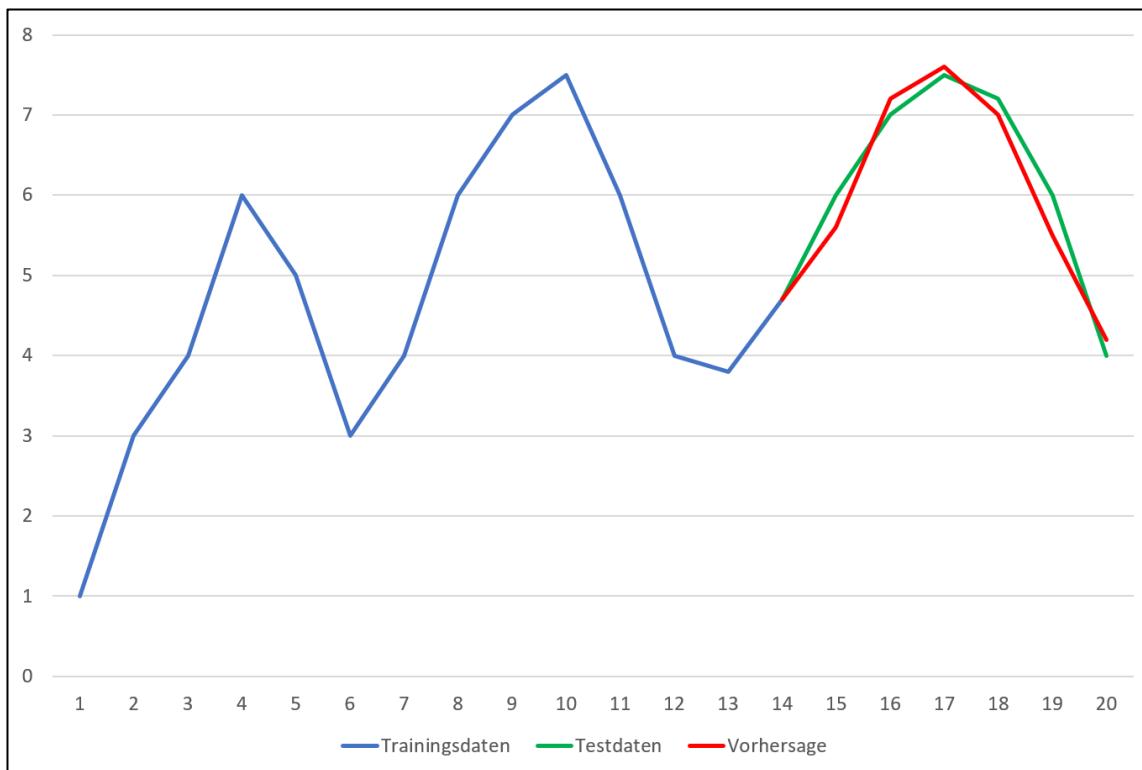


Quelle: Eigene Darstellung

Dieser Prozess wird für alle Beobachtungen in den **Testdaten** so lange wiederholt, bis die gesamten Testdaten abgearbeitet sind. Die Vorhersagen bilden dann wie in Abbildung 31 dargestellt ebenfalls eine **Zeitreihe  $\hat{y}(t)$** . Die **vorhergesagte Zeitreihe  $\hat{y}(t)$**  kann dann mit den **Testdaten von  $y(t)$**  verglichen werden.<sup>235</sup>

**Abbildung 31: Rolling-Forecast**

<sup>235</sup> Vgl. Hyndman, R. J., Athanasopoulos, G., Forecasting, S. 31 ff.



Quelle: Eigene Darstellung

### 2.6.2 Metriken

Für den Vergleich von n Vorhersagen  $\hat{y}(t)$  mit den tatsächlichen Werten einer Zeitreihe  $y(t)$  bieten sich folgende Metriken an:

#### 1. Bestimmtheitsmaß ( $R^2$ )

Das Bestimmtheitsmaß ( $R^2$  oder auch R2-Wert) gibt an, zu welchem Anteil die Varianz einer Variable  $y$  durch eine andere Variable (wie beispielsweise Vorhersagen  $\hat{y}(t)$ ) erklärt werden kann.<sup>236</sup> Es gilt dabei folgende Formel:

**Formel 10: Bestimmtheitsmaß  $R^2$**

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=0}^{n-1} (y_i - \hat{y}_i)^2}{\sum_{i=0}^{n-1} (\hat{y}_i - y_i)^2} \quad (1)$$

Quelle: In Anlehnung an *scikit-learn developers*, Metriken, 2021.

<sup>236</sup> Vgl. Patel, A., Vishwas, B. V., Time Series Analysis Python, 2020, S. 70 f.

Das Bestimmtheitsmaß nimmt stets einen Wert zwischen null und eins an. Je näher der Wert an eins ist, desto höher ist der Anteil der erklärbaren Varianz bzw. desto näher sind die Vorhersagen an den tatsächlichen Werten.<sup>237</sup>

## 2. Mittlerer absoluter Fehler

Der mittlere absolute Fehler (englisch „Mean Absolute Error“ bzw. MAE) misst die durchschnittliche absolute Differenz der Vorhersagen  $\hat{y}$  von den tatsächlichen Werten  $y$ . Es wird dabei mit den mathematischen Beträgen der Werte gerechnet, da sich ansonsten negative und positive Fehler ausgleichen würden.<sup>238</sup> Es gilt folgende Formel:

**Formel 11: Mittlerer absoluter Fehler MAE**

$$\text{MAE}(y, \hat{y}) = \frac{1}{n} * \sum_{i=0}^{n-1} |y_i - \hat{y}_i| \quad (1)$$

Quelle: In Anlehnung an *scikit-learn developers*, Metriken, 2021.

Der mittlere absolute Fehler ist in der Einheit des entsprechenden Merkmals skaliert und nimmt Werte von null bis Unendlich an, wobei null keinen Abweichungen entspräche. Je höher der MAE, desto weiter sind die Vorhersagen vom tatsächlichen Wert entfernt.<sup>239</sup>

## 3. Mittlerer quadrierter Fehler und Wurzel

Der mittlere quadrierte Fehler (englisch „Mean Squared Error“ bzw. MSE) misst die durchschnittliche quadrierte Differenz der Vorhersagen  $\hat{y}$  von den tatsächlichen Werten  $y$ .<sup>240</sup> Es gilt folgende Formel:

**Formel 12: Mittlerer quadrierter Fehler MSE**

$$\text{MSE}(y, \hat{y}) = \frac{1}{n} * \sum_{i=0}^{n-1} (y_i - \hat{y}_i)^2 \quad (1)$$

Quelle: In Anlehnung an *scikit-learn developers*, Metriken, 2021.

Der mittlere quadrierte Fehler nimmt Werte von null bis Unendlich an. Wie auch beim MAE gilt beim MSE, dass null keinen Abweichungen entspräche. Anders als beim MAE fallen größere Abweichungen beim MSE allerdings stärker ins

<sup>237</sup> Vgl. Korstanje, J., Advanced Forecasting, 2021, S. 28 f.

<sup>238</sup> Vgl. Auffarth, B., Machine Learning for Time Series, 2021, S. 107 ff.

<sup>239</sup> Vgl. Lazzeri, F., Machine Learning, 2021, S. 17 ff.

<sup>240</sup> Vgl. Patel, A., Vishwas, B. V., Time Series Analysis Python, 2020, S. 70 f.

Gewicht und werden daher stärker bestraft. Der MSE ist aufgrund der Quadrierung der Abweichungen allerdings nicht mehr in der Einheit des jeweiligen Merkmals skaliert.<sup>241</sup> Daher wird häufig die Wurzel aus dem MSE gebildet (englisch „Root Mean Squared Error“ bzw. RMSE), die dann wieder in der Einheit des Merkmals skaliert und daher besser interpretierbar ist.<sup>242</sup> Es gilt dann folgende Formel:

**Formel 13: Wurzel des mittleren quadrierten Fehlers RMSE**

$$\text{RMSE}(y, \hat{y}) = \sqrt{\text{MSE}} \quad (1)$$

$$= \sqrt{\frac{1}{n} * \sum_{i=0}^{n-1} (y_i - \hat{y}_i)^2} \quad (2)$$

Quelle: In Anlehnung an Korstanje, J., Advanced Forecasting, 2021.

**4. Mittlerer absoluter prozentualer Fehler**

Der mittlere absolute prozentuale Fehler (englisch „Mean Absolute Percentage Error“ bzw. MAPE) misst die durchschnittliche prozentuale Abweichung der Voraussagen  $\hat{y}$  von den tatsächlichen Werten  $y$ .<sup>243</sup> Es gilt folgende Formel:

**Formel 14: Mittlerer absoluter prozentualer Fehler MAPE**

$$\text{MAPE}(y, \hat{y}) = \frac{1}{n} * \sum_{i=0}^{n-1} \frac{|y_i - \hat{y}_i|}{\max(\epsilon, |y_i|)} \quad (1)$$

Quelle: In Anlehnung an scikit-learn developers, Metriken, 2021.

Falls der tatsächliche Wert  $y_i = 0$  ist, dann wird in der im Folgenden verwendeten Implementierung stattdessen der Term  $\epsilon$  als kleinstmögliche positive Zahl verwendet, da sich der MAPE ansonsten aufgrund einer Division durch null nicht berechnen ließe.<sup>244</sup> Der MAPE ist im Bereich von null bis Unendlich skaliert und kann daher als prozentualer Wert interpretiert werden, wobei ein Wert von eins genau 100 % entspricht. Je höher der MAPE, desto höher sind die prozentualen Abweichungen.<sup>245</sup>

<sup>241</sup> Vgl. Aggarwal, C., Neural Networks, 2018, S. 175 ff.

<sup>242</sup> Vgl. Korstanje, J., Advanced Forecasting, 2021, S. 26 f.

<sup>243</sup> Vgl. Auffarth, B., Machine Learning for Time Series, 2021, S. 111.

<sup>244</sup> Vgl. scikit-learn developers, Metriken, 2021.

<sup>245</sup> Vgl. Lazzeri, F., Machine Learning, 2021, S. 46 ff.

### 3 Erstellung des Vorhersagmodells

In diesem Kapitel wird die Erstellung des Vorhersagmodells beschrieben. Dafür werden die verfügbaren Daten gesammelt und analysiert, dann wird jeweils ein ARIMA-Modell und ein neuronales Netz trainiert und optimiert. Die Modelle werden verglichen und das bessere der beiden Modelle wird in eine graphische Benutzeroberfläche eingebunden.

#### 3.1 Vorgehensmodell und Werkzeuge

Für die Erstellung des Vorhersagmodells muss zunächst ein geeignetes Vorgehensmodell ausgewählt werden. Die Entwicklung soll möglichst flexibel gestaltet werden und lässt sich in zwei wesentliche Teile gliedern, die sich im Vorgehensmodell wiederfinden müssen: Die Analyse der verfügbaren Daten und die Erstellung des Vorhersagmodells. Weiterhin muss ein geeignetes Werkzeug für praktische Umsetzung ermittelt werden.

##### 3.1.1 Vorgehensmodell

Für die Bearbeitung von Anwendungsfällen aus dem Bereich des maschinellen Lernens bieten sich

- Cross Industry Standard Process for Data Mining (CRISP-DM)
- Knowledge Discovery in Databases (KDD)
- Sample, Explore, Modify, Model and Assess (SEMMA)

als Vorgehensmodell an.<sup>246</sup>

Beim KDD-Modell werden die zu untersuchenden Daten ausgewählt (Selection), dann um beispielsweise Fehler und fehlende Daten bereinigt (Preprocessing) und in ein für die Analyse geeignetes Format übertragen (Transformation). Anschließend werden die Daten untersucht und verarbeitet (Data Mining) und ausgewertet (Interpretation).<sup>247</sup> Rücksprünge sind dabei in jede Phase möglich<sup>248</sup>, allerdings lassen sich die Analyse und die Modellierung nicht klar in eine eigene Phase trennen.

Bei SEMMA werden zunächst repräsentative, leicht zu verarbeitende Stichproben aus den Daten gezogen (Sample), welche dann analysiert werden (Explore). Danach

---

<sup>246</sup> Vgl. Ghavami, P., Analytics Methods, 2019, S. 49 ff.

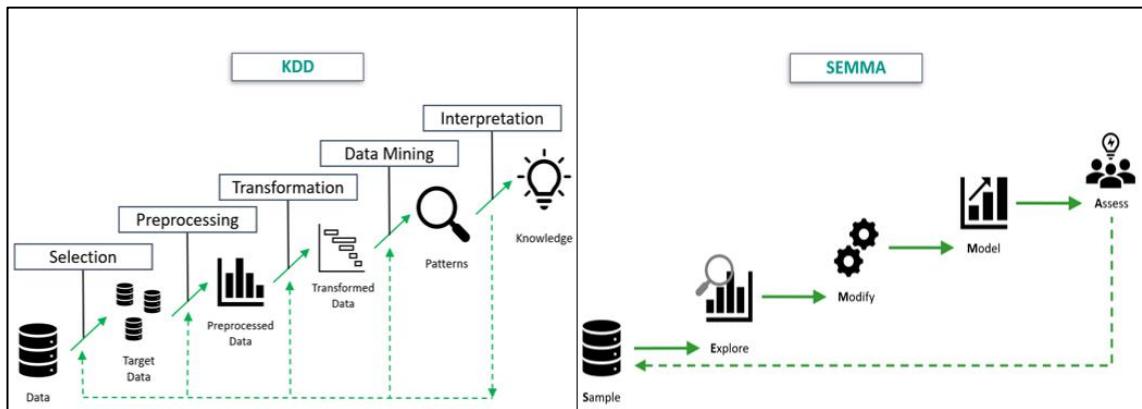
<sup>247</sup> Vgl. Hawamdeh, S., Chang H.-C., Knowledge Management, 2018, S. 32 ff.

<sup>248</sup> Vgl. Khosrow-Pour, M., Advanced Methodologies, 2018, S. 518 ff.

## Erstellung des Vorhersagemodells

werden die Daten in ein geeignetes Format übertragen (Modify), womit dann ein Modell erstellt (Model) und bewertet wird (Asses).<sup>249</sup> Hier ließen sich zwar die Analyse und die Modellerstellung in eigene Phasen trennen, allerdings sind Rücksprünge dabei nur an den Anfang des Prozesses möglich.<sup>250</sup>

Abbildung 32: KDD und SEMMA



Quelle: In Anlehnung an Ghavami, P., Analytics Methods, 2019, S. 53.;

In Anlehnung an Hawamdeh, S., Chang H.-C., Knowledge Management, 2018, S. 39.

Das „Cross Industry Standard Process for Data Mining“-Modell (CRISP-DM) ist ein weit verbreitetes Vorgehensmodell. Es gliedert sich in sechs Phasen:<sup>251</sup>

1. **Business Understanding:** Die erste Phase des Modells widmet sich dem Verständnis des Anwendungsfalls aus fachlicher Sicht. Weiterhin werden Ziele, Anforderungen und Erfolgskriterien für die Entwicklung festgelegt.<sup>252</sup>
2. **Data Understanding:** In der zweiten Phase werden die verfügbaren Daten gesammelt und analysiert. Ziel ist es, ein Verständnis über die Beschaffenheit und Aussagekraft der Daten zu erreichen. Ein wichtiger Aspekt ist dabei auch die Bewertung der Datenqualität.<sup>253</sup>
3. **Data Preparation:** In der dritten Phase werden die Daten für das Modeling vorbereitet. Fehlende und falsche Daten werden entfernt oder korrigiert und die Daten werden in ein einheitliches, für Algorithmen verarbeitbares Format übertragen.<sup>254</sup>

<sup>249</sup> Vgl. Ghavami, P., Analytics Methods, 2019, S. 49 ff.

<sup>250</sup> Vgl. Hawamdeh, S., Chang H.-C., Knowledge Management, 2018, S. 39 ff.

<sup>251</sup> Vgl. Luber, S., Litzel, N., CRISP-DM, 2019.

<sup>252</sup> Vgl. Ghavami, P., Analytics Methods, 2019, S. 54 ff.

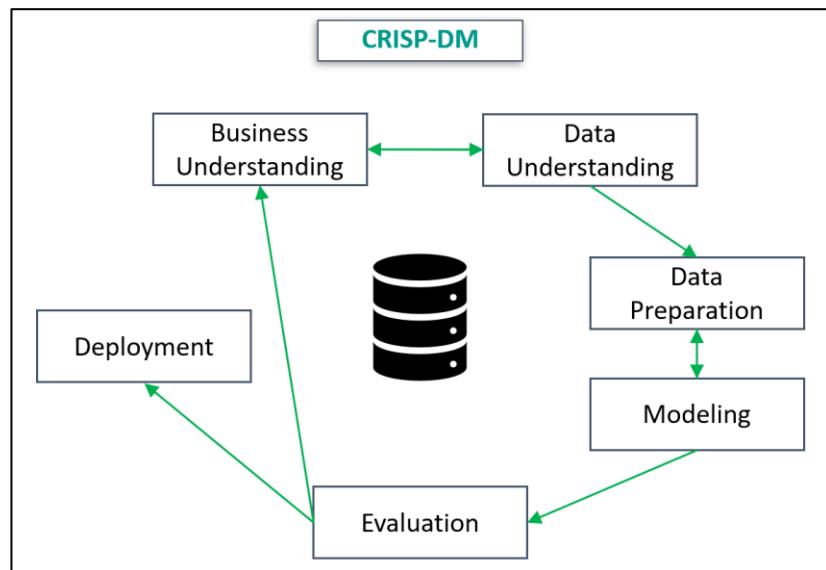
<sup>253</sup> Vgl. Nisbet, R., Miner, G., Yale, K., Statistische Analysen, 2017, S. 40 ff.

<sup>254</sup> Vgl. Luber, S., Litzel, N., CRISP-DM, 2019.

## Erstellung des Vorhersagemodells

4. **Modeling:** In der vierten Phase werden Algorithmen auf die Daten angewandt, um Vorhersagemodelle zu erstellen. Die Modelle werden untersucht und optimiert, wobei die in der zweiten Phase bei der Analyse gesammelten Informationen wesentlich sind.<sup>255</sup>
5. **Evaluation:** Das erstellte Vorhersagemodell wird getestet und anhand der in der ersten Phase aufgestellten Kriterien bewertet. Falls die Qualität des Modells nicht ausreichend ist, beginnt der Prozess von vorne.<sup>256</sup>
6. **Deployment:** Falls das Modell die Kriterien erfüllt und einsatzbereit ist, wird es in eine produktive Umgebung überführt.<sup>257</sup>

Abbildung 33: CRISP-DM



Quelle: In Anlehnung an IBM, CRISP-DM Übersicht, 2021.

Die Abbildung 33 zeigt das CRISP-DM-Modell. Dabei sind besonders die möglichen Rücksprünge zwischen den einzelnen Phasen zu beachten (beispielsweise vom Modeling zur Data Preparation oder von der Evaluation zum Business Understanding). Dadurch bietet CRISP-DM ein hohes Maß an Flexibilität.<sup>258</sup> Außerdem lassen sich die Analyse und die Erstellung des Vorgehensmodells durch die Phasen Data Understanding und Modeling sauber trennen. Da CRISP-DM viel Flexibilität bietet und

<sup>255</sup> Vgl. Ghavami, P., Analytics Methods, 2019, 54 ff.

<sup>256</sup> Vgl. Nisbet, R., Miner, G., Yale, K., Statistische Analysen, 2017, S. 40 ff.

<sup>257</sup> Vgl. Luber, S., Litzel, N., CRISP-DM, 2019.

<sup>258</sup> Vgl. Ghavami, P., Analytics Methods, 2019, S. 54 ff.

auch die Trennung von Analyse und Modellierung ermöglicht, wird es als am besten geeignetes Vorgehensmodell ausgewählt.

### 3.1.2 Python

Python ist eine objektorientierte Programmiersprache, die Anfang der 1990er Jahre van Guido van Rossum entwickelt wurde. Da die Sprache ursprünglich als Übungssprache für Schüler und Studenten zur Einführung in die Programmierung gedacht war, zeichnet sich Python durch eine besonders einfache Syntax aus.<sup>259</sup> Python hat sich vorrangig im wissenschaftlich-mathematischen Bereich<sup>260</sup> durchgesetzt<sup>261</sup>, da es zahlreiche Bibliotheken und Packages für statistische und mathematische Zwecke<sup>262</sup> sowie den Bereich der künstlichen Intelligenz gibt.<sup>263</sup>

Dementsprechend bietet sich Python auch für die Analyse und Vorhersage von Zeitreihen an. Neben weiteren Packages bietet Python mit den Packages pandas für die Datenverarbeitung, statsmodels für die Zeitreihenanalyse und -vorhersage, scikit-learn für maschinelles Lernen, TensorFlow für neuronale Netze und NumPy und SciPy für wissenschaftlich-mathematische und statistische Funktionen sowie Matplotlib und seaborn für Visualisierungen einen breiten Umfang an Funktionalitäten für die Datenanalyse und Zeitreihenvorhersage.<sup>264</sup> Eine genaue Übersicht über die verwendete Software und die entsprechenden Packages ist im Anhang zu finden (Anhang 2: Verwendete Software).

## 3.2 Business Understanding<sup>265</sup>

Ziele und Hintergrund der Untersuchung wurden bereits in der Einleitung beziehungsweise dem Kapitel 2.1 detailliert dargestellt. Es soll ein Vorhersagemodell erstellt werden, welches den Stromverbrauch des Bundeslandes Baden-Württemberg (Regelzone der TransnetBW) für den jeweils nächsten Tag anhand historischer Stromverbrauchs- und Wetterdaten vorhersagen soll. Die Vorhersage soll dann zum Beispiel für die Planung von Regelenergie-Kapazitäten nutzbar sein, um die Effizienz und die Stabilität des Stromnetzes zu verbessern.

---

<sup>259</sup> Vgl. *Python Software Foundation*, Python, 2022.

<sup>260</sup> Vgl. *Python Software Foundation*, Python, 2022.

<sup>261</sup> Vgl. TIOBE Software BV, TIOBE Index August 2021, 2021.

<sup>262</sup> Vgl. *Python Software Foundation*, Python Applications, 2022.

<sup>263</sup> Vgl. Hirschle, J., Machine Learning für Zeitreihen, 2020, S. 9.

<sup>264</sup> Vgl. Lazzeri, F., Machine Learning, 2021, S. 21 ff. und S. 62 ff.

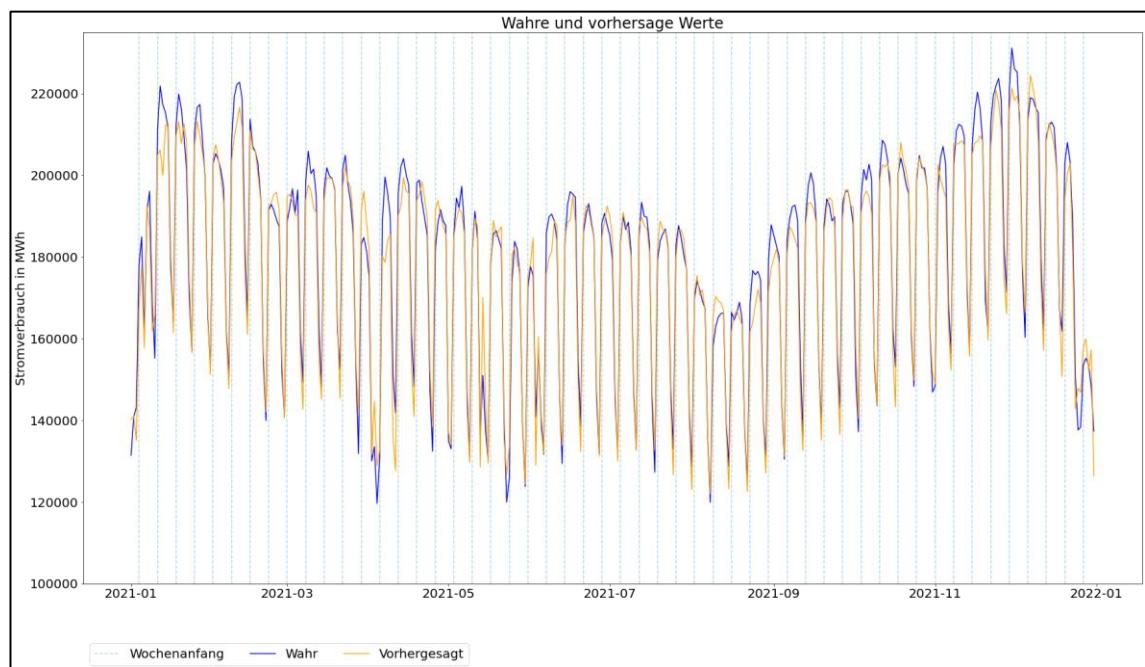
<sup>265</sup> [1-Business Understanding](#)

## Erstellung des Vorhersagemodells

Wie bereits beschrieben erstellen die Stromanbieter für die Abnehmer in ihrem Bilanzkreis eine Prognose, welche dann beim Stromnetzbetreiber gemeldet wird. Die Summe dieser einzelnen Prognosen wird von der Bundesnetzagentur auf dem Portal für Strommarktdaten SMARD veröffentlicht. Ziel ist es, ein Modell zu erstellen, welches den Stromverbrauch besser prognostizieren kann als die Summe der einzelnen Prognosen der Stromanbieter. Deren Prognose dient also als sogenanntes Baseline-Modell<sup>266</sup>, welches vom neu entwickelten Modell übertroffen werden soll. Als Metrik wird der mittlere absolute prozentuale Fehler (MAPE) verwendet. Das zu entwickelnde Modell soll die Baseline um 10% übertreffen.

Die Daten vom 01.01.2015 bis zum 31.12.2020 werden als Trainingsdaten zur Modellerstellung genutzt, die Daten vom 01.01.2021 bis zum 31.12.2021 werden als Testdaten zur Bewertung genutzt. Die Aufbereitung der Daten erfolgt nach demselben Vorgehen wie im Kapitel 3.3 für die weiteren Strommarktdaten und wird daher hier nicht weiter erläutert. Die Abbildung 34 zeigt die Baseline (orange) im Vergleich zu den tatsächlichen Werten (blau) im Zeitraum der Testdaten. Die Baseline hat einen MAPE von etwa 2,44%.

**Abbildung 34: Baseline**



<sup>266</sup> Vgl. Auffarth, B., Machine Learning for Time Series, 2021, S. 106 f.

Quelle: Eigene Darstellung<sup>267</sup>

Es wird daher als Erfolgskriterium festgelegt, einen MAPE von 2,44% um mindestens 10% zu unterbieten. Wird ein MAPE von 2,2% erreicht ( $2,44\% * 0,9$ ), gilt die Erstellung des Modells als erfolgreich. Eine Abweichung von ~2,2% oder weniger wird daher im Folgenden als „akzeptierte Abweichung“ bezeichnet. Größere Abweichungen müssen über die Testdaten im Mittel ausgeglichen werden.

### 3.3 Data Understanding<sup>268</sup>

Der historische Stromverbrauch des Landes Baden-Württemberg wird vom Portal für Strommarktdaten SMARD<sup>269</sup> von der Bundesnetzagentur im CSV-Format bezogen und in tägliche Auflösung umgewandelt, außerdem werden Kalenderdaten ergänzt. Die genaue Beschreibung für Beschaffung und Aufbereitung der Daten ist im Notebook<sup>270</sup> zu finden. Es werden die Daten vom 01.01.2015 bis zum 31.12.2021 einschließlich verwendet. Die CSV-Datei fasst folgende Merkmale:

**Tabelle 5: Merkmale in CSV-Datei mit Stromverbrauch**

Merkmaile	Beschreibung
<b>datum</b>	Datum im Format yyyy-mm-dd (Index)
<b>verbrauch</b>	Stromverbrauch in MWh
<b>monat</b>	Monat in Textform
<b>wochentag</b>	Wochentag in Textform
<b>feiertag</b>	Feiertag in Textform (NaN entspricht keinem Feiertag)
<b>arbeitstag</b>	1 -> entspricht Arbeitstag (Montag-Freitag, kein Feiertag) 0 -> entspricht arbeitsfreiem Tag (Samstag, Sonntag oder Feiertag)

Quelle: Eigene Darstellung

Die Wetterdaten werden von der Internetseite WeatherAPI.com<sup>271</sup> im JSON-Format bezogen und in CSV-Dateien umgewandelt. Die Beschreibung für Beschaffung und Aufbereitung beziehungsweise Umwandlung der Daten ist im Notebook<sup>272</sup> zu finden.

<sup>267</sup> [1-Business Understanding/01-Baseline \(Baseline-Prognose bei Testdaten\)](#)

<sup>268</sup> [2-Data Understanding](#)

<sup>269</sup> Bundesnetzagentur (<https://www.smard.de/>)

<sup>270</sup> [2-Data Understanding/Datenbeschaffung/01-Verbrauch](#)

<sup>271</sup> WeatherAPI.com (<https://www.weatherapi.com/>)

<sup>272</sup> [2-Data Understanding/Datenbeschaffung/02-Wetter](#)

Um eine möglichst repräsentative und umfangreiche Abdeckung des Landes Baden-Württemberg zu erreichen, werden folgende vier Städte ausgewählt, für die Wetterdaten erfasst werden:

**Tabelle 6: Städte für Wetterdaten**

Stadt	Einwohnerzahl	Lage
<b>Stuttgart</b>	630.000 (größte Stadt)	Zentrum
<b>Mannheim</b>	310.000 (zweitgrößte Stadt)	Nordwesten
<b>Freiburg</b>	230.000 (viertgrößte Stadt)	Südwesten
<b>Ulm</b>	126.000 (sechstgrößte Stadt)	Osten

Quelle: Eigene Darstellung<sup>273</sup>

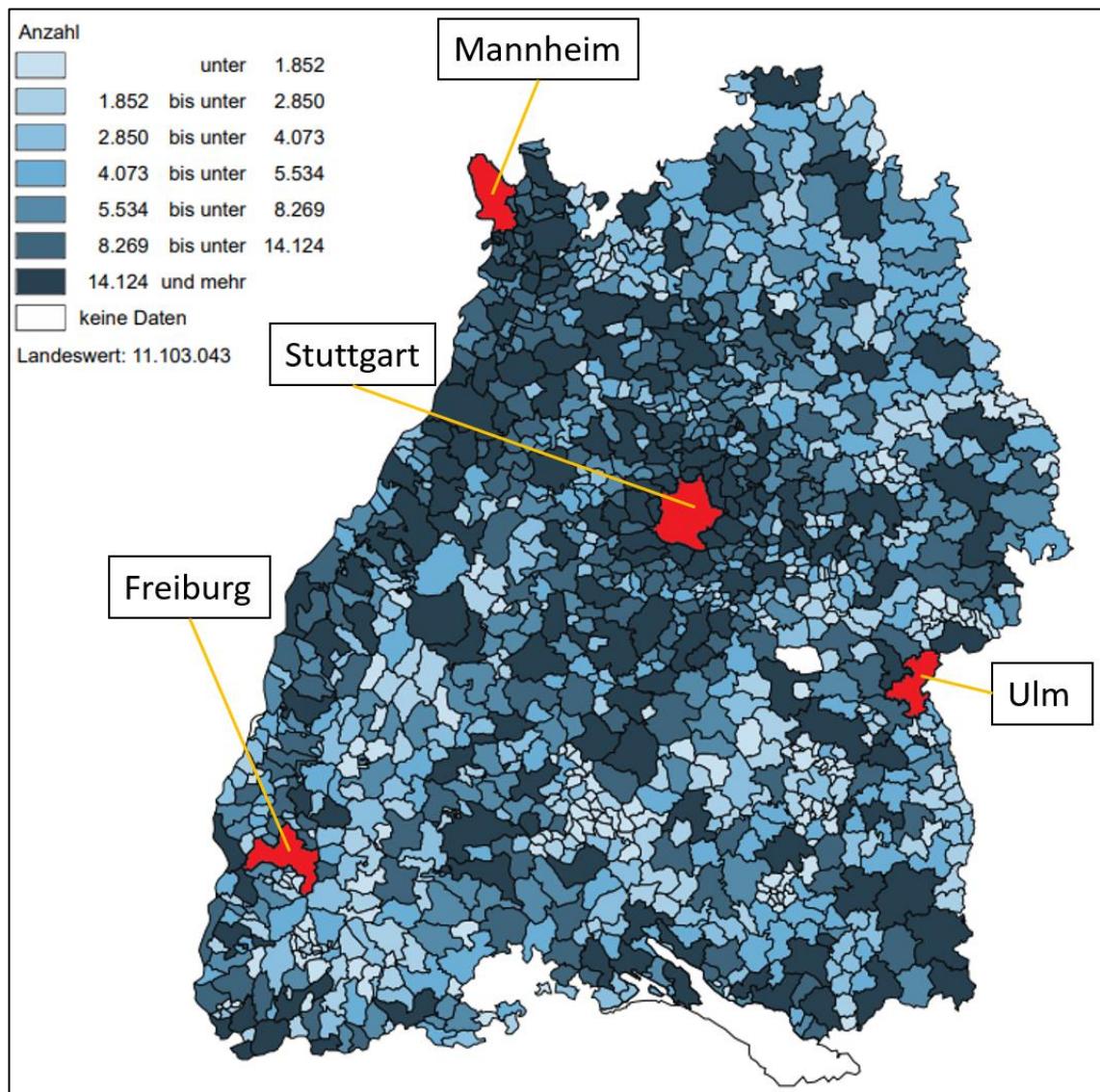
Die drittgrößte Stadt Karlsruhe und die fünftgrößte Stadt Heidelberg werden aufgrund der räumlichen Nähe zu Mannheim beziehungsweise Stuttgart nicht miterfasst. Die Abbildung 35 zeigt die für die weitere Analyse verwendeten Städte.

**Abbildung 35: Verwendete Städte für Wetterdaten**

---

<sup>273</sup> Vgl. *Statista*, Einwohnerzahlen, 2021.

## Erstellung des Vorhersagemodells



Quelle: Eigene Darstellung<sup>274</sup>

Es werden ebenfalls die Daten vom 01.01.2015 bis zum 31.12.2021 einschließlich verwendet. Die Daten werden jeweils für eine Stadt in einer eigenen CSV-Datei erfasst. Die CSV-Dateien fassen folgende Merkmale:

**Tabelle 7: Merkmale der CSV-Datei für die Wetterdaten**

Merkmal	Beschreibung
<b>datum</b>	Datum im Format yyyy-mm-dd (Index)
<b>hoechsttemperatur</b>	Tageshöchsttemperatur in Grad Celsius

<sup>274</sup> Basierend auf *Statistisches Landesamt Baden-Württemberg, Karten, 2022.*

durchschnitts-temperatur	Tagesdurchschnittstemperatur in Grad Celsius
tiefsttempera-tur	Tagestiefsttemperatur in Grad Celsius
wetter	Textbeschreibung des Wetters
sichtweite	Durchschnittliche Sichtweite in Kilometern
feuchtigkeit	Luftfeuchtigkeit in %
niederschlag	Niederschlag in Millimetern pro Quadratmeter
windgeschwin-digkeit	Durchschnittliche Windgeschwindigkeit in Stundenkilome-tern
sonnenauf-gang	Zeitpunkt des Sonnenaufgangs in hh:mm AM
sonnenunter-gang	Zeitpunkt des Sonnenuntergangs in hh:mm AM

Quelle: Eigene Darstellung<sup>275</sup>

Die entsprechenden Notebooks sind im Anhang zu finden.<sup>276</sup>

### 3.3.1 Analyse des Stromverbrauchs<sup>277</sup>

Die Abbildung 36 zeigt den täglichen Stromverbrauch vom 01.01.2015 bis zum 31.12.2021. Der gleitende Durchschnitt über 28 Tage ist in rot eingezeichnet. Es ist ein deutliches, sich jährlich wiederholendes Muster zu erkennen. Der Stromverbrauch ist im Januar mit durchschnittlich circa 200.000 MWh besonders hoch und fällt dann bis etwa Mai um 20 % auf durchschnittlich 160.000 MWh ab, wobei vor allem im April gelegentlich große Unregelmäßigkeiten erkennbar sind. Im Juni und Juli steigt der Verbrauch wieder auf durchschnittlich 170.000 MWh und fällt bis September auf durchschnittlich 160.000 MWh zurück. Ab September steigt der Verbrauch dann auf durchschnittlich 180.000 bis 190.000 MWh an. Auffällig ist außerdem ein starker Abfall des Stromverbrauchs auf etwa 140.000 MWh Ende Dezember. Es lässt sich hier also eine deutliche jährliche Saisonalität erkennen.

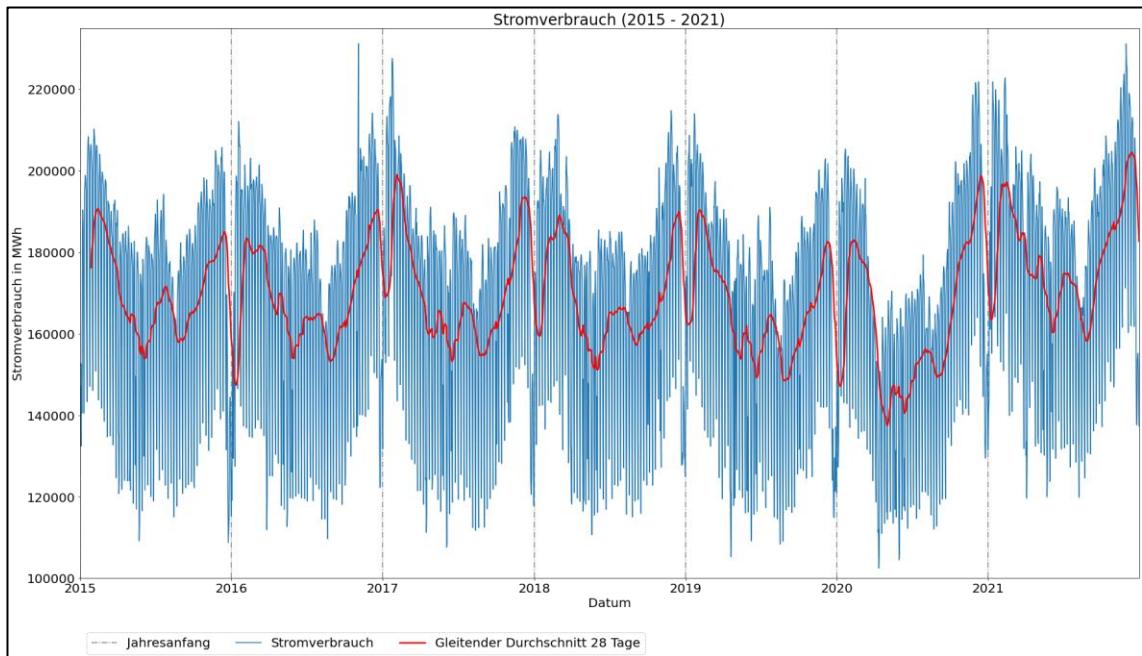
Abbildung 36: Stromverbrauch (2015-2021)

<sup>275</sup> Vgl. WeatherAPI.com, API-Dokumentation, 2022.

<sup>276</sup> [2-Data Understanding/Datenbeschaffung](#)

<sup>277</sup> [2-Data Understanding/01-Stromverbrauch](#)

## Erstellung des Vorhersagemodells



Quelle: Eigene Darstellung<sup>278</sup>

Die Abbildung 37 zeigt die gleitenden 14-Tage Durchschnitte jedes Jahres übereinander gelegt. Darin ist die oben beschriebene Saisonalität deutlich zu erkennen. Auffällig ist außerdem, dass der Stromverbrauch im Jahr 2020 (rot) von März bis August deutlich unter den Werten der anderen Jahre liegt. Dies kann vor allem durch den Rückgang der Industrieproduktion<sup>279</sup> im Zuge des Lockdowns aufgrund der Corona-Pandemie zurückgeführt werden.<sup>280</sup> Die Einschränkungen für die Industrie wurden im zweiten Halbjahr größtenteils zurückgenommen. Aufgrund der anhaltenden Pandemie ist der Stromverbrauch im Telekommunikationsbereich sowie für private Haushalte<sup>281</sup> weiterhin signifikant gestiegen.<sup>282</sup> So ist zu erkennen, dass der Stromverbrauch seit Ende 2020 und über das ganze Jahr 2021 (blau) im Vergleich zu den Vorjahren etwas höher ist. Bis auf einige wenige Ausschläge (vor allem im zweiten Quartal 2021) folgt der Stromverbrauch seit Mitte 2020 aber wieder der oben beschriebenen Saisonalität.

**Abbildung 37: Stromverbrauch (gleitender 14-Tage Durchschnitt 2015-2021)**

<sup>278</sup> [2-Data Understanding/01-Stromverbrauch \(Stromverbrauch nach Jahren\)](#)

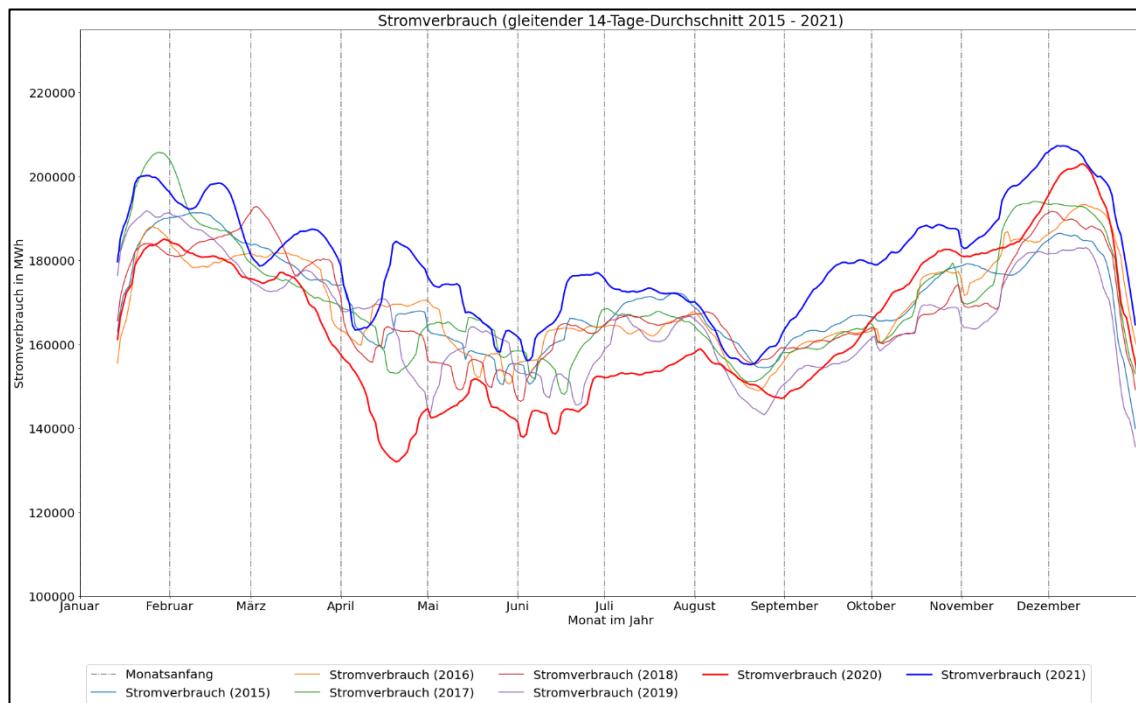
<sup>279</sup> Vgl. *Institut für Weltwirtschaft (IfW Kiel)*, Stromverbrauch im Lockdown, 2020.

<sup>280</sup> Vgl. Eder, S., Strom- und Energieverbrauch im Lockdown, 2020.

<sup>281</sup> Vgl. *Mitteldeutscher Rundfunk*, Stromverbrauch privater Haushalte, 2021.

<sup>282</sup> Vgl. *Bundesverband der Energie- und Wasserwirtschaft*, Energieversorgungsbericht, 2021.

