Europaschule Gymnasium Stephaneum,
Dr.-W.Külz-Platz 16,
06449 Aschersleben

Besondere Lernleistung in Informatik

Jahrgangsstufe 12

2023/2024

Betreuender Lehrer: Herr Czech

Machine Learning zum Analysieren, Vorhersagen und Darstellen von Wetterdaten

Leon A. Hofmann
Am Holzanger 11
06466 Gatersleben
leon.hofmann1510@gmail.com

Tim Schrader
Heinrich-Grauthoff-Straße 32
39439 Güsten
t.schrader2604@gmail.com

15.04.2024

Inhaltsverzeichnis

1. Einleitung

1.1 Motivation und Danksagung

Als engagierte Schüler, die ihre Leidenschaft für Mathematik, Informatik und Technik über den regulären Informatikunterricht hinaus vertiefen, haben wir in dieser Besonderen Lernleistung nicht nur unser Wissen erweitert, sondern auch wertvolle Erfahrungen gesammelt. Unser gemeinsames Ziel, eine Karriere in den Bereichen Mathematik und Informatik einzuschlagen, wurde durch dieses Projekt weiter gestärkt. Die Anwendung von Machine Learning-Techniken auf Wetterdaten hat uns nicht nur vor spannende Herausforderungen gestellt, sondern uns auch die Möglichkeit gegeben, unsere Fähigkeiten in der praktischen Anwendung zu festigen.

Die Analyse und Vorhersage von Wetterdaten erwies sich als äußerst anspruchsvoll, jedoch haben wir durch unsere gemeinsame Anstrengung Lösungen entwickelt und deutlich Fortschritte erzielt. Dies hat nicht nur unsere technischen Fähigkeiten verbessert, sondern auch unsere Teamarbeit und Problemlösungsfähigkeiten gestärkt.

Wir möchten an dieser Stelle einen besonderen Dank an Dr. Stefanie Hollborn vom Deutschen Wetterdienst, Prof. Dr. Fabian Transchel und Oliver Otto von der Hochschule Harz und unserem Informatiklehrer Herr Bernd Czech aussprechen. Wir bedanken uns ebenfalls für die Kooperation vom OpenWeather-Team, die uns wertvolle Wetterdaten zur Verfügung gestellt haben. Die Unterstützung und Expertise hat es uns ermöglicht, noch größere Fortschritte zu erzielen. Die gewonnenen Erkenntnisse und Fähigkeiten werden uns zweifellos auf unserem weiteren Weg von großem Nutzen sein.

Abschließend möchten wir betonen, dass wir die Besondere Lernleistung als eine wertvolle Gelegenheit betrachtet haben, unsere eigenen Grenzen auszutesten und uns weiterzuentwickeln. Es hat uns nicht nur fachlich, sondern auch persönlich bereichert, und wir sind stolz darauf, dass die Wetterstation und unsere Applikation über unsere Zeit beim Stephaneum hinaus in Verwendung bleiben wird.

1.2 Einführung

Machine Learning (gen. ML) ist ein Teilbereich der Künstlichen Intelligenz (gen. KI), der sich darauf konzentriert, Algorithmen und Modelle zu entwickeln. Diese ermöglichen es Computern, Muster in Daten zu erkennen und daraus zu lernen, ohne explizit programmiert zu werden. Im Wesentlichen geht es darum, Computern beizubringen, Aufgaben zu erfüllen,

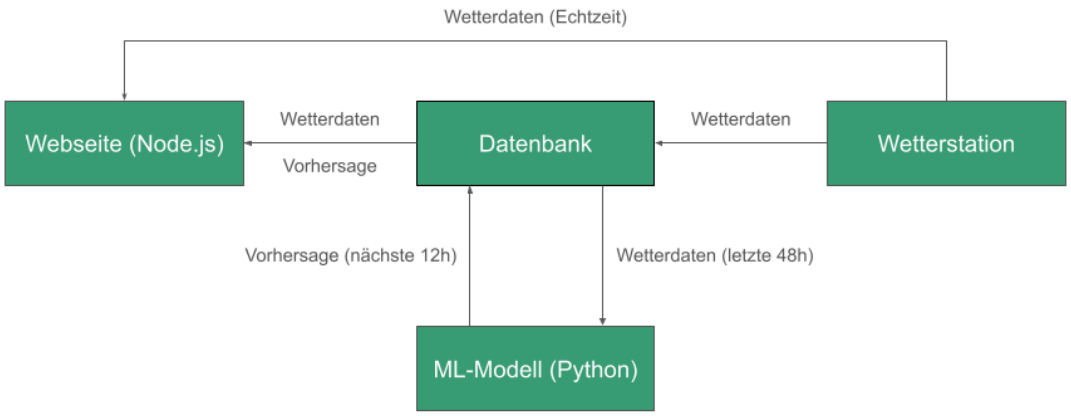
indem sie Erfahrungen aus Daten ziehen. Die jeweilige Vorgehensweise, um eine Aufgabe zu lösen, kann in drei Hauptkategorien differenziert werden. Beim *Überwachten Lernen* werden Trainingsdaten bereitgestellt, mit denen das Modell lernt, eine Zuordnung zwischen Eingaben und den entsprechenden Ausgaben herzustellen. Das Ziel besteht darin, dass das Modell in der Lage ist, für neue, bisher ungesehene Daten genaue Vorhersagen oder Klassifikationen zu machen. Im *Unüberwachten Lernen* erhält der Algorithmus keine vorgegebenen Ausgabewerte. Stattdessen erkennt er selbst Muster und Strukturen in den Daten. Das Ziel ist es, verborgene Beziehungen und Gruppierungen in den Daten zu identifizieren. *Bestärkendes Lernen* bezieht sich auf ein System, das durch positive oder negative Bewertung von Aktionen (Belohnungen oder Bestrafungen) lernt, welche Handlungen vorteilhaft sind. Das Modell trifft Entscheidungen, um eine bestimmte Belohnung zu maximieren und lernt dabei durch den Versuch und die entstehenden Fehlversuche. Machine Learning spielt eine große Rolle in der zukunftsorientierten Forschung und schon jetzt ist es in den Alltag von Millionen Menschen integriert. Die erfolgreichsten Anwendungsbereiche sind personalisierte Dienstleistungen, Finanzdienstleistungen, Bild- und Videoerkennung, Landwirtschaft und Sprach- und Textverarbeitung. [1]

Die Arbeit befasst sich mit dem Verwenden von ML zum Analysieren, Vorhersagen und Darstellen von Wetterdaten. Angesichts der zunehmenden Komplexität und Menge von meteorologischen Informationen ist es entscheidend, fortschrittliche Algorithmen einzusetzen, um präzisere Vorhersagen zu ermöglichen und komplexe Zusammenhänge im Wettergeschehen besser zu verstehen. Das Hauptziel unseres Forschungsprojekts besteht darin, eine fortschrittliche und zuverlässige Plattform für Wettervorhersagen am Schulstandort mittels ML und Echtzeitmessungen zu entwickeln.

2. Theoretischer Teil

2.1 Projektstruktur

Das entwickelte System folgt einer klaren Struktur. Es besteht aus vier wesentlichen Komponenten, welche alle essentiell sind, für das Funktionieren der gesamten Applikation. Die Webseite, Datenbank, Wetterstation und das ML-Modell.



Das Modell nutzt die letzten 48 Stunden an Daten aus der Datenbank, um die nächsten 12 Stunden vorherzusagen. In der Datenbank werden sowohl die aufgezeichneten Daten der Wetterstation als auch die Vorhersagen des Modells gespeichert. Die Daten[1] der Station werden stündlich gesichert, was bedeutet, dass täglich 24 Datensätze erstellt werden. Jeder Datensatz enthält die 13 Parameter des aktuellen Wetters zum Zeitpunkt der vollen Stunde (Tabelle unten). Diese gespeicherten Daten werden verwendet, um sie schließlich auf der Webseite anzuzeigen. Es werden auch Echtzeitdaten direkt von der Wetterstation auf der Webseite angezeigt.

Parameter	**Datentyp**	**Einheit**	**Wird vorhergesagt**
Strahlung	Ganzzahl	w/m²	Nein
UV-Index	Ganzzahl	1	Nein
Luftfeuchtigkeit	Ganzzahl	%	Ja
Temperatur	Dezimalzahl	Celsius	Ja

[1] Die Daten der Wetterstation werden durch den Drittanbieter *wunderground.com* gesendet

gefühlte Temperatur	Dezimalzahl	Celsius	Nein
Tautemperatur	Dezimalzahl	Celsius	Nein
Windrichtung	Ganzzahl	km/h	Nein
Windgeschwindigkeit	Dezimalzahl	km/h	Ja
Windkühle	Dezimalzahl	Celsius	Nein
Böengeschwindigkeit	Dezimalzahl	km/h	Nein
Luftdruck	Dezimalzahl	Hektopascal	Ja
Niederschlag (24h)	Dezimalzahl	mm	Ja
Niederschlag (aktuelle Stunde)	Dezimalzahl	mm	Nein

2.3 Grundlagen überwachtes Lernen

Überwachtes Lernen ist eine Unterkategorie des maschinellen Lernens und der künstlichen Intelligenz. Es werden Datensätze verwendet, um Algorithmen zu trainieren. Diese Algorithmen versuchen Muster in den Daten zu erkennen und zukünftige Eingaben korrekt zu klassifizieren oder vorherzusagen. Dies geschieht, indem das Modell seine Gewichtungen anpasst.

Daten

Die verwendeten Daten haben einen entscheidenden Einfluss auf die Leistungsfähigkeit und Genauigkeit des Modells. Deshalb müssen diese mit Bedacht ausgewählt werden. Die Daten enthalten immer *Features*[2] und *Targets*[3]. Ein Feature ist eine charakteristische Eigenschaft oder ein Attribut eines Datenpunktes, das dazu dient, Vorhersagen oder Klassifizierungen zu ermöglichen. Beispiele für Features könnten sein: Temperatur, Alter, Geschlecht oder andere messbare oder kategorische Merkmale, die relevant für das zu lösende Problem sind. Das Target oder auch die Zielvariable ist die zu prognostizierende oder zu klassifizierende Variable. Die Datensätze werden zusätzlich in ein Trainings- und ein Testset aufgeteilt. Das Trainingsset wird benutzt, um das Modell zu trainieren, wobei das Testset für die Validierung essentiell ist. Denn mit diesem kann die Genauigkeit festgestellt werden. Das Trainingsset enthält bereits korrekte Paare von Features und Targets. Dadurch wird es dem Modell ermöglicht, mit der Zeit zu lernen. Die Genauigkeit wird während des

[2] Features werden auch als x-Werte bezeichnet
[3] Targets werden auch als y-Werte bezeichnet

Lernprozesses mit einer sogenannten Verlustfunktion berechnet, dadurch passt es sich an, bis der Fehler so klein wie möglich ist.

