Hochschule Düsseldorf
University of Applied Sciences

Fachbereich Medien
Faculty of Media

Smart Energy- Dezentrale Energieversorgung

Prognose des Strombedarfs durch Einsatz von Maschine Learning am Beispiel New Hampshire

Suvapna Maheswaran

710533

Masterarbeit im Studiengang

M.Sc. Medieninformatik

November 2022

Betreuer Innen:

Prof. Gabi Schwab-Trapp

Prof. Dr. Florian Huber

HSD

Abschlussarbeit

1) im Studiengang M.Sc. Medieninformatik / PO 2010
(BSc, BEng, MSc) Name

2) Kandidat/in Maheswaran, Suvapna Matrikelnummer: 710533
(Name, Vorname)

Anschrift: Telefon: (0178 6716728

Gutleutstr. 16 40223 Düsseldorf
Straße und Hausnummer PLZ Ort

3) Die Arbeit wird angefertigt ☒ in der HSD

☐ extern bei ..... Telefon: (.....) .....

....
Straße und Hausnummer PLZ Ort

4) Thema der Arbeit (In Druckschrift):

Smart Enery - Dezentrale Energieversorgung, Prognose des Strombed
durch Einsatz von Machine Learning am Beispiel von New Hampshi

5) Prüfer/innen:

1. Prof. Gabi Schwab-Trapp (akademischer Grad) Name Erstprüfer/in
2. Prof. Dr. Florian Huber (akademischer Grad) Name Zweitprüfer/in

---

6) Vom Prüfungsamt auszufüllen:

Die Arbeit ist ausgegeben worden am 01.07.22 HZ 2ü

Die Arbeit ist abzugeben spätestens am 30.12.22 HZ 3ü

Hochschule Düsseldorf
Prüfungs-Support
Münsterstr. 156
40476 Düsseldorf
Tel. +49 (0) 211 4351 - 0
www.hs-duesseldorf.de

Der Kandidat ist zurückgetreten am ..... HZ.....

Die Frist wird auf Antrag vom ..... verlängert bis ..... HZ.....

Die Arbeit ist im StudienbCro abgegeben worden am ..... HZ.....

Die Arbeit wurde aufgegeben mit Datum des Poststempels vom ..... HZ.....

Bemerkungen:

Dokumentation Bachelorarbeit / Masterarbeit FB M HSD v31 01/2019

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, dass ich diese Arbeit eigenständig verfasst und keine anderen als die angegebenen und bei Zitaten kenntlich gemachten Quellen und Hilfsmittel benutzt habe.

_____

Datum, Unterschrift

Kontaktinformationen

Suvapna Maheswaran
Gurlittstraße 16, Zusatz:103, 40223 Düsseldorf
Suvapnamaheswaran@gmx.de

Zusammenfassung

Die vorliegende Masterarbeit dokumentiert die einzelnen Schritte der Konzeption und der Umsetzung einer Lastprognose an dem Beispiel New Hampshire. Die Motivation sowie die einzelnen Schritte der Konzeptentwicklung werden detailliert vorgestellt. Im Folgenden wird eine multivariate Zeitreihenanalyse mittels LSTM untersucht und mit einer univariaten Zeitreihenanalyse mittels LSTM und Random Forest verglichen. Die verwendeten Daten kombinieren das reale Verbraucherverhalten einer Ladezone mit Objekten, die Photovoltaikanlagen einschließen sowie die zugehörigen meteorologischen Daten aus dem Jahr 2019.

Inhaltsverzeichnis

Glossar

Bagging: Bootstrap-Aggregation, auch Bagging genannt, ist ein Meta-Algorithmus für maschinelles Lernen, der die Stabilität und Genauigkeit von Algorithmen für maschinelles Lernen in der statistischen Klassifikation und Regression verbessern soll.

Bootstrap-Datenprobe: Mittels Bagging erstellte Datenprobe aus einem Datensatz.

Dezentrale Energieversorgung: Systeme mit verteilten Energieressourcen (distributed energy resources, DER) bestehen aus kleinen technologischen Einheiten zur Stromerzeugung wie etwa Sonnenkollektoren oder Windrädern, die alternativen oder verbesserten traditionellen Strom liefern.

Feature: Ein Feature ist eine Eingabevariable - die x-Variable. Ein einfaches Projekt für maschinelles Lernen könnte ein einziges Merkmal verwenden, während ein anspruchsvolles Projekt für maschinelles Lernen hunderte/tausende Features von Features verwenden könnte

Flexibilität: In Zusammenhang mit der Energiewirtschaft bedeutet Flexibilität die Fähigkeit des Stromsystems, die Stromerzeugung oder den Stromverbrauch als Reaktion auf erwartete oder andere Schwankungen zu ändern.

Gradient: Als Gradient oder Gradienten (lateinisch gradiens, schreitend) wird der Verlauf der Änderung (Gefälle oder Anstieg) einer Größe auf einer bestimmten Strecke bezeichnet.

Label: Ein Label ist die Größe, die vorhergesagt wird - die y-Variable. Das Label könnte der zukünftige Weizenpreis, die Art des Tieres auf einem Bild, die Bedeutung eines Audioclips und vieles Weitere sein.

Lastprofil, Lastgang, Lastkurve: Je nach Zeitachse auch Tages- oder Jahresgang wird in der Energiewirtschaft der zeitlichen Verlauf der abgenommenen Leistung über eine zeitliche Periode bezeichnet.

Lastprognose: Unter Lastprognose wird in der Energiewirtschaft die Vorhersage des elektrischen Energieverbrauches eines bestimmten geografischen Gebietes bzw. eines Versorgungsnetzes bezeichnet.

Nachhaltige Entwicklung: Sie bezeichnet eine Entwicklung, die den Bedürfnissen der jetzigen Generation dient, ohne die Möglichkeiten künftiger Generationen zu gefährden.
Heute bezieht sich der Begriff in der Regel auf soziale, ökonomische und ökologische Aspekte der Nachhaltigkeit.

Optimerungsmodell: Ein Optimierungsmodell besteht aus einer Zielfunktion und mehreren Nebenbedingungen. Anhand der Zielfunktion werden alle verschiedenen Lösungsalternativen bewertet und daraus die optimale Lösung entworfen.

Verbrauchsstellen: Verbrauchsstellen sind die Punkte, an denen die Energie an die Endkunden des Käufers, aus denen sich die Endkundenlast zusammensetzt, geliefert und gemessen wird.

Verteilnetz: Das Verteilnetz ist das Stromnetz der niedrigsten Spannungsebene. Über das Stromverteilnetz sind die Endverbraucher, also die einzelnen Haushalte, an die Netze der höheren Spannungsebenen angebunden und werden mit Elektrizität versorgt.

Volatilität: Solarenergie und Windenergie sind in der Vergangenheit als volatile Energieträger bezeichnet worden, da die von ihnen gelieferte Energiemenge nicht mit 100-prozentiger Sicherheit vorhersagbar ist.

1 Einleitung

1.1 Motivation

Der Energiesektor unterliegt einem stetigen Wandel. Aktuell befindet sich unsere Gesellschaft vor der Aufgabe herkömmlicher Energiequellen wie Kohlekraft oder der Atomenergie durch erneuerbare Energien zu ersetzen. Angetrieben durch die Folgen des menschengemachten Klimawandels stehen insbesondere fossile Brennstoffe zur Stromerzeugung in der berechtigten Kritik. Neue Technologien wie z.B. Windenergie, Solarenergie, Elektrofahrzeuge und Wärmepumpen bieten dabei die Chance CO2 einzusparen und gewinnen daher zunehmend an Relevanz. Diese sind fundamental für eine nachhaltige Energieerzeugung sowie einen effizienten Energieverbrauch. Dennoch führen verschiedene Energiequellen auf der Angebotsseite immer mehr zu Volatilität und weniger Flexibilität im Stromsystem. Als Folge resultieren neue Herausforderungen, mit denen das Stromnetz, primär das Verteilernetz konfrontiert wird.[1] Vor diesem Hintergrund rückt die zunehmende Verbreitung von dezentraler Energieversorgung in den Vordergrund. In Anbetracht dessen können in vielen Situationen Überlastungsprobleme an verschiedenen Punkten im Verteilernetz auftreten und gravierende Schäden und potenzielle Stromausfälle verursachen.[2] Innovative Methoden sind daher erforderlich, um das Stromsystem zuverlässiger und effizienter zu gestalten[3]. Diesbezüglich werden ein klarer Einblick in das Netz und geeignete Werkzeuge für die Prognose des zukünftigen Energieverbrauchs und der Energieerzeugung benötigt[4]. Die vorliegende Arbeit befasst sich hierbei mit der Analyse und der Prognose des Energieverbrauchs. Die Lastprognose ist ein wichtiger Bestandteil der Planung und des Betriebs von Energiesystemen. Sie ermöglicht Versorgungsunternehmen, Lasten zu prognostizieren, um die Stabilität zwischen Angebot und Nachfrage aufrechtzuerhalten, die Stromerzeugungskosten zu senken und die Betriebsplanung und zukünftige Kapazitätsplanung zu verwalten[5].Besonders die

[1] (Grid Management System to solve local Congestion, 2019)

[2] (FLECH Services to Solve Grid Congestion, 2018), (Congestion Management in distribution grid networks through active power control of flexible distributed energy resources, 2019)

[3] (FLECH Services to Solve Grid Congestion, 2018)

[4] (Grid Management System to solve local Congestion, 2019)

[5] (Kwon, et al., 2020)

kurzfristige Lastprognose ist für die Bereitstellung und Wartung von Einheiten, den Stromaustausch und die Aufgabenplanung von Stromerzeugungs- und -verteilungsanlagen grundlegend. Seit der Etablierung von modernen Energiesystemen wie Solar-PV (Photovoltaik) und Windenergie ist die eingespeiste Leistung durch spontane Veränderungen der Wetterbedingungen intermittierend und die kurzfristige Vorhersage nimmt folglich an Bedeutung zu[6]. Entsprechend kann aus wirtschaftlicher Sicht eine genaue Lastprognose die Betriebskosten der Stromversorgungsunternehmen senken[7]. Aus der beschriebenen Problematik leitet sich die Aufgabe ab, eine praxisnahe Lastprognose zu entwickeln. Dafür werden reale Daten einer Ladezone des Bundesstaats New Hampshire des Jahres 2019 herangezogen, die Objekte mit Photovoltaikanlagen einschließt. Daher werden für eine verbesserte Prognoselogik weitere Variablen wie meteorologische Daten, Informationen bezüglich Kalenderdaten sowie Zeitangaben hinzugefügt.

1.2 Struktur der Arbeit

In Kapitel zwei „Aktueller Stand der Technik“ werden zunächst zwei Projekte aus dem Energiebereich vorgestellt, die ähnliche Ziele im Vergleich zu dieser Arbeit verfolgen, sowie zwei ausgewählte Algorithmen, die zur Realisierung der kurzfristigen Vorhersage verwendet werden. Das Kapitel soll einen Einblick in den Energiebereich in Bezug auf maschinelles Lernen vermitteln und den Stand der Technik präsentieren. Das darauffolgende Kapitel drei dient der präzisen Formulierung der Zielsetzung der vorliegenden Arbeit. Die einzelnen Abschnitte beschreiben hierbei die Funktion einer Lastprognose. Im vierten Kapitel „Methodik und Konzeption“ steht die Beschreibung der Methodik und Konzeption im Fokus. Zu Beginn des Kapitels werden die verwendeten Datensätze vorgestellt. Im weiteren Verlauf werden verschiedene Verfahren präsentiert, die für die Umsetzung verwendet werden. Aufbauend auf das Kapitel vier „Methodik und Konzeption“ folgt das Kapitel fünf „Umsetzung“, welches die Anwendung der zuvor vorgestellten Verfahren inkludiert sowie die Vorbereitung der Daten für die Prognose. Im Anschluss werden die verschiedenen Modelle und deren Ergebnisse vorgestellt, evaluiert und verglichen. Die Ergebnisse führen zu einer Annahme, die im Nachfolgenden in einem Experiment untersucht wird. Abschließend endet die Arbeit mit einem kurzen Fazit und einem Ausblick.

[6] (Kong, et al., 2019)

[7] (Random forest based ensemble system for short term load forecasting, 2012)

2 Aktueller Stand der Technik

2.1 Vergleichbare Projekte

Es gibt viele verschiedene Ansätze, Prognosen in den Energiesektor mit einzubeziehen. Im Folgenden werden zwei ausgewählte Projekte vorgestellt, die ähnliche Ziele wie die vorliegende Arbeit verfolgen.

2.1.1 Deep DHC

Das Forschungsprojekt „DeepDHC“ dient dazu, eine verbesserte Wärmelastprognose zu entwickeln, die es Wärmeversorgern ermöglicht, eine höchst wirtschaftliche, ökologische, versorgungssichere und netzdienliche Planung ihrer Wärmekraftwerke, Energiespeicher und Power-to-X-Anlangen zu erreichen. Auf Grundlage der Prognose kann der Einsatz von fossilen Spitzenlastkesseln und die Integration erneuerbarer Energien in Fernwärmenetze vermieden werden. In der Praxis sind bislang für Fernwärmelastprognosen nur einfache Prognoseverfahren mit hohen Unsicherheiten im Einsatz. Das Projekt verfolgt das Ziel, die vorkommenden Probleme mit neuen Methoden des maschinellen Lernens zu lösen. Für einen optimierten Betrieb ist jedoch der genaue Zeitpunkt sowie die Höhe der zu erwartenden Lastspitze im Netz notwendig. Daher stellten die Fernwärmeversorger Ulm GmbH und ZAK Energie GmbH reale Betriebsdaten bereit, um die Verfahren praxisnah zu entwickeln und zu validieren[8]. Das untersuchte Teilnetz der Fernwärme Ulm umfasst eine Gesamtlänge von 40 km und deckt den jährlichen Wärmebedarf von 110 Haushalten. Dabei wurden Messdaten der letzten 15 Jahre mit einem stündlichen Intervall sowie 20 Parametern verwendet, die insgesamt 131400 Datenreihen bilden. Die Parameter umfassen dabei Messdaten des Fernwärmenetzes sowie Wetterdaten.

Das entwickelte Verfahren DeepDHC kombiniert neuronale Netze und Entscheidungsbäume, um die Stärken beider Verfahren zu vereinen. So werden konventionelle maschinelle Lernverfahren vom Typ Entscheidungsbäume wie Ada-Boots und Random Forest mit künstlichen neuronalen Netzen wie Feed-Forward-Netzen und Long-Short-Term-Memory-Zellen verbunden.

[8] (AGFW), (HS-Kempten), (Faber, et al., 2018)

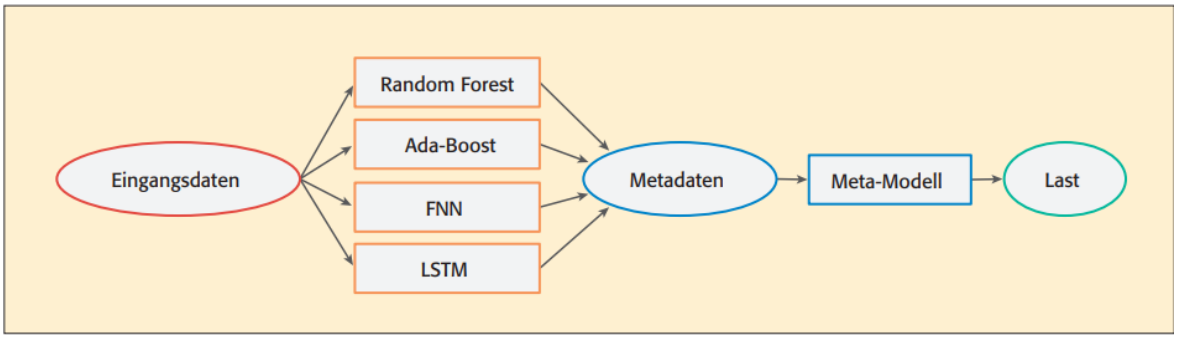


Abbildung 1: DeepDHC- Verfahren [9]

Das Projekt endet im März 2023 und wird im weiteren Projektverlauf optimiert. Zusätzlich wurde das entwickelte Verfahren bereits in Form eines Lastprognose-Tools in die Fernwärme Ulm GmbH integriert, um stündliche aktualisierte Lastprognosen im laufenden Betrieb bereitzustellen. Die Bewertung des wirtschaftlichen Nutzens der optimierten Lastprognose erfolgt mithilfe eines Optimierungsmodells, in dem der Anlagepark sowie das Wärmenetz abgebildet sind[10].

2.1.2 Netzmanagement – System

Um die Energiewende schneller zu gestalten, wurde die Entwicklung des Netzmanagement-Systems als Teil des Projekts „Interflex“ initiiert und in der Stadt Eindhoven durchgeführt. Mit dem Projekt wurde beabsichtigt lokale Lasten vorherzusagen und mit einem intelligenten Algorithmus zu entscheiden, wo und wann Flexibilität erforderlich ist. „Interflex bieten somit eine Lösung, um Engpässe zu vermeiden. Das System wurde von dem Unternehmen Enexis Netbeheer entwickelt, das unter anderem das Strom- und Gasnetz im Norden, Osten und Süden der Niederlande für etwa 2.6 Millionen Kunden baut und verwaltet. Ein Verteilnetzbetreiber ist in der Lage, die Flexibilität auf der Nachfrageseite zu nutzen, um Lastspitzen zu kontrollieren und die Verstärkung zu verhindern oder zu verschieben. Dazu benötigt es jedoch einen tieferen Einblick in das Lastverhalten, um Überlastungsproblemen entgegenzuwirken und zu einer verbesserten Netzkapazität beizutragen.

Da diese Problematik stark ortsabhängig ist, sind Informationen über Anlagen, historische Messungen über die Spitzenlast, den Netzzustand, die Schwere des Engpasses, die maximal verfügbare Kapazität und die Wetterdaten der einzelnen Gebiete notwendig.

[9] (Faber, et al., 2018)

[10] (HS-Kempten), (Faber, et al., 2018)

Allerdings sind intelligente Produkte für diesen speziellen Anwendungsfall nicht geeignet. Aus diesem Grund sind ein Vorhersagemodell sowie eine spezifische Kommunikation zwischen den Komponenten im Netz erforderlich. Abbildung 2 zeigt den Aufbau der einzelnen Komponenten des Netzmanagement Systems.