## Erstellung des Vorhersagemodells



Quelle: Eigene Darstellung<sup>283</sup>

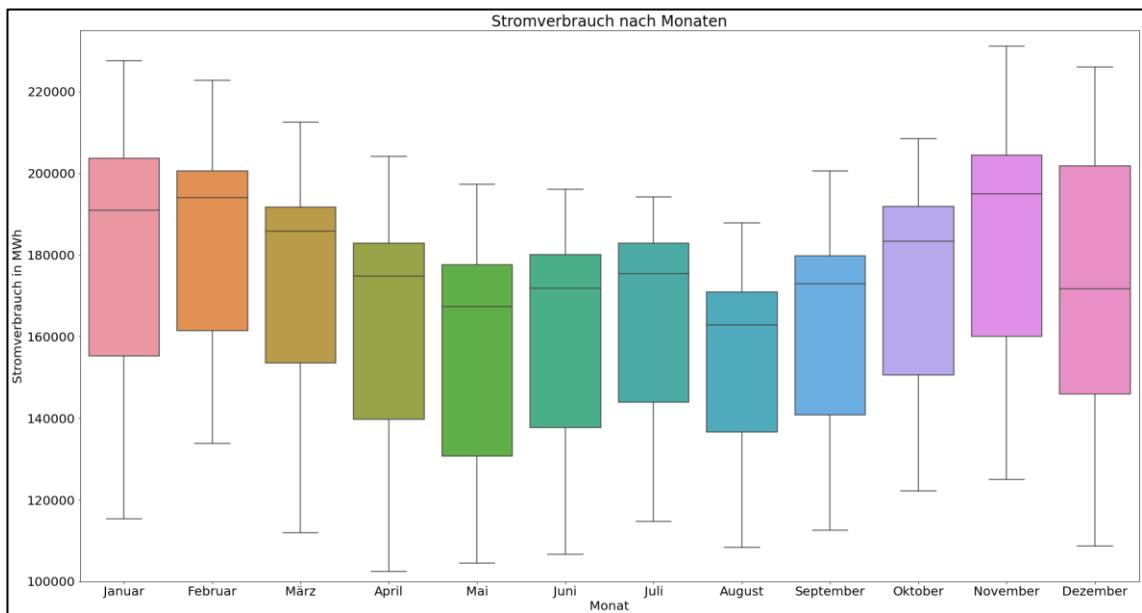
Die Abbildung 38 zeigt Boxplots des Stromverbrauchs nach Monaten über die Jahre 2015 bis 2021 hinweg. Auch hier ist das bereits beschriebene jährliche Verhalten des Stromverbrauchs deutlich erkennbar. Es ist auffällig, dass der Stromverbrauch im Dezember relativ niedrig ist. Wie beispielsweise in Abbildung 37 (und Abbildung 39) erkennbar ist, folgt der Stromverbrauch trotzdem dem oben beschriebenen Muster, allerdings fällt der Verbrauch in der Zeit um die Weihnachtsfeiertage ab etwa dem 20. Dezember sehr stark ab. In diesem Zeitraum steigt der Verbrauch privater Haushalte an, gleichzeitig fällt der Verbrauch bei Industrie und Gewerbe aufgrund der Feiertage und der damit verbundenen Urlaubssaison stark ab.<sup>284</sup> Dieser Effekt reicht bis in die erste Januarwoche des Folgejahres hinein und führt zu niedrigeren Durchschnittswerten im Dezember und Januar.

**Abbildung 38: Boxplots des Stromverbrauchs nach Monaten**

<sup>283</sup> [2-Data Understanding/01-Stromverbrauch \(gleitende Durchschnitte\)](#)

<sup>284</sup> Vgl. Bundesverband der Energie- und Wasserwirtschaft, Stromverbrauch im Dezember, 2020.

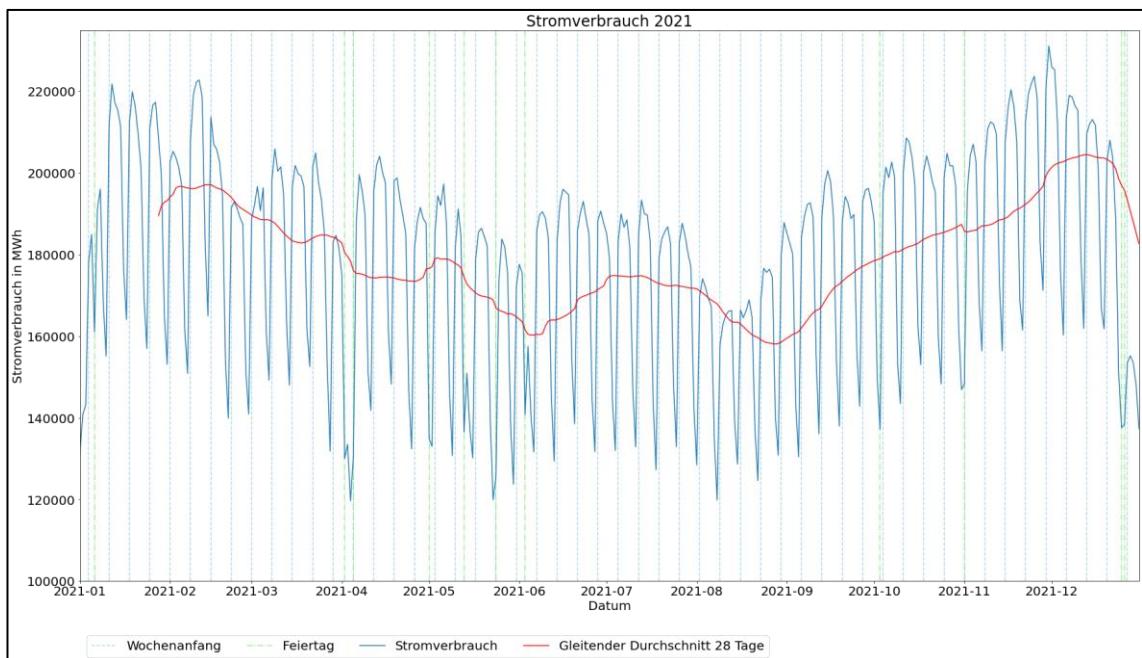
## Erstellung des Vorhersagemodells



Quelle: Eigene Darstellung<sup>285</sup>

Die Abbildung 39 zeigt den Stromverbrauch des Jahres 2021. Zunächst ist erkennbar, dass der Stromverbrauch neben der jährlichen auch einer wöchentlichen Saisonalität folgt. Außerdem erkennt man, dass der Stromverbrauch an den durch grüne Linien gekennzeichneten Feiertagen besonders niedrig ist.

**Abbildung 39: Stromverbrauch 2021**



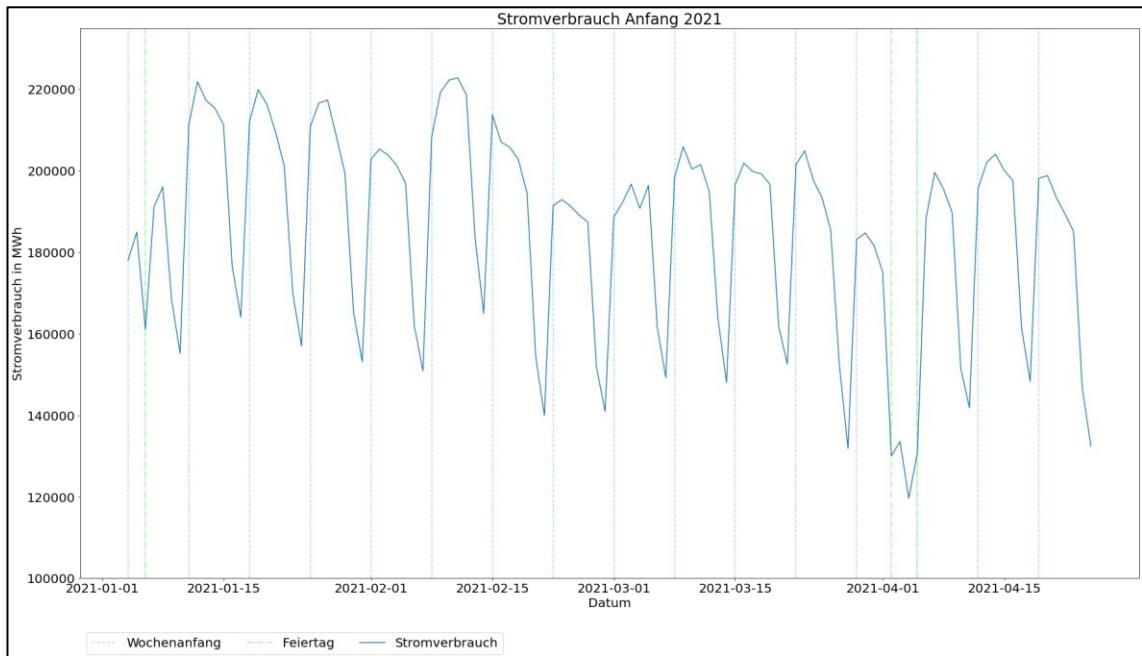
<sup>285</sup> [2-Data Understanding/01-Stromverbrauch \(Stromverbrauch nach Monaten\)](#)

## Erstellung des Vorhersagemodells

Quelle: Eigene Darstellung<sup>286</sup>

Die Abbildung 40 zeigt dieses Verhalten noch deutlicher.

**Abbildung 40: Stromverbrauch Anfang 2021**



Quelle: Eigene Darstellung<sup>287</sup>

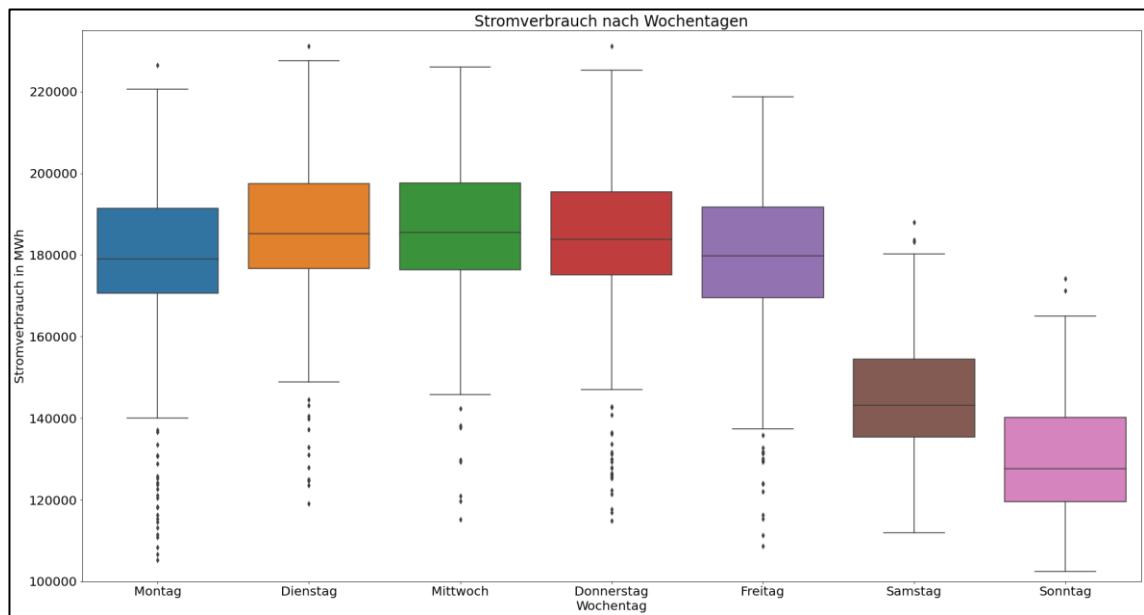
Die Boxplots des Stromverbrauchs nach Wochentagen in Abbildung 41 zeigen ebenfalls die wöchentliche Saisonalität, bei der der Stromverbrauch von Montag bis Freitag durchschnittlich wesentlich höher ist als an Samstagen und Sonntagen. Es ist auch erkennbar, dass der Verbrauch montags und freitags im Median etwas geringer als dienstags, mittwochs und donnerstags ist. Dabei handelt es sich aber um sehr geringe Unterschiede. Ausreißer gibt es überwiegend nach unten.

**Abbildung 41: Stromverbrauch nach Wochentagen**

<sup>286</sup> [2-Data Understanding/01-Stromverbrauch \(Stromverbrauch je Jahr\)](#)

<sup>287</sup> [2-Data Understanding/01-Stromverbrauch \(Stromverbrauch 2021\)](#)

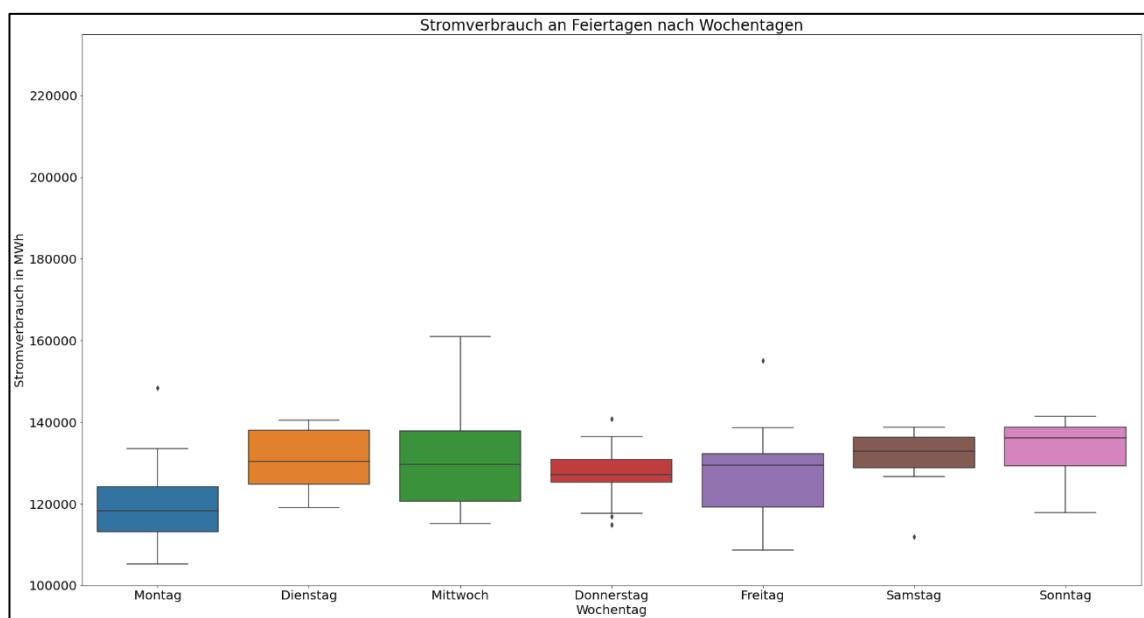
## Erstellung des Vorhersagemodells



Quelle: Eigene Darstellung<sup>288</sup>

An Feiertagen ist der Stromverbrauch ähnlich niedrig wie an den Wochenenden, wie auch in Abbildung 39 erkennbar ist. Die Abbildung 42 zeigt Boxplots des Stromverbrauchs an Feiertagen jeweils nach Wochentagen. Hier ist zu erkennen, dass der Stromverbrauch unabhängig vom Wochentag an Feiertagen etwa gleichhoch bei 130.000 bis 140.000 MWh liegt.

**Abbildung 42: Stromverbrauch an Feiertagen nach Wochentagen**



<sup>288</sup> [2-Data Understanding/01-Stromverbrauch \(Stromverbrauch nach Wochentagen\)](#)

Quelle: Eigene Darstellung<sup>289</sup>

Es lässt sich also ein starker Unterschied zwischen den Tagen Montag bis Freitag zu Samstagen, Sonntagen und Feiertagen erkennen. Daher werden die Beobachtungen für die weitere Analyse in jeweils zwei Kategorien eingeteilt und entsprechend markiert:

1. Arbeitstage (Montag bis Freitag und kein Feiertag)
2. Arbeitsfreie Tage (Samstag, Sonntag oder Feiertag)

Neben der Datenanalyse wird auch ein einfaches Regressionsmodell entwickelt, um zu überprüfen, wie gut sich der Verlauf des Stromverbrauchs mit dem Einfluss der verfügbaren Merkmale abbilden lässt. Dafür wird eine Regressionsfunktion sechsten Grades mit den verfügbaren Merkmalen und den Daten von 2015 bis 2018 trainiert und mit den Daten von 2019 getestet. Da das Level des Stromverbrauchs ab 2020 wie bereits oben beschrieben gestiegen ist, werden die Daten vor 2020 für diese Analyse verwendet. Die Ergebnisse der Vorhersage sind in der Abbildung 43 zu sehen. Das Modell weicht im Schnitt etwa 7,8% ab und kann im Wesentlichen nur ungefähr die wöchentliche Saisonalität sowie Feiertage abbilden. Die Regressionsanalyse mit den weiteren Merkmalen kann dem Notebook entnommen werden.<sup>290</sup>

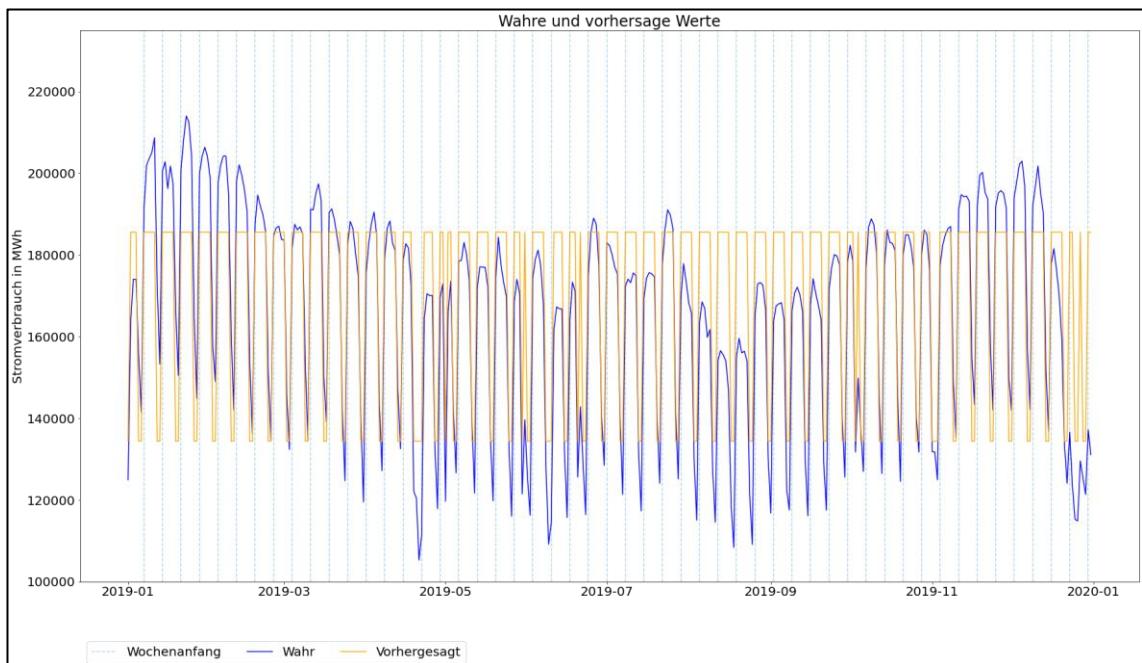
**Abbildung 43: Regression mit Arbeitstag**

---

<sup>289</sup> [2-Data Understanding/01-Stromverbrauch \(Stromverbrauch nach Feiertagen und Wochentagen\)](#)

<sup>290</sup> [2-Data Understanding/01-Stromverbrauch \(Regressionsanalyse\)](#)

## Erstellung des Vorhersagemodells



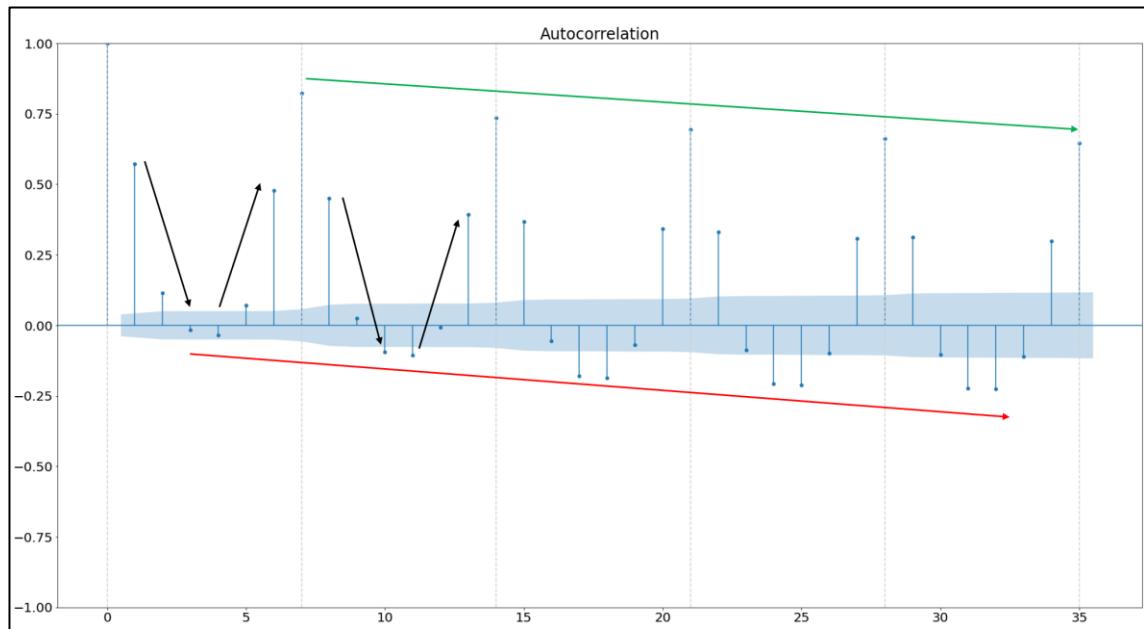
Quelle: Eigene Darstellung<sup>291</sup>

Die Abbildung 44 zeigt das einfache Autokorrelationsdiagramm des Stromverbrauchs mit den ersten 35 Lags. Hier spiegelt sich die wöchentliche Saisonalität wider, so ist beispielsweise jedes siebte Lag (bei grauen gestrichelten Linien) besonders stark positiv einfach-autokorriert (grün markiert), auch die daran grenzenden sechsten und achten Lags sind vergleichsweise mittelstark korriert. Die Stärke dieser Korrelationen nimmt mit der Zeit ab. Umgekehrt sind die Lags in der Mitte der wöchentlichen Saison (jedes zweite bis fünfte Lag) schwach negativ einfach-autokorriert. Diese negative Korrelation nimmt mit der Zeit zu (rot markiert). Daraus lässt sich schließen, dass vor allem das korrespondierende Lag der letzten saisonalen Periode und das unmittelbar vorherige Lag einen besonders starken Einfluss auf die aktuelle Beobachtung zu haben scheinen.

**Abbildung 44: Einfache Autokorrelation des Stromverbrauchs**

<sup>291</sup> [2-Data Understanding/01-Stromverbrauch \(Regression mit Arbeitstag\)](#)

## Erstellung des Vorhersagemodells



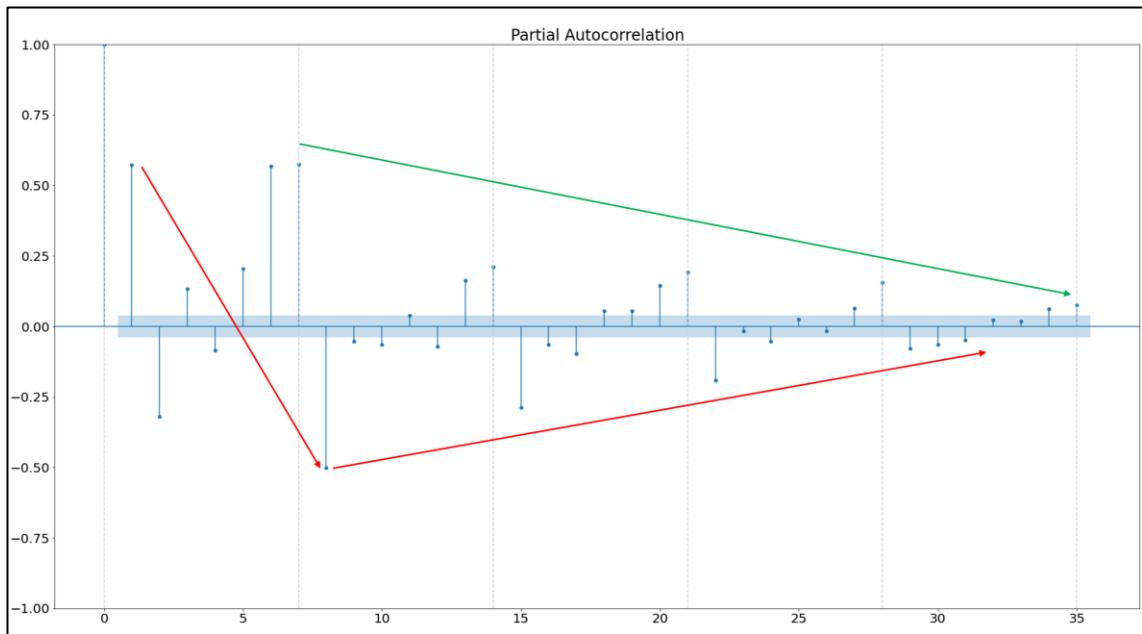
Quelle: Eigene Darstellung<sup>292</sup>

Die Abbildung 45 zeigt das partielle Autokorrelationsdiagramm des Stromverbrauchs mit ebenfalls 35 Lags. Auch hier zeigt sich die wöchentliche Saisonalität (grau markiert). Es ist auffällig, dass das erste Lag einer Saison zunächst mittelstark positiv, ab der zweiten Saison aber mittelstark negativ und dann immer schwächer korreliert (rot markiert). Das jeweils sechste und siebte Lag korreliert in der ersten Saison mittelstark positiv, diese Korrelation nimmt allerdings schon in der zweiten Periode sehr stark ab (grün markiert). Es sind also das zweite, sechste, siebte und achte Lag jeweils mittelstark partiell-autokorreliert.

**Abbildung 45: Partielle Korrelation des Stromverbrauchs**

<sup>292</sup> [2-Data Understanding/01-Stromverbrauch \(Einfache Autokorrelation\)](#)

## Erstellung des Vorhersagemodells



Quelle: Eigene Darstellung<sup>293</sup>

Sowohl der ADF-Test wie auch der KPSS-Test ergeben, dass es sich um stationäre Daten handelt. Eine Stationarisierung ist daher nicht weiter notwendig.

**Abbildung 46: ADF- und KPSS-Test**

Ergebnisse des ADF-Test:		Ergebnisse des KPSS-Test:	
Test-Statistik	-5.211151	Teststatistik	0.325023
p-Wert	0.000008	p-Wert	0.100000
#Lags verwendet	27.000000	Verwendete Lags	13.000000
Anzahl verwendeter Beobachtungen	2529.000000	Critical Value (10%)	0.347000
Critical Value (1%)	-3.432938	Critical Value (5%)	0.463000
Critical Value (5%)	-2.862684	Critical Value (2.5%)	0.574000
Critical Value (10%)	-2.567379	Critical Value (1%)	0.739000
dtype: float64		dtype: float64	
Nullhypothese wird verworfen, Daten sind stationär		Nullhypothese wird beibehalten, Daten sind stationär	

Quelle: Eigene Darstellung<sup>294</sup>

### 3.3.2 Analyse des Einflusses durch die Temperatur<sup>295</sup>

Es stehen drei unterschiedliche Temperaturwerte je Stadt zur Verfügung: Die jeweilige Tagesstiefst-, Tageshöchst-, und Tagesdurchschnittstemperatur. Da sowohl bei der Tagesstiefst-<sup>296</sup>, als auch bei der Tageshöchst-<sup>297</sup> und der

<sup>293</sup> [2-Data Understanding/01-Stromverbrauch \(Partielle Autokorrelation\)](#)

<sup>294</sup> [2-Data Understanding/01-Stromverbrauch \(Stationarität\)](#)

<sup>295</sup> [2-Data Understanding/02a-Temperatur \(Vergleich\);](#)

[2-Data Understanding/02b-Temperatur](#)

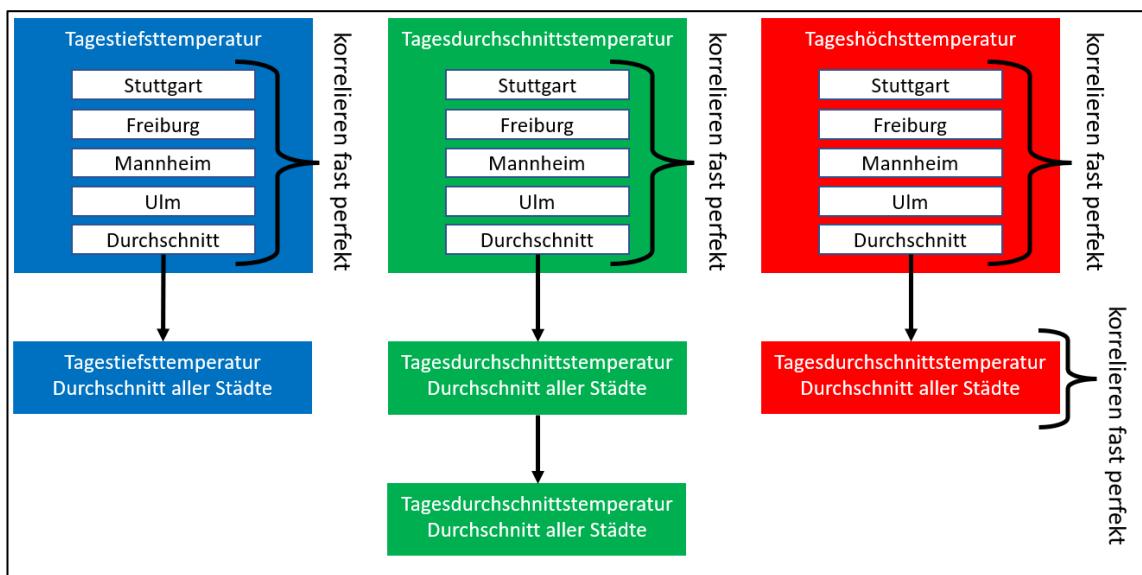
<sup>296</sup> [2-Data Understanding/02a-Temperatur \(Vergleich\) \(Tiefsttemperatur\)](#)

<sup>297</sup> [2-Data Understanding/02a-Temperatur \(Vergleich\) \(Höchsttemperatur\)](#)

## Erstellung des Vorhersagemodells

Tagesdurchschnittstemperatur<sup>298</sup> die Werte der jeweiligen vier Städte samt Durchschnitt fast perfekt miteinander korrelieren, kann zunächst auf die einzelnen Werte je Stadt verzichtet werden und es kann sich nur auf den Durchschnitt des jeweiligen Temperaturwertes beschränkt werden. Da aber auch die über alle Städte durchschnittliche Tagesstief-, Tageshöchst-, und Tagesdurchschnittstemperatur fast perfekt miteinander korrelieren, kann sich im weiteren Verlauf auf den Durchschnitt der Tagesdurchschnittstemperatur aller Städte (im Folgenden zur Vereinfachung nur „Temperatur“) am jeweiligen Tag beschränkt werden.<sup>299</sup> Die Abbildung 47 veranschaulicht dies, genauereres ist dem Notebook zu entnehmen.<sup>300</sup>

Abbildung 47: Korrelationen der Temperaturwerte



Quelle: Eigene Darstellung

Die Abbildung 48 zeigt den Stromverbrauch (rot) und die Temperatur (blau) in einem Liniendiagramm. Die Temperatur folgt einer klaren jährlichen Saisonalität mit einer Amplitude von etwa -10°C bis +33°C, sie erreicht jeweils Mitte des Jahres ihre Höchstwerte und fällt dann zum Ende des Jahres wieder. Diese Saisonalität läuft dem Stromverbrauch entgegengesetzt. Es ist zu erkennen, dass der Stromverbrauch mit steigender Temperatur zu fallen scheint. Die Merkmale korrelieren leicht negativ mit Korrelationskoeffizienten um etwa -0,35.

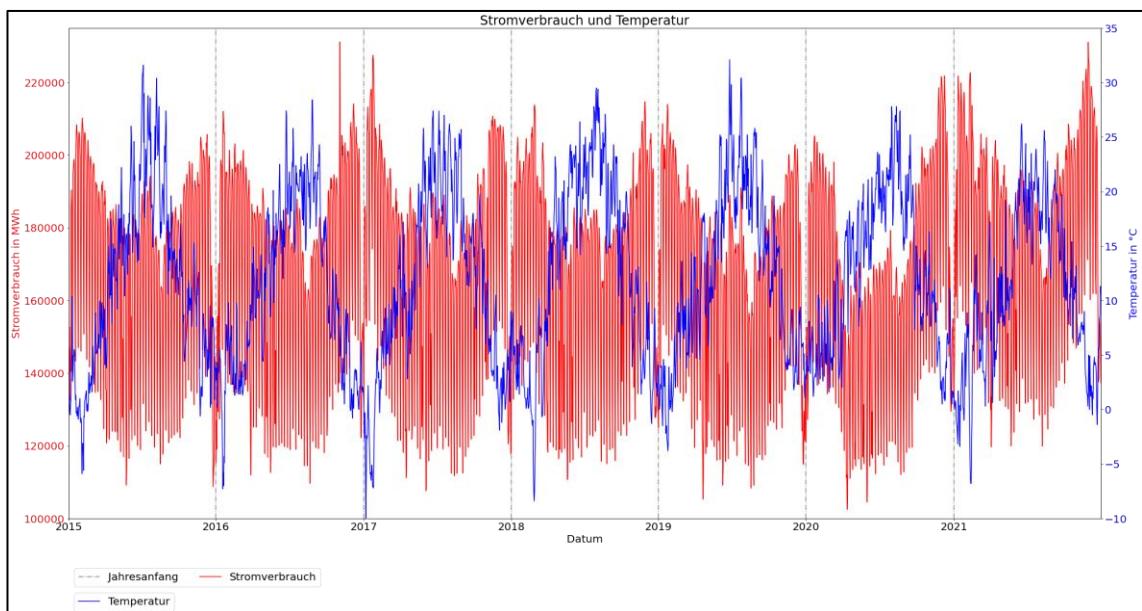
<sup>298</sup> [2-Data Understanding/02a-Temperatur \(Vergleich\) \(Durchschnittstemperatur\)](#)

<sup>299</sup> [2-Data Understanding/02a-Temperatur \(Vergleich\) \(Vergleich\)](#)

<sup>300</sup> [2-Data Understanding/02a-Temperatur \(Vergleich\)](#)

## Erstellung des Vorhersagemodells

**Abbildung 48: Stromverbrauch und Temperatur**



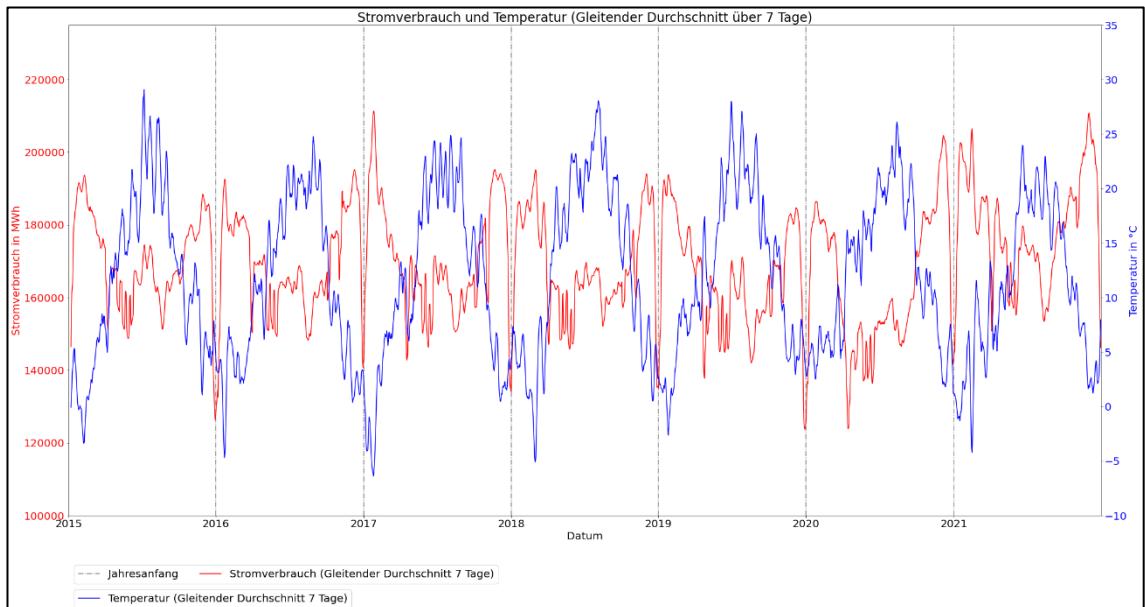
Quelle: Eigene Darstellung<sup>301</sup>

Zur Vereinfachung zeigt die Abbildung 49 nur die gleitenden Durchschnitte über jeweils sieben Tage. Hier werden die entgegengesetzten saisonalen Verläufe etwas deutlicher. Der Stromverbrauch fällt zunächst mit steigender Temperatur. Es ist auch erkennbar, dass sich dieser Effekt ab einer Temperatur von etwa 20°C bis 24°C wieder umkehrt und der Stromverbrauch ab diesem Punkt mit der Temperatur ansteigt. Die gleitenden Durchschnitte korrelieren mit Korrelationskoeffizienten von circa -0,55 schon etwas stärker.

**Abbildung 49: Stromverbrauch und Temperatur (gleitender Durchschnitt)**

<sup>301</sup> [2-Data Understanding/02b-Temperatur \(Verlauf\)](#)

## Erstellung des Vorhersagemodells



Quelle: Eigene Darstellung<sup>302</sup>

Dieser Zusammenhang von Temperatur und Stromverbrauch zeigte sich in ähnlicher Form auch in anderen Untersuchungen. Der konkrete Zusammenhang ist von verschiedenen Faktoren abhängig und variiert daher in unterschiedlichen Umgebungen. Generell ist es so, dass bei niedrigeren Temperaturen mehr Energie (beziehungsweise Strom) für beispielsweise das Heizen von Gebäuden benötigt wird. Mit steigender Temperatur nimmt dieser Bedarf ab, dem entgegengesetzt steigt aber der Bedarf an Strom für Kühlensysteme. Ab einem gewissen Kipppunkt gleicht die Einsparung beim Heizen den Mehrbedarf beim Kühlen nicht mehr aus und der Stromverbrauch steigt wieder.<sup>303</sup>

Dieser Effekt lässt sich auch in Abbildung 50 erkennen. Hier ist der Stromverbrauch nach gruppierter Temperatur jeweils für Arbeits- und arbeitsfreie Tage in Boxplots abgetragen. Es zeigt sich ein fallender Stromverbrauch bis zu einer Temperatur von etwa 20°C, ab etwa 24°C steigt der Stromverbrauch dann wieder leicht an.

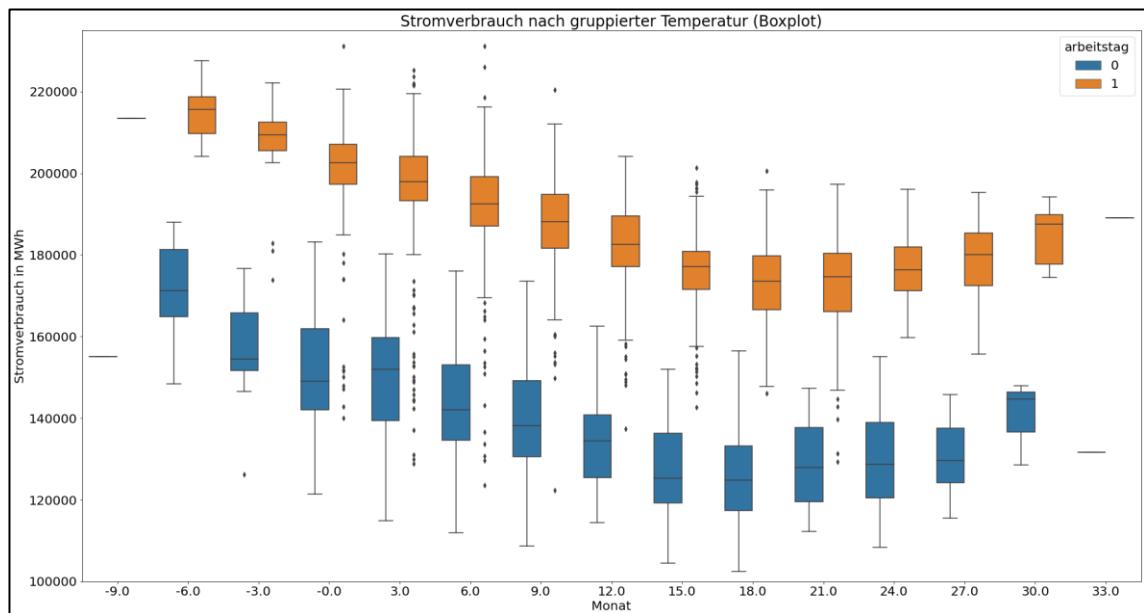
**Abbildung 50: Stromverbrauch nach Temperatur (Boxplot)**

<sup>302</sup> [2-Data Understanding/02b-Temperatur \(Verlauf der Durchschnitte\)](#)

<sup>303</sup> Vgl. Yao, J., Electricity and Temperature, 2021;

Zhang, C., Liao, H., Mi, Z., Klimaeinfluss auf Stromverbrauch, 2019;  
Nischkauer, H., Temperaturabhängigkeit des Stromverbrauchs, 2005

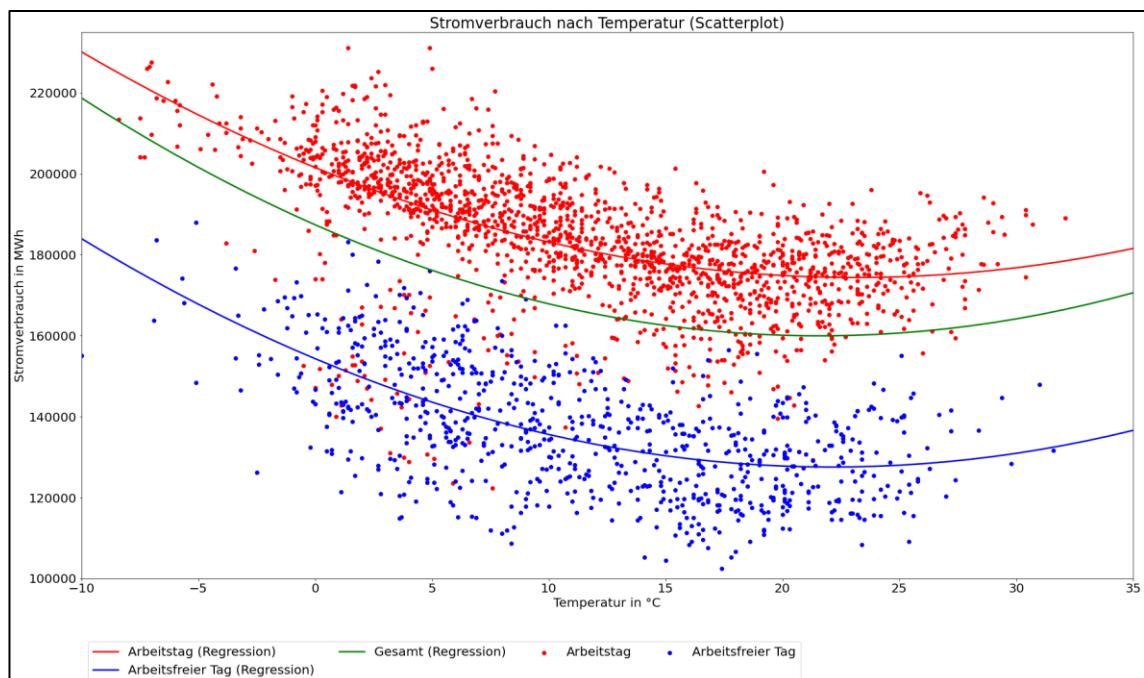
## Erstellung des Vorhersagemodells



Quelle: Eigene Darstellung<sup>304</sup>

Die Abbildung 51 zeigt den Stromverbrauch nach der Temperatur jeweils für Arbeits- und arbeitsfreie Tage in einem Scatterplot mit jeweils einer Regression auf die Punkte. Hier lässt sich der oben beschriebene Zusammenhang ebenfalls erkennen.

**Abbildung 51: Stromverbrauch nach Temperatur (Scatterplot)**



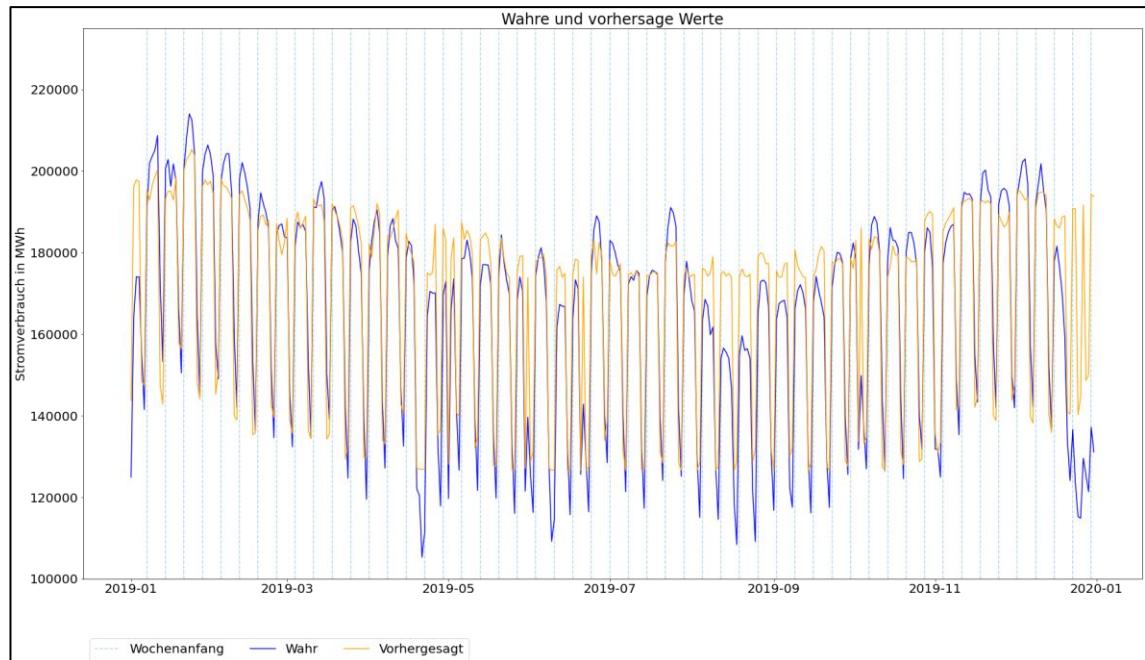
<sup>304</sup> [2-Data Understanding/02b-Temperatur \(Stromverbrauch nach Temperatur\(Boxplot\)\)](#)

## Erstellung des Vorhersagemodells

Quelle: Eigene Darstellung<sup>305</sup>

Das bereits beschriebene Regressionsmodell (siehe Abbildung 43) wird zusätzlich mit der Temperatur trainiert. Dadurch lässt sich die Abweichung auf etwa 5,4 % reduzieren, da sich jetzt die jährliche Saisonalität besser abbilden lässt.