Klassifizierung und Regression

Es gibt zwei hauptsächliche Methoden, in die sich das überwachte Lernen einteilen lässt: Die Klassifizierung und die Regression.

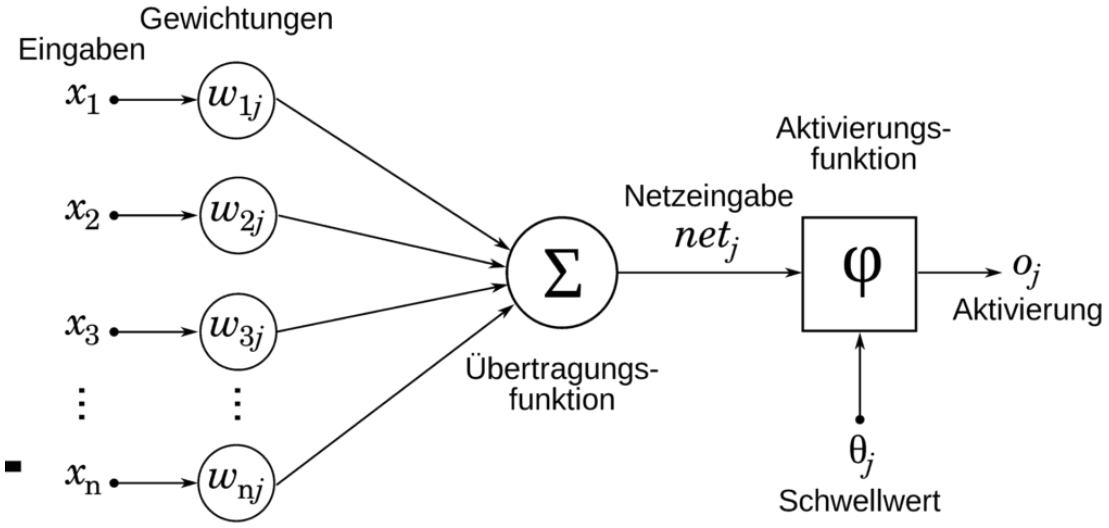
Die Klassifizierung ist eine Methode, die dazu dient, Daten in vordefinierte Kategorien einzuteilen. Ein Klassifizierungsproblem ist beispielsweise die Spam-Erkennung in E-Mails. In diesem Fall handelt es sich also entweder um Spam oder nicht. Die Genauigkeit ist dabei leicht zu bestimmen (siehe unten).

$$\text{Genauigkeit} = \frac{\text{Anzahl korrekter Vorhersagen}}{\text{Gesamtanzahl der Vorhersagen}} \times 100\%$$

Regression ist wiederum ein Lernverfahren, das zur Vorhersage kontinuierlicher Werte verwendet wird. Das Ziel besteht darin, die Beziehung zwischen den unabhängigen Variablen und der abhängigen Variablen zu modellieren, um genaue Vorhersagen zu generieren. Eine typische Verlustfunktion für Regressionsmodelle ist der mittlere quadratische Fehler, der die Qualität der Anpassung des Modells an die Daten misst. [2] [3]

Neuronales Netz

Es handelt sich bei der Vorhersage von Wetterparametern um ein Regressionsproblem. Es gibt verschiedene Möglichkeiten, ein solches Problem zu lösen. Das bekannteste Möglichkeit ist das Neuronale Netz. Ein Neuronales Netz versucht die Vernetzung des menschlichen Gehirns nachzubilden. Es besteht aus Neuronen, die Eingaben, Gewichte und Schwellenwerte verarbeiten.

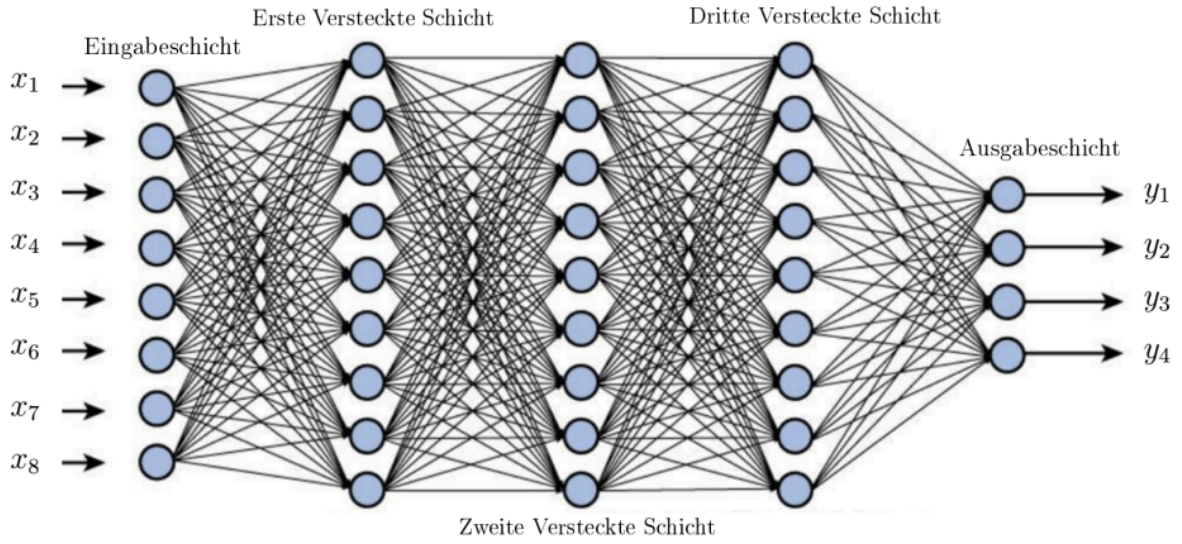


https://datasolut.com/wp-content/uploads/2019/11/neuronale_netze_aktivierung-1024x485.png (Zugriff am 15.04.2024)

Neuronale Netze lernen beispielsweise mit dem Gradientenabstieg, indem sie die Verlustfunktion minimieren. Dabei werden die Gewichte der Netze durch Iterationen angepasst, um die Vorhersagen des Modells kontinuierlich zu verbessern. Zunächst wird der Verlust, also die Differenz zwischen den Modellvorhersagen und den tatsächlichen Werten, für einen Teil eines Trainingssets ermittelt. Anschließend wird der Gradient der Verlustfunktion bezüglich der Gewichte berechnet. Dieser Gradient zeigt die Richtung an, in der die Gewichte geändert werden müssen, um den Verlust zu verringern. Die Gewichte werden dann entlang des Gradienten in kleinen Schritten angepasst, um den Verlust zu reduzieren. Dieser Prozess wird wiederholt, bis ein vordefiniertes Stoppkriterium erreicht ist. Der Gradientenabstieg ermöglicht es dem neuronalen Netz, die optimalen Gewichte zu finden, die die besten Vorhersagen für die gegebenen Daten liefern.

Zusätzlich gibt es für jedes Neuron eine Aktivierungsfunktion. Sie bestimmt, ob und in welchem Maße ein Neuron aktiviert wird und wie stark es auf eingehende Werte reagiert. Die Aktivierungsfunktion nimmt die gewichteten Eingaben eines Neurons zusammen mit einem Schwellenwert als Eingabe und erzeugt eine Ausgabe, die anzeigt, ob das Neuron aktiviert wird. Die Aktivierungsfunktion sorgt dafür, dass ein Netz nicht linear ist und ermöglicht es dadurch, komplexe Muster in den Daten zu erfassen. Bekannte Aktivierungsfunktionen sind die Sigmoid-Funktion, die ReLU Funktion (Rectified Linear Unit), die Tanh Funktion (Tangens hyperbolicus) und die Softmax Funktion, die je nach Anwendung und Netzwerkarchitektur verwendet werden. [4]

Ein einfaches neuronales Netz kann zum Beispiel wie folgt aussehen.



https://artemoppermann.com/wp-content/uploads/2021/08/image-199.png (Zugriff am 15.04.2024)

Aktivierungsfunktionen

Aktivierungsfunktionen werden verwendet, damit neuronale Netze nicht linear sind. In der Regel erfolgt in einem Neuron nach der Berechnung mit Gewichtungen und Schwellwerten eine Aktivierungsfunktion. Diese erfüllen verschieden Zwecke, eine häufig verwendete Aktivierungsfunktion ist die ReLU Funkion, bei der alle Werte unter null gleich null sind. Die Funktion sieht denach folgendermaßen aus: f (x) = max(0,x). Wir verwenden im Neuron des LSTM Modells die Sigmoid und Tanh Funktion. Die Sigmoid Funktion wird verwendet um den Wert zwischen 0 und 1 zu bringen und lautet $f(x) = \frac{1}{1+e^{-x}}$. Die tanh Funktion wird verwendet, um den Wert zwischen -1 und 1 zu bringen und lautet $f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$.

Datenverarbeitung

Wenn ein neuronales Netz effizient und mit einem niedrigen Verlustwert funktionieren soll, müssen die benutzten Daten optimiert werden. Dieser Vorgang wird als Datenverarbeitung bezeichnet. Dabei gibt es verschiedene Methoden, wie die Datensätze verändert werden, damit der Algorithmus ein besseres Ergebnis erzielt.