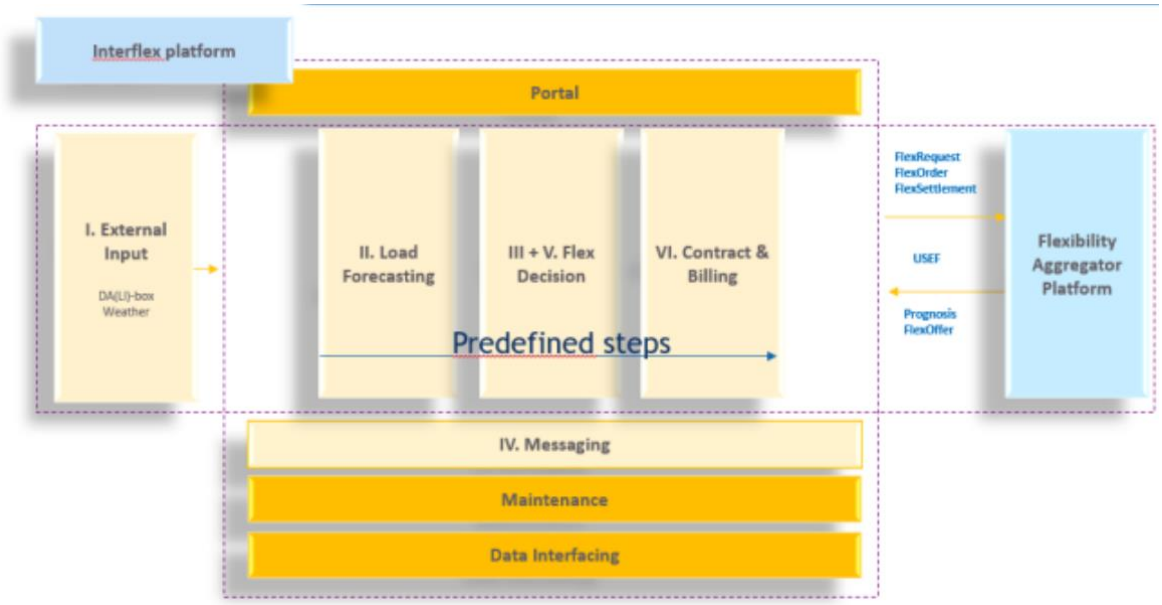


Abbildung 2: Netzmanagement-System [11]

Im Folgenden werden die nummerierten Komponenten 1-6 näher erläutert:

Schritt 1 definiert die externe Eingabe, den Empfang und die Speicherung historischer Daten. Dieses Modul verarbeitet 40.000.000 Messungen pro Tag sowie Wetterdaten, die eine lokale Wettervorhersage einschließlich Windgeschwindigkeit, Temperatur und Sonneneinstrahlung enthalten. Alle Daten werden in einer Big-Data-Cloud-Umgebung gespeichert, um Trends für eine mögliche zukünftige Nutzung zu analysieren.

Schritt 2 beinhaltet die Lastprognose, die zur Vorhersage von Engpässen verwendet wird und die 15-Minuten-Werte sowie die Wetterprognose als Eingabe nutzt[12]. Für die Zeitreihe wurden drei verschiedene Modelle des maschinellen Lernens in Betracht gezogen, ein lineares Regressionsmodell, ein XGBoost-Regressionsmodell und ein Random-Forest-Regressionsmodell. Durch verschiedene Tests und Kriterien fiel die Wahl letztendlich auf das XGBoost-Regressionsmodell, welches die besten Leistungen erzielte.

[11] (Grid Management System to solve local Congestion, 2019)

[12] (Grid Management System to solve local Congestion, 2019)

Dazu konnte bewiesen werden, dass durch die Anwendung der Zeitreihenzerlegung und die Einbindung der Wetterinformationen sowie zeitbezogene Merkmale die Leistungen deutlich verbessern[13].

Schritt 3 beschreibt ein Entscheidungsmodell, dessen Hauptziel es ist, das System automatisch ohne menschliche Interaktion arbeiten zu lassen. Es bestimmt das Ausmaß und die Dauer der Überlastung auf der Grundlage der Lastprognose und der Transformatorenkapazität.

Schritt 4 stellt die Nachrichtenübermittlung dar. Das Senden von Flex-Anfragen und Empfangen von Flex-Angeboten dient dazu, Aggregatoren zu benachrichtigen, die in der Lage sind, die lokale Überlastung zu beheben.

Im **5. Schritt** können, nachdem der Bedarf und der Preis bekannt gegeben werden, die Aggregatoren ein geeignetes Angebot abgeben. Falls das Angebot die Überlastung nicht beheben kann, wird dieser Prozess iterativ wiederholt, bis der Engpass gelöst oder kein geeignetes Angebot mehr vorhanden ist.

Schritt 6 stellt die Grundlast dar, die sich ergibt, wenn der Aggregator nicht liefern würde. Da einige flexible Lasten wie E-Fahrzeuge weniger vorhersehbar sind, können die Aggregatoren angeben, wie die tatsächliche Grundlast aussieht.

Alle Module sind auf diese Weise aufgebaut und konzipiert, damit neue oder aktualisierte Module leicht in das vorhandene System integriert werden können.

Die erste Version des Netzmanagement-Systems wurde 2019 entwickelt und getestet[14].

2.2 Algorithmen

Die folgenden Abschnitte stellen die verwendeten Algorithmen vor, um ein zunehmendes Verständnis für die einzelnen Konzepte zu erlangen.

2.2.1 LSTM

Für eine stündliche Lastvorhersage, die verschiedene Variablen mit einschließt, wird ein Modell benötigt, welches diesen Herausforderungen gerecht wird. Daher wird eine Methode verwendet, die auf tiefen neuronalen Netzwerken basiert, um aus den Eingaben zu lernen

[13] (Short-Term Load Forecasting on MV/LV Transformer Level, 2019)

[14] (Grid Management System to solve local Congestion, 2019), (Short-Term Load Forecasting on MV/LV Transformer Level, 2019)

und reichhaltige Merkmale zu extrahieren[15]. Long Short-Term Memory, auch kurz LSTM genannt, ist eine spezielle Art von rekurrenten neuronalen Netzen. Sie wurden 1997 von Hochreiter & Schmidhuber eingeführt, um Fehlerrückflussprobleme zu überwinden[16]. Aufgrund ihrer Fähigkeiten aus komplexen nichtlinearen Mustern und der automatischen Merkmalsextraktionsfähigkeit zu lernen, haben LSTMs an Popularität gewonnen[17]. Standardmäßig bestehen alle rekurrenten neuronalen Netze aus rekursiven Netzmodulen, die in Form einer Kette auftreten. Diese ermöglichen es, aus vorangegangenen Phänomenen zu lernen und unterscheiden sich somit von herkömmlichen Netzen. Dieser rekursive Fluss kann jedoch zum Problem des verschwindenden und explodierenden Gradienten führen. LSTMs weisen ebenfalls kettenartige Strukturen auf, die dieses Problem jedoch mit einem speziellen Gating-Mechanismus entschärfen. Damit kann die nächste LSTM-Einheit in jedem Zeitschritt wählen, ob sie aus der Zelle lesen, in sie schreiben oder sie zurücksetzen will[18]. Zudem können LSTMs die Korrelation zwischen benachbarten Zeitpunkten, die tagesbezogene Korrelation und die wochenbezogene Korrelation erfassen, um die Genauigkeit der Lastprognose zu verbessern[19]. In Bezug auf Lastprognosen übertreffen LSTM-Ansätze meist konkurrierende Algorithmen, wie in (Kong, et al., 2019) bewiesen wurde. Ebenso verwenden (Bouktif, et al., 2018), (Muzaffar, et al., 2019) und (Jiao, et al., 2018) ein LSTM basierendes Netzwerk mit verschiedenen Konfigurationen, um Prognosemodelle für kurz- bis mittelfristige Gesamtlastprognosen zu erstellen, welches andere maschinelle Lernansätze übertrifft[20]. Besonders attraktiv sind LSTMs für die Modellierung sequenzieller Daten wie Zeitreihen, da sie kontextbezogene Informationen aus vergangenen Eingaben enthalten. Die hohe Leistung wird durch die Speicherung langfristiger Abhängigkeiten erreicht. Dank des internen Speichers eignet sie sich ideal für Probleme mit sequenzabhängigem Verhalten, wie z. B. der Stromlast und-nachfrage[21].

[15] (Kwon, et al., 2020)

[16] (Multivariate Time Series Forecasting with LSTM for Madrid, Spain pollution, 2019)

[17] (Bouktif, et al., 2018)

[18] (Multivariate Time Series Forecasting with LSTM for Madrid, Spain pollution, 2019), (Ushiku, 2021), (Hochreiter, et al., 1997), (Chatterjee, et al., 2020)

[19] (Jiao, et al., 2018)

[20] (Bouktif, et al., 2018), (Muzaffar, et al., 2019), (Jiao, et al., 2018)

[21] (Bouktif, et al., 2018)

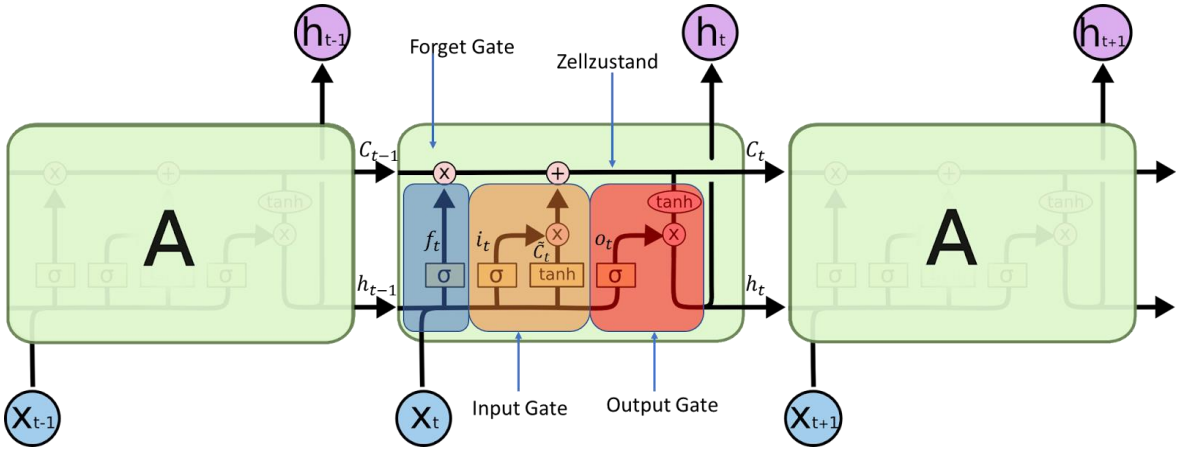


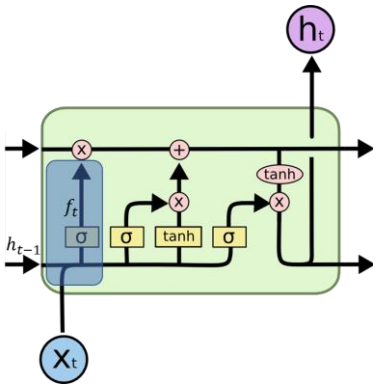
Abbildung 3: LSTM - Zelle, in Anlehnung an (Multivariate Time Series Forecasting with LSTM for Madrid, Spain pollution, 2019), (Chatterjee, et al., 2020)

Abbildung 3 zeigt den Aufbau einer LSTM-Einheit. Die einzelnen Pfeile verdeutlichen jeweils die Übertragung eines ganzen Vektors vom Ausgang eines Knotens zu den Eingängen der anderen. Der Vektor C bildet den Zustand, der die Zelle modifiziert und wird als Langzeitgedächtnis bezeichnet. Der Vektor h bildet den versteckten Vektor, auch als Kurzzeitgedächtnis bezeichnet. Die rosafarbenen Kreise stehen für punktuelle Operationen wie die Vektoraddition und Multiplikation, während die gelben Kästchen die Sigmoid- und Tanh-Funktion darstellen. Zusammenlaufende Linien definieren Verkettungen, während eine sich gabelnde Linie bedeutet, dass ihr Inhalt kopiert und die Kopien an verschiedene Stellen weitergegeben werden. Deutlich zu erkennen sind die drei markierten Bereiche, die die verschiedenen Gate-Einheiten darstellen.

Ein LSTM hat insgesamt drei Gates, um den Zellzustand zu schützen und zu kontrollieren. Das Forget Gate definiert dabei den blau markierten, das Input Gate den orange markierten und das Output Gate den rot markierten Bereich. Diese werden nacheinander durchlaufen und deren Ausgabewert wird zum Zellzustand hinzu multipliziert oder addiert.

Der erste Schritt im LSTM besteht darin, zu entscheiden, welche Informationen aus dem Zellzustand verworfen werden. Diese Entscheidung wird mittels Forget Gate gesteuert. Darauf folgt das Input-Gate, das entscheidet, ob die Speicherzelle aktualisiert wird und zuletzt das Output-Gate, welches definiert, ob die Werte über den aktuellen Zellzustand sichtbar sind. Jedes der Gates nimmt den verborgenen Zustand und den aktuellen Eingang x als Eingaben auf. Dabei werden die Vektoren verkettet und bilden ein Sigmoid, welches die Zahlen

0 bis 1 ausgeben kann. Zusätzlich verfügt das Input- und das Output-Gate über eine Tanh-Funktion, dessen Ausgabewerte von -1 bis 1 reichen können, um den Werten eine Gewichtung hinzuzufügen[22].



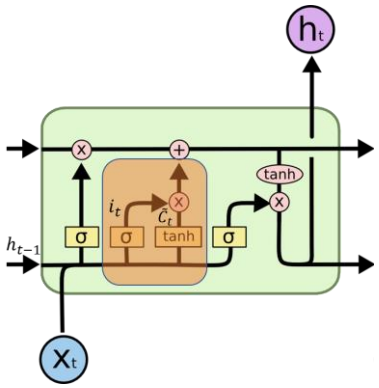
$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$

Abbildung 4: Forget Gate, in Anlehnung an (Multivariate Time Series Forecasting with LSTM for Madrid, Spain pollution, 2019), (Chatterjee, et al., 2020)

Bei einer detaillierten Betrachtung der LSTM Einheit und der einzelnen Gates definiert x_t den Eingangszustand. h_{t-1} stellt den historischen Wert des vorherigen Schritts $(t-1)$ dar. Diese werden anschließend miteinander verkettet und führen zum ersten Gate. Dabei verläuft der erste Schritt über die Sigmoid-Funktion. Die Ausgabewerte können, wie bereits erwähnt, zwischen 0 und 1 liegen. 0 bedeutet, dass das Gate effektiv geschlossen ist und keine Informationen passieren, 1 hingegen bedeutet, dass das Gate weit offen ist und alle Informationen weitergegeben werden. Im Forget-Gate wird demnach entschieden, alles zu vergessen oder nur einen Teil zu verwerfen. Dessen Ausgabe f_t wird anschließend mit dem aktuellen Zellzustand Multipliziert[23].

[22] (Multivariate Time Series Forecasting with LSTM for Madrid, Spain pollution, 2019), (Ushiku, 2021), (Hochreiter, et al., 1997), (Chatterjee, et al., 2020)

[23] (Multivariate Time Series Forecasting with LSTM for Madrid, Spain pollution, 2019), (Ushiku, 2021), (Hochreiter, et al., 1997), (Chatterjee, et al., 2020)

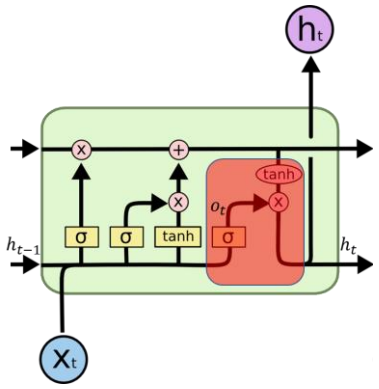


$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$

$$\tilde{C}_t = tanh(W h_{t-1} + U x_t + b)$$

Abbildung 5: Input Gate, in Anlehnung an (Multivariate Time Series Forecasting with LSTM for Madrid, Spain pollution, 2019), (Chatterjee, et al., 2020)

Im Anschluss folgt das Input Gate, welches entscheidet, alle vorangegangenen Informationen oder nur einen Teil zum Eingangszustand zu addieren. Hier wird ebenfalls mit der integrierten Sigmoid-Funktion bestimmt, welche Werte aktualisiert werden und mittels $tanh$ die Gewichtung der einzelnen Werte identifiziert[24].



$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$$

$$h_t = o_t \circ tanh(C_t)$$

Abbildung 6: Output Gate, in Anlehnung an (Multivariate Time Series Forecasting with LSTM for Madrid, Spain pollution, 2019), (Chatterjee, et al., 2020)

[24] (Multivariate Time Series Forecasting with LSTM for Madrid, Spain pollution, 2019), (Ushiku, 2021), (Hochreiter, et al., 1997), (Chatterjee, et al., 2020)

Zuletzt das Output Gate, das bestimmt, ob alle oder keine Werte ausgegeben werden. Dies funktioniert ebenfalls unter der Verwendung der Sigmoidfunktion sowie $tanh$ für die Gewichtung des Wertes. Miteinander verkettet wird abschließend der Wert an die nächste LSTM-Einheit weitergegeben. Auf diese Weise wiederholt sich der Prozess bis zur letzten LSTM-Einheit. Eine detaillierte Beschreibung der einzelnen Formeln und des Aufbaus bezüglich der LSTMs findet sich in (Hochreiter, et al., 1997).[25]

2.2.2 Random Forest

Zum Vergleich zum LSTM wird der Random Forest Algorithmus verwendet. Der Random Forest Algorithmus wurde 2001 von L. Breimann vorgeschlagen und ist ein überwachtes Lernverfahren, das als universelles Klassifizierungs- und Regressionsverfahren angewendet wird. Die Popularität des Random Forest ergibt sich durch die breite Anwendbarkeit auf eine Vielzahl an Prognosen[26]. Die Hauptvorteile des Modells liegen in seiner Verallgemeinerungsfähigkeit, der eingebauten Kreuzvalidierung und der geringen Empfindlichkeit gegenüber Parameterwerten[27]. Daher gilt der Random Forest auch als ein sehr praktischer und einfach zu bedienender Algorithmus, da dessen Standard-Hyperparameter oft zu einem guten Ergebnis führen und das Problem der Überanpassung seltener auftritt[28]. Selbst eine schlechte Wahl der verwendeten Parameter beeinträchtigt die Vorhersagegenauigkeit nicht drastisch. Genauso wie dieser Algorithmus in (Random forests model for one day ahead load forecasting, 2015) und (Random forest based ensemble system for short term load forecasting, 2012) Verwendung gefunden hat, wird dieser hier verwendet, um schrittweise die Last von einer Stunde im Voraus zu prognostizieren. Die grundlegenden Elemente des Random Forest bestehen aus CART-Bäumen (Klassifizierungs- und Regressionsbäumen), die von Bootstrap-Datenproben kombiniert werden. Die Proben werden aus dem ursprünglichen Datensatz generiert und anschließend vorhergesagt. Abschließend wird ein Prädiktor aus jeder Stichprobe erstellt und mittels Mehrheitsabstimmung oder Mittelwertbildung aggregiert[29]. Abbildung 7 veranschaulicht diesen Prozess.