**Abbildung 52: Regression mit Arbeitstag und Temperatur**



Quelle: Eigene Darstellung<sup>306</sup>

Es zeigt sich also ein kausaler Zusammenhang zwischen der Temperatur und dem Stromverbrauch, der sich insgesamt in einer leichten negativen Korrelation äußert, die allerdings nicht-linear ist und nur bis zum Kipppunkt bei etwa 20 bis 24 °C gilt.

### 3.3.3 Analyse des Einflusses durch die Tagesstunden<sup>307</sup>

Aufgrund der geographischen Nähe der vier Städte unterscheiden sich Sonnenauf- und Sonnenuntergang jeweils nur um wenige Minuten mit marginalen Differenzen. Es werden daher nur die Daten von Stuttgart repräsentativ für das gesamte Bundesland verwendet. Eine genauere Beschreibung ist im Notebook zu finden.<sup>308</sup> Aus dem Sonnenauf- und Sonnenuntergang wird für jeden Tag die Anzahl an Tagesstunden

<sup>305</sup> [2-Data Understanding/02b-Temperatur Stromverbrauch nach Temperatur \(Scatterplot\)](#)

<sup>306</sup> [2-Data Understanding/02b-Temperatur \(Regression mit Arbeitstag und Temperatur\)](#)

<sup>307</sup> [2-Data Understanding/03a-Tagesstunden \(Vergleich\)](#);

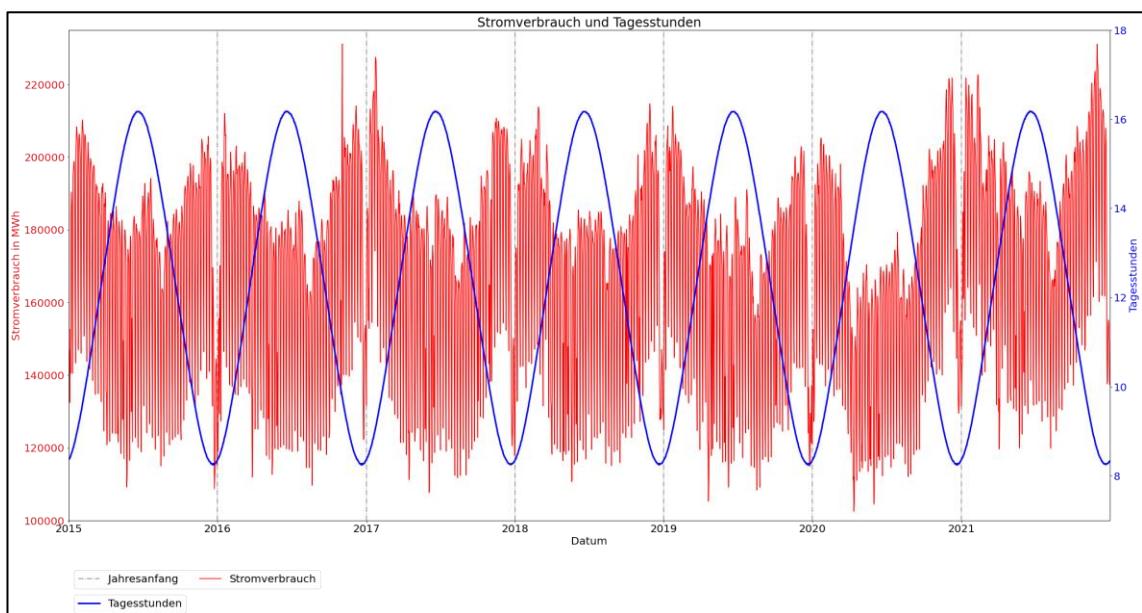
[2-Data Understanding/03b-Tagesstunden](#)

<sup>308</sup> [2-Data Understanding/03a-Tagesstunden \(Vergleich\)](#)

## Erstellung des Vorhersagemodells

berechnet. Die Abbildung 53 zeigt den Stromverbrauch (rot) und die Tagesstunden (blau) als Liniendiagramm. Die Tagesstunden folgen aufgrund der astronomischen Bedingungen einer fast perfekten jährlichen Saisonalität, welche Mitte des Jahres ihr Maximum und jeweils Ende des Jahres ihr Minimum erreicht. Die Tagesstunden bewegen sich mit einer Amplitude von etwa acht Tagesstunden im Winter bis zu etwa 16 Tagesstunden im Sommer. Der Verlauf der Tagesstunden ist ähnlich wie bei der Temperatur dem Stromverbrauch entgegengesetzt, auch die leicht negative Korrelation von circa -0,3 ist sehr ähnlich.

**Abbildung 53: Stromverbrauch und Tagesstunden**



Quelle: Eigene Darstellung<sup>309</sup>

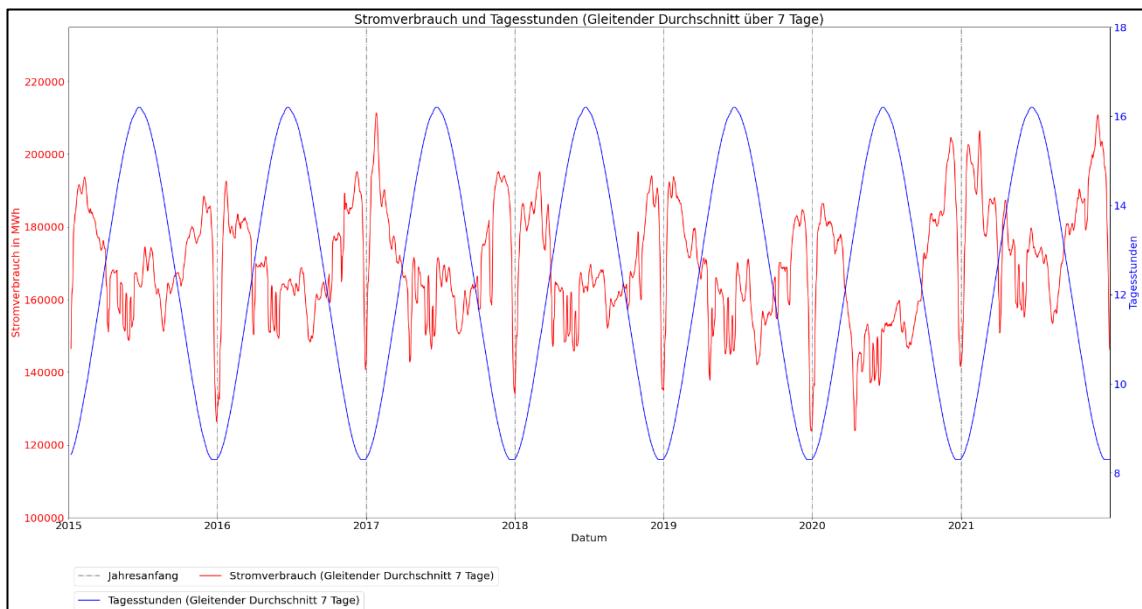
Diese Bewegung wird bei der Betrachtung des gleitenden Durchschnitts über sieben Tage in Abbildung 54 noch deutlicher. Auch die negative Korrelation mit Koeffizienten von etwa -0,6 fällt etwas stärker aus.

**Abbildung 54: Stromverbrauch und Tagesstunden (gleitender Durchschnitt)**

---

<sup>309</sup> [2-Data Understanding/03b-Tagesstunden \(Verlauf\)](#)

## Erstellung des Vorhersagemodells



Quelle: Eigene Darstellung<sup>310</sup>

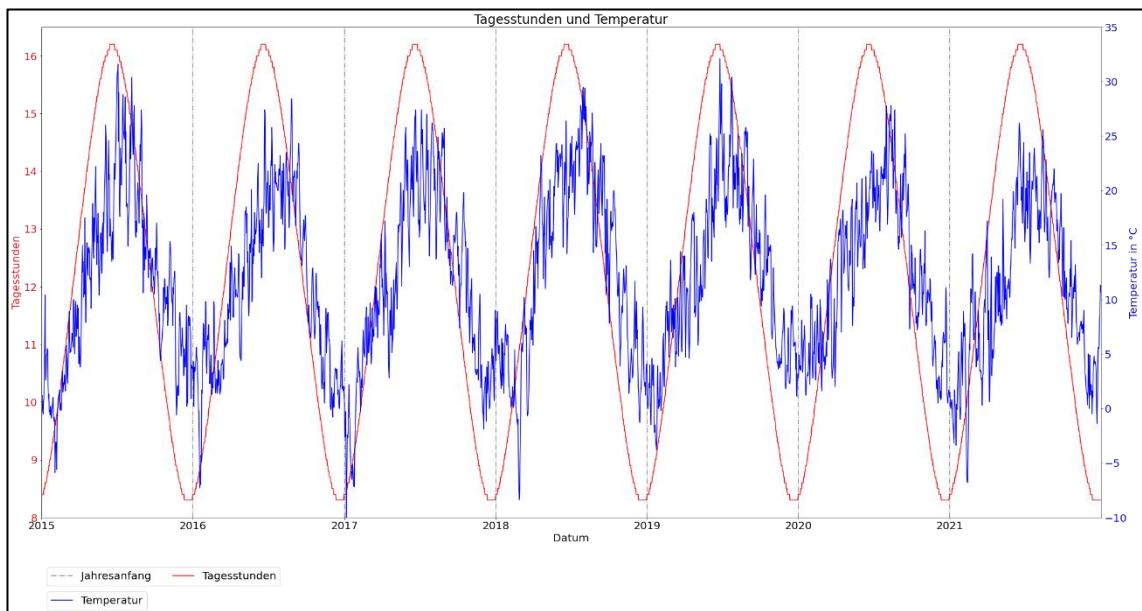
Die Anzahl an Tagesstunden hängt wie auch die Temperatur von meteorologisch-astronomischen Faktoren ab<sup>311</sup>, daher erreichen beide Merkmale ihr Maximum im Sommer und fallen im Winter wieder. Die Abbildung 55 zeigt die gemeinsame Auf- und Abwärtsbewegung beziehungsweise die Saisonalität der beiden Merkmale. Temperatur und Tagesstunden korrelieren miteinander, die Kausalität liegt aber in beiden Fällen in natürlich-bedingten Faktoren begründet.

**Abbildung 55: Tagesstunden und Temperatur**

<sup>310</sup> [2-Data Understanding/03b-Tagesstunden \(Verlauf der Durchschnitte\)](#)

<sup>311</sup> Vgl. Ahrens, C., Henson, R., Meteorologie, 2021, S. 59 ff.

## Erstellung des Vorhersagemodells



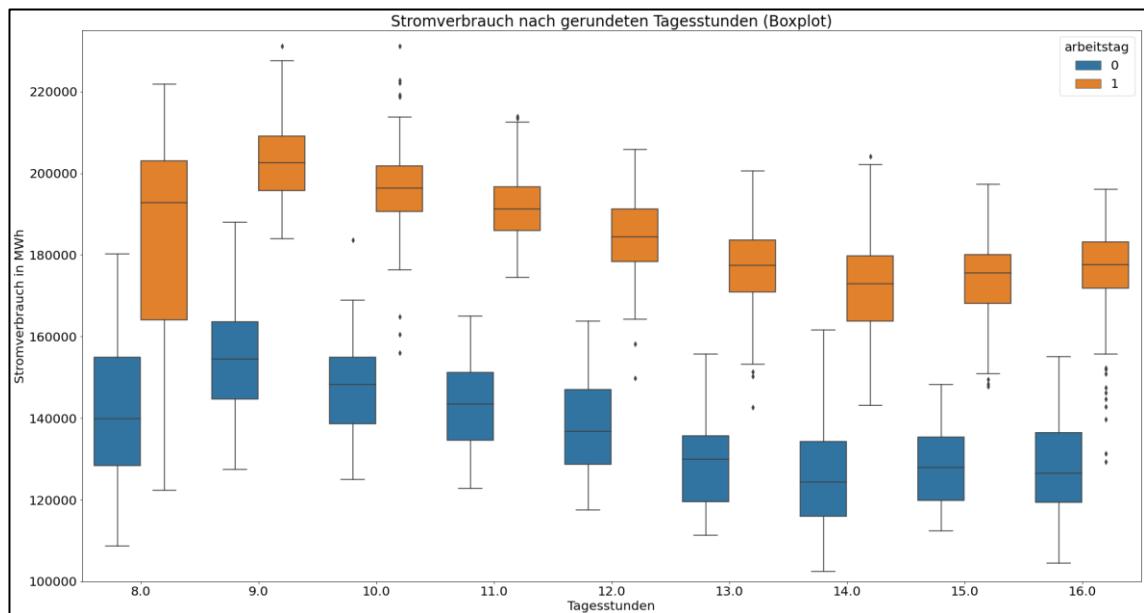
Quelle: Eigene Darstellung<sup>312</sup>

Die Abbildung 56 zeigt den Stromverbrauch nach der Anzahl an Tagesstunden in Boxplots, jeweils nach Arbeits- und arbeitsfreien Tagen getrennt. Es fällt zunächst auf, dass der Stromverbrauch mit steigenden Tagesstunden fällt. Gegen Ende Dezember erreichen die Tagesstunden ihr Minimum, wie bereits beschrieben ist der Stromverbrauch in diesem Zeitraum aufgrund der Feiertage und Urlaubssaison besonders niedrig. Der niedrige Verbrauch bei wenigen Tagesstunden ist daher auf diese Besonderheit zurückzuführen. Von etwa neun Tagesstunden aus gesehen fällt der Stromverbrauch dann, je länger die Tage werden. Ab etwa 14,5 Tagesstunden beginnt der Stromverbrauch aber wieder zu steigen.

**Abbildung 56: Stromverbrauch nach Tagesstunden (Boxplot)**

<sup>312</sup> [2-Data Understanding/03b-Tagesstunden \(Tagesstunden und Temperatur\)](#)

## Erstellung des Vorhersagemodells



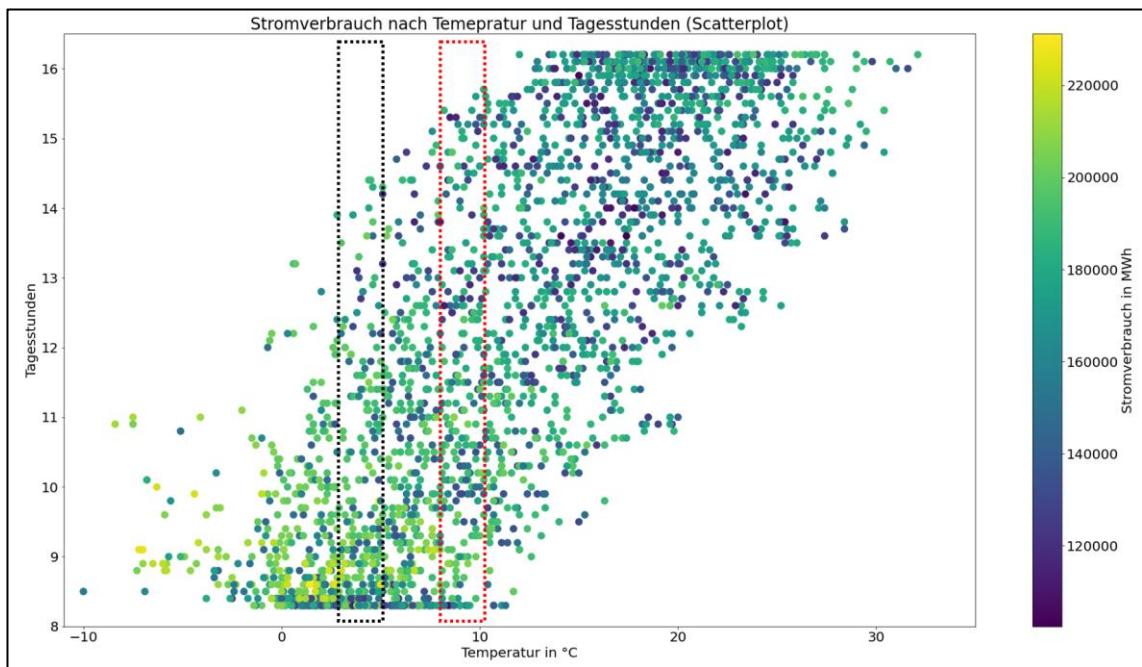
Quelle: Eigene Darstellung<sup>313</sup>

Die Abbildung 57 zeigt den Stromverbrauch nach Temperatur und Tagesstunden im Scatterplot, gelbe Punkte repräsentieren Tage mit besonders hohem Stromverbrauch, blaue Tage haben einen niedrigeren Stromverbrauch. Hier zeigt sich noch einmal die Korrelation von Temperatur und Tagesstunden. Die Punkte im schwarzen und roten Kasten haben jeweils eine sehr ähnliche Temperatur und unterscheiden sich im Wesentlichen durch die Anzahl an Tagesstunden. Es ist dabei zu erkennen, dass der Stromverbrauch an Tagen mit ähnlicher Temperatur und wenigen Tagesstunden etwas höher ist als an Tagen mit vielen Tagesstunden, da sich im unteren Bereich der Kästen wesentlich mehr gelbe Punkte befinden als weiter oben.

**Abbildung 57: Stromverbrauch nach Temperatur und Tagesstunden**

<sup>313</sup> [2-Data Understanding/03b-Tagesstunden \(Stromverbrauch nach Tagesstunden \(Boxplot\)\)](#)

## Erstellung des Vorhersagemodells



Quelle: Eigene Darstellung<sup>314</sup>

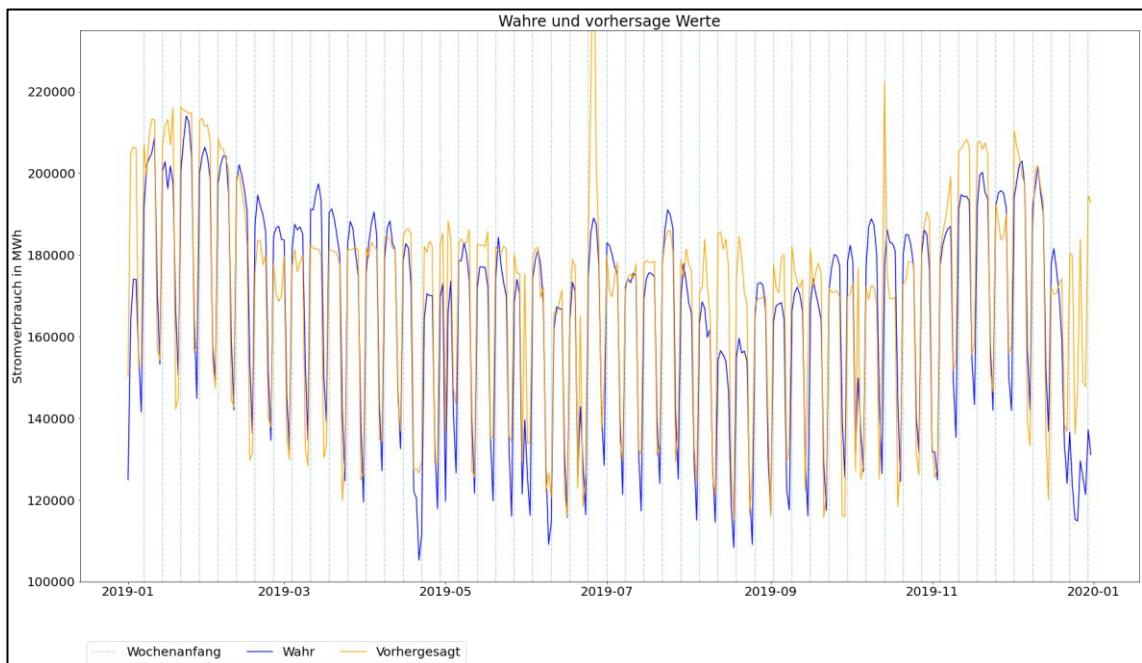
Es lässt sich insgesamt festhalten, dass die Temperatur und die Tagesstunden stark miteinander korrelieren. Trotzdem bietet die Anzahl an Tagesstunden einen geringen, aber doch erkennbaren zusätzlichen Informationsgewinn, der von einem Modell gegebenenfalls erfasst werden kann.

Das in Abbildung 52 gezeigte Regressionsmodell wird zusätzlich durch die Anzahl an Tagesstunden erweitert. Die Abbildung 58 zeigt die Ergebnisse des Regressionsmodells. Der generelle Verlauf des Stromverbrauchs lässt sich an vielen Stellen wie zu erwarten etwas besser abbilden. Auffällig ist aber, dass es an manchen Stellen sehr starke Ausreißer gibt, weshalb sich die Abweichung insgesamt durch hinzuziehen der Tagesstunden nicht verbessert.

**Abbildung 58: Regression mit Arbeitstag, Temperatur Tagesstunden**

<sup>314</sup> [2-Data Understanding/03b-Tagesstunden \(Stromverbrauch nach Tagesstunden und Temperatur \(Scatterplot\)\)](#)

## Erstellung des Vorhersagemodells



Quelle: Eigene Darstellung<sup>315</sup>

### 3.3.4 Analyse des Einflusses durch die Luftfeuchtigkeit<sup>316</sup>

Bei dem in den Daten enthaltenen Merkmal für die Luftfeuchtigkeit handelt es sich um die relative Luftfeuchtigkeit . Sie gibt also an, wie viel Prozent der maximal möglichen Feuchte tatsächlich in der Luft enthalten ist. Da die Luftfeuchtigkeit in allen Städten sehr stark miteinander korreliert, kann auch hier wie bei der Temperatur der Durchschnitt über alle Städte verwendet werden. Genaueres kann dem entsprechenden Notebook entnommen werden.<sup>317</sup> Die Abbildung 59 zeigt den Stromverbrauch (rot) und die Luftfeuchtigkeit (blau) mit einer Amplitude von etwa 30% bis 100% in einem Liniendiagramm. Es ist erkennbar, dass sich Stromverbrauch und Luftfeuchtigkeit ähnlich verhalten. Beide Merkmale erreichen Mitte des Jahres ihr Maximum und fallen zum Ende des Jahres hin wieder.

**Abbildung 59: Stromverbrauch und Luftfeuchtigkeit**

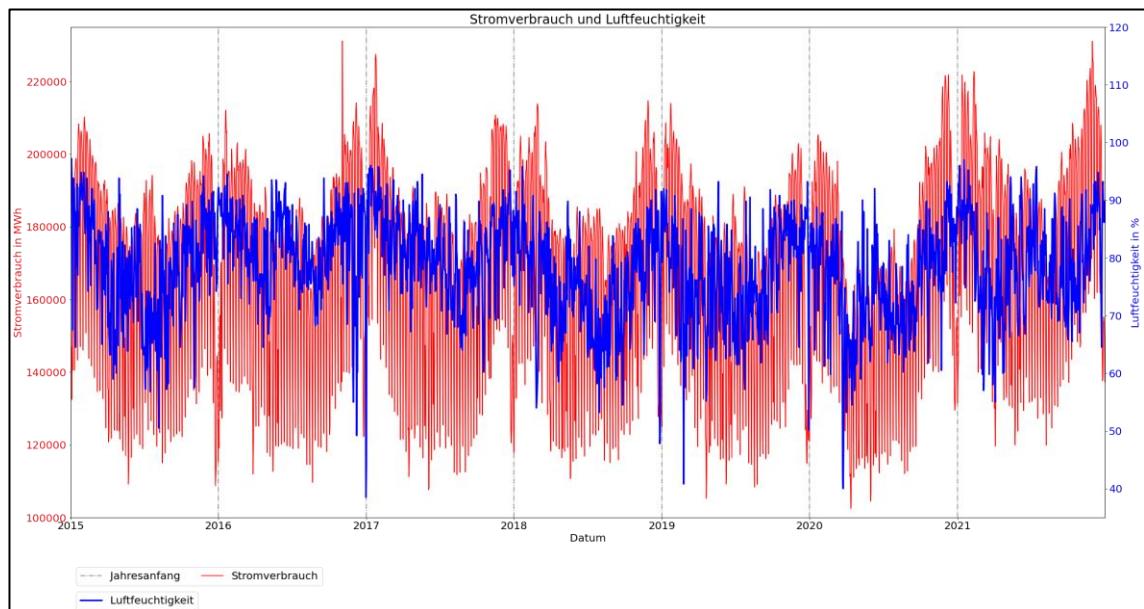
<sup>315</sup> [2-Data Understanding/03b-Tagesstunden \(Regression mit Arbeitstag, Temperatur, Tagesstunden\)](#)

<sup>316</sup> [2-Data Understanding/04a-Luftfeuchtigkeit \(Vergleich\)](#);

[2-Data Understanding/04b-Luftfeuchtigkeit](#)

<sup>317</sup> [2-Data Understanding/04a-Luftfeuchtigkeit \(Vergleich\)](#)

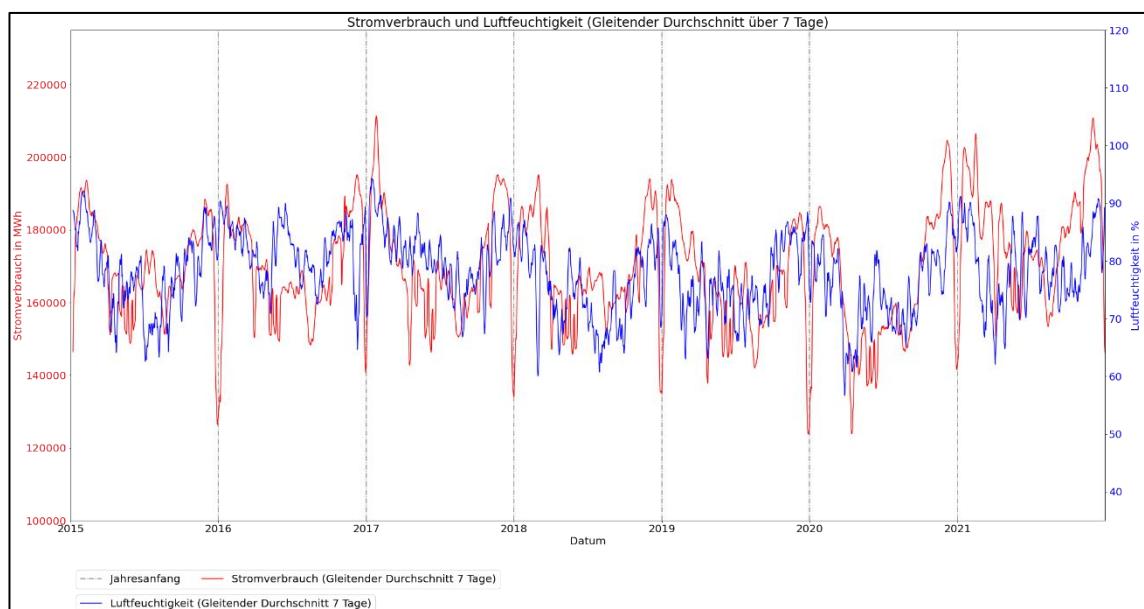
## Erstellung des Vorhersagemodells



Quelle: Eigene Darstellung<sup>318</sup>

Die Abbildung 60 zeigt die gleitenden Durchschnitte über sieben Tage. Hier ist die sehr ähnliche Bewegung noch deutlicher, es handelt sich um eine leicht positive Korrelation von etwa +0,4. Es ist auch erkennbar, dass sich die beiden Merkmale beispielsweise Mitte 2016 oder 2018 teilweise asynchron bewegen.

**Abbildung 60: Stromverbrauch und Luftfeuchtigkeit (gleitender Durchschnitt)**



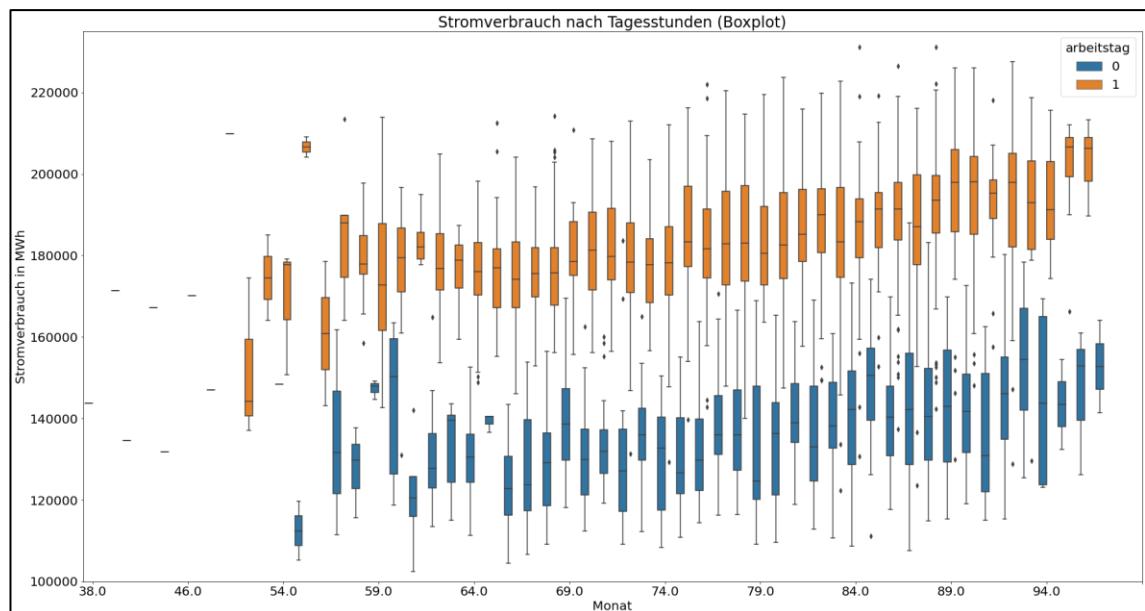
<sup>318</sup> [2-Data Understanding/04b-Luftfeuchtigkeit \(Verlauf\)](#)

## Erstellung des Vorhersagemodells

Quelle: Eigene Darstellung<sup>319</sup>

Die Boxplots des Stromverbrauchs nach Luftfeuchtigkeit jeweils nach Arbeits- und arbeitsfreien Tagen in der Abbildung 61 zeigen, dass der Stromverbrauch mit steigender Luftfeuchtigkeit zuzunehmen scheint.

**Abbildung 61: Stromverbrauch nach Luftfeuchtigkeit (Boxplot)**



Quelle: Eigene Darstellung<sup>320</sup>

Dabei muss ein wichtiger Effekt beachtet werden. Je wärmer die Luft ist, desto mehr Wasser kann sie aufnehmen. Die gleiche Menge an Wasser in der Luft führt bei einer höheren Temperatur also zu einer niedrigeren relativen Luftfeuchtigkeit als bei geringeren Temperaturen.<sup>321</sup> So zeigt sich auch in Abbildung 62, dass sich Temperatur und Luftfeuchtigkeit in entgegengesetzte Richtungen bewegen. Sie korrelieren mit Koeffizienten von etwa -0,45 mittelstark negativ.

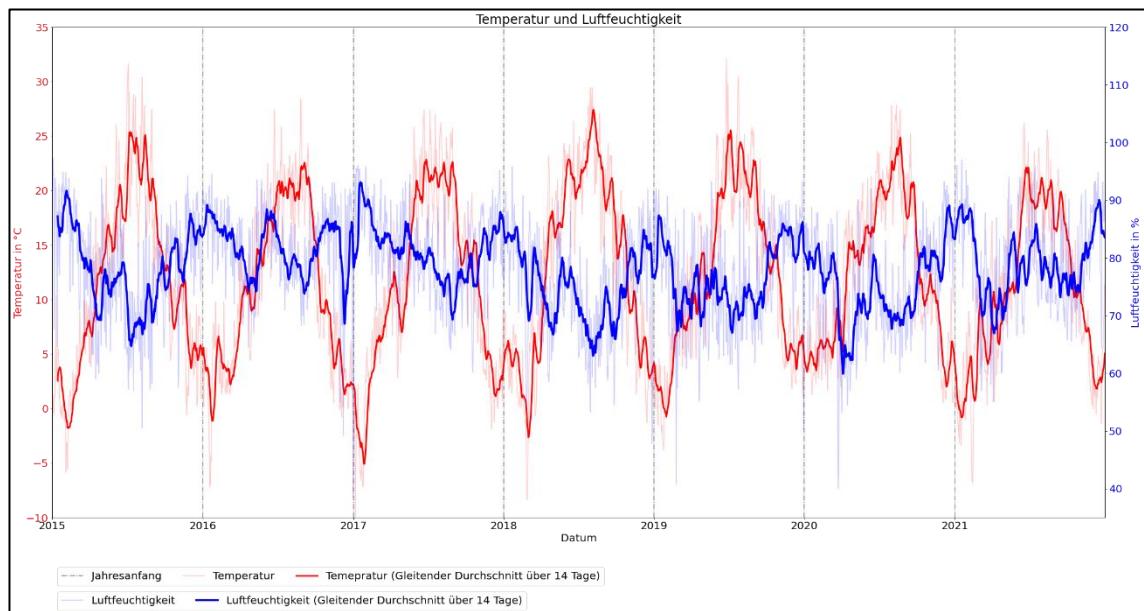
**Abbildung 62: Temperatur und Luftfeuchtigkeit**

<sup>319</sup> [2-Data Understanding/04b-Luftfeuchtigkeit \(Verlauf der Durchschnitte\)](#)

<sup>320</sup> [2-Data Understanding/04b-Luftfeuchtigkeit \(Stromverbrauch nach Luftfeuchtigkeit \(Boxplot\)\)](#)

<sup>321</sup> Vgl. Häckel, H., Meteorologie, 2021, S. 51 ff.

## Erstellung des Vorhersagemodells



Quelle: Eigene Darstellung<sup>322</sup>

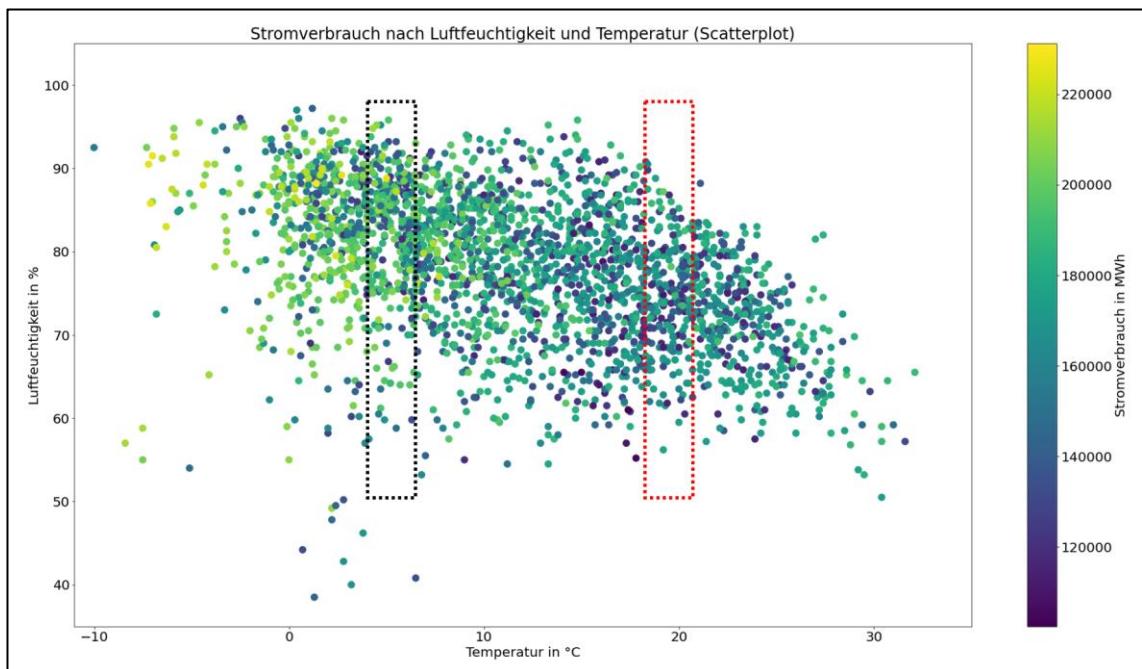
Eine höhere Temperatur geht häufig mit einer niedrigeren relativen Luftfeuchtigkeit einher.<sup>323</sup> Der niedrigere Stromverbrauch bei höherer relativer Luftfeuchtigkeit ist daher eher auf die niedrigere Temperatur zurückzuführen. Es handelt sich hierbei also um eine Scheinkorrelation, da die Veränderungen beider Merkmale voneinander unabhängig sind, aber im kausalen Zusammenhang mit der Temperatur stehen. Dadurch erklären sich auch die an manchen Stellen beobachtbaren asynchronen Verläufe des Stromverbrauchs und der Luftfeuchtigkeit. Die Abbildung 63 zeigt den Stromverbrauch nach Luftfeuchtigkeit und Temperatur als Scatterplot. Die im roten und schwarzen Kasten eingegrenzten Punkte haben in etwa die gleiche Temperatur, unterscheiden sich aber durch die Luftfeuchtigkeit. Es zeigt sich hier, dass die Luftfeuchtigkeit keinen oder nur einen sehr geringen Einfluss auf den Stromverbrauch zu haben scheint.

**Abbildung 63: Stromverbrauch nach Luftfeuchtigkeit u. Temperatur (Scatterplot)**

<sup>322</sup> [2-Data Understanding/04b-Luftfeuchtigkeit \(Luftfeuchtigkeit und Temperatur\)](#)

<sup>323</sup> Vgl. Hering, E., Schönfelder, G., Wissenschaft und Technik, 2018, S. 474 ff.

## Erstellung des Vorhersagemodells



Quelle: Eigene Darstellung<sup>324</sup>

Andere Untersuchungen zeigten zwar, dass die relative Luftfeuchtigkeit durchaus einen Einfluss auf den Stromverbrauch privater Haushalte haben kann, dieser hält sich aber in Grenzen<sup>325</sup>. Zudem wird nur etwa ein Viertel des Stromverbrauchs durch private Haushalte verursacht.<sup>326</sup> Da im Rahmen dieser Untersuchung der Stromverbrauch des gesamten Bundeslandes samt Industrie und Gewerbe betrachtet wird, fällt die Luftfeuchtigkeit hier nur gering bis gar nicht ins Gewicht und kann im weiteren Verlauf ignoriert werden. Dies bestätigt sich auch, wenn das in Abbildung 58 gezeigte Regressionsmodell durch die Luftfeuchtigkeit erweitert wird. Die Abbildung 64 zeigt die Ergebnisse des erweiterten Regressionsmodells. Der Stromverbrauch kann wesentlich schlechter abgebildet werden. Die Abweichung verschlechtert sich auf etwa 9,5%. Das zusätzliche Merkmal scheint keinen zusätzlichen Informationsgewinn zu bringen, stattdessen werden die anderen, aussagekräftigeren Merkmale scheinbar sogar verwässert, wodurch die Qualität des Modells abnimmt.

**Abbildung 64: Regression mit Arbeitstag, Temperatur, Tagesstunden und Luftfeuchtigkeit**

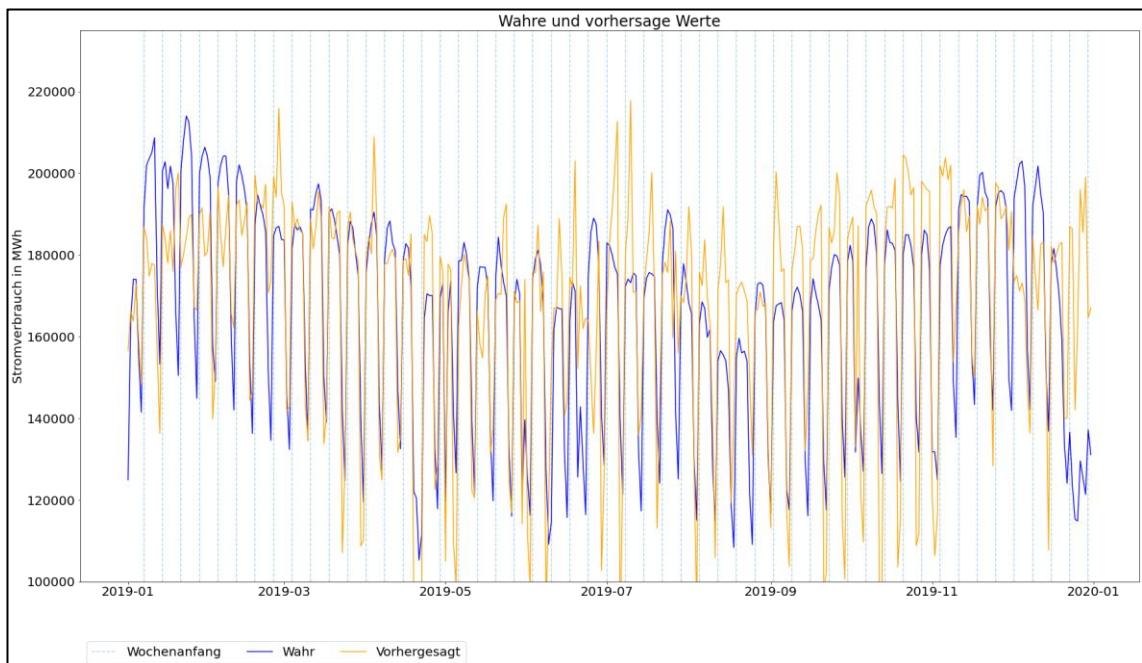
<sup>324</sup> [2-Data Understanding/04b-Luftfeuchtigkeit \(Stromverbrauch nach Luftfeuchtigkeit und Temperatur \(Scatterplot\)\)](#)

<sup>325</sup> Vgl. Maia-Silva, D., Kumar, R., Nateghi, R., Humidity and Electricity Demand, 2020;

Vgl. Kang, J., Reiner, D., Weather and Electricity Consumption, 2021.

<sup>326</sup> Vgl. Bundesverband der Energie- und Wasserwirtschaft, Stromverbrauch, 2021.

## Erstellung des Vorhersagemodells



Quelle: Eigene Darstellung<sup>327</sup>

### 3.3.5 Analyse der weiteren Wetterdaten

Neben den bisher analysierten Merkmalen gibt es außerdem noch den Niederschlag in Millimetern pro Quadratmeter<sup>328</sup>, die Windgeschwindigkeit in Stundenkilometern<sup>329</sup>, die Sichtweite in Kilometern<sup>330</sup> und eine zusammengefasste Beschreibung der Wetterbedingungen<sup>331</sup>. Die genauen Analysen können den Notebooks entnommen werden und sind hier nur sehr verkürzt dargestellt.

Die Wetterbedingungen lassen sich in fünf Kategorien einteilen, jeweils für nebliges, verregnetes, verschneites, bewölktes oder klares Wetter. Die Abbildung 65 zeigt den Stromverbrauch je Wetterbedingung nach Arbeits- und arbeitsfreien Tagen getrennt. Es zeigt sich, dass der Stromverbrauch bei allen Wetterbedingungen ähnlich ist.