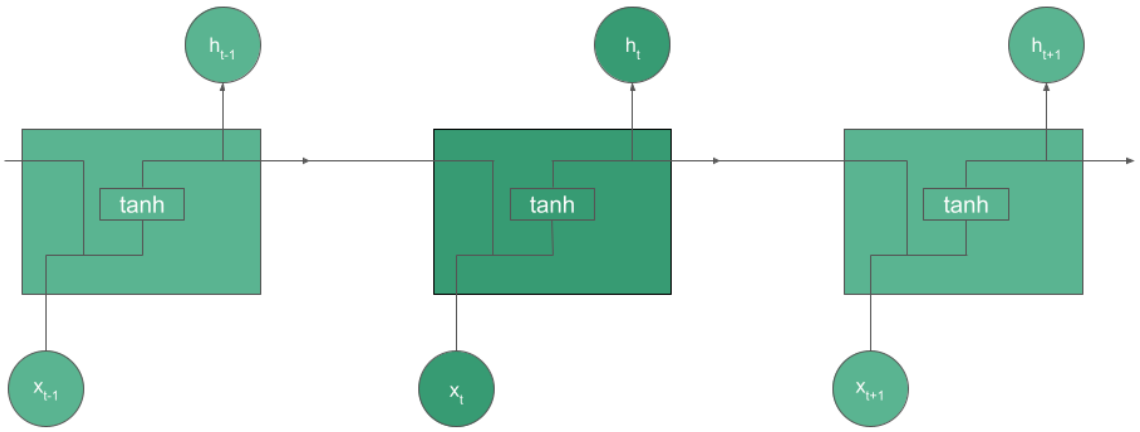
Es kann je nach Anwendungszweck sinnvoll sein, mehrere Features zu kombinieren. Das wird als *Feature Engineering* bezeichnet. Eine Anwendung dafür ist die Umwandlung von Zeitdaten in neue, potenziell aussagekräftigere Features. Angenommen, ein Datenset hat ein Zeitstempel-Feature, das den Zeitpunkt angibt, zu dem eine bestimmte Aktion ausgeführt wurde. Anstatt nur den Zeitstempel zu verwenden, können zusätzliche Features extrahiert werden, die möglicherweise für das Modell relevanter sind. Wie beispielsweise welche

Jahreszeit oder welcher Feiertag zu dem Zeitstempel gehört. Wenn ein Netz mit zu großen Zahlen rechnet, kann das zu Problemen führen. Beim *Feature Scaling* werden die Daten normalisiert. Dann werden die Zahlen meist in einen Bereich zwischen 0 und 1 oder -1 und 1 umgewandelt. Dadurch wird verhindert, dass Zahlen mit einer großen Differenz unterschiedlich hoch gewichtet werden und es dadurch zu verfälschten Ergebnissen kommt.

Essentiell bei der Verarbeitung der Daten ist es ebenfalls dafür zu sorgen, dass ausschließlich relevante Features verwendet werden. Durch das Entfernen von unwichtigen oder redundanten Merkmalen kann die Vorhersage verbessert und der Verlustwert gesenkt werden. Ein weiteres wichtiges Thema sind unvollständige Daten. Wenn Daten fehlen, können verschiedene Techniken angewendet werden, um diese Lücken zu füllen. Zum Beispiel können fehlende numerische Werte durch den Mittelwert ersetzt werden, während fehlende kategoriale Werte durch den häufigsten Wert ersetzt werden können. Außerdem müssen Ausreißer beachtet werden, also einzelne Werte, die stark von der Norm abweichen. Wenn diese unbeachtet bleiben und häufig im Trainingsset vorkommen, kann es ebenfalls zu einem schlechteren Ergebnis führen. [2]

2.4 Modell

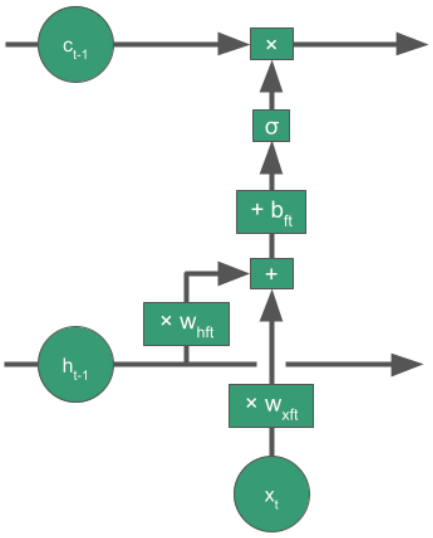
Das verwendete Machine Learning Modell ist eine besondere Form eines neuronalen Netzes. Es handelt sich um ein rekurrentes neuronales Netz. Die Besonderheit in diesem besteht darin, dass innerhalb einer Schicht ein Zusammenhang zwischen den Neuronen geschaffen wird. Ein einfaches, rekurrentes neuronales Netz hat zum Beispiel eine Schicht, in welcher die Neuronen ihre Werte nicht nur an die nächste Schicht weitergeben, sondern auch an das Neuron neben ihnen. In einem Neuron wird also der Input und der Output des vorherigen Neurons mit den dazugehörigen Gewichtungen multipliziert und dann mit dem Schwellwert addiert. Der entstandene Wert wird dann durch die tanh Funktion zwischen -1 und 1 gesetzt.



Modell eines rekurrenten neuronalen Netzes

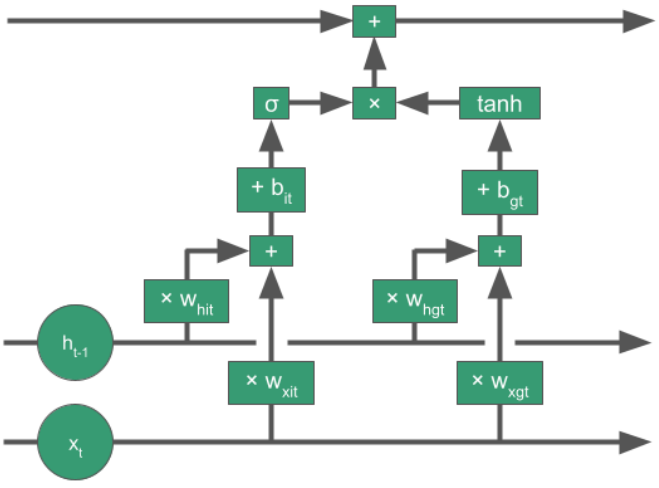
Ein LSTM Modell ist eine erweiterte Version von einem rekurrenten neuronalen Netz. Der Unterschied besteht darin, dass nicht nur die jeweiligen Outputs an das nächste Neuron weitergegeben werden, sondern auch ein Zellstatus. Ein Neuron benutzt demnach zum Berechnen seinen Input x, den Output h und Zellstatus c des vorherigen Neurons. Dabei ist der Output des vorherigen Neurons das Kurzzeitgedächtnis und der Zellstatus das Langzeitgedächtnis

Ein Neuron im LSTM Modell ist zudem in 3 Abschnitte eingeteilt, die alle unterschiedliche Funktionen erfüllen. Im ersten Abschnitt wird bestimmt, wie groß der Anteil ist, der vom Langzeitgedächtnis gemerkt werden soll. Dieser Abschnitt wird als Forget Gate bezeichnet. In diesem Abschnitt wird der Output des vorherigen Neurons als Kurzzeitgedächtnis eingeführt. Dies wird dann mit einer Gewichtung multipliziert und mit dem Produkt des Inputs und dessen Gewichtung addiert. Die entstandene Summe wird dann mit dem Schwellwert addiert und in eine Sigmoidfunktion eingesetzt. Somit entsteht ein Wert, der dann mit dem Langzeitgedächtnis multipliziert wird, um zu bestimmen, wie viel von diesem bestehen bleibt.



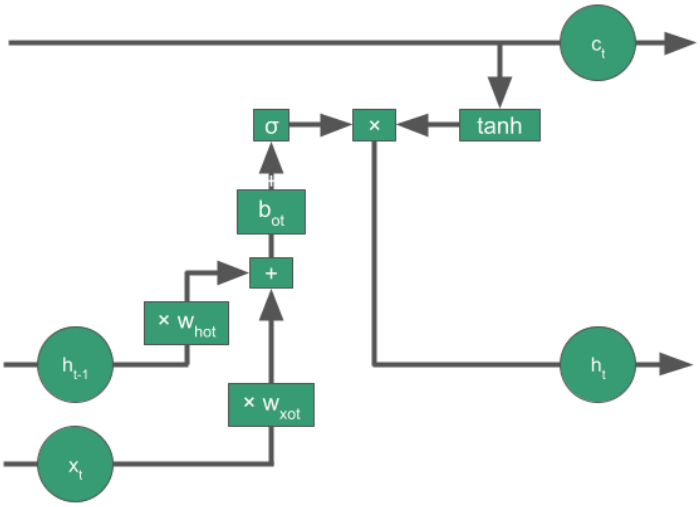
Flussdiagramm des Forget Gates

Im zweiten Abschnitt, auch Input Gate genannt, wird ein potentielles Langzeitgedächtnis erstellt und gleichzeitig berechnet, wie groß dessen Einfluss auf das derzeitige Langzeitgedächtnis ist. Die Berechnung des Einflusses erfolgt wie im Forget Gate. Das potentielle Langzeitgedächtnis wird auf ähnliche Weise berechnet, nur dass statt der Sigmoidfunktion eine tanh Funktion genutzt wird.



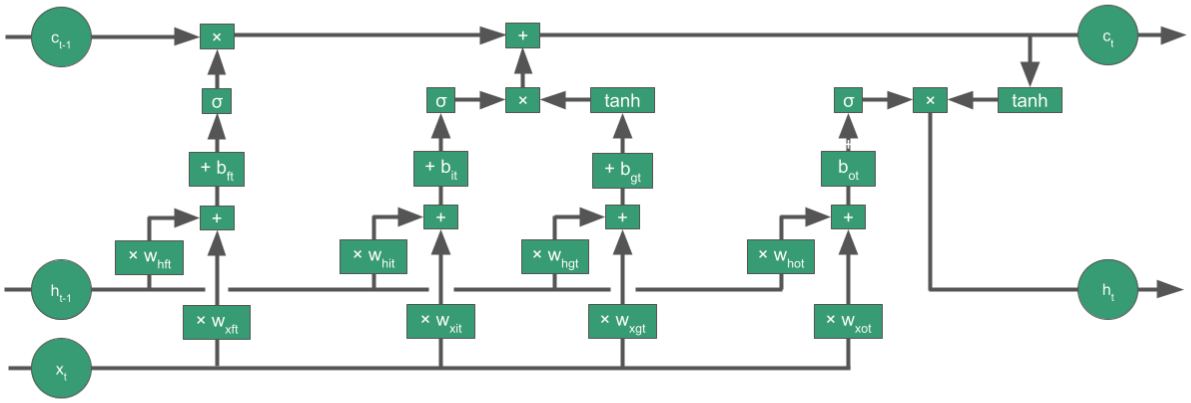
Flussdiagramm des Input Gates

Im letzten Abschnit, dem Output Gate, wird der Wert des Langzeitgedächtins durch eine tanh Funktion zwischen -1 und 1 gebracht. Der Einfluss des entstandenen Wertes erfolgt erneut wie in den anderen Gates. Das Endergebnis ist dann der Output des Neurons und der im Input Gate berechnete Wert für das Langzeitgedächtnis ist dann der neue Zellstatus.