[25] (Multivariate Time Series Forecasting with LSTM for Madrid, Spain pollution, 2019), (Ushiku, 2021), (Hochreiter, et al., 1997), (Chatterjee, et al., 2020)

[26] (Biau, et al., 2016)

[27] (Short-Term Load Forecasting Using Random Forests, 2014)

[28] (Tran, 2019)

[29] (Biau, et al., 2016), (Sagar, et al., 2020)

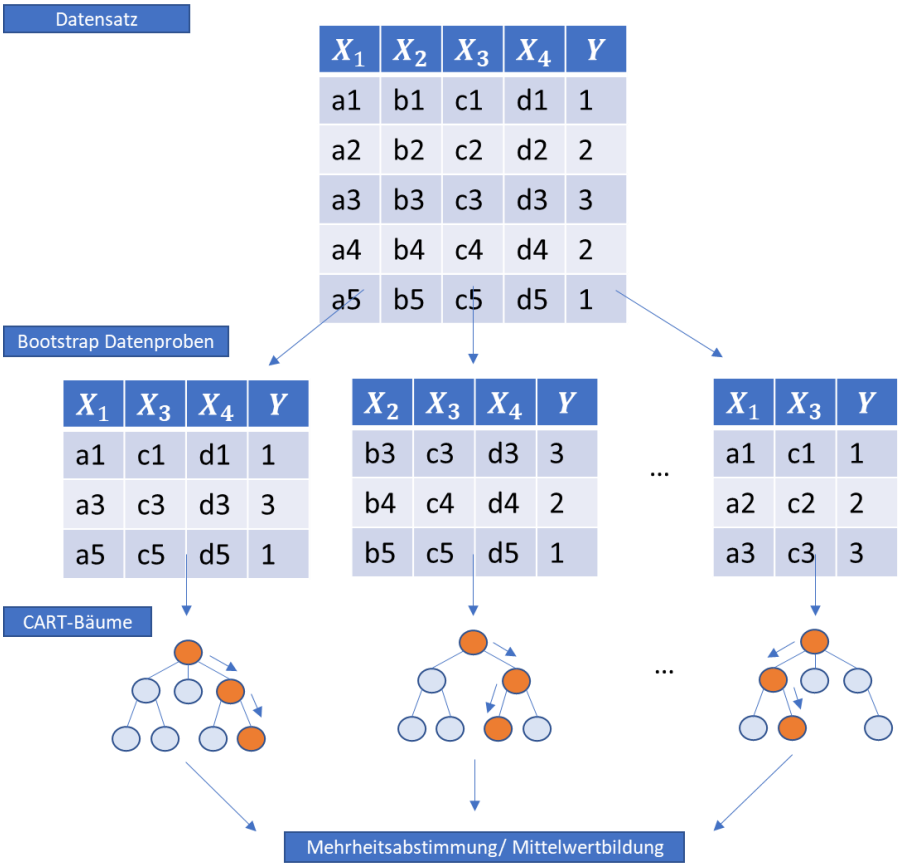


Abbildung 7: Random Forest

Die Verwendung von mehreren verschiedenen Bäumen anstelle eines einzelnen Baumes verbessert die Prognosegenauigkeit außerhalb des Trainings-Datensatzes aufgrund der individuellen Prognose jedes einzelnen Baumes. Dies gewährleistet eine stabile und robuste Vorhersage, da mehr Möglichkeiten im Trainingsdatensatz umfasst werden[30]. Ein weiterer Vorteil ist, dass der Algorithmus sowohl für Klassifizierungs- als auch für Regressionsprobleme verwendet werden kann, die den Großteil der aktuellen maschinellen Lernsysteme ausmachen[31].

Die allgemeine Definition des Random Forests von Breiman (2001) lautet wie folgt:

"Ein Random Forest ist ein Klassifikator, der aus einer Sammlung von baumstrukturierten Klassifikatoren {h(x, k), k = 1,...} besteht, wobei die {k } unabhängige, identisch verteilte

[30] (Sagar, et al., 2020)

[31] (Tran, 2019)

Zufallsvektoren sind und jeder Baum eine Einheitsstimme für die beliebteste Klasse bei Eingabe x abgibt."[32]

Genauer erklärt bedeutet die Definition, dass für jeden Baum ein Zufallsvektor erzeugt wird. Dieser Zufallsvektor ist unabhängig von den vorherigen Vektoren 1, ... k-1 zu betrachten, weist jedoch dieselbe Verteilung auf. Auf diese Weise wird ein Baum auf Basis der Trainingsmenge und k erzeugt und führt schließlich zu einem Klassifikator h(x,k), wobei x einen Eingabevektor darstellt. Die Art und Dimensionalität des Zufallsvektors hängt dabei von seiner Verwendung bei der Baumkonstruktion ab[33]. Eine detaillierte Beschreibung bezüglich der Herleitung des Random Forest findet sich in (Breiman, 2001).

3 Zielsetzung

3.1 Ziel der Arbeit

Aus dem vorherigen Kapitel ist zu entnehmen, dass verschiedene Algorithmen zielführend sein können. Jedoch ist zu pointieren, dass das Projekt Deep DHC, LSTMs als bestes Einzelverfahren hervorhebt und der Random Forest Algorithmus in beiden Projekten Verwendung gefunden hat[34]. Ziel dieser Arbeit ist es, eine Prognose mittels LSTM basierter multivariater Zeitreihenanalyse zu implementieren und zu untersuchen, ob die Zugabe von weiteren Variablen sowie der Ansatz der neuronalen Netze ein verbessertes Ergebnis erzielen kann. Die zum Vergleich stehenden Modelle sind die LSTM basierte univariate Zeitreihenanalyse und die Random Forest basierte univariate Zeitreihenanalyse. Inwieweit sich univariate und multivariate Zeitreihenanalysen voneinander unterscheiden, wird in Abschnitt 4.3. genauer erläutert. Der folgende Abschnitt konkretisiert den Nutzen der Lastprognose.

3.1.1 Lastprognose

Die Lastprognose ist für verschiedene Komponenten von Bedeutung. Bestandteil dieser sind Energieversorger, ISOs, Finanzinstitute und weitere Teilnehmer, die sich an der Stromer-

[32] (Breiman, 2001)

[33] (Breiman, 2001)

[34] (Faber, et al., 2018), (Short-Term Load Forecasting on MV/LV Transformer Level, 2019)

zeugung, der Stromübertragung und der Stromverteilung beteiligen. Vor allem für die Planung und die betrieblichen Entscheidungen von Versorgungsunternehmen können genaue Modelle hilfreich sein. Zusätzlich sind mit der Deregulierung, Lastprognosen essenzieller Bestandteil der Energiewirtschaft geworden. Insbesondere in Bezug auf Schwankungen von Angebot und Nachfrage, die Veränderung von Wetterbedingungen sowie steigende Energiepreise. Daher kann eine frühzeitige Umsetzung von Entscheidungen zu einer Verbesserung der Netzzuverlässigkeit und zur Verringerung von Geräteausfällen und Stromausfällen führen.

Lastprognosen können in drei Kategorien unterteilt werden:

- Eine kurzfristige Prognose, die in der Regel von einer Stunde bis zu einer Woche reicht.
- Eine mittelfristige Prognose, die eine Prognose für eine Woche bis zu einem Jahr einschließt.
- Eine langfristige Prognose, die länger als ein Jahr umfasst[35].

Diese Arbeit verfolgt den Ansatz der kurzfristigen Lastprognose.

4 Methodik und Konzeption

4.1 Datensammlung

Die verwendeten Daten für die Lastprognose, die in dieser Arbeit verwendet werden, stammen aus zwei Quellen:

1. Energieverbrauch New Hampshire
2. Meteorologischen Daten

Die Datensätze umfassen den Zeitraum vom 01-01-2019 bis zum 30-12-2019.

4.1.1 Energieverbrauch New Hampshire

Der Datenbestand für die Stromlast-Nachfrage stammt aus internen Datenbanken der CGI Group Inc. Diese Datensammlung beinhaltet reale Intervalldaten verschiedener Zähler einer Lastzone in New Hampshire. Diese Lastzone versorgt in 24 Intervallen pro Tag 85.000 Zähler für 365 Tage. Die folgende Abbildung zeigt einen Ausschnitt der Datenlage.

[35] (Genethliou, et al., 2005)

```
CSPT_ID,LOAD_DATE,INTERVAL,USAGE
6000000249,1/1/2019,1,49
6000000249,1/1/2019,2,50
6000000249,1/1/2019,3,50
6000000249,1/1/2019,4,51
6000000249,1/1/2019,5,49
6000000249,1/1/2019,6,52
6000000249,1/1/2019,7,50
6000000249,1/1/2019,8,48
```

Abbildung 8: Intervalldaten_2019.csv

Die Datensammlung beinhaltet insgesamt vier Attribute:

Cspt ID: Die Consumption Points ID identifiziert die Abnahmestellen, an denen die Energie an die Kunden des Käufers und in die Räumlichkeiten der Privatkunden geliefert wird.
Load Date: Das Lastdatum liefert die Information über den Tag der Energieeinspeisung.
Interval: Das Intervall definiert die Zeitintervalle und ist gleichzusetzen mit Zeitstunden.
Usage: Der Verbrauch umfasst die tatsächlich eingespeiste Energie in Kilowattstunden.

4.1.2 Meteorologischen Daten

Die meteorologischen Daten werden in der Prognose als zusätzliches Feature benötigt, um ein verbessertes Modell zu erstellen und genauere Vorhersagen zu treffen. Renewables.ninja spezialisiert sich auf Simulationen der stündlichen Stromerzeugung von Wind- und Solarkraftwerken überall auf der Welt. Dessen zugehöriges Web Tool wurde entwickelt, um Wetter- und Energiedaten in wissenschaftlicher Qualität für eine breitere Gemeinschaft verfügbar zu machen[36]. Dieses Tool wurde verwendet, um die Wetterdaten für die Stadt New Hampshire zu ermitteln.

[36] (Pfenninger, et al.)

```
time,local_time,temperature,radiation_surface,radiation_toa
2019-01-01 00:00,2018-12-31 19:00,2.252,0,0
2019-01-01 01:00,2018-12-31 20:00,1.988,0,0
2019-01-01 02:00,2018-12-31 21:00,1.505,0,0
2019-01-01 03:00,2018-12-31 22:00,1.348,0,0
2019-01-01 04:00,2018-12-31 23:00,1.318,0,0
2019-01-01 05:00,2019-01-01 00:00,1.180,0,0
2019-01-01 06:00,2019-01-01 01:00,1.448,0,0
2019-01-01 07:00,2019-01-01 02:00,2.069,0,0
2019-01-01 08:00,2019-01-01 03:00,2.689,0,0
2019-01-01 09:00,2019-01-01 04:00,3.050,0,0
2019-01-01 10:00,2019-01-01 05:00,3.519,0,0
2019-01-01 11:00,2019-01-01 06:00,3.047,0,0
2019-01-01 12:00,2019-01-01 07:00,2.785,15.056,56.181
2019-01-01 13:00,2019-01-01 08:00,3.942,100.515,250.941
```

Abbildung 9: weather_solar.csv

Die Abbildung 9 zeigt einen Ausschnitt der Datensammlung, bestehend aus fünf Attributen:

Time: Das Datum und die UCT bzw. die heute gültige stündliche Weltzeit.
Local Time: Das Datum und die stündliche Ortszeit, in diesem Fall New Hampshire.
Temperature: Die Temperatur in der Einheit Celsius.
Radiation Surface: Die Sonneneinstrahlung in Bodennähe
Radiation Toa: Die Sonneneinstrahlung über der Atmosphäre[37]

4.2 Deskriptive und Explorative Datenanalyse

Bevor die Modellentwicklung realisiert wird, ist das Verständnis über die Daten essenziell. Die deskriptive Datenanalyse wird dazu verwendet, in einer Stichprobe erworbene Daten und ihre Ausprägungen tabellarisch, grafisch oder anhand von Kennwerten darzustellen. Daraufhin folgt die explorative Datenanalyse, die vielmehr dazu dient, signifikante Informationen aus den Daten zu extrahieren, die in der einfachen Betrachtung der deskriptiven Datenanalyse nicht unmittelbar zu erkennen sind. Überdies fungiert die Analyse zur Ermittlung von Mustern und Zusammenhängen, die anhand von Darstellungen und Berechnungen eruiert werden können[38]. Bestandteil sind Berechnungen grundlegender, statischer Eigenschaften, wie der Mittelwert und die Varianz sowie die Untersuchungen von Korrelationen zwischen ausgewählten Variablen. Ein weiteres Ziel der explorativen Datenanalyse ist neben

[37] (Pfenninger, et al.)

[38] (Schäfer, 2010)

dem umfassenden Einblick in den Datensatz im Rahmen der Datenqualität die Überprüfung der Daten auf Vollständigkeit[39].

4.3 Zeitreihenanalyse

Die Zeitreihenanalyse analysiert den Verlauf von Werten in einem spezifischen Zeitraum und dient zur Erkennung von Mustern und aussagekräftigen Erkenntnissen. Unter Berücksichtigung von historischen Werten und deren Einflussfaktoren kann eine Prognose über die Entwicklung zukünftiger Ereignisse getroffen werden[40]. Zusätzlich zu beachten ist, dass zwischen univariaten und multivariaten Zeitreihen unterschieden wird. Der signifikante Unterschied liegt bei der Betrachtung der zeitabhängigen Variablen. Bei einer univariaten Zeitreihe wird lediglich eine Variable berücksichtigt, hingegen werden im Fall einer multivariaten Zeitreihe mehrere Variablen einbezogen. Genauer erläutert hängt bei einer multivariaten Zeitreihenanalyse jede Variable von ihren vergangenen Werten sowie von anderen vergangenen Variablenwerten ab[41]. Gleichgesetzt zum täglichen Lastprofil, welches ebenfalls abhängig von verschiedenen Variablen ist, die sich unterschiedlich auswirken können, beispielsweise Wetterbedingungen, Kalendertage und Zeitangaben[42].

4.3.1 Komponenten

Häufig sind typische Verlaufsmuster in einer Zeitreihe zu erkennen, die sich in bestimmte Komponenten einteilen lassen. Bestandteil der relevanten Komponenten sind die Trendkomponente, die Saisonkomponente, die Konjunkturkomponente und die Restkomponente. Die Trendkomponente definiert die allgemeine Entwicklungstendenz einer Zeitreihe. Diese kann konstant steigend, sinkend oder unverändert verlaufen. Die Saisonkomponente umfasst wiederkehrende Schwankungen und wird bedingt durch jahreszeitliche oder institutionelle Einflüsse verursacht. Die Konjunkturkomponente beschreibt regelmäßige Schwankungen einer Zeitreihe mit einer Länge von über einem Jahr, die auf wechselnde wirtschaftliche Aktivitä-

[39] (Buxmann, et al.)

[40] (Statista)

[41] (Multivariate Time Series Forecasting with LSTM for Madrid, Spain pollution, 2019)

[42] (Son, et al., 2022)

ten zurückzuführen sind. Die Restkomponente schließt indessen alle weiteren Schwankungen ein, die nicht zuvor genannt wurden, wie unerwartete Schwankungen oder Fehler[43]. Zur Untersuchung von Zeitreihen empfiehlt es sich mit einer visuellen Überprüfung zu beginnen, um rein deskriptive Eigenschaften zu identifizieren. Dadurch können ein Trend oder eine saisonale Bewegung erkannt werden[44].

4.3.2 Stationarität

Stationarität ist eine der Schlüsselkomponenten der Zeitreihenanalyse. Nichtstationäre Daten sind in der Regel unvorhersehbar, können nicht modelliert oder vorhergesagt werden. Daher ist es vorteilhaft, wenn die untersuchte Zeitreihe Eigenschaften der Stationarität aufweist, um konsistente und zuverlässige Ergebnisse zu erhalten. Stationarität ist gegeben, wenn die Daten einen konstanten Mittelwert und eine konstante Varianz aufweisen; genauer gesagt, wenn die statischen Eigenschaften nicht mit der Zeit variieren. Um eine Zeitreihe auf Stationarität zu überprüfen, können verschiedene Konzepte angewendet werden[45].

Ein Konzept basiert auf einer bekannten Überprüfung mittels der Berechnung des Mittelwertes und der Varianz. Die zusammenfassende Statistik des Mittelwerts und der Varianz können das Verhalten der Daten widerspiegeln. Werden die Daten in zufällige Partitionen unterteilt und die Werte berechnet, können diese Aufschluss darüber geben, ob eine Reihe stationär ist. Das bedeutet, wenn der beobachtete Faktor variiert und statistisch signifikant ist, wird von einer nichtstationären Zeitreihe gesprochen. Werden demnach die vorhandenen Daten in zwei Partitionen geteilt und untersucht, sollten die Werte nicht stark voneinander abweichen[46].

Ein weiteres Konzept führt zu einer genaueren Überprüfung der Stationarität einer Zeitreihe, welches auf einem statischen Test beruht. Dafür werden standardmäßig Einheitswurzel-Tests verwendet. Das Konzept hinter diesen Tests besagt, dass Zeitreihen, die eine Einheitswurzel enthalten, die Varianz systematisch in Abhängigkeit von der Zeit verändern und damit beweisen, dass eine Reihe nicht stationär ist.

[43] (Statista), (Kirchgässner, et al., 2005)

[44] (Kirchgässner, et al., 2005)

[45] (Savit, et al., 1996)

[46] (Vishwas, et al., 2020), (Stoetzer, 2020)

Es gibt eine Reihe von Methoden, die den Einheitswurzeltest beinhalten. Einer der weit verbreitetsten ist der Erweiterte Dickey Fuller Test, auch ADF-Test genannt. Dieser basiert auf dem Dickey Fuller Test und berücksichtigt zusätzlich verzögerte Differenzen einer Reihe, die die Autokorrelation eliminiert[47]. Bei der Prüfung der stationären Einheitswurzel mittels ADF-Test sind sowohl die Nullhypothese als auch die Alternativhypothese (H1) von besonderer Bedeutung:

- Nullhypothese (H0): Wenn sie nicht zurückgewiesen werden kann, deutet dies darauf hin, dass die Zeitreihe eine Einheitswurzel hat. Dies wiederum bedeutet, dass sie nicht stationär ist und von einer zeitabhängigen Struktur ausgegangen werden kann.
- Alternativhypothese (H1): Die Nullhypothese wird verworfen und deutet darauf hin, dass die Zeitreihe keine Einheitswurzel hat. Folglicherweise handelt es sich um eine stationäre Zeitreihe mit einer zeitabhängigen Struktur.

