Universität Siegen

Wirtschaftswissenschaften
Wirtschaftsinformatik
Wirtschaftsrecht

Vorhersage von Unfallzahlen

Mithilfe