universität wien

MAGISTERARBEIT / MASTER'S THESIS

Titel der Magisterarbeit / Title of the Master's Thesis

„Modelle zur Prognose der Leistung von Photovoltaikanlagen mittels Wettervorhersagen“

verfasst von / submitted by

Maximilian Pfeiffer, BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Magister der Sozial- und Wirtschaftswissenschaften (Mag. rer. soc. oec.)

Wien, 2022 / Vienna 2022

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 066 951
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Magisterstudium Statistik
Betreut von / Supervisor:	Ass.-Prof. Mag. Lukas Steinberger, BA Bakk. PhD

Vorwort

In der vorliegenden Masterarbeit geht es um die Prognose der Leistung von Solaranlagen durch Machine Learning Modelle mit Hilfe von Wetterdaten. Das Thema habe ich insbesondere aufgrund meines Interesses an Künstlicher Intelligenz gewählt. Es ist spannend zu sehen, wie die Verwendung von Daten damit in vielen kleinen Bereichen das Leben digitalisiert und effizienter macht. Von daher war ich froh, dass ich mich in dieser Arbeit weiter damit beschäftigen konnte. Aus Daten Informationen ziehen, verwenden und insbesondere auch die visuelle Aufbereitung der Ergebnisse haben mir großen Spaß gemacht.

Entstanden ist diese Arbeit aufgrund der Anfrage des Unternehmens Nymea Energy an den Betreuer meiner Masterarbeit Prof. Lukas Steinberger, bei dem ich mich an dieser Stelle ganz herzlich bedanken möchte. Vielen Dank für die Unterstützung, die Anregungen und konstruktive Kritik während des Erstellens dieser Arbeit.

Weiterhin vielen Dank an das Unternehmen Nymea Energy [1] für das Bereitstellen der Daten der Photovoltaikanlagen.

Abschließend auch noch ein sehr großes Dankeschön an alle Kommilitonen, Freunde und Familienmitglieder, die mich während der Masterarbeit und während des gesamten Studiums auf vielfältige Weise unterstützt haben.

[1] https://www.nymea.energy/

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Akronyme

CNN Convolutional Neural Network.

Dhi Diffuse Horizontal Irradiance.

Dni Direct Normal Irradiance.

Ebh Direct Beam Horizontal Irradiance.

FNN Feedforward Neural Network.

Ghi Global Horizontal Irradiance.

kNN k-Nearest Neighbour.

kWh Kilowattstunde.

LSTM Long Short-Term Memory.

MAE Mean Absolute Error.

ML Machine Learning.

MSE Mean Squared Error.

nMAE Normalised Mean Absolute Error.

nMBE Normalised Mean Bias Error.

nRMSE Normalised Root Mean Squared Error.

NWP Numerical Weather Prediction.

PV Photovoltaik.

ReLU Rectified Linear Unit.

RF Random Forest.

RMSE Root Mean Squared Error.

RNN Recurrent Neural Network.

SE Squared Error.

SVM Support Vector Machine.

1 Einleitung

1.1 Einführung und bisherige Arbeiten

Netzstabilisierung und Eigenverbrauch-Optimierung sind nur zwei der relevanten Fragestellungen, die im Zuge des stetigen Ausbaus erneuerbarer Energien mehr und mehr an Bedeutung gewinnen (vgl. Ahmed et al., 2020). Auch die Anzahl der Photovoltaikanlagen hat weltweit und insbesondere auch in Österreich deutlich zugenommen. So stieg deren Gesamtleistung zwischen 2005 und 2020 im Schnitt um 34,5 % pro Jahr, wobei das zukünftige Ziel ist, auf einer Million Hausdächer PV-Anlagen installiert zu haben (vgl. BMK, 2021). Die steigende Bedeutung der Solarenergie rückt eine Reihe an Problemen, die von Schwankungen in der Leistung der PV-Anlagen bedingt sind, in den Vordergrund. Dazu gehört einerseits die Gewährleistung von Netzstabilität, aber beispielsweise auch, dass Schwankungen den Betreibern der PV-Anlagen eine effiziente Eigennutzung erschweren. Die Variabilität der Leistung ist einerseits periodisch, bedingt durch den Sonnenstand und die Ausrichtung der Anlage, und damit auch berechenbar. Andererseits ist sie aber zusätzlich durch das Wetter beeinflusst, wie beispielsweise durch Luftfeuchtigkeit und Bewölkung, Bedingungen deren Vorhersage und Auswirkungen deutlich schwerer und ungenauer zu erfassen sind.

Die Prognose von Solarstromproduktion ist ein bereits intensiv erforschtes Gebiet, jedoch unterscheiden sich die zahlreichen Publikationen stark voneinander, was sie nicht immer vergleichbar macht. Beispielsweise geben Ahmed et al. (2020) einen Überblick über eine große Anzahl an Veröffentlichungen, die sich mit Vorhersagen der Leistung von PV-Anlagen beschäftigen. Je nach Anwendung ist der Prognosezeitraum entscheidend. Besonders häufig werden kurzfristige Prognosen betrachtet, sowie 24 Stunden im Voraus getätigte stündliche Vorhersagen (vgl. Ahmed et al., 2020). Bedeutende Unterschiede gibt es weiterhin bei den untersuchten Standorten, ob und welche Wetterdaten verwendet werden, sowie bei der Methodik. Die Autoren beschreiben, dass bei kurzfristigen Vorhersagen die Verwendung von Satellitenbildern zum Erkennen der Wolkenmenge, -dicke und -bewegung recht gute Prognose-Ergebnisse liefert. Bei Vorhersagen von mehr als 4 Stunden kommen jedoch meist nur noch übliche numerisch berechnete Wettervorhersagen (Numerical Weather Prediction) zum Einsatz. Die in der Literatur untersuchten Modelle umfassen zum Beispiel klassische Zeitreihenmodelle wie etwa ARIMA, jedoch wurden in den letzten Jahren meist Machine Learning Modelle verwendet. Zu den erfolgreichsten gehören dabei unter anderem verschiedene Neuronale Netze wie CNN und LSTM (vgl. Ahmed et al., 2020). Auch Gigoni et al. (2018) betrachten verschiedene Machine Learning Modelle (kNN, Random Forest, SVM und FNN) im direkten Vergleich miteinander. Sie modellieren eine 24-Stunden-Vorhersage der stündlichen PV-Leistung anhand von Wetterdaten zweier verschiedener Quellen, wobei diese bereits Vorhersagen zur Sonneneinstrahlung enthalten. Insgesamt werden Daten zu 32 PV-Anlagen betrachtet, es gewinnt ein gewichtetes Ensemble aller anderen Modelle, während im Einzelnen ein Random Forest Modell die besten Ergebnisse liefert. Die Autoren beobachten zudem, dass bei sehr klarem Wetter die Genauigkeit steigt und bei mittlerer Bewölkung etwas sinkt. Bewertet werden die Modelle mit den in solchen Publikationen häufigsten Metriken MAE, RMSE und MBE, bzw. deren nor-

malisierten Werten (vgl. Gigoni et al., 2018). Die Beobachtung, dass an sonnigen Tagen die Prognosen genauer sind, wird von vielen anderen Autoren geteilt, und insbesondere auch, dass die durchschnittliche Genauigkeit in Regionen mit wechselhaftem Wetter, wie z.B. in Österreich, geringer ist, als in sehr sonnigen südlich gelegenen Regionen (vgl. Ahmed et al., 2020).

1.2 Inhalt der Arbeit

Thema dieser Arbeit ist die Anwendung verschiedener Machine Learning Modelle zur Vorhersage der stündlichen Energieproduktion zweier PV-Anlagen, welche beide in Großschönau in Niederösterreich stehen. Die Daten zur Stromproduktion beider Anlagen wurden von dem Unternehmen Nymea Energy [2] bereitgestellt. Diese Arbeit orientiert sich im Aufbau an der möglichen Anwendung, Vorhersagen zu nutzen um den Eigenverbrauch zu optimieren, beispielsweise durch intelligentes Aufladen von Elektroautos mit dem eigenen Solarstrom.

Um die Vorhersagen zu berechnen wird auf zwei Datensätze mit Wetterdaten zurückgegriffen, welche aus verschiedenen Quellen stammen. Für diese Datensätze werden die Modelle getrennt angepasst. Der erste Datensatz stammt vom Online-Dienst OpenWeather (n. d.) und enthält Wetterdaten, der zweite von Solcast (2019), ein Unternehmen, das neben Wetterdaten insbesondere auch Daten zur Sonneneinstrahlung bereitstellt.

Methodisch werden 5 Modelle für jede Anlage und jeden Datensatz betrachtet: Ein einfaches lineares Regressionsmodell als Benchmark-Modell, ein Random Forest, sowie zwei Neuronale Netze (FNN und LSTM). Zusätzlich wird ein Ensemble Modell aus den bisher genannten Modellen untersucht. Ziel ist, die Datensätze auf ihre Eignung zur Vorhersage zu überprüfen, die verschiedenen Machine Learning Modelle zu vergleichen, sowie statistisch darzustellen, wie gut diese Prognosen der Stromproduktion ausfallen. In Abbildung 1 ist der Prozess dieser Arbeit schematisch dargestellt.

Der weitere Aufbau dieser Arbeit ist wie folgt: Kapitel 2 enthält eine genaue Beschreibung der Daten und wie diese aufbereitet werden. Weiterhin folgt eine Korrelationsanalyse zwischen Leistung und einzelner Wetterdaten. In Kapitel 3 befindet sich eine theoretische Einführung der einzelnen Machine Learning Modelle. Kapitel 4 befasst sich mit der Methodik, mit einer genauen Beschreibung der Experimente, der Implementierung in R und einer Beschreibung wie die Ergebnisse evaluiert werden. In Kapitel 5 werden die Ergebnisse dargestellt und besprochen, ein Fazit der gesamten Arbeit folgt schlussendlich in Kapitel 6.

[2] https://www.nymea.energy/

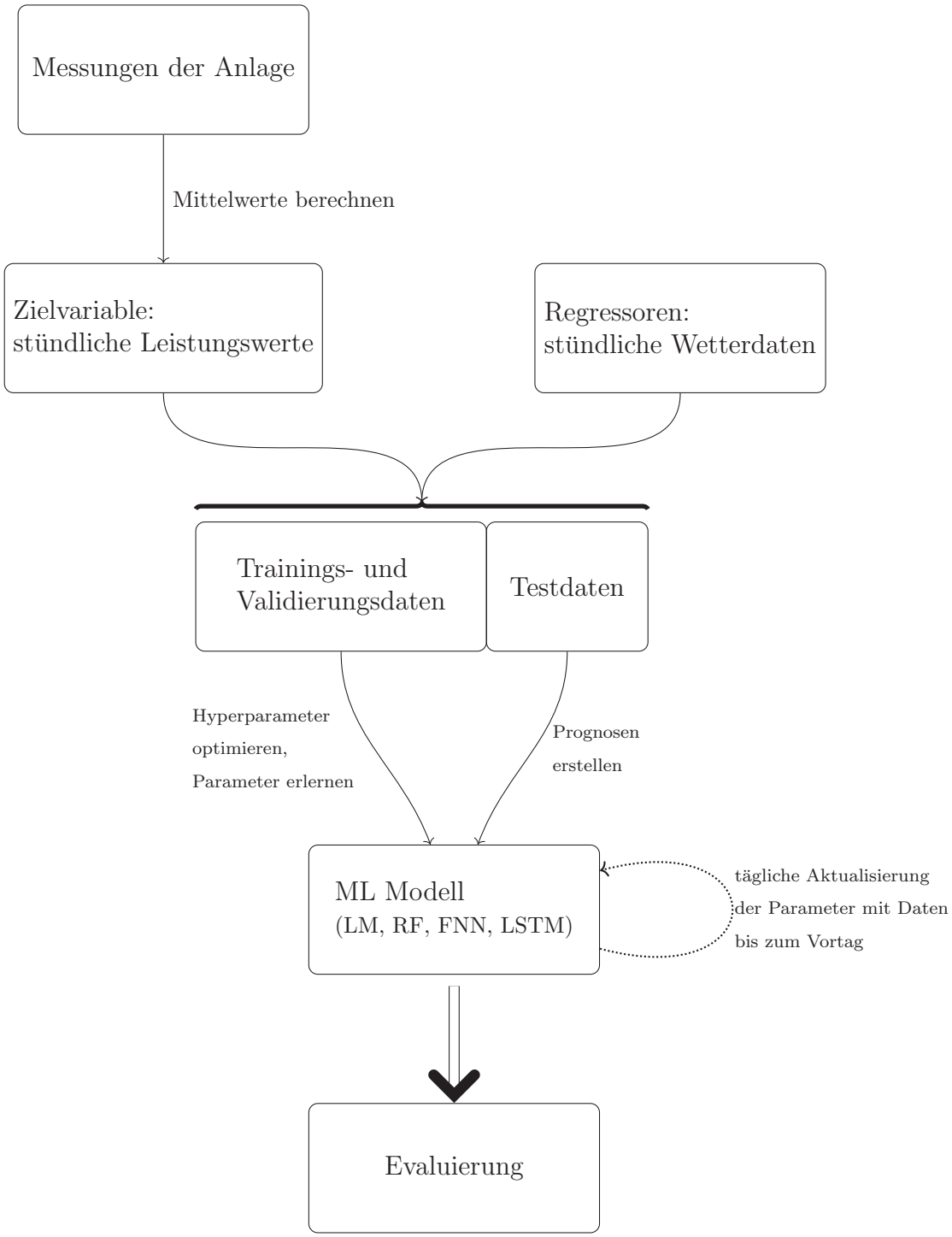

Abbildung 1: Schematischer Überblick über den Ablauf der Experimente

2 Daten

Die Zeitreihen zum Energieertrag stammen von 2 Photovoltaikanlagen, beide befinden sich in Großschönau, einem Ort in Niederösterreich. Um die Leistung der Anlagen zu prognostizieren, werden numerische Wetter- und Sonneneinstrahlungsprognosen genutzt, es wird insgesamt auf 2 Datensätze aus verschiedenen Quellen zurückgegriffen. Der erste Datensatz beinhaltet ausschließlich gewöhnliche Wetterdaten, der zweite zusätzlich spezielle zur Sonneneinstrahlung.

2.1 Daten zum Energieertrag der Anlagen

2.1.1 Rohdaten

Die PV-Anlagen werden im Folgenden mit Anlage K und Anlage S bezeichnet. Der Ertrag der Anlagen ist in kWh angegeben. Die Abstände der Messungen betragen etwa 30 Sekunden und erfolgten ausschließlich bei Tageslicht, weshalb deren Anzahl von Tag zu Tag etwas schwankt. Für beide Anlagen stammen die Messungen aus dem Jahr 2021, jedoch nicht genau an den gleichen Tagen, der Zeitraum ist in Tabelle 1 zu sehen.