Das Ergebnis des Tests wird in Form von p-Werten angegeben. Dabei ist es wichtig zu verstehen, wie dieser zu interpretieren ist. Liegt der p-Wert unter dem Schwellenwert, also **$p < 0.05$**, wird die Nullhypothese zurückgewiesen und zeigt, dass die Zeitreihe stationär ist. Wird der Schwellenwert jedoch überschritten, beweist es das Gegenteil und sagt aus, dass die Reihe nicht stationär ist. Standardmäßig werden bei dem ADF-Test die Teststatistik, der p-Wert und die kritischen Werte bei 1%, 2,5%, 5% und 10% Konfidenzintervall betrachtet[48].

5 Umsetzung

5.1 Anwendung Deskriptiver und Explorativer Datenanalyse

Zur Visualisierung der Daten wird Pandas verwendet, ein Open-Source-Tool zur Datenanalyse und -manipulation, das auf der Programmiersprache Python aufbaut. Eine Datenvisualisierung dient dazu, die Daten einfacher zu verstehen sowie Muster, Trends und Aus-

[47] (Vishwas, et al., 2020), (Stoetzer, 2020)

[48] (Vishwas, et al., 2020)

reißer in großen Datensätzen leichter zu erkennen. Vor allem Pandas bietet die Möglichkeit, Daten mit Hilfe der Matplotlib zu plotten. Es ist eine Bibliothek, die es ermöglicht, mathematische Plots aller Art zu erstellen[49].

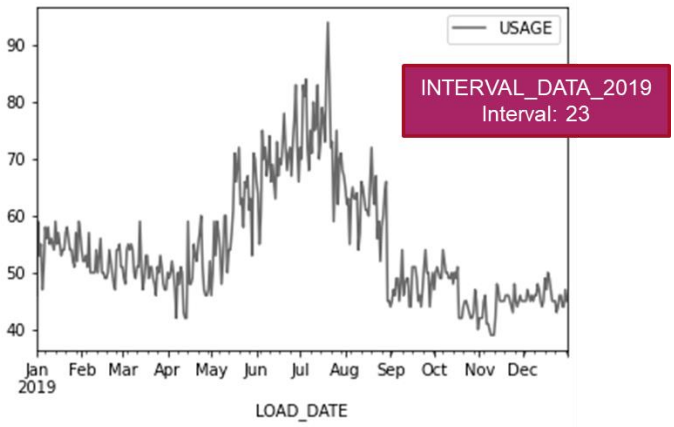


Abbildung 10: CSPT_ID: 6000000249, Interval 23

Wie in Abschnitt 4.3 bereits erwähnt, fungiert die deskriptive und explorative Datenanalyse zum Verständnis der verfügbaren Daten. In diesem Abschnitt wird die zuvor genannte Methodik vertieft und angewendet. Zur Realisierung einer angemessenen Veranschaulichung werden Zeitreihendiagramme eingesetzt. Zeitreihendiagramme sind grafische Darstellungen, die zur Analyse der Datenaktivität über einen bestimmten Zeitraum verwendet werden. Sie sind nützlich für kurz- und langfristige Tendenzen von Daten[50]. Für die Datenanalyse wurden verschiedene Verbrauchsstellen sowie Zeitintervalle untersucht und analysiert. Abbildung 3 zeigt ein Beispiel für eine Verbrauchsstelle zum 23. Intervall. In diesem Fall wird das gesamte Jahr 2019 betrachtet. Es ist deutlich zu erkennen, dass der Verbrauch zur Mitte des Jahres seine Spitzenlast erreicht und zum Ende des Jahres bedeutend sinkt. Es ist bekannt, dass der Datensatz Objekte mit Photovoltaikanlagen beinhaltet. Dies führt im Sommer zu einem hohen Überangebot und im Winter zu einem Erzeugungsmangel. Das folgert, dass zur Sommerzeit aufgrund der höheren Sonneneinstrahlungen die gespeicherte Energie

[49] (IBM), (Pandas)

[50] (IBM)

dazu genutzt wird, um die Objekte im Winter zu versorgen und infolgedessen weniger Energie eingespeist werden muss[51].

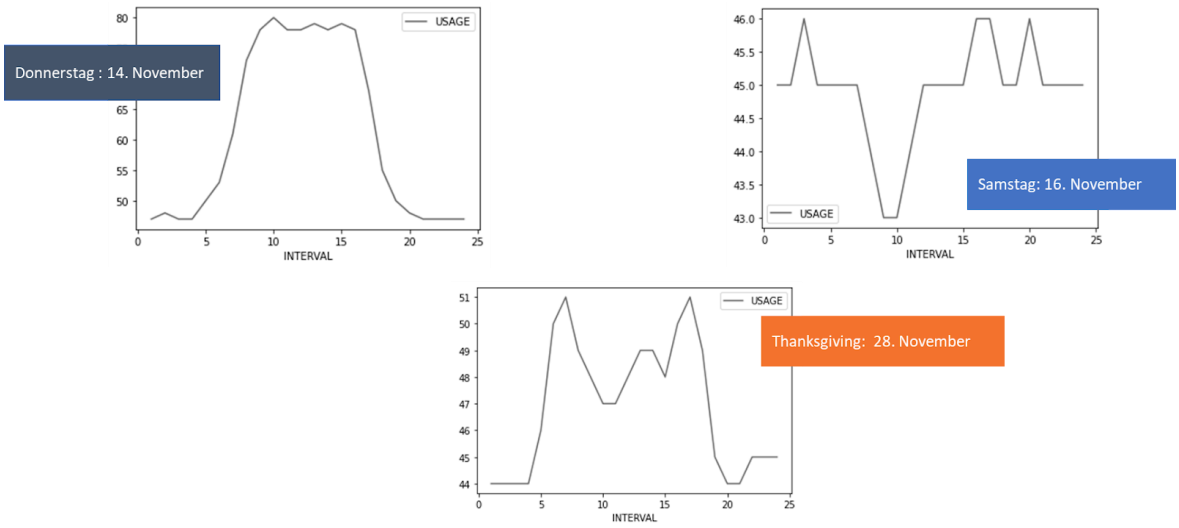


Abbildung 11: CSPT_ID: 6000000249, Tagesvergleich

Eine Auswahl an einer detaillierten Ansicht auf die einzelnen Tage zeigt, dass diese bestimmten Mustern folgen, die sich wöchentlich wiederholen können. Abbildung 11 veranschaulicht die möglichen Muster, die sich an Wochentagen, Wochenenden und Feiertagen gruppieren lassen. Zudem hat die Analyse ergeben, dass der pauschale Verbrauch an den Wochenenden geringer ist als an den Wochentagen. Die einzelnen Diagramme weisen den Strombedarf in kWh für 24 Stunden vor und zeigen explizit die Verteilung der Last mit den Zeitpunkten der Spitzenlast und der Schwachlast. Im Folgenden werden die einzelnen Gruppen in der oben aufgeführten Abbildung näher evaluiert und analysiert.

[51] (Electricity load forecasting for Urban area using weather forecast information, 2016), (Systemeffizienz bei regenerativer Stromerzeugung, 2019)

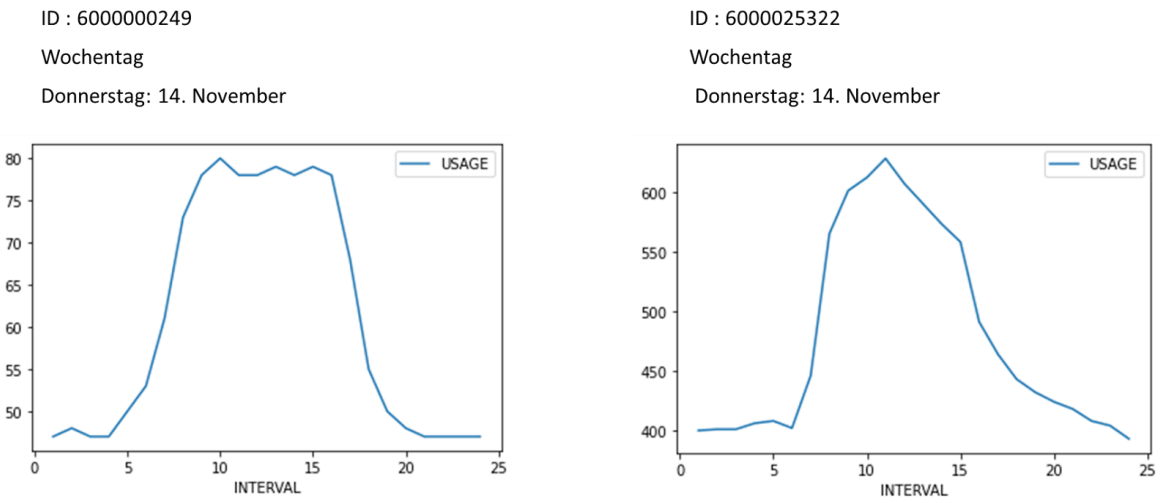


Abbildung 12: Wochentag Zeitreihendiagramme

Abbildung 12 veranschaulicht zwei verschiedene Tages-Lastprofile von zufällig gewählten Verbrauchsstellen an einem Wochentag. Diese zeigen eine deutliche Lastspitze ab dem fünften Intervall, welche sich mit leichten Schwankungen bis zum 20. Intervall hält.
Da bekannt ist, dass die Anzahl der Intervalle sich auf 24 beschränkt, sind diese den Zeitstunden gleichzusetzen. Folglich lässt es darauf schließen, dass die Spitzenlast morgens nach Tagesbeginn steigt, sich über den Tag hält und am Abend zum Tagesende wieder sinkt.

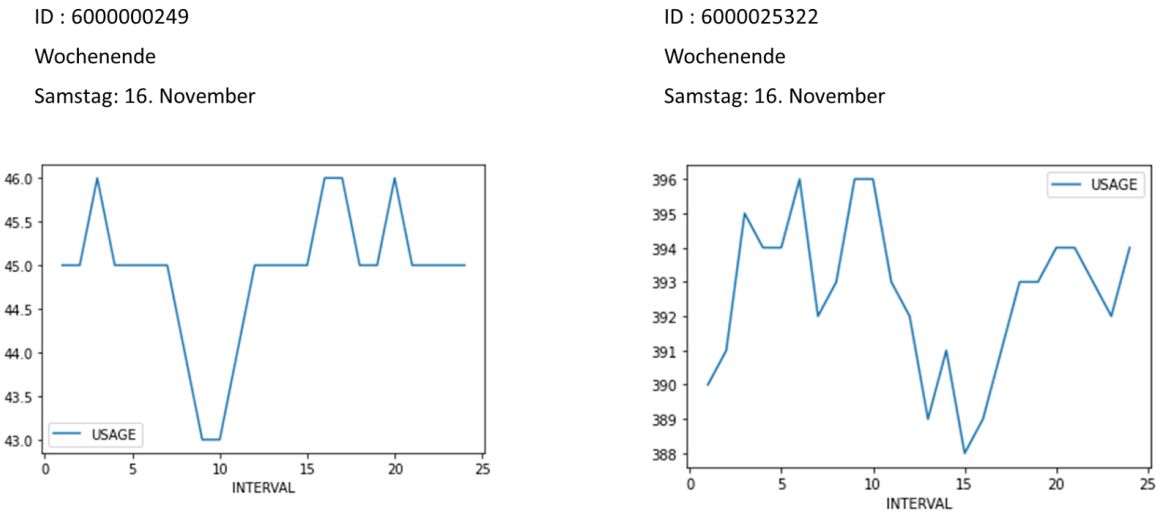


Abbildung 13: Wochenende Zeitdiagramme

In Abbildung 13 werden die Tage-Lastprofile abgebildet, derselben Verbrauchsstellen an einem Wochenende. Verglichen mit den Lastprofilen an den Wochentagen weisen diese Schwankungen über den Tag verteilt auf. Die Verteilung der Last mit den Zeitpunkten der

Spitzenlast und Schwachlast weicht außerdem bei den einzelnen Verbrauchsstellen voneinander ab und ist keinem eindeutigen Muster wie in der vorherigen Abbildung 12 zuzuordnen.

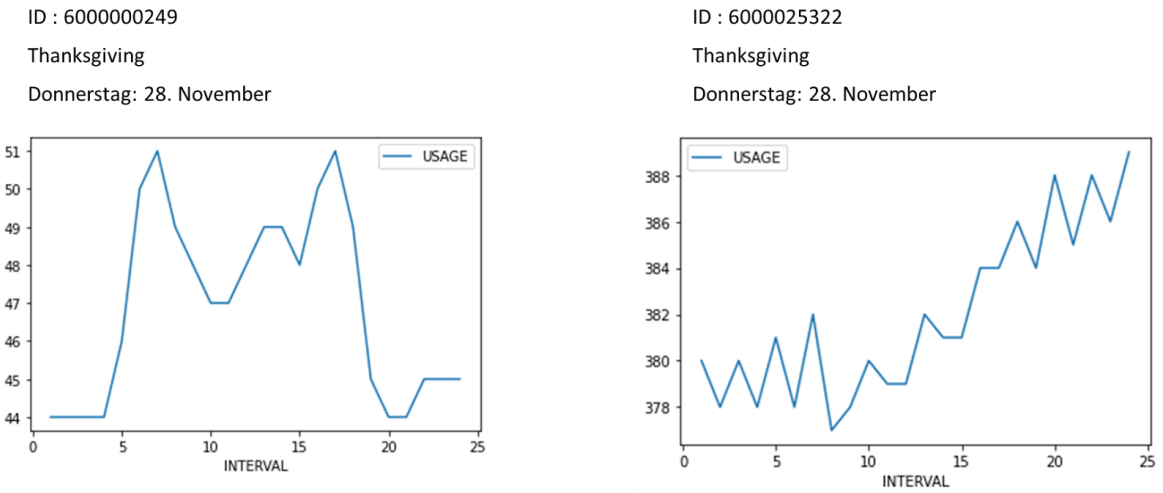


Abbildung 14: Feiertag Zeitreihendiagramme

Abbildung 14 stellt die Tages-Lastkurven der entsprechenden Verbrauchsstelle an einem Feiertag dar. Da die verwendeten Daten aus New Hampshire, USA stammen, wurde ein staatlicher Feiertag der Vereinigten Staaten gewählt. Thanksgiving ist ein in den Vereinigten Staaten und in Kanada gefeiertes Erntedankfest und findet jedes Jahr am vierten Donnerstag des Monats November statt[52]. Die Lastkurve der ID 6000000249 weist einen vergleichbaren Verlauf auf, wie in den Diagrammen der Abbildung 12. Es ist eine Lastspitze vom 5. Intervall zu erkennen, die sich bis zum 20. Intervall hält, welche jedoch mittelmäßige Schwankungen beinhaltet. Die Lastkurve der ID 6000025322 hingegen zeigt eine minimale exponentielle Steigung über den Tag verteilt auf. Durch die deutlichen Unterschiede ist zu verstehen, dass der Lastgang ebenfalls an den Feiertagen kein bestimmtes Muster wie an den Wochentagen aufweist.

52

Abbildung 15: Historische Wetterdaten und Verbrauch New Hampshire 2019

Ein ergänzender Aspekt sind weitere Variablen, die die Lastkurve beeinflussen. Neben Datumsangaben wie Monat, Wochentag, Feiertag oder Werktag sind auch die meteorologischen Daten für die Vorhersage von entscheidender Bedeutung. Es besteht eine enge Korrelation zwischen Wettervorhersagen und der Nachfrage in bestimmten Regionen, vor allem da der Datenbestand Objekte mit Photovoltaikanlagen mit einschließt. Nachweislich steigt die Nachfrage nach Energie, wenn die Gradzahl unter 10° C fällt oder 23° C übersteigt. Grund dafür ist der Kühl- und Heizbedarf[53]. Deutlich zu erkennen ist dies in der obigen Abbildung 15, die den wöchentlichen Bedarf an Energie in Relation zu der wöchentlichen Temperatur anzeigt. Dabei stellt der blaue Graph die Temperatur und der rote Graph den Verbrauch für das ganze Jahr 2019 dar. Es ist deutlich zu erkennen, dass zu Beginn des Jahres aufgrund der niedrigen Temperaturen, die bis zu -5 Grad Celsius sinken, der Verbrauch konträr von 800000 auf 900000 kWh innerhalb einer Woche steigt. Außerdem ist ersichtlich, dass mit fortlaufend steigenden Temperaturen zur Mitte des Jahres genauso der Verbrauch steigt und somit die zuvor genannte Theorie bestätigt wird.

5.2 Anwendung Zeitreihenanalyse

In diesem Teil werden die Methoden, die in Abschnitt 4.3.2. thematisiert wurden, angewendet. Die Analyse wird hier ausschließlich zur Untersuchung einer einzelnen Variable, dem Energieverbrauch, angewendet. Im Vordergrund stehen die zukünftigen Entwicklungen dieser Variable aus ihren historischen Werten zu erläutern[54].

[53] (Electricity load forecasting for Urban area using weather forecast information, 2016)

[54] (Stoetzer, 2020)

5.2.1 Komponenten

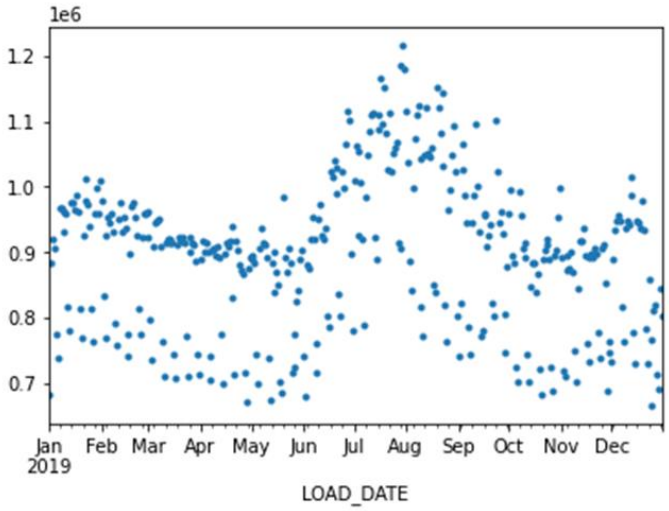


Abbildung 16: Verbrauch der gesamten Lastzone

Der erste Schritt einer Zeitreihenanalyse ist die grafische Darstellung der Variable in einem Streudiagramm[55]. Dafür wurden die gesamten Verbräuche für die jeweiligen Tage gruppiert dargestellt. Bei genauer Betrachtung sind erhebliche Erhöhungen zu den Monaten Juli bis September zu erkennen sowie leichte Erhöhungen zu den Monaten Februar und Dezember. Dieselben Erkenntnisse wurden bereits im vorherigen Abschnitt 5.1. gezogen. Der signifikante Unterschied hierbei ist, dass Abbildung 16 den Verbrauch der gesamten Verbrauchszone darstellt anstelle einer einzelnen Verbrauchsstelle, siehe Abbildung 11-14. Zusätzlich ist aufgrund der Beobachtungen in Abbildung 15 darauf zu schließen, dass diese Schwankungen wetterbedingt auftreten. Zusammengefasst ist aus den gesamten Beobachtungen zu folgern, dass diese Zeitreihe eine ausgeprägte saisonale Komponente besitzt.