**Abbildung 65: Stromverbrauch nach Wetterbedingungen (Boxplot)**

<sup>327</sup> [2-Data Understanding/04b-Luftfeuchtigkeit \(Regression mit Arbeitstag, Temperatur, Tagesstunden und Luftfeuchtigkeit\)](#)

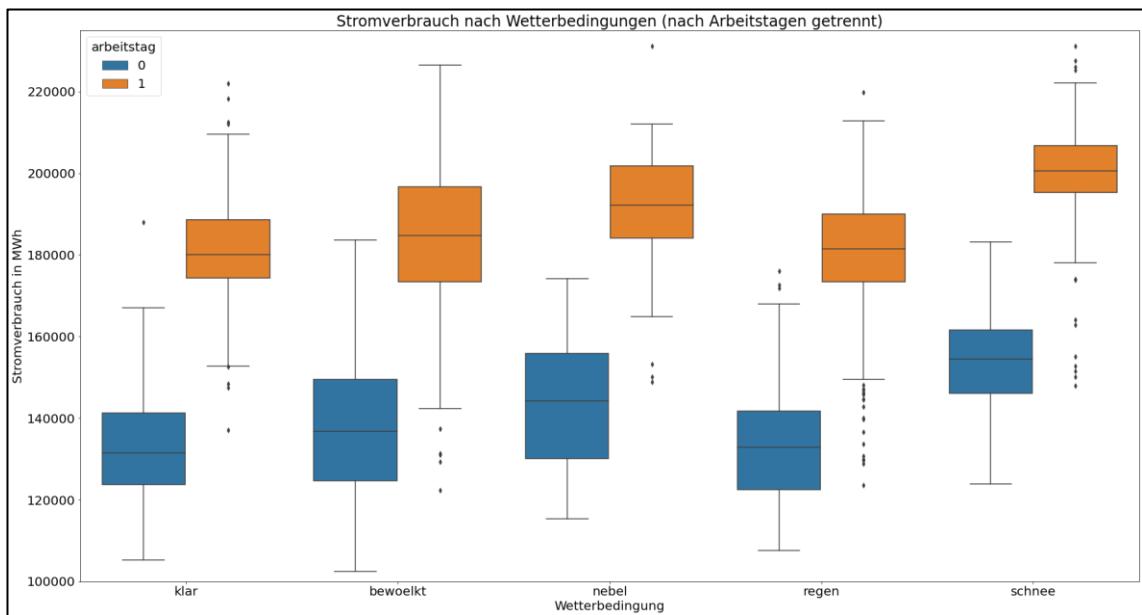
<sup>328</sup> [2-Data Understanding/06a-Niederschlag \(Vergleich\); 2-Data Understanding/06b-Niederschlag](#)

<sup>329</sup> [2-Data Understanding/07a-Windgeschwindigkeit \(Vergleich\); 2-Data Understanding/07b-Windgeschwindigkeit](#)

<sup>330</sup> [2-Data Understanding/08a-Sichtweite \(Vergleich\); 2-Data Understanding/08b-Sichtweite](#)

<sup>331</sup> [2-Data Understanding/05-Wetter](#)

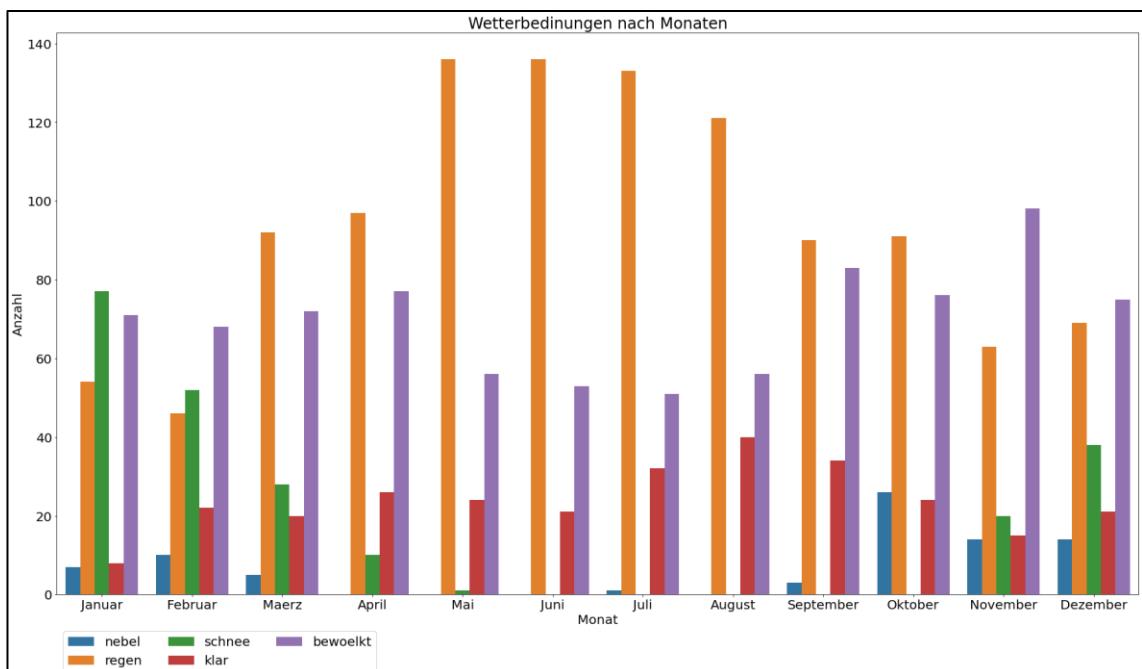
## Erstellung des Vorhersagemodells



Quelle: Eigene Darstellung<sup>332</sup>

Die etwas höheren Werte beim nebligen und verschneiten Wetter sind darauf zurückzuführen, dass diese Wetterbedingungen vorrangig in kälteren Monaten auftreten, wie man aus der Abbildung 66 entnehmen kann.

**Abbildung 66: Wetterbedingungen nach Monaten**



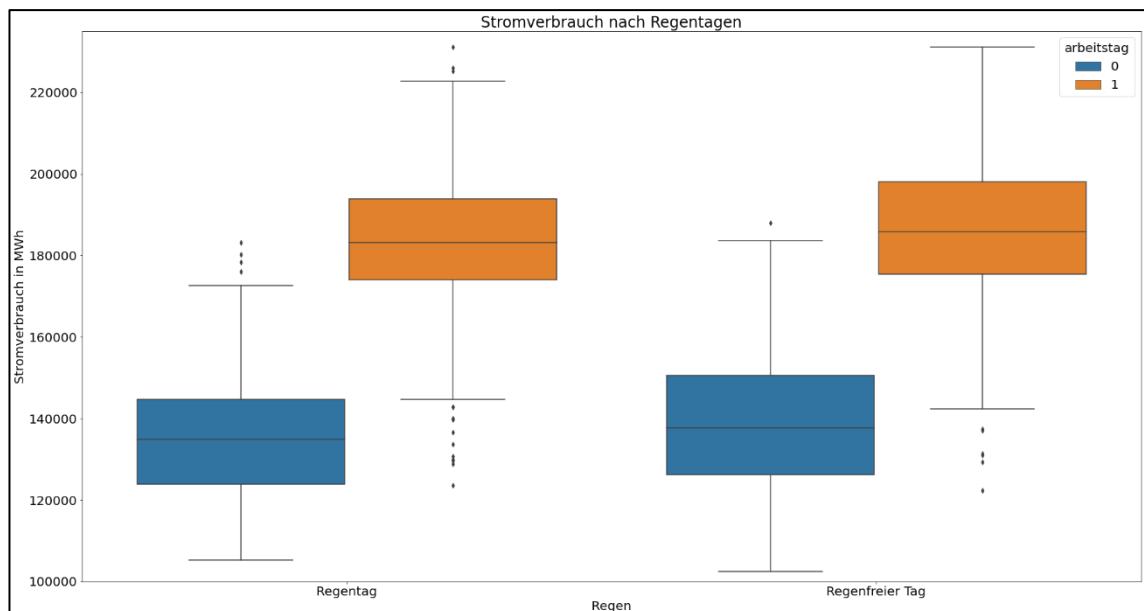
<sup>332</sup> [2-Data Understanding/05-Wetter \(Stromverbrauch nach Wetterbedingungen\)](#)

## Erstellung des Vorhersagemodells

Quelle: Eigene Darstellung<sup>333</sup>

Der Niederschlag hat keinen erkennbaren Einfluss auf den Stromverbrauch. So unterscheiden sich Regentage kaum von Tagen ohne Regen, wie in der Abbildung 67 erkennbar ist.

**Abbildung 67: Stromverbrauch nach Regentagen**



Quelle: Eigene Darstellung<sup>334</sup>

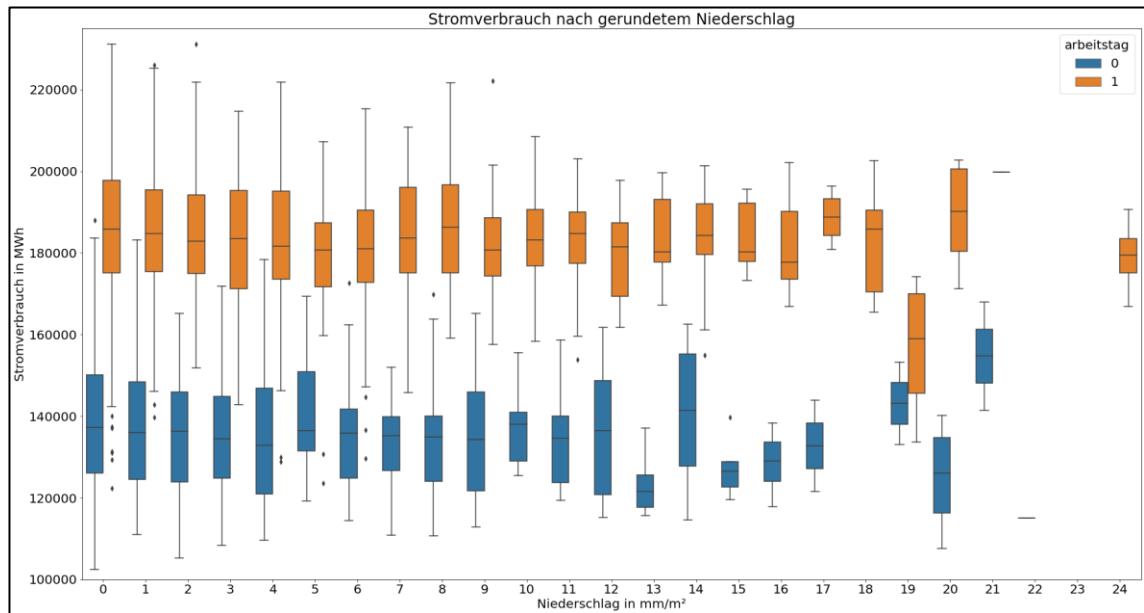
Die Abbildung 68 zeigt den Stromverbrauch nach Niederschlag jeweils an Arbeits- und arbeitsfreien Tagen, auch hier kann kein Unterschied anhand des Niederschlags festgestellt werden.

**Abbildung 68: Stromverbrauch nach gerundetem Niederschlag**

<sup>333</sup> [2-Data Understanding/05-Wetter \(Wetterbedingungen nach Monaten\)](#)

<sup>334</sup> [2-Data Understanding/06b-Niederschlag \(Stromverbrauch nach Regentagen \(Boxplot\)\)](#)

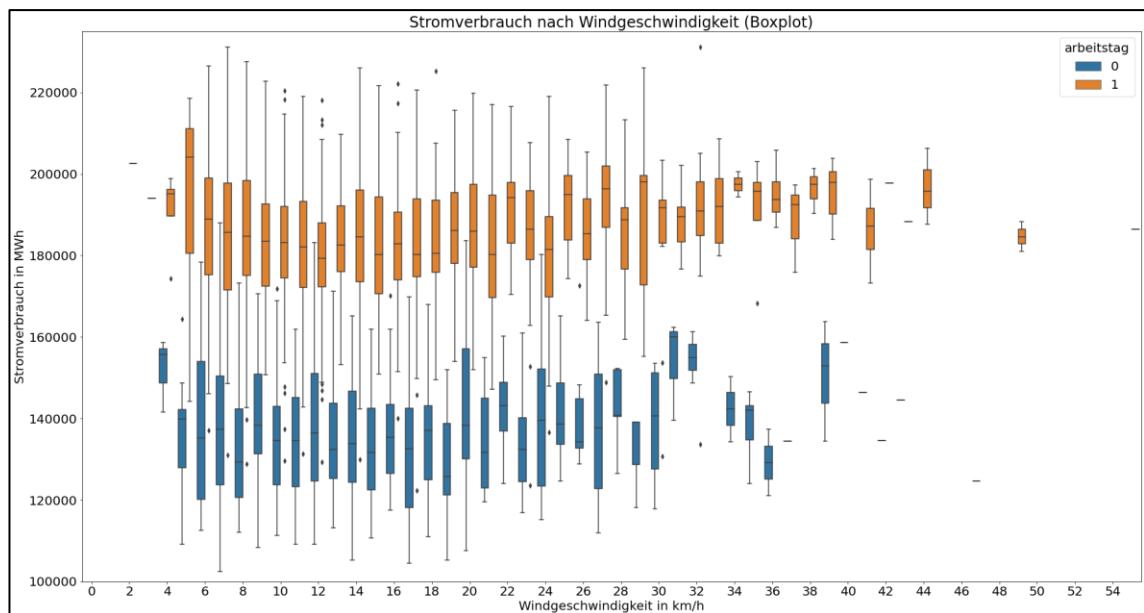
## Erstellung des Vorhersagemodells



Quelle: Eigene Darstellung<sup>335</sup>

Ebenso lässt sich auch bei der Windgeschwindigkeit kein Zusammenhang zum Stromverbrauch erkennen, wie beispielsweise in der Abbildung 69 erkennbar ist.

**Abbildung 69: Stromverbrauch nach Windgeschwindigkeit (Boxplot)**



Quelle: Eigene Darstellung<sup>336</sup>

<sup>335</sup> 2-Data Understanding/06b-Niederschlag (Stromverbrauch nach Niederschlag (Boxplot))

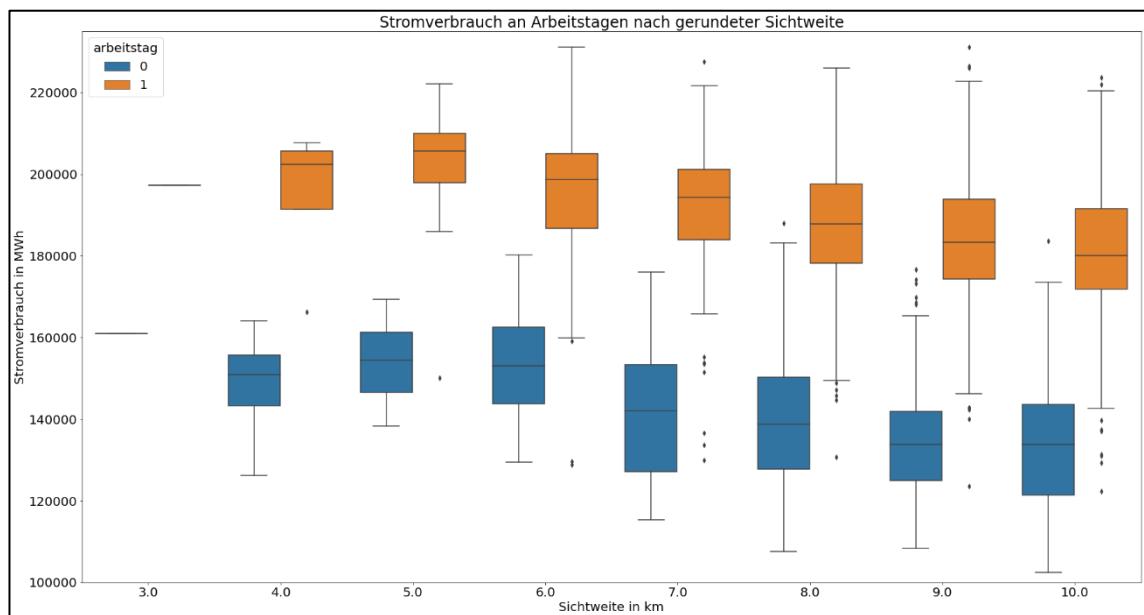
<sup>336</sup> 2-Data Understanding/07b-Windgeschwindigkeit (Stromverbrauch nach Windgeschwindigkeit (Boxplot))

## Erstellung des Vorhersagemodells

Bei der Sichtweite lässt sich ebenso kein Effekt auf den Stromverbrauch erkennen.

Zwar zeigt sich auf den ersten Blick eine leichte Abnahme des Verbrauchs mit steigender Sichtweite, wie in Abbildung 70 zu sehen ist.

**Abbildung 70: Stromverbrauch nach gerundeter Sichtweite**



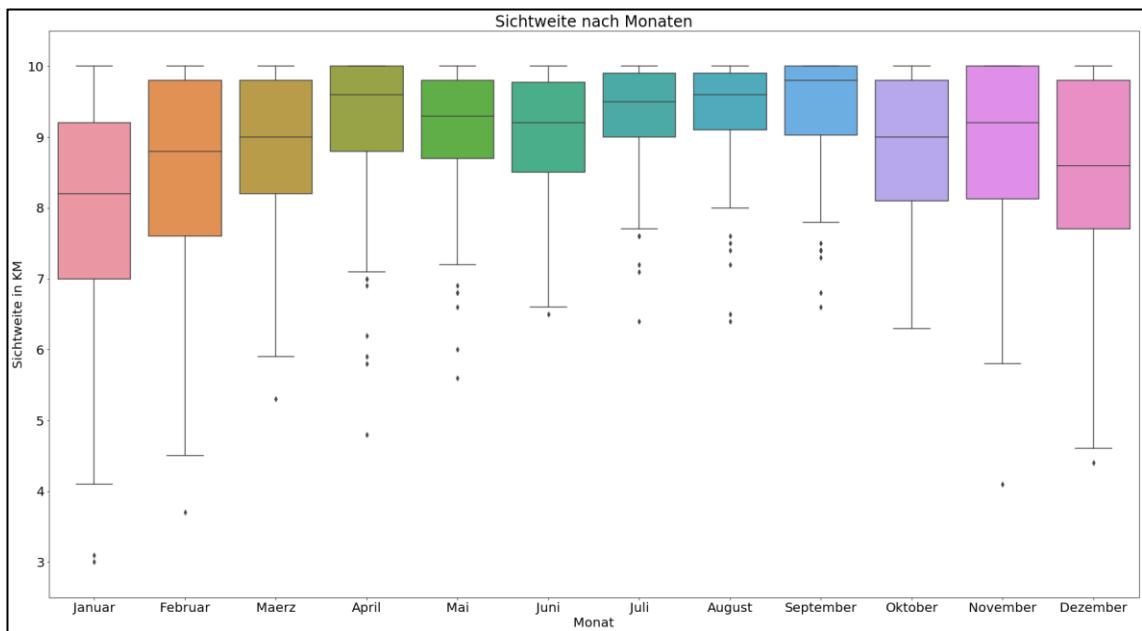
Quelle: Eigene Darstellung<sup>337</sup>

Dieser Effekt lässt sich aber auf das überwiegend klare Wetter in den Sommermonaten mit einem niedrigen Stromverbrauch zurückführen. Im Winter ist die Sicht aufgrund der Wetterbedingungen oft eingeschränkt, wie der Abbildung 71 und der Abbildung 66 entnommen werden kann.

**Abbildung 71: Sichtweite nach Monaten**

<sup>337</sup> [2-Data Understanding/08b-Windgeschwindigkeit \(Stromverbrauch nach Sichtweite \(Boxplot\)\)](#)

## Erstellung des Vorhersagemodells



Quelle: Eigene Darstellung<sup>338</sup>

Auch das in Abbildung 58 gezeigte Regressionsmodell lässt sich nicht durch Niederschlag, Windgeschwindigkeit oder Sichtweite verbessern. Die genaueren Analysen können in den Notebooks eingesehen werden.

### 3.4 Data Preparation<sup>339</sup>

In diesem Kapitel wird die Erstellung eines fertigen Datensatzes für das weitere Modeling beschrieben. Es werden zunächst alle Daten geladen. Dann findet die Feature Selection<sup>340</sup> statt, bei der die irrelevanten Merkmale entfernt werden. Beim anschließenden Feature Engineering<sup>341</sup> werden gegebenenfalls neue Merkmale aus den bestehenden Merkmalen generiert. Im letzten Schritt werden die Daten im Rahmen der sogenannten Featurization<sup>342</sup> skaliert. Die Schritte können im entsprechenden Notebook nachverfolgt werden.<sup>343</sup>

Im Rahmen der Feature Selection werden neben dem Stromverbrauch die Merkmale Temperatur, Sonnenaufgang, Sonnenuntergang sowie der Indikator für Arbeitstage

<sup>338</sup> [2-Data Understanding/08b-Windgeschwindigkeit \(Sichtweite nach Monaten\)](#)

<sup>339</sup> [3-Data Preparation](#)

<sup>340</sup> Vgl. Raschka, S., Mirjalili, V., Machine Learning mit Python, 2021, S. 150 ff.

<sup>341</sup> Vgl. Lazzeri, F., Machine Learning, 2021, S. 89 ff.

<sup>342</sup> Vgl. Körner, C., Waaijer, K., Machine Learning, S. 248 ff.

<sup>343</sup> [3-Data Preparation/01-Data Preparation \(Training und Test\)](#)

ausgewählt. Aus dem Sonnenaufgang und Sonnenuntergang wird im Rahmen des Feature Engineering die Anzahl an Tagesstunden berechnet.

Die Merkmale bewegen sich auf sehr unterschiedlichen Amplituden. Der Stromverbrauch reicht beispielsweise von 102.469 bis 231.190, die Temperatur hingegen nur von -10 bis +32,1. Diese stark unterschiedlichen Amplituden beziehungsweise Skalen können den Anlernprozess vieler Algorithmen erschweren, da die Koeffizienten gegebenenfalls sehr stark unterschiedlich groß oder klein sein müssen.<sup>344</sup> Außerdem können die Abstände zwischen bestimmten Datenpunkten nicht mehr richtig interpretiert werden.<sup>345</sup> Beispielsweise ist eine Änderung der Temperatur um 10 °C nach oben sehr viel bedeutsamer als eine Änderung des Stromverbrauchs um 100 MWh, in absoluten Zahlen ist die Änderung allerdings sehr viel geringer. Daher werden die Merkmale im Rahmen der Featurization normalisiert<sup>346</sup>, also auf eine Skala zwischen null und eins umgewandelt. Da es sich ausschließlich um metrisch-intervall- beziehungsweise metrisch-verhältnisskalierte Merkmale oder dichotome Daten ohne fehlende oder erkennbar falsche Werte oder potenziell problematische Ausreißer handelt, müssen keine weiteren Anpassungen an den Daten gemacht werden.

Die Abbildung 72 zeigt eine Übersicht über die (unskalierten) Daten samt Beschafffenheit und zusammenfassender Statistik.

Abbildung 72: Unskalierte Daten nach Data Preparation

	verbrauch	arbeitstag	temperatur	tagesstunden
datum				
2015-01-01	126197	0	-2.5	8.4
2015-01-02	147085	1	-0.0	8.4
2015-01-03	141426	0	1.2	8.4
2015-01-04	132446	0	-0.2	8.4
2015-01-05	152611	1	-0.5	8.4

	count	mean	std	min	25%	50%	75%	max
verbrauch	2557.0	169329.063355	27116.871253	102469.0	147992.0	175584.0	190001.0	231190.0
arbeitstag	2557.0	0.687133	0.463751	0.0	0.0	1.0	1.0	1.0
temperatur	2557.0	11.706492	7.845246	-10.0	5.2	11.2	18.0	32.1
tagesstunden	2557.0	12.254634	2.700311	8.3	9.7	12.3	14.9	16.2

Quelle: Eigene Darstellung<sup>347</sup>

<sup>344</sup> Vgl. Hirschle, J., Machine Learning für Zeitreihen, 2020, S. 158.

<sup>345</sup> Vgl. Auffarth, B., Machine Learning for Time Series, 2021, S. 69 f.

<sup>346</sup> Vgl. Raschka, S., Mirjalili, V., Machine Learning mit Python, 2021, S. 148 ff.

347 vgl. Rasika, S., Mirjam, V., Machine Learning mit Python, 3-Data Preparation/01-Data Preparation (Training und Test)

### 3.5 Modeling (ARIMA)<sup>348</sup>

Wie bereits in Kapitel 2.4 beschrieben basiert ein ARIMA-Modell zunächst auf den endogenen Daten der vorherzusagenden Zeitreihe, welche dann um exogene Daten erweitert werden. Aus diesem Grund wird im ersten Schritt ein ARIMA-Modell ausschließlich mit endogenen Daten entwickelt, welches im zweiten Schritt dann um exogene Daten erweitert wird. Die Entwicklung erfolgt nach dem sogenannten Box-Jenkins-Ansatz<sup>349</sup>, wobei zunächst ein Parametersatz ermittelt wird, der dann in einem Modell angewandt, getestet und bewertet wird, um anschließend möglicherweise bessere Parametersätze zu ermitteln. Zur Bewertung werden der MAPE, die Log-Likelihood und das AIC verwendet. Dieser Prozess wird wiederholt, bis ein möglichst gutes Modell gefunden wurde. Die Implementierung des SARIMAX-Modells erfolgt mit der Klasse SARIMAX<sup>350</sup> aus der Bibliothek statsmodels. Zur Vereinfachung wird ARIMA im weiteren als Synonym für SARIMA bzw. SARIMAX verwendet.

Die Autokorrelationen wurden bereits in der Abbildung 44 und der Abbildung 45 analysiert und können im Notebook eingesehen werden.<sup>351</sup> Aus den Autokorrelationsdiagrammen lässt sich schließen, dass die unmittelbar vorherigen Beobachtungen und die korrespondierenden Beobachtungen der letzten saisonalen Perioden einen starken Einfluss auf die aktuelle Beobachtung haben.

#### 3.5.1 Parameterbestimmung mit auto\_arima()<sup>352</sup>

Die Bibliothek pmdarima bietet unter anderem die Funktion auto\_arima()<sup>353</sup>. Dabei werden verschiedene ARIMA-Ordnung ausprobiert und anhand des AIC verglichen. Dadurch soll ein Parametersatz gefunden werden, mit dem sich die Zeitreihe durch möglichst wenige Koeffizienten gut abbilden lässt, um Überanpassungen an die Daten zu verhindern. Die Abbildung 73 zeigt die Ergebnisse der auto\_arima()-Funktion.

**Abbildung 73: Ergebnisse von auto\_arima()**

---

<sup>348</sup> [4-Modeling](#)

<sup>349</sup> Vgl. Chatfield, C., Xing, H., Time Series Analysis, 2019, S. 123 ff.

<sup>350</sup> statsmodels, statsmodels, 2022.

<sup>351</sup> [4-Modeling/01-Analyse](#)

<sup>352</sup> [4-Modeling/02a-ARIMA \(endogen\) \(auto\\_arima\(\)\)](#)

<sup>353</sup> alkaline-ml, auto\_arima(), 2021.

```

Performing stepwise search to minimize aic
ARIMA(2,0,2)(1,0,1)[7] intercept      : AIC=-5830.350, Time=4.60 sec
ARIMA(0,0,0)(0,0,0)[7] intercept      : AIC=-705.558, Time=0.20 sec
ARIMA(1,0,0)(1,0,0)[7] intercept      : AIC=-4833.713, Time=1.79 sec
ARIMA(0,0,1)(0,0,1)[7] intercept      : AIC=-3350.364, Time=1.04 sec
ARIMA(0,0,0)(0,0,0)[7]                : AIC=4297.801, Time=0.11 sec
ARIMA(2,0,2)(0,0,1)[7] intercept      : AIC=-3440.116, Time=3.71 sec
ARIMA(2,0,2)(1,0,0)[7] intercept      : AIC=-4797.181, Time=3.95 sec
ARIMA(2,0,2)(2,0,1)[7] intercept      : AIC=-5824.050, Time=7.09 sec
ARIMA(2,0,2)(1,0,2)[7] intercept      : AIC=-5777.461, Time=9.22 sec
ARIMA(2,0,2)(0,0,0)[7] intercept      : AIC=-2210.906, Time=1.99 sec
ARIMA(2,0,2)(0,0,2)[7] intercept      : AIC=-3938.336, Time=7.21 sec
ARIMA(2,0,2)(2,0,0)[7] intercept      : AIC=-5236.242, Time=6.53 sec
ARIMA(2,0,2)(2,0,2)[7] intercept      : AIC=-5948.055, Time=10.27 sec
ARIMA(1,0,2)(2,0,2)[7] intercept      : AIC=inf, Time=8.71 sec
ARIMA(2,0,1)(2,0,2)[7] intercept      : AIC=-5824.093, Time=7.98 sec
ARIMA(3,0,2)(2,0,2)[7] intercept      : AIC=-5508.323, Time=15.84 sec
ARIMA(2,0,3)(2,0,2)[7] intercept      : AIC=-5472.185, Time=11.56 sec
ARIMA(1,0,1)(2,0,2)[7] intercept      : AIC=-5852.356, Time=7.06 sec
ARIMA(1,0,3)(2,0,2)[7] intercept      : AIC=-5888.663, Time=9.38 sec
ARIMA(3,0,1)(2,0,2)[7] intercept      : AIC=-5922.667, Time=9.41 sec
ARIMA(3,0,3)(2,0,2)[7] intercept      : AIC=-5859.165, Time=12.10 sec
ARIMA(2,0,2)(2,0,2)[7]                : AIC=inf, Time=5.48 sec

Best model: ARIMA(2,0,2)(2,0,2)[7] intercept
Total fit time: 145.221 seconds

```

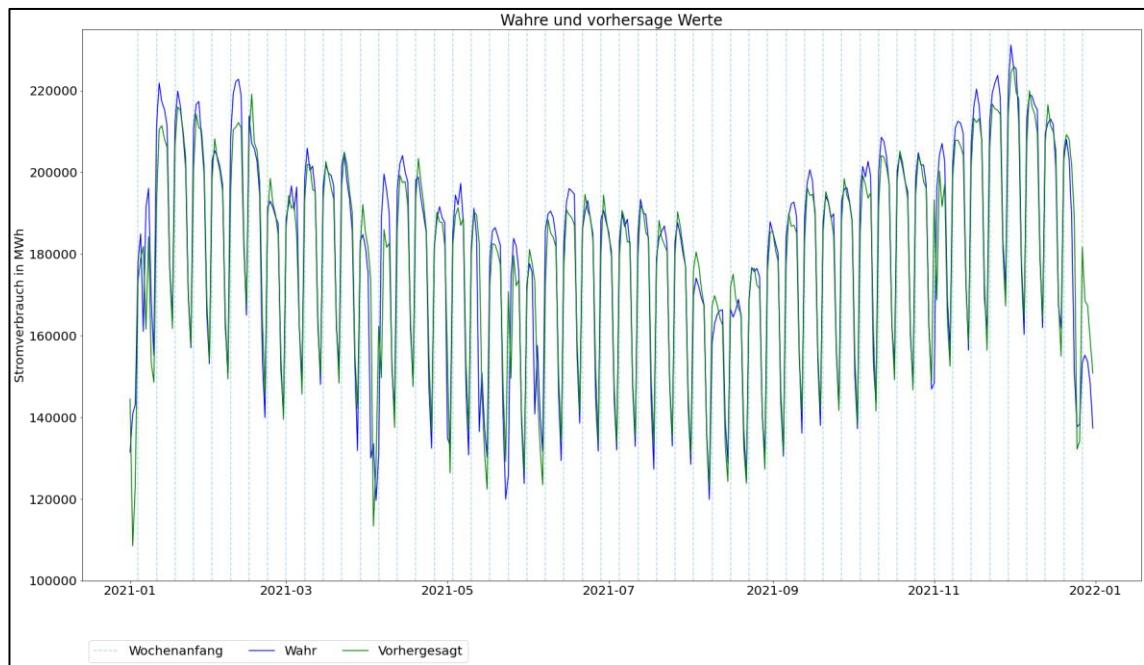
Quelle: Eigene Darstellung<sup>354</sup>

Die Funktion ermittelt ein ARIMA(2,0,2)(2,0,2)7 als bestes Modell. Es werden also die unmittelbar vorherigen, sowie die korrespondierenden Beobachtungen der letzten Saison verwendet, wie es sich auch anhand der ACF- und PACF-Diagramme ableiten lässt. Das Modell wird mit den Trainingsdaten erstellt und anhand der Testdaten bewertet. Die Abbildung 74 zeigt die Vorhersagen des Modells (grün) und die wahren Werte (blau). Das Modell kann die jährliche und wöchentliche Saisonalität abbilden und weicht im Schnitt um etwa 3,21% ab. Die Log-Likelihood liegt bei 3.038 und das AIC bei -6.058. Die Werte sind allerdings nicht absolut interpretierbar und können daher erst im Kontext anderer Modelle bewertet werden.

<sup>354</sup> [4-Modeling/02a-ARIMA \(endogen\) \(auto\\_arima\(\)\)](#)

## Erstellung des Vorhersagemodells

**Abbildung 74: Ergebnisse von ARIMA(2,0,2)(2,0,2)7**



Quelle: Eigene Darstellung<sup>355</sup>

Die Abbildung 75 zeigt die Koeffizienten des Modells, „ar.L1“ und „ar.L2“ stehen für die Koeffizienten der Lags im AR-Modell (p), „ma.L1“ und „ma.L2“ dementsprechend für die Koeffizienten im MA-Modell (q). „ar.S.L7“ und „ar.S.L14“ steht für die Koeffizienten im saisonalen AR-Modell (P), „ma.S.7“ und „ma.S.14“ für die Koeffizienten im saisonalen MA-Modell (Q). In der Spalte „P>|z|“ kann der p-Wert der jeweiligen Regression abgelesen werden. Liegt der Wert unter 0,05, dann kann man davon ausgehen, dass der erkannte Zusammenhang nicht nur zufällig in der jeweiligen Stichprobe auftritt, sondern ein tatsächlicher Zusammenhang in der Zeitreihe ist.

**Abbildung 75: ARIMA(2,0,2)(2,0,2)7**

<sup>355</sup> [4-Modeling/02a-ARIMA \(endogen\) \(ARIMA\(2,0,2\)\(2,0,2\)7\)](#)

## Erstellung des Vorhersagemodells

	coef	std err	z	P> z	[0.025	0.975]
ar.L1	0.9434	0.050	18.823	0.000	0.845	1.042
ar.L2	0.0162	0.045	0.358	0.720	-0.072	0.105
ma.L1	-0.2361	0.051	-4.665	0.000	-0.335	-0.137
ma.L2	-0.2770	0.022	-12.526	0.000	-0.320	-0.234
ar.S.L7	0.0215	0.056	0.383	0.702	-0.088	0.131
ar.S.L14	0.9783	0.056	17.455	0.000	0.868	1.088
ma.S.L7	0.0038	0.052	0.074	0.941	-0.098	0.106
ma.S.L14	-0.9595	0.051	-18.780	0.000	-1.060	-0.859

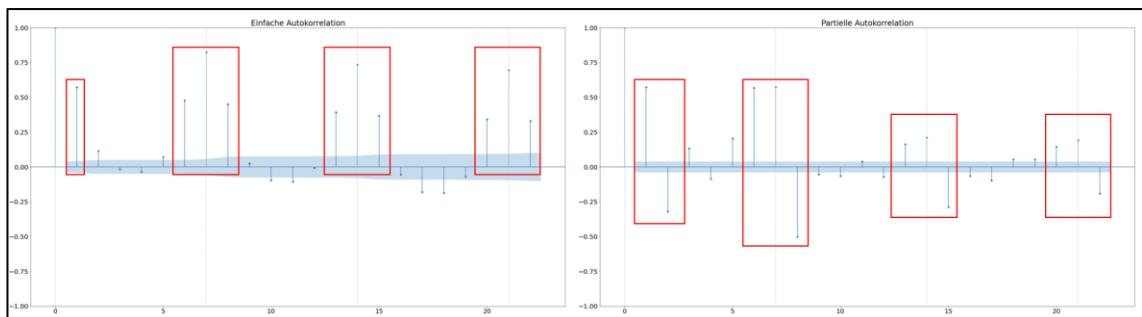
Quelle: Eigene Darstellung<sup>356</sup>

Die Lags ar.L2, ar.S.L7 und ma.S.L7 scheinen nicht statistisch-signifikant zu sein, worauf auch die niedrigen Z-Werte hindeuten. Wenn diese Lags aus dem ARIMA-Modell entfernt werden, ergibt sich ein ARIMA(1,0,2)([2],0,[2])7. Das Modell kommt mit wesentlich weniger Koeffizienten aus, kann die Zeitreihe mit einem MAPE von etwa 3,32% allerdings auch etwas schlechter abbilden. Das AIC steigt dadurch von etwa -6.057 auf -5.930 an, die Log-Likelihood fällt von 3.038 auf etwa 2.971.<sup>357</sup> Das Modell wird also durch die niedrigere Ordnung verschlechtert.

### 3.5.2 Parameterbestimmung mit ACF und PACF<sup>358</sup>

Anhand der Autokorrelationsdiagramme lassen sich verschiedene potenziell geeignete Ordnungen ablesen. In der Abbildung 74 sind Lags mit signifikanter einfacher und partieller Autokorrelation eingezzeichnet.

Abbildung 76: Einfache und partielle Autokorrelation



Quelle: Eigene Darstellung<sup>359</sup>

<sup>356</sup> 4-Modeling/02a-ARIMA (endogen) (ARIMA(2,0,2)(2,0,2)7)

<sup>357</sup> 4-Modeling/02a-ARIMA (endogen) (ARIMA(1,0,2)([2],0,[2])7)

<sup>358</sup> 4-Modeling/02a-ARIMA (endogen) (Lags mit ACF/PACF)

<sup>359</sup> 4-Modeling/01-Analyse

## Erstellung des Vorhersagemodells

Es lassen sich dementsprechend weitere, unterschiedliche Ordnungen für das ARIMA-Modell anwenden. Eine Erweiterung der ARIMA-Module durch zusätzliche Lags führt stellenweise zu einer Verbesserung des MAPE und der Log-Likelihood, allerdings sind diese Verbesserungen gering ausgeprägt und führen daher gleichzeitig zu einem Anstieg des AIC. Die Tabelle 8 gibt eine Übersicht über verschiedene ARIMA-Modelle.

**Tabelle 8: ARIMA-Modelle mit endogenen Daten**

	Modell	MAPE	Log-L.	AIC	Kommentar
1	ARIMA(2,0,2)(2,0,2)7	3,21%	3.038	-6.058	auto_arima()
2	ARIMA(1,0,2)([2],0,[2])7	3,32%	2.971	-5.930	auto_arima() signifikant
3	ARIMA(3,0,3)(3,0,3)7	3,25%	3.003	-5.979	Dritte Ordnungen
4	ARIMA(3,0,2)(0,0,0)7	9,34%	1.103	-2.194	Dritte Ordnungen signifikanten Lags
5	ARIMA(4,0,4)(4,0,4)7	3,19%	3.052	-6.070	Vierte Ordnungen
6	ARIMA([1,2,6],0,2)(3,0,2)7	3,20%	3.035	-6.048	Stark autokorrelierte Lags aus ACF/PACF
7	ARIMA(3,0,2)(2,0,2)7	3,22%	3.030	-6.039	Größeres AR-Modul
8	ARIMA([1,2,4],0,2)(2,0,2)7	3,17%	3.031	-6.043	Größeres AR-Modul mit stark autokorrierten Lags aus PACF
9	ARIMA(1,0,1)(1,0,1)7	3,28%	2.979	-5.947	Erste Ordnungen
10	ARIMA(2,0,1)(2,0,1)7	3,32%	2.956	-5.898	Kleinere MA/SMA-Module
11	ARIMA(1,0,2)(1,0,2)7	3,34%	2.982	-5.950	Kleinere AR/SAR-Module
12	ARIMA(2,0,0)(2,0,0)7	3,82%	2.599	-5.188	Deaktivierung von MA/SMA-Modulen
13	ARIMA(0,0,2)(0,0,2)7	8,94%	1.148	-2.286	Deaktivierung von AR/SAR-Modulen
14	ARIMA(0,0,0)(2,0,2)7	6,02%	1.965	-3.920	Deaktivierung von nicht-saisonalen Modulen mit ACF/PACF
15	ARIMA(0,0,0)(2,0,1)7	6,06%	1.961	-3.914	Deaktivierung von nicht-saisonalen Modulen mit auto_arima()
16	ARIMA(2,0,2)(0,0,0)7	9,50%	1.100	-2.190	Deaktivierung von saisonalen Modulen mit ACF/PACF
17	ARIMA(2,0,5)(0,0,0)7	8,59%	1.358	-2.699	Deaktivierung von saisonalen Modulen mit auto_arima()
18	ARIMA(4,0,2)(0,0,0)7	8,67%	1.139	-2.264	Deaktivierung von saisonalen Modulen mit ACF/PACF

Quelle: Eigene Darstellung<sup>360</sup>

<sup>360</sup> [4-Modeling/02a-ARIMA \(endogen\)](#)

## Erstellung des Vorhersagemodells

Ein ARIMA([1,2,6],0,2)(3,0,2)7 verwendet ein erweitertes AR- und saisonales AR-Modul (siehe Tabelle 8, Zeile 6). Dadurch sinkt der MAPE minimal auf 3,20% an, die Log-Likelihood (Log-L.) fällt allerdings auf 3.035 und das AIC steigt dementsprechend auf -6.048 an. Der Einbezug weiterer Lags hat das Modell also insgesamt eher verschlechtert und die (minimale) Verbesserung des MAPE ist zufälligerweise entstanden.<sup>361</sup> Beim ARIMA([1,2,4],0,2)(2,0,2)7 wird ein ähnlicher Ansatz verfolgt, allerdings wird nur das nicht-saisonale AR-Modul um ein signifikant partiell-autokorrigiertes Lag erweitert (siehe Tabelle 8, Zeile 8). Dadurch sinkt der MAPE zwar auf 3,17%, gleichzeitig fällt die Log-Likelihood auf 3.031 und das AIC steigt minimal auf -6.043. In der Abbildung 77 können die Koeffizienten abgelesen werden. In der Spalte „P>|Z|“ ist erkennbar, dass viele Koeffizienten eher zufällige Zusammenhänge abzubilden scheinen. Die Verbesserung beim MAPE ist daher auf eine Überanpassung an die Daten zurückzuführen.

Abbildung 77: ARIMA([1,2,4],0,2)(2,0,2)7

	coef	std err	z	P> z	[0.025	0.975]
ar.L1	0.4958	0.226	2.192	0.028	0.053	0.939
ar.L2	0.3562	0.141	2.530	0.011	0.080	0.632
ar.L4	0.0416	0.056	0.743	0.457	-0.068	0.151
ma.L1	0.2061	0.226	0.911	0.362	-0.237	0.649
ma.L2	-0.2821	0.026	-10.778	0.000	-0.333	-0.231
ar.S.L7	0.0082	0.008	1.071	0.284	-0.007	0.023
ar.S.L14	0.9918	0.008	129.803	0.000	0.977	1.007
ma.S.L7	0.0063	0.009	0.708	0.479	-0.011	0.024
ma.S.L14	-0.9892	0.009	-108.617	0.000	-1.007	-0.971

Quelle: Eigene Darstellung<sup>362</sup>

Ein ARIMA(4,0,4)(4,0,4)7 verwendet umfangreich erweiterte Module der vierten Ordnung (siehe Tabelle 8, Zeile 5). Der MAPE fällt dadurch auf 3,19%, die Log-Likelihood steigt auf 3.052 und auch das AIC fällt auf -6.070 ab. Es scheint sich also zunächst um ein etwas besseres Modell als ARIMA(2,0,2)(2,0,2)7 zu handeln. Mit insgesamt 16 Koeffizienten handelt es sich allerdings um ein sehr umfangreiches ARIMA-Modell, welches wie in Kapitel 2.4.5 beschrieben tendenziell eher zu Überanpassungen neigt. Die Übersicht über die Koeffizienten im Modell deutet ebenfalls

<sup>361</sup> 4-Modeling/02a-ARIMA (endogen) (ARIMA([1,2,6],0,2)(3,0,2)7)

<sup>362</sup> 4-Modeling/02a-ARIMA (endogen) (ARIMA([1,2,4],0,2)(2,0,2)7)

## Erstellung des Vorhersagemodells

auf eine Überanpassung hin. Von den insgesamt 16 Koeffizienten sind lediglich sechs signifikant ausgeprägt. Die über die übrigen Koeffizienten generalisierten Informationen sind aufgrund der hohen p-Werte als zufällig zu betrachten. Es handelt sich hier also um ein an die Daten überangepasstes Modell.

**Abbildung 78: ARIMA(4,0,4)(4,0,4)7**

	coef	std err	z	P> z	[0.025	0.975]
ar.L1	1.1421	1.041	1.097	0.272	-0.898	3.182
ar.L2	-0.3893	0.923	-0.422	0.673	-2.198	1.420
ar.L3	0.0967	0.540	0.179	0.858	-0.961	1.154
ar.L4	0.0956	0.157	0.608	0.543	-0.212	0.404
ma.L1	-0.4393	1.042	-0.422	0.673	-2.481	1.603
ma.L2	-0.0008	0.477	-0.002	0.999	-0.935	0.933
ma.L3	0.1254	0.472	0.266	0.790	-0.799	1.050
ma.L4	-0.1285	0.129	-0.995	0.320	-0.382	0.125
ar.S.L7	-0.9954	0.107	-9.278	0.000	-1.206	-0.785
ar.S.L14	0.1197	0.077	1.557	0.119	-0.031	0.270
ar.S.L21	1.0622	0.074	14.377	0.000	0.917	1.207
ar.S.L28	0.8134	0.109	7.456	0.000	0.600	1.027
ma.S.L7	1.0490	0.096	10.946	0.000	0.861	1.237
ma.S.L14	-0.0797	0.073	-1.084	0.278	-0.224	0.064
ma.S.L21	-1.0623	0.069	-15.340	0.000	-1.198	-0.927
ma.S.L28	-0.8268	0.096	-8.625	0.000	-1.015	-0.639

Quelle: Eigene Darstellung<sup>363</sup>

Teilweise führen umfangreichere Module aber auch zu Verschlechterungen. Ein ARIMA(3,0,3)(3,0,3)7 führt beispielsweise zu einem MAPE von 3,25%, die Log-Likelihood fällt auf 2.971 und das AIC steigt auf -5.930 (siehe Tabelle 8, Zeile 3).<sup>364</sup>

ARIMA(2,0,2)(2,0,2)7 lässt sich also durch zusätzliche Lags in den Modulen nicht verbessern. Andererseits lassen sich die Module auch nicht verkleinern, da die Qualität des Modells dann abnimmt. Ein ARIMA(1,0,1)(1,0,1)7 erreicht beispielsweise nur einen MAPE von 3,28%, die Log-Likelihood fällt im Vergleich zu ARIMA(2,0,2)(2,0,2)7 auf 2.979 ab, auch das AIC steigt auf -5.947 stark an (siehe Tabelle 8, Zeile 9).<sup>365</sup> Dieser Effekt wird beim Abschalten einzelner Module noch stärker sichtbar. Beim ARIMA(0,0,2)(0,0,2)7, also einem reinen MA-Modell, fällt der MAPE auf 8,94% und die Log-Likelihood auf 1.148. Aufgrund dessen steigt das AIC

<sup>363</sup> 4-Modeling/02a-ARIMA (endogen) (ARIMA(4,0,4)(4,0,4)7)

<sup>364</sup> 4-Modeling/02a-ARIMA (endogen) (ARIMA(3,0,3)(3,0,3)7)

<sup>365</sup> 4-Modeling/02a-ARIMA (endogen) (ARIMA(1,0,1)(1,0,1)7)

trotz des wesentlich weniger umfangreichen Modells auf -2.286 sehr stark an (siehe Tabelle 8, Zeile 13).<sup>366</sup>

Es zeigt sich außerdem, dass die saisonalen und nicht-saisonalen Module ähnlich bedeutsam für das Modell sind. Beim ARIMA(0,0,0)(2,0,2)7 werden beispielsweise die nicht-saisonalen Module p und q ausgestellt. Der MAPE fällt mit 6,02% sehr hoch aus, auch die Log-Likelihood von 1.965 und das AIC von -3.920 fallen ähnlich schlecht aus (siehe Tabelle 8, Zeile 14).<sup>367</sup> Umgekehrt werden bei einem ARIMA(2,0,5)(0,0,0)7 die saisonalen Module P und Q deaktiviert. Hier wird ein MAPE von 8,59% erreicht, auch eine Log-Likelihood von lediglich 1.358 und ein ACI von -2.699 sind vergleichsweise schlechter (siehe Tabelle 8, Zeile 15).<sup>368</sup>

Insgesamt zeigt sich also, dass ein ARIMA(2,0,2)(2,0,2)7 die Daten am besten abbilden kann. Wenn weniger Lags in die Module einbezogen werden beziehungsweise die Ordnung verringert wird, verschlechtert sich das Modell aufgrund von Unteranpassungen und mangelnder Fähigkeit zur Generalisierung. Umgekehrt kann das Modell durch höhere Ordnungen beziehungsweise umfangreichere Module mit mehr Lags nicht ausreichend verbessert werden. Die Qualität nimmt bei Verwendung zusätzlicher Lags kaum zu und die Modelle neigen zu starken Überanpassungen. Dies ist ein starker Hinweis darauf, dass die in den Daten enthaltenen, generalisierbaren Informationen vom ARIMA-Algorithmus erkannt werden und das vorhandene Potenzial ausgeschöpft ist. Es zeigt sich außerdem, dass sowohl saisonale und nicht-saisonale Module benötigt werden.