2.1.2 Datenaufbereitung

Der Rohdatensatz wird zunächst auf fehlerhafte Daten analysiert und bereinigt. Dazu wird die Anzahl der Messungen, sowie eine grafische Darstellung der Leistungskurve eines jeden einzelnen Tages betrachtet. Dabei lassen sich leicht einige Abweichungen erkennnen, z.B. sind an einigen Tagen zu große zeitliche Lücken zwischen Messungen, an anderen ist statt einer typischen Glocken-Kurve mit Schwankungen, eine teilweise konstante Leistung gemessen worden. Diese Art der Messungen werden als fehlerhaft eingestuft und aus dem Datensatz entfernt. Weiterhin werden doppelte Messwerte entfernt. Einige Leistungsdaten sind negativ, jedoch scheint dies ein Fehler bei der Datenspeicherung zu sein. Eine Korrektur des Vorzeichens reicht aus.

Ziel ist die Vorhersage des stündlichen Ertrags der Anlagen. Daher wird dieser mit den bereinigten Daten der beiden Anlagen über den Mittelwert der Messungen berechnet. Der in den Experimenten verwendete Datensatz hat schlussendlich folgende Struktur.

2.1.3 Finaler Datensatz

Für Anlage S stehen Daten zum Energieertrag von 1.917 Stunden, verteilt auf 122 Tage im Zeitraum März bis Juli 2021, zur Verfügung. Die in diesem Zeitraum maximal gemessene Leistung beträgt 64.502 kWh, der Medianertrag liegt bei 16.909. Für Anlage K existieren 2.660 stündliche Werte, verteilt auf 186 Tage im Zeitraum Februar bis November 2021, die maximal gemessene Leistung beträgt 18.687 kWh, im Median beträgt die Leistung 4.448 kWh.

Die Daten enthalten ausschließlich den Ertrag der bei Tageslicht gemessen wird, eine Prognose der Leistung bei Nacht würde die Genauigkeit verzerren.

In den Experimenten werden beide Anlagen getrennt betrachtet. Die genaue Ausrichtung und Kapazität der Anlagen ist nicht bekannt, jedoch lässt sich an dem Verlauf der Leistungskurve erkennen, dass diese verschieden sind (vgl. Abbildung 2 auf Seite 7).

	März	April	Mai	Juni	Juli	Aug	Sept	Okt	Nov
Anl. S	8	29	31	30	24				
Anl. K	7	9	14	30	31	31	30	31	3

Tabelle 1: Anzahl der Tage mit Messungen in den einzelnen Monaten 2021

2.2 Numerische Wetterdaten

Als Prädiktorvariablen werden 2 Datensätze aus unterschiedlichen Quellen betrachtet. Der erste Datensatz enthält numerische Wetterdaten. Zusätzlich verwendet werden sonnengeometrische Daten, sowie aus diesen berechnete Kennzahlen zur Sonnenenergie. Der zweite Datensatz beinhaltet sowohl numerische Wetterdaten, als auch bereits unter dessen Einbeziehung errechnete Werte zur Sonneneneinstrahlung.

Beide Datensätze enthalten Nowcasting Daten. Das heißt, es handelt sich nicht um zuvor getätigte Prognosen, aber auch nicht um genaue Messungen des Wetters direkt an der Anlage. Stattdessen sind es Vorhersagen zum aktuellen Zeitpunkt bzw. berechnete Werte, unter anderem aus Satellitenbildern oder Messstationen im Umland. Eine genaue Beschreibung folgt im Rest des Kapitels.

2.2.1 Datensatz 1

Dieser setzt sich aus 2 Teilen zusammen, aus Wetterprognosen, sowie aus Daten zur Clear Sky Irradiance, also der Sonneneinstrahlung, welche bei klarem Himmel zu erwarten wäre:

1. Wetterdaten von OpenWeather.
   Diese beinhalten numerische Prognosen zum lokalen Wetter. In Tab. 2 ist eine Auflistung der Variablen zu sehen, die in dieser Arbeit Verwendung finden. Über die Berechnung der Daten macht der Anbieter keine genauen Angaben. Laut der Website erfolgt sie jedoch anhand anderer bekannter NWP-Modelle, sowie von Satellitenbildern, Bodenmessstationen und Wetterradarkarten. Über die Genauigkeit gibt der Anbieter nur wenige Daten, der MAE der Temperatur-Prognose soll jedoch bei rund 0,5 Grad liegen. Für andere Variablen wie etwa der Bewölkung ist keine Angabe zu finden (vgl. OpenWeather, n. d.).

   Tabelle 2 listet den verwendeten Teil der Variablen des Datensatzes, zusammen mit ihrer Pearson-Korrelation zum Energieertrag der letzten Stunde, auf.

Variable	Name und Einheit	Korrelation zum Energieertrag	
		Anlage S	Anlage K
humidity	Luftfeuchtigkeit [%]	0,526	0,571
temp	Temperatur [°C]	0,401	0,448
wind_deg	Windrichtung [Grad]	0,187	0,183
clouds_all	Bewölkung [%]	0,120	0,140
wind_speed	Windgeschwindigkeit [m/s]	0,056	0,100
rain_1h	Regenmenge letzte Stunde [mm]	0,112	0,084
snow_1h	Schneemenge letzte Stunde [mm]	0,056	0,055
pressure	Luftdruck [hPa]	0,058	0,018
weather_id	Wetter ID zur Beschreibung	-	-

Tabelle 2: Variablen der Wettervorhersage und ihre Korrelation zur Stromproduktion (Datensatz 1)

2. Clear Sky Irradiance
   Die Stromproduktion einer Solaranlage hängt größtenteils von der Sonnenenergie die auf sie trifft, ab. Das Wetter beeinflusst diese Sonneneinstrahlung, beispielsweise durch Bewölkung oder durch Aerosole in der Atmosphäre. Der bisherige Datensatz enthält ausschließlich Wetterbedingungen, jedoch keine Daten zur Sonneneinstrahlung. In der Literatur existieren eine Vielzahl an mathematischen Modellen, welche die Sonneneinstrahlung auf eine Fläche bei klarem Himmel berechnen. Diese werden Clear Sky Modelle genannt. Untereinander unterscheiden sie sich in ihrer Genauigkeit und Komplexität, d.h. insbesondere welche Parameter berücksichtigt werden. Für einfachere Modelle reichen meist sonnengeometrische Angaben, so etwa der Höhenwinkel und das Azimuth der Sonne, die sich für gegebene Koordinaten berechnen lassen. Bei komplexeren Modellen spielen auch z.B. die Höhe über dem Meeresspiegel, sowie atmosphärische Bedingungen, wie etwa Aerosole, Ozon und Luftfeuchtigkeit, eine Rolle (vgl. Sun et al., 2019).

   In dieser Arbeit werden zwei bekanntere Modelle verwendet, zum Einen das Modell ESRA (Digital European Solar Radiation Atlas) (vgl. Rigollier et al., 2000), und als Zweites das Modell von Ineichen und Perez (2002). Parameter dieser zwei Modelle sind jeweils der berechenbare Sonnenstand (in Grad), der Längen- und Breitengrad des Standorts der Anlagen, das Datum, die Höhe über dem Meeresspiegel, sowie der Linke Turbidity Koeffizient. Letzterer beschreibt die Streuung und Absorption der Sonnenenergie in der Atmosphäre durch Aerosole und Wasserpartikel. Da die verfügbaren Wetterdaten keine Aerosol-Prognosen enthalten, werden für diese Arbeit monatliche Durchschnittswerte des Koeffizienten verwendet, diese werden mit Hilfe der geographischen Position online von SoDa (2010) extrahiert (vgl. Yang, 2018). Ausgabe beider Modelle ist die Global Horizontal Irradiance (Ghi). Diese Kennzahl bezeichnet die gesamte Sonnenenergie, welche auf eine horizontale Oberfläche trifft. Sie ist auch die Summe aus Dni und Dhi. Diese beiden Variablen werden aber nur mit dem ESRA-Modell berechnet. Direct Normal Irradiance (Dni) bezeichnet die Sonnenergie, welche auf eine senkrecht zur Sonne stehende Oberfläche trifft. Sie ist die Strahlung, die es ohne Verluste

durch Streuung und Absorption in der Atmosphäre (durch Ozon, Wasserdampf und Aerosolen), gäbe. Diffuse Horizontal Irradiance (Dhi) ist die Sonnenenergie, welche auf eine horizontale Oberfläche trifft, aber nicht direkt von der Sonne stammt, sondern durch Streuung und Absorption. Alle Strahlungsdaten werden in der Einheit Watt pro Quadratmeter (W/m^2) ausgegeben.

Variable	Name und Einheit	Korrelation zum Energieertrag	
		Anlage S	Anlage K
ghi_ip	Ghi (Ineichen Perez Modell) $[W/m^2]$	0,760	0,779
ghi_esra	Ghi (ESRA Modell) $[W/m^2]$	0,760	0,763
dhi_esra	Dhi (ESRA Modell) $[W/m^2]$	0,724	0,706
dni_esra	Dni (ESRA Modell) $[W/m^2]$	0,700	0,669
sun_altitude	Sonnenstand $[Grad]$	0,781	0,630
sun_azimuth	Azimuth	0,136	0,146

Tabelle 3: Variablen der Sonnengeometrie und Clear Sky Irradiance, sowie ihre Korrelation zur Stromproduktion (Datensatz 1)

In Abbildung 2 sind die alle 30 Sekunden erfolgten Messungen der beiden Anlagen beispielhaft an zwei sonnigen Tagen abgebildet. Zusätzlich ist die mit den Clear Sky Modellen berechnete Sonnenenergie in Form der 3 Variablen Ghi, Dni und Dhi abgebildet. Es ist klar zu sehen, dass der Verlauf der Kurven sehr ähnlich ist, jedoch ist bei Anlage K eine Schiefe nach rechts zu erkennen, und bei Anlage S eher eine nach links. Daraus ist zu schließen, dass die PV-Anlagen verschiedene Ausrichtungen haben.

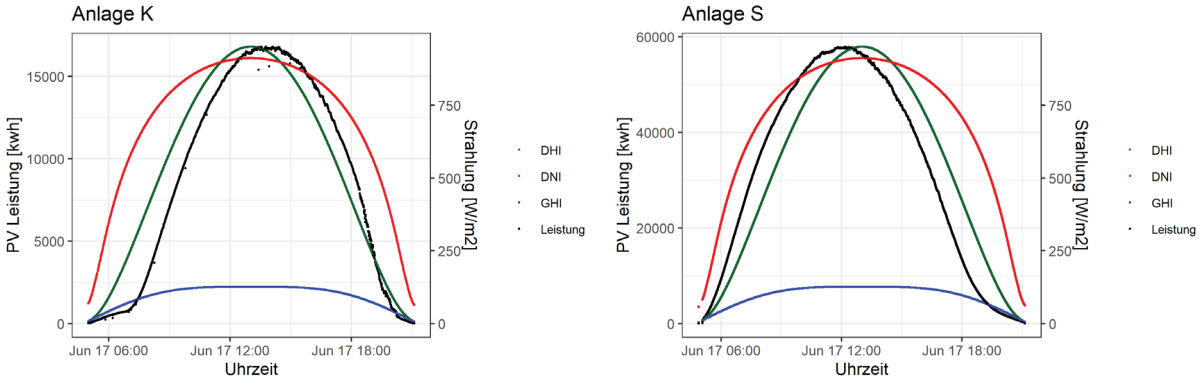

Abbildung 2: Gemessene Stromproduktion an einem sonnigen Tag und berechnete Clear Sky Irradiance Werte (ESRA-Modell)

Abgesehen von *rain_1h* und *snow_1h* geben die Variablen in Tab. 2 die Werte jeweils zum Ende der Stunde an, daher werden als Input-Variablen in Modellen mit Datensatz 1 zusätzlich die Werte der vorherigen Stunde verwendet. Weiterhin wird das Datum und die Uhrzeit der Messungen als Feature gesetzt. Mit den Features aus Tabelle 3

hat dieser Datensatz somit insgesamt 22 Regressoren, die als Eingabe in die Modelle dienen.

2.2.2 Datensatz 2

Dieser Datensatz wurde von Solcast bereitgestellt, einem privaten Unternehmen, welches sich auf Solarenergie-Vorhersagen spezialisiert hat, und auch Daten für wissenschaftliche Arbeiten zur Verfügung stellt. Eine Beschreibung und Erklärung der Variablen des Datensatzes befindet sich auch auf der Website des Anbieters (vgl. Solcast, 2019), und als Veröffentlichung von Bright (2019). In diesem wird auch die Methodik des Unternehmens kurz beschrieben wird und es erfolgt eine Evaluierung der Genauigkeit.

Der Datensatz enthält eine ganze Reihe an Features, insbesondere Werte zur Sonnenenergie: Ghi, Dhi, Dni, Ebh, GtiFixedTilt und GtiTracking. Anders als in Datensatz 1 geben diese Variablen aber nicht die Werte bei klarem Himmel an, sondern die Werte die aufgrund des bestehenden Wetters zu erwarten sind. GtiFixedTilt und GtiTracking bezeichnen hier die Global Tilted Irradiance, d.h. die gesamte Sonnenenergie, welche auf eine (fixierte oder sonnenverfolgende) Fläche mit Neigung trifft (vgl. Solcast, 2019; Bright, 2019). Diese Neigung kann angegeben werden. Da wir aber nichts über die Ausrichtung der Solarmodule beider Anlagen wissen, wurden Standardwerten die eine optimale Ausrichtung beschreiben würden, angegeben. Diese Standardwerte beschreiben Module mit 40.2 Grad horizontaler Neigung, die in Richtung Süden ausgerichtet sind. Die weiteren Variablen umfassen numerische Prognosen zum Wetter. Diese sind Cloud Opacity (Bewölkung, in %), Air temp (Temperatur, 2m über dem Boden, in °C), Dewpoint (Taupunkt, 2m über dem Boden, in °C), Relative humidity (relative Luftfeuchtigkeit, 2m über dem Boden, in %), Sfc pressure (Luftdruck, in hPa), Wind speed (Windgeschwindigkeit in m/s), Wind direction (Windrichtung, in Grad),
Precip Water (Ausfällbares Niederschlagswasser, in kg/m2), Snow depth (Schneehöhe, in cm), Albedo (durchschnittliche Spiegelung der Oberfläche von Licht, Werte zwischen 0 und 1) (vgl. Solcast, 2019; Bright, 2019).

Variable	Korrelation zum Energieertrag	
	Anlage S	Anlage K
Ghi	0,934	0,929
GtiFixedTilt	0,914	0,913
GtiTracking	0,852	0,829
Zenith	0,761	0,767
Ebh	0,795	0,764
RelativeHumidity	0,609	0,735
Dni	0,695	0,641
Dhi	0,475	0,558
AirTemp	0,412	0,509
CloudOpacity	0,484	0,445
WindSpeed10m	0,068	0,189
WindDirection10m	0,211	0,174
Azimuth	0,147	0,140
AlbedoDaily	0,022	0,125
SurfacePressure	0,116	0,070
SnowWater	0,109	0,063
DewpointTemp	0,027	0,027
PrecipitableWater	0,040	0,002

Tabelle 4: Variablen und ihre Korrelation zur Stromproduktion (Datensatz 2)

Solcast stellt die Daten in einer Auflösung von bis zu 5 min zur Verfügung, für diese Arbeit wird eine Auflösung von 60 min betrachtet, da die stündliche Leistung prognostiziert wird. Die geografische Auflösung in der Daten zur Sonnenenergie angeboten werden beträgt 2km (vgl. Solcast, 2019).