Flussdiagramm des Output Gates

Im gesamten Neuron wird demnach das Langzeitgedächtnis aktualisiert und auf dessen Grundlage der neue Output gebildet.



Flussdiagramm des gesamten LSTM Neurons

3. Methodik

Die methodische Vorgehensweise zur Realisierung dieser Forschungsarbeit lässt sich in mehrere Schritte unterteilen. Zunächst wurden verschiedene Herangehensweisen erkundet, um die optimale Methodik für die vorhandene Problemstellung zu finden. Parallel dazu wurden Gespräche mit Experten aus den Bereichen Wetter und Machine Learning geführt. Diese Expertenmeinungen wurden dokumentiert und fließen als essenzielle Komponente in unsere Besondere Lernleistung ein. Die wichtigsten Erkenntnisse aus diesen Gesprächen wurden herausgearbeitet und spielen eine zentrale Rolle für die Ausrichtung der weiteren Arbeit. Die technische Umsetzung erfolgt durch die Programmierung eines eigenen Systems, das eine Datenbank, eine Webapplikation und mehrere ML-Modelle enthält.

3.1 Wetterstation

https://www.froggit.de/images/product_images/original_images/HP2000_galerie.jpg (Zugriff am 15.04.2024)

Eine Wetterstation wurde großzügig vom Verban ehemaliger Schüler des Stephaneums (VeSt) bereitgestellt. Nach dem Erhalt der Station wurde sie auf dem Dach der Schule installiert und begann, Daten zu sammeln. Dabei wurde ihre Zuverlässigkeit überprüft, indem sichergestellt wurde, dass sie regelmäßig Daten sendet. Um die Qualität der Daten zu überprüfen, wurden die aufgenommenen Daten mit lokalen Wetterdiensten abgeglichen. Zu Beginn traten einige Probleme auf, wie etwa unerwartete Ausfälle der Station und Schwierigkeiten mit der Stromversorgung. Doch nachdem diese Hindernisse überwunden waren, erwies sie sich als äußerst zuverlässig.

Die kontinuierliche Erfassung von Wetterdaten liefert wertvolle Informationen und ohne diese wäre es nur mithilfe von Drittanbietern möglich, Vorhersagen zu machen. Dies gibt der Applikation einen großen Vorteil, weil die Qualität der Daten in der eigenen Hand liegt. Zusätzlich ist es für die Schüler, Lehrer und Eltern vorteilhaft, denn sie wissen mit Hilfe der Echtzeitdaten der Daten zu jeder Zeit genau, wie die Wetterlage am Schulstandort ist. Durch

die sehr gute Lage ist es außerdem möglich, personalisierte Empfehlungen, basierend auf der Vorhersage und Analyse des Wetters, zu erstellen.

3.2 Die ersten Versuche

Zu Beginn des Projekts liegt das Hauptaugenmerk bei der Recherche, wobei besonders das Buch ”Neural Networks from scratch in Python”[4] von , der Kurs “Artificial Intelligence & Machine Learning from scratch”[5] und die App “Brilliant”[6] zur persönlichen Weiterentwicklung beigetragen haben. Dabei besteht die Priorität eine bestmögliche Vorbereitung für Expertengespräche zu schaffen und Anregungen möglichst präzise in Übungsprojekten umzusetzen. Der nächste Schritt besteht darin Machine Learning Algorithmen in Python zu programmieren, die sich vorerst auf andere Datensätze als Wetter konzentrieren. Ein Teil der gewonnenen Erkenntnisse wurden in einem Projekt (weiterhin bezeichnet als Bienenprojekt) verarbeitet. [5]

Bienenprojekt

Um gesammelte theoretische Kenntnisse über Python und Machine Learning in praktischer Umsetzung zu vertiefen, wurde ein Projekt erarbeitet, in welchem die Methoden des unüberwachten Lernens angewendet wurden. Thema des Projektes sind Objekte, dargestellt als Bienen, welche die Aufgabe haben zu lernen, Blumen selbstständig zu finden.

Dies wurde umgesetzt, indem eine Klasse für ein neuronales Netz erstellt wurde. Mit dieser Klasse ist es möglich, eine einzelne Schicht des neuronalen Netzes zu erstellen und die Neuronenanzahl sowie die Aktivierungsfunktion anzupassen. Für jede Biene wurde dann ein einzelnes Netz erstellt, welches sich aus mehreren Schichten zusammensetzt. Zur Berechnung wird NumPy verwendet, welches eine effiziente Lösung zur Berechnung der Skalarprodukte der Input- und Gewichtungsmatrizen liefert.

[4] zu erwerben unter nnfs.io
[5] zu finden auf udemy.com
[6] zu finden auf brilliant.org

```
 import numpy as np

class Layer_Dense:
    def __init__(self, n_inputs, n_neurons, weights=0, biases=0):
        self.n_inputs = n_inputs
        self.n_neurons = n_neurons
        if weights == 0: # Gewichtungs- und Schwellwertarchitektur wird erstellt
            self.weights = np.random.rand(n_inputs, n_neurons) * 2 - 1
        else:
            self.weights = weights
        if biases == 0:
            self.biases = np.random.rand(1, n_neurons) * 2 - 1
        else:
            self.biases = biases
        self.output = None
    def forward(self, inputs): # Outputs der Neuronen werden berechnet
        self.output = np.dot(inputs, self.weights) + self.biases
    def adjust(self, alpha): # Gewichtungen und Schwellwerte werden um den Mutationsfaktor alpha
angepasst
        self.weights = np.minimum(np.maximum(self.weights + alpha * ((np.random.rand(self.n_inputs,
self.n_neurons) * 2 - 1)), -1), 1)
        self.biases = self.biases + alpha * (np.random.rand(1, self.n_neurons) * 2 - 1)

class Activation_ReLU:
    def forward(self, inputs): # ReLU Funktion wird verwendet
        self.output = np.maximum(0, inputs)

class Sigmoid:
    def __init__(self,
                 y_range: float=1,
                 smoothness: float=1): # Parameter für Sigmoid Funktion werden festgelegt
        self.y_range = y_range
        self.smoothness = smoothness
    def forward(self, inputs): # Sigmoid Funktion wird verwendet
        self.output = self.y_range / (1 + np.exp(-self.smoothness * inputs))

class No_Activation:
    def forward(self, inputs): # wenn keine Aktivierungsfunktion vorhanden ist bleiben die Werte
gleich
        self.output = inputs
```

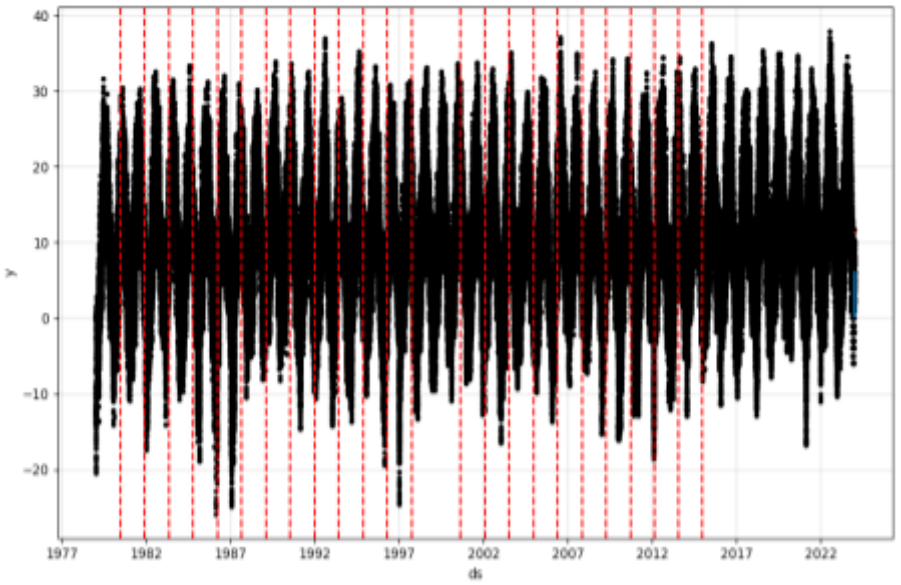
Eine Generation besteht aus einer frei wählbaren Anzahl an Bienen. Im Beispiel werden 100 Bienen pro Generation genutzt (unten rechts, mit Anzahl der gesammelten Blumen pro Biene) Um die Bienen nun lernfähig zu machen, werden die neuronalen Netze von einem ausgewählten Prozentsatz der besten Bienen als Erbmaterial verwendet. Nach einer festgelegten Zeit werden diese dann um einen Mutationsfaktor zufällig verändert und für eine neue Generation genutzt. Die Bienen passen sich also nach dem Prinzip des Darwinismus an.

Screenshot der Bienensimulation

Durch dieses Übungsprojekt und die Programmierung eines vollständig funktionsfähigen neuronalen Netzes können Prozesse, durch die Verbindung zur Praxis, nun besser nachvollzogen werden. Dieses Projekt hat demnach nicht nur motiviert, sondern auch das allgemeine Verständnis deutlich gestärkt.

3.3 Vorhersagen mit Prophet

Bevor überhaupt mit Experten gesprochen wurde, wurde als erstes Metas Prophet als potenzieller Modell erforscht. Prophet ist eine Open-Source-Bibliothek und dient der Generierung von Zeitreihenmodellen, mithilfe von überwachtem Lernen. Es setzt sich aus drei Komponenten zusammen: einer Wachstumsfunktion, einer saisonalen Funktion und einer Event-Funktion. Die Wachstumsfunktion stellt den Trend der Daten dar. Bei dieser wird mit Regression eine lineare Funktion erstellt, welche einen durchschnittlichen konstanten An- oder Abstieg der Daten darstellt. Zusätzlich nutzt Prophet sogenannte Change Points, um diese Daten in Abschnitte zu unterteilen, bei denen der Anstieg stark variiert.



Darstellung der Change Points durch rote Linien am Beispiel der Temperatur

Fourierreihe

Eine Fourierreihe ist eine Funktionsreihe, welche sich aus Sinus- und Kosinusfunktionen zusammensetzt.

$$y = c + \sum_{n=1}^{\infty} a_n cos(nx) + \sum_{n=1}^{\infty} b_n sin(nx)$$

In der Funktion dient die Variable c der Verschiebung entlang der y-Achse. Darauf folgen die Summen aller Sinus- und Kosinusfunktionen. Diese können durch die Variablen a und b gestreckt oder gestaucht werden. Zudem dauert die Periode $\frac{2\pi}{n}$, wobei eine Periode n-mal alle 2π vorkommt. Durch die mit n höher werdende Frequenz und die Möglichkeit der Streckung und Stauchung, kann diese Funktion komplexe Kurven darstellen.

Die Variablen $\widehat{m}$ und $\widehat{b}$ in der linearen Funktion unterscheiden sich zwischen den Change Points. Diese allgemeine Wachstumsfunktion bildet die Basis des Trends und wird durch die zusätzlichen Funktionen erweitert. Die saisonale Funktion nutzt eine Fourierreihe, um eine periodische Änderung der Daten auszudrücken. Dabei ist x dann $\frac{2\pi t}{z}$, wobei t die Zeit darstellt und z die Zeitspanne der Saisonalität. Bei der jährlichen Fourierreihe beträgt die Zeitspanne 365.25, eine Periode dauert demnach $\frac{365.25}{n}$. Bei der jährlichen Fourierreihe sind es 10 Werte, es gibt deshalb 10 Sinus- und 10 Kosinusfunktionen. Somit wird eine Kurve generiert, welche die Werte in Abhängigkeit der Zeit über einen bestimmten Zeitraum darstellt. Dies geschieht jährlich, wöchentlich und täglich.

Vor- und Nachteile

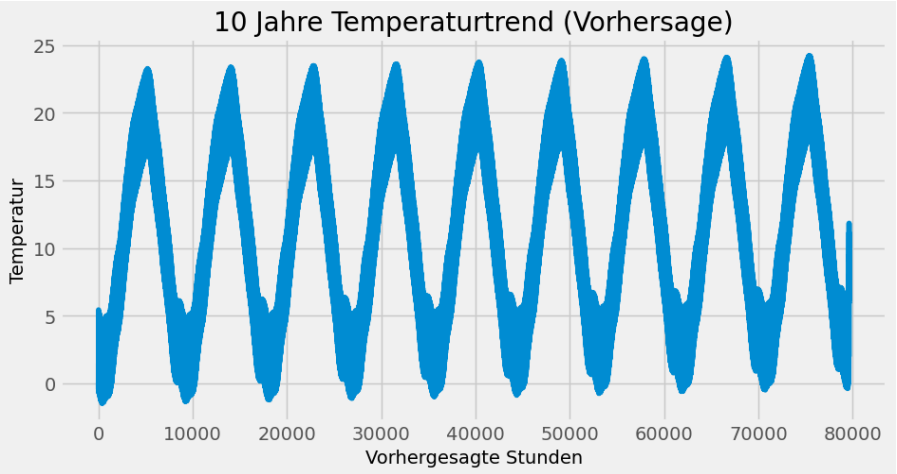
Der Vorteil von Meta's Prophet liegt darin, dass große Mengen an Daten verarbeitet und komplexe Muster identifiziert werden können. Durch die Nutzung von fortschrittlichen Algorithmen kann es somit Vorhersagen über zukünftige Trends treffen. Es erfordert zudem keine tiefe Fachkenntnis in Statistik oder Datenwissenschaften, was es für eine breite Palette von Benutzern zugänglich macht. Prophet kann zudem Ausreißer automatisch erkennen und behandeln, was die Genauigkeit der Vorhersagen verbessern kann. Zusätzlich ist es möglich, Prophet an verschiedene Szenarien anzupassen und es für verschiedene Arten von Zeitreihendaten zu verwenden, einschließlich Finanzdaten, Verkaufsprognosen oder auch Wetterdaten.

Prophet bietet zweifellos eine leistungsstarke Plattform für die Vorhersage von Zeitreihendaten, einschließlich Wetterdaten. Dennoch gibt es einige Nachteile, die bei der Nutzung dieses Tools berücksichtigt werden sollten. Ein wesentlicher Nachteil liegt in den begrenzten Anpassungsmöglichkeiten. Während das Modell flexibel genug ist, um verschiedene Arten von Zeitreihendaten zu behandeln, ist es für spezifische Anwendungsfälle und komplexe Wetterphänomene nicht ausreichend anpassbar. Zusätzlich kann Prophet lokale Effekte und kurzfristige Änderungen nicht ausreichend berücksichtigen, die sich auf das Wetter auswirken können. Dies beeinträchtigt die Genauigkeit der Vorhersagen erheblich.

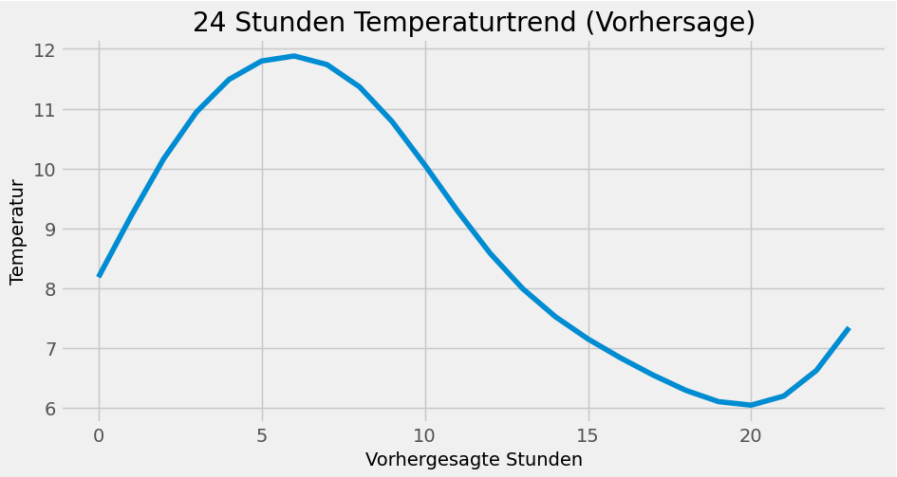
Ergebnisse

Mit Prophet ist es gelungen, nachvollziehbare Ergebnisse zu erzielen. Die eingesetzten Wetterdaten umfassten den Zeitraum von 1979 bis zum jeweils aktuellen Tag. Vor der Anwendung des Prophetmodells wurden die Daten aufbereitet und in das erforderliche Format für die Analyse umgewandelt. In den dadurch generierten Prognosen zeichnen sich klare Muster, saisonale Schwankungen und langfristige Trends ab. Dies belegt die Fähigkeit des Modells, komplexe Zusammenhänge in den vorliegenden Zeitreihendaten aufzudecken. Insbesondere bei saisonalen Mustern in den Temperaturdaten war Prophet in der Lage, den Wechsel der Jahreszeiten präzise vorherzusagen.

Wichtig zu verstehen ist, dass das Modell für jede Vorhersage immer über 350.000 Datensätze einbezieht und wöchentliche sowie tägliche Schwankungen nicht extra betrachtet werden. Die vorhergesagten Temperaturen können sich deshalb auch deutlich von den Vorhersagen der Wetterdiensten unterscheiden, weil der gesamte Zeitraum berechnet wird und somit hauptsächlich langfristige Trends betrachtet werden.



Vorhersage mehrerer Jahre mit Prophet



Beispiel einer Tagesvorhersage mit Prophet

Aus diesen Gründen wurden bessere Lösungsmöglichkeiten gesucht, weil das Ziel war, die Vorhersage so gut wie möglich, gemessen an den wirklichen Gegebenheiten, zu gestalten. Deshalb wurde für die Entwicklung weiterer Modelle die Python Bibliothek TensorFlow verwendet. [6]

3.4 Expertengespräche

Gespräch mit Dr. Stefanie Hollborn

Die Kunst der Wettervorhersage stellt sich als ein hochkomplexer Prozess dar, verdeutlicht Dr. Stefanie Hollborn, Mathematikerin beim Deutschen Wetterdienst. Insbesondere hebt sie hervor, dass präzise Vorhersagen nicht nur von meteorologischen Daten abhängen, sondern auch von einer umfassenden Palette externer Informationen, die in einem globalen Modell integriert werden müssen.

Physikalische Größen wie die Temperatur seien dabei die grundlegenden Eingangsparameter, aber auch Satellitendaten spielen eine entscheidende Rolle in modernen Berechnungen. Das Erstellen eines globalen Modells, das eine Vielzahl von Datenquellen berücksichtigt, sei von essenzieller Bedeutung, um die Vielschichtigkeit atmosphärischer Prozesse entsprechend genau abzubilden.

Interessanterweise diskutiert Dr. Hollborn auch die Herausforderungen bei der Integration von künstlicher Intelligenz in den modernen Vorhersageprozess. Obwohl KI in der Lage ist, statistische Aussagen zu liefern, würden ihre Fähigkeiten in längeren Zeiträumen an Grenzen stoßen. Insbesondere betont sie, dass direkte Daten nur bis zu einem bestimmten Punkt reichen würden und die Beteiligung physikalischer Abläufe für langfristige Vorhersagen unerlässlich sei. Trotz dieser Herausforderungen sieht Dr. Hollborn auch Potenzial für KI, insbesondere in der kurzfristigen Vorhersage durch ein Netzwerk von Wetterstationen. Dabei könnten nicht nur statistische Aussagen abgebildet, sondern auch Empfehlungen, beispielsweise zur Kleidungswahl, abgeleitet werden.

Gespräch mit Prof. Dr. Fabian Transchel

Im Gespräch mit Prof. Dr. Fabian Transchel von der Hochschule Harz stellt sich heraus, dass die Wetterberechnung mit einer einzelnen Wetterstation aufgrund der globalen Natur des Wetters nicht exakt funktionieren könne. Jedoch hat er verschiedene Modellansätze kritisch abgewogen. Als Methode zur Wettervorhersage sieht Prof. Dr. Transchel Convolutional Neural Networks als optimale Wahl. Das ist kurz zusammengefasst ein regulierter Typ eines neuronalen Netzes, das mit Hilfe von Filtern oder Kernel-Optimierung selbständig Feature-Engineering erlernt. Er schlägt vor, mehrere Wetterstationen zu einem Netzwerk zu verbinden, wobei die Gewichtung auf der Nähe zum Hauptstandort basiert. Die Inputvariablen können beispielsweise Temperaturmessungen an verschiedenen Orten und Zeiten sein. Die Entscheidung, ob das Netzwerk am Ende fully-connected[7] ist oder nicht, sowie die Vermeidung von Datenlücken durch Interpolation[8], erfordere experimentelle Überlegungen.

Für die Umsetzung empfiehlt Prof. Dr. Transchel die Verwendung von TensorFlow in Python und Tensorflow.js für die Darstellung. Diese Softwaretools bieten laut ihm eine solide Grundlage für die Implementierung fortschrittlicher Wettervorhersagemodelle.

[7] alle Neuronen sind in aufeinanderfolgenden Schichten miteinander verbunden
[8] Verfahren, bei dem unbekannte Datenpunkte innerhalb eines vorhandenen Datensatzes geschätzt oder berechnet werden, basierend auf bekannten Datenpunkten

Umsetzen der Erkenntnisse der Gespräche

Wettervorhersagen zu erstellen ist ein hochkomplexer Prozess, dem sich große Institutionen wie der Deutsche Wetterdienst täglich widmen. Um daraus ein Projekt zu entwickeln wurde das Wissen mit den Möglichkeiten abgewogen. Es ist nicht möglich, ohne spezielle Hard- und Software extrem komplexe Wetterdaten zu verarbeiten, wie sie etwa Meteorologen verwenden. Hinzu kommt das Problem, dass diese oft nicht frei verfügbar sind. Es kann demzufolge lediglich mit Daten gearbeitet werden, die beispielsweise von einer privaten Wetterstation aufgenommen werden und diese sind in diesem Fall auf Temperatur, Taupunkt, Luftdruck, Feuchtigkeit, Wind und Regen begrenzt. Für die Datenanalyse in diesem Projekt finden nur diese Parameter Verwendung. Es kann also kein globales Modell erstellt werden und somit auch keine exakte Vorhersage, trotzdem können gute Vorhersagen erstellt werden, die den Prognosen der Wetterdienste ähneln. Es ist ebenfalls aus finanziellen Mitteln nicht möglich, ein Netz aus Wetterstationen zu erstellen und damit Vorhersagen zu errechnen, da für jeden Standort Daten gekauft werden müssten. Mit diesem Wissen wurde das weitere Vorgehen geplant.

3.5 ML-Modelle mit TensorFlow

TensorFlow ist eine Open-Source-Bibliothek für maschinelles Lernen und künstliche Intelligenz, die von Google entwickelt wurde. Es bietet eine Plattform für die Erstellung und Ausführung von verschiedenen ML-Modellen. Mit TensorFlow können komplexe Algorithmen für Aufgaben wie Bilderkennung, Sprachverarbeitung, Textanalyse, Zeitreihenprognosen und vieles mehr erstellt werden. Die Bibliothek zeichnet sich durch ihre Skalierbarkeit und Flexibilität aus. TensorFlow bietet auch eine Vielzahl von Tools für die Entwicklung und das Training von Modellen in verschiedenen Programmiersprachen, darunter Python, C++, JavaScript und Swift. Dank der umfangreichen Sammlung von vorgefertigten Modellen kann TensorFlow genutzt werden, um schnell Prototypen zu erstellen und komplexe Anwendungen für maschinelles Lernen und künstliche Intelligenz zu entwickeln. TensorFlow hat sich zu einer der führenden Plattformen im Bereich des maschinellen Lernens entwickelt und wird von einer großen Entwicklergemeinschaft weltweit unterstützt.

Bei der sorgfältigen Auswahl der Tools, die in der Besonderen Lernleistung verwendet werden sollen, wurde TensorFlow einerseits aufgrund der sehr guten Performance ausgewählt. Andererseits bietet das Tool zudem, wie eine Art Baukasten, alle populären modernen Algorithmen zur Arbeit mit neuronalen Netzen. Dadurch ergibt sich der Vorteil, dass der Fokus auf der Entwicklung eines Modells liegt und weniger auf der Entwicklung der Modelle an sich. Dadurch wird das Thema “Machine Learning zum Analysieren, Vorhersagen und Darstellen von Wetterdaten” effizienter untersucht. [7]

3.6 Entwicklung des Backends

Das Backend, also alles was bei der entwickelten Applikation im Hintergrund passiert, wurde mit NodeJS in JavaScript geschrieben. Eine NodeJS-Anwendung besteht in diesem Fall aus mehreren Modulen: Der “server.js”-Datei, den *views*, den *routes* und den *models*.

Die server.js ist die Datei, die ausgeführt wird, um die Applikation zu starten. In dieser wird zuallererst Express importiert. Express ist ein minimalistisches, flexibles Web-Framework für Node.js, das die Erstellung von Webanwendungen und APIs durch eine einfache Routing-Struktur und Middleware-Unterstützung vereinfacht. Essentiell ist dabei die Initialisierung des App-Objektes. Der Body-parser ist notwendig für eine problemfreie Kommunikation bei der Übertragung von REST-Anfragen mit JSON Inhalt.

```
const express = require('express')
const app = express()
const expressLayouts = require("express-ejs-layouts")
const bodyParser = require('body-parser')

app.use(bodyParser.json())
app.use(bodyParser.urlencoded({ extended: true }))
```

Als nächstes werden die Router definiert. In Express ermöglicht ein Router die Organisation von Endpunkten und deren zugehöriger Logik in separaten Modulen, wodurch die Codebasis strukturiert wird. Durch die Zuweisung von Routen zu spezifischen Handlern ermöglicht der Router die effiziente Verarbeitung von HTTP-Anfragen.

```
const indexRouter = require('./routes/index')
const getRouter = require('./routes/get')
const postRouter = require('./routes/post')
app.use('/', indexRouter)
app.use('/get', getRouter)
app.use('/post', postRouter)
```

Als letztes werden in der server.js noch die Applikationseinstellungen eingestellt. Unter anderem in welchem Ordner sich die views befinden, welche View-engine verwendet wird und in welchem Ordner die CSS und weiteren Assetsdateien sind.

```
app.set("view engine", "ejs")
app.set("views", __dirname + "/views")
app.set("layout", "layouts/layout")
app.use(expressLayouts)
app.use(express.static(__dirname+'/public'))

app.use(express.json())
```

In einem View befindet sich die HTML-Struktur der Seite, welche schlussendlich im Browser beim Benutzer angezeigt wird. Das besondere an der verwendeten ejs-Viewengine ist, dass

Informationen vom Backend sehr einfach in das Frontend integriert werden können. Wenn die Seite vom Router geladen wird, werden die Platzhalter durch Variablen ersetzt, wie im folgenden Beispiel. Die Variable kann dabei jeder mögliche Wert sein, wie etwa die Tagestemperatur oder die Regenwahrscheinlichkeit.

```
<!DOCTYPE html>
<html lang="en">
<body>
  <div>
    <h1>Das ist ein Beispieltext</h1>

    <h2>Variable:</h2>
    <p><%= variable %></p>
  </div>
</body>
</html>
```

In der Routerdatei wird die Seite geladen, die der Benutzer aufruft. Alle Variablen, die auf der Seite angezeigt werden sollen, müssen dort übergeben werden. Die get-Funktion übernimmt diese Aufgabe. Dort können zum Beispiel die Variablen aus der Datenbank abgefragt werden und mit res.render wird schließlich die Webseite geladen. [8]

```
const { Route } = require("express")
const express = require("express")
const router = express.Router()

// Laden der Index-Seite
router.get('/', async (req, res) => {
    const temp = await loadFromDatabase('temp')
    res.render('index', { temperatures: temp })
})

module.exports = router
```

In anderen Routerdateien, wie sie in der server.js ebenfalls zu finden sind, befindet sich die REST-API[9]. Das ist eine Schnittstelle für den Zugriff und die Manipulation von Daten über das HTTP-Protokoll. Sie ermöglicht die Interaktion zwischen verschiedenen Systemen oder Anwendungen über standardisierte HTTP-Methoden wie GET, POST, PUT und DELETE, wobei Ressourcen über eindeutige URLs angesprochen werden.

Als letztes sind die Modelle entscheidend, denn sie vereinfachen den Umgang mit Funktionen, die immer wieder gebraucht werden. Ein Beispiel für ein in der Applikation verwendetes Modell ist das Entry-Modell. Ein Entry stellt einen Eintrag in die Datenbank dar und dieser kann durch einen Konstruktur definiert werden, mit allen erforderlichen Werten, welche in die Datenbank eingetragen werden. Also die Daten, die stündlich von der

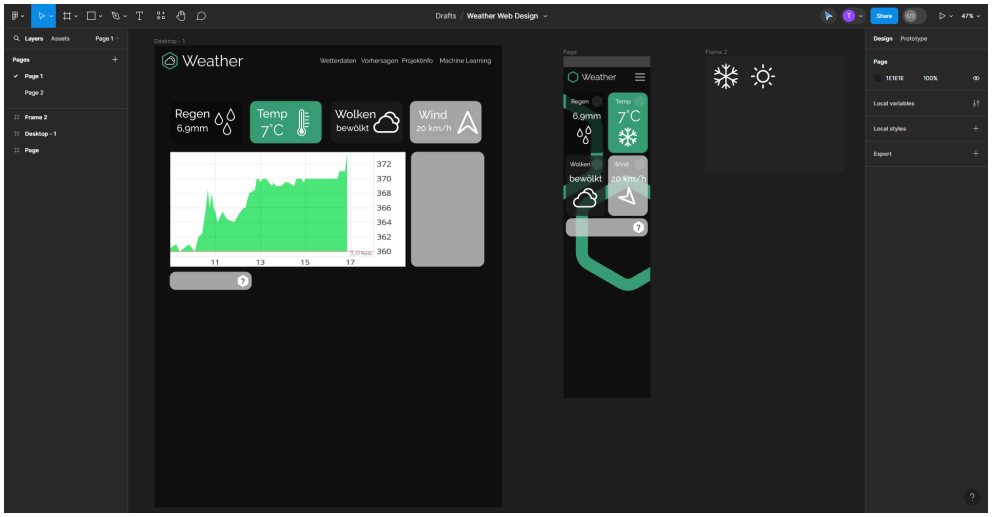
[9] Representational State Transfer

Wetterstation gespeichert werden. Dann kann im Entry-Modell die Funktion genutzt werden, um diesen Eintrag in der Datenbank zu speichern. [9]

Zusammenfassend stellt die Node.js-Anwendung mit Express das Backend für die Applikation dar. Sie besteht aus der Hauptdatei "server.js", in der Express initialisiert wird und Router definiert werden, um Endpunkte und ihre Logik zu organisieren. Die Views enthalten die HTML-Struktur der Seiten und können mit variablen Informationen aus dem Backend über die ejs-Viewengine ergänzt werden. Die Router laden die entsprechenden Seiten und verwalten REST-APIs für den Zugriff und die Manipulation von Daten über das HTTP-Protokoll. Modelle wie das Entry-Modell vereinfachen den Umgang mit Datenbankoperationen und anderen wiederkehrenden Funktionen.

3.7 Entwicklung des Frontends

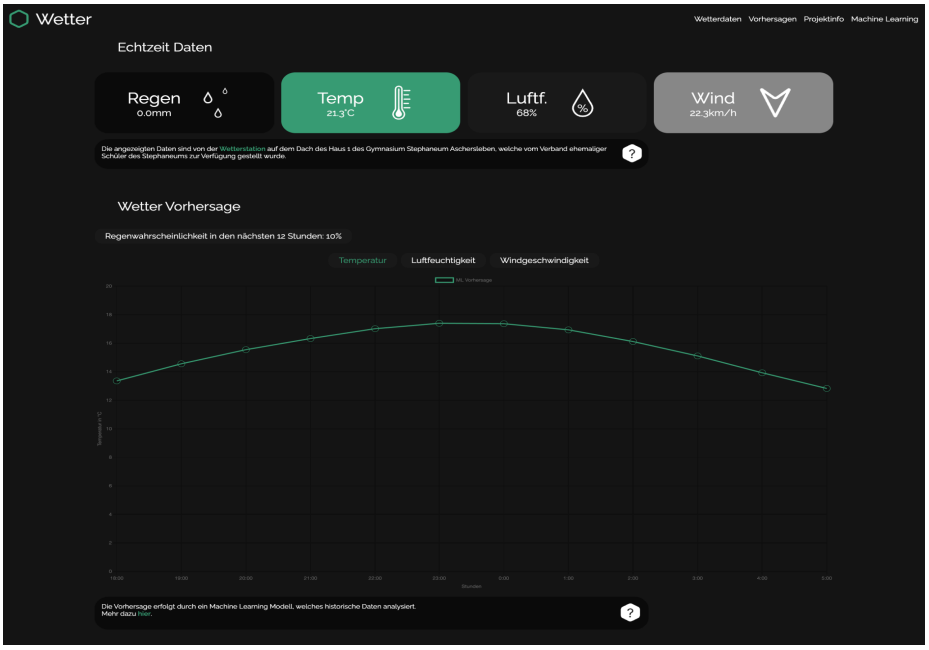
Bevor das Design in CSS programmiert werden kann, muss vorher ein Konzept erstellt werden. Das wurde mit Figma gemacht. Figma ist eine webbasierte Plattform für das Erstellen von Benutzeroberflächen und interaktiven Prototypen. Mit der intuitiven Benutzeroberfläche und seinen kollaborativen Funktionen hat sich das Programm zu einer beliebten Wahl für Webdesigner entwickelt. So wurde wie im Bild zu sehen ein erster Prototyp der Desktop- und Handyversion erstellt. [10]



Screenshot des Prototyps in Figma

Danach wurde zuerst die HTML-Struktur erstellt und das Design mithilfe von CSS umgesetzt. So konnte effektiv eine benutzerfreundliche und schlichte Oberfläche für die Nutzer der Applikation geschaffen werden.

Screenshots von der Handyversion der App



Screenshot von der Desktopversion der App

4. Ergebnisse

4.1 Vergleich der Modelle

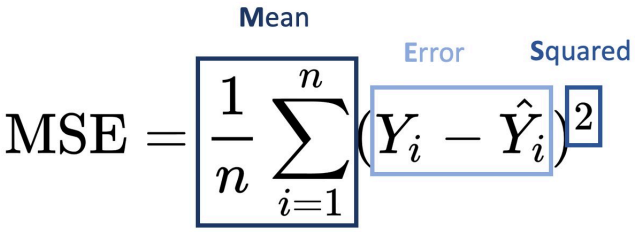
Auf der Webseite soll nur das Beste der erforschten Modelle die Werte berechnen, damit die Vorhersagen so genau wie möglich sind. Bei der Entscheidung, welches Modell für die Applikation verwendet werden soll, wurden alle getesteten Modelle verglichen. Dieser Vergleich ist in der Tabelle dargestellt. Beim Convolutional Neural Network und beim LSTM-Modell wurde für alle Tests ein Array mit den letzten 48 Werte als Features für das Netz verwendet.

Die Vorhersage der Temperatur und der Windgeschwindigkeit funktioniert von allen Werten am besten. Es wurde auch versucht Luftdruck, Luftfeuchtigkeit und stündlichen Niederschlag gut vorauszusagen, aber mit weniger Erfolg. Die Werte von Luftdruck und Luftfeuchtigkeit sind beim LSTM zu weit von der Realität entfernt, um zuverlässig zu funktionieren. Das Vorhersagen des Niederschlags hat stündlich nicht funktioniert und wurde nicht weiter untersucht. Auch das Prophetmodell wurde aufgrund von zu hohen Abweichungen nicht weiter berücksichtigt. Das Convolution Neural Network ist zwar teilweise besser als das LSTM, aber es eignet sich weniger gut für die Vorhersage von Zeitreihen, deshalb wurde bei allen Vorhersagen auf das LSTM gesetzt, auch wenn es leicht hinten liegt.

	Convolutional Neural Network	LSTM
	Temperatur	
Abweichung 12 Stunden	4,45 °C	3,29 °C
Abweichung 24 Stunden	6,11 °C	5,20 °C
Abweichung 48 Stunden	9,42 °C	10,89 °C
	Windgeschwindigkeit	
Abweichung 12 Stunden	0,19 km/h	1,28 km/h
Abweichung 24 Stunden	2,05 km/h	1,56 km/h
Abweichung 48 Stunden	3,86 km /h	2,23 km/h
	Luftdruck	
Abweichung 12 Stunden	50,05 hPa	87,24 hPa
Abweichung 24 Stunden	51,77 hPa	-
Abweichung 48 Stunden	51,86 hPa	-

	Luftfeuchtigkeit	
Abweichung 12 Stunden	6,73 %	-
Abweichung 24 Stunden	7,01 %	-
Abweichung 48 Stunden	9,28 %	-

Die Zahlenwerte in der Tabelle wurden jeweils mit dem sogenannten *Mean Squared Error* berechnet. Der auf Deutsch “mittlere quadratische Fehler” ist eine Metrik zur Bewertung der Genauigkeit eines Vorhersagemodells, indem die durchschnittliche quadratische Abweichung zwischen den tatsächlichen und vorhergesagten Werten berechnet wird. Je niedriger der Wert, desto besser ist die Vorhersageleistung des Modells. Dieser Wert wird folgendermaßen berechnet. [11]



https://suboptimal.wiki/images/mse_5.jpg (Zugriff am 15.04.2024)

4.2 Verwendung im Alltag

Die Entwicklung dieser Applikation zielt darauf ab, jedem Nutzer einen Einblick in die Möglichkeiten von Machine Learning (ML) in kurzer Zeit zu geben. Ein derartiges Projekt hat es an unserer Schule bisher noch nie gegeben und soll dazu dienen, Künstliche Intelligenz für die Schüler greifbarer zu machen und ihr Interesse an diesem zukunftsweisenden Bereich zu wecken. Ein Hauptanwendungsfall dieser Applikation besteht selbstverständlich im Ablesen der Wettervorhersage für die nächsten 12 Stunden. Dabei werden entscheidende Faktoren wie Temperatur, Luftfeuchtigkeit, Regenwahrscheinlichkeit und Windgeschwindigkeit dargestellt. Dies ermöglicht es den Nutzern, sich auf das Wetter vorzubereiten und ihre Aktivitäten und Kleidungswahl entsprechend zu planen.

Die Applikation steht natürlich allen Schülern, Lehrern, Eltern und allen Bewohnern der Stadt Aschersleben zur Verfügung. Durch ihre intuitive Benutzeroberfläche ist sie einfach zu bedienen und bietet einen unmittelbaren Nutzen für den täglichen Gebrauch. Darüber hinaus informiert die Webseite ausführlich darüber, wie das Projekt entstanden ist und wie ML im

Allgemeinen funktioniert. Dies trägt dazu bei, das Verständnis für ML und seine Anwendungen zu verbessern und das Interesse an diesem spannenden Bereich zu fördern.

4.3 Diskussion

Die Besondere Lernleistung basiert auf einem Ansatz, der die Bereiche des maschinellen Lernens und der Meteorologie miteinander verknüpft. Die Methodik zur Entwicklung und Evaluation von Wettervorhersagemodellen wird durch eine Kombination aus Literaturrecherche und Expertengesprächen sowie praktischer Umsetzung in Programmierprojekten gestaltet.

Zu Beginn des Projekts lag der Fokus auf der Recherche, wobei Ressourcen wie das Buch "Neural Networks from scratch in Python", der Kurs "Artificial Intelligence & Machine Learning from scratch" und die App "Brilliant" als wichtige Quellen dienten. Ziel war es, eine solide Grundlage für Expertengespräche zu schaffen und Erkenntnisse präzise in Übungsprojekten umzusetzen. Dabei wurde auch die Programmierung von Machine-Learning-Algorithmen in Python priorisiert, wobei zunächst andere Datensätze als Wetter betrachtet wurden. Ein Teil der gewonnenen Erkenntnisse floss dann in das "Bienenprojekt" ein.

Die Gespräche mit Experten wie Dr. Stefanie Hollborn vom Deutschen Wetterdienst und Prof. Dr. Fabian Transchel von der Hochschule Harz lieferten wertvolle Einblicke in die Komplexität der Wettervorhersage und die verschiedenen Ansätze zur Modellierung. Dr. Hollborn betonte die Bedeutung eines globalen Modells, das eine Vielzahl von Datenquellen integriert. Prof. Dr. Transchel schlug Convolutional Neural Networks als optimale Methode für die Wettervorhersage vor und empfahl die Verwendung von TensorFlow in Python für die Implementierung. Die Umsetzung der Erkenntnisse aus den Expertengesprächen und der Literaturrecherche erfolgte unter Berücksichtigung der vorhandenen Ressourcen und Einschränkungen. Aufgrund der begrenzten Verfügbarkeit und Komplexität von Wetterdaten wurde beschlossen, sich auf Daten von einer einzelnen Wetterstation am Schulstandort zu konzentrieren, die auf Temperatur, Taupunkt, Luftdruck, Feuchtigkeit, Wind und Regen beschränkt ist. Dies ermöglichte zwar keine globale Modellierung, dennoch konnten gute Vorhersagen erstellt werden, die den Prognosen der Wetterdienste ähneln.

Die Implementierung erfolgte erst mit Hilfe von Prophet, einem Modell zur Zeitreihenanalyse, das klare Muster, saisonale Schwankungen und langfristige Trends in den Daten identifizieren konnte. Es ist jedoch wichtig zu beachten, dass das Modell wöchentliche und

tägliche Schwankungen nicht explizit berücksichtigt und sich die vorhergesagten Temperaturen vom wirklichen Wetter stark unterscheiden können. Die Wahl fiel dann auf TensorFlow als Framework für die Implementierung der Modelle. Aufgrund der Leistungsfähigkeit und der umfangreichen Unterstützung moderner Algorithmen. Die Flexibilität und Skalierbarkeit von TensorFlow ermöglichten eine effiziente Entwicklung und Evaluation der Wettervorhersagemodelle.

Die Ergebnisse der durchgeführten Wettervorhersagemodelle zeigen, dass die Vorhersagen für Temperatur, Luftfeuchtigkeit und Windgeschwindigkeit nur geringfügig von den tatsächlichen Werten abweichen, während die Vorhersagen für andere Parameter deutlich schlechter sind. Insbesondere bei der Verwendung eines Netzwerks, das mehrere Wetterstationen miteinander verbindet, wie von Prof. Dr. Transchel empfohlen, wäre es wahrscheinlich möglich gewesen, signifikant bessere Vorhersagen zu erzielen. Die Entscheidung, statt eines Convolutional Neural Networks ein LSTM zu verwenden, erwies sich als vorteilhaft, da dieses Modell bessere Vorhersagen lieferte. Auch durch die Integration der Fourierreihe als Features in das LSTM könnten die Vorhersagen an Genauigkeit gewinnen. Dennoch bleibt festzuhalten, dass die Vorhersagegenauigkeit insgesamt durch die begrenzten Datenquellen, die Reduzierung auf einen Standort und die Komplexität des Vorhersageprozesses eingeschränkt war. Die Darstellung in einer Applikation und das Verbinden von Frontend und Backend funktionierte reibungslos und ist äußerst zuverlässig.

Insgesamt wurden die gewonnenen Erkenntnisse aus der Literaturrecherche und den Expertengesprächen sorgfältig in die Entwicklung und Umsetzung der Wettervorhersagemodelle und deren Darstellung gesetzt, wobei auf die vorhandenen Ressourcen und Einschränkungen geachtet wurde. Die gewählten Methoden ermöglichten trotz der Komplexität des Themas die Erstellung guter Vorhersagen einiger Parameter und trugen zur Weiterentwicklung im Bereich der Wettervorhersage mit Machine Learning bei.

5. Schluss und Ausblick

In Anbetracht der erfolgreichen Entwicklung einer Wetterapp, die mithilfe von Machine Learning-Modellen eine gute Vorhersage des Wetters mit einem Unterschied von nur wenigen Grad Celsius ermöglicht, eröffnen sich vielfältige Perspektiven für zukünftige Entwicklungen auf diesem Gebiet. Die Schaffung einer solchen Anwendung, zeigt wie weit sich die Verwendung von Machine Learning-Technologien im Bereich der Wettervorhersage voranschreiten könnte. Die Erprobung verschiedener ML-Modelle hat gezeigt, dass die kontinuierliche Erforschung und Optimierung dieser Modelle von entscheidender Bedeutung ist, um präzisere Vorhersagen zu ermöglichen.

Durch die Analyse und Vorhersage von Wetterdaten sowie deren Darstellung in Echtzeit konnten wir bereits feststellen, dass die Vorhersagen unter normalen Wetterverhältnissen der Realität sehr nahe kommen. Diese App stellt somit oft eine verlässliche Quelle für die Wettervorhersage dar. Dennoch ist es wichtig zu betonen, dass Vorhersagen trotz ihrer Genauigkeit mit Vorsicht genutzt werden sollten. Sie unterliegen verschiedenen Einflüssen, die vom Modell nicht einbezogen werden und können aus diesem Grund von der Realität abweichen.

Um die Leistungsfähigkeit des Machine Learning-Modells weiter zu verbessern, empfehlen wir die Integration mehrerer Wetterstationen in ein deutlich vielschichtigeres Netzwerk. Dadurch können mehr Daten einbezogen und das Wetter würde nicht auf einen Standort reduziert werden, was zu einer besseren Vorhersage des globalen Wetters führt. Zusätzliche Daten wie beispielsweise zur Bewölkung oder Satellitenbilder würden die Vorhersagen deutlich verbessern. Das Einbeziehen dieser Daten würde es ermöglichen, frühzeitig auf Veränderungen zu reagieren und die Qualität der Vorhersagen zu steigern. Meteorologen arbeiten schon heute an Modellen mit Künstlicher Intelligenz und wir sind der Meinung, dass dies sehr wahrscheinlich die Zukunft der modernen Wettervorhersage sein wird.

In Zusammenfassung lässt sich sagen, dass die Besondere Lernleistung im Bereich Machine Learning zur Analyse, Vorhersage und Darstellung von Wetterdaten eine herausfordernde, aber äußerst wertvolle Erfahrung war. Durch intensive Arbeit und die Zusammenarbeit mit Experten konnten wir ein umfassendes Verständnis in diesem Fachgebiet erlangen. Diese Erfahrung ist ein solides Fundament für unseren zukünftigen Werdegang im Bereich der Mathematik und Informatik.

Wir bedanken und für die Unterstützung von

Dr. Stefanie Hollborn vom

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

Prof. Dr. Fabian Transchel und Oliver Otto von der

HOCHSCHULE harz

Dem Team von

OpenWeather

Für das freundliche zur Verfügung stellen von Wetterdaten

Quellen

[1] DataSolut. (o.D.). Was ist Machine Learning? Verfügbar unter: https://datasolut.com/was-ist-machine-learning/. Zugriffsdatum: 15. April 2024.
[2] IBM. (o.D.). Überwachtes Lernen. Verfügbar unter: https://www.ibm.com/de-de/topics/supervised-learning#:~:text=%C3%9Cberwachtes%20Lernen%2C%20auch%20bekannt%20als,oder%20Ergebnisse%20korrekt%20vorhersagen%20k%C3%B6nnen. Zugriffsdatum: 15. April 2024.
[3] DataCamp. (o.D.). Supervised Machine Learning. Verfügbar unter: https://www.datacamp.com/blog/supervised-machine-learning. Zugriffsdatum: 15. April 2024.
[4] DataSolut. (o.D.). Wie funktioniert ein Künstliches Neuronales Netzwerk. Verfügbar unter: https://datasolut.com/neuronale-netzwerke-einfuehrung/. Zugriffsdatum: 15. April 2024.
[5] Udemy. (o.D.). Artificial Intelligence: A Complete Introduction. Verfügbar unter: https://www.udemy.com/course/artificial-intelligence-a-complete-introduction/?couponCode=LETSLEARNNOWPP. Zugriffsdatum: 15. April 2024.
[6] Facebook. (o.D.). Prophet. Verfügbar unter: https://facebook.github.io/prophet/. Zugriffsdatum: 15. April 2024.
[7] TensorFlow. (o.D.). About TensorFlow. Verfügbar unter: https://www.tensorflow.org/about. Zugriffsdatum: 15. April 2024.
[8] Kinsta. (o.D.). Was ist Express.js? Verfügbar unter: https://kinsta.com/de/wissensdatenbank/was-ist-express-js/. Zugriffsdatum: 15. April 2024.
[9] Red Hat. (o.D.). Was ist eine REST-API? Verfügbar unter: https://www.redhat.com/de/topics/api/what-is-a-rest-api. Zugriffsdatum: 15. April 2024.
[10] Figma. (o.D.). Figma. Verfügbar unter: https://www.figma.com/. Zugriffsdatum: 15. April 2024.
[11] DatabaseCamp. (o.D.). Mittlerer quadratischer Fehler (MSE). Verfügbar unter: https://databasecamp.de/statistik/mse. Zugriffsdatum: 15. April 2024.

Eigenständigkeitserklärung

Hiermit versicheren wir, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst wurde. Wir haben keine anderen als die angegebenen Quellen und Hilfsmittel verwendet sowie Textpassagen, die wörtlich oder sinngemäß aus fremden Quellen übernommen wurden, kenntlich gemacht.

Leon Hofmann
Gatersleben, 25.03.2024

Tim Schrader
Güsten, 15.04.2024

Anhang

- Kommentierter Quelltext Node.js
- Kommentierter Quelltext Pythonmodelle
- Kommentierter Quelltext Bienenprojekt
- Zeitdokumentation
- Aufgabenteilung