Die Abbildung 79 zeigt die Residuen des ARIMA(2,0,2)(2,0,2)7 auf den Testdaten. Die akzeptierte Abweichung von bis zu ~2,2% ist nach oben und unten in grau eingezzeichnet. Während das Modell wie in schon Abbildung 74 gezeigt die wöchentliche und jährliche Saisonalität erfassen kann, gibt es stellenweise starke Abweichungen. Gegen Ende Dezember und Anfang Januar sind die Abweichungen besonders stark.

**Abbildung 79: Residuen von ARIMA(2,0,2)(2,0,2)7**

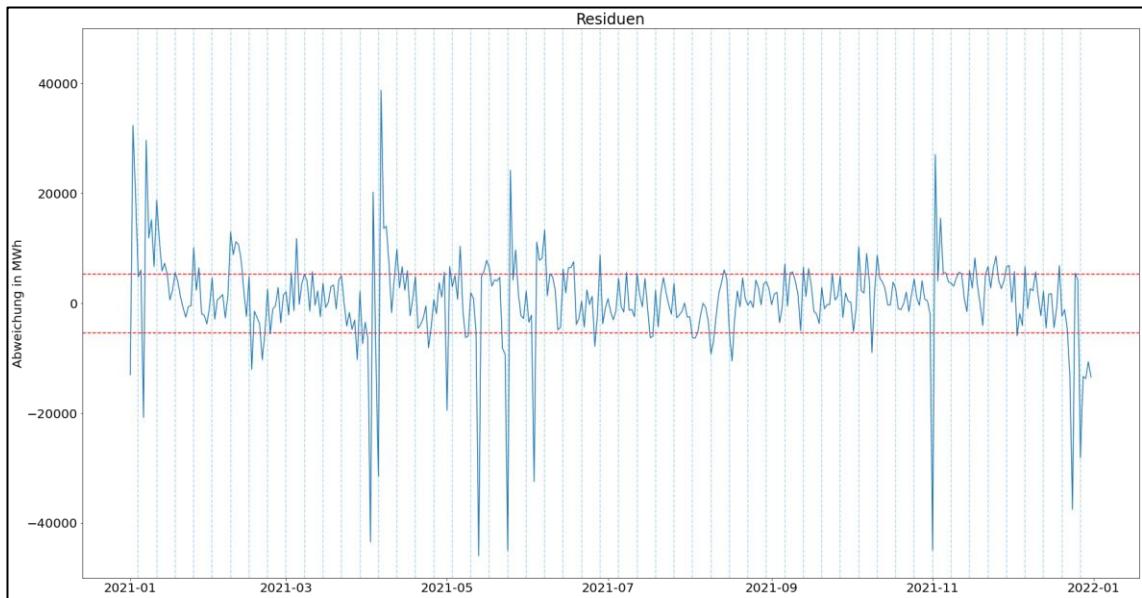
---

<sup>366</sup> 4-Modeling/02a-ARIMA (endogen) (ARIMA(0,0,2)(0,0,2)7)

<sup>367</sup> 4-Modeling/02a-ARIMA (endogen) (ARIMA(0,0,0)(2,0,2)7)

<sup>368</sup> 4-Modeling/02a-ARIMA (endogen) (ARIMA(2,0,5)(0,0,0)7)

## Erstellung des Vorhersagemodells



Quelle: Eigene Darstellung<sup>369</sup>

### 3.5.3 Erweiterung durch exogene Daten<sup>370</sup>

Das im ersten Schritt entwickelte ARIMA(2,0,2)(2,0,2)7 kann durch Einbezug der exogenen Daten verbessert werden. Die Tabelle 9 zeigt verschiedene, ausgewählte Ordnungen und deren Ergebnisse. Die jeweils verwendeten exogenen Daten sind in den [eckigen Klammern] eingetragen.

Wenn der Indikator für Arbeitstage hinzugezogen wird, verbessert sich der MAPE des Modells auf 2,25%. Die Log-Likelihood erreicht 4.342, das AIC erreicht -8.665. Das Modell wird also durch die exogene Variable erheblich verbessert (siehe Tabelle 9, Zeile 1).<sup>371</sup> Wird zusätzlich noch die Temperatur einbezogen, lässt sich der MAPE auf 2,22% steigern, auch fallen Log-Likelihood mit 4.353 und das AIC mit -8.684 etwas besser aus (siehe Tabelle 9, Zeile 2).<sup>372</sup>

<sup>369</sup> [4-Modeling/02a-ARIMA \(endogen\) \(Ergebnis\)](#)

<sup>370</sup> [4-Modeling/02b-ARIMA \(exogen\)](#)

<sup>371</sup> [4-Modeling/02b-ARIMA \(exogen\) \(Arbeitstag\)](#)

<sup>372</sup> [4-Modeling/02b-ARIMA \(exogen\) \(Arbeitstag, Temperatur\)](#)

## Erstellung des Vorhersagemodells

**Tabelle 9: ARIMA-Modelle mit exogenen Daten**

Modell		MAPE	Log-L.	AIC	Kommentar
1	ARIMA(2,0,2)(2,0,2)7 [Arbeitstag]	2,25%	4.342	-8.665	Mit Arbeitstag
2	ARIMA(2,0,2)(2,0,2)7 [Arbeitstag, Temperatur]	2,22%	4.353	-8.684	Mit Arbeitstag und Temperatur
4	ARIMA(2,0,2)(2,0,2)7 [Arbeitstag, Temperatur, Tagesstunden]	2,26%	4.343	-8.662	Mit Arbeitstag, Temperatur und Tagesstunden
5	ARIMA(3,0,3)(3,0,3)7 [Arbeitstag, Temperatur, Tagesstunden]	2,25%	4.333	-8.634	Dritte Ordnungen mit Tages- stunden
	ARIMA(3,0,3)(3,0,3)7 [Arbeitstag, Temperatur]	2,23%	4.346	-8.661	Dritte Ordnungen ohne Tages- stunden
6	ARIMA(4,0,4)(4,0,4)7 [Arbeitstag, Temperatur, Tagesstunden]	2,23%	4.360	-8.680	Vierte Ordnungen mit Tages- stunden
	ARIMA(4,0,4)(4,0,4)7 [Arbeitstag, Temperatur]	2,24%	4.350	-8.661	Vierte Ordnungen ohne Tages- stunden
7	ARIMA([1,2,4],0,2)(2,0,2)7 [Arbeitstag, Temperatur, Tagesstunden]	2,27%	4.321	-8.616	Signifikante Lags mit Tagesstun- den
	ARIMA([1,2,4],0,2)(2,0,2)7 [Arbeitstag, Temperatur]	2,21%	4.328	-8.631	Signifikante Lags ohne Tages- stunden
8	ARIMA(1,0,1)(1,0,1)7 [Arbeitstag, Temperatur, Tagesstunden]	2,38%	4.220	-8.425	Erste Ordnungen mit Tagesstun- den
	ARIMA(1,0,1)(1,0,1)7 [Arbeitstag, Temperatur]	2,31%	4.235	-8.426	Erste Ordnungen ohne Tages- stunden

Quelle: Eigene Darstellung<sup>373</sup>

Durch Einbezug der Tagesstunden steigt der MAPE wieder auf 2,26%, auch fallen Log-Likelihood mit 4.343 und AIC mit -8.662 wieder etwas schlechter aus (siehe Tabelle 9, Zeile 3). Die Abbildung 80 zeigt die Koeffizienten des Modells. Es zeigt sich, dass Arbeitstag und Temperatur signifikant sind, die Tagesstunden hingegen liegen mit einem p-Wert von 0,081 etwas über dem üblichen Signifikanzniveau von 0,05. Das Modell kann also nur stark eingeschränkt Informationen aus den Tagesstunden generalisieren.<sup>374</sup>

<sup>373</sup> [4-Modeling/02b-ARIMA \(exogen\)](#)

<sup>374</sup> [4-Modeling/02b-ARIMA \(exogen\) \(Arbeitstag, Temperatur, Tagesstunden\)](#)

## Erstellung des Vorhersagemodells

Abbildung 80: ARIMA(2,0,2)(2,0,2) [Arbeitstag, Temperatur, Tagesstunden]

	coef	std err	z	P> z	[0.025	0.975]
arbeitstag	0.3027	0.002	124.670	0.000	0.298	0.307
temperatur	-0.0528	0.014	-3.817	0.000	-0.080	-0.026
tagesstunden	-0.0781	0.045	-1.746	0.081	-0.166	0.010
ar.L1	0.6820	0.119	5.738	0.000	0.449	0.915
ar.L2	0.2630	0.112	2.353	0.019	0.044	0.482
ma.L1	0.0644	0.117	0.551	0.582	-0.165	0.293
ma.L2	-0.1532	0.024	-6.365	0.000	-0.200	-0.106
ar.S.L7	0.1063	0.034	3.135	0.002	0.040	0.173
ar.S.L14	0.8878	0.033	26.569	0.000	0.822	0.953
ma.S.L7	-0.0027	0.028	-0.096	0.924	-0.057	0.052
ma.S.L14	-0.8778	0.023	-38.047	0.000	-0.923	-0.833

Quelle: Eigene Darstellung<sup>375</sup>

Wie der Tabelle 9 und dem Notebook<sup>376</sup> zu entnehmen ist, zeigt sich dieser Effekt auch bei anderen Ordnungen. Ein ARIMA([1,2,4],0,2)(2,0,2)7 [Arbeitstag, Temperatur, Tagesstunden] erreicht einen MAPE von 2,27%, eine Log-Likelihood von 4.321 und ein AIC von -8.616. Werden die Tagesstunden aus dem Modell entfernt, verbessert sich der MAPE auf 2,21%, die Log-Likelihood auf 4.328 und auch das AIC auf -8.631 (siehe Tabelle 9, Zeile 7).<sup>377</sup> Ein ARIMA(3,0,3)(3,0,3)7 [Arbeitstag, Temperatur] verschlechtert sich durch Hinzuziehen der Tagesstunden beim MAPE von 2,23% auf 2,25%, bei der Log-Likelihood von 4.346 auf 4.333 und beim AIC von -8.661 auf 8.634 (siehe Tabelle 9, Zeile 5). Auch überschreitet der p-Wert für den Koeffizienten der Tagesstunden mit etwa 0,392 das Signifikanzniveau von 0,05 deutlich.<sup>378</sup> Während also die Temperatur und der Indikator für Arbeitstage das Modell verbessert, wird es durch Einbezug der Tagesstunden leicht verschlechtert. Die Analyse der Tagesstunden im Kapitel 3.3.3 zeigt zwar, dass die Tagesstunden den Stromverbrauch beeinflussen, dieser Einfluss ist allerdings eher gering und mit der Temperatur teilweise scheinkorreliert. Der ARIMA-Algorithmus scheint aufgrund des geringen zusätzlichen Informationsgehalts des Merkmals tendenziell keine Informationen aus den Tagesstunden generalisieren zu können. Aus diesem Grund wird im weiteren Modeling auf die Tagesstunden als exogenes Merkmal verzichtet.

<sup>375</sup> 4-Modeling/02b-ARIMA (exogen) (Arbeitstag, Temperatur, Tagesstunden)

<sup>376</sup> 4-Modeling/02b-ARIMA (exogen) (Andere Ordnungen)

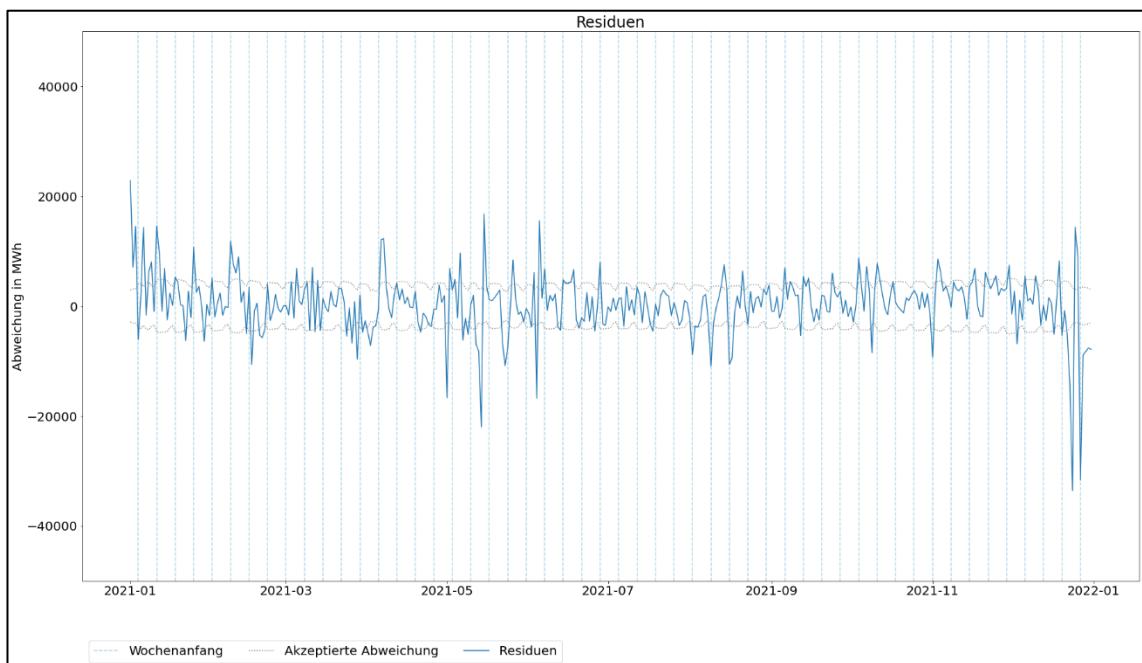
<sup>377</sup> 4-Modeling/02b-ARIMA (exogen) (ARIMA([1,2,4],0,2)(2,0,2)7)

<sup>378</sup> 4-Modeling/02b-ARIMA (exogen) (ARIMA(3,0,3)(3,0,3)7)

## Erstellung des Vorhersagemodells

Die Abbildung 81 zeigt die Residuen des ARIMA(2,0,2)(2,0,2)7 [Arbeitstag, Temperatur]. Es ist erkennbar, dass das Modell immer noch Ende Dezember und Anfang Januar sowie stellenweise im Jahresverlauf besonders große Abweichungen hat.

**Abbildung 81: Residuen von ARIMA(2,0,2)(2,0,2) [Arbeitstag, Temperatur]**



Quelle: Eigene Darstellung<sup>379</sup>

### 3.5.4 Verbesserungsansätze durch weitere exogene Daten<sup>380</sup>

Bei den Residuen des ARIMA(2,0,2)(2,0,2)7 [Arbeitstag, Temperatur] ist auffällig, dass es vor allem Ende Dezember und Anfang Januar sehr starke Abweichungen gibt (Abbildung 81). Es handelt sich dabei um die Zeit um die Weihnachtsfeiertage und die Tage unmittelbar danach. Wie in Kapitel 3.3.1 beschrieben folgt der Stromverbrauch in diesem Zeitraum zwar dem wöchentlichen Muster, ist aber aufgrund der Urlaubssaison und dem damit verbundenen geringeren Stromverbrauch im Industrie- und Handelssektor allgemein etwas niedriger.<sup>381</sup> Der ARIMA-Algorithmus kann diesen Effekt nicht aus den Daten generalisieren und weicht daher in diesem Zeitraum besonders stark ab.

<sup>379</sup> [4-Modeling/02b-ARIMA \(exogen\) \(Ergebnis\)](#)

<sup>380</sup> [4-Modeling/02c-ARIMA \(Verbesserung\)](#):

[4-Modeling/02d-ARIMA \(Feiertage\)](#)

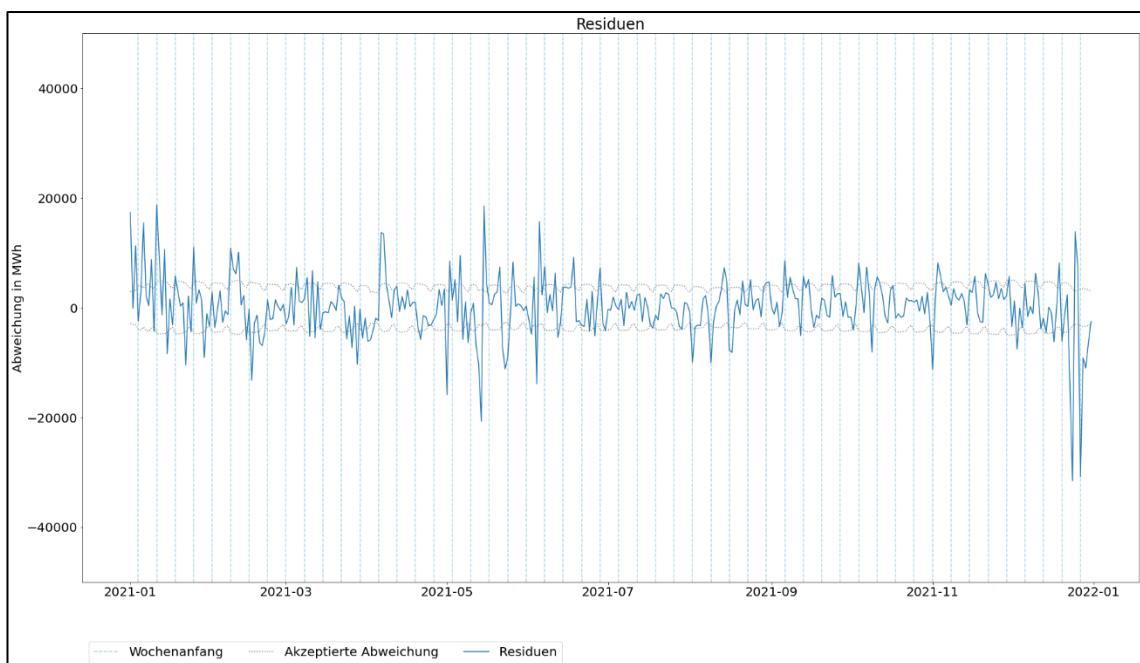
<sup>381</sup> Vgl. *Bundesverband der Energie- und Wasserwirtschaft*, Stromverbrauch im Dezember, 2020.

## Erstellung des Vorhersagemodells

Durch einen Rücksprung in die Data Preparation kann den exogenen Daten ein zusätzliches Merkmal hinzugefügt werden. Es handelt sich dabei um einen Indikator für die Urlaubssaison. Das Merkmal wird in der Woche vor und nach Weihnachten auf 1 und an allen übrigen Tagen auf 0 gesetzt. Beim Modeling wird das Merkmal zusätzlich zum Indikator für Arbeitstage und der Temperatur als drittes exogenes Merkmal an ARIMA(2,0,2)(2,0,2)7 übergeben.

Das damit trainierte Modell wird allerdings nicht verbessert. Der MAPE ist mit 2,24% etwas höher, die Log-Likelihood fällt auf 4.334, aufgrund der höheren Komplexität steigt das AIC auf -8.644 an. Beim Betrachten der Residuen in Abbildung 82 ist erkennbar, dass das neue Merkmal kaum Einfluss auf die Urlaubssaison um die Weihnachtsfeiertage hat.

**Abbildung 82: Residuen von ARIMA(2,0,2)(2,0,2)7 [Arbeitstag, Temperatur, Urlaubssaison]**



Quelle: Eigene Darstellung<sup>382</sup>

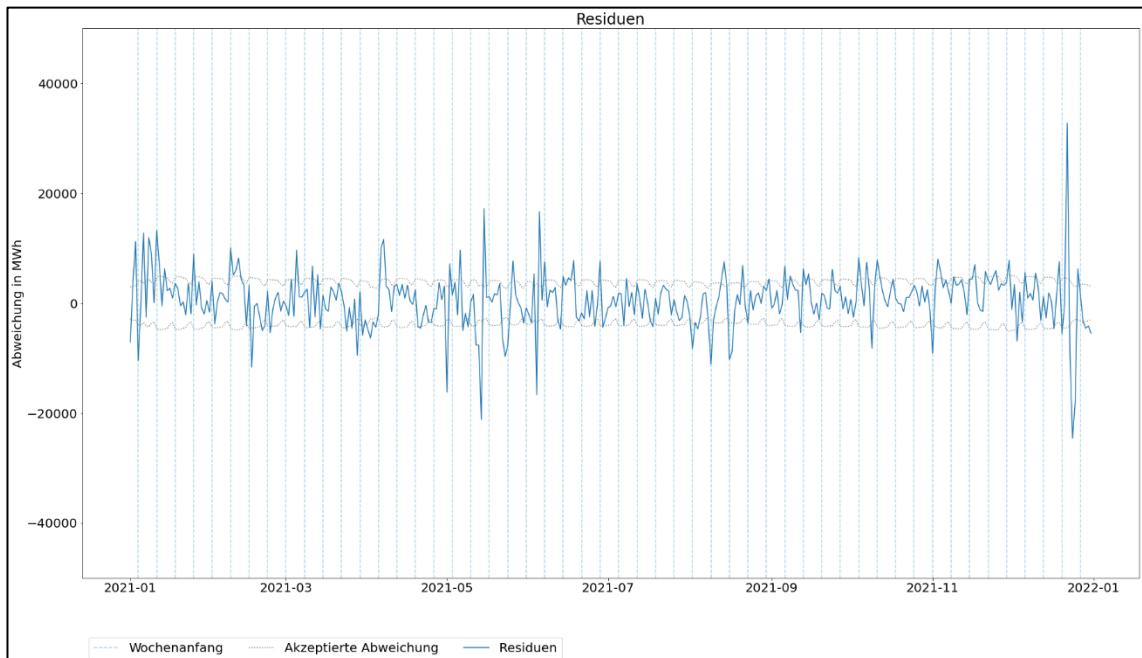
Wird stattdessen der Indikator für Arbeitstage im gleichen Zeitraum (Woche vor und nach Weihnachtsfeiertagen) jeden Tag auf 0 gesetzt, wodurch diese Tage als arbeitsfreie Tage verarbeitet werden, verbessert sich das Modell erheblich. Der MAPE lässt sich dadurch auf 2,1% senken. Die Log-Likelihood steigt auf 4.557 und das AIC ist mit -9.092 sehr viel niedriger. An den Residuen in Abbildung 83 erkennt man, dass

<sup>382</sup> [4-Modeling/02c-ARIMA \(Verbesserung\) \(Urlaubssaison\)](#)

## Erstellung des Vorhersagemodells

die Werte im Zeitraum um Weihnachten etwas besser abgebildet werden können. Allerdings bleiben die Abweichungen an den Weihnachtsfeiertagen selbst besonders hoch. Insgesamt wird das Modell trotzdem verbessert.<sup>383</sup> Genaueres ist dem Notebook zu entnehmen.<sup>384</sup>

**Abbildung 83: Residuen von ARIMA(2,0,2)(2,0,2) [Arbeitstag, Temperatur] (neu)**



Quelle: Eigene Darstellung<sup>385</sup>

In der Abbildung 83 ist erkennbar, dass es an bestimmten Stellen besonders große Abweichungen gibt. In der Abbildung 84 sind die Feiertage durch orangene Linien eingezzeichnet. Es ist zu erkennen, dass die Abweichungen an und besonders kurz nach Feiertagen entstehen. Feiertage selbst können bis auf einige Ausnahmen gut vorhergesagt werden.

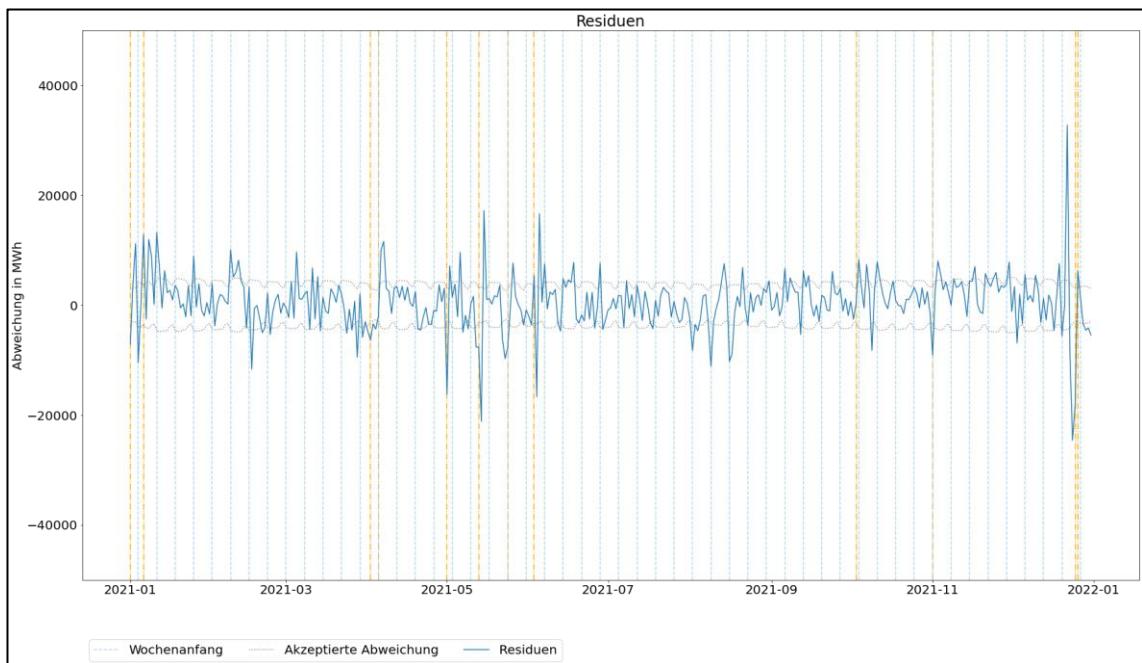
**Abbildung 84: Residuen um Feiertage herum**

<sup>383</sup> 4-Modeling/02c-ARIMA (Verbersserung) (Urlaubssaison als arbeitsfreie Tage)

<sup>384</sup> 4-Modeling/02c-ARIMA (Verbersserung)

<sup>385</sup> 4-Modeling/02c-ARIMA (Verbersserung) (Urlaubssaison als arbeitsfreie Tage)

## Erstellung des Vorhersagemodells



Quelle: Eigene Darstellung<sup>386</sup>

Da der niedrigere Stromverbrauch an Feiertagen einer gewissen Systematik folgt, handelt es sich nicht um Ausreißer im rein mathematischen Sinne. Dennoch durchbrechen sie die siebentägige Saisonalität und verzerrn das Modell aufgrund der Abweichung vom normalen Verlauf des Stromverbrauchs stark und sind daher nur schwer im Modell abzubilden. Zusätzlich führen Feiertage zu einer Verschlechterung der Vorhersagen für die Tage unmittelbar danach. Die nicht-saisonalen AR- und MA-Module haben einen sehr starken Einfluss auf die Vorhersage und verwenden jeweils die letzten zwei Tage vor der vorherzusagenden Beobachtung als Eingabeparameter. Wenn ein Feiertag mit außergewöhnlich niedrigem Verbrauch als Eingangs-wert an die Module übergeben wird, dann wird die Vorhersage für den nächsten Tag dadurch stark verzerrt. Hieraus ergeben sich die starken Abweichungen unmittelbar nach Feiertagen.<sup>387</sup>

Derartige Probleme könnten kompensiert werden, beispielsweise indem zusätzliche exogene Merkmale in das Modell integriert werden. Es können zum Beispiel nach demselben Prinzip wie bei den Arbeitstagen weitere Indikatoren hinzugefügt werden, etwa ein Indikator für Feiertage oder ein Indikator dafür, dass der vorherige oder vorvorherige Tag ein Feiertag war. Im Modell können dann Koeffizienten für die

<sup>386</sup> 4-Modeling/4d-ARIMA (Feiertage) (Abweichung an Feiertagen)

<sup>387</sup> 4-Modeling/4d-ARIMA (Feiertage) (Abweichung an Feiertagen)

jeweiligen Indikatoren angelernt werden, um die oben beschriebenen Verzerrungen an Feiertagen und den Tagen danach auszugleichen. Diese Ausgleichskoeffizienten bringen aber wiederum zwei Probleme mit sich.

Zunächst tritt das Problem nicht bei jedem Feiertag auf. Folgt auf den Feiertag beispielsweise ein Samstag oder Sonntag kommt es gegebenenfalls zu geringeren oder gar keinen Verzerrungen. In diesen Fällen würden die Ausgleichskoeffizienten für Feiertage selbst Verzerrungen verursachen, welche durch weitere Ausgleichskoeffizienten kompensiert werden müssten. Es werden also gegebenenfalls zahlreiche weitere Merkmale benötigt. Indikator-Merkmale sind vergleichbar mit konditionalen Konstanten: Steht der Indikator auf eins, wird der Koeffizient in der Regressionsgleichung addiert. Steht der Indikator auf null, hat der Term keinen Einfluss auf die Regressionsgleichung. Der Ansatz, alle möglichen Ausreißer und Sonderfälle durch eigene (konditionale) Merkmale und Koeffizienten einzufangen, widerspricht allerdings dem Grundgedanken des maschinellen Lernens: der Generalisierung von Informationen aus Daten. Stattdessen handelt es sich eher um eine durch maschinelles Lernen unterstützte Verschachtelung von Bedingungen.

Trotzdem kann der Ansatz aus experimentellen Gründen wie im Notebook einsehbar<sup>388</sup> testweise verfolgt werden. Hier zeigt sich ein weiteres Problem: Die Verzerrung durch Feiertage passiert vier bis fünf Mal pro Jahr, ist also insgesamt etwa 30 Mal in den Daten enthalten. Bei insgesamt 2.557 Datensätzen ist dies verschwindend selten. Der ARIMA-Algorithmus erhält zu wenig Beispieldaten, um diese Effekte gut zu erkennen (der erste Ansatz zur Verbesserung des Modells durch einen Indikator für die Urlaubssaison am Anfang des Kapitels scheitert möglicherweise ebenfalls daran). Aus diesem Grund führt die Integration der oben beschriebenen Indikatoren auch nicht zum gewünschten Ergebnis. Zwar lassen sich Log-Likelihood und AIC etwas verbessern, am MAPE und dem grundsätzlichen Problem ändert sich durch die Indikatoren allerdings wenig.<sup>389</sup> Ein typischer Ansatz zur Lösung des Mangels an Beispieldaten wäre üblicherweise das sogenannte Oversampling<sup>390</sup>, bei dem unter anderem künstliche Datensätze zu den Trainingsdaten hinzugefügt werden, um der Unterrepräsentierung der entsprechenden Datensätze entgegenzuwirken.

---

<sup>388</sup> [4-Modeling/2d-ARIMA \(Feiertage\)](#)

<sup>389</sup> [4-Modeling/2d-ARIMA \(Feiertage\) \(Merkmal für Feiertage\)](#)

<sup>390</sup> Vgl. Brownlee, J., Oversampling, 2021.

Der Zeitreihe können jedoch nicht einfach bestimmte Datensätze hinzugefügt werden, da sich ansonsten die Beschaffenheit der Zeitreihe (mit Autokorrelationen, Saisonalitäten etc.) und die Zeitabhängigkeit der Beobachtungen sehr schnell zu stark verändert kann.<sup>391</sup> Es gibt zwar mittlerweile Ansätze für entsprechendes Oversampling im Zusammenspiel mit ARIMA oder Zeitreihen allgemein<sup>392</sup>, diese erfordern aber ihrerseits eine eigene Untersuchung und sollen daher hier nicht weiter betrachtet werden.

### 3.5.5 Zwischenfazit zu ARIMA<sup>393</sup>

Es zeigt sich, dass ein ARIMA(2,0,2)(2,0,2)7 [Arbeitstag, Temperatur] den Stromverbrauch mit einer durchschnittlichen Abweichung von etwa 2,1% vorhersagen kann. Insgesamt kann das ARIMA-Modell die Baseline also um etwa 0,34 Prozentpunkte beziehungsweise fast 14% übertreffen. Das Erfolgskriterium eines MAPE von höchstens 2,2% wird damit erreicht. Die Abbildung 85 zeigt die Vorhersagen des ARIMA-Modells im Vergleich zur Baseline. Es ist auch hier zu erkennen, dass die Vorhersagen von ARIMA (grün) die meiste Zeit über etwas genauer an den tatsächlichen Werten (blau) sind als die Baseline (grün). Die Baseline hat ebenfalls größere Abweichungen an Feiertagen und zur Weihnachtszeit, ist hier allerdings etwas besser als ARIMA. Die Schwäche bei Feiertagen ist also definitiv eine Restriktion des ansonsten genaueren ARIMA-Modells.

**Abbildung 85: ARIMA und Baseline**

---

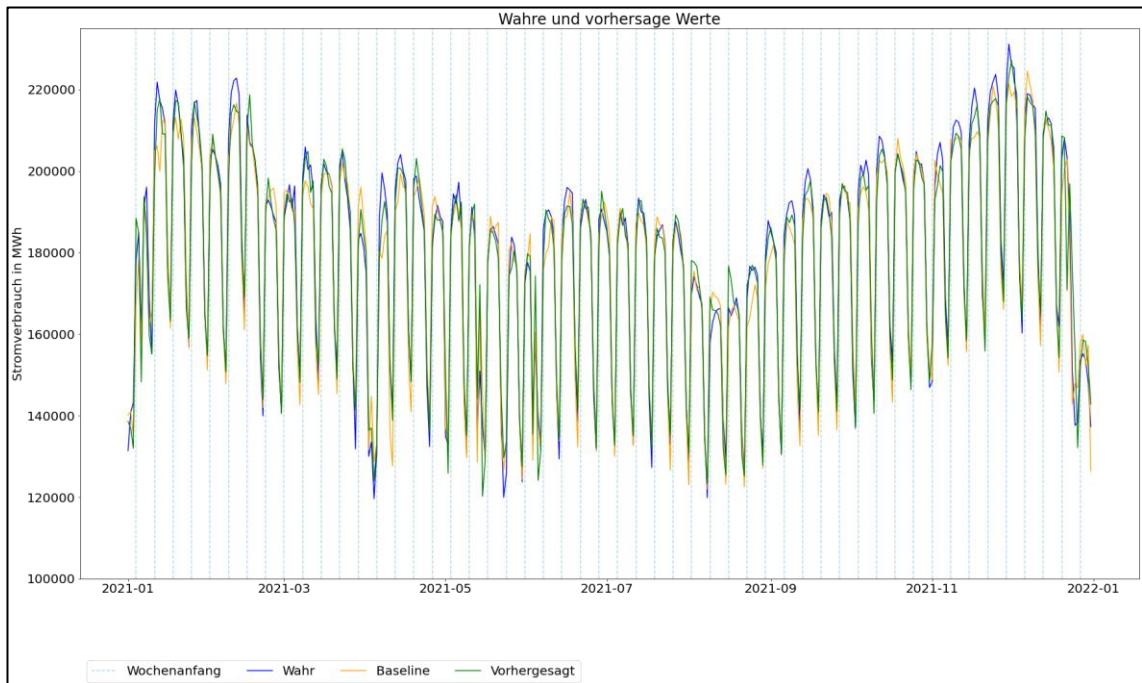
<sup>391</sup> Vgl. Moniz, N., Branco, P., Torgo, L., Oversampling in Forecasting, 2017.

<sup>392</sup> Vgl. Chen, G., Guo, X., Oversampling ARIMA, 2021;

Vgl. Moniz, N., Branco, P., Torgo, L., Oversampling in Forecasting, 2017.

<sup>393</sup> [4-Modeling/02e-ARIMA \(Zwischenfazit\)](#)

## Erstellung des Vorhersagemodells



Quelle: Eigene Darstellung<sup>394</sup>

Die Modellierung mittels des ARIMA-Algorithmus wird daher als erfolgreich abgeschlossen.

### 3.6 Modeling (LSTM)<sup>395</sup>

Bei der Entwicklung neuronaler Netze gilt es, eine möglichst optimale Architektur (Schichten, Neuronen etc.) zu finden, welche dann möglichst effektiv trainiert wird (Anzahl Epochen, Größe der Batches etc.). Aufgrund der vielen unterschiedlichen (Hyper-)Parameter ist es unrealistisch, sämtliche möglichen Kombinationen der Parameter auszuprobieren. Aus diesem Grund wird versucht, das neuronale Netz Schritt für Schritt näherungsweise zu optimieren, indem zunächst eine geeignete Architektur und anschließend ein entsprechendes Training ermittelt wird. Die Implementierung des neuronalen Netzes erfolgt mit den Bibliothek TensorFlow<sup>396</sup>.

Aufgrund der umfänglichen Entwicklung werden hier nur die Ansätze, die generelle Methodik bei der Entwicklung sowie die fertigen Ergebnisse dargestellt. Der gesamte Prozess der Entwicklung mit genaueren Erklärungen und Ausführungen kann den Notebooks entnommen werden. Die einzelnen trainierten Modelle werden jeweils mit

<sup>394</sup> 4-Modeling/02e-ARIMA (Zwischenfazit) (Baseline)

<sup>395</sup> 4-Modeling

<sup>396</sup> TensorFlow.org, TensorFlow, 2022.

ihren Schichten und einer laufenden Nummer benannt (bspw. LSTM-1 oder LSTM-DENSE-3).

Bei der Entwicklung neuronaler Netze wird häufig ein Validierungsdatensatz aus den Trainingsdaten entnommen. Die Validierungsdaten werden dann nicht in das Training einbezogen, dienen allerdings nach jeder Epoche zum Test des Modells, um die Effektivität beziehungsweise die Veränderung nach der Epoche zu bewerten.<sup>397</sup> Da die Datengrundlage insgesamt mit 2.557 Beobachtungen relativ klein ist, findet das Training außerdem mit 5-facher Kreuzvalidierung statt. Dabei werden die Trainingsdaten in fünf Teildatensätze aufgeteilt, wobei stets vier für das Training und einer für die Validierung verwendet wird. Dadurch wird vermieden, dass die Güte des Modells aufgrund günstiger oder ungünstiger Aufteilung oder Schwankungen der Trainings- und Validierungsdaten verzerrt wird.<sup>398</sup> Die abschließende Bewertung der Modelle findet auch während der Entwicklung anhand der Testdaten statt. Die Ergebnisse auf den Testdaten werden zur besseren Interpretierbarkeit zusätzlich mit denen aus der Kreuzvalidierung verglichen. Eine genauere Beschreibung der Implementierung ist im entsprechenden Notebook für die Entwicklung zu finden.<sup>399</sup>

Anders als bei ARIMA werden die exogenen Daten beim neuronalen Netz von Anfang an in das Modell einbezogen. Während sich endogene ARIMA-Modelle einfach durch exogene Daten erweitern lassen, ist bei neuronalen Netzen gegebenenfalls eine völlig andere Architektur notwendig, wenn weitere Daten in das Modell integriert werden. Daher ist eine einfache Erweiterung wie bei ARIMA nicht möglich. Insofern bringt eine rein endogene Entwicklung nur einen begrenzten Mehrwert. Aus experimentellen Gründen wurde dennoch ein neuronales Netz mit nur endogenen Daten entwickelt, welches im Notebook<sup>400</sup> einsehbar ist. Dabei handelt es sich aber ausdrücklich nicht um einen Teil dieser Untersuchung.

Theoretisch lassen sich neuronale Netze endlos durch zusätzliche Schichten und Neuronen erweitern. Je komplizierter und umfangreicher ein Netz wird, desto eher wird es anfällig für Überanpassungen, verschwindende/explodierende Gradienten etc.<sup>401</sup> Die Netze sollten nach Möglichkeit so einfach wie möglich gehalten werden,

---

<sup>397</sup> Vgl. Lazzeri, F., Machine Learning, 2021, S. 42 ff.

<sup>398</sup> Vgl. Raschka, S., Mirjalili, V., Machine Learning mit Python, 2021, S. 217 ff.

<sup>399</sup> [4-Modeling/03a-LSTM \(Entwicklung\)](#)

<sup>400</sup> [4-Modeling/Experimentelle Ansätze/01-LSTM \(endogen\)](#)

<sup>401</sup> Vgl. Aggarwal, C., Neural Networks, 2018, S. 30 ff. und 34 ff.

um auf ressourcenintensive Überkapazitäten im Netz und beim Training zu verzichten. Aus diesem Grund wird zunächst ein sehr simples Netz entwickelt, welches dann nach und nach erweitert wird. Als Verlustfunktion wird der Root Mean Squared Error (RMSE) gewählt. Wie in Kapitel 2.6.2 dargestellt, bestraft der RMSE größere Abweichungen stärker als kleinere Abweichungen. Im Anwendungsfall bedeutet das, dass Abweichungen über ~2,2% (also über der akzeptierten Abweichung) stärker bestraft werden als Abweichungen unter ~2,2% (unter der akzeptierten Abweichung).

Im Rahmen dieser Untersuchung sollen folgende Hyperparameter optimiert werden:

- Anzahl und Art von Schichten (samt Anzahl der Neuronen und Aktivierung)
- Anzahl der Epochen und Größe der Batches
- Lernrate
- Ggf. Regularisierung und Dropouts
- Länge des Zeitfensters (für LSTM-Schichten)

Die Autokorrelationsdiagramme in Abbildung 44 und Abbildung 45 zeigen, dass in den ersten zwei Wochen vergleichsweise viele signifikant autokorrelierte Lags liegen. Die Entwicklung des ARIMA(2,0,2)(2,0,2)7-Modells zeigt ebenfalls, dass sich dieser Zeitraum für die Modellierung gut eignet. Daher wird die Fenstergröße initial auf 14 festgelegt. Die Abbildung 86 zeigt ein Zeitfenster. Für die Vorhersage des Stromverbrauchs am 15.01.2015 werden die Stromverbrauchsdaten vom 01.01. bis zum 14.01. verwendet (grün markiert). Bei den exogenen Daten sind nicht die Daten der vorherigen Tage, sondern die Wetterdaten des aktuellen Tages relevant (rot markiert). Das Modell muss den Stromverbrauch des vorherigen Tages und die exogenen Daten des vorherzusagenden Tages zu dessen Stromverbrauch mappen. Das Zeitfenster zur Vorhersage des 15.01. fasst also die endogenen Daten vom 01.01. bis zum 14.01. und die (versetzen) exogenen Daten vom 02.01. bis zum 15.01.:

Abbildung 86: Zeitfenster für Vorhersage des 15.01.2015

## Erstellung des Vorhersagemodells

	datum	verbrauch	arbeitstag	temperatur	tagesstunden
1	2015-01-01	126197	0	-2.5	8.4
2	2015-01-02	147085	1	-0.0	8.4
3	2015-01-03	141426	0	1.2	8.4
4	2015-01-04	132446	0	-0.2	8.4
5	2015-01-05	152611	1	-0.5	8.4
6	2015-01-06	140474	0	0.6	8.4
7	2015-01-07	184936	1	0.9	8.5
8	2015-01-08	190395	1	4.2	8.5
9	2015-01-09	186902	1	5.8	8.5
10	2015-01-10	150311	0	10.4	8.6
11	2015-01-11	140494	0	2.6	8.6
12	2015-01-12	194573	1	3.2	8.6
13	2015-01-13	196391	1	4.9	8.6
14	2015-01-14	198853	1	4.9	8.7
15	2015-01-15	198241	1	5.5	8.7
16	2015-01-16	194680	1	5.3	8.8
17	2015-01-17	161724	0	2.5	8.8
18	2015-01-18	143315	0	0.9	8.8

Quelle: Eigene Darstellung

### 3.6.1 LSTM-Schichten<sup>402</sup>

Zunächst wird ein einfaches neuronales Netz erstellt. Die Eingabeschicht fasst vier Neuronen (eines je Merkmal) und erwartet jeweils 14 Werte (einen je Zeitschrift im Fenster). In der Ausgabeschicht gibt es lediglich ein Neuron ohne Aktivierungsfunktion, in dem die Vorhersage final erzeugt wird. Dazwischen liegt eine verdeckte LSTM-Schicht mit 32 Neuronen, welche mittels tanh-Funktion aktiviert werden. Als Lernrate wird 0,001 festgelegt. Das Training erfolgt zunächst über 50 Epochen mit Batches à 16 Datensätzen.<sup>403</sup>

Die Abbildung 87 zeigt die Lernkurve der Verlustfunktion für das Modell. Die durchschnittliche Verlustfunktion über alle Kreuzvalidierungen hinweg ist für die Trainingsdaten in blau und für die Validierungsdaten in rot eingezeichnet (die Funktionen der einzelnen Kreuzvalidierungen sind in gestrichelten Linien eingezeichnet). Die Verlustfunktion fällt in den ersten Epochen sehr stark, weitere Epochen tragen schon sehr früh wenig zur Reduktion der Verlustfunktion bei. Ab etwa der 20. bis 25. Epoche beginnt das Modell mit der sogenannten Konvergenz<sup>404</sup>. Ab diesem Punkt haben

<sup>402</sup> 4-Modeling/03b-LSTM (LSTM)

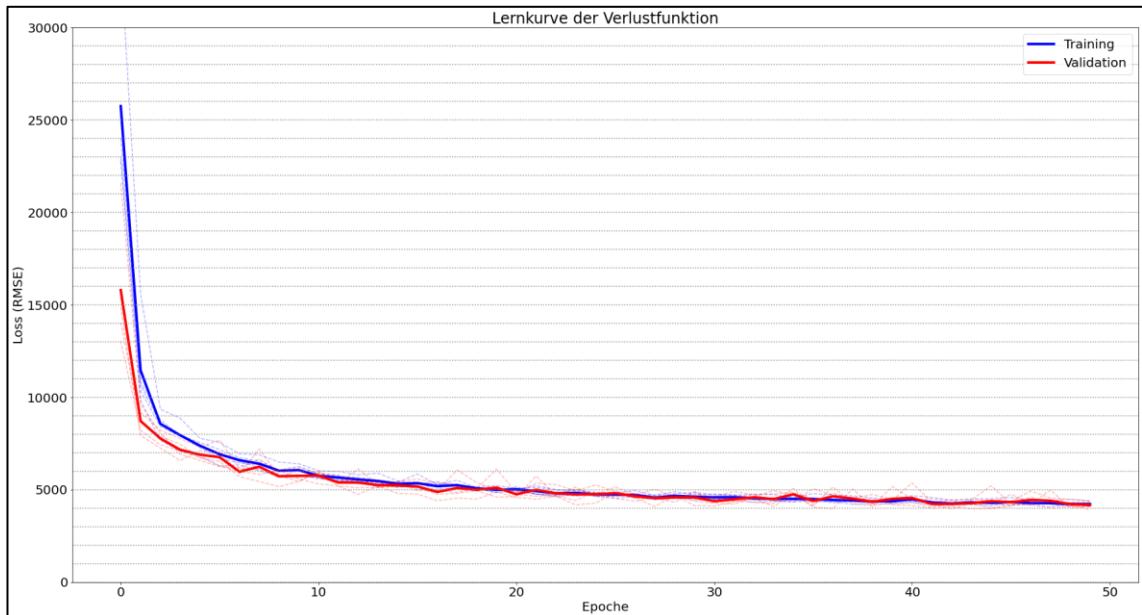
<sup>403</sup> 4-Modeling/03b-LSTM (LSTM) (LSTM-1)

<sup>404</sup> Vgl. Raschka, S., Mirjalili, V., Machine Learning mit Python, 2021, S. 447 ff.

## Erstellung des Vorhersagemodells

zusätzliche Epochen keinen erkennbaren zusätzlichen Nutzen und führen zu keinen Verbesserungen.

**Abbildung 87: Lernkurve von LSTM-1**



Quelle: Eigene Darstellung<sup>405</sup>

Das Modell erzielt bei den Testdaten mit einem MAPE von nur 2,39% bereits sehr gute Werte (1,89% auf Trainingsdaten). Das Netz kann offensichtlich sehr schnell die Beziehungen in den Daten anlernen, weswegen die Verlustfunktion mit den ersten Epochen bereits sehr stark fällt. Bemerkenswert ist auch, dass die Verlustfunktion der Validierungsdaten anders als erwartet am Anfang niedriger ist als die der Trainingsdaten. Dies ist ebenfalls ein starkes Indiz für die Schnelligkeit, mit der das Modell am Anfang lernt. Während die Verlustfunktion für die Validierungsdaten am Ende der Epoche berechnet wird, wird die Verlustfunktion der Trainingsdaten nach jedem einzelnen Batch (für die Backward-Propagation) berechnet. Das Modell lernt allerdings mit jedem Batch dazu. Datensätze, die in früheren Batches durch das Netz geleitet wurden, werden daher von einem etwas schlechter angelernten Netz bewertet als die Datensätze in späteren Batches. Die Validierungsdaten hingegen werden

<sup>405</sup> [4-Modeling/03b-LSTM \(LSTM\) \(LSTM-1\)](#)

erst am Ende der Epoche bewertet, wenn das Modell alle verfügbaren Trainingsdaten anlernen konnte.<sup>406</sup>

Ein derartiger Verlauf der Lernkurve kann auf eine zu hohe Lernrate hindeuten.<sup>407</sup> Daher werden in den nächsten Schritten niedrigere Lernraten ausprobiert. Bei einer Lernrate von 0,005 verläuft die Lernkurve zwar sehr ähnlich, allerdings verbessert sich das Modell hierdurch auf einen MAPE von 2,20% auf den Testdaten. Bei den Trainingsdaten fällt der MAPE auf 2,04%. Da sich MAPE bei Trainings- und Testdaten angenähert haben, scheint das Modell mit einer niedrigeren Lernrate besser generalisieren zu können.<sup>408</sup> Wird die Lernrate allerdings noch niedriger gewählt, dann führt dies zu einer Verschlechterung des Modells auf den Trainings- und Testdaten. Am Verlauf der Lernkurve ändert sich durch die niedrigere Lernrate nur bedingt etwas.<sup>409</sup> Das Modell wird also mit einer Lernrate von 0,005 weiter trainiert. Um zu überprüfen, ob das Modell aufgrund einer zu niedrigeren oder zu hohen Batchgröße Schwierigkeiten hat, das Minimum der Verlustfunktion zu erreichen, werden unterschiedliche Batchgrößen ausprobiert. Dadurch lässt sich das Modell allerdings nicht verbessern.<sup>410</sup> Die Lernrate und die Batchgröße sind also angemessen gewählt. Das Modell ist mit einer Schicht aus 32 Neuronen relativ simpel. Mit weiteren Neuronen (64 und 128) lassen sich aber nur bedingt Verbesserungen erzielen. Anhand der Verlustfunktion lassen sich bei den Modellen keine sonderlich starken Überanpassungen erkennen. Trotzdem können die Modelle die Trainingsdaten etwas besser erkennen als die Testdaten. Es scheint also zumindest eine leichte Überanpassung vorzuliegen, allerdings lässt sich diese nicht durch beispielsweise Regularisierung kompensieren, da bereits leichte Regularisierungen zu Unteranpassungen führen. Der MAPE auf den Testdaten bewegt sich durchweg um etwa 2,3 bis 2,4% herum. Es ist also davon auszugehen, dass das einschichtige Modell das Potenzial zur Generalisierung erschlossen hat und eine grundlegend andere Architektur nötig ist.<sup>411</sup>

Es wird daher versucht, das Modell durch eine zusätzliche Schicht zu verbessern, da Netze mit mehreren verdeckten Schichten häufig besser dazu in der Lage sind,

---

<sup>406</sup> Vgl. Ameisen, E., ML Applications, 2020, S. 166 ff.;  
Vgl. Rosebrock, A., Validation Loss, 2019.

<sup>407</sup> Vgl. Brownlee, J., Learning Rates, 2020.

<sup>408</sup> [4-Modeling/03b-LSTM \(LSTM\) \(LSTM-2\)](#)

<sup>409</sup> [4-Modeling/03b-LSTM \(LSTM\) \(LSTM-3\)](#)

<sup>410</sup> [4-Modeling/03b-LSTM \(LSTM\) \(LSTM-4\);](#)

[4-Modeling/03b-LSTM \(LSTM\) \(LSTM-5\)](#)

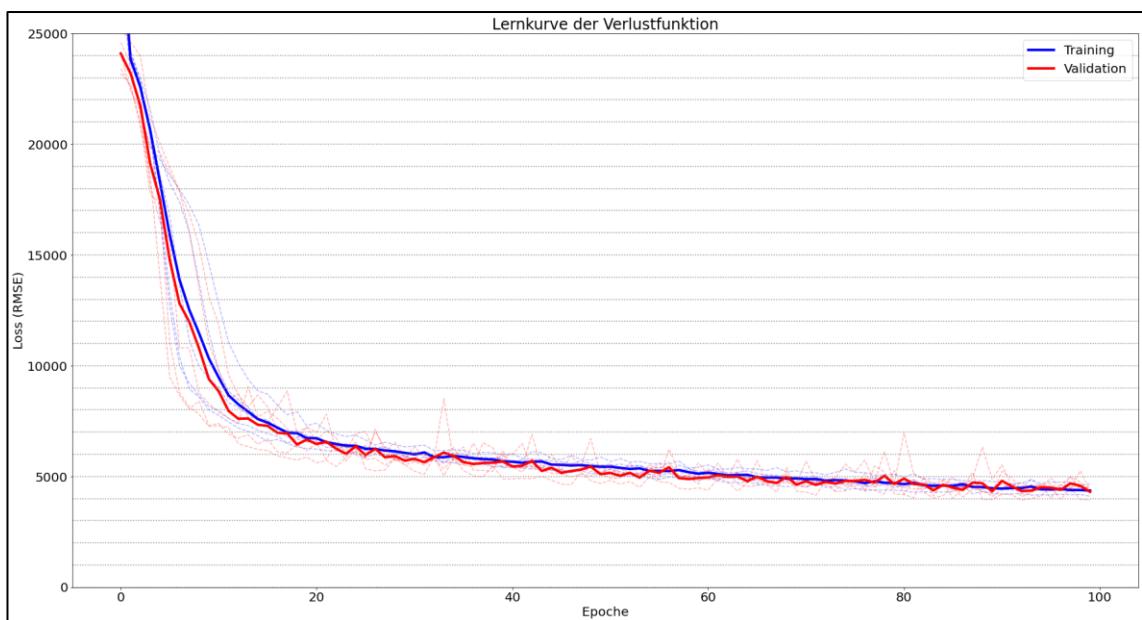
<sup>411</sup> [4-Modeling/03b-LSTM \(LSTM\)](#)

## Erstellung des Vorhersagemodells

auch nicht-lineare Zusammenhänge (wie etwa bei der Temperatur) zu erkennen.<sup>412</sup> Es wird daher eine weitere LSTM-Schicht zum Modell hinzugefügt. Um keine Flaschenhälse zu erzeugen, die Generalisierungen der vorherigen Schichten zunichtemachen können, sollte eine Schicht stets gleich viele oder mehr Neuronen haben als die vorherige (ausgenommen ist selbstverständlich die Ausgabeschicht).<sup>413</sup>

Die zusätzliche Schicht führt jedoch nur stark eingeschränkt zu einer Verbesserung. Die Netze verhalten sich ähnlich wie Netze mit nur einer Schicht. Die Abbildung 88 zeigt die Lernkurve eines Netzes aus zwei LSTM-Schichten à 32 und 64 Neuronen. Die Konvergenz des Modells beginnt zwar etwas später, allerdings ist auch hier keine Überanpassung zu erkennen. Auf Dropouts oder Regularisierungen kann also zunächst verzichtet werden.

**Abbildung 88: Lernkurve von LSTM-LSTM-3**



Quelle: Eigene Darstellung<sup>414</sup>

Veränderungen an der Lernrate oder der Batchgröße führen ebenfalls zu keinen Verbesserungen bei den Modellen, da hierdurch meist eine Unteranpassung entsteht.<sup>415</sup> Der MAPE bewegt sich bei den zweischichtigen LSTM-Modellen auf den Testdaten im Vergleich zu den einschichtigen Netzen auf einem etwas höheren Niveau

<sup>412</sup> Vgl. Aggarwal, C., Neural Networks, 2018, S. 76 ff.

<sup>413</sup> Vgl. Hirschle, J., Machine Learning für Zeitreihen, 2020, S. 226.

<sup>414</sup> [4-Modeling/03c-LSTM \(LSTM-LSTM\) \(LSTM-LSTM-3\)](#)

<sup>415</sup> [4-Modeling/03c-LSTM \(LSTM-LSTM\) \(LSTM-LSTM-5\)](#)

zwischen 2,4% und 2,6%.<sup>416</sup> Da zwei LSTM-Schichten mit einem Fenster von 14 Zeitschritten bereits 28 Zeitschichten ergeben, wird statt der tanh-Funktion auch testweise die ReLU-Funktion eingesetzt, da sie bei tieferen Netzen anders als tanh nicht so sehr zum Problem der verschwindenden Gradienten neigt.<sup>417</sup> Das Modell lässt sich allerdings auch nicht durch den Einsatz der ReLU-Funktion verbessern.<sup>418</sup>

Eine genauere Beschreibung kann den Notebooks<sup>419</sup> entnommen werden. Die Netze haben zusammengefasst zwei Gemeinsamkeiten: Erstens lernen sie sehr schnell in den frühen Epochen, spätere Epochen führen nur zu geringen Verbesserungen, bis die Konvergenz einsetzt. Vor allem bei den zweischichtigen Netzen sind sehr leichte Überanpassungen in den späten Epochen zu erkennen, welche sich aber nur eingeschränkt durch Regularisierung beheben lassen. Zweitens bewegt sich der MAPE auf den Testdaten durchgängig auf einem ähnlichen Level. Bei einschichtigen Netzen wird etwa 2,2% bis 2,4% erreicht (1,9% bei den Trainingsdaten), bei den zweischichtigen Netzen wird ein Wert um etwa 2,5% erreicht (1,7% bei den Trainingsdaten). Hier lässt sich zwar eine Überanpassung erkennen, es ist allerdings nicht ungewöhnlich, dass bekannte Trainingsdaten besser erkannt werden als unbekannte Testdaten (siehe auch Kapitel 2.4.5). Dieses Verhalten lässt sich in vielen Fällen auch nicht vollständig verhindern.<sup>420</sup> Das Netz könnte nun durch weitere Anpassungen am Training oder der Architektur verbessert werden, allerdings deutet sich an, dass das Potenzial eines reinen LSTM-Netzes bereits erschlossen ist. Es zeigt sich, dass Erweiterungen mit mehr Neuronen (oder Schichten etc.) nicht zu einer besseren Generalisierung führen. Es wird daher ein anderer Ansatz zur Verbesserung verfolgt.

### 3.6.2 LSTM- und Dense-Schichten<sup>421</sup>

Eine LSTM-Schicht erreicht bereits gute Ergebnisse. Es bietet sich nun an, die LSTM-Schicht durch eine (wie in Kapitel 2.5.1) beschriebene „einfache“ Schicht zu

<sup>416</sup> [4-Modeling/03c-LSTM \(LSTM-LSTM\)](#)

<sup>417</sup> Vgl. Patterson, J., Gibson, A., Deep Learning, 2017, S. 69 ff.;  
Vgl. Plaat, A., Reinforcement Learning, 2020, S. 149 ff.

<sup>418</sup> [4-Modeling/03c-LSTM \(LSTM-LSTM\) \(LSTM-LSTM-4\)](#)

<sup>419</sup> [4-Modeling/03b-LSTM \(LSTM\);](#)  
[4-Modeling/03c-LSTM \(LSTM-LSTM\)](#)

<sup>420</sup> Vgl. Raschka, S., Mirjalili, V., Machine Learning mit Python, 2021, passim, insb. S. 39 ff.;  
Vgl. Géron, A., Machine Learning, 2020, passim., insb. S. 24 ff.

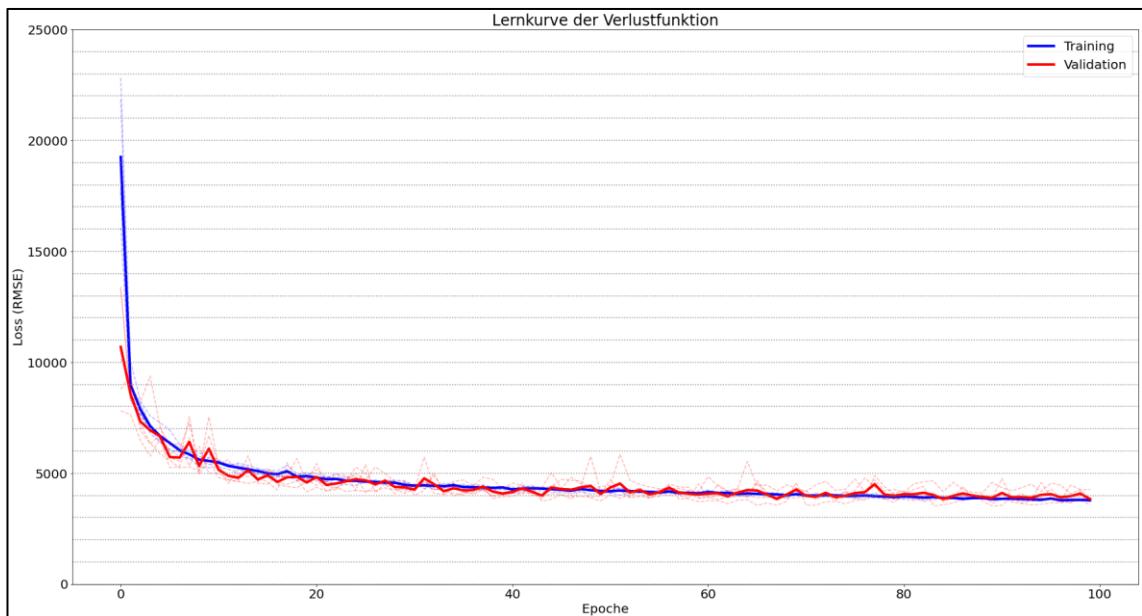
<sup>421</sup> [4-Modeling/03d-LSTM \(LSTM-DENSE\);](#)  
[4-Modeling/03e-LSTM \(DENSE-LSTM\)](#)

## Erstellung des Vorhersagemodells

ergänzen. Die sogenannte Dense-Schicht kann entweder zur Vor- oder Nachverarbeitung der Daten entsprechend vor oder nach der LSTM-Schicht eingefügt werden.

Eine zusätzliche Dense-Schicht hinter der LSTM-Schicht führt zu keinen erkennbaren Verbesserungen. Die Lernkurve zeigt unabhängig von der Anzahl der Neuronen durchweg das gleiche Verhalten wie bei einer oder zwei LSTM-Schichten und lässt sich auch durch andere Lernraten nicht verbessern. Die Abbildung 89 zeigt die Lernkurve eines Netzes mit einer LSTM-Schicht à 32 und einer Dense-Schicht à 64 Neuronen über 100 Epochen. Hier zeigt sich wieder, dass das Netz sehr schnell anlernt. Die Konvergenz beginnt zwischen der 80. und 90. Epoche, Übergangspassagen sind bei der Lernkurve nicht erkennbar. Mit einem MAPE von 2,55% auf den Testdaten und 1,76% auf den Trainingsdaten ist das Modell sehr ähnlich zu den zweischichtigen LSTM-Netzen.<sup>422</sup>

**Abbildung 89: Lernkurve von LSTM-DENSE-2**



Quelle: Eigene Darstellung<sup>423</sup>

Durch andere Architekturen und Trainingsparameter lässt sich keine Verbesserung erzielen. Größere Netze mit beispielsweise 64 LSTM- und 128 Dense-Neuronen neigen zu einer stärker ausgeprägten Überanpassung.<sup>424</sup> Die Netze verhalten sich

<sup>422</sup> 4-Modeling/03d-LSTM (LSTM-DENSE) (LSTM-DENSE-2)

<sup>423</sup> 4-Modeling/03d-LSTM (LSTM-DENSE) (LSTM-DENSE-2)

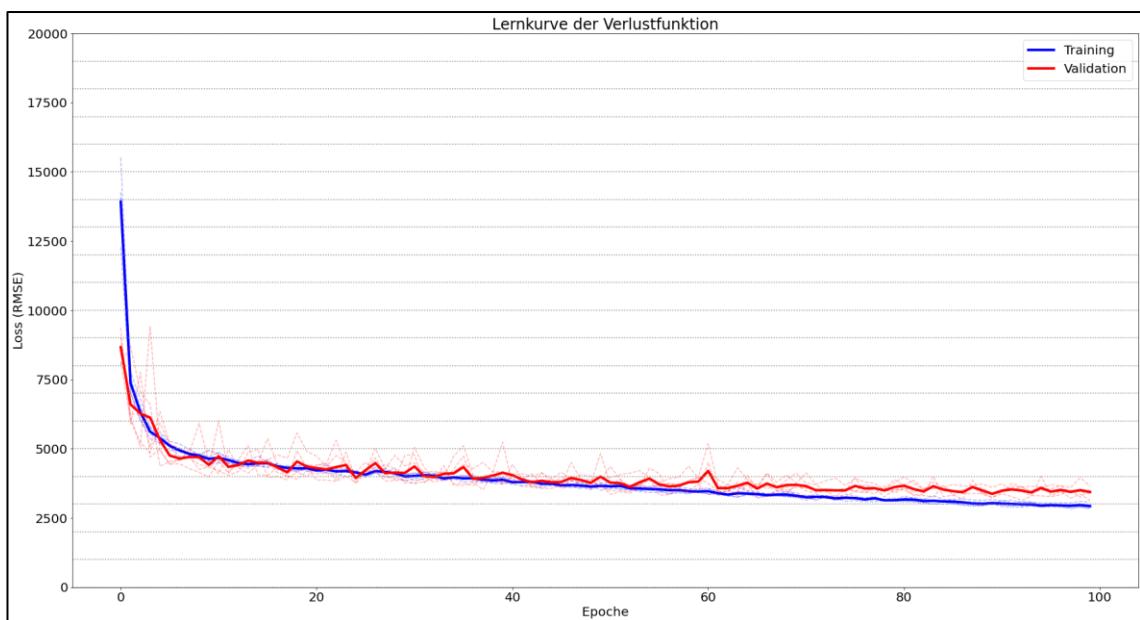
<sup>424</sup> 4-Modeling/03d-LSTM (LSTM-DENSE) (LSTM-DENSE-3)

## Erstellung des Vorhersagemodells

allgemein sehr ähnlich wie die zweischichtigen LSTM-Netze, auch der MAPE bewegt sich meistens um etwa 2,5% auf den Test- und 1,8% auf den Trainingsdaten.<sup>425</sup> Der Ansatz wird daher aus ähnlichen Gründen verworfen.

Wenn die Dense-Schicht aber vor der LSTM-Schicht eingefügt wird, führt dies zu einer erkennbaren Verbesserung. Die Abbildung 90 zeigt die Lernkurve einer Dense- und LSTM-Schicht mit jeweils 64 Neuronen. Das Modell erreicht einen MAPE von 2,15% auf den Testdaten und 1,33% auf den Trainingsdaten. Das Training verläuft in Bezug auf die Konvergenz und Überanpassungen jedoch ähnlich wie bei den vorangegangenen Modellen.<sup>426</sup> Trotzdem ist das Modell schon etwas besser als die Pendants aus zwei LSTM- oder einer Dense- und LSTM-Schicht. Die Vorverarbeitung der Daten durch eine Dense-Schicht verbessert die Vorhersagen also.

**Abbildung 90: Lernkurve von DENSE-LSTM-3**



Quelle: Eigene Darstellung<sup>427</sup>

Wenn die LSTM-Schicht auf 128 Neuronen erweitert wird, führt dies zu einer leichten Verschlechterung auf den Testdaten, da sich das Modell dann an die Trainingsdaten überanpasst. Die Überanpassung setzt nach etwa 60. Epochen ein.<sup>428</sup> Sie lässt sich durch den Einsatz einer leichten L2-Regularisierung in beiden Schichten aber

<sup>425</sup> [4-Modeling/03d-LSTM \(LSTM-DENSE\)](#)

<sup>426</sup> [4-Modeling/03e-LSTM \(DENSE-LSTM\) \(DENSE-LSTM-3\)](#)

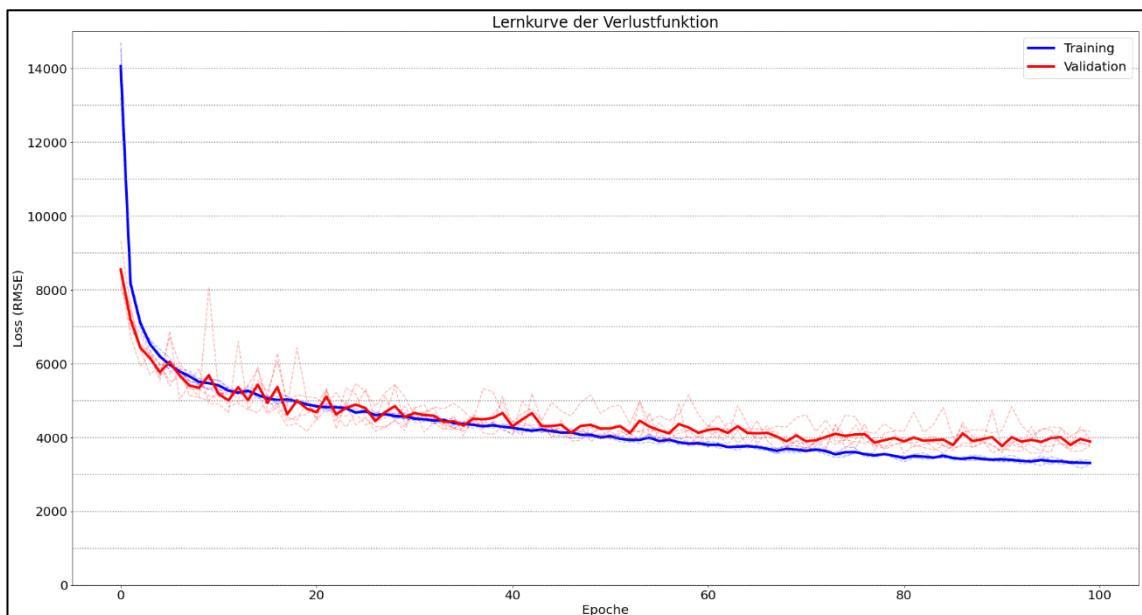
<sup>427</sup> [4-Modeling/03e-LSTM \(DENSE-LSTM\) \(DENSE-LSTM-3\)](#)

<sup>428</sup> [4-Modeling/03e-LSTM \(DENSE-LSTM\) \(DENSE-LSTM-4\)](#)

## Erstellung des Vorhersagemodells

zumindest teilweise beheben. Ein Modell aus einer Dense-Schicht à 64 und einer LSTM-Schicht à 128 Neuronen mit L2-Regularisierung von 0,0001 erreicht nach 100 Epochen einen MAPE von 2,17% auf den Test- und 1,34% auf den Trainingsdaten. Es ist damit zwar ähnlich dem Netz mit nur 64 Neuronen in der LSTM-Schicht, allerdings ist in der Lernkurve in Abbildung 91 zu erkennen, dass die Konvergenz auf den Testdaten nun etwas später einsetzt. Das Modell wurde nur 100 Epochen trainiert, die Konvergenz wird in diesem Zeitraum nicht erreicht.<sup>429</sup> Da die Modelle also über mehr Epochen trainiert werden müssen, wird das Training um 50 Epochen erweitert.

**Abbildung 91: Lernkurve von DENSE-LSTM-5**



Quelle: Eigene Darstellung<sup>430</sup>

Wie bereits oben beschrieben neigen tiefere Netze zum Problem der verschwindenden Gradienten. Explodierende Gradienten können bei der Verwendung der tanh-Funktion ausgeschlossen werden, da die Funktion Werte im Bereich von -1 bis +1 erzeugt. Ab der etwa 40. Epoche bringen zusätzliche Epochen nur noch eine sehr kleine Verbesserung der Verlustfunktion mit sich, was möglicherweise an verschwindenden Gradienten liegt. Daher wird die tanh-Aktivierungsfunktion in der Dense-Schicht durch die ReLU-Funktion ersetzt. Da die Funktion auch Werte über +1 erzeugen kann, könnte sie verschwindende Gradienten zumindest teilweise beheben

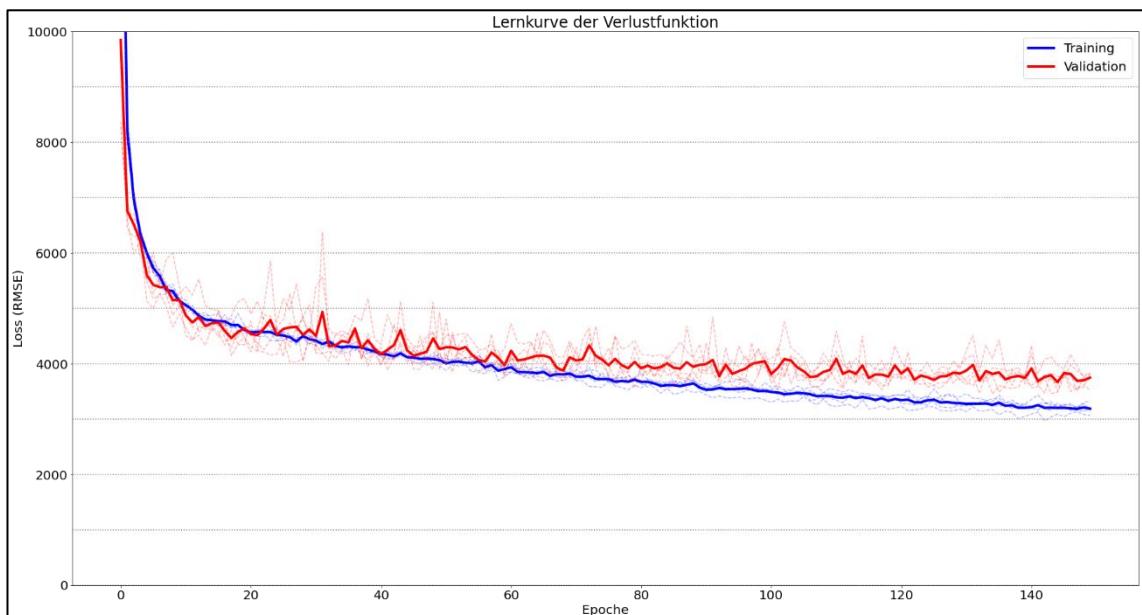
<sup>429</sup> [4-Modeling/03e-LSTM \(DENSE-LSTM\) \(DENSE-LSTM-5\)](#)

<sup>430</sup> [4-Modeling/03e-LSTM \(DENSE-LSTM\) \(DENSE-LSTM-5\)](#)

## Erstellung des Vorhersagemodells

(vorausgesetzt, diese treten überhaupt im Netz auf).<sup>431</sup> Durch den Einsatz der ReLU-Funktion lässt sich das Modell auch tatsächlich verbessern. Der MAPE fällt auf den Testdaten auf 1,99%, auf den Trainingsdaten auf 1,35%. Es liegt also immer noch eine leichte Überanpassung vor. Dabei handelt es sich aber wie bereits beschrieben um ein nicht unbedingt ungewöhnliches Phänomen, da die Trainingsdaten häufig etwas besser erkannt werden als die Testdaten. In der Abbildung 92 ist die Verlustfunktion des Modells abgebildet. Die Konvergenz auf den Testdaten setzt nun nach 120 Epochen ein.<sup>432</sup>

**Abbildung 92: Verlustfunktion von DENSE-LSTM-9**



Quelle: Eigene Darstellung<sup>433</sup>

Die Überanpassung an die Trainingsdaten lässt sich nicht durch eine stärkere Regularisierung (und auch nicht in Verbindung mit anderen Lernraten) beheben. Der Lernprozess wird dadurch zu stark an der Generalisierung gehindert und die Abweichungen nehmen sowohl auf den Testdaten wie auch auf den Trainingsdaten zu. Dropouts führen ebenfalls zu Unteranpassungen. Selbst ein Dropout von nur 5% zwischen den beiden Schichten führt teilweise zu einem Anstieg des MAPE auf fast 2,6% auf den Test- und 1,6% auf den Trainingsdaten.<sup>434</sup> Die Batchgrößere ist mit 8

<sup>431</sup> Vgl. Patterson, J., Gibson, A., Deep Learning, 2017, S. 69 ff.;  
Vgl. Plaat, A., Reinforcement Learning, 2020, S. 149 ff.

<sup>432</sup> [4-Modeling/03e-LSTM \(DENSE-LSTM\) \(DENSE-LSTM-9\)](#)

<sup>433</sup> [4-Modeling/03e-LSTM \(DENSE-LSTM\) \(DENSE-LSTM-9\)](#)

<sup>434</sup> [4-Modeling/03e-LSTM \(DENSE-LSTM\) \(DENSE-LSTM-12\)](#)

ebenfalls angemessen gewählt. Größere Batches führen nicht dazu, dass das globale Minimum der Verlustfunktion besser erreicht werden kann.<sup>435</sup> Bisher wurde bei den meisten Modellen eine Fenstergröße von 14 Zeitschritten gewählt. Wenn das Fenster auf sieben Zeitschritte verkleinert wird, kommt es zu einer leichten Überanpassung an die Trainingsdaten. Der MAPE auf den Testdaten steigt auf etwa 2,15% an, auf den Trainingsdaten sinkt er sogar leicht auf 1,3%. Das Modell kann also mit einem kürzeren Fenster schlechter generalisieren oder es werden gegebenenfalls wichtige (Kontext-)Informationen vorenthalten. Die LSTM-Schicht kann ihre Stärke, die Verarbeitung eines Zeitschritts im Kontext der vorangegangenen Werte, nicht voll ausspielen.<sup>436</sup> Umgekehrt führt ein größeres Fenster mit 21 Beobachtungen zu einer noch stärkeren Anpassung an die Trainingsdaten, da der MAPE auf den Testdaten auf 2,2% ansteigt und auf den Trainingsdaten sogar auf 1,26% fällt.<sup>437</sup> Die Fenstergröße ist mit 14 Zeitschritten angemessen, das Netz verwendet also Beobachtungen aus einem ähnlichen Zeitraum wie auch schon die ARIMA-Modelle. Ein ähnliches Verhalten zeigt sich auch bei der Überprüfung anderer Fensterlängen bei den LSTM-DENSE-Modellen.<sup>438</sup>

Die besten Ergebnisse lassen sich durch eine Dense-Schicht aus 64 Neuronen mit ReLU-Funktion und einer nachgelagerten LSTM-Schicht aus 128 Neuronen mit tanh-Funktion erzielen. Die Fensterlänge wird auf 14 festgelegt. Das Training erfolgt mit einer Lernrate von 0,0005, in beiden Schichten wird eine L2-Regularisierung von 0,0001 eingefügt. Insgesamt sind dadurch nur 100 Epochen mit Batches der Größe 8 notwendig.<sup>439</sup> Die Abbildung 93 zeigt die Lernkurve des Modells. Die Überanpassungen ab der etwa 60. Epoche lassen sich nicht durch Regularisierung oder Drop-outs beheben. Die Konvergenz auf den Testdaten wird nach etwa 100 Epochen erreicht (zum Vergleich kann die Lernkurve von DENSE-LSTM-9<sup>440</sup> herangezogen werden, da das Modell mit ansonsten gleichen Hyperparametern über 150 Epochen trainiert wurde. Darin ist die Konvergenz ab der 90. bis 100. Epoche erkennbar).

**Abbildung 93: Lernkurve von DENSE-LSTM-16 (fertiges Modell)**

---

<sup>435</sup> [4-Modeling/03e-LSTM \(DENSE-LSTM\) \(DENSE-LSTM-11\)](#)

<sup>436</sup> [4-Modeling/03e-LSTM \(DENSE-LSTM\) \(DENSE-LSTM-14\)](#)

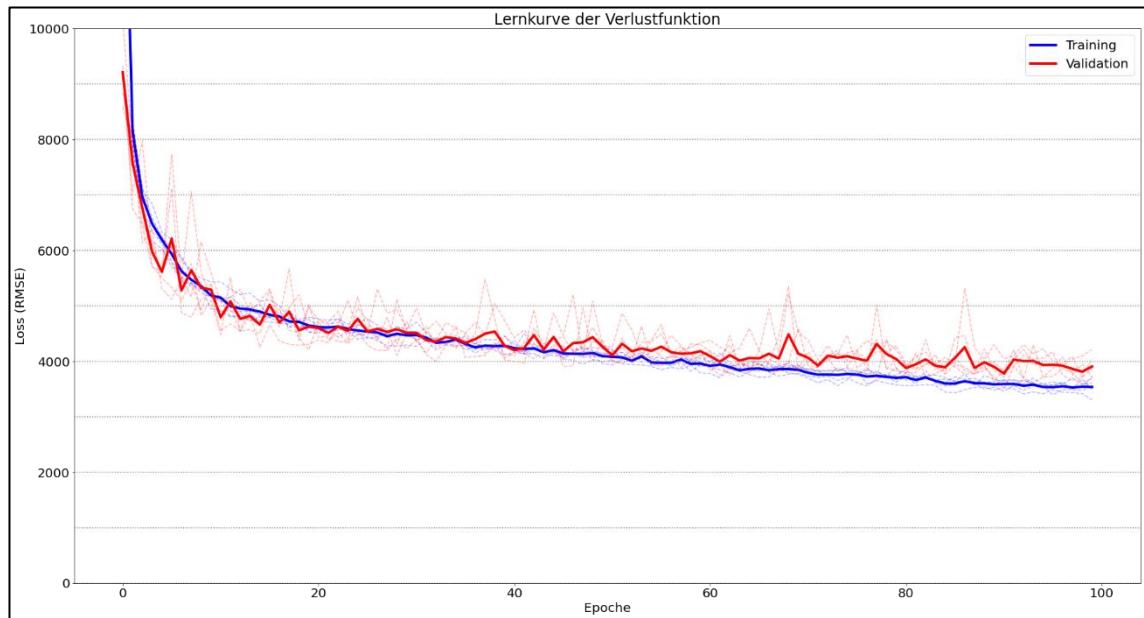
<sup>437</sup> [4-Modeling/03e-LSTM \(DENSE-LSTM\) \(DENSE-LSTM-15\)](#)

<sup>438</sup> [4-Modeling/03d-LSTM \(LSTM-DENSE\) \(LSTM-DENSE-7\)](#)

<sup>439</sup> [4-Modeling/03e-LSTM \(DENSE-LSTM\) \(DENSE-LSTM-16\)](#)

<sup>440</sup> [4-Modeling/03e-LSTM \(DENSE-LSTM\) \(DENSE-LSTM-9\)](#)

## Erstellung des Vorhersagemodells



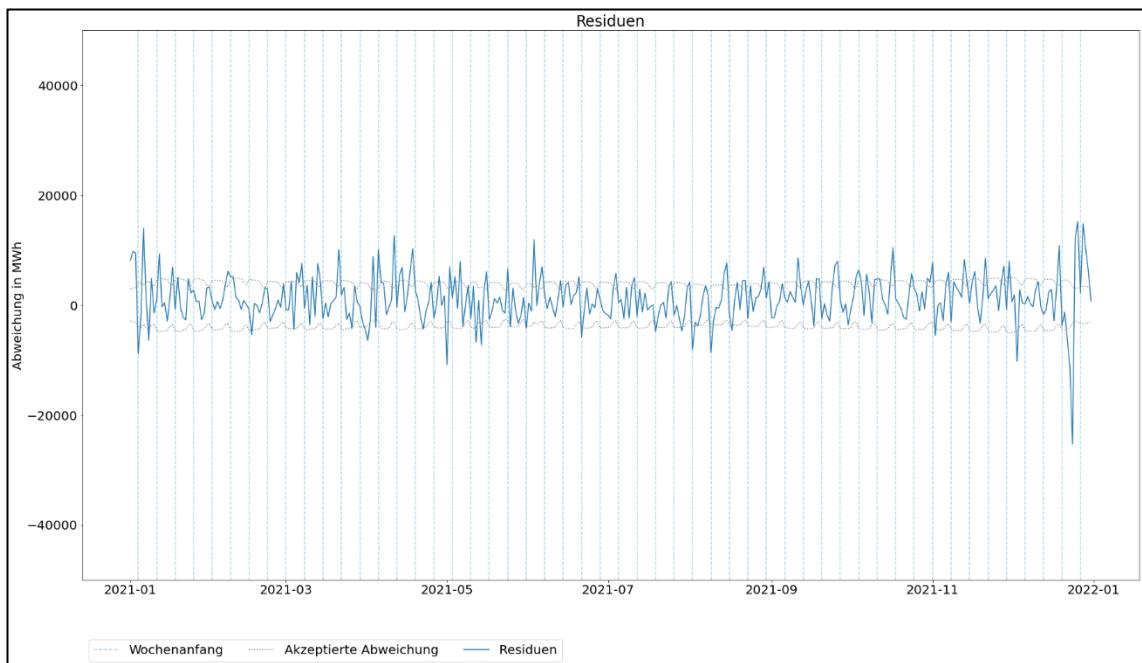
Quelle: Eigene Darstellung<sup>441</sup>

Das Modell erreicht einen MAPE von 1,94% auf den Testdaten und 1,37% auf den Trainingsdaten. Diese Werte sind konsistent mit den während der Kreuzvalidierungen beobachteten Ergebnissen. Die Abbildung 94 zeigt die Residuen des Modells. Dabei ist zu erkennen, dass es anders als bei ARIMA keine größeren Abweichungen um Feiertage gibt. Die meisten Residuen bewegen sich unter oder nur geringfügig über der akzeptierten Abweichung. Allerdings kommt es an den Weihnachtsfeiertagen zu erheblichen Abweichungen.

**Abbildung 94: Residuen von DENSE-LSTM-16**

<sup>441</sup> [4-Modeling/03e-LSTM \(DENSE-LSTM\) \(DENSE-LSTM-16\)](#)

## Erstellung des Vorhersagemodells



Quelle: Eigene Darstellung<sup>442</sup>

Anders als bei ARIMA lässt sich die Bedeutung der einzelnen Merkmale für das Modell nicht ohne weiteres aus dem Modell ablesen. Stattdessen wird die Bedeutung der einzelnen Merkmale ermittelt, indem das entsprechende Merkmal bei den Testdaten zufällig durchmischt beziehungsweise permutiert wird. Je stärker die Vorhersagen dann im Vergleich zu den normalen Testdaten abweichen, desto bedeutender ist das Merkmal für die Erstellung der Vorhersagen im Modell. Wenn sich die Abweichung der Vorhersagen durch die zufällige Durchmischung nicht oder nur geringfügig ändert, ist das Merkmal unbedeutend.<sup>443</sup> Die Tabelle 10: Merkmalsrelevanz bei LSTM zeigt die Ergebnisse auf den normalen und durchmischten Testdaten:

**Tabelle 10: Merkmalsrelevanz bei LSTM**

Durchmisches Merktal	MAPE Testdaten (regulär: 1,94%)	MAPE Trainingsdaten (regulär 1,37%)
<b>Verbrauch</b>	12,58%	12,55%
<b>Arbeitstag</b>	13,74%	16,11%

<sup>442</sup> [4-Modeling/03e-LSTM \(DENSE-LSTM\) \(DENSE-LSTM-16\)](#)

<sup>443</sup> Vgl. Audevert, A., Banachewicz, K., Massaron, L., Machine Learning Cookbook, 2021, S. 145 ff.

<b>Temperatur</b>	2,8%	2,37%
<b>Tagesstunden</b>	3,1%	3,28%

Quelle: Eigene Darstellung<sup>444</sup>

Es zeigt sich, dass der Stromverbrauch und der Indikator für Arbeitstage besonders viel Einfluss auf die Vorhersagen haben. Die Temperatur und Tagesstunden haben zwar einen geringeren Einfluss, scheinen dem Modell aber trotzdem einen gewissen Informationsgehalt für die Generalisierung zu bieten. Wird die Temperatur durchmischt, fällt der MAPE auf den Testdaten von 1,94% um fast einen Prozentpunkt auf 2,8%. Das entspricht einer relativen Steigerung von etwa 40%, die Vorhersagen sind also sehr viel schlechter. Ähnlich verhält es sich mit den Tagesstunden. Es werden daher weiterhin alle vier Merkmale für die Modellierung genutzt und kein Merkmal aus dem Modell entfernt.

### 3.6.3 Zwischenfazit zu LSTM<sup>445</sup>

Ein neuronales Netz mit einer Dense-Schicht à 64 Neuronen und einer LSTM-Schicht à 128 Neuronen kann den Stromverbrauch mit einer durchschnittlichen Abweichung von etwa 1,94% vorhersagen. Die Baseline wird um 0,5 Prozentpunkte beziehungsweise etwa 20% übertroffen. Das Erfolgskriterium eines MAPE von höchsten 2,2% ist damit erreicht. Die Vorhersagen des Netzes (rot) sind in der Abbildung 95 im Vergleich zu Baseline (orange) und dem tatsächlichen Stromverbrauch (blau) eingezeichnet. Da LSTM vor allem die wöchentliche Saisonalität etwas besser abbilden kann, liegen die Vorhersagen meist näher an den tatsächlichen Werten als die Baseline. Lediglich um die Weihnachtsfeiertage herum kann das Baseline-Modell nicht übertroffen werden. Anders als ARIMA hat LSTM kein Problem mit den übrigen Feiertagen.

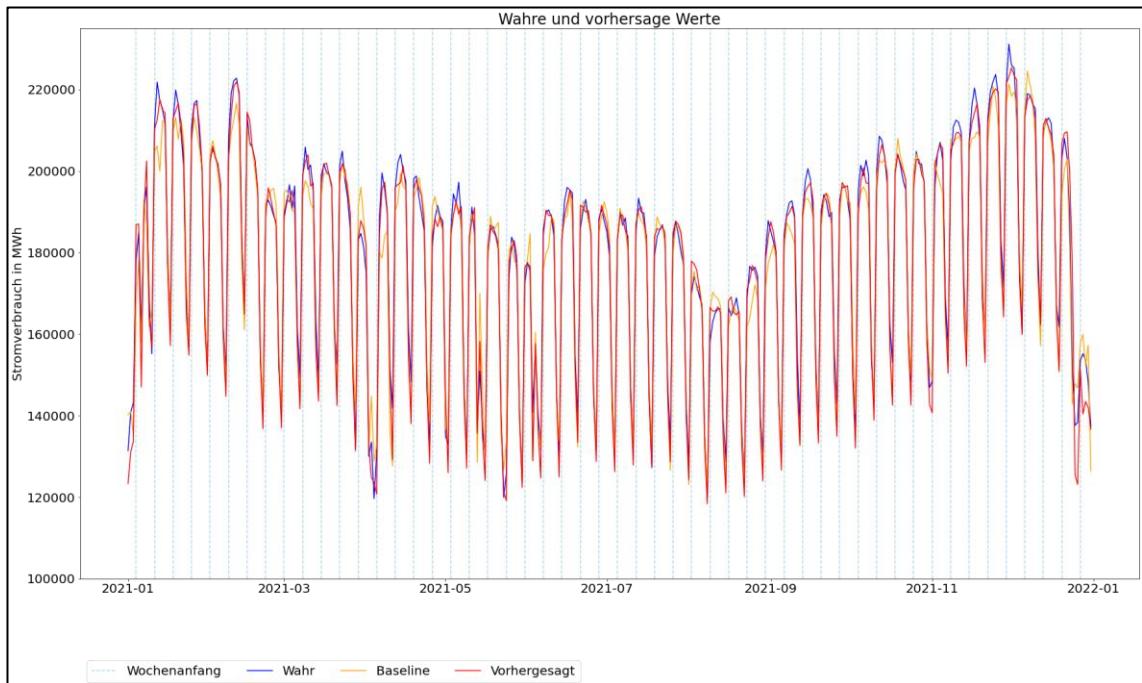
Abbildung 95: LSTM und Baseline

---

<sup>444</sup> [4-Modeling/03f-LSTM \(Merkmalsrelevanz\)](#)

<sup>445</sup> [4-Modeling/03g-LSTM \(Zwischenfazit\)](#)

## Erstellung des Vorhersagemodells



Quelle: Eigene Darstellung<sup>446</sup>

Das neuronale Netz ließe sich zwar durch mehr Schichten, zusätzliche Neuronen etc., andere Hyperparameter oder weitere exogene Daten theoretisch endlos erweitern, allerdings ist das Erfolgskriterium bereits erfüllt und das Modell soll wie am Anfang beschrieben möglichst einfach gehalten werden. Aus diesem Grund wird die Modellierung für LSTM erfolgreich abgeschlossen.

### 3.7 Evaluation<sup>447</sup>

Ein Teil der Evaluation ist bereits im jeweiligen Zwischenfazit in den Kapiteln 3.5.5 und 3.6.3 erfolgt. In diesem Kapitel wird das ARIMA-Modell mit dem neuronalen Netz (im Folgenden „LSTM-Modell“) anhand der Testdaten verglichen. Zusätzlich werden die Modelle statt wie bisher mit historischen Wetterdaten noch einmal mit historischen Wettervorhersagen getestet, um einen realistischen Einsatz zu simulieren. Abschließend wird das für die Vorhersage besser geeignete Modell ausgewählt. Das Kapitel enthält weiterhin eine Beschreibung der Restriktionen des Modells.

<sup>446</sup> 4-Modeling/03g-LSTM (Zwischenfazit) (Vergleich mit Baseline)

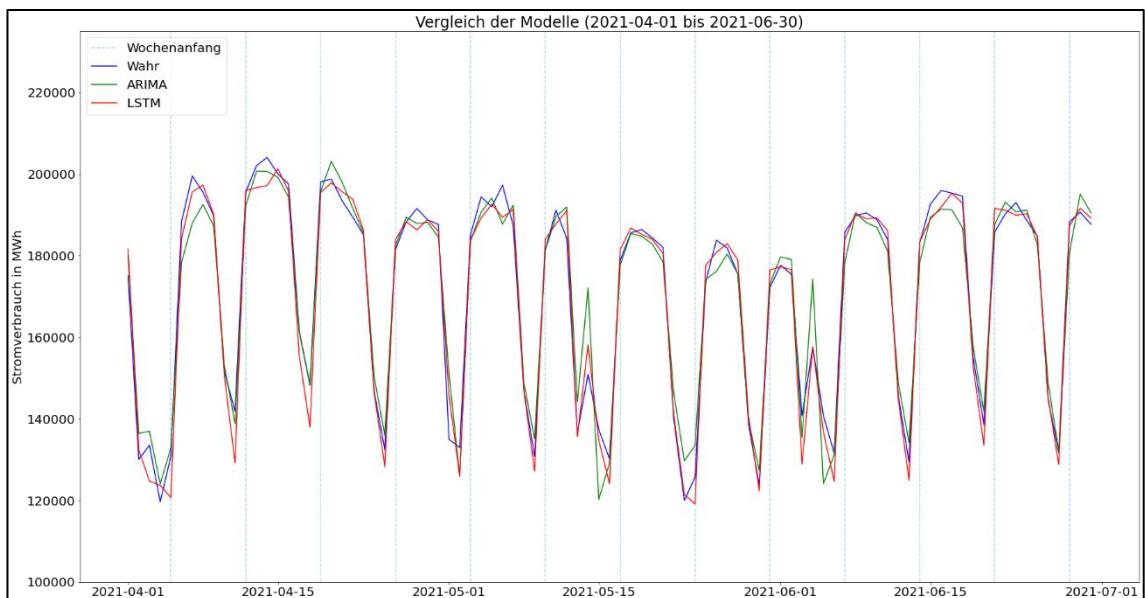
<sup>447</sup> 5-Evaluation

## Erstellung des Vorhersagemodells

### 3.7.1 Vergleich von ARIMA und LSTM<sup>448</sup>

Das Baseline-Modell erreicht auf den Testdaten einen MAPE von 2,44%. Das ARIMA-Modell weicht durchschnittlich um 2,10% ab, das LSTM-Modell erreicht sogar einen MAPE von nur 1,94%. Damit ist das LSTM-Modell 0,16 Prozentpunkte genauer als ARIMA. Eine genauere Auswertung ist dem jeweiligen Zwischenfazit (Kapitel 3.5.5 und 3.6.3) zu entnehmen. Die Abbildung 96 zeigt beispielsweise die Vorhersagen im Zeitraum von April bis Juni 2021. Das LSTM-Modell (rot) kann die wöchentliche Saisonalität besser abbilden als ARIMA (grün). Auch Feiertage (etwa Mitte Mai) führen beim LSTM-Modell zu praktisch keinen Verzerrungen.

**Abbildung 96: ARIMA und LSTM im Vergleich**



Quelle: Eigene Darstellung<sup>449</sup>

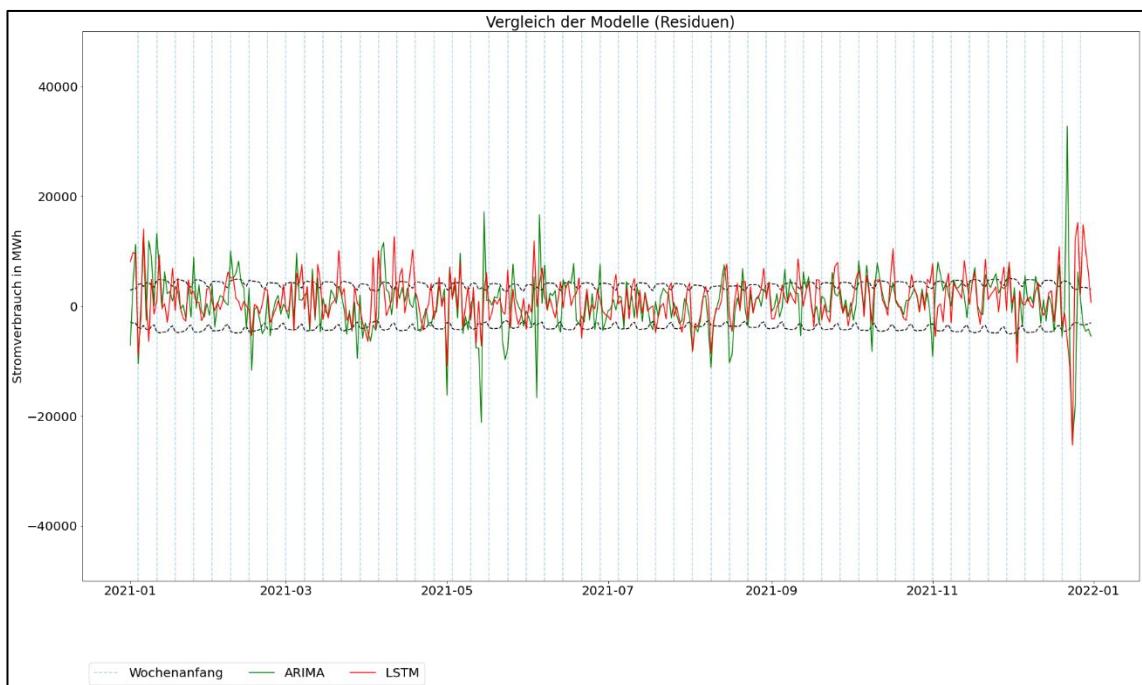
Die Abbildung 97 zeigt die Residuen der jeweiligen Modelle. Hier ist noch einmal erkennbar, dass LSTM (rot) an Feiertagen und den Tagen unmittelbar danach sehr viel bessere Vorhersagen macht als ARIMA (grün). Zwar gibt es beim LSTM-Modell stellenweise ebenfalls größere Ausreißer, diese sind jedoch sehr viel seltener und weniger stark ausgeprägt als bei ARIMA. In der Zeit Ende Dezember und Anfang Januar sind die Prognosen bei beiden Modellen sehr schlecht. In diesem Zeitraum werden beide Modelle von der Baseline geschlagen.

<sup>448</sup> [5-Evaluation/01-Vergleich ARIMA und LSTM](#)

<sup>449</sup> [5-Evaluation/01-Vergleich ARIMA und LSTM \(Diagramme\)](#)

## Erstellung des Vorhersagemodells

**Abbildung 97: Residuen von LSTM und ARIMA**



Quelle: Eigene Darstellung<sup>450</sup>

Ein genauerer Vergleich ist im Notebook einzusehen.<sup>451</sup>

### 3.7.2 Test mit Wettervorhersage<sup>452</sup>

Bisher wurden die Modelle mit historischen Daten trainiert und getestet. Wie bereits erwähnt müssen die exogenen Daten für den vorherzusagenden Tag zum Zeitpunkt der Erstellung der Prognose an das Modell übergeben werden. Da diese Werte in der Zukunft liegen, handelt es sich bei den exogenen Daten selbst um vorhergesagte Werte. Mögliche Vorhersagefehler bei exogenen Daten wirken sich auf die Vorhersagen des Modells aus und können diese eventuell verschlechtern.

Der Indikator für Arbeitstage und die Anzahl an Tagesstunden lassen sich praktisch exakt vorhersagen, Abweichungen sind hier nicht zu erwarten. Bei der Temperatur handelt es sich allerdings um einen vorhergesagten Wert, der in der Praxis durchaus vom tatsächlichen Wert abweichen kann. Um die Modelle unter realistischen Bedingungen zu testen, wurde für den Zeitraum vom 01.08. bis zum 31.12.2021 jeweils um 23:59 des Vortages die aktuelle Wettervorhersage für den nächsten Tag von

<sup>450</sup> 5-Evaluation/01-Vergleich ARIMA und LSTM (Diagramme)

<sup>451</sup> 5-Evaluation/01-Vergleich ARIMA und LSTM

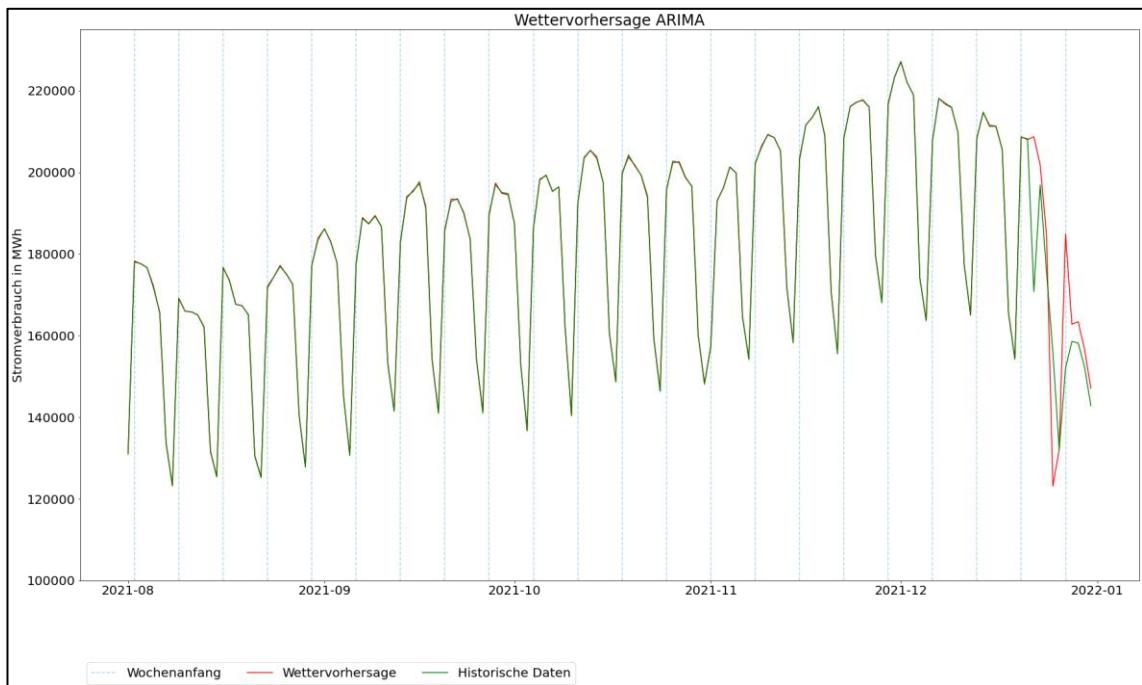
<sup>452</sup> 5-Evaluation/02-Test mit Wettervorhersage

## Erstellung des Vorhersagemodells

WeatherAPI.com gespeichert. Eine genauere Beschreibung der Beschaffung und Aufbereitung der Daten befindet sich im Notebook.<sup>453</sup> Die vorhergesagte Temperatur weicht im Durchschnitt etwa 1,2°C beziehungsweise circa 10% vom tatsächlichen (historischen) Wert ab.<sup>454</sup> Es können nun mit diesen Daten von beiden Modellen Vorhersagen erzeugt werden, um zu überprüfen, wie sich die Modelle unter realistischen Bedingungen im Einsatz verhalten hätte. Dafür werden die exogenen Daten für den jeweils vorherzugsagenden Tag in den Daten durch die vorhergesagten Werte ausgetauscht. Abweichungen sind nur bei der Temperatur zu erwarten.

Das ARIMA-Modell verschlechtert seinen MAPE auf etwa 2,22%. Wie in der Abbildung 98 erkennbar ist, scheint das Modell kaum durch die Vorhersagefehler bei der Temperatur beeinflusst zu werden. Die Vorhersagen mit den historischen Wetterdaten sind in grün, mit der Wettervorhersage in rot eingezeichnet. Nur in der Urlaubsaison gegen Ende Dezember kommt es zu erheblichen Abweichungen.<sup>455</sup> Das Erfolgskriterium kann allerdings trotzdem erfüllt werden.

**Abbildung 98: Wettervorhersage ARIMA**



<sup>453</sup> [2-Data Understanding/Datenbeschaffung/03-Wettervorhersage](#)

<sup>454</sup> [5-Evaluation/02-Test mit Wettervorhersage \(Temperatur\)](#)

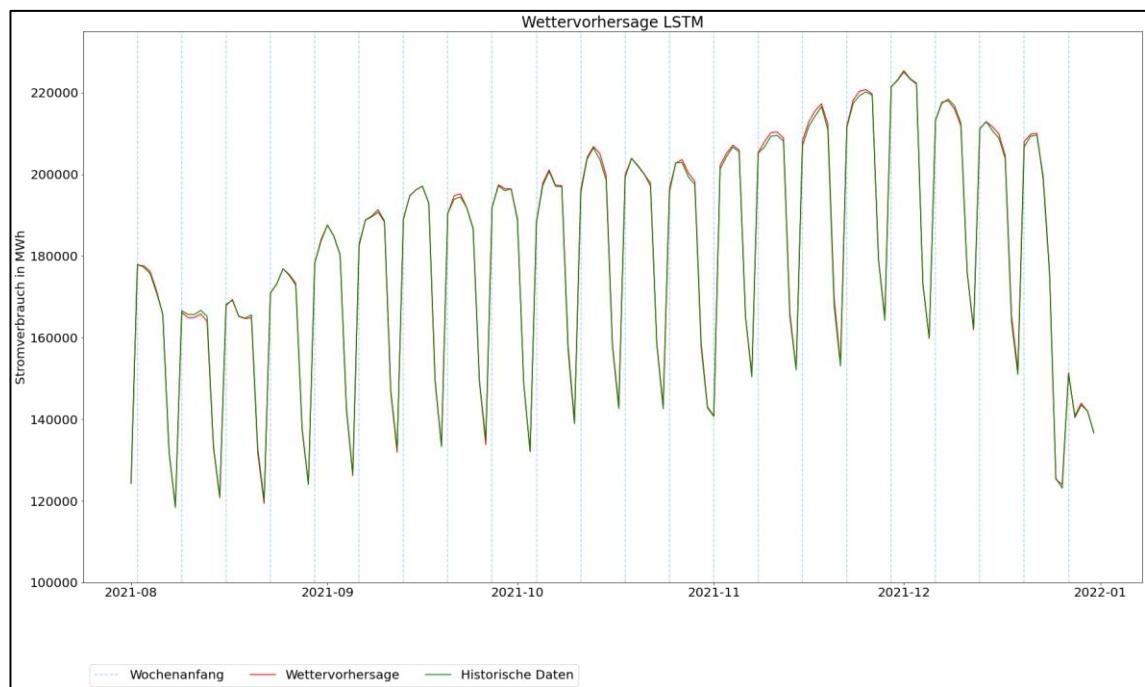
<sup>455</sup> [5-Evaluation/02-Test mit Wettervorhersage \(Statistiken\)](#)

## Erstellung des Vorhersagemodells

Quelle: Eigene Darstellung<sup>456</sup>

Beim LSTM-Modell fallen die Abweichungen geringer aus als bei ARIMA. Der MAPE verschlechtert sich auf 2,09%. Wie die Abbildung 99 zeigt, hat die Abweichung der Temperatur durch die Wettervorhersage nur stellenweise Einflüsse auf die Vorhersagen des Modells. Die Vorhersagen mit den historischen Wetterdaten sind in grün, mit der Wettervorhersage in rot eingezeichnet. <sup>457</sup> Das Erfolgskriterium gilt also auch für das LSTM-Modell weiterhin als erfüllt.

**Abbildung 99: Wettervorhersage LSTM**



Quelle: Eigene Darstellung<sup>458</sup>

Es ist dabei zu erwähnen, dass diese Überprüfung nicht beweist, dass die Modelle allgemein nicht auf die Temperatur angewiesen sind. Die vorhergesagte Temperatur weicht im Schnitt nur 1,2°C von der echten Temperatur ab und ist damit immer noch sehr nah am tatsächlichen Wert. Trotzdem kommt es schon zu einer erkennbaren Verschlechterung der Vorhersagen bei beiden Modellen. Die Temperatur bietet den Modellen erwiesenermaßen einen zusätzlichen Informationsgehalt (siehe dafür Kapitel 3.5.3 und 3.6.2), allerdings sind die Modelle einigermaßen robust gegen derartig

<sup>456</sup> 5-Evaluation/02-Test mit Wettervorhersage (ARIMA)

<sup>457</sup> 5-Evaluation/02-Test mit Wettervorhersage (Statistiken)

<sup>458</sup> 5-Evaluation/02-Test mit Wettervorhersage (LSTM)

geringe Vorhersagefehler bei einem einzigen Merkmal. Es ist daher davon auszugehen, dass die Modelle die Erfolgskriterien auch unter realistischen Bedingungen erfüllen können, allerdings handelt es sich beim hier untersuchten Zeitraum von vier Monaten nicht um einen repräsentativen Zeitraum, zumal die schwer prognostizierbaren Weihnachtsfeiertage ebenfalls im Zeitraum liegen.

### 3.7.3 Modellauswahl und Restriktionen

Im Vergleich der beiden Modelle hat sich gezeigt, dass das LSTM-Modell den Stromverbrauch mit einem MAPE von 1,94% besser vorhersagen kann als das ARIMA-Modell mit 2,10%. Das LSTM-Modell ist allgemein etwas genauer als ARIMA, zusätzlich ist aber die Vorhersage von Feiertagen wesentlich genauer und es kommt sehr viel seltener zu starken Abweichungen. Außerdem ist das LSTM-Modell weniger anfällig für Abweichungen aufgrund von Vorhersagefehlern bei den exogenen Daten. Da das LSTM-Modell genauer, zuverlässiger und robuster ist als das ARIMA-Modell, wird es als das besser geeignete Modell ausgewählt. Die genauen Informationen zum Aufbau und Training des Modells können dem entsprechenden Notebook entnommen werden.<sup>459</sup>

Das Modell kann zwar die Baseline und das erforderliche Qualitätskriterium übertreffen, allerdings gibt es verschiedene Restriktionen, die beim Einsatz des Modells unbedingt zu beachten sind. Die wichtigsten Restriktionen werden hier aufgelistet und beschrieben, außerdem wird ein Ausblick auf Verbesserungsansätze gegeben:

1. Das Modell basiert zu einem großen Teil auf Autokorrelationen in den Stromverbrauchsdaten. Es handelt sich dabei zwar um statistisch signifikante, starke Korrelationen, allerdings sind dies keine Kausalitäten. Der Stromverbrauch hängt in Wirklichkeit nicht vom Verbrauch der letzten Tage ab, sondern von zahlreichen Faktoren, die sich nicht alle in einem Modell abbilden lassen und die gegebenenfalls nicht einmal bekannt sind.
2. Das Modell ist zwar einigermaßen robust gegen Vorhersagefehler bei den exogenen Daten, die Genauigkeit der Vorhersagen hängt trotzdem auch von der Richtigkeit der exogenen Daten beziehungsweise der Datenqualität allgemein ab.

---

<sup>459</sup> [4-Modeling/03g-LSTM \(Zwischenfazit\)](#)

3. Das Modell erreicht zwar eine sehr geringe durchschnittliche Abweichung, trotzdem sind in den Testdaten einige wenige Tage mit Abweichungen von bis zu 20.000 MWh zu erkennen. Besonders an und um die Weihnachtsfeiertage sind die Prognosen sehr unzuverlässig. Es sind zwar ansonsten keine größeren Abweichungen an Feiertagen erkennbar, allerdings handelt es sich bei den Feiertagen um schwer vorhersagbare Tage. Gerade an diesen Tagen sollte die Zuverlässigkeit des Modells also besonders kritisch betrachtet werden. Das Modell kann durchaus zusätzliche Informationen für beispielsweise die Kapazitätenplanung beim Stromnetzbetreiber liefern, eine Absicherung durch ergänzende Maßnahmen sollte dennoch trotzdem erfolgen.
4. Das Modell nutzt Informationen, die aus historischen Daten generalisiert werden konnten. Unvorhergesehene oder außergewöhnliche Umstände (wie beispielsweise der Corona-Lockdown, Wetterextreme etc.), die nicht oder nicht ausreichend in den Trainingsdaten enthalten sind, lassen sich nicht im Modell berücksichtigen und können daher nicht vorhergesagt werden.
5. Die Analyse und die Modellierung wurden mit den historischen Daten von Baden-Württemberg durchgeführt. Daher gelten die hier gewonnenen Erkenntnisse ausschließlich für das gesamte Bundesland Baden-Württemberg. Die Ergebnisse und auch das Modell sind sehr wahrscheinlich nicht auf andere Bereiche beziehungsweise Entitäten übertrag- oder skalierbar. Das Modell lässt sich also nicht auf andere Bundesländer, ganze Staaten oder einzelne Städte (auch nicht in Baden-Württemberg) etc. anwenden. In anderen Kontexten gibt es möglicherweise völlig andere Zusammenhänge, die andere Parameter, Architekturen etc. erfordern und nicht in diesem Modell abgebildet sind. Vom Einsatz des Modells in anderen Kontexten ist daher abzusehen.
6. Weiterhin sind die gewonnenen Erkenntnisse streng genommen nur für den Zeitraum von 2015 bis 2021 gültig. Zusammenhänge und Begebenheiten können (und werden) sich mit der Zeit ändern. Dieses Phänomen wird auch als Data oder Concept Drift<sup>460</sup> bezeichnet. Wie stark und schnell sich die Daten ändern, kann in weiteren Untersuchungen geprüft werden. Allerdings muss bedacht werden, dass das Modell nach einer gewissen Zeit entweder durch zusätzliche Daten aktualisiert, neu trainiert oder ganz in seiner

---

<sup>460</sup> Vgl. Putatunda, S., Practical Machine Learning, 2021, S. 31 ff.;

Vgl. Lakshmanan, V., Robinson, S., Munn, M., Machine Learning Design, 2020, S. 221 ff.

Architektur geändert werden muss. Jedenfalls kann das Modell unverändert nur für eine begrenzte Zeit eingesetzt werden.

7. Die Modellierung wurde als erfolgreich abgeschlossen, da das Modell die Erfolgskriterien erfüllt. Dieser Untersuchung liegt allerdings ausdrücklich nicht der Anspruch zugrunde, das bestmögliche Modell entwickelt zu haben. Das Modell lässt sich wie bereits erwähnt durch weitere Schichten, mehr Neuronen, andere Architekturen und ein anderes Training erweitern und gegebenenfalls verbessern. Das Modell kann durch mehr Daten oder andere Merkmale weiterhin optimiert werden. Zusätzlich bieten sich auch andere Algorithmen wie etwa die Vektorautoregression<sup>461</sup> oder die sogenannten konvolutionalen neuronalen Netze<sup>462</sup> an, die nicht Teil dieser Untersuchung sind, sich aber auch für die Zeitreihenvorhersage eignen. Eventuell lassen sich mehrere Modelle miteinander kombinieren, was auch als Ensemble<sup>463</sup> bezeichnet wird. Es kann beispielsweise ein Modell speziell für die Vorhersage von Feiertagen entwickelt werden, dass dann mit einem Modell für die „übrigen“ Tage kombiniert wird. In jedem Fall gibt es zahlreiche weitere Verbesserungsansätze und alternative Herangehensweisen, die in weiteren Untersuchungen überprüft werden könnten und gegebenenfalls bessere Ergebnisse erzielen.
8. Wie in Kapitel 2.1 dargestellt muss immer möglichst gleich viel Strom ins Netz eingespeist und gleichzeitig wieder entnommen werden. Eine Prognose des täglichen Stromverbrauchs kann zwar bei der Planung von Kapazitäten helfen, allerdings wäre eine stündliche oder viertelstündliche Prognose möglicherweise hilfreicher. In einer weiteren Untersuchung könnte also versucht werden, die Prognosen mit zeitlich kürzeren Horizonten zu erstellen.

### 3.8 Deployment<sup>464</sup>

Das fertige Modell wird mit den Daten vom 01.01.2015 bis zum 31.12.2021 trainiert. Es werden also alle verfügbaren Daten genutzt, ein weiterer Test wie bei der Entwicklung entfällt. Die Skripte für die Datenvorbereitung und das Training sind im

---

<sup>461</sup> Vgl. Patel, A., Vishwas, B. V., Time Series Analysis Python, 2020, S. 172 ff.

<sup>462</sup> Vgl. Hirschle, J., Machine Learning für Zeitreihen, 2020, S. 203 ff.

<sup>463</sup> Vgl. Raschka, S., Mirjalili, V., Machine Learning mit Python, 2021, S. 247 ff.

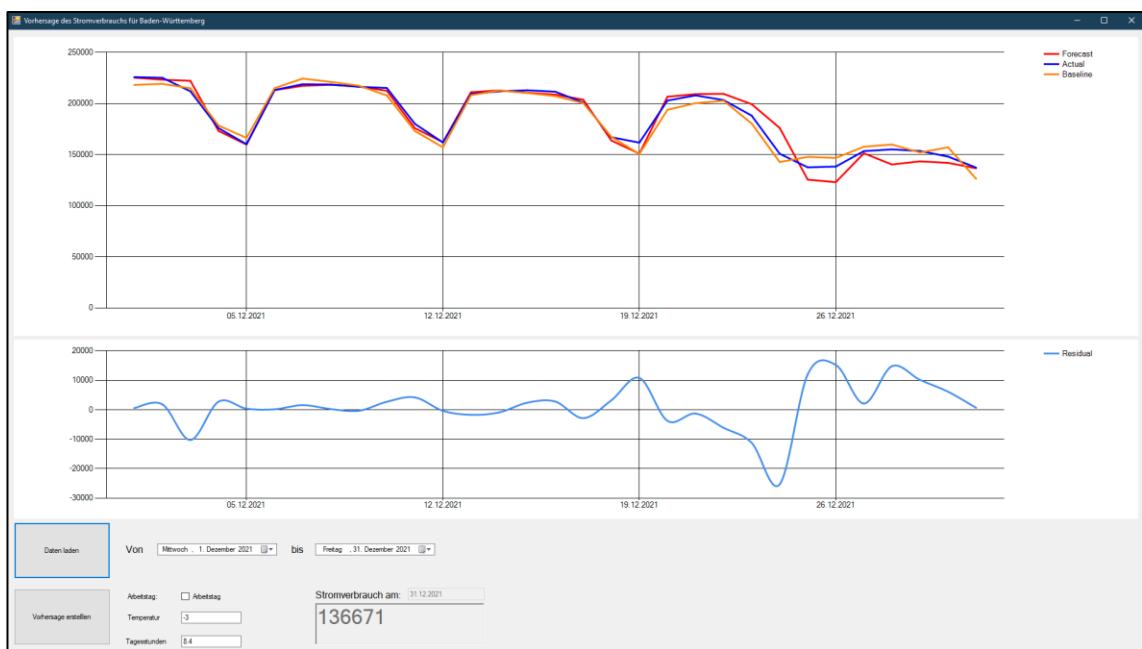
<sup>464</sup> [6-Deployment](#)

## Erstellung des Vorhersagemodells

Anhang einzusehen, es handelt sich um das gleiche Vorgehen wie im Data Understanding beziehungsweise im Modeling.<sup>465</sup>

Das fertige Modell ist in eine Benutzeroberfläche integriert. Die Benutzeroberfläche ist in C# entwickelt und kann im Anhang eingesehen werden. Die Anwendung kann über die Datei „Vorhersage.bat“ im Anhang gestartet werden. Für die Lauffähigkeit müssen das .NET Framework sowie Python mit den Bibliotheken pandas, NumPy, joblib, TensorFlow/Keras und scikit-learn installiert sein. Die Abbildung 100 zeigt die Oberfläche der Anwendung. Durch den Knopf „Daten laden“ werden die historischen Daten (gesamtes Jahr 2021) geladen. Im oberen Diagramm sind die Vorhersagen (rot), der tatsächliche Stromverbrauch (blau) und die Vorhersagen der Baseline (orange) abgetragen, im unteren Diagramm werden die Residuen (blau) gezeigt.

**Abbildung 100: Oberfläche (Daten laden)**



Quelle: Eigene Darstellung

Unten links können der Indikator für Arbeitstage, die Temperatur und die Anzahl an Tagesstunden eingetragen werden. Mit dem Knopf „Vorhersage erstellen“ werden die Daten per Python-Skript<sup>466</sup> an das Modell übergeben, welches eine Vorhersage

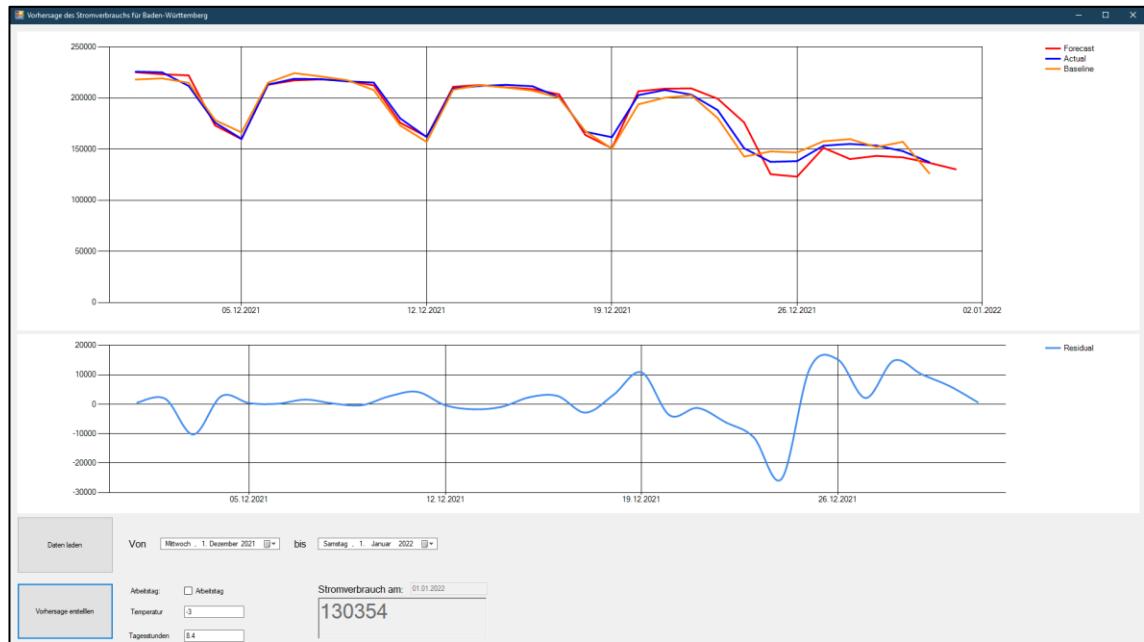
<sup>465</sup> [6-Deployment/data\\_preparation](#), [6-Deployment/model\\_creation](#)

<sup>466</sup> [6-Deployment/forecasting](#)

## Erstellung des Vorhersagemodells

für den nächsten Tag erzeugt. Die Vorhersage wird unten im Zahlenfeld angezeigt und in der oberen Grafik aktualisiert (siehe Abbildung 101).

**Abbildung 101: Oberfläche (Vorhersage erstellen)**



Quelle: Eigene Darstellung

## Zusammenfassung

Da moderne Gesellschaften und Volkswirtschaften sehr stark von elektrischem Strom abhängig sind, ist die Stabilität des Stromnetzes grundlegend wichtig. Strom kann nicht im erforderlichen Maß gespeichert werden, daher muss stets gleichviel Strom ins Netz eingespeist werden, wie dem Netz entnommen wird. Bei Abweichungen von Verbrauch und Einspeisung muss teure Regelenergie eingespeist werden. Dadurch entstehen hohe Kosten und die Stabilität des Stromnetzes wird gefährdet. Um Erzeugungs- und Reservekapazitäten optimal zu planen, sind genaue Prognosen des Stromverbrauchs notwendig. Die bisher verwendeten Prognosen haben eine durchschnittliche Abweichung von 2,4%.

Beim Stromverbrauch handelt es sich um eine Zeitreihe, also eine nach Zeitpunkten geordnete Abfolge von Werten. Zeitreihen bestehen aus systematischen Komponenten wie dem Level, dem Trend oder der Saisonalitäten, welche sich modellieren lassen. Nicht-systematische Komponenten oder Einflüsse sind in den nur bedingt modellierbaren Residuen zusammengefasst. Wenn Zeitreihen starke Autokorrelationen, also Korrelationen mit eigenen, vergangenen Werten aufweisen, dann können diese Autokorrelationen für Vorhersagen verwendet werden.

Es gibt unter anderem zwei gängige Algorithmen zur Zeitreihenvorhersage: ARIMA und LSTM. Bei ARIMA wird eine Regression auf einen Teil der Zeitreihe gebildet, die dann durch Korrekturen auf Basis vorheriger Vorhersagefehler zusätzlich verbessert wird. Bei LSTM wird ein neuronales Netz erstellt, welches Eingangswerte im Kontext eines Zeitfensters verarbeitet. Zur Bewertung eines Vorhersagemodells lässt man das Modell für einen bestimmten Zeitraum, für den die tatsächlichen Werte bekannt sind, Vorhersagen erstellen. Die Vorhersagen können dann mit den echten Werten wie bei einer Regression anhand verschiedener Metriken verglichen werden.

Der Stromverbrauch folgt einer jährlichen Saisonalität, welche ihr Maximum im Winter erreicht. Im Sommer fällt der Stromverbrauch zunächst, steigt aber ab Temperaturen von etwa 24°C wieder an. Außerdem folgt er einer wöchentlichen Saisonalität, da der Verbrauch montags bis freitags wesentlich höher ist als an Wochenenden, ausgenommen sind Feiertage. Der Stromverbrauch hängt weiterhin von der Temperatur und der Anzahl an Tagesstunden ab. Andere Wetterbedingungen haben keinen erkennbaren Einfluss auf den Stromverbrauch.

Es wurde jeweils ein ARIMA- und ein LSTM-Modell zur Vorhersage des Stromverbrauchs erstellt. Das ARIMA-Modell verwendet die letzten zwei, das siebte und das vierzehnte Lag sowie die Temperatur und einen Indikator für Arbeitstage. Es lässt sich damit der Stromverbrauch mit einer Abweichung von durchschnittlich 2,10% vorhersagen, allerdings hat das Modell Schwächen bei Feiertagen und in der Urlaubssaison um Weihnachten. Das LSTM-Modell besteht aus einer Dense-Schicht à 64 Neuronen und einer LSTM Schicht à 128 Neuronen. Es nutzt neben dem Stromverbrauch der letzten 14 Tage den Indikator für Arbeitstage, die Temperatur und die Anzahl an Tagesstunden der letzten 13 Tage und des vorherzusagenden Tages für die Erstellung von Vorhersagen. Es erreicht eine durchschnittliche Abweichung von 1,94%. Feiertage stellen zunächst kein erkennbares Problem dar. Bis auf den Zeitraum um Weihnachten können beide Modelle die Baseline übertreffen. Das LSTM-Modell wird aufgrund seiner höheren Zuverlässigkeit und Robustheit als besseres Modell gewählt. Das Modell wurde in eine Benutzeroberfläche integriert.

## Fazit

Der Stromverbrauch des Bundeslandes Baden-Württemberg lässt sich bei einem Horizont von einem Tag durch ein zweischichtiges neuronales Netz mit einer Abweichung von etwas unter 2% besser als durch das Baseline-Modell vorhersagen. Die Untersuchung kann also erfolgreich abgeschlossen werden.

Die Zeitreihe des Stromverbrauchs folgt einer jährlichen und einer wöchentlichen Saisonalität. Vor allem die wöchentliche Saisonalität lässt sich gut für die Modellierung verwenden. Der Stromverbrauch ist von montags bis freitags erkennbar höher als samstags und sonntags. Eine Ausnahme sind Feiertage, die in etwa mit Samstagen und Sonntagen verglichen werden können. Die Zeitreihe des Stromverbrauchs fasst einige signifikante Autokorrelationen. Für die Vorhersage des Stromverbrauchs sind vor allem die letzten beiden Wochen und besonders die beiden korrespondierenden Tage in diesen Wochen entscheidend.

Weiterhin ist der Stromverbrauch von der Tagesdurchschnittstemperatur abhängig. Zunächst fällt der Stromverbrauch mit steigender Temperatur, da weniger Energie für Heizungen etc. benötigt wird. Ab einer Temperatur von 20 bis 24°C beginnt der Stromverbrauch wieder mit der Temperatur zu steigen, da dann mehr Energie für Kühlungen etc. benötigt wird. Weiterhin ist der Stromverbrauch von der Anzahl der Tagesstunden abhängig, da längere Tage zu einem leicht geringeren Stromverbrauch führen. Andere Wetterbedingungen scheinen zunächst teilweise auch einen Einfluss auf den Stromverbrauch zu haben, dabei handelt es sich aber meist um Scheinkorrelationen, deren Kausalität in der Temperatur begründet liegt.

Das neuronale Netz besteht aus einer Dense-Schicht mit 64 Neuronen (mit ReLU) und einer LSTM-Schicht mit 128 Neuronen (mit tanh). Das Modell konvergiert ab etwa 100 Epochen. Dabei kommt es zu leichten Überanpassungen, die sich nicht beheben lassen, ohne das Modell signifikant zu verschlechtern. Die aus den Daten generalisierten Zusammenhänge werden schnell vom Algorithmus erkannt, weitere Epochen führen dann nur noch zu geringen Verbesserungen.

Die durchschnittliche Abweichung ist mit etwas unter 2% zwar sehr gering und die Baseline kann bei Weitem übertroffen werden, dennoch kann es gelegentlich zu größeren Abweichungen kommen. Entsprechende Restriktionen und Verbesserungsansätze sind im Kapitel 3.7.3 dargestellt.

## Anhang

### Anhang 1: Quelltexte

Die Quelltexte wurden in Python-Notebooks entwickelt und sind im Anhang sowohl als Notebook (.ipynb-Datei) und im HTML-Format enthalten. Die Notebooks sind nach ihrer Phase geordnet. Die Wetterdaten sind aus Lizenzgründen nicht auf GitHub hochgeladen, alle übrigen Dateien sind auf GitHub zu finden unter.:

<https://github.com/ArneDecker/Masterthesis>

Die folgende Tabelle gibt eine Übersicht über den Anhang:

Pfad	Datei	Link zu Notebook	Link zu HTML
1-Business Understanding	01-Baseline	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/1-Business%20Understanding/01-Baseline.ipynb">https://github.com/ArneDecker/Masterthesis/blob/main/1-Business%20Understanding/01-Baseline.ipynb</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/1-Business%20Understanding/01-Baseline.html">https://github.com/ArneDecker/Masterthesis/blob/main/1-Business%20Understanding/01-Baseline.html</a>
1-Business Understanding	02-Weitere Baseline-Modelle	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/1-Business%20Understanding/02-Weitere%20Baseline-Modelle.ipynb">https://github.com/ArneDecker/Masterthesis/blob/main/1-Business%20Understanding/02-Weitere%20Baseline-Modelle.ipynb</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/1-Business%20Understanding/02-Weitere%20Baseline-Modelle.html">https://github.com/ArneDecker/Masterthesis/blob/main/1-Business%20Understanding/02-Weitere%20Baseline-Modelle.html</a>
2-Data Understanding/	01-Verbrauch	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-">https://github.com/ArneDecker/Masterthesis/blob/main/2-</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-">https://github.com/ArneDecker/Masterthesis/blob/main/2-</a>

Datenbeschaf-fung		<a href="#">Data%20Understanding/Datenbeschaf-fung/01-Verbrauch.ipynb</a>	<a href="#">Data%20Understanding/Datenbeschaf-fung/01-Verbrauch.html</a>
2-Data Under-standing/ Datenbeschaf-fung	02-Wetter	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/Datenbeschaffung/02-Wetter.ipynb">https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/Datenbeschaffung/02-Wetter.ipynb</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/Datenbeschaffung/02-Wetter.html">https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/Datenbeschaffung/02-Wetter.html</a>
2-Data Under-standing/ Datenbeschaf-fung	03-Wettervorhersage	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/Datenbeschaffung/03-Wettervorhersage.ipynb">https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/Datenbeschaffung/03-Wettervorhersage.ipynb</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/Datenbeschaffung/03-Wettervorhersage.html">https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/Datenbeschaffung/03-Wettervorhersage.html</a>
2-Data Under-standing	01-Stromverbrauch	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/01-Stromverbrauch.ipynb">https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/01-Stromverbrauch.ipynb</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/01-Stromverbrauch.html">https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/01-Stromverbrauch.html</a>
2-Data Under-standing	02a-Temepratur (Ver-gleich)	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/02a-Temperatur%20(Ver-gleich).ipynb">https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/02a-Temperatur%20(Ver-gleich).ipynb</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/02a-Temperatur%20(Ver-gleich).html">https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/02a-Temperatur%20(Ver-gleich).html</a>

2-Data Understanding	02b-Temperatur	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/02b-Temperatur.ipynb">https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/02b-Temperatur.ipynb</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/02b-Temperatur.html">https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/02b-Temperatur.html</a>
2-Data Understanding	03a-Tagesstunden (Vergleich)	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/03a-Tagesstunden%20(Vergleich).ipynb">https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/03a-Tagesstunden%20(Vergleich).ipynb</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/03a-Tagesstunden%20(Vergleich).html">https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/03a-Tagesstunden%20(Vergleich).html</a>
2-Data Understanding	03b-Tagesstunden	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/03b-Tagesstunden.ipynb">https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/03b-Tagesstunden.ipynb</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/03b-Tagesstunden.html">https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/03b-Tagesstunden.html</a>
2-Data Understanding	04a-Luftfeuchtigkeit (Vergleich)	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/04a-Luftfeuchtigkeit%20(Vergleich).ipynb">https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/04a-Luftfeuchtigkeit%20(Vergleich).ipynb</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/04a-Luftfeuchtigkeit%20(Vergleich).html">https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/04a-Luftfeuchtigkeit%20(Vergleich).html</a>
2-Data Understanding	04b-Luftfeuchtigkeit	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/04b-Luftfeuchtigkeit.ipynb">https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/04b-Luftfeuchtigkeit.ipynb</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/04b-Luftfeuchtigkeit.html">https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/04b-Luftfeuchtigkeit.html</a>

2-Data Understanding	05-Wetter	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/05-Wetter.ipynb">https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/05-Wetter.ipynb</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/05-Wetter.html">https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/05-Wetter.html</a>
2-Data Understanding	06a-Niederschlag (Vergleich)	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/06a-Niederschlag%20(Vergleich).ipynb">https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/06a-Niederschlag%20(Vergleich).ipynb</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/06a-Niederschlag%20(Vergleich).html">https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/06a-Niederschlag%20(Vergleich).html</a>
2-Data Understanding	06b-Niederschlag	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/06b-Niederschlag.ipynb">https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/06b-Niederschlag.ipynb</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/06b-Niederschlag.html">https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/06b-Niederschlag.html</a>
2-Data Understanding	07a-Windgeschwindigkeit (Vergleich)	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/07a-Windgeschwindigkeit%20(Vergleich).ipynb">https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/07a-Windgeschwindigkeit%20(Vergleich).ipynb</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/07a-Windgeschwindigkeit%20(Vergleich).html">https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/07a-Windgeschwindigkeit%20(Vergleich).html</a>
2-Data Understanding	07b-Windgeschwindigkeit	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/07b-Windgeschwindigkeit.ipynb">https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/07b-Windgeschwindigkeit.ipynb</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/07b-Windgeschwindigkeit.html">https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/07b-Windgeschwindigkeit.html</a>
2-Data Understanding	08a-Sichtweite (Vergleich)	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-">https://github.com/ArneDecker/Masterthesis/blob/main/2-</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-">https://github.com/ArneDecker/Masterthesis/blob/main/2-</a>

		<a href="#">Data%20Understanding/08a-Sichtweite%20(Vergleich).ipynb</a>	<a href="#">Data%20Understanding/08a-Sichtweite%20(Vergleich).html</a>
2-Data Understanding	08b-Sichtweite	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/08b-Sichtweite.ipynb">https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/08b-Sichtweite.ipynb</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/08b-Sichtweite.html">https://github.com/ArneDecker/Masterthesis/blob/main/2-Data%20Understanding/08b-Sichtweite.html</a>
3-Data Preparation	01-Data Preparation (Training und Test)	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/3-Data%20Preparation/01-Data%20Preparation%20(Training%20und%20Test).ipynb">https://github.com/ArneDecker/Masterthesis/blob/main/3-Data%20Preparation/01-Data%20Preparation%20(Training%20und%20Test).ipynb</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/3-Data%20Preparation/01-Data%20Preparation%20(Training%20und%20Test).html">https://github.com/ArneDecker/Masterthesis/blob/main/3-Data%20Preparation/01-Data%20Preparation%20(Training%20und%20Test).html</a>
3-Data Preparation	02-Data Preparation (Wettervorhersage)	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/3-Data%20Preparation/02-Data%20Preparation%20(Wettervorhersage).ipynb">https://github.com/ArneDecker/Masterthesis/blob/main/3-Data%20Preparation/02-Data%20Preparation%20(Wettervorhersage).ipynb</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/3-Data%20Preparation/02-Data%20Preparation%20(Wettervorhersage).html">https://github.com/ArneDecker/Masterthesis/blob/main/3-Data%20Preparation/02-Data%20Preparation%20(Wettervorhersage).html</a>
3-Data Preparation	03-Rücksprung aus Modeling	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/3-Data%20Preparation/03-R%C3%BCcksprung%20aus%20Modeling.ipynb">https://github.com/ArneDecker/Masterthesis/blob/main/3-Data%20Preparation/03-R%C3%BCcksprung%20aus%20Modeling.ipynb</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/3-Data%20Preparation/03-R%C3%BCcksprung%20aus%20Modeling.html">https://github.com/ArneDecker/Masterthesis/blob/main/3-Data%20Preparation/03-R%C3%BCcksprung%20aus%20Modeling.html</a>

4-Modeling	01-Analyse	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/4-Modeling/01-Analyse.ipynb">https://github.com/ArneDecker/Masterthesis/blob/main/4-Modeling/01-Analyse.ipynb</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/4-Modeling/01-Analyse.html">https://github.com/ArneDecker/Masterthesis/blob/main/4-Modeling/01-Analyse.html</a>
4-Modeling	02a-ARIMA (endog)	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/4-Modeling/02a-ARIMA%20(endogen).ipynb">https://github.com/ArneDecker/Masterthesis/blob/main/4-Modeling/02a-ARIMA%20(endogen).ipynb</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/4-Modeling/02a-ARIMA%20(endogen).html">https://github.com/ArneDecker/Masterthesis/blob/main/4-Modeling/02a-ARIMA%20(endogen).html</a>
4-Modeling	02b-ARIMA (exogen)	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/4-Modeling/02b-ARIMA%20(exogen).ipynb">https://github.com/ArneDecker/Masterthesis/blob/main/4-Modeling/02b-ARIMA%20(exogen).ipynb</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/4-Modeling/02b-ARIMA%20(exogen).html">https://github.com/ArneDecker/Masterthesis/blob/main/4-Modeling/02b-ARIMA%20(exogen).html</a>
4-Modeling	02c-ARIMA (Verbesserung)	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/4-Modeling/02c-ARIMA%20(Verbesserung).ipynb">https://github.com/ArneDecker/Masterthesis/blob/main/4-Modeling/02c-ARIMA%20(Verbesserung).ipynb</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/4-Modeling/02c-ARIMA%20(Verbesserung).html">https://github.com/ArneDecker/Masterthesis/blob/main/4-Modeling/02c-ARIMA%20(Verbesserung).html</a>
4-Modeling	02d-ARIMA (Feiertage)	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/4-Modeling/02d-ARIMA%20(Feiertage).ipynb">https://github.com/ArneDecker/Masterthesis/blob/main/4-Modeling/02d-ARIMA%20(Feiertage).ipynb</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/4-Modeling/02d-ARIMA%20(Feiertage).html">https://github.com/ArneDecker/Masterthesis/blob/main/4-Modeling/02d-ARIMA%20(Feiertage).html</a>
4-Modeling	02e-ARIMA (Zwischenfazit)	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/4-Modeling/02e-ARIMA%20(Zwischenfazit).ipynb">https://github.com/ArneDecker/Masterthesis/blob/main/4-Modeling/02e-ARIMA%20(Zwischenfazit).ipynb</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/4-Modeling/02e-ARIMA%20(Zwischenfazit).html">https://github.com/ArneDecker/Masterthesis/blob/main/4-Modeling/02e-ARIMA%20(Zwischenfazit).html</a>

4-Modeling/Experimentelle Ansätze	01-LSTM (endogen)	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/4-Modeling/Experimentelle%20Ans%C3%A4tze/01-LSTM%20(endogen).ipynb">https://github.com/ArneDecker/Masterthesis/blob/main/4-Modeling/Experimentelle%20Ans%C3%A4tze/01-LSTM%20(endogen).ipynb</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/4-Modeling/Experimentelle%20Ans%C3%A4tze/01-LSTM%20(endogen).html">https://github.com/ArneDecker/Masterthesis/blob/main/4-Modeling/Experimentelle%20Ans%C3%A4tze/01-LSTM%20(endogen).html</a>
4-Modeling/Experimentelle Ansätze	02-LSTM (einfach exogen)	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/4-Modeling/Experimentelle%20Ans%C3%A4tze/02-LSTM%20(einfach%20exogen).ipynb">https://github.com/ArneDecker/Masterthesis/blob/main/4-Modeling/Experimentelle%20Ans%C3%A4tze/02-LSTM%20(einfach%20exogen).ipynb</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/4-Modeling/Experimentelle%20Ans%C3%A4tze/02-LSTM%20(einfach%20exogen).html">https://github.com/ArneDecker/Masterthesis/blob/main/4-Modeling/Experimentelle%20Ans%C3%A4tze/02-LSTM%20(einfach%20exogen).html</a>
5-Evaluation	01-Vergleiche ARIMA und LSTM	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/5-Evaluation/01-Vergleich%20ARIMA%20und%20LSTM.ipynb">https://github.com/ArneDecker/Masterthesis/blob/main/5-Evaluation/01-Vergleich%20ARIMA%20und%20LSTM.ipynb</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/5-Evaluation/01-Vergleich%20ARIMA%20und%20LSTM.html">https://github.com/ArneDecker/Masterthesis/blob/main/5-Evaluation/01-Vergleich%20ARIMA%20und%20LSTM.html</a>
5-Evaluation	02-Test mit Wettervorhersage	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/5-Evaluation/02-Test%20mit%20Wettervorhersage.ipynb">https://github.com/ArneDecker/Masterthesis/blob/main/5-Evaluation/02-Test%20mit%20Wettervorhersage.ipynb</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/5-Evaluation/02-Test%20mit%20Wettervorhersage.html">https://github.com/ArneDecker/Masterthesis/blob/main/5-Evaluation/02-Test%20mit%20Wettervorhersage.html</a>
6-Deployment	data_preparation	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/6-Deployment/python/data_preparation.ipynb">https://github.com/ArneDecker/Masterthesis/blob/main/6-Deployment/python/data_preparation.ipynb</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/6-Deployment/python/data_preparation.html">https://github.com/ArneDecker/Masterthesis/blob/main/6-Deployment/python/data_preparation.html</a>

## Anhang

6-Deployment	model_creation	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/6-Deployment/python/forecasting.ipynb">https://github.com/ArneDecker/Masterthesis/blob/main/6-Deployment/python/forecasting.ipynb</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/6-Deployment/python/forecasting.html">https://github.com/ArneDecker/Masterthesis/blob/main/6-Deployment/python/forecasting.html</a>
6-Deployment	forecasting	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/6-Deployment/python/model_creation.ipynb">https://github.com/ArneDecker/Masterthesis/blob/main/6-Deployment/python/model_creation.ipynb</a>	<a href="https://github.com/ArneDecker/Masterthesis/blob/main/6-Deployment/python/model_creation.html">https://github.com/ArneDecker/Masterthesis/blob/main/6-Deployment/python/model_creation.html</a>

Die Anwendung mit der Benutzeroberfläche ist in folgendem Ordner hinterlegt:

<https://github.com/ArneDecker/Masterthesis/tree/main/6-Deployment/Vorhersage>

## Anhang 2: Verwendete Software

Für die Entwicklung wurde folgende Software verwendet:

Als Entwicklungsumgebung für die Notebooks wurde Anaconda Individual Edition (Version 2021.05 für 64-Bit) verwendet. Als Editor wurde JupyterLab (Version 3.0.14) in Microsoft Edge (Version 98.0.1108.43 für 64-Bit) verwendet. Es wurde Anaconda Python (Version 3.8.8 MSC v.1916 AMD64 für 64-Bit) verwendet. Die Umgebung war auf Microsoft Windows 10 Home (Version 10.0.19042 Build 19042 für 64-Bit) installiert. Die Benutzeroberfläche wurde in C# im .NET Framework (Version 4.7.2 für 64-Bit) mit Microsoft Visual Studio Community Edition (Version 17.0.5 für 64-Bit) entwickelt.

Die folgende Tabelle gibt eine Übersicht über die verwendete Software

Name	Verwendung	Version	Webseite/Dokumentation
Microsoft Windows 10 Home	Betriebssystem	10.0.19042 Build 19042 (64-Bit)	<a href="https://www.microsoft.com/de-de/windows/get-windows-10">https://www.microsoft.com/de-de/windows/get-windows-10</a>
Microsoft Visual Studio Community Edition	Entwicklungsumgebung/Editor für Oberfläche	17.0.5 (64-Bit)	<a href="https://visualstudio.microsoft.com/de/">https://visualstudio.microsoft.com/de/</a>
.NET Framework	Entwicklungsumgebung/Programmiersprache	4.7.2 (64-Bit)	<a href="https://dotnet.microsoft.com/en-us/download/dotnet-framework">https://dotnet.microsoft.com/en-us/download/dotnet-framework</a>
IronPython	C#-Bibliothek	2.7.12	<a href="https://ironpython.net/">https://ironpython.net/</a>
Microsoft Edge	Browser	98.0.1108.43 (64-Bit)	<a href="https://www.microsoft.com/de-de/edge?r=1">https://www.microsoft.com/de-de/edge?r=1</a>
JupyterLab	Editor	3.0.14 (64-Bit)	<a href="https://jupyter.org/">https://jupyter.org/</a>
Anaconda Individual Edition	Entwicklungsumgebung	2021.05 (64-Bit)	<a href="https://www.anaconda.com/products/individual">https://www.anaconda.com/products/individual</a>
Python	Programmiersprache für Notebooks	3.8.8 MSC v.1916 64 bit (AMD64)	<a href="https://www.python.org/">https://www.python.org/</a>
pandas	Python-Bibliothek	1.2.4	<a href="https://pandas.pydata.org/">https://pandas.pydata.org/</a>

matplotlib	Python-Bibliothek	3.3.4	<a href="https://matplotlib.org/">https://matplotlib.org/</a>
seaborn	Python-Bibliothek	0.11.1	<a href="https://seaborn.pydata.org/">https://seaborn.pydata.org/</a>
NumPy (numpy)	Python-Bibliothek	1.19.5	<a href="https://numpy.org/">https://numpy.org/</a>
SciPy (scipy)	Python-Bibliothek	1.7.1	<a href="https://scipy.org/">https://scipy.org/</a>
statsmodels	Python-Bibliothek	0.13.0	<a href="https://www.statsmodels.org/stable/index.html">https://www.statsmodels.org/stable/index.html</a>
scikit-learn	Python-Bibliothek	0.24.1	<a href="https://scikit-learn.org/stable/">https://scikit-learn.org/stable/</a>
pmdarima	Python-Bibliothek	1.8.2	<a href="http://alkaline-ml.com/pmdarima/">http://alkaline-ml.com/pmdarima/</a>
TensorFlow	Python-Bibliothek	2.7.0	<a href="https://www.tensorflow.org/">https://www.tensorflow.org/</a>
Keras	Python-Bibliothek	2.7.0	<a href="https://keras.io/">https://keras.io/</a>

Python-Bibliotheken:

**pandas:**

pandas development team, T. (2020). pandas-dev/pandas: Pandas (latest) [Computer software]. Zenodo. <https://doi.org/10.5281/zenodo.3509134>

Stéfan van der Walt (Ed.). (2010). Data Structures for Statistical Computing in Python. In Proceedings of the 9th Python in Science Conference (pp. 56–61). <https://doi.org/10.25080/Majora-92bf1922-00a>

**matplotlib:**

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3), 90–95.  
<https://doi.org/10.1109/MCSE.2007.55>

**seaborn:**

Waskom, M. L. (2021). seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60), 3021.  
<https://doi.org/10.21105/joss.03021>

**NumPy:**

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., ... Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825), 357–362. <https://doi.org/10.1038/s41586-020-2649-2>

**SciPy:**

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., ... SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17, 261–272.  
<https://doi.org/10.1038/s41592-019-0686-2>

**statsmodels:**

Seabold, S., & Perktold, J. (2010). statsmodels: Econometric and statistical modeling with python. 9th Python in Science Conference.

**scikit-learn:**

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., & Varoquaux, G. (2013). API design for machine learning software: experiences from the scikit-learn project. *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 108–122.

**pmdarima:**

Smith, T. G., & others. (2017–). pmdarima: ARIMA estimators for Python. <http://www.alkaline-ml.com/pmdarima>

**TensorFlow:**

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., ... Zheng, X. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. <https://www.tensorflow.org/>

**Keras:**

Chollet, F., & others. (2015). Keras. <https://keras.io>.

**Quellenverzeichnis**

## Monographien

*Aggarwal, C.* (Neural Networks, 2018):

Neural Networks and Deep Learning, Cham: Springer, 2018

*Ahrens, C., Henson, R.* (Meteorologie, 2021):

Meteorology Today: An Introduction to Weather, Climate, and the Environment, Boston: Cengage Learning, 2021

*Ameisen, E.* (ML Applications, 2020):

Building Machine Learning Powered Applications, Sebastopol: O'Reilly, 2020

*Audevert, A., Banachewicz, K., Massaron, L.*, (Machine Learning Cookbook, 2021):

Machine Learning Using TensorFlow Cookbook, Birmingham: Packt Publishing

*Auffarth, B.* (Machine Learning for Time Series, 2021):

Machine Learning for Time-Series with Python, Birmingham: Packt Publishing, 2021

*Babcock, J., Bali, R.* (Generative AI, 2021):

Generative AI with Python and TensorFlow 2, Birmingham: Packt Publishing, 2021

*Berk, R.* (Statistisches Lernen, 2020):

Statistical Learning from a Regression Perspective, 3. Auflage, Cham: Springer, 2020

*Box, G., Jenkins, G., Reinsel, G., Ljung, G.* (Time Series Analysis, 2016):

Time Series Analysis, 5. Auflage, Hoboken: Wiley, 2016

*Chatfield, C., Xing, H.* (Time Series Analysis, 2019):

The Analysis of Time Series, 7. Auflage, Boca Raton: CRC Press, 2019

*Claster, W.* (Mathematik, 2020):

Mathematics and Programming for Machine Learning with R, Boca Raton: CRC Press, 2020

*Deshpande, A., Kumar, M.* (AI for Big Data, 2018):

Artificial Intelligence for Big Data, Birmingham: Packt Publishing, 2018

*Frochte, J.* (Grundlagen des Machine Learning, 2020):

Maschinelles Lernen – Grundlagen und Algorithmen in Python, 3. Auflage, München: Hanser Verlag, 2020

*Géron, A.* (Machine Learning, 2020):

Praxiseinstieg Machine Learning mit Scikit-Learn, Keras und TensorFlow, 2. Auflage, Heidelberg: dpunkt Verlag, 2020

*Ghavami, P.* (Analytics Methods, 2019):

Big Data Analytics Methods, 2. Auflage, Berlin: De Gruyter, 2019

*Häckel, H.* (Meteorologie, 2021):

Meteorologie, 9. Auflage, Stuttgart: UTB Verlag, 2021

*Hawamdeh, S., Chang H.-C.* (Knowledge Management, 2018):

Analytics and Knowledge Management, Boca Raton: CRC Press, 2018

*Hering, E., Schönfelder, G.* (Wissenschaft und Technik, 2018):

Sensoren in Wissenschaft und Technik, 2. Auflage, Wiesbaden: Springer Vie-weg, 2018

*Hirschle, J.* (Machine Learning für Zeitreihen, 2020):

Machine Learning für Zeitreihen, München: Hanser Verlag, 2020

*Hyndman, R. J., Athanasopoulos, G.* (Forecasting, 2018):

Forecasting: Principles and Practice, 2. Auflage, OTexts, 2018

*Jansen, S.* (Machine Learning for Trading, 2020):

Machine Learning for Algorithmic Trading, 2. Auflage, Birmingham: Packt Publishing, 2020

*Khosrow-Pour, M.* (Advanced Methodologies, 2018):

Advanced Methodologies and Technologies in Network Architecture, Mobile Computing, and Data Analytics, Hershey: IGI Global, 2018

*Körner, C., Waaijer, K.* (Machine Learning, 2020):

Mastering Azure Machine Learning, Birmingham: Packt Publishing, 2020

*Korstanje, J.* (Advanced Forecasting, 2021):

Advanced Forecasting with Python: With State-of-the-Art-Models Including LSTMs, Facebook's Prophet, and Amazon's DeepAR, New York City: Apress, 2021

*Lahshmanan, V., Robinson, S., Munn, M.* (Machine Learning Design, 2020):

Machine Learning Design Patterns, Sebastopol: O'Reilly, 2020

*Lazzeri, F.* (Machine Learning, 2021):

Machine Learning for Time Series Forecasting with Python, Indianapolis: Wiley, 2021

*Miles, T.* (Applied Time Series Analysis, 2019):

Applied Time Series Analysis, London: Academic Press, 2019

*Montgomery, D., Jennings, C., Kulahci, M.* (Time Series, 2015):

Introduction to Time Series Analysis and Forecasting, 2.Auflage, Hoboken: Wiley, 2015

*Niederhausen, H., Burkert, A.* (Elektrischer Strom, 2014):

Elektrischer Strom, Wiesbaden: Springer Verlag, 2014

*Nisbet, R., Miner, G., Yale, K* (Statistische Analysen, 2017):

Handbook of Statistical Analysis and Data Mining Applications, 2. Auflage, London: Academic Press, 2017

*Pal, A., Prakash, P.* (Practical Time Series Analysis, 2017):

Practical Time Series Analysis, Birmingham: Packt Publishing, 2017

*Patel, A., Vishwas, B. V.* (Time Series Analysis Python, 2020):

Hands-on Time Series Analysis with Python, New York City: Apress, 2020

*Patterson, J., Gibson, A.* (Deep Learning, 2017):

Deep Learning, Sebastopol: O'Reilly, 2017

*Plaat, A.* (Reinforcement Learning, 2020):

Learning to Play – Reinforcement Learning and Games, Basel: Springer Nature, 2020

*Putatunda, S.* (Practical Machine Learning, 2021):

Practical Machine Learning for Streaming Data with Python, New York City: Apress, 2021

*Raschka, S., Mirjalili, V.* (Machine Learning mit Python, 2021):

Machine Learning mit Python und Keras, TensorFlow 2 und Scikit-learn, 3. Auflage, Frechen: mitp Verlag, 2021

*Schiffer, H.-W.* (Energiemarkt Deutschland, 2018):

Energiemarkt Deutschland, 1. Auflage, Wiesbaden: Springer Vieweg, 2018

*Shmueli, G., Lichtendahl, K. C.* (Time Series Forecasting, 2016):

Practical Time Series Forecasting with R, 2. Auflage, Green Cove Springs: Axelrod Schnall Publishers, 2016

## Aufsätze und Artikel

*Athiyarath, S., Paul, M., Krishnaswamy, S.* (Forecasting Techniques, 2020):

A Comparative Study and Analysis of Time Series Forecasting Techniques in:  
*Springer Nature* (Hrsg.), SN Computer Science, 2020, S. 174

*Chen, G., Guo, X.* (Oversampling ARIMA, 2021):

Research on Oversampling Algorithm for Imbalanced Datasets Based On ARIMA Model in: *Chinese Control and Decision Conference (CCDC)* (Hrsg.), 2021 33rd Chinese Control and Decision Conference (CCDC), 2021, S. 2384-2389

*Hochreiter, S., Schmidhuber, J.* (LSTM, 1997):

Long Short-term Memory in: *MIT Press* (Hrsg.), Neural Computation 9, 1997, S. 1735-1780

*Kang, J., Reiner, D.* (Weather and Electricity Consumption, 2021):

What is the effect of weather on household electricity consumption? Empirical evidence from Ireland in: *University of Cambridge Energy Policy Research Group* (Hrsg.), 2021

*Lederer, J.* (Aktivierungsfunktionen, 2021):

Activation Functions in Artificial Neural Networks: A Systematic Overview

*Maia-Silva, D., Kumar, R., Nateghi, R.* (Humidity and Electricity Demand, 2020):

The critical role of humidity in modeling summer electricity demand across the United States in: *Nature Communications* (Hrsg.), 2020

*Moniz, N., Branco, P., Torgo, L.* (Oversampling in Forecasting, 2017):

Resampling strategies for imbalanced time series forecasting in: *Springer Professional* (Hrsg.), International Journal of Data Science and Analytics 3, 2017, S. 161-181

*Shuai, M. et al.* (Folgen von Stromausfällen, 2018):

Review of Economic Loss Assessment of Power Outages in: *Procedia Computer Science* (Hrsg.), 2018

*Le Guen, V., Thome, N.* (Loss Functions, 2019):

Shape and Time Distortion Loss for Training Deep Time Series Forecasting Models, 2019

*Yao, J.* (Electricity and Temperature, 2021):

Electricity Consumption and Temperature: Evidence from Satellite Data, 2021

*Zhang, C., Liao, H., Mi, Z.* (Klimaeinfluss auf Stromverbrauch, 2019):

Climate impacts: temperature and electricity consumption in: *Natural Hazards* (Hrsg.), 2019

## Internet-Quellen

Die Internetquellen sind in GitHub im PDF- und HTML-Format archiviert:

<https://github.com/ArneDecker/Masterthesis/tree/main/Ausarbeitung/Internetquellen>

<https://github.com/ArneDecker/Masterthesis/blob/main/Ausarbeitung/Internetquellen/HTML.zip>

[https://github.com/ArneDecker/Masterthesis/blob/main/Ausarbeitung/Internetquellen/PDF\\_1.zip](https://github.com/ArneDecker/Masterthesis/blob/main/Ausarbeitung/Internetquellen/PDF_1.zip)

[https://github.com/ArneDecker/Masterthesis/blob/main/Ausarbeitung/Internetquellen/PDF\\_2.zip](https://github.com/ArneDecker/Masterthesis/blob/main/Ausarbeitung/Internetquellen/PDF_2.zip)

*alkaline-ml* (auto\_arima(), 2021):

pmdarima.arima.auto\_arima, <[https://alkaline-ml.com/pmdarima/modules/generated/pmdarima.arima.auto\\_arima.html](https://alkaline-ml.com/pmdarima/modules/generated/pmdarima.arima.auto_arima.html)>(2021) [Zugriff 2022-04-02]

*Amprion GmbH* (Amprion, 2021):

Amprion verbindet, <<https://www.amprion.net/Amprion/>> [Zugriff 2021-08-18]

*Aunkofer, B.* (Deep Learning, 2019):

Deep Learning, <<https://data-science-blog.com/blog/2019/01/13/training-eines-neurons-mit-dem-gradientenverfahren/>>(2019-01-13) [Zugriff 2021-12-20]

*Brownlee, J.* (Baseline Predictions, 2019):

How to Make Baseline Predictions for Time Series Forecasting with Python, <<https://machinelearningmastery.com/persistence-time-series-forecasting-with-python/>>(2019-08-21) [Zugriff 2022-02-10]

*Brownlee, J.* (Learning Rates, 2020):

Understand the Impact of Learning Rate on Neural Network Performance, <<https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/>>(2020-09-12) [Zugriff 2022-01-28]

*Brownlee, J.* (Oversampling, 2021):

Random Oversampling and Undersampling for Imbalanced Classification,  
<<https://machinelearningmastery.com/random-oversampling-and-under-sampling-for-imbalanced-classification/>>(2021-01-05) [Zugriff 2022-01-19]

*Brownlee, J.* (Rolling Forecast, 2017):

How to Create an ARIMA Model for Time Series Forecasting in Python,  
<<https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/>>(2017-01-09) [Zugriff 2022-02-10]

*Brownlee, J.* (Time Series Decomposition, 2020):

How to Decompose Time Series Data into Trend and Seasonality,  
<<https://machinelearningmastery.com/decompose-time-series-data-trend-seasonality/>>(2020-12-10) [Zugriff 2021-08-24]

*Bundesamt für Bevölkerungsschutz und Katastrophenhilfe* (Kritische Infrastruktur, 2021):

Kritische Infrastrukturen, <[https://www.kritis.bund.de/SubSites/Kritis/DE/Einfuehrung/einfuehrung\\_node.html](https://www.kritis.bund.de/SubSites/Kritis/DE/Einfuehrung/einfuehrung_node.html)>(2020) [Zugriff 2021-09-02]

*Bundesamt für Bevölkerungsschutz und Katastrophenhilfe* (Stromausfall, 2019):

Stromausfall, <<https://www.bbk.bund.de/SharedDocs/Downloads/DE/Mediathek/Publikationen/Buergerinformationen/stromausfall-vorsorge-selbst-hilfe.pdf?blob=publicationFile&v=9>>(2019-01) [Zugriff 2021-09-02]

*Bundesministerium für Wirtschaft und Energie* (Bruttostromverbrauch, 2016):

Was bedeutet "Bruttostromverbrauch"? , <<https://www.bmwi-energie-wende.de/EWD/Redaktion/Newsletter/2016/01/Meldung/direkt-erklaert.html>>(2016-01-12) [Zugriff 2022-01-02]

*Bundesministerium für Wirtschaft und Energie* (Erneuerbare Energien, 2022):

Erneuerbare Energien, <<https://www.bmwi.de/Redaktion/DE/Dossier/erneuerbare-energien.html>>(2021) [Zugriff 2022-01-19]

*Bundesministerium für Wirtschaft und Energie* (Konventionelle Energieträger, 2022):

Konventionelle Energieträger, <<https://www.bmwi.de/Redaktion/DE/Dossier/konventionelle-energietraeger.html>>(2021) [Zugriff 2022-01-17]

*Bundesministerium für Wirtschaft und Energie* (Stromnetz, 2022):

Ein Stromnetz für die Energiewende, <<https://www.bmwi.de/Redaktion/DE/Dossier/netze-und-netzausbau.html>>(2022) [Zugriff 2022-01-17]

*Bundesnetzagentur* (Ausgleichsenergie, 2022):

Ausgleichsenergie, <<https://www.smard.de/page/home/wiki-article/446/528>>(2022) [Zugriff 2022-01-19]

*Bundesnetzagentur* (Regelenergie, 2022):

Regelenergie, <[https://www.bundesnetzagentur.de/DE/Sachgebiete/ElektrizitaetundGas/Unternehmen\\_Institutionen/Versorgungssicherheit/Engpass-management/Regelenergie/start.html](https://www.bundesnetzagentur.de/DE/Sachgebiete/ElektrizitaetundGas/Unternehmen_Institutionen/Versorgungssicherheit/Engpass-management/Regelenergie/start.html)>(2022) [Zugriff 2022-02-08]

*Bundesnetzagentur* (Regelreserve, 2022):

Regelreserve, <<https://www.smard.de/page/home/wiki-article/446/396>>(2022) [Zugriff 2022-02-10]

*Bundesnetzagentur* (Stromerzeugung, 2022):

Stromerzeugung, <<https://www.smard.de/page/home/wiki-article/446/636>>(2022) [Zugriff 2022-02-10]

*Bundesnetzagentur* (Stromerzeugung im Sommer, 2017):

Stromerzeugung im Sommer – Erneuerbare mit Spitzenwerten, <<https://www.smard.de/page/home/topic-article/444/3678>>(2017-08-17) [Zugriff 2022-02-10]

*Bundesverband der Energie- und Wasserwirtschaft* (Energieversorgungsbericht, 2021):

Die Energieversorgung 2020 - Jahresbericht, <[https://www.bdew.de/documents/6851/Jahresbericht\\_2020\\_final\\_Aktualisierte\\_Fassung\\_10Mai2021.pdf](https://www.bdew.de/documents/6851/Jahresbericht_2020_final_Aktualisierte_Fassung_10Mai2021.pdf)>(2021-05-10) [Zugriff 2021-11-10]

*Bundesverband der Energie- und Wasserwirtschaft* (Stromverbrauch, 2021): Stromverbrauch in Deutschland nach Verbrauchergruppen 2020, <<https://www.bdew.de/service/daten-und-grafiken/stromverbrauch-deutschland-verbrauchergruppen/>>(2021-05-03) [Zugriff 2022-01-19]

*Bundesverband der Energie- und Wasserwirtschaft* (Stromverbrauch im Dezember, 2020):

Zehn mal weniger Strom..., <<https://www.bdew.de/presse/presseinformationen/zehn-mal-weniger-strom/>>(2020-11-27) [Zugriff 2022-01-11]

*Die Bundesregierung* (Energiewende, 2022):

## Quellenverzeichnis

Erneuerbare Energien - Ein neues Zeitalter hat begonnen, <<https://www.bundesregierung.de/breg-de/themen/energiewende/energie-erzeugen/erneuerbare-energien-317608>>(2022) [Zugriff 2022-01-20]

*Eder, S.* (Strom- und Energieverbrauch, 2020):

Strom- und Energieverbrauch im Lockdown: <<https://www.vdi-nachrichten.com/technik/energie/energie-und-stromverbrauch-in-deutschland-singen-2020-deutlich/>>(2020-11-04) [Zugriff 2022-01-10]

*Enerdata* (Stromverbrauchsprognose weltweit 2050, 2022):

Final electricity consumption, <<https://eneroutlook.enerdata.net/forecast-world-electricity-consumption.html>>(2022) [Zugriff 2022-02-10]

*Fraunhofer ISE* (Strommix, 2021):

Nettostromerzeugung in Deutschland 2020: erneuerbare Energien erstmals über 50 Prozent, <<https://www.ise.fraunhofer.de/de/presse-und-medien/news/2020/nettostromerzeugung-in-deutschland-2021-erneuerbare-energien-erstmals-ueber-50-prozent.html#:~:text=Die%20Bio-masse%20lag%20mit%2045,50%20Prozent%20der%20C3%20fentlichen%20Nettostromerzeugung>>(2021-01-04) [Zugriff 2022-01-20]

*IBM* (CRISP-DM Übersicht, 2021):

CRISP-DM-Hilfe – Übersicht, <<https://www.ibm.com/docs/de/spss-modeler/SaaS?topic=dm-crisp-help-overview>>(2021) [Zugriff 2021-09-08]

*IBM* (Neurale Netzwerke, 2020):

Neurale Netzwerke, <<https://www.ibm.com/de-de/cloud/learn/neural-networks>>(2020-08-17) [Zugriff 2021-12-15]

## Quellenverzeichnis

*IBM* (Recurrent Neural Networks, 2020):

Recurrent Neural Networks, <<https://www.ibm.com/cloud/learn/recurrent-neural-networks>>(2020-09-14) [Zugriff 2021-12-25]

*IBM* (Overfitting, 2021):

Overfitting, <<https://www.ibm.com/cloud/learn/overfitting>>(2021-03-03) [Zugriff 2021-11-30]

*IBM* (Underfitting, 2021):

Underfitting, <<https://www.ibm.com/cloud/learn/underfitting>>(2021-03-23) [Zugriff 2021-12-21]

*Institut für Weltwirtschaft (IfW Kiel)* (Stromverbrauch im Lockdown, 2020):

<<https://www.ifw-kiel.de/de/publikationen/medieninformationen/2020/corona-deutscher-stromverbrauch-deutlich-unter-normalniveau/>>(2020-09-04) [Zugriff 2022-02-10]

*Internationale Energieagentur* (Stromverbrauchsprognose weltweit 2030, 2020):

Global electricity demand by scenario, 2010-2030, <<https://www.iea.org/data-and-statistics/charts/global-electricity-demand-by-scenario-2010-2030>>(2020-10-12) [Zugriff 2022-01-02]

*Luber, S., Litzel, N.* (CRISP-DM, 2019):

Was ist CRISP-DM?, <<https://www.bigdata-insider.de/was-ist-crisp-dm-a-815478/>>(2019-04-10) [Zugriff 2022-02-08]

*Luber, S., Litzel, N.* (LSTM, 2018):

Was ist ein Long Short-Term Memory?, <[https://www.bigdata-insider.de/was-  
ist-ein-long-short-term-memory-a-774848/](https://www.bigdata-insider.de/was-ist-ein-long-short-term-memory-a-774848/)>(2018-11-12) [Zugriff 2022-01-27]

*Luber, S., Litzel, N.* (rekurrente neuronale Netze, 2019):

Was ist ein rekurrentes neuronales Netz (RNN)?, <[https://www.bigdata-insi-  
der.de/was-ist-ein-rekurrentes-neuronales-netz-rnn-a-843274/](https://www.bigdata-insider.de/was-ist-ein-rekurrentes-neuronales-netz-rnn-a-843274/)>(2019-07-05)  
[Zugriff 2022-01-25]

*Mitteldeutscher Rundfunk* (Stromverbrauch privater Haushalte, 2021):

<[https://www.mdr.de/nachrichten/sachsen/corona-erhoehter-stromver-  
brauch-lockdown-100.html](https://www.mdr.de/nachrichten/sachsen/corona-erhoehter-stromver-<br/>brauch-lockdown-100.html)>(2021-02-11) [Zugriff 2022-02-10]

*Nischkauer, H.* (Temperaturabhängigkeit des Stromverbrauchs, 2005):

Temperaturabhängigkeit des Strom- und Gasverbrauchs,  
<[https://www.e-con-  
trol.at/documents/1785851/1811528/WP15\\_TEMPVER\\_DOKU.pdf](https://www.e-con-<br/>trol.at/documents/1785851/1811528/WP15_TEMPVER_DOKU.pdf)>(2005-  
11) [Zugriff 2022-01-05]

*Paschotta, Dr. R* (Regelenergie, 2021):

Regelenergie, <<https://www.energie-lexikon.info/regelenergie.html>>(2021-  
07-05) [Zugriff 2021-08-21]

*Python Software Foundation* (Python Applications, 2022):

Applications for Python, <<https://www.python.org/about/apps/>>(2022) [Zugriff  
2022-02-10]

*Python Software Foundation* (Python, 2022):

General Python FAQ, <<https://docs.python.org/3/faq/general.html>>(2022) [Zugriff 2022-02-10]

Rosebrock, A. (Validation Loss, 2019):

Why is my validation loss lower than my training loss?,  
<<https://www.pyimagesearch.com/2019/10/14/why-is-my-validation-loss-lower-than-my-training-loss/>>(2019-10-14) [Zugriff 2022-01-28]

scikit-learn developers (Metriken, 2021):

Metrics and scoring: quantifying the quality of predictions, <[https://scikit-learn.org/stable/modules/model\\_evaluation.html#r2-score](https://scikit-learn.org/stable/modules/model_evaluation.html#r2-score)>(2021) [Zugriff 2021-12-28]

scikit-learn developers (Overfitting, 2021):

Underfitting vs. Overfitting, <[https://scikit-learn.org/stable/auto\\_examples/model\\_selection/plot\\_underfitting\\_overfitting.html](https://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html)>[Zugriff 2021-12-21]

Statista (Anzahl Stromnetzbetreiber, 2021):

Anzahl der Stromnetzbetreiber in Deutschland in den Jahren 2010 bis 2020,  
<<https://de.statista.com/statistik/daten/studie/152937/umfrage/anzahl-der-stromnetzbetreiber-in-deutschland-seit-2006/>>(2021-01-28) [Zugriff 2022-01-18]

Statista (Einwohnerzahlen, 2021):

Einwohnerzahl der größten Städte in Baden-Württemberg im Jahr 2020,  
<<https://de.statista.com/statistik/daten/studie/1066733/umfrage/groesste-staedte-in-baden-wuerttemberg/>>(2021-06) [Zugriff 2022-01-10]

Statista (Kausalität, 2021):

Definition Kausalität, <<https://de.statista.com/statistik/lexikon/definition/74/kausalitaet/>>(2021) [Zugriff 2021-12-10]

*Statista* (Korrelation, 2021):

Definition Korrelation, <<https://de.statista.com/statistik/lexikon/definition/77/korrelation/>>(2021) [Zugriff 2021-08-24]

*Statista* (Korrelationskoeffizient, 2021):

Definition Korrelationskoeffizient, <<https://de.statista.com/statistik/lexikon/definition/78/korrelationskoeffizient/>>(2021) [Zugriff 2021-08-24]

*Statista* (Nettostromerzeugung, 2021):

Nettostromerzeugung in Deutschland in den Jahren 2005 bis 2020, <<https://de.statista.com/statistik/daten/studie/307090/umfrage/nettostromerzeugung-in-deutschland/>>(2021-04) [Zugriff 2022-01-19]

*Statista* (Nettostromverbrauch, 2022):

Nettostromverbrauch in Deutschland in den Jahren 1991 bis 2020, <<https://de.statista.com/statistik/daten/studie/164149/umfrage/netto-stromverbrauch-in-deutschland-seit-1999/>>(2022-01-20) [Zugriff 2022-01-19]

*Statista* (Nettostromverbrauch weltweit, 2021):

Net consumption of electricity worldwide in select years from 1980 to 2018, <<https://www.statista.com/statistics/280704/world-power-consumption/>>(2021-07-05) [Zugriff 2022-02-02]

*Statista* (Scheinkorrelation, 2021):

Definition Scheinkorrelation, <<https://de.statista.com/statistik/lexikon/definition/118/scheinkorrelation/>>(2021) [Zugriff 2021-12-10]

*Statista* (Stromausstauschsaldo, 2020):

Stromausstauschsaldo Deutschlands in den Jahren 1990 bis 2020, <<https://de.statista.com/statistik/daten/studie/153533/umfrage/stromimportsaldo-von-deutschland-seit-1990/>>(2020-12) [Zugriff 2022-01-19]

*Statista* (Strommix, 2017):

Deutschlands Strom wird immer grüner, <<https://de.statista.com/infografik/11733/anteil-erneuerbarer-energien-an-der-stromerzeugung-in-deutschland/>>(2017-11-08) [Zugriff 2021-02-10]

*Statistics Solutions* (Korrelationskoeffizienten, 2022):

Correlation (Pearson, Kendall, Spearman), <<https://www.statisticssolutions.com/free-resources/directory-of-statistical-analyses/correlation-pearson-kendall-spearman/>>(2022) [Zugriff 2022-01-10]

*Statistisches Bundesamt (Destatis)* (Energieerzeugung, 2022):

Erzeugung, <[https://www.destatis.de/DE/Themen/Branchen-Unternehmen/Energie/Erzeugung/\\_inhalt.html](https://www.destatis.de/DE/Themen/Branchen-Unternehmen/Energie/Erzeugung/_inhalt.html)>(2021) [Zugriff 2022-02-10]

*Statistisches Landesamt Baden-Württemberg* (Karten, 2022):

Interaktive Karten, <<https://www.statistik-bw.de/Intermaptiv/>>(2022) [Zugriff 2022-01-10]

*statsmodels* (SARIMAX, 2022):

statsmodels.tsa.statespace.sarimax.SARIMAX,  
<[https://www.statsmodels.org/dev/examples/notebooks/generated/statespace\\_sarimax\\_stata.html](https://www.statsmodels.org/dev/examples/notebooks/generated/statespace_sarimax_stata.html)>(2022-02-09) [Zugriff 2022-02-10]

*statsmodels* (Stationarity, 2022):

Stationarity and detrending (ADF/KPSS), <[https://www.statsmodels.org/dev/examples/notebooks/generated/stationarity\\_detrending\\_adf\\_kpss.html](https://www.statsmodels.org/dev/examples/notebooks/generated/stationarity_detrending_adf_kpss.html)>(2022-02-09) [Zugriff 2022-02-10]

*Tableau Software* (Time Series Forecasting, 2022):

Time Series Forecasting: Definition, Applications, and Examples,  
<<https://www.tableau.com/learn/articles/time-series-forecasting>>(2022)  
[2022-01-29]

*TenneT GmbH* (TenneT, 2022):

Über TenneT, <<https://www.tennet.eu/de/unternehmen/profil/ueber-tennet/>>  
(2022) [Zugriff 2022-01-18]

*TensorFlow.org* (LSTM, 2022):

tf.keras.layers.LSTM, <[https://www.tensorflow.org/api\\_docs/python/tf/keras/layers/LSTM](https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM)>(2022-02-03) [Zugriff 2022-02-05]

*TensorFlow.org* (TensorFlow, 2022):

Why TensorFlow, <<https://www.tensorflow.org/about>>(2022) [Zugriff 2022-02-10]

*TensorFlow.org* (Time Series Forecasting, 2022):

Time Series Forecasting, <[https://www.tensorflow.org/tutorials/structured\\_data/time\\_series](https://www.tensorflow.org/tutorials/structured_data/time_series)>(2022-01-26) [Zugriff 2022-01-25]

*TIOBE Software BV* (TIOBE Index August 2021, 2021):

TIOBE Index for August 2021, <<https://www.tiobe.com/tiobe-index>>(2021-09) [Zugriff 2021-09-08]

*TransnetBW GmbH* (TransnetBW, 2021):

Wer wir sind, <<https://www.transnetbw.de/de/unternehmen/portraet/wer-wir-sind>> [Zugriff 2021-08-18]

*WeatherAPI.com* (API-Dokumentation, 2022):

<<https://www.weatherapi.com/docs/>>(2022) [Zugriff 2022-02-10]

*50Hertz GmbH* (50Hertz, 2021):

Das ist 50Hertz, <<https://www.50hertz.com/de/Unternehmen>>(2021) [Zugriff 2021-08-18]

## **Ehrenwörtliche Erklärung**

Hiermit versichere ich, dass die vorliegende Arbeit von mir selbstständig und ohne unerlaubte Hilfe angefertigt worden ist, insbesondere dass ich alle Stellen, die wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen sind, durch Zitate als solche gekennzeichnet habe. Ich versichere auch, dass die von mir eingereichte schriftliche Version mit der digitalen Version übereinstimmt. Weiterhin erkläre ich, dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde/Prüfungsstelle vorgelegen hat. Ich erkläre mich damit einverstanden, dass die Arbeit der Öffentlichkeit zugänglich gemacht wird. Ich erkläre mich damit einverstanden, dass die Digitalversion dieser Arbeit zwecks Plagiatsprüfung auf die Server externer Anbieter hochgeladen werden darf. Die Plagiatsprüfung stellt keine Zurverfügungstellung für die Öffentlichkeit dar.

Langenfeld, den 12.02.2022



Arne Decker