In der Arbeit von Bright (2019) befindet sich eine detaillierte Analyse der Genauigkeit der Werte. Für unsere Klimazone wird beispielsweise ein nRMSE von durchschnittlich 15,1% angegeben, wobei an bewölkten Tagen der Schnitt bei 21,3% liegt, und an rein sonnigen bei 2,7%.

3 Theorie der Modelle

Dieses Kapitel behandelt eine kurze Einführung in die mathematische und algorithmische Theorie der verwendeten Modelle. Dabei gehen wir in erster Linie darauf ein, was zum Verständnis der Methodik in dieser Arbeit notwendig ist. Weitergehende Details werden ausgespart, für diese ist auf die jeweils verwendete Literatur verwiesen.

Angenommen es liegt eine Datenmenge vor:

$$\begin{aligned}&(x_t, y_t),\ t = 1, .., N,\\ &y_t \in \mathbb{R},\\ &x_t \in \mathbb{R}^d,\ x_t = (x_{t,1}, ..., x_{t,d})\end{aligned}$$

d ist die Dimension des Feature Raums, y_t sind die Werte der Zielvariable.

3.1 Lineare Regression

Als einfaches Benchmark-Modell wird eine lineare Regression durchgeführt:

$$\hat{y}_t = \hat{\beta}_0 + \sum_{j=1}^{d} x_{t,j}\ \hat{\beta}_j$$

mit den Parametern $\hat{\beta}_j,\ j = 0, ..., d$ und Schätzwerten $\hat{y}_t$.

3.2 Random Forest

Random Forest bezeichnet ein Verfahren das von Breimann (2001) entwickelt wurde. Es ist eine spezielle Form des Bagging (Bootstrap Aggregating), angewandt auf Decision Trees, oder in diesem Fall Regression Trees. Die folgende Einführung zu Regression Trees und Random Forest orientiert sich an Hastie et al. (2009), Kapitel 8.2 und 15.

Angenommen wir haben eine Datenmenge (wie oben beschrieben). Ein Regression Tree $T = T(x)$ ist eine Funktion, die die Menge der Regressoren in L verschiedene Mengen $B_1, ..., B_L \in \mathbb{R}^d$ aufteilt, und allen Datenpunkten, die in einer gemeinsamen Menge liegen, einen gemeinsamen Wert als Prognose zuweist.
Soll beispielsweise der MSE zwischen Prognosewert und wahrem Wert minimiert werden, so wird idealerweise der Mittelwert c_l aller Zielvariablen aus dieser Menge als Prognosewert gewählt. Somit hat der Schätzer folgende Form (vgl. Hastie et al., 2009 S. 305):

$$T(x_t) = \sum_{l=1}^{L} c_l \cdot \mathbb{I}\{x_t \in B_l\}$$

Die Struktur des Modells kann anhand eines Baumes, wie in Abb 3, dargestellt werden.

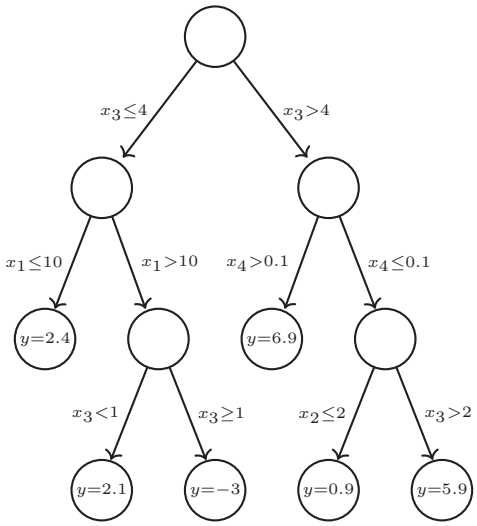

Abbildung 3: Ein binärer Regression Tree mit 6 Blättern

Algorithmisch erfolgt die Unterteilung in Mengen sukzessiv, d.h. die Datenmenge wird Schritt für Schritt in kleinere Mengen gesplittet, bis L Mengen am Ende der Entscheidungskette entstehen, diese werden Blätter genannt. Wird bei jedem Split eine Menge nur in zwei neue aufgeteilt, spricht man von einem binären Baum. Meist wird hier nur ein Feature der Daten als Kriterium eines Splits herangezogen.

Soll ein neuer Datenpunkt vorhergesagt werden, durchläuft er die Knoten bis er in einem Blatt landet und bekommt den gleichen Wert wie alle anderen Punkte in dieser Menge zugeordnet.

Da die mögliche Anzahl an Bäumen einer Datenmenge meist riesig ist, wird die Struktur eines Baumes in der Regel greedy erlernt, dafür gibt es eine Vielzahl an Algorithmen. Greedy bedeutet, dass beim Erlernen des Modells die Parameter, also die Splits, eines Baumes nicht gleichzeitig, sondern nacheinander bestimmt werden. Angenommen die Trainingsdaten sind bereits in $\tilde{L}$ Mengen unterteilt und haben dort den Mittelwert $\hat{y}_t$ als Prognose zugewiesen bekommen, dann ist der Verlust, auch Impurity genannt, gegeben durch:

$$MSE(\hat{y}, y) = \frac{1}{n} \sum_{t=1}^{N} (\hat{y}_t - y_t)^2$$

Ein Algorithmus wählt nun im nächsten Schritt für jede der Mengen den Split aus, der die Verlustfunktion am meisten senkt. Dies erfolgt, bis ein Stopp-Kriterium erreicht ist, oder keine weitere Verbesserung möglich ist.

Ein Random Forest (für Regression) ist ein Modell das eine spezielle Form des Bagging (Bootstrap Aggregating) auf Regression Trees anwendet. Regression Trees sind Modelle mit meist kleinem Bias, aber hoher Varianz. Bagging zielt darauf ab diese zu verringern, denn statt einem Modell wird eine große Anzahl an Modellen erlernt. Jedes dieser gibt eine eigene Prognose ab, die Ausgabe des gesamten Modells setzt sich als Mittelwert aller einzelnen zusammen. Damit sich die einzelnen Modelle unterschei-

den, wird jedes auf einem eigenen Bootstrap Sample erlernt. Ein Bootstrap Sample ist eine Datenmenge der gleichen Größe wie der gesamte Trainingsdatensatz, welche durch zufälliges Ziehen mit Zurücklegen bestimmt wird. Als Konsequenz unterliegen diese Samples der gleichen Verteilung. Zudem hat die gemittelte Prognose den gleichen Erwartungswert (und Bias) wie die der einzelnen Modelle.

Dagegen ändert sich die Varianz im Ensemble Modell: Haben B Zufallsvariablen die Varianz σ^2 und paarweise Korrelation ρ, so beträgt die Varianz des Mittelwerts (vgl. Hastie et al., 2009, S. 588):

$$\tilde{\sigma} = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

Ist die Korrelation $\rho \neq 0$, so ist damit auch bei steigender Anzahl der Modelle die gemeinsame Varianz $\tilde{\sigma}$ nach unten beschränkt.
Um die Korrelation gering zu halten, weist die Methode Random Forest eine Besonderheit gegenüber des üblichen Baggings auf. Bei jedem einzelnen Regression Tree wird bei Erlernen eines neuen Splits nicht der beste unter allen möglichen gewählt. Stattdessen wird zufällig eine zuvor festgelegte Anzahl s an Features gezogen, sodass der Split nur anhand eines dieser Features bestimmt wird. Die Anzahl s kann zwischen 1 und d (Dimension des Feature Raums) gewählt werden, in der Regression ist ein häufiger Standardwert $d/3$. Genau sollte er aber abhängig von den Trainigsdaten bestimmt werden. Eine sehr klein gewählte Zahl s reduziert zwar die Korrelation der Bäume, könnte aber auch zur Überbewertung nicht relevanter Features führen (vgl. Hastie et al., 2009).

Vor dem Trainieren des Modells werden folgende einschränkende Hyperparameter bestimmt: Die Anzahl s an Features die bei jedem Split gezogen werden. Zudem die Anzahl an Bäumen innerhalb des Random Forest, wobei eine zu große Anzahl, abgesehen vom Rechenaufwand kein Nachteil ist. Weiterhin kann die minimale Anzahl an Daten innerhalb eines Blattes festgesetzt werden, sowie die maximale Tiefe der Bäume, um zu verhindern dass einzelne Regression Trees die Daten überanpassen.

3.3 Künstliche Neuronale Netze

Unter Machine Learning Modellen haben Künstliche Neuronale Netz in den letzten Jahren stark an Bedeutung gewonnen. Auch bei der Vorhersage von Solarenergie sind sie in den letzten Jahren zum Standard geworden. Im direkten Vergleich schlagen sie meist andere ML Modelle (vgl. Ahmed et al., 2020). In dieser Arbeit werden 2 Arten von Neuronalen Netzen verwendet: Feedforward Neural Network (FNN) und Long Short-Term Memory (LSTM) Neural Networks. Die folgende theoretische Einführung orientiert sich an Goodfellow et al. (2016), Kapitel 6,7 und 10.

3.3.1 Feedforward Neural Networks

FNN's (Feedforward Neural Networks), auch MLP's (Multilayer Perceptrons) genannt, werden oft als die Basismodelle der künstlichen Neuronalen Netze gesehen. Sie gehören in den Bereich des Supervised Learning und dienen im Prinzip der Approximation von Funktionen. Grundsätzlich ist ein FNN eine spezielle nichtlineare Abbildung, welche

durch die Aneinanderkettung affiner Abbildungen und nichtlinearer Funktionen entsteht. Das Netz besteht aus Schichten, auch Layer genannt. Ein Layer ist die Verkettung einer affinen Transformation und einer nichtlinearen Aktivierungsfunktion, welche komponentenweise angewandt wird. Ein FNN mit m Layern kann demnach wie folgt dargestellt werden:

$$x = x_0 \xrightarrow{f_1} f_1(x_0) = x_1 \xrightarrow{f_2} f_2(x_1) = x_2 \xrightarrow{f_3} .... \xrightarrow{f_m} \hat{y} = f_m(x_{m-1})$$

mit

$$f_j(x_{j-1}) = \sigma_j(W_j x_{j-1} + b_j)$$
$$x_j \in \mathbb{R}^{d_j},\ j = 0, ..., m$$
$$W_j \in \mathbb{R}^{d_j \times d_{j-1}},\ b_j \in \mathbb{R}^{d_j}$$

Die Einträge von $x = x_0$ werden manchmal auch Input Layer genannt. f_m bildet den Output Layer, die übrigen Funktionen f_j werden Hidden Layer genannt. Die Einträge der Matrizen $W_j = W_j(\theta)$ und der Bias Vektoren $b_j = b_j(\theta)$ bestehen aus den Parametern θ, deren optimale Werte anhand der Trainingsdaten erlernt werden. σ_j ist eine (zuvor gewählte) Aktivierungsfunktion in $\mathbb{R}$. Sie wird komponentenweise angewandt und sollte in der Regel nichtlinear sein. In der Literatur existiert eine Vielzahl an möglichen Aktivierungsfunktionen, eine der häufig genutzten ist die ReLu (Rectified Linear Unit) Funktion. Sie ist stückweise linear und erhält damit viele positive Eigenschaften linearer Modelle, wie deren generalisierende Eigenschaften oder etwa eine leichtere Optimierung mittels Gradientenverfahren (vgl. Goodfellow et al., 2016, S.170).

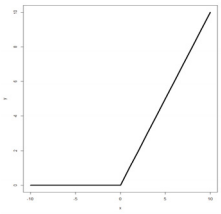

Abbildung 4: ReLU- Aktivierungsfunktion

$$Relu(x) = max(0, x)$$

Die letzte Schicht eines FNN wird der Struktur der Zielvariable angepasst, d.h. sie hat eine entsprechende Breite, sowie eine der Verteilung entsprechende Aktivierungsfunktion. In der Regression bietet sich hier eine einfache lineare an, d.h. der Output Layer besteht ausschließlich aus einer affinen Vektor-zu-Skalar Transformation.

Anders als bisher, kann ein FNN mathematisch auch so beschrieben werden, dass die Schichten des Netzes aus parallel geschalteten Funktionen bestehen, welche jeweils einen Vektor in ein Skalar umwandeln. Diese Skalare werden als Vektor zusammengesetzt und an alle Funktionen des nächsten Layers gereicht. Die einzelnen Funktionen werden auch Neuronen genannt, da sie lose von Wissen über den biologischen Aufbau

von Nervensystemen inspiriert wurden. Ein Neuron ist also eine Aktivierungsfunktion, angewandt auf die Summe eines Vektor-Produkt mit einem Skalar. Abb. 5 veranschaulicht diesen Aufbau. Eine Kante stellt das Weiterreichen eines Skalars dar, die Knoten des Graphs sind die Neuronen.

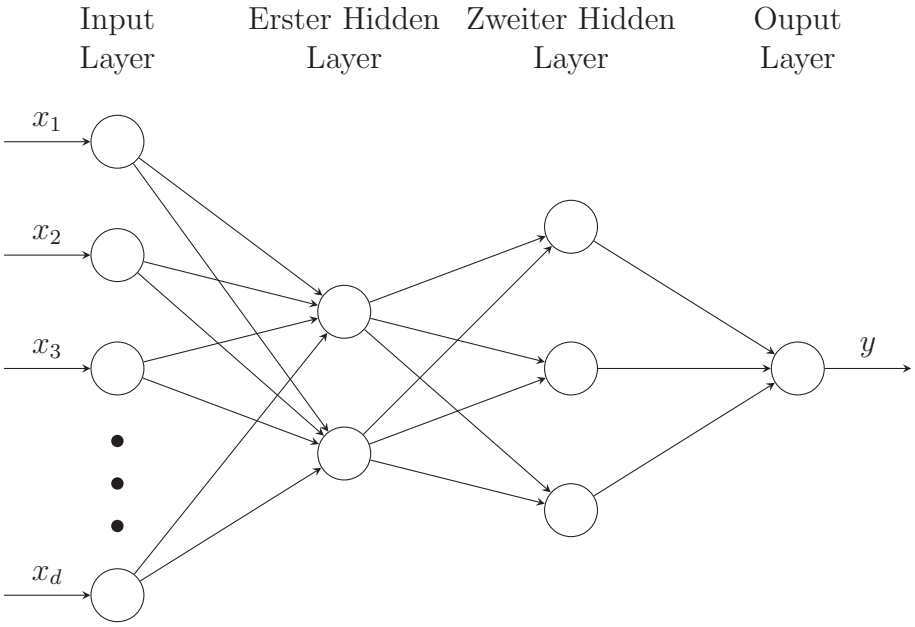

Abbildung 5: Ein FNN mit 2 Hidden Layer, welche 2 bzw. 3 Units haben, sowie einem skalaren Output.

Bevor ein FNN die optimalen Parameter erlernen kann, muss das genaue Design anhand einer Reihe von Hyperparametern festgelegt werden. Dazu gehören unter anderem die Tiefe (Anzahl der Layer), die Breite der einzelnen Layer (Anzahl der Neuronen), die Aktivierungsfunktion(en), die Verlustfunktion, sowie der genaue Lern-Algorithmus.

Neuronale Netze werden fast immer mit Gradientenverfahren, abgekürzt SGD (stochastic gradient descent), erlernt. SGD basiert darauf, dass ein NN als Verkettung differenzierbarer Funktionen ebenfalls differenzierbar ist, also der Gradient der Verlustfunktion in Bezug auf die Parameter kann berechnet werden. Um Minima zu erreichen, könnten in der Theorie Nullstellen berechnet werden, in der Praxis wird der Gradient jedoch in kleinen Schritten verringert. Ein Trainingsschritt des NN beinhaltet das Ziehen einer Teilmenge der Trainingsdaten (Batch), und der Erstellung von Prognosen mit dem aktuellen Netzwerk. Daraufhin wird der Verlust, sowie der Gradient bezüglich der bestehenden Parameter, berechnet. Diese werden dann ein wenig angepasst, sodass sich der Verlust verringert. Der Algorithmus zur Berechnung der Gradientenfunktion nennt sich Backpropagation, da der Gradient ausgehend vom Verlust rückwärts durch das Neuronale Netz berechnet wird. In der Praxis gibt es eine Vielzahl an SGD Verfahren, sie sind Teil von Optimierern wie Adagrad oder RMSProp (vgl. Chollet und Allaire, 2018).

Damit ein Modell nicht nur Trainingsdaten gut erlernt, sondern auch verallgemeinern

kann und neue Daten möglichst genau so gut vorhersagt, exisitieren verschiedenste Strategien, die unter dem Begriff Regularisierung zusammengefasst werden. Bei Neuronalen Netzen häufig implementiert wird die L_2-Regularisierung, auch Weight Decay genannt, sowie Dropout-Layer. Bei Anwendung der L_2-Regularisierung wird der Verlustfunktion ein Strafterm zugefügt, damit vor allem hohe Gewichte der Parameter θ der affinen Transformationen $W_j(\theta)$ bestraft werden. Ist $J(x; y; \theta)$ die Verlustfunktion, so hat die L_2-regularisierte Verlustfunktion die Form (vgl. Goodfellow et al., 2016 S. 227-230):

$$\tilde{J}(x; y; \theta) = \frac{\alpha}{2}\theta'\theta + J(x; y; \theta)$$

Dropout Layer sind Layer die zwischen 2 regulären Layer innerhalb eines FNN geschaltet werden. Ihre Funktionweise ist, dass während der Trainingsphase eine festgelegte Proportion an zufällig ausgewählten Neuronen fallen gelassen werden, d.h. dass deren Parameter gleich Null gesetzt werden (vgl. Srivastava et al., 2014). Durch das zufällige Auslassen von Neuronen werden viele leicht unterschiedliche Netze trainiert, sodass insgesamt ein robusteres entstehen kann.

Der Name feedforward kommt daher, dass es zwischen den Schichten keinen Rückfluss an Informationen gibt. Dies unterscheidet ein FNN beispielsweise zu einem Recurrent Neural Network (RNN). Ein LSTM ist ein spezielles RNN und wird im nächsten Abschnitt erklärt.

3.3.2 Long Short-Term Memory Networks

LSTM Netzwerke gelten als besonders gut geeignet auch langfristige Abhängigkeiten von Zeitreihen erfolgreich zu erlernen. Ob dies in dem Kontext dieser Arbeit ein Vorteil ist, ist fraglich, doch zumindest die generelle Eigenschaft von rekurrenten Netzwerken, die zeitliche Komponente zu erfassen, ist eine Motivation diese Methode hier einzusetzen. In anderen Publikationen zur Prognose von Solarenergie hat diese Methode vielversprechende Ergebnisse geliefert (vgl. Kumari und Toshniwal, 2021).

Um die Funktionsweise von LSTM-Netzwerken zu erklären folgt zunächst eine kurze Einführung zu simplen rekurrenten Netzwerken. In den zuvor besprochenen FNN's ist der Informationsfluss dadurch beschränkt, dass jeweils ein einzelner Input Vektor über das Netzwerk auf einen Output Vektor abgebildet wird. Hiervon unterscheidet sich ein RNN, da das Netzwerk theoretisch auf den gesamten historischen Input zurückgreifen kann um den Output zu modellieren (vgl. Graves, 2012, S. 18). Ein einfaches RNN ist im Prinzip ähnlich wie ein FNN mit nur einem Hidden Layer aufgebaut, wobei zusätzlich zum regulären Input auch noch Informationen der Hidden Layer vergangener Daten über eine rekurrente Verbindung als Input miteinfließen (vgl. Graves, 2012 S. 19). Anschaulich ist dies in Abb. 6 zu sehen:

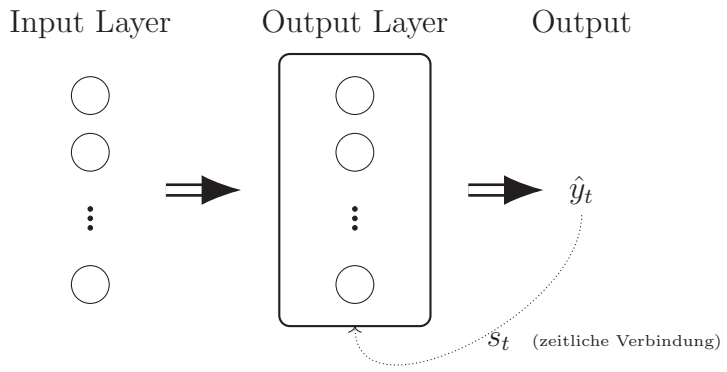

Abbildung 6: Aufbau eines simplen RNN mit einem Layer

Mathematisch sieht dies wie folgt aus:

$$\hat{y}_t = \sigma_j(Wx_t + Us_{t-1} + b)$$

wobei W und U Matrizen sind, b der Bias Vektor, x_t der Input zum Zeitpunkt t und $\hat{y}_t$ der entsprechende Output. Der State Vektor ist in diesem Fall einfach der Output Vektor des zeitlich letzten Outputs: $s_0 = 0,\ s_{t-1} = \hat{y}_{t-1},\ t = 2, 3, ...$ (vgl. Chollet und Allaire, 2018 S. 218).

LSTM sind eine Form von rekurrenten Netzwerken, welche ursprünglich von Hochreiter und Schmidhuber (1997) entwickelt wurden. Sie gelten als sehr effektiv in der Praxis. Ihre Stärke liegt darin, dass sie Informationen über Pfade lange Zeit speichern können und dabei selbst erlernen, wann historische Informationen wieder vergessen werden sollen (vgl. Goodfellow et al., 2016, S. 404). Anstelle der gewöhnlichen Neuronen in den Hidden Layer der FNN's bestehen LSTM Netzwerke aus Blöcken bzw. Zellen, die miteinander rekurrent verbunden sind und auch innerhalb ihrer Struktur rekurrente Schleifen haben, sowie mehreren Einheiten die den Informationsfluss kontrollieren (vgl. Goodfellow et al., 2016).

In Abb. 7 (vgl. Goodfellow et al., 2016; Graves, 2012) ist ihre Struktur dargestellt:

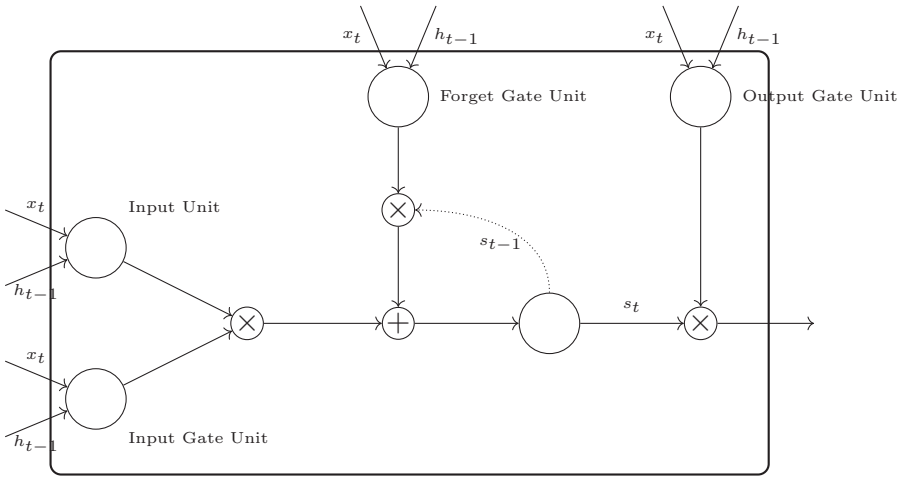

Abbildung 7: Aufbau eines Blocks innerhalb eines LSTM Netzwerks

Jeder Block i enthält zum Zeitpunkt t demnach eine State Unit $s_t^{(i)}$ mit rekurrenter Schleife, die von einer Forget Gate Unit $f_t^{(i)}$ kontrolliert wird, zudem eine Input Gate Unit $g_t^{(i)}$ und eine Output Gate Unit $q_t^{(i)}$. Wie bei regulären Neuronen bestehen die Units aus einer Aktivierungsfunktion, in diesem Fall der Sigmoidfunktion, und einer affinen Transformation. Bezeichnet x_t den aktuellen Input Vektor und h_t den Vektor der Hidden Layer, so sieht das ganze mathematisch folgendermaßen aus (vgl. Goodfellow et al., 2016 S. 406-407)):

$$
\begin{aligned}
f_t^{(i)} &= \sigma(U^{(f)\prime} x_t + W^{(f)\prime} h_{t-1} + b^{(f)}) \\
g_t^{(i)} &= \sigma(U^{(g)\prime} x_t + W^{(g)\prime} h_{t-1} + b^{(g)}) \\
q_t^{(i)} &= \sigma(U^{(q)\prime} x_t + W^{(q)\prime} h_{t-1} + b^{(q)}) \\
s_t^{(i)} &= f_t^{(i)} s_{t-1}^{(i)} + g_t^{(i)} \sigma(U^{(s)\prime} x_t + W^{(s)\prime} h_{t-1} + b^{(s)}) \\
h_t^{(i)} &= tanh(s_t^{(i)}) q_t^{(i)}
\end{aligned}
$$

mit den Vektoren $U^{(\cdot)}$, $W^{(\cdot)}$ und Bias Skalaren $b^{(\cdot)}$, die für jede Unit eigene Parameter enthalten.

3.4 Ensemble Modell

Ensemble Modelle sind Methoden, die mehrere Modelle in einem vereinen. Dabei können diese verschiedene Strukturen haben oder aber der gleichen Grundstruktur entspringen. Ein Ensemble wurde bereits als Random Forest vorgestellt. In diesem wurden die Prognosen vieler Regression Trees gebündelt, um die Schwächen einzelner auszugleichen. Nach dem gleichen Prinzip lassen sich auch die Vorhersagen grundsätzlich verschiedener Modelle bündeln und somit ein Ensemble Modell bauen, das die Schwächen einzelner ausgleicht. Die Idee dahinter ist, dass möglichst verschiedene Modelle die Daten auf verschiedene Art und Weise erlernen und sich somit ergänzen können. Eine simple Vorgehensweise ist, den (gewichteten) Mittelwert der Prognosen aller

verfügbaren Modelle zu betrachten. Beispielsweise haben die Autoren Gigoni et al. (2018) ein solch einfaches Ensemble aus 5 verschiedenen Modellen genutzt, um eine 24 Stunden Vorhersage der Stromproduktion von 32 PV-Anlagen zu erstellen. Diese hat laut den Autoren die einzelnen Modelle übertroffen.

Diese Ensemble Methode wird als letztes Modell in dieser Arbeit verwendet: Sei x_t ein Vektor und seien $M^{(j)}(x_t) = \hat{y}_t^{(j)}$, $j = 1, ..., 4$, die entsprechenden Vorhersagen der 4 Modelle, dann ist die Prognose des Ensemble Modells gegeben durch:

$$\hat{y}_t = M(x_t) = \sum_{j=1}^{4} c_j \hat{y}_t^{(j)},$$

wobei

$$0 \leq c_j \leq 1 \text{ und } \sum_{j=1}^{4} c_j = 1.$$

Die optimalen Gewichte c_j können beispielsweise mit Optimierungsalgorithmen, mit einem Random Search oder aber einem Grid Search bestimmt werden.

4 Methodik

4.1 Modellentwicklung und Hyperparameter-Optimierung

Das Ziel ist, mittels der gegebenen Wetter- und Solarenergieprognosen Modelle zu erlernen, die die stündliche Leistung der PV-Anlagen prognostizieren. Da es sich bei den Daten um Zeitreihen handelt, wird diese Struktur bei der Entwicklung der Versuche berücksichtigt. Für jeden der Machine Learning Ansätze wird jeweils ein eigenes Modell für beide PV-Anlagen und für beide Datensätze der numerischen Wetterdaten angepasst. Die Daten werden jeweils in einen Trainings- und in einen Testdatensatz unterteilt. Letzterer enthält die letzten 25% der Tage, an denen es Messungen gibt. Für Anlage K befinden sich somit 2103 Messungen im Trainingsdatensatz und 557 im Testdatensatz, für Anlage S 1411 und 506.

Die genaue Architektur der Machine Learning Modelle wird durch ihre Hyperparameter bestimmt, diese müssen festgelegt werden bevor ein Modell trainiert wird. Um sie optimal auszuwählen, wird ein sinnvoller Bereich an Werten festgelegt, um daraus eine möglichst große Anzahl an Modellen zu trainieren und zu evaluieren und so die beste Kombination der Hyperparameter zu finden. Für diesen Schritt werden die Trainingsdaten erneut in 2 Mengen unterteilt. Die (zeitlich) ersten 70 % der Trainingsdaten dienen ausschließlich dem Erlernen des Modells mit verschiedenen Hyperparameter-Kombinationen. Die restlichen 30 % umfassen den Validierungs-Datensatz, anhand derer die Modelle bewertet werden. Somit soll verhindert werden, dass Informationen bereits in die Entwicklung des Modells einfließen und die Evaluierung verzerren. Die ML Modelle werden nach dem mittleren quadratischen Fehler (MSE) optimiert.

4.2 Feature-Engeneering

Die Features beider Datensätze, wie etwa Temperatur oder Bewölkung, haben alle sehr unterschiedliche Skalen. Um potentielle Probleme zu vermeiden, werden vor dem Trainieren der Modelle die Werte normalisiert. D.h. für $i = 1, ..., d$ und $j = 1, ..., N$ wird

$$x_{i,j} \longrightarrow \frac{x_{i,j} - \hat{\mu}_j}{\hat{\sigma}_j},$$

wobei d die Dimension der erklärenden Variable ist und N die Anzahl der verfügbaren Daten. $\hat{\mu}_j$ und $\hat{\sigma}_j$ bezeichnen die empirischen Mittelwerte und Varianzen der Feature-Variable, berechnet mit den Daten des Trainingsdatensatzes. Die Testdaten werden mit den gleichen Werten $\hat{\mu}_j$ und $\hat{\sigma}_j$ normalisiert.

4.3 Evaluierung

Die Modelle werden mit dem Tesdatensatz evaluiert. Entsprechend dem Anwendungsfall und da der Trainingsdatensatz nicht sehr groß ist, wird bei der Auswertung das jeweilige Modell jeden Tag aktualisiert. Das heißt, vor der Prognose eines jeden Datenpunktes wird jedes Modell mit allen Daten bis zum Vortag trainiert. Dies entspricht damit einer 24 Stunden Vorhersage, was in der Regel auch als day-ahead prediction in der Literatur bezeichnet wird (vgl. Ahmed et al., 2020).

Sei also N die Anzahl der insgesamt verfügbaren Datenpunkte $z_t = (x_t, y_t)_{t=1}^N$ und N' die Anzahl der davon verfügbaren Testdaten. In der Testphase werden zu den Regressoren $(x_{N-N'+1}, ..., x_N)$ Prognosen erstellt und mit den wahren Werten $(y_{N-N'+1}, ..., y_N)$ verglichen. Sei x_j ein Datenpunkt im Testdatensatz und der $m-$te Datenpunkt eines Tages. Sei $M(\theta, x)$ ein ML Modell mit bereits gewählten Hyperparametern, dann werden die Parameter θ des Modells anhand der Daten $(z_1, ..., z_{j-m})$ erlernt, um anschließend eine Prognose $\hat{y}_j = M(\hat{\theta}, x_j)$ zu Erstellen. Die Prognosen der Datenpunkte des gleichen Tages werden mit dem selben erlernten Modell erstellt. Da im Laufe des Jahres die Anzahl der Stunden mit Tageslicht variiert, haben auch die Tage des Datensatzes eine je unterschiedliche Anzahl an Datenpunkten, zwischen 10 und 18. Anschaulich ist die Evaluierung im Folgenden dargestellt:

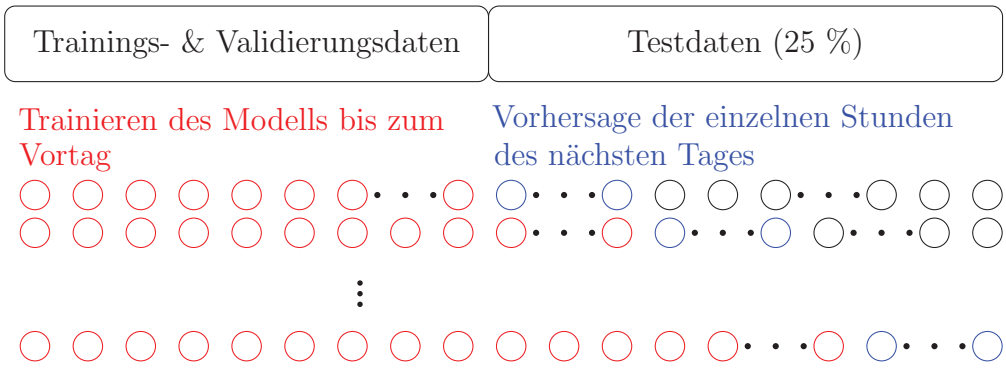

Abbildung 8: Schematische Darstellung der Erstellung der Prognosen des Testdatensatzes

Um die Genauigkeit der Vorhersagen zu betrachten und vergleichbar zu machen, wird eine Reihe an Metriken eingesetzt. Die in der Literatur zu diesem Thema am häufigsten verwendeten Maße sind der MAE (Mean Absolute Error), MAPE (Mean Absolute Percentage Error), RMSE (Root Mean Square Error), MBE (Mean Bias Error) und das Bestimmtheitsmaß R^2. Um Ergebnisse von PV-Anlagen mit verschiedenen Kapazitäten vergleichbar zu machen werden meist zusätzlich normalisierte Werte betrachtet. Die Art der Normalisierung ist in der Literatur nicht einheitlich. So wird häufig der Wert einer Metrik entweder durch die Maximalleistung oder aber durch die durchschnittliche Leistung der Testdaten dividiert (vgl.Bright, 2019; Zhang et al., 2015).

Bei der Auswertung finden in dieser Arbeit folgende Maße Verwendung: Der MAE, da er er die mittlere Abweichung der Vorhersagen von den wahren Werten angibt und somit gut verständlich ist. Der RMSE als wichtigstes Maß in dieser Arbeit, da er zusätzlich einzelnen großen Abweichungen mehr Gewicht gibt. Die normalisierten Werte dienen dazu, die verschiedenen Anlagen vergleichbar zu machen. Der nMBE zeigt an, ob und wie stark die Vorhersagen die wahren Werte im Mittel eher überschätzt (nMBE>0) oder unterschätzt (nMBE<0) haben. Weiterhin wird auch das Bestimmtheitsmaß R^2 angegeben. Seien $(\hat{y}_{N-N'+1}, ..., \hat{y}_N)$ die prognostizierten Werte im Testdatensatz, dann berechnen sich diese Maße wie folgt:

$$RMSE = \sqrt{\frac{1}{N'} \sum_{j=N-N'+1}^{N} (y_j - \hat{y}_j)^2}$$

$$MAE = \frac{1}{N'} \sum_{j=N-N'+1}^{N} |y_j - \hat{y}_j|$$

$$MBE = \frac{1}{N'} \sum_{j=N-N'+1}^{N} (\hat{y}_j - y_j)$$

$$R^2 = 1 - \frac{\sum_{j=N-N'+1}^{N} (y_j - \hat{y}_j)^2}{\sum_{j=N-N'+1}^{N} (y_j - \overline{y})^2}$$

Um die normalisierten Werte zu erhalten, werden die Maße mit dem Faktor $\frac{1}{\overline{y}} = \frac{1}{\frac{1}{N'} \sum_{j=N-N'+1}^{N} y_j}$ multipliziert.

4.4 Implementierung der Modelle und Wahl der Hyperparameter

Die gesamte Implementierung erfolgt in R . In den folgenden Abschnitten wird auf die Details zu den einzelnen Modelle eingegangen.

4.4.1 Lineare Regression

Um die Ergebnisse der Machine Learning Modelle einordnen zu können, wird zunächst als einfaches Baseline Modell eine Lineare Regression durchgeführt. Implementiert wird

es mittels der R-Funktion *stats* :: *lm*().

4.4.2 Random Forest

Das Random Forest Modell wird mit den R-Packages *Ranger* (vgl. Wright und Ziegler, 2017) und *mlr3* (vgl. Lang et al., 2019) implementiert. Wie bereits im theoretischen Teil erklärt, sind einige Hyperparameter zu optimieren:

- *num.tree*: die Anzahl der Bäume
- *mtry*: Anzahl der zufällig versuchten Features an jedem Split
- *min.node.size*: die minimal erlaubte Anzahl an Datenpunkten innerhalb eines Blatts
- *max.depth*: maximale Tiefe eines Baums

Zur Optimierung wird zunächst ein breiter Wertebereich für jede einzelne Variable festgelegt, diese sind in Tabelle 5 zu sehen. Da in der Regel die Anzahl der Bäume nach oben hin keinen Nachteil mit sich bringt, wird sie zunächst auf 300 festgelegt und anschließend überprüft, ob dies ausreichend war (vgl Abbildung 9 2. Zeile, rechts).

Hyperparameter	Wertebereich
mtry	1-18
min.node.size	2 - 40
max.depth	2-30
num.trees	300

Tabelle 5: Random Forest Hyperparameterraum

Aus dem Raum der Hyperparameter werden je 2000 Modelle mit zufälligen Kombinationen berechnet und ausgewertet, die Kombination mit dem geringsten Verlust wird verwendet. In Tabelle 6 sind die optimalen Parameter für alle 4 Modelle abgebildet:

	Datensatz 1		Datensatz 2	
Hyperparameter	Anlage S	Anlage K	Anlage S	Anlage K
mtry	14	4	13	10
min.node.size	4	20	4	4
max.depth	22	7	27	21
num.trees	300	300	300	300

Tabelle 6: Hyperparameter

Abbildung 9 zeigt Schwankungen der Risikofunktion (RMSE) gegenüber der Werte der Hyperparameter beispielhaft anhand der Ergebnisse für Datensatz 1 und Anlage K. Dort zeigt sich (1. Zeile, links), dass eine Einschränkung der Tiefe der Bäume eher negative Auswirkungen hat. Eine Einschränkung der Blattgröße scheint keinen Einfluss

zu haben. Links unten in der Abbildung ist zu sehen, dass sowohl eine zu große als auch eine zu kleine Zahl an Splits die Prognosen verschlechtert. Vor der Auswertung anhand des Testdatensatzes wird mit den optimalen Hyperparametern noch überprüft ob 300 Bäume ausreichend sind. Wie rechts unten in Abbildung 9 zu sehen ist, ist dies der Fall. Beachtet man die RMSE Werte auf der y-Achse, so sieht es aus, dass selbst Modelle mit deutlich weniger Bäumen (d.h. unter 50) nur geringfügig schlechtere Ergebnisse liefern.

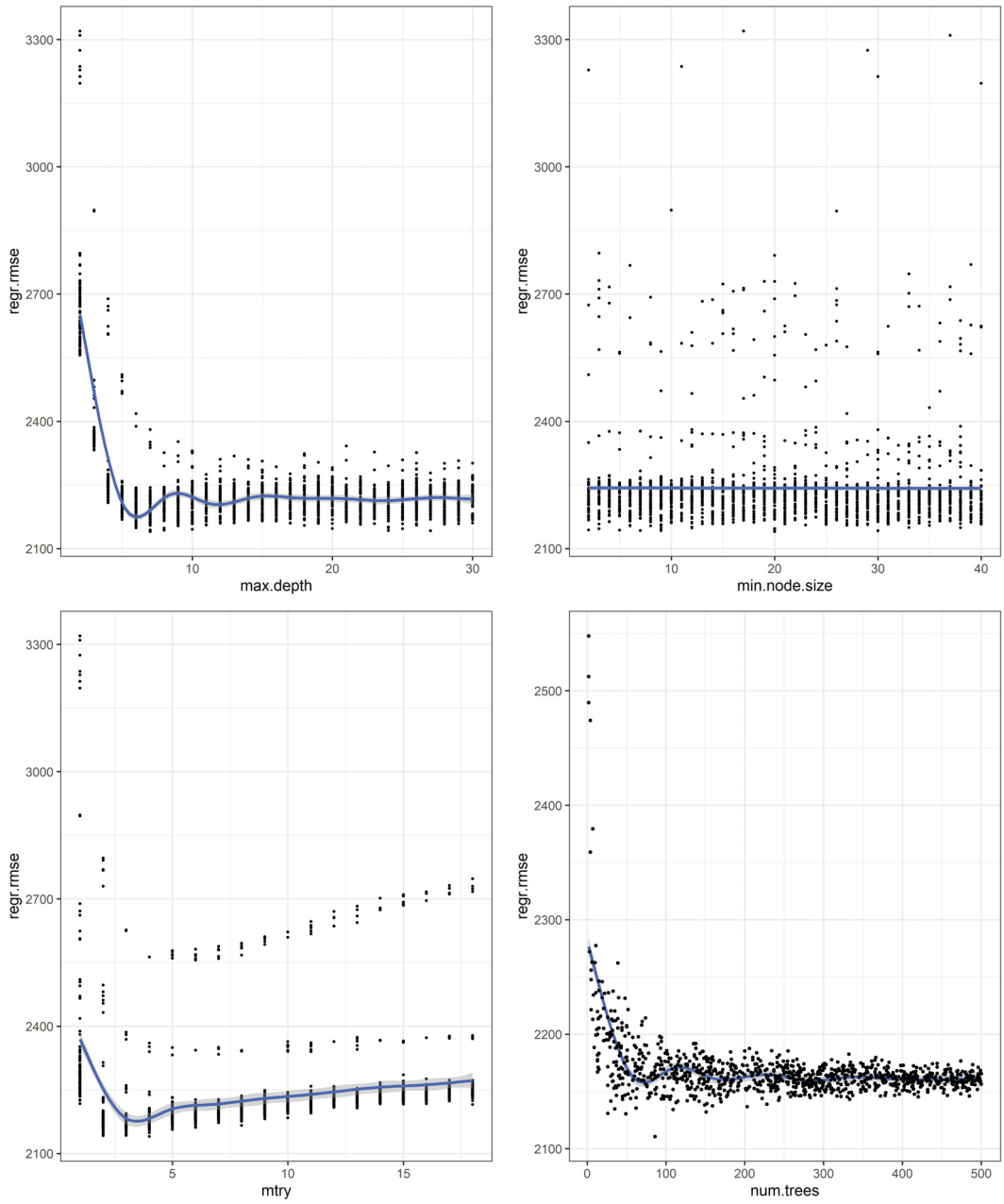

Abbildung 9: Visualisierung des Einflusses der Hyperparameter auf die Risikofunktion. Hier als Beispiel zu Datensatz 1 und Anlage K

4.4.3 FNN

Die Neuronalen Netze sind mit dem R-Interface zu *Keras* (vgl. Chollet, Allaire et al., 2017) und *Tensorflow* (vgl. Allaire et al., 2016) implementiert. Die genaue Architektur der Netzwerke wird auch hier durch einen Random Search bestimmt. Zunächst wird eine Menge (Hyperparameterraum) an möglichen Strukturen festgelegt. Mit zufällig gewählten Kombinationen aus diesem Raum werden Modelle erlernt und anhand der Validierungsdaten ausgewertet. Im Tabelle 7 sind die Wertebereiche der Hyperparameter abgebildet.

Hyperparameter	Wertebereich
Anzahl der Hidden Layer	1 - 4
Units/Einheiten	4 - 512
Dropout Quote	0 - 0,5
Weight Decay / L2- Regularisierung	0 - 0,5
Batch Size	2 - 64

Tabelle 7: FNN Hyperparameterraum

Ein weiterer Parameter ist die Anzahl der Epochs, das heißt wie oft das Netz im Lernprozess die gesamten Trainingsdaten durchläuft. Mit Keras Callbacks wird während des Trainingsprozesses der Wert der Verlustfunktion der Validierungsdaten gespeichert, sodass der Lernprozess abgebrochen wird, wenn 5 Epochs lang keine Verbesserung messbar ist. Die Aktivierungsfunktion ist in allen Modellen die ReLU()-Funktion, der Optimierungsalgorithmus *RMSProp*, die Verlustfunktion der RMSE, und der Output Layer ist linear.

Für jedes Modell werden zunächst etwa 100 zufällige Kombinationen der Hyperparameter getestet. Eine erste Auswertung zeigt, dass eine kleine Batch-Size deutlich bessere Ergebnisse produziert, zudem haben Netze mit 2 oder 3 Hidden Layer den geringsten Verlust, sowie Netze mit nur einem Dropout-Layer. Dementsprechend wird der Hyperparameterraum weiter eingegrenzt auf einen Dropout-Layer, die Batch Size wird auf 2 festgelegt, die Anzahl der Hidden Layer auf 2-3. Eine erneute Optimierung wird gestartet und die Ergebnisse betrachtet. Erzielen mehrere Netzwerke ungefähr den gleichen Verlust, wird das einfachere vorgezogen. In der folgenden Tabelle sind die gewählten Hyperparameter, sowie die sich damit ergebende Anzahl der trainierbaren Parameter in den Modellen, eingetragen:

Hyperparameter	Datensatz 1		Datensatz 2	
	Anlage S	Anlage K	Anlage S	Anlage K
Anzahl der Hidden Layer	3	3	2	2
Units/Einheiten	96; 48; 32	128; 96; 32	128; 64	128; 64
Dropout Quote (Layer 1)	0,4	0,4	0	0
Weight Decay	0,05	0,5	0,01	0,01
Batch Size	2	2	2	2
Anzahl der Parameter	8.657	18.721	10.753	10.753

Tabelle 8: Hyperparameter der FNN und die Anzahl der Parameter in den Modellen

Als Beispiel ist im Folgenden die Implementierung des FNN für Anlage K und Datensatz 1 zu sehen:

```
model <- keras_model_sequential() %>%
  layer_dense(units = 128, activation = "relu",
              input_shape = dim(train_data)[2],
              kernel_regularizer = regularizer_l2(0.5)) %>%
  layer_dropout(rate = 0.4)%>%
  layer_dense(units = 96, activation = "relu",
              input_shape = dim(train_data)[2],
              kernel_regularizer = regularizer_l2(0.5)) %>%
  layer_dense(units = 32, activation = "relu",
              input_shape = dim(train_data)[2],
              kernel_regularizer = regularizer_l2(0.5)) %>%
  layer_dense(units = 1)

  model %>% compile(
    optimizer = "rmsprop",
    loss = "mse",
    metrics = c("mae")
    )
```

Listing 1: FNN Model für Anlage K und Datensatz 1

4.4.4 LSTM

Die Hyperparameter Optimierung erfolgt wie bei FNN's. Als Regularisierer werden hierbei jedoch Dropout und Recurrent Dropout Layer verwendet. Um die gesamte Anzahl an möglichen Kombinationen etwas einzugrenzen, werden deren Werte in jedem Layer gleich gesetzt. Da es sich zeigt, dass LSTM Networks langsamer konvergieren, wird für kleinere Netzwerke die Learning Rate des Optimierers heraufgesetzt. Tabelle 9 zeigt die Ergebnisse der Hyperparameter Optimierung.

Hyperparameter	Datensatz 1		Datensatz 2	
	Anlage S	Anlage K	Anlage S	Anlage K
Anzahl der Hidden Layer	2	2	3	3
Units/Einheiten	128; 64	128; 64	32; 96; 64	128; 96; 32
Dropout Quote	0,4	0,4	0	0,2
Recurrent Dropout Quote	0	0	0,2	0,4
Batch Size	2	2	2	2
Anzahl der Parameter	127.809	127.809	97.345	178,209

Tabelle 9: Hyperparameter der LSTM Netzwerke und die Anzahl der Parameter in den Modellen

Es folgt ein Code-Beispiel für Datensatz 1, Anlage K:

```
model <- keras_model_sequential() %>%
  layer_lstm(units = 128, activation = "relu",
             input_shape = c(dim(partial_train_data)[2],dim(partial_
  train_data)[3]),
             return_sequences = TRUE,
             dropout =0.4, recurrent_dropout = 0) %>%
  layer_lstm(units=64, activation = "relu",
             dropout=0.4, recurrent_dropout = 0) %>%
  layer_dense(units = 1)
```

Listing 2: LSTM Model in R für Datensatz 1

4.4.5 Ensemble Modell

Zunächst müssen die Gewichtungen c_j der 4 einzelnen Modelle festgelegt werden. Dazu wird hier ein recht simpler Ansatz verfolgt. Die 4 Modelle werden wie bisher anhand der Trainingsdaten erlernt und mittels der Validierungsdaten ausgewertet. Seien y_t die wahren Werte und $\hat{y}_t^{(j)},\ j = 1, ...4, t = 1, ..., N'$ die Prognosen der 4 Modelle (Lineare Regression, Random Forest, FNN, LSTM), wobei N' die Größe des Validierungsdatensatzes bezeichnet. Minimiert werden soll der RMSE, d.h. gesucht ist:

$$\min_{c_j} \sum_{t=1}^{N'} (y_t - \sum_{j=1}^{4} c_j \hat{y}_t^{(j)})^2$$

Es wird ein einfacher Random Search im zulässigen Bereich der c_j verwendet, um die ideale Kombination zu finden.

Es zeigt sich, dass in der optimalen Kombination der Prognosen das lineare Modell jeweils nichts beiträgt, d.h. die Gewichtungen betragen 0. Es wird daher eine erneuter Random Search, eingeschränkt auf ein Ensemble Modell der 3 verbliebenen Modelle, durchgeführt.

Folgende Gewichtungen ergeben sich, es lässt sich jedoch kein Muster zwischen den einzelnen Modellen erkennen:

Modell	Datensatz 1		Datensatz 2	
	Anlage S	Anlage K	Anlage S	Anlage K
Lineare Regression	0	0	0	0
Random Forest	0,388	0,771	0,003	0,440
FNN	0,199	0,125	0,030	0,439
LSTM	0,413	0,104	0,967	0,121

Tabelle 10: Gewichtungen im Ensemble Modell

4.5 Feature Importance

Die meisten Machine Learning Modelle, insbesondere auch Random Forest und Neuronale Netze, gelten als Black-Box Modelle, da nach dem Erlernen der Parameter ihre Funktionsweise nicht einsehbar ist. Dennoch gibt es Möglichkeiten Einblicke zu bekommen. Interessant ist bei dieser Arbeit der Beitrag der einzelnen Wetter-Variablen auf die Prognosefähigkeit. In diesem Abschnitt wird dies anhand der Random Forest Modelle betrachtet. Die Wichtigkeit der Features innerhalb eines Modells kann mittels *Permutation Feature Importance* abgeschätzt werden. Hierbei werden die Daten eines einzelnen Features manipuliert und anschließend das Modell neu ausgewertet, sodass der Effekt der Manipulation auf den Verlust (RMSE) des Modells berechnet werden kann (vgl. Becker et al., 2022 Kapitel 8.5). Je stärker der Verlust dadurch wächst, desto wichtiger ist das einzelne Feature im erlernten Modell.

In Abbildung 10, erstellt mit R-Package *IML* (vgl. Molnar et al., 2018), ist dieser Index für alle 4 Modelle und deren Features abgebildet. Der Index zeigt den Faktor an, mit dem sich der Fehler verschlechtert hat. Beispielsweise bedeutet 2 eine Verdopplung des RMSE unter der Permutation der Werte des Features.

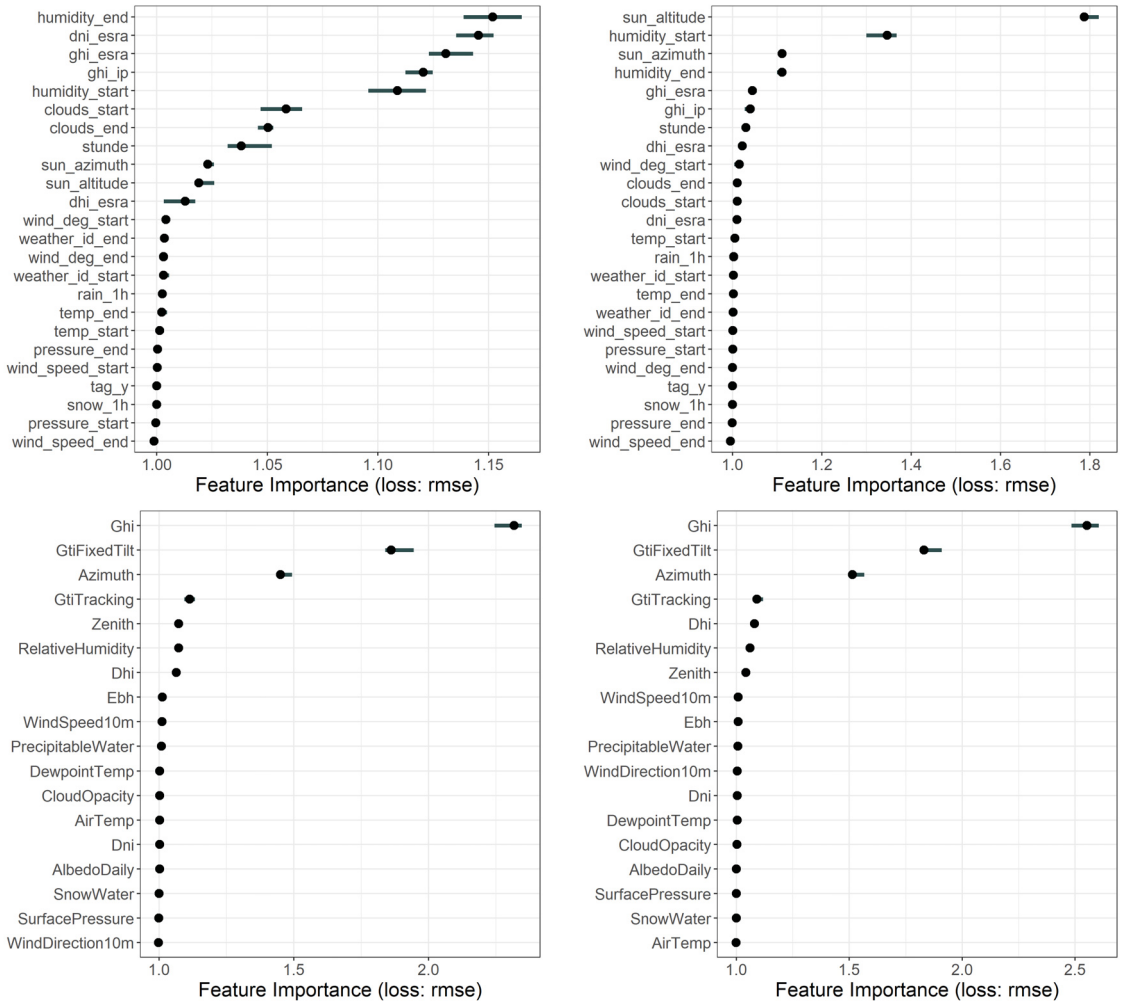

Abbildung 10: Permutation Feature Importance, links: Anlage K, rechts: Anlage S, oben: Datensatz 1, unten: Datensatz 2

Bei Datensatz 1 (erste Zeile in Abbildung 10) ist zu erkennen, dass die sonnengeometrischen Daten eine wichtige Rolle spielen. Unter den Wetter Features sind vor allem die Luftfeuchtigkeit und Bewölkung ein entscheidender Faktor. Die genaue Bedeutung ist natürlich nicht einfach daraus ablesbar, da einige Variablen sehr ähnliche Informationen enthalten. So ist zum Beispiel die Sonnenhöhe (sun_altitude) stark korreliert mit den Werten Ghi, Dni und Dhi, die auch untereinander stark korrelieren. Diese Daten geben lediglich einen Hinweis auf die Wichtigkeit innerhalb des erlernten Modells. So ist es in Datensatz 2 nicht überraschend, dass die Wettervariablen kaum eine Rolle zu spielen scheinen, da Werte wie der GHI bereits die Sonnenenergie unter Beachtung des Wetters angibt.

4.6 Laufzeiten

In Tabelle 11 sind die gemessenen Laufzeiten für das Trainieren eines Modells mit den bereits festgelegten Hyperparametern zu sehen. Die Vorhersage neuer Daten ist bei

allen Modellen sehr schnell ($\ll 1s$). Die Hyperparameter Optimierung dauert hingegen länger, entsprechend der Anzahl an Modellen die getestet werden. Die Berechnungen wurden in R auf einem Notebook mit Intel Core i5-10210U CPU (1.6GHz) durchgeführt.

	Datensatz 1		Datensatz 2	
	Anlage S	Anlage K	Anlage S	Anlage K
Lineares Modell	$\ll 1s$	$\ll 1s$	$\ll 1s$	$\ll 1s$
Random Forest	$\approx 1s$	$\approx 1s$	$\approx 1s$	$\approx 1s$
FNN	$\approx 27s$	$\approx 36s$	$\approx 80s$	$\approx 85s$
LSTM	$\approx 106s$	$\approx 90s$	$\approx 189s$	$\approx 202s$

Tabelle 11: Laufzeiten (in Sekunden) für das Trainieren der Parameter von jeweils einem Modell

5 Resultate

Alle 5 Modelle werden anhand der Testdaten evaluiert, die Ergebnisse befinden sich in Tabellen 12, 13, 14 und 15 und enthalten die Werte der Maße RMSE, nRMSE, MAE, nMAE, nMBE und R^2.

Abbildung 11 zeigt die Entwicklung des kumulierten quadratischen Fehlers der einzelnen Modelle im zeitlichen Verlauf der Testdaten. Insgesamt zeigt sich, dass die Verwendung eines Ensemble Modells hier keinen erkennbaren Vorteil bringt.

Unter den einzelnen Modellen liefert für Datensatz 1 das LSTM Network den kleinsten Verlust. Für Anlage S weisen alle Machine Learning Modelle eine ähnlich gute Leistung, und eine kleine Verbesserung gegenüber der Lineare Regression, auf. Bei Anlage K hebt sich hingegen das LSTM Network deutlicher von allen anderen ab. Für Datensatz 2 liefert hingegen Random Forest bei beiden Anlagen die Beste Leistung.

Abbildungen 12 und 13 vergleichen die prognostizierten Werte mit den gemessenen Werten. Die Prognosen kommen vom jeweils besten Modell des Datensatzes (LSTM bzw. Random Forest). Wie zu erwarten ist, sind die Abweichungen der Werte im Bereich niedriger Leistung dementsprechend kleiner als bei hoher Leistung der PV-Anlagen. Die größten Abweichungen sind eher bei mittlerer Stromproduktion der Anlagen zu erkennen. Tendenziell scheinen die Modelle bei hoher Leistung diese leicht zu unterschätzen und dafür bei geringer bis mittlerer Leistung etwas zu überschätzen.

Insgesamt zeigt sich in den Tabellen und Grafiken, dass Datensatz 1 im Vergleich zu Datensatz 2 doch eher schwache Ergebnisse liefert. Während mit Datensatz 2 ein nRMSE von 19,81 % (Anlage S) und 23,83 % (Anlage K) erreicht wird, liegt dieser bei Datensatz 1 bei 35,52 % bzw. 41,61 %. Es ist also offensichtlich, dass sich mit den verwendeten Daten von Solcast hier deutlich bessere Prognosen erstellen lassen als mit den verwendeten Wetterdaten von OpenWeather.

Zwischen den Anlagen bestehen offensichtlich auch Unterschiede. Während mit Datensatz 2 bei Anlage S ein nRMSE von 19,81 % erreicht wird, liegt dieser bei Anlage K bei 23,83 %. Diese Unterschiede könnten wohl aufgrund der verschiedenen Größen der Datensätze der PV-Anlage zustande kommen. Insbesondere liegen die Zeiträume der Testdaten für Anlage S im Sommer, und für Anlage K im Herbst in dem weniger sonnige Tage zu erwarten sind. Dies würde sich mit den Beobachtungen anderer Autoren decken, die speziell die Prognosegenauigkeit unter verschiedenen Bedingungen untersucht haben und dabei beobachten konnten, dass vor allem bei klarem Himmel die Vorhersagen deutlich besser sind (vgl. Gigoni et al., 2018 Bright, 2019). Auch in den Abbildungen 12 und 13 ist zu erkennen, dass Ausreißer eher bei mittlerer Leistung zu beobachten sind, statt bei besonders hoher Leistung, die bei klarem Himmel erzielt wird. Ein weiterer Unterschied zwischen beiden Anlagen zeigt sich im nMBE, der bei Anlage K für alle Modelle deutlich positiv ist und damit im Schnitt eine Überschätzung der wahren Werte aufzeigt. Dagegen ist für Anlage S der nMBE näher bei 0 oder sogar negativ. Ein negativer Wert zeigt eine Unterschätzung der Werte im Mittel an.

Abbildungen 14 und 15 zeigen den Tagesverlauf von gemessener und prognostizierter Leistung im Vergleich an einigen ausgewählten Tagen. Dies sind die 4 Tage mit kleinstem Verlust, die 4 mit größtem Verlust und die 4 Tage, deren Verlust rund um den Medianwert liegen. Diese Abbildungen werden hier nur für Anlage K angegeben, da sich bei beiden Anlagen ungefähr das gleiche Bild ergibt. Verwendet werden jeweils die Prognosen des besten Modells, d.h. LSTM Networks für Datensatz 1 und Random Forest für Datensatz 2. Es zeigt sich, dass das Modell für Datensatz 2 selbst an Tagen mit tendenziell schlechter Prognosegenauigkeit durchaus noch brauchbare Ergebnisse liefert. Es sind jedoch kleine Ausreißer einzelner Stunden die große Lücken zwischen Prognose und echter Leistung reißen. Zumindest bilden für beide Datensätze die prognostizierten Daten den glockenförmigen Verlauf der echten Messungen meist ab, dies war jedoch auch zu erwarten, da sonnengeometrische Daten Teil des Inputs der Modelle sind.

	RMSE	nRMSE	MAE	nMAE	nMBE	R^2
Lineares Modell	2197,61	47,98 %	1716,52	37,47 %	1,89 %	0,73
Random Forest	2071,19	45,22 %	1500,31	32,75 %	3,92 %	0,76
FNN	2095,08	45,74 %	1529,68	33,39 %	1,33 %	0,75
LSTM	1906,12	41,61 %	1398,32	30,53 %	2,95 %	0,80
Ensemble	2032,38	44,37 %	1478,90	32,29 %	3,50 %	0,77

Tabelle 12: Ergebnisse für Anlage K, Datensatz 1

	RMSE	nRMSE	MAE	nMAE	nMBE	R^2
Lineares Modell	1353,25	29,54 %	1040,64	22,72 %	3,66 %	0,90
Random Forest	1091,56	23,83 %	687,67	15,01 %	2,07 %	0,93
FNN	1207,11	26,35 %	783,28	17,10 %	4,15 %	0,92
LSTM	1223,50	26,71 %	791,89	17,29 %	8,03 %	0,92
Ensemble	1119,14	24,43 %	697,07	15,22 %	3,70 %	0,93

Tabelle 13: Ergebnisse für Anlage K, Datensatz 2

	RMSE	nRMSE	MAE	nMAE	nMBE	R^2
Lineares Modell	8687,44	39,67 %	6498,97	29,68 %	-2,38 %	0,76
Random Forest	7968,17	36,39 %	5327,24	24,33 %	-0,59 %	0,80
FNN	8009,85	36,58 %	5559,83	25,39 %	-4,62 %	0,79
LSTM	7845,43	35,83 %	5467,46	24,97 %	-4,49 %	0,80
Ensemble	7667,99	35,02 %	5243,37	23,95 %	-3,00 %	0,81

Tabelle 14: Ergebnisse für Anlage S, Datensatz 1

	RMSE	nRMSE	MAE	nMAE	nMBE	R^2
Lineares Modell	5186,57	23,69 %	3965,70	18,11 %	0,15 %	0,91
Random Forest	4338,37	19,81 %	2812,04	12,84 %	-1,21 %	0,94
FNN	4601,43	21,01 %	3162,34	14,44 %	-0,25 %	0,93
LSTM	4689,21	21,42 %	3006,43	13,73 %	0,10 %	0,93
Ensemble	4657,43	21,27 %	2983,27	13,62 %	0,08 %	0,93

Tabelle 15: Ergebnisse für Anlage S, Datensatz 2

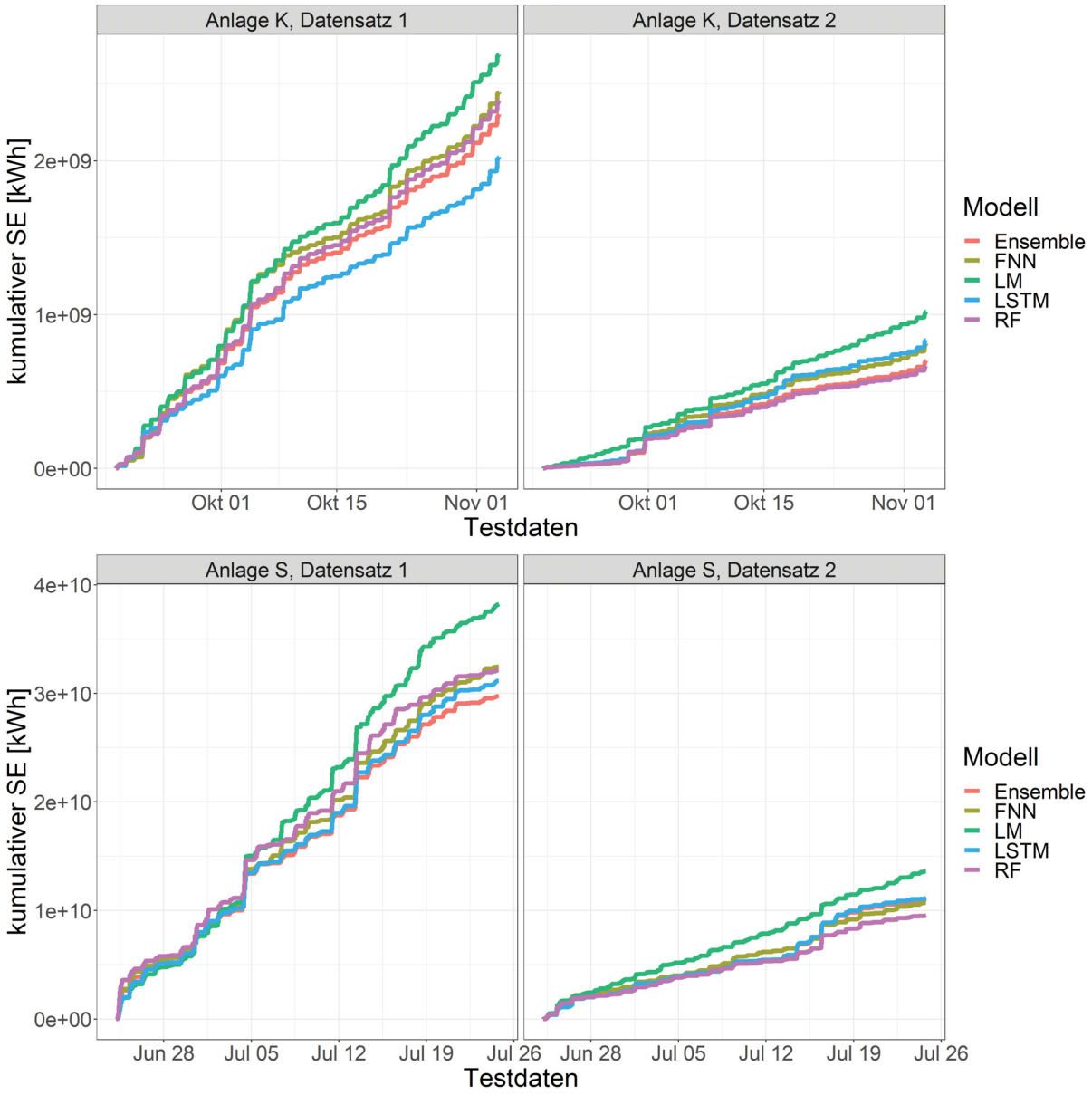

Abbildung 11: Kumulativer quadratischer Fehler (SE) der Modelle

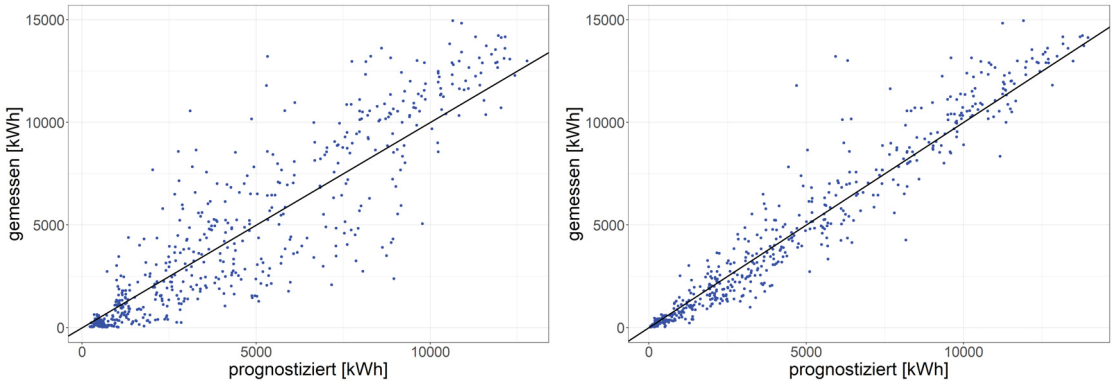

Abbildung 12: Anlage K: prognostizierte vs. gemessene Leistung (links: Datensatz 1, LSTM Modell; rechts: Datensatz 2; Random Forest Modell)

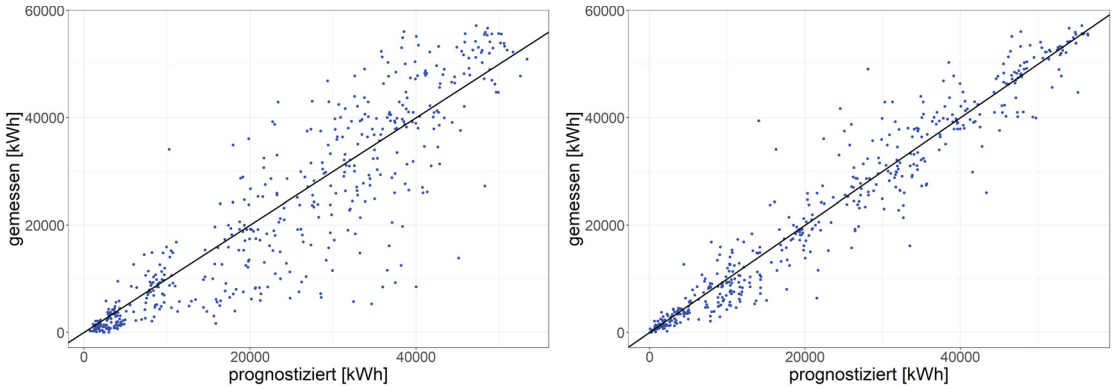

Abbildung 13: Anlage S: prognostizierte vs. gemessene Leistung (links: Datensatz 1, LSTM Modell; rechts: Datensatz 2, Random Forest Modell)

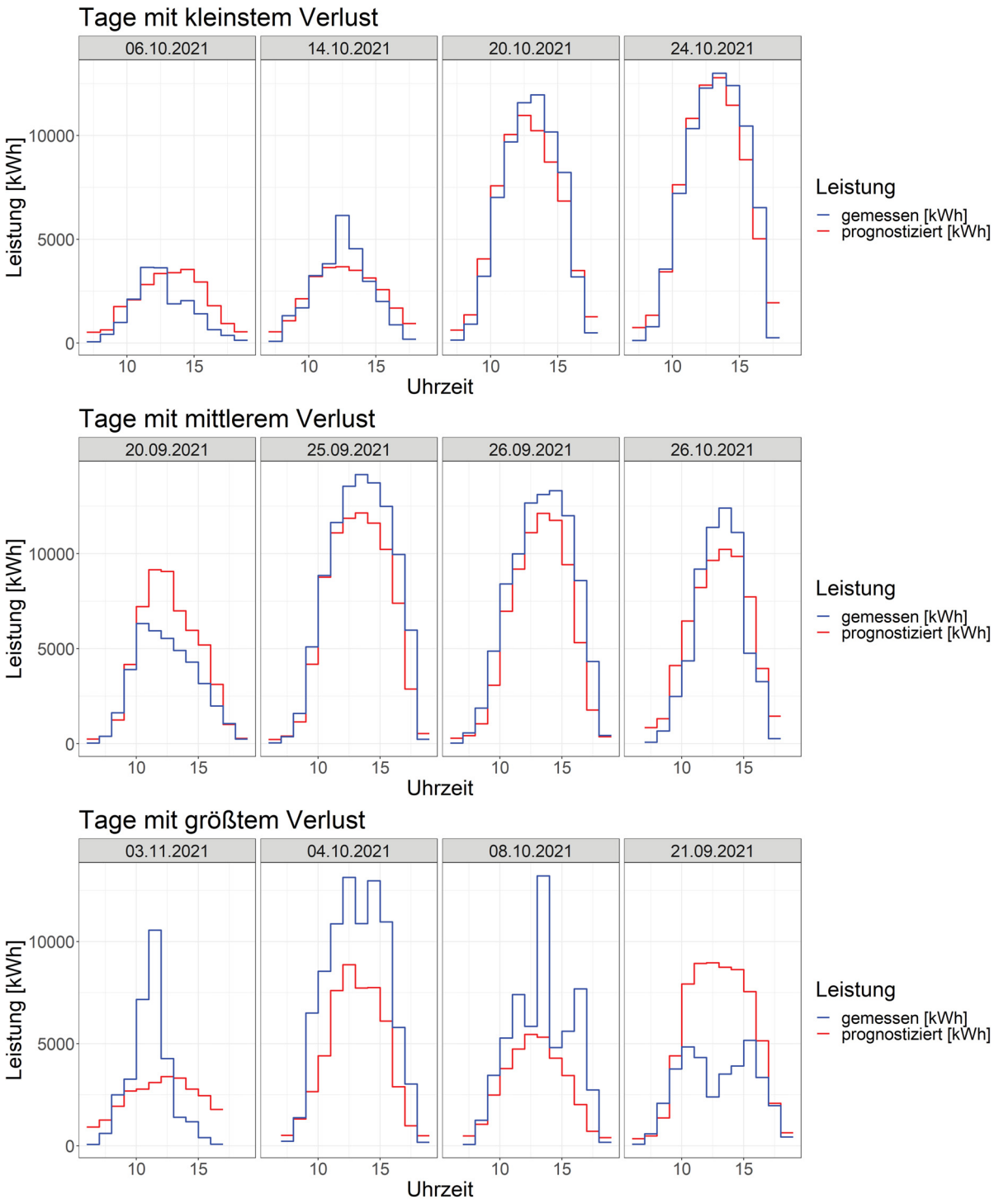

Abbildung 14: Gemessene und prognostizierte Leistung verschiedener Tage (Anlage K, Datensatz 1), Modell: LSTM

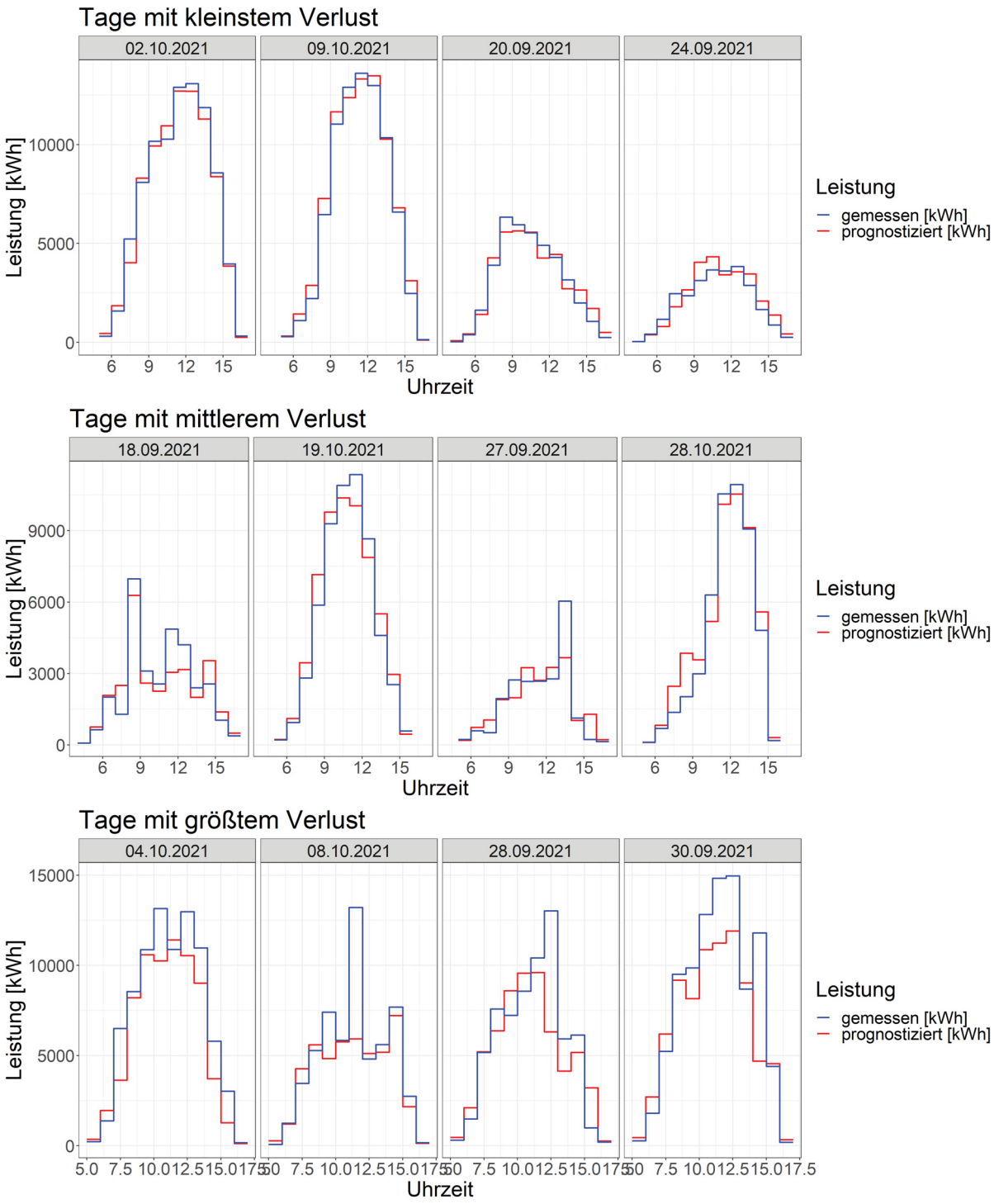

Abbildung 15: Gemessene und prognostizierte Leistung verschiedener Tage (Anlage K, Datensatz 2), Modell: RF

6 Fazit

In dieser Arbeit wurden zwei Datensätze mit Wetterprognosen auf ihre Tauglichkeit zur Vorhersage der Stromproduktion zweier PV-Anlagen in Großschönau (Niederösterreich), untersucht. Dazu wurden verschiedene Machine Learning Modelle implementiert und verglichen. Mit Hilfe von Maßen und Abbildungen wurden die Daten der Wettervorhersagen, sowie die damit erstellten Prognosen analysiert und grafisch visualisiert.

Während der erste Datensatz reine Wetterprognosen, sowie mit der Sonnengeometrie berechnete Solareinstrahlungsdaten (bei wolkenlosem Himmel) enthielt, bestand der zweite Datensatz, neben Wettervorhersagen, zusätzlich aus Prognosen der tatsächlichen Sonneneinstrahlung. Im Vergleich lieferte insbesondere Datensatz 2 deutlich bessere Prognosen bei einem nRMSE von 19,81 % und 23,83 %.

Bei der Gegenüberstellung der Machine Learning Modelle hoben sich insbesondere LSTM Networks bei Datensatz 1 mit dem niedrigsten Verlust hervor. Bei Datensatz 2 lieferte ein Random Forest die besten Ergebnisse. Ein einfaches gewichtetes Ensemble Modell aus allen ML Modellen konnte keine Verbesserung aufweisen.

Zusätzlich wurde eine Analyse der Rolle der einzelnen Wetter Features betrachtet. Bei Datensatz 1 zeigte sich, dass insbesondere die Luftfeuchtigkeit und Bewölkung eine wichtige Rolle unter den Wetterbedingungen spielt. Bei Datensatz 2 spielten die Wetterbedingungen als eigene Input Features kaum eine Rolle, da sie bereits in der Variable der Sonneneinstrahlung eingepreist sind.

Eine grafische Aufbereitung der Tage mit besonders viel, mittlerem und besonders niedrigem Verlust zeigte, dass der Verlauf der Leistungskurve recht gut zu prognostizieren ist, die exakten Werte können jedoch in Einzelfällen etwas stärker abweichen. Diese Abweichungen waren besonders bei Datensatz 1 deutlicher und häufiger zu beobachten.

Abschließend ist zu erwähnen, dass eine Möglichkeit an diese Arbeit anzuknüpfen ist, kürzere Prognoseintervalle als eine Stunde zu betrachten. Zum Beispiel bietet Solcast Intervalle von bis zu 5 Minuten an. Weiterhin könnte eine aufbauende Arbeit das Einbauen von Fehlern in die Wetterdaten beinhalten, um die Unsicherheit zu simulieren, die in der Regel steigt je früher die Wettervorhersage getätigt wird.

Literatur

Ahmed, R., Sreeram, V., Mishra, Y. & Arif, M. (2020). A review and evaluation of the state-of-the-art in PV solar power forecasting: Techniques and optimization. *Renewable and Sustainable Energy Reviews*, *124*. https://doi.org/https://doi.org/10.1016/j.rser.2020.109792

BMK. (2021). *Bundesministerium für Klimaschutz, Umwelt, Energie, Mobilität, Innovation und Technologie (BMK): Energie in Österreich.* https://www.bmk.gv.at/dam/jcr:bbe5cd73-a161-46fc-8c80-2eb5fc500acb/Energie_in_OE2021_UA.pdf

Gigoni, L., Betti, A., Crisostomi, E., Franco, A., Tucci, M., Bizzarri, F. & Mucci, D. (2018). Day-Ahead Hourly Forecasting of Power Generation From Photovoltaic Plants. *IEEE Transactions on Sustainable Energy*, *9*(2), 831–842. https://doi.org/10.1109/TSTE.2017.2762435

OpenWeather. (n. d.). *Weather forecasts, nowcasts and history in fast and elegant way.* https://openweathermap.org/accuracy-and-quality (letzter Zugriff: 07.05.2022)

Solcast. (2019). *Global solar irradiance data and PV system power output data.* https://solcast.com/ (letzter Zugriff: 07.05.2022)

R Core Team. (2013). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing. Vienna, Austria. http://www.R-project.org/

Sun, X., Bright, J. M., Gueymard, C. A., Acord, B., Wang, P. & Engerer, N. A. (2019). Worldwide performance assessment of 75 global clear-sky irradiance models using Principal Component Analysis. *Renewable and Sustainable Energy Reviews*, *111*, 550–570. https://doi.org/https://doi.org/10.1016/j.rser.2019.04.006

Rigollier, C., Bauer, O. & Wald, L. (2000). On the clear sky model of the ESRA — European Solar Radiation Atlas — with respect to the heliosat method. *Solar Energy*, *68*(1), 33–48. https://doi.org/https://doi.org/10.1016/S0038-092X(99)00055-9

Ineichen, P. & Perez, R. (2002). A new airmass independent formulation for the Linke turbidity coefficient. *Solar Energy*, *73*(3), 151–157. https://doi.org/https://doi.org/10.1016/S0038-092X(02)00045-2

SoDa. (2010). *Linke turbidity (Tl) factor worldwide.* https://www.soda-pro.com/help/general-knowledge/linke-turbidity-factor (letzter Zugriff: 01.04.2022)

Yang, D. (2018). SolarData: An R package for easy access of publicly available solar datasets. *Solar Energy*, *171*, A3–A12. https://doi.org/https://doi.org/10.1016/j.solener.2018.06.107

Bright, J. (2019). Solcast: Validation of a satellite-derived solar irradiance dataset. *Solar Energy*, *189*, 435–449. https://doi.org/10.1016/j.solener.2019.07.086

Breimann, L. (2001). Random Forests. *Machine Learning*, *45*, 5–32. https://doi.org/https://doi.org/10.1023/A:1010933404324

Hastie, T., Tibshirani, R. & Friedmann, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Second Edition.* Springer.

Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep Learning* [http://www.deeplearningbook.org]. MIT Press.

Chollet, F. & Allaire, J. (2018). *Deep Learning with R*. Manning.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, *15*(56), 1929–1958. http://jmlr.org/papers/v15/srivastava14a.html

Kumari, P. & Toshniwal, D. (2021). Deep learning models for solar irradiance forecasting: A comprehensive review. *Journal of Cleaner Production*, *318*, 128566. https://doi.org/10.1016/j.jclepro.2021.128566

Graves, A. (2012). *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer. https://doi.org/10.1007/978-3-642-24797-2

Hochreiter, S. & Schmidhuber, J. (1997). Long Short-term Memory. *Neural computation*, *9*, 1735–80. https://doi.org/10.1162/neco.1997.9.8.1735

Zhang, J., Florita, A., Hodge, B.-M., Lu, S., Hamann, H. F., Banunarayanan, V. & Brockway, A. M. (2015). A suite of metrics for assessing the performance of solar power forecasting. *Solar Energy*, *111*, 157–175. https://doi.org/https://doi.org/10.1016/j.solener.2014.10.016

Wright, M. N. & Ziegler, A. (2017). *ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R*. https://doi.org/10.18637/jss.v077.i01

Lang, M., Binder, M., Richter, J., Schratz, P., Pfisterer, F., Coors, S., Au, Q., Casalicchio, G., Kotthoff, L. & Bischl, B. (2019). mlr3: A modern object-oriented machine learning framework in R. *Journal of Open Source Software*. https://doi.org/10.21105/joss.01903

Chollet, F., Allaire, J. et al. (2017). R Interface to Keras.

Allaire, J., Eddelbuettel, D., Golding, N. & Tang, Y. (2016). *tensorflow: R Interface to TensorFlow*. https://github.com/rstudio/tensorflow

Becker, M., Binder, M., Bischl, B., Foss, N., Kotthoff, L., Lan, M., Pfisterer, F., Reich, N., Richter, J., Schratz, P., Sonabend, R. & Pulatovoo, D. (2022). *mlr3book*. https://mlr3book.mlr-org.com

Molnar, C., Bischl, B. & Casalicchio, G. (2018). iml: An R package for Interpretable Machine Learning. *JOSS*, *3*(26), 786. https://doi.org/10.21105/joss.00786

Abstract

Die Anzahl der Photovoltaikanlagen weltweit und auch in Österreich ist stetig am wachsen. Die aufgrund des Sonnenstands und der Wetterbedingungen stark schwankende Leistung der Anlagen motiviert dazu, die Prognose Möglichkeiten dieser zu untersuchen. Insbesondere die Verwendung von Wettervorhersagen und die Anwendung von Machine Learning Modellen gelten als vielversprechend. In dieser Arbeit werden verschiedene Machine Learning Ansätze genutzt, um die stündliche Stromproduktion zweier PV-Anlagen in Niederösterreich zu prognostizieren. Als Eingabe in die Modelle werden dazu zwei Datensätze aus unterschiedlichen Quellen genutzt. Ersterer enthält einfache stündliche Wetterdaten, sowie zusätzlich sonnengeometrische Daten. Der zweite Datensatz enthält neben Wetterbedingungen spezielle Prognosen zur Sonneneinstrahlung. Beide Datensätze und Quellen, sowie die Funktionsweise der Machine Learning Modelle, werden detailliert vorgestellt. Eine genaue Analyse und grafische Darstellung der Prognosefähigkeiten der einzelnen Modelle erfolgt. Insbesondere LSTM Neuronale Netze, sowie Random Forest Modelle, heben sich mit dem geringsten Verlust hervor. Ein Vergleich der Ergebnisse zeigt weiterhin, dass der zweite Datensatz mit Prognosen zur Sonneneinstrahlung deutlich genauere Vorhersagen ermöglicht