5.2.2 Stationarität

Ein weiterer Bestandteil der Zeitreihenanalyse ist das Erkennen der Stationarität. Ein schnelles und einfaches Verfahren ist die Überprüfung der zusammenfassenden Statistik[56]. Dafür werden die Daten in zwei Gruppen geteilt und deren Varianz und Mittelwert miteinander verglichen. Zur Durchführung wird die Bibliothek Pandas verwendet, ein Datenanalyse- und

[55] (Stoetzer, 2020)

[56] (Vishwas, et al., 2020)

-Manipulations-Tool, das auf der Programmiersprache Python aufbaut. Diese stellt Funktionen zur Berechnung des Mittelwerts und der Varianz bereit[57]. Die folgende Abbildung zeigt einen Codeausschnitt, in dem der Energieverbrauch in zwei Partitionen geteilt wird und dessen Werte mithilfe der Funktionen mean(), var() berechnet und ausgegeben werden.

```
#Import packages
import pandas as pd


#mean and variance
data = pd.read_csv('./new_df.csv')
X = data.USAGE


split = round(len(X) / 2)
X1, X2 = X[0:split], X[split:]
mean1, mean2 = X1.mean(), X2.mean()
var1, var2 = X1.var(), X2.var()
print('mean1=%f, mean2=%f' % (mean1, mean2))
print('variance1=%f, variance2=%f' % (var1, var2))

```

Abbildung 17: Codeausschnitt 1 stationary.py

Abbildung 18 zeigt das Ergebnis der Ausführung. Dabei wird deutlich, dass die Mittel- und Varianzwerte für jede Gruppe ähnlich, aber nicht identisch sind. Es gibt eine Vielzahl an Bereichen, beispielsweise der Verkauf, der Einzelhandel, das Gesundheitswesen etc., in denen durch unregelmäßige Verhaltensmuster die Werte aufgrund von Saisonalität und zunehmenden Trends ständig variieren[58]. Folglich ist es möglich, dass die Abweichungen der Ergebnisse von der vorher angenommenen Saisonalität stammen.

```
mean1=36675.504110, mean2=38211.708676
variance1=44250474.692150, variance2=66562413.560692

In [27]:
```

Abbildung 18: Codeausschnitt 2 stationary.py

Aus diesem Grund wird zur weiteren Überprüfung der Stationarität die Ermittlung der Nullhypothese, wie bereits in Abschnitt 4.3.2.2 erwähnt, durchgeführt. Die Intuition hinter einem Einheitswurzeltest ist die Bestimmung, wie stark eine Zeitreihe von einem Trend bestimmt wird. Dabei wird das Ergebnis anhand des p-Werts aus dem Test interpretiert. Ein p-Wert unter einem Schwellenwert von 0.05 deutet darauf hin, dass die Nullhypothese zurückweisen

[57] (Pandas)

[58] (Vishwas, et al., 2020)

wird. Ein p-Wert über dem Schwellenwert deutet jedoch darauf hin, dass die Nullhypothese nicht zurückgewiesen wird und die Reihe nicht stationär ist.

Im Folgenden wird die Berechnung des Augmented-Dickey-Fuller-Tests für den Energieverbrauch veranschaulicht[59]. Die statsmodels-Bibliothek stellt die Funktion adfuller() bereit, die den Test implementiert[60].

```
193    #check adfuller
194    from statsmodels.tsa.stattools import adfuller
195    result = adfuller(df.USAGE.dropna())
196    print(result)
197    print('ADF Statistic: %f' % result[0])
198    print('p-value: %f' % result[1])
199    print('Critical Values:')
200    for key, value in result[4].items():
201        print('\t%s: %.3f' % (key, value))
```

Abbildung 19: Codeauschnitt 1 stationary.py

Der Codeausschnitt 1 zeigt die Anwendung der adfuller()-Methode auf den Energieverbrauch, mit der Eliminierung von Null-Werten sowie die explizite Ausgabe der Teststatistik, dem p-Wert und den kritischen Werten bei 1%, 2,5%, 5% und 10%.

```
ADF Statistic: -11.399389
p-value: 0.000000
Critical Values:
    1%: -3.431
    5%: -2.862
    10%: -2.567
```

Abbildung 20: Codeauschnitt 2 1 stationary.py

Die Ausführung gibt in Abbildung 20 den Teststatistikwert von -11 sowie den P-Wert von 0 aus. Als Teil der Ausgabe wird eine Nachschlagetabelle zur Bestimmung der ADF-Statistik ausgegeben. Dabei ist erkennbar, dass der Statistikwert von -11 kleiner als der Wert von -3,442 bei 1%, -2,862 bei 5% und -2.567 bei 10% ist. Dies deutet darauf hin, dass die Nullhypothese mit einem Signifikanzniveau von weniger als 1% abgelehnt werden kann, der Prozess keine Einheitswurzel hat und die Zeitreihe stationär ist. Infolgedessen müssen keine weiteren Schritte eingeleitet werden, um die Zeitreihe in eine stationäre Reihe zu konvertieren[61].

[59] (Vishwas, et al., 2020)

[60] (Statsmodels)

[61] (Vishwas, et al., 2020), (Stoetzer, 2020)

5.3 Daten Vorbereiten und Zusammenführen

Damit ein idealer Machine Learning Algorithmus implementiert werden kann, müssen die vorhandenen Daten geprüft und vorverarbeitet werden. Das Ziel dabei ist, die Komplexität aus den Daten zu entfernen, um den Lernalgorithmus zu beschleunigen und möglichst wenig Daten zu entfernen, um die Vorhersagekraft zu stärken. Dabei werden standardmäßig drei Schritte befolgt. Beginnend mit der Datenbereinigung, die dazu dient, fehlende Werte mit einem alternativen Wert zu ersetzen, beispielsweise mit einem Mittelwert, Median oder einem passenden Wert, abhängig von der Datenlage. Im Anschluss folgt die Datenauswahl, die sich mit der Variablenauswahl für die Modellentwicklung beschäftigt. In diesem Schritt können Variablen exkludiert werden, die eine größere Menge an fehlenden Daten aufweisen oder zu wenig Aussagekraft haben. Der letzte Schritt, die Transformation der Variablen, dient dazu, die Variablen in geeignete Features umzuwandeln, die der Algorithmus korrekt interpretieren kann. Dazu müssen beispielsweise Kategorien in numerische Werte umgewandelt oder numerische Daten derart transformiert werden, dass sie sich alle auf derselben Skala befinden. In den folgenden Abschnitten werden die vorhandenen Daten auf die beschriebene Weise verarbeitet und für den Machine Learning Algorithmus vorbereitet. Zur Durchführung der Aufbereitung und Bereinigung der Datensätze stehen mehrere Tools zur Verfügung. Beispiele sind die Bibliotheken Pandas, NumPy oder Scikit-learn auf Basis der Programmiersprache Python[62].

5.3.1 Daten Bereinigen

Bevor die Daten für das maschinelle Lernverfahren genutzt werden können, müssen diese im Voraus auf Vollständigkeit geprüft und bereinigt werden. Als Bereinigung werden das Erkennen und das Korrigieren von fehlerhaften Daten bezeichnet. Es gibt verschiedene Methoden, die angewendet werden können; im Allgemeinen wird zwischen einer manuellen, automatisierten und teilautomatisierten Erkennung und Behebung von fehlerhaften Werten unterschieden. Die automatisierte Erkennung funktioniert mithilfe von Programmcode, der fehlerhafte oder fehlende Werte in den Datensätzen automatisch erkennt. Manuell bedeutet, dass die Datensätze ohne Programmcode Datensatz für Datensatz einzeln inspiziert werden. Dies kann jedoch bei einem längeren Datensatz mit mehreren tausend Einträgen sehr kom-

[62] (Buxmann, et al.)

plex werden. Hierzu wurde der Datensatz zu Beginn manuell betrachtet, um spezielle Auffälligkeiten wie fehlende Einträge oder auffällige Werte zu erkennen. Im Anschluss wurde mittels Python Code eine Funktion durchgeführt, die den Datensatz nach fehlenden Einträgen durchsucht. Dies kann jedoch bei einem längeren Datensatz mit mehreren tausenden Einträgen sehr komplex werden. In dieser Arbeit wurde eine teilautomatisierte Erkennung durchgeführt, um fehlerhafte und fehlende Einträge zu erkennen. Dazu wurde der Datensatz zu Beginn manuell betrachtet, um spezielle Auffälligkeiten wie fehlende Einträge oder auffällige Werte zu erkennen. Im Anschluss wurde mittels Python Code eine Funktion durchgeführt, die den Datensatz nach fehlenden Einträgen durchsucht[63]. Abbildung 21 bis Abbildung 23 zeigen Codeausschnitte und die Anzahl der Nullwerte, die in den Datensätzen gefunden wurden.

```
#look for Null-Values in Dataset
print('Count of missing values:\n',df4.shape[0]-df4.count())
```

Abbildung 21: Codeausschnitt 1 heatmap.py

```
Count of missing values:
 time           0
local_time     0
temperature    0
dtype: int64
```

Abbildung 22: Codeausschnitt 2 heatmap.py

```
Count of missing values:
 LOAD_DATE      0
INTERVAL        0
USAGE           0
TEMP            0
DAY_OF_WEEK     0
IS_WEEKEND      0
IS_HOLIDAY      0
dtype: int64
```

Abbildung 23: Codeausschnitt 3 heatmap.py

Da keine fehlenden oder fehlerhaften Einträge gefunden wurden, mussten diese auch nicht ersetzt oder imputiert werden. Diese Überprüfung wurde für den Datensatz des Energieverbrauchs sowie für die Wetterdaten durchgeführt. Da in beiden Datensätzen keine Auffälligkeiten vorhanden waren, konnten diese für den weiteren Verlauf im vorhandenen Zustand weiterverwendet werden.

[63] (Matzka, 2021)

5.3.2 Kalenderdaten

In dieser Arbeit war es von besonderer Wichtigkeit, bestimmte Faktoren beziehungsweise spezielle Features mit einzubinden. Neben den Wetterdaten war es bedeutend, die Kalenderdaten mit einzubeziehen. Dies ist bereits in Abschnitt 5.1 zu beobachten, da deutlich wird, dass der Energieverbrauch abhängig von den verschiedenen Kalendertagen ist. Aus diesem Grund wurden die verschiedenen Informationen zu den Kalenderdaten aus der Variable LOAD_DATE extrahiert und in separate Variablen zum Datensatz hinzugefügt. Inbegriffen ist der Tag der Woche als numerischer Wert, die Information, ob es sich um einen Wochentag oder ein Wochenende handelt als boolescher Wert und zuletzt der boolesche Wert, ob es sich um einen Feiertag handelt. Folgende Abbildungen zeigen die verwendeten Funktionen, um diese Informationen zu extrahieren und abzubilden.

```
#getweekends
#finalDf["LOAD_DATE"] = pd.to_datetime(groupBy["LOAD_DATE"])
new_df["DAY_OF_WEEK"] = new_df["LOAD_DATE"].dt.weekday
```

Abbildung 24: Codeausschnitt Tag der Woche

Mithilfe des Datetime-Moduls, welches Python zur Verfügung stellt, ist es möglich, den Tag als numerischen Wert darzustellen. In diesem Fall beginnen die Werte bei 0 für den Montag und zählen hoch bis zu dem Wert 6 für den Sonntag.

```
# check if the date is weekend or not
new_df["IS_WEEKEND"] = new_df["DAY_OF_WEEK"] >= 5
```

Abbildung 25: Codeausschnitt Wochenende

Die Ermittlung der Wochenenden wurde im Anschluss mithilfe der Information über den Tag der Woche bestimmt. So konnten alle Zahlen, die kleiner gleich fünf waren, folglich als Wochentage definiert werden.

```
#is an holiday?
from pandas.tseries.holiday import USFederalHolidayCalendar as calendar
dr = pd.date_range(start='2019-01-01', end='2019-12-31')
cal = calendar()
holidays = cal.holidays(start=dr.min(), end=dr.max())
new_df['IS_HOLIDAY'] = new_df['LOAD_DATE'].isin(holidays)
```

Abbildung 26: Codeausschnitt Feiertag

Die Definition der Feiertage wurde mittels der Bibliothek Pandas ermittelt, welche bereits einen US-Feiertagskalender inkludiert. Mit Hilfe der vorhandenen Funktionen konnte bestimmt werden, welche Tage im Jahr 2019 Feiertage waren und diese als boolesche Werten ausdrücken.

5.3.3 Harmonisierung der Zeitreihen

Die Datensätze des Energieverbrauchs und die Wetterdaten der Region in New Hampshire sollen nun verknüpft werden. Jedoch weist der Datensatz mit dem Energieverbrauch Intervalle beginnend bei 1 bis 24 vor. Verglichen dazu verwendet der Wetterdatendatensatz Zeitstunden von 00:00 Uhr bis 23:00 Uhr. Aufgrund der unterschiedlichen Start- und Endzeiten ergibt sich ein Problem. Daher musste vor einer Zusammenführung der Daten eine Harmonisierung erfolgen. Dafür ist ein Zeitpunkt zu definieren, zu dem die Werte bestimmt werden sollen. In diesem Fall wurde eine Haupt-Datenquelle bestimmt, deren Zeitreihe für die Harmonisierung verwendet wird[64]. Als Haupt-Datenquelle wurde der Datensatz mit den Intervallen gewählt. Mittels Python-Codes wurden die Stundenzahlen in Integer Werte konvertiert, beginnend mit dem Intervall 1 anstelle der Zeitstunde 00:00 Uhr. In Abbildung 27 ist ein Codeausschnitt zu finden, der diese Konvertierung veranschaulicht. Im Anschluss wurden beide Datensätze anhand des Datums und der Intervalle aufeinander gemappt und schließlich miteinander verknüpft.

```
#df2.replace({'INTERVAL' : { '00:00:00' : '1', '01:00:00' : '2', '02:00:00'
df2['INTERVAL'] = df2['INTERVAL'].astype(str).str.replace('00:00:00','1')
df2['INTERVAL'] = df2['INTERVAL'].astype(str).str.replace('01:00:00','2')
df2['INTERVAL'] = df2['INTERVAL'].astype(str).str.replace('02:00:00','3')
df2['INTERVAL'] = df2['INTERVAL'].astype(str).str.replace('03:00:00','4')
```

Abbildung 27: Codeausschnitt 4 heatmap.py

Die folgende Abbildung zeigt einen Ausschnitt aus dem finalen Datensatz, der für alle weiteren Schritte verwendet wurde.

[64] (Matzka, 2021)

Index	LOAD_DATE	INTERVAL	USAGE	temperature	radiation_surface	radiation_toa	DAY_OF_WEEK	IS_WEEKEND	IS_HOLIDAY
0	2019-01-01 00:00:00	1	25687	2.252	0	0	1	False	True
1	2019-01-01 00:00:00	2	25507	1.988	0	0	1	False	True
2	2019-01-01 00:00:00	3	25177	1.505	0	0	1	False	True
3	2019-01-01 00:00:00	4	25161	1.348	0	0	1	False	True
4	2019-01-01 00:00:00	5	25674	1.318	0	0	1	False	True
5	2019-01-01 00:00:00	6	27276	1.18	0	0	1	False	True
6	2019-01-01 00:00:00	7	29109	1.448	0	0	1	False	True
7	2019-01-01 00:00:00	8	29666	2.069	0	0	1	False	True
8	2019-01-01 00:00:00	9	29547	2.689	0	0	1	False	True
9	2019-01-01 00:00:00	10	30278	3.05	0	0	1	False	True
10	2019-01-01 00:00:00	11	30202	3.519	0	0	1	False	True
11	2019-01-01 00:00:00	12	29974	3.047	0	0	1	False	True
12	2019-01-01 00:00:00	13	29798	2.785	15.056	56.181	1	False	True
13	2019-01-01 00:00:00	14	29190	3.942	100.515	250.941	1	False	True
14	2019-01-01 00:00:00	15	29592	5.293	222.506	412.648	1	False	True
15	2019-01-01 00:00:00	16	29602	6.514	318.715	520.719	1	False	True

Abbildung 28: finaler Datensatz

5.3.4 Feature Auswahl

Für die Feature Auswahl wurde wie bereits bekannt eine Vorauswahl durchgeführt. Da für diese Arbeit die Untersuchung der meteorologischen Daten und die Kalendertage signifikant waren, wurden diese selbstverständlich mit einbezogen. Da es wichtig ist nachzuvollziehen, wie die einzelnen Variablen in Korrelation zu dem Energieverbrauch stehen, wurde mithilfe von Python eine Heatmap erstellt, um diese Korrelation visuell zu veranschaulichen.

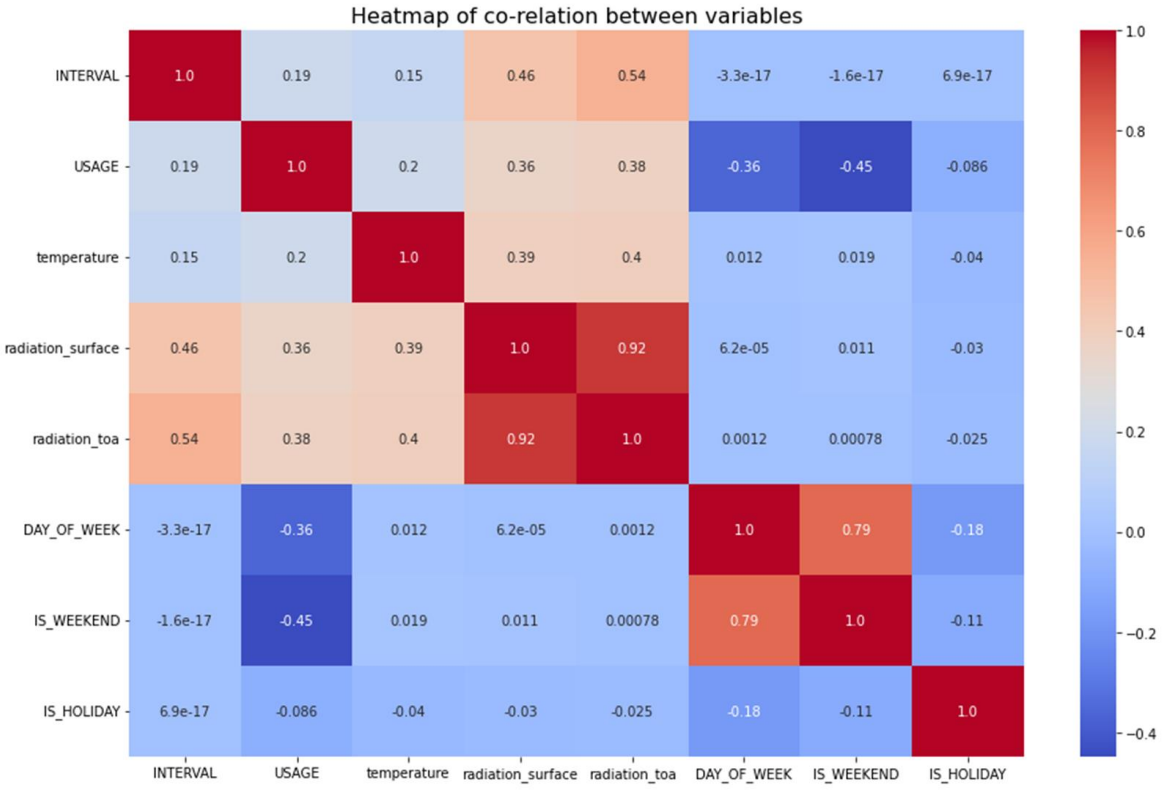


Abbildung 29: Heatmap

Die Abbildung veranschaulicht die einzelnen Features und deren Korrelation zueinander. Dabei bedeutet die Farbe Rot positiv und Blau negativ. Je kräftiger die Farbe ist, desto größer ist die Korrelation. Aus der Heatmap geht hervor, dass alle Variablen negativ mit „Usage“ korrelieren. „DayofWeek“ und „isWeekend“ sind die beiden am stärksten negativ korrelierten Merkmale. Hinzu kommt, dass beide Merkmale multikollinear sind. Die negative Korrelation zwischen zwei Variablen definiert, dass eine Variable zunimmt, während die andere abnimmt und umgekehrt. Diese Beziehung kann eine Kausalität zwischen beiden Variablen darstellen. Hervorzuheben ist die Korrelation der Sonneneinstrahlung mit der Temperatur, dem Intervall und dem Verbrauch, was auf die Tatsache zurückzuführen ist, dass der Datenbestand Objekte mit Photovoltaikanlagen beinhaltet.

5.4 Prognose der Stromlast

Für die Stromlastprognose werden die vorgestellten Algorithmen aus Kapitel 3 verwendet und gegenübergestellt. Die Experimente umfassen dabei die univariaten und multivariaten Zeitreihenanalysen. Zu Beginn wurde die univariate Zeitreihenanalyse mittels LSTM und Random Forest durchgeführt, mit dem Blick ausschließlich auf den Verbrauch. Im Anschluss wurde eine multivariate Zeitreihenanalyse durchgeführt bzw. eine Analyse mit allen im vorherigen Abschnitt genannten Variablen. Die Gründe für eine univariate Zeitreihenanalyse mit folgender multivariater Zeitreihenanalyse dienten zum einen dazu, ein besseres Verständnis für den LSTM Algorithmus zu erlangen und zum anderen beide Methoden und Algorithmen miteinander zu vergleichen. In den folgenden Abschnitten werden die einzelnen Algorithmen und deren Ergebnisse vorgestellt und evaluiert.

5.4.1 LSTM univariate Zeitreihenanalyse

Für die Verwendung des LSTM-Modells wurde die Deep-Learning-API Keras verwendet, die in Python geschrieben ist und auf der Plattform für maschinelles Lernen TensorFlow aufbaut[65]. Für die Umsetzung der univariaten Zeitreihenanalyse wird lediglich eine Variable betrachtet (Multivariate Time Series Forecasting with LSTM for Madrid, Spain pollution, 2019). Das bedeutet, es werden nur die historischen Daten des Verbrauchs in Kilowattstunden eingesetzt und durch eine bestimmte Methode als überwachtes Lernen ausgedrückt. Dabei wird eine Folge von historischen Zeitschritten als Eingabe festgelegt und der nächste

[65] (Keras)

Zeitschritt als Ausgabe genutzt. Die Größe des zu betrachtenden Fensters kann dabei nach Bedarf definiert werden und nach Länge einen spürbaren Einfluss auf die Prognoseergebnisse haben[66]. Abbildung 30 zeigt anhand eines Beispiels, wie die beschriebene Methode arbeitet. Das Beispiel verwendet im ersten Schritt X1 bis X24 als Eingabemerkmale, um den Wert X25 vorherzusagen. Anschließend werden X2 bis X25 als Eingabemerkmale für die Vorhersage von X26 verwendet, daraufhin wiederholt sich dasselbe ein weiteres Mal. Dieser Schritt wiederholt sich mehrfach bis zum letzten Datenfeld der Zeitreihe. Die einzelnen Werte für die Eingabe und Ausgabe werden separat gesammelt und im weiteren Verlauf eingesetzt.

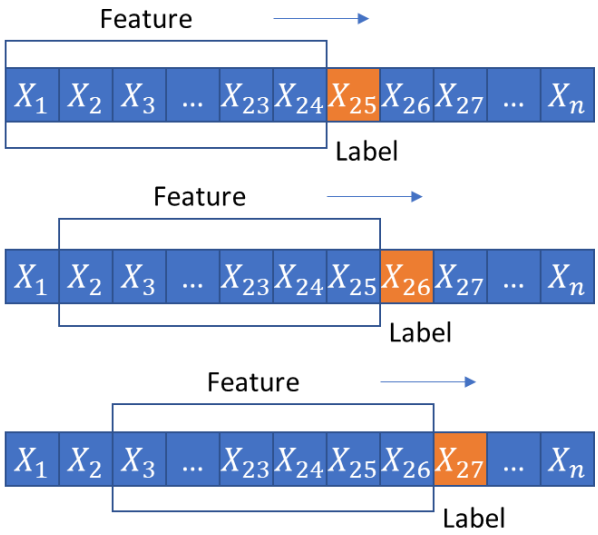


Abbildung 30: Sliding Window Methode

Schließlich werden die Eingabemerkmale in das von dem LSTM erwartete 3D-Format umgestaltet, nämlich [Anzahl der Stichproben, Anzahl der Zeitschritte, Anzahl der Merkmale]. Die Anzahl der Stichproben gibt die Anzahl der untersuchten Daten an. Die Anzahl der Zeitschritte beträgt je ein Zeitintervall, gleichzusetzen mit einer Stunde. Des Weiteren wird die Anzahl mit acht Merkmalen festgelegt.

Das Modell definiert eine Eingabeschicht mit einer versteckten Schicht, die 50 LSTM-Neuronen einschließt und eine Ausgabeschicht, die eine Einzelwertvorhersage erzeugt. Das Netz wird für 100 Epochen trainiert und verwendet eine Batch Size von 64. Die Batch Size gibt an, wie viele Input-Datensätze auf einmal durch das Netzwerk geschleust werden. Es splittet

[66] (Kahraman, et al.)

das Trainingsset in kleinere Segmente, aus denen gelernt wird, um beispielsweise die unterliegende Rechnerarchitektur zu berücksichtigen. Sobald alle Trainingsdaten das Netzwerk einmal vollständig durchlaufen haben, wird von einer Epoche gesprochen. Die Anzahl der Epochen nimmt Einfluss auf die Prognose. Grundsätzlich lässt sich feststellen, je mehr Daten das Netzwerk zur Verfügung hat und je öfter es diese gesehen hat, desto kleiner ist der Fehlerwert[67].

```
72
73  model = Sequential()
74  model.add(LSTM(50, input_shape=(trainX.shape[1], trainX.shape[2])))
75  model.add(Dense(1))
76  model.compile(loss='mae', optimizer='adam')
77  history = model.fit(trainX, trainY, epochs=100, batch_size=64, validation_data=(testX, testY), verbose=1, shuffle=False)
78
```

Abbildung 31: Codeausschnitt 1 LSTM_uni.py

Abschließend kann mit dem trainierten Modell eine Prognose für die Testmenge erstellt werden, die mittels Root Mean Square Error (RMSE) ausgewertet wird. Der RMSE berechnet die Differenz zwischen den beobachteten Daten und dem geschätzten Wert[68]. Dieser liegt bei 1458.167 und wird im weiteren Verlauf mit den weiteren Algorithmen verglichen. Zusätzlich zu jeder Prognose werden die zugehörigen Diagramme erstellt, die zum einen den Trainings- und den Testverlust während des Trainings anzeigen und zum anderen die erwarteten Werte den vorhergesagten Werten gegenüberstellen.

[67] (Wennker, 2020)

[68] (Bedendo, et al., 2014)

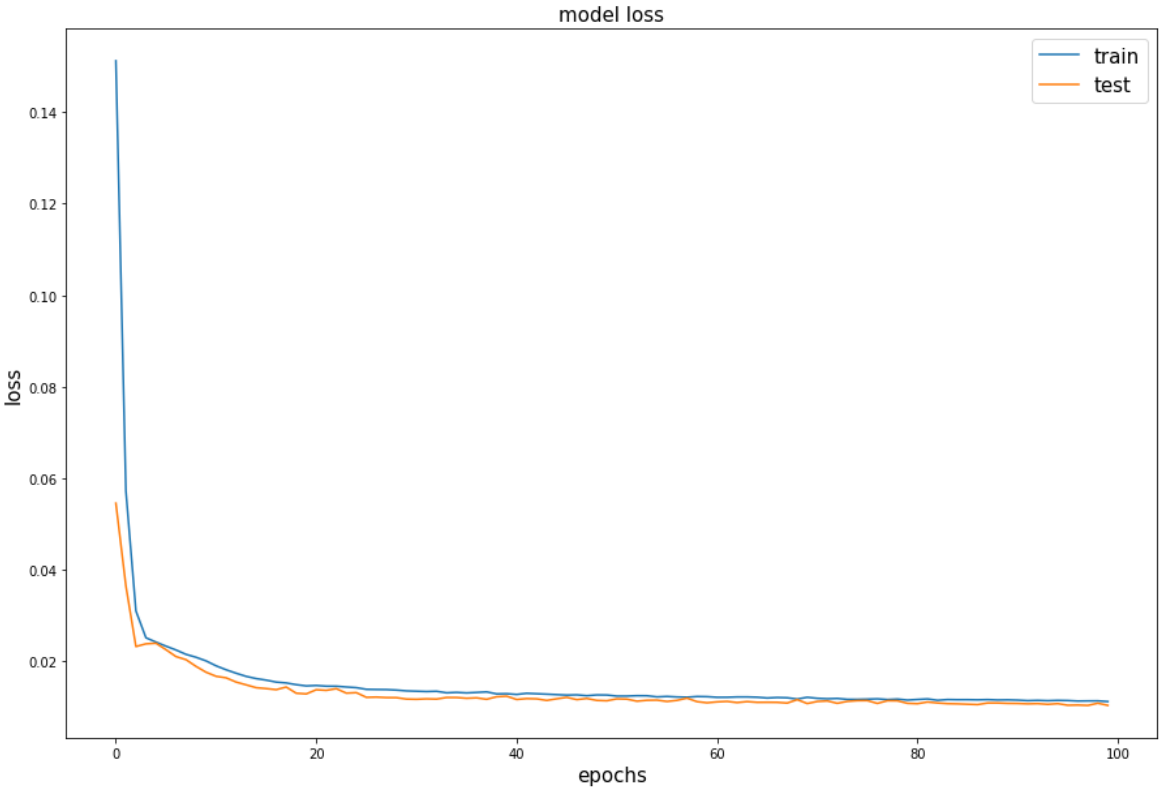


Abbildung 32: LSTM_uni Trainings-und Testverlust

Abbildung 32 zeigt den Verlauf der Trainings- und Testverlustkurve. Dabei definiert die X-Achse die Epochen und die Y-Achse den Fehlerwert. Es ist zu beobachten, dass beide Kurven bis zu einem gewissen Punkt der Stabilität stark sinken, dabei befindet sich die Testkurve mit einem größeren Abstand zur Trainingskurve. Ab Punkt der Stabilität sinken beide Kurven mit einem minimalen Abstand zueinander leicht weiter und enden bei einem sehr niedrigen Fehlerwert. Diese Merkmale weisen darauf hin, dass kein Over- oder Underfitting vorhanden ist.

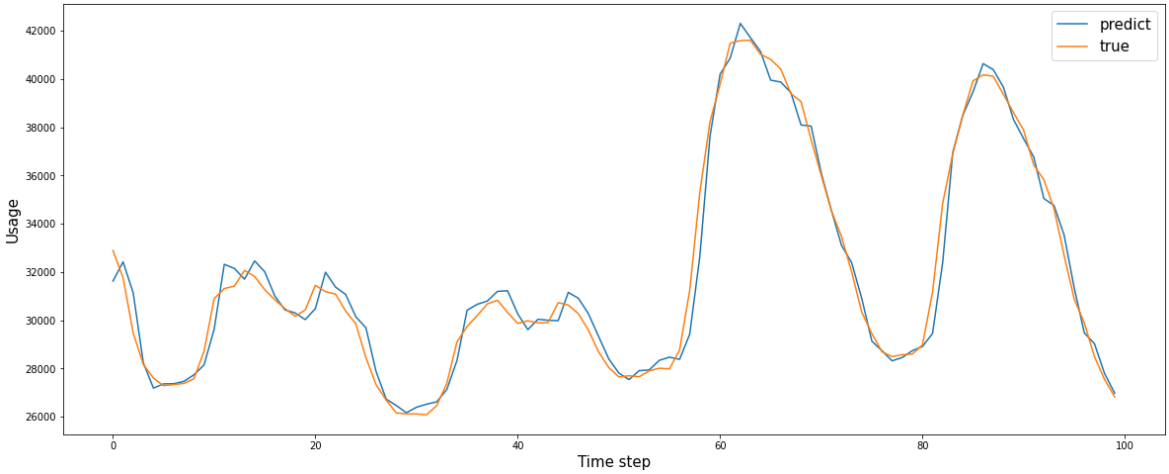


Abbildung 33: LSTM_uni erwartete und prognostizierte Werte

Abbildung 33 stellt abschließend einen Ausschnitt der letzten 100 erwarteten und prognostizierten Werte in einem Diagramm dar. Dabei zeigt die X-Achse die einzelnen Zeitschritte, auf denen die stündlichen Intervalle beruhen und die Y-Achse den Verbrauch in Kilowattstunden. Es ist deutlich zu sehen, dass das Modell sehr gute Ergebnisse erzielt und in der Lage ist, die meisten Steigungen und Senkungen genau zu verfolgen. Die erworbenen Ergebnisse implizieren folglich, dass es möglich ist, durch univariate Zeitreihenanalysen mittels LSTM den Verbrauch mit großer Genauigkeit vorherzusagen.

5.4.2 Random Forest univariate Zeitreihenanalyse

Für die Verwendung des Random Forest wurde die Scikit-learn-API verwendet, welche die Klasse „RandomForestRegressor“ in Python anbietet[69]. Hierbei werden wie im vorherigen Modell nur die historischen Daten des Verbrauchs in Kilowattstunden eingesetzt und durch dieselbe Methode wie in Abbildung 30 als überwachtes Lernen ausgedrückt. Der einzige Unterschied in der Programmierung zeigt sich dadurch, dass die Trainingsdaten in kein bestimmtes Format für das Modell umgestaltet werden müssen und die Klasse sofort verwendet werden kann.

```
69
70    model = RandomForestRegressor(n_estimators=100,max_depth=5)
71    model.fit(trainX, trainY)
72
```

Abbildung 34: Codeausschnitt 1 RF_uni.py

[69] (Scikit-learn)

Der „RandomForestRegressor“ bringt verschiedene Hyperparameter mit, die Einfluss auf das Resultat haben können. Bei der Umsetzung war zu beobachten, dass die Tiefe des Baums einen signifikanten Unterschied bewirkte - je kleiner die Zahl, desto ungenauer war die Prognose. Ab der Tiefe Fünf war letztlich kein erkennbarer Unterschied mehr zu erkennen. Daher wurde für dieses Modell die Tiefe Fünf gewählt. Angesichts der Anzahl der Bäume wurden 100 festgelegt, da diese Zahl sich positiv auf die Laufzeit auswirkte. Durch die Zugabe der Hyperparameter waren ebenso Unterschiede an der Test- und Trainingskurve zu beobachten. Durch das Hinzufügen des Parameters für die maximale Tiefe war eine Bewegung beider Kurven zueinander deutlich. Wenn indessen keine Tiefe angegeben wurde, kann durch den Kurvenverlauf eine Überanpassung ermittelt werden. Dies ist möglicherweise auf die Tatsache zurückzuführen, dass, wenn keine maximale Anzahl für die Tiefe angegeben wird, die Knoten so lange erweitert werden, bis alle Blätter vollständig ausgefüllt sind[70]. Ergänzend lässt sich aus dem aktuellen Kurvenverlauf in Abbildung 35 ableiten, dass mit einer größeren Datenlage die Kurven irgendwann zueinander laufen würden.

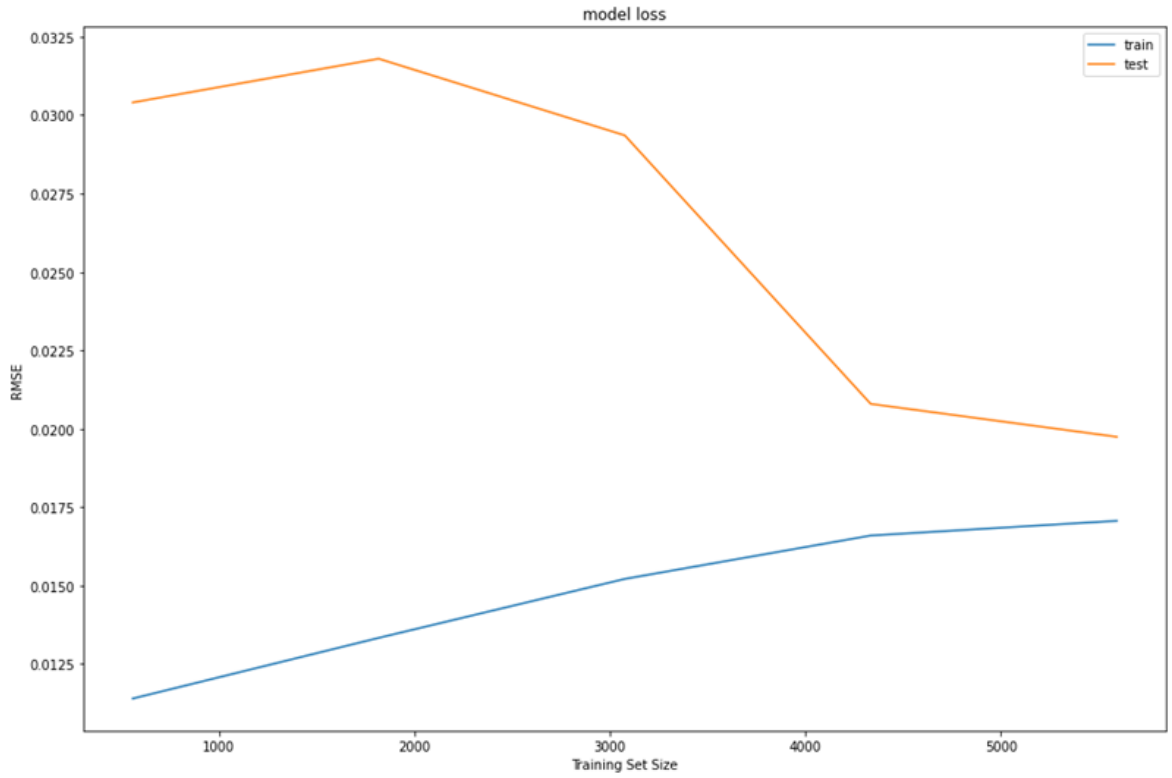


Abbildung 35: RF_uni Trainings-und Testverlust

[70] (Scikit-learn)

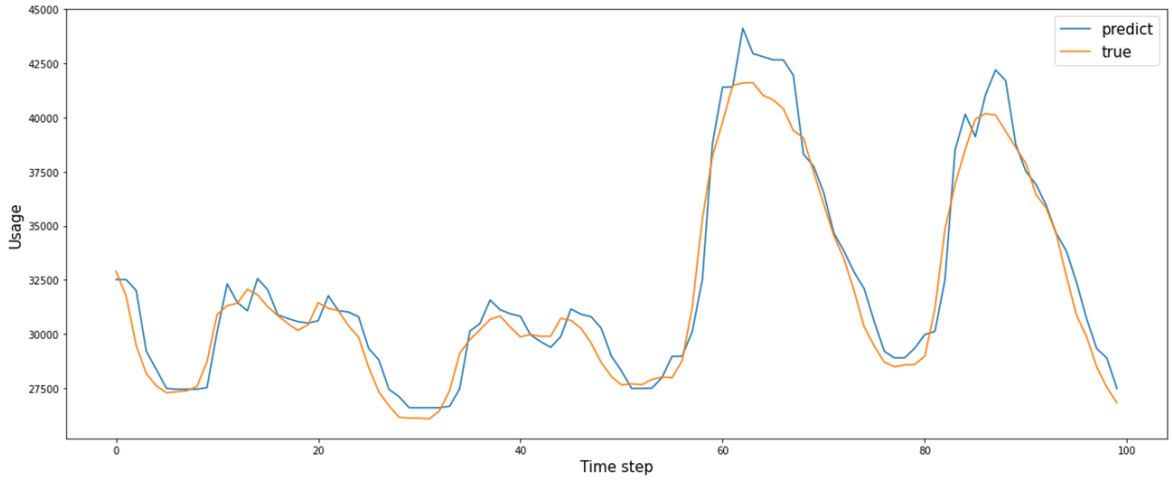


Abbildung 36: RF_uni erwartete und prognostizierte Werte

Das Liniendiagramm in Abbildung 36 bildet auch hier die letzten 100 Zeitschritte der Prognose ab. Der Verlauf ist sehr ähnlich dem vorherigen Modell. Trotz der auffälligen Spitzen in der Kurve der vorhergesagten Werte schneidet dieses Modell mit einem RMSE-Wert von 1308.446 minimal besser ab und ermöglicht ebenso eine gute Prognose.

5.4.3 LSTM multivariate Zeitreihenanalyse

Zuletzt wird die multivariate Zeitreihenanalyse durchgeführt, die in dieser Arbeit fokussiert wird. Die Umsetzung gleicht der univariaten Zeitreihe sehr. Sie unterscheiden sich jedoch dadurch, dass die Zeitreihe mehr als eine zeitabhängige Variable besitzt. Jede Variable hängt von ihren vergangenen Werten ab sowie von Vergangenheitswerten anderer Variablen[71]. Dies inkludiert die Temperatur, Informationen zu den einzelnen Kalendertagen, Zeitintervallen, Solareinstrahlung sowie den Verbrauch in Kilowattstunden. Es wird eine ähnliche Methode wie in Abschnitt 5.4.1. verwendet, um die Daten für ein überwachtes Lernen umzustrukturieren. Das betrachtete Fenster weist dabei jedoch nicht nur historische Werte des Verbrauchs auf, sondern ebenfalls die oben genannten Variablen, um den nächsten Zeitschritt vorherzusagen. Abbildung 37 verdeutlicht dies genauer.

[71] (Multivariate Time Series Forecasting with LSTM for Madrid, Spain pollution, 2019)

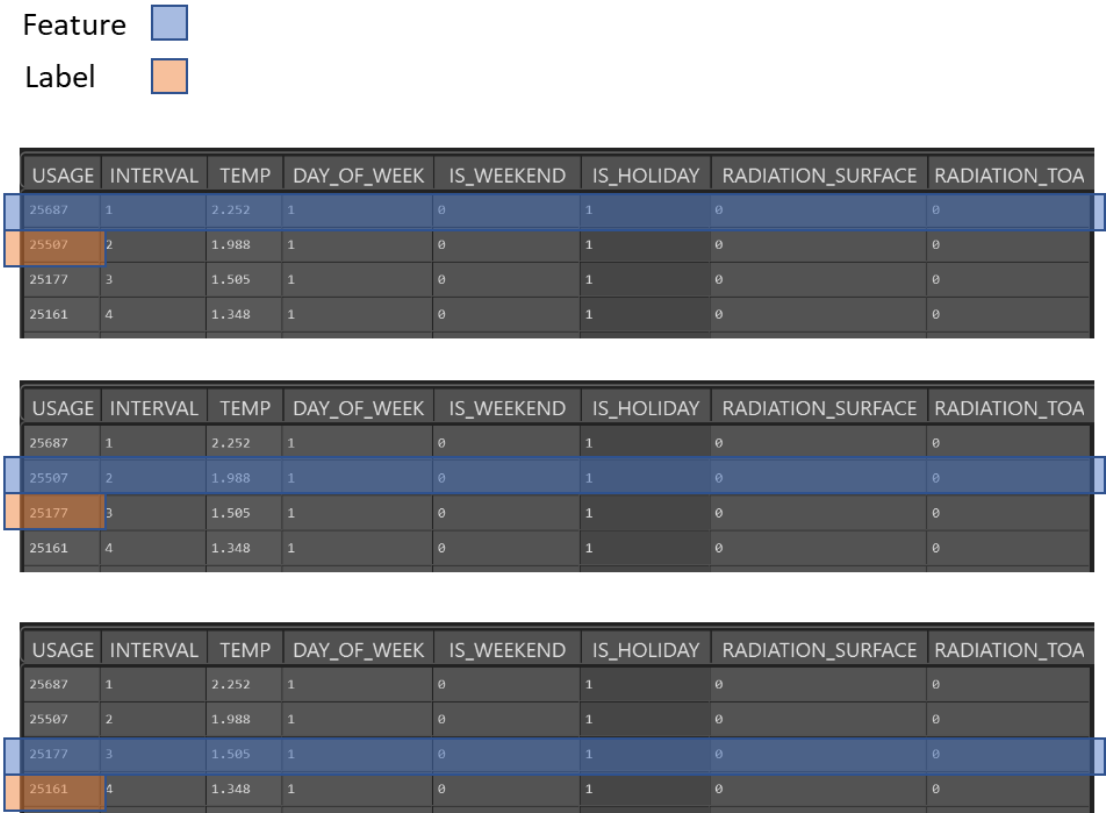


Abbildung 37: LSTM multivariate supervised learning

Anschließend werden die Eingabemerkmale für die LSTM Modellanforderungen umgewandelt. Die Anzahl der Zeitschritte beträgt eine Zeitstunde und die Anzahl der Merkmale sechs.

Innerhalb der Umsetzung wurden die verschiedenen Experimente durchgeführt, um die Hyperparameter derart anzupassen, dass ein weitgehend positives Ergebnis erzielt werden konnte. Dabei wurde die Anzahl der LSTM-Nodes modifiziert sowie die Anzahl der Epochen und die Batchsize. Auch eine Zugabe von Dropouts oder das Hinzufügen weiterer Schichten hat die Prognose teilweise verschlechtert. Letztendlich hatte ausschließlich die Erhöhung der Anzahl der Epochen positive Auswirkungen auf das Ergebnis. Jedoch hatte eine enorme Erhöhung bis zu 6000 zwar den RMSE-Wert verbessert, erforderte allerdings eine erhöhte Laufzeit und veränderte die Trainings- und Testkurve negativ. Infolgedessen wurde eine Anzahl von 1000 Epochen gewählt. Somit ergab sich ein Modell mit 50 Neuronen in der ersten versteckten Schicht und einem Neuron in der Ausgabeschicht für die Vorhersage des Verbrauchs. Die Eingabeform interpretiert dabei 1000 Trainingsepochen mit einer Batch-Size von 64.

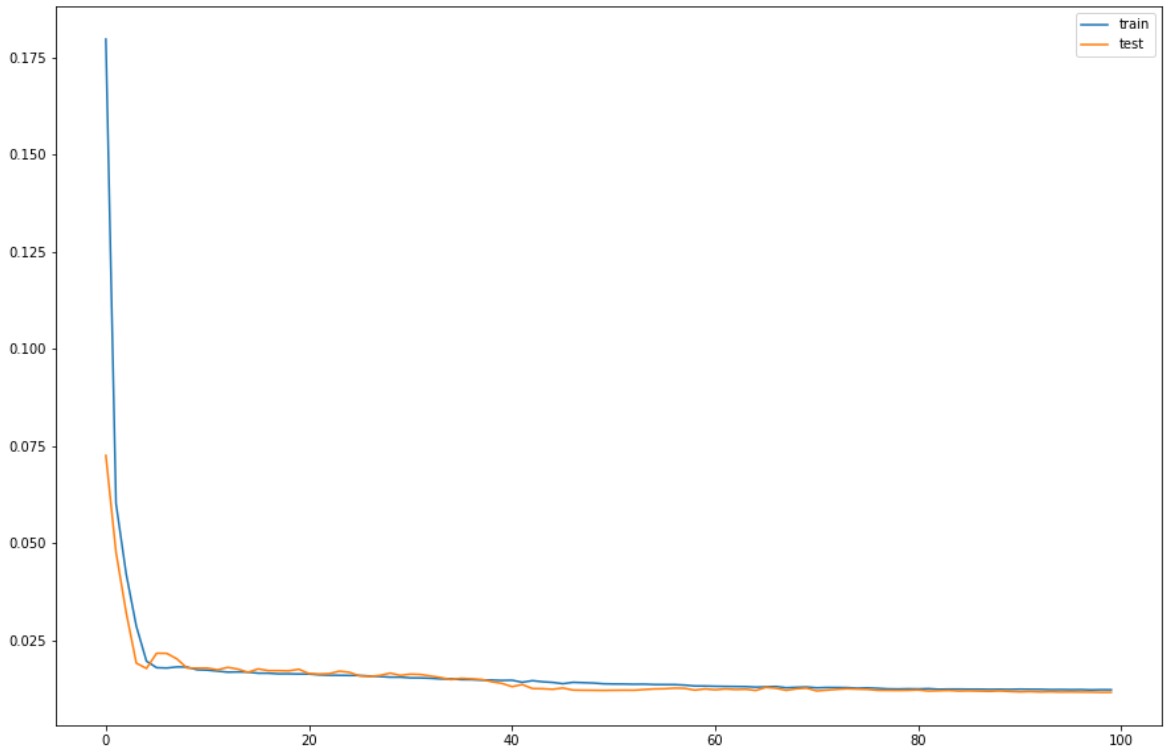


Abbildung 38: LSTM_multi Trainings-und Testverlust

Abbildung 38 zeigt den Verlauf der Trainings- und Testverlustkurve. Dabei definiert die x-Achse die Anzahl der Epochen und die y-Achse den Fehlerwert. Es wird deutlich, dass beide Kurven mit einem minimalen Abstand zueinander bis zu einem gewissen Punkt der Stabilität stark sinken. Ab dem Wert von ca. 0.025 sinken beide Kurven leicht weiter und enden bei einem Fehlerwert von nahezu 0.00. Diese Merkmale deuten ebenso auf ein positives Ergebnis hin.

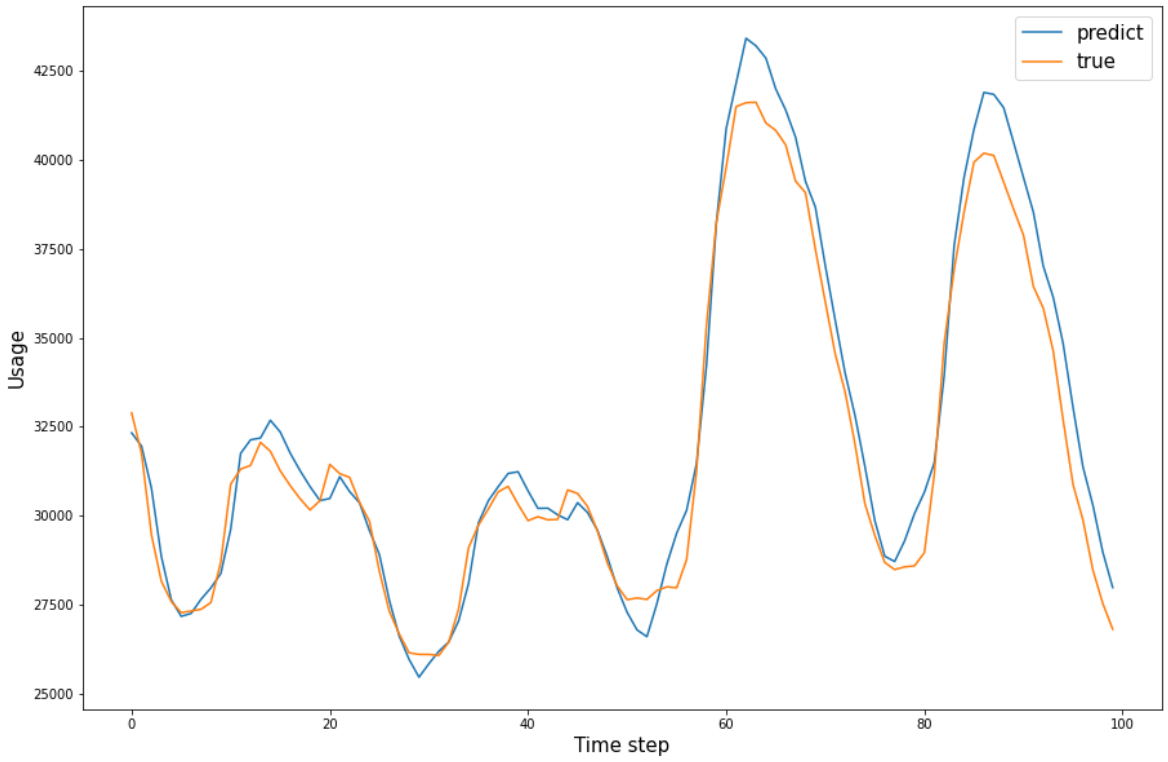


Abbildung 39: LSTM_multi erwartete und prognostizierte Werte

Ein Blick auf den Ausschnitt der letzten 100 erwarteten und prognostizierten Werte zeigt, dass verglichen mit den anderen Algorithmen ebenfalls sehr gute Ergebnisse erzielt werden können. Auch hier ist das Modell in der Lage, die meisten Steigungen und Senkungen genau zu verfolgen. Verglichen mit dem LSTM-Modell und dem Random-Forest-Modell mittels einer univariaten Zeitreihenanalyse erzielt die Prognose den besten RMSE-Wert von 1173.426. Trotz allem liegt dieser nicht weit von den anderen entfernt. Zusammengefasst implizieren die erworbenen Ergebnisse, dass es gleichermaßen möglich ist, durch eine multivariate Zeitreihenanalyse mittels LSTM den Verbrauch mit großer Genauigkeit in New Hampshire vorherzusagen. Dennoch wurde erwartet, dass die Zugabe von weiteren Variablen den Algorithmus weitaus verbessern würde, was sich jedoch nicht betätigt hat.

5.5 Experiment

Mit Hilfe der gemessenen RMSE-Werte, die in der nachstehenden Tabelle zusammengetragen wurden, ist festzustellen, dass das Modell mit der multivariaten Zeitreihenanalyse das

beste Resultat realisiert. Allerdings unterscheiden sich die einzelnen Werte nur minimal voneinander. Anhand dessen ergibt sich folgende weiterführende Frage: Wieso hat die Zugabe von weiteren Informationen keinen wesentlichen Einfluss auf die Prognose?

Modell	**RMSE**
LSTM univariate	1458.167
Random Forest univariate	1308.446
LSTM multivariate	1173.426

Aus diesem Grund wurde ein Experiment durchgeführt, um diese Fragestellung zu untersuchen. Die grundlegende Annahme dabei ist, dass dieser Anwendungsfall mit der vorhandenen Datenlage nicht komplex genug ist, um die Fähigkeiten der einzelnen Modelle völlig auszuschöpfen und somit keine deutlichen Unterschiede zwischen den Modellen zu erkennen sind.
Für das Experiment wurde eine Mittelwertberechnung mittels Programmcode durchgeführt. Dabei wurden alle historischen Werte eines bestimmten Wochentages, beispielsweise eines Donnerstags, zusammengetragen und dessen Mittelwert bestimmt. Das Ergebnis stellte anschließend die Prognose für den letzten Donnerstag des Jahres dar. Diese Berechnung wurde für jeden Wochentag durchgeführt, um auf diese Weise eine Vorhersage für die letzte Woche des Jahres nachzuahmen.

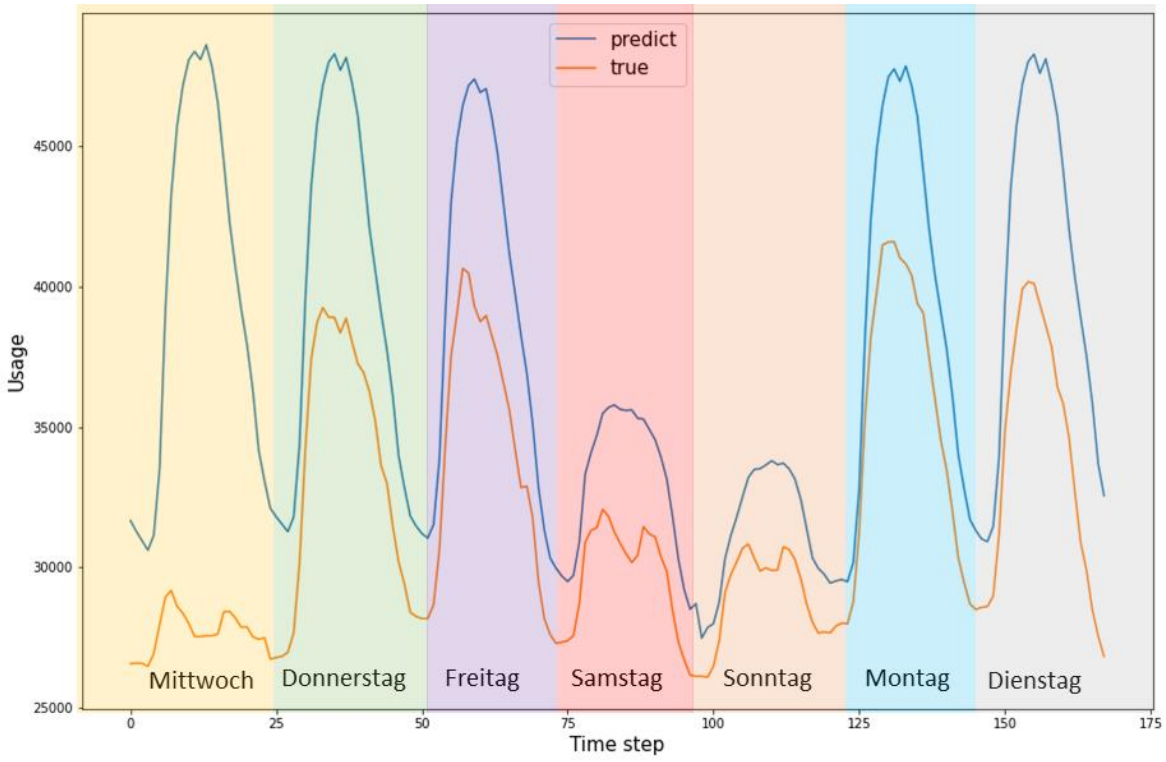


Abbildung 40: Mittelwertberechnung

Abbildung 40 repräsentiert die Ergebnisse vom 25. Dezember 2019 bis zum 31. Dezember 2019. Es ist auffällig, dass allein durch eine Mittelwertberechnung der reguläre Verlauf einer Woche nachgebildet werden konnte. Tatsächlich verlaufen die Kurven nicht wie bei einer Prognose mit einem minimalen Abstand zueinander, jedoch sind alle signifikanten Steigungen und Abfälle zu erkennen. Durch die explorative Datenanalyse ist bekannt, dass an den Wochentagen ein höherer Verbrauch vorliegt als an den Wochenenden. Dies ist ebenfalls deutlich in dem Ergebnis zu beobachten. Der Grund für den deutlich geringeren Verbrauch am Mittwoch, den 25. Dezember könnte damit zusammenhängen, dass dieser der 1.Weihnachtstag und somit ein Feiertag ist[72]. Zusammenfassend bestätigt dies das vorhandene Ergebnis und bestärkt die Annahme, dass beim Stromverbrauch kontinuierlich wiederholende Muster auftreten und daher alle drei Modelle ein ähnliches Ergebnis erzielen.

[72] (OfficeHolidays)

6 Fazit und Ausblick

Lastprognosen können bei der Gestaltung eines geeigneten Energiesystems helfen, das Stromnetz auf die volatile Energieerzeugung aus nachhaltigen Energiequellen vorzubereiten und somit eine nachhaltige Entwicklung fördern. Für die Umsetzung solcher Lastprognosen gibt es verschiedene Ansätze und Algorithmen. Diese Arbeit fokussierte die Methode der multivariaten Zeitreihenanalyse mittels LSTM und vergleicht diese mit der univariaten Zeitreihenanalyse mittels LSTM sowie der univariaten Zeitreihenanalyse mittels Random Forest. Anhand realer Daten eines Energieversorgungsunternehmens wurde untersucht, ob die Zugabe von mehreren korrelierenden Variablen mit der Kombination des neuronalen Ansatzes eine Lastprognose verbessert. Um sich dem Thema der Prognose im Energiesektor zu nähern, wurden zu Beginn zwei Projekte aus dem Energiebereich vorgestellt, die in ihren Lösungen sowohl LSTMs als auch den Random Forest Algorithmus verwenden. Um eine Lastprognose zu realisieren, musste zunächst das Ziel dieser Arbeit eingegrenzt werden. Als Grundlage dienten dazu die vorgestellten Projekte und Algorithmen. Die Erstellung der Lastprognose umfasste eine umfangreiche Datenanalyse. Dabei gaben die untersuchten Daten Aufschluss über die Korrelationen zwischen den Kalendertagen, den Zeitangaben und den meteorologischen Daten. Zudem wird ersichtlich, dass sich der Stromverbrauch besonders mit den meteorologischen Variablen stark in Abhängigkeit verhält, da der verwendete Datensatz Objekte mit Photovoltaikanlagen mit einschließt. Des Weiteren zeigen die analysierten Daten wiederkehrende Muster auf, wodurch sich der Verbrauch an den Feiertagen und den Wochenenden klar von den Wochentagen unterscheiden lässt. Aus den gewonnenen Resultaten wurden schließlich die Prognosen mittels Random Forest und LSTM durchgeführt. Da LSTMs in diesem Einsatzgebiet häufig verwendet werden und üblicherweise gute Resultate erzielen, wurde erwartet, dass dieser Ansatz auch in der vorliegenden Arbeit übereinstimmende Erfolge vorweist. Jedoch ist in Kapitel 5.4. „Umsetzung “ zu beobachten, dass die vorhergesagten Werte aller verwendeten Modelle um ca. 1-2% von den echten Werten entfernt liegen. Bei genauer Betrachtung unterscheiden sich die Modelle nur um eine Prognosegenauigkeit von ca. 0,5%. Obwohl die Vorhersage des multivariaten Modells die beste Prognosegenauigkeit erreicht, hat die Zugabe von weiteren Variablen und die umfangreiche Auseinandersetzung mit dem Hyperparameter-Tuning die Genauigkeit nur um ein paar Prozent verbessert. Ein Experiment legte dar, dass der Stromverbrauch bereits mit der Berechnung von Mittelwerten verhältnismäßig einfach prognostizierbar ist und die Kurvenverläufe

einer regulären Kalenderwoche nachgezeichnet werden können. Dadurch ist ersichtlich, dass keine höhere Prognosegenauigkeit durch ein verbesserungswürdiges Modell gelöst werden kann, sondern die vorhandenen Daten nicht für die praxisorientierte Verwendung dieses Ansatzes ausreichen. Angesichts dieser Erkenntnisse konnten neue Fragestellungen formuliert werden. Beispielweise gilt es zu klären, ob fluktuierende und folglich schwervorhersehbare Systeme für weiterführende Untersuchungen spannender wären. Ein Fallbeispiel wäre die Anfangszeit der Corona-Pandemie oder die aktuell extremen Wetterschwankungen auf Grund des anthropogenen Klimawandels. Auf diese Weise wäre es möglich, die ursprünglich zu erwarteten Unterschiede zwischen den Modellen deutlicher zu erkennen. Das multivariate Modell erreichte keine signifikant höhere Prognosegenauigkeit als die anderen Modelle trotz der Zugabe von den ausgewählten Variablen, besonders der Sonneneinstrahlung. Dabei wäre der zu erwartende Effekt aufgrund der Photovoltaikanlagen plausibel gewesen. Demnach ist davon auszugehen, dass nicht mehr Daten, sondern eine Menge verschiedener Datensätze die Prognose des Modells verbessern könnte. Außerdem könnten bessere Aussagen bezüglich der Leistungsfähigkeit und den Schwächen gemacht werden.

Letztendlich konnte in der vorliegenden Arbeit nicht in Erfahrung gebracht werden, wozu die multivariate Zeitreihenanalyse mittels LSTM im Stande wäre. Ein weiteres Experiment erwies jedoch, dass bereits bei einem minimal fluktuierenden Datensatz der neuronale Ansatz bessere Ergebnisse erzielte. Dafür wurden einzelne Profile aus dem vorhandenen Datensatz entnommen und prognostiziert. Eine erhöhte Prognosegenauigkeit bis zu 3% konnte schließlich bei den neuronalen Ansätzen festgestellt werden. Die in dieser Arbeit verwendete Literatur als auch die vorgestellten Projekte belegen ebenso, dass LSTMs zu weitaus mehr im Stande sind, jedoch wesentlich schwieriger nachzuvollziehen sind. Der Random Forest Algorithmus hingegen ist auf Grund der geringen Empfindlichkeit gegenüber den Parameterwerten einfacher zu verstehen und einzustellen. Im Vergleich dazu werden LSTMs häufiger im Energiesektor verwendet und führen zu genaueren Prognosen.

Zusammenfassend beweisen die erarbeiteten Ergebnisse, dass eine eindeutige Korrelation zwischen dem Stromverbrauch, den Kalendertagen, den Zeitangaben und den meteorologischen Daten besteht. Folglich wäre es für eine Lastprognose vorteilhaft, die genannten Variablen mit einzubeziehen sowie das in dieser Arbeit entwickelte Modell weiter zu verwenden und anzupassen. Im weiteren Verlauf könnte auf diese Weise die Lastprognose als Teil eines Netzmanagementsystems fungieren, um Überlastungsprobleme zu lösen oder um das

Energiesystem flexibler und nachhaltiger zu gestalten. Daher wäre es sinnvoll, sowohl die gesammelten Erkenntnisse als auch das entwickelte Modell mit einer verbesserten Datenlage zu testen. Letztendlich bietet die vorliegende Arbeit eine Menge an Diskussionsansätzen und Möglichkeiten für eine weitere Verwendung. Exemplarisch hierfür wäre die Organisation, welche die Daten zur Verfügung gestellt hat, um das Modell mit weiteren realen Daten zu testen. Gleichermaßen wäre eine fortsetzende, wissenschaftliche Arbeit sinnvoll, welche die erarbeiteten Fortschritte weiter untersucht.

7 Literaturverzeichnis

AGFW. AGFW Der Energieeffizienzverband für Wärme, Kälte und KWK e.V. *Digitalisierung in der Fernwärme.* [Online] [Zitat vom: 28. September 2022.] https://www.agfw.de/digitalisierung.

Bedendo, Andre, Reimbold, Manuel und Sausen, Airam. 2014. Evaluation of Model ARX For Elastic Masses MEMS Using the Indexes RMSE, AIC, and BIC. *Journal of Control, Automation and Electrical Systems.* 2014, 25.

Biau, Gérard und Scornet, Erwan. 2016. A random forest guided tour. *TEST.* 25, 2016.

Bouktif, Salah, et al. 2018. Optimal Deep Learning LSTM Model for Electric Load Forecasting using Feature Selection and Genetic Algorithm: Comparison with Machine Learning Approaches. *Energies.* 2018.

Breiman, Leo. 2001. Random Forests. *Machine Learning.* 45, 2001.

Buxmann, Peter und Schmidt, Holger. Grundlagen der Künstlichen Intelligenz und des Maschinellen Lernens. *Künstliche Intelligenz.*

Chatterjee, Ayan, Gerdes, Martin W. und Martine, Santiago G. 2020. Statistical Explorations and Univariate Timeseries Analysis on COVID-19 Datasets to Understand the Trend of Disease Spreading and Death. *Sensors.* 2020.

Chatterjee, Ayan, Gerdes, Martin W. und Martinez, Santiago G. 2020. Statistical Explorations and Univariate Timeseries Analysis on COVID-19 Datasets to Understand the Trend of Disease Spreading and Deat. *Sensors.* 2020.

Congestion Management in distribution grid networks through active power control of flexible distributed energy resources. **Ciavarella, R., Di Somma, M. und Graditi, G.: Valenti, M. 2019.** Milan, Italy : IEEE, 2019. 978-1-5386-4723-3.

Electricity load forecasting for Urban area using weather forecast information. **Dehalwar, Vasudev, et al. 2016.** Shanghai : s.n., 2016. 978-1-5090-3068-2.

Faber, Till, Finkenrath, Matthias und Gross, Johannes. 2018. Innovative Lastprognosen mit »Deep Learning«-Methoden. *EuroHeat&Power.* 2018.

FB Medien. 2014. Prüfungsordnungen. [Online] 2014. [Zitat vom: 09. 10 2014.] http://medien.fh-duesseldorf.de/studienbuero/pruefungsordnung.aspx.

FLECH Services to Solve Grid Congestion. **Masood, Arsalan, et al. 2018.** s.l. : IEEE, 2018. 978-1-5386-8550-1.

Genethliou, Dora und Feinberg, Eugene A. 2005. Load Forecasting. [Buchverf.] Joe H. Chow, Felix F. Wu und James Momoh. *Applied Mathematics for Restructured Electric Power Systems.* s.l. : Springer, 2005.

Grid Management System to solve local Congestion. **Steegh, Robert, Van Cuijk, Ton und Pourashar Khomami, Hadis. 2019.** Madrid : s.n., 2019.

Hochreiter, Sepp und Schmidhuber, Jürgen. 1997. Long Short-Term Memory. *Neural Computation* . 1997.

HS-Kempten. *deepDHC - Deep Learning for District Heating and Cooling: Entwicklung modernster maschineller Lernverfahren für die hochgenaue Fernwärmelastprognose.* [Online] [Zitat vom: 28. September 2022.] https://forschung.hs-kempten.de/forschungsprojekt/55-deepdhc?forschungsschwerpunkt_id=156.

IBM. Übersicht über Zeitreihendiagramme. *IBM.* [Online] [Zitat vom: 11. 10 2022.] https://www.ibm.com/docs/de/qradar-on-cloud?topic=management-time-series-chart-overview.

Jiao, Runhai, et al. 2018. Short-Term Non-Residential Load Forecasting Based on Multiple Sequences LSTM Recurrent Neural Network. *IEEE Access* . 2018, 6.
Kahraman, Aysegul, et al. Comparison of the Effect of Regularization Techniques and Lookback Window Length on Deep Learning Models in Short Term Load Forecasting. [Buchverf.] Yusheng Xue, Yuping Zheng und Damir Novosel. *Proceedings of 2021 International Top-Level Forum on Engineering Science and Technology Development Strategy.* s.l. : Springer.
Keras. Keras. [Online] [Zitat vom: 11. 10 2022.] https://keras.io/.
Kirchgässner, Gebhard und Wolters, Jürgen. 2005. *Einführung in die moderne Zeitreihenanalyse.* s.l. : Vahlen , 2005. 3800632683.
Kong, Weicong, et al. 2019. Short-Term Residential Load Forecasting Based on LSTM Recurrent Neural Network. *IEEE Transactions on Smart Grid.* 10, 2019.
Kwon, Bo-Sung, Park, Rae-Jun und Song, Kyung-Bin. 2020. Short-Term Load Forecasting Based on Deep Neural Networks Using LSTM Layer. *Journal of Electrical Engineering & Technology volume* . 15, 2020.
Matzka, Stephan. 2021. *Künstliche Intelligenz in den Ingenieur-wissenschaften.* s.l. : Springer, 2021. 978-3-658-34641-6.
Multivariate Time Series Forecasting with LSTM for Madrid, Spain pollution. **Alhirmizy, Shaheen und Qader, Banaz. 2019.** Iraq : s.n., 2019. 978-1-7281-4037-7.
Muzaffar, Shahzad und Afshari, Afshin. 2019. Short-Term Load Forecasts Using LSTM Networks. *Energy Procedia.* 2019, Bd. 158.
OfficeHolidays. Federal Holidays in New Hampshire in 2019. *Office Holidays.* [Online] [Zitat vom: 2022. 11 10.] https://www.officeholidays.com/countries/usa/new-hampshire/2019.
Pandas. Pandas. [Online] [Zitat vom: 11. 10 2022.] https://pandas.pydata.org/.
Pfenninger, Stefan und Staffell, Iain. Renewables.ninja. [Online] [Zitat vom: 11. 10 2022.] https://www.renewables.ninja/about.
Random forest based ensemble system for short term load forecasting. **Cheng, Ying-Ying, Chan, Patrick P.K und Qiu, Zhi-Wei. 2012.** China : IEE, 2012. 978-1-4673-1487-9.
Random forests model for one day ahead load forecasting. **Lahouar, Ali und Ben Hadj Slama, Jaleleddine. 2015.** Tunisia : s.n., 2015. 978-1-4799-7947-9.
Sagar, B. S. Daya, et al. 2020. *Encyclopedia of Mathematical Geosciences.* s.l. : Springer, 2020.
Savit, Robert und Manuca, Radu. 1996. Stationarity and nonstationarity in time series analysis. *Physica D: Nonlinear Phenomena.* 1996, 99.
Schäfer, Thomas. 2010 . *Statistik 1 - Deskriptive und Explorative Datenanalyse.* s.l. : VS Verlag für Sozialwissenschaften, 2010 . 978-3-531-16939-2.
Scikit-learn. sklearn.ensemble.RandomForestRegressor. [Online] [Zitat vom: 11. 10 2022.] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html.
Short-Term Load Forecasting on MV/LV Transformer Level. **Fonteijn, Rik, et al. 2019.** Madrid : CIRED, 2019.
Short-Term Load Forecasting Using Random Forests. **Dudek, Grzegorz. 2014.** Polen : Springer, 2014. 978-3-319-11310-4.
Son, Jihoo, et al. 2022. Day-Ahead Short-Term Load Forecasting for Holidays Based on Modification of Similar Days' Load Profiles. *IEEE Access.* 2022.
Stackoverflow. 2014. Java Generics. [Online] 2014. [Zitat vom: 09. 10 2014.] http://stackoverflow.com/questions/490091/java-generics.

Statista. Definition Zeitreihenanalyse. *Statista.* [Online] [Zitat vom: 11. 10 2022.] https://de.statista.com/statistik/lexikon/definition/144/zeitreihenanalyse/.
Statsmodels. statsmodels.tsa.stattools.adfuller. *statsmodels.* [Online] [Zitat vom: 11. 10 2022.] https://www.statsmodels.org/dev/generated/statsmodels.tsa.stattools.adfuller.html.
Stoetzer, Matthias-W. 2020. *Regressionsanalyse in der empirischen Wirtschafts und Sozialforschung Band 2.* s.l. : Springer , 2020. 978-3-662-61438-9.
Systemeffizienz bei regenerativer Stromerzeugung. **Brauner, Günther. 2019.** s.l. : Springer , 2019. 978-3-658-24854-3.
Tran, Hieu. 2019. *Survey of Machine Learning and Data Mining Techniques used in Multimedia System.* 2019. 10.13140/RG.2.2.20395.49446/1.
Ushiku, Yoshitaka. 2021. Long Short-Term Memory. [Buchverf.] Katsushi Ikeuchi. *Computer Vision.* s.l. : Springer, 2021.
Vishwas, B V und Patel, Ashish. 2020. *Hands-on Time Series Analysis with Python.* s.l. : Springer, 2020. 978-1-4842-5992-4.
Wennker, Phil. 2020. Machine Learning. *Künstliche Intelligenz in der Praxis.* s.l. : Springer, 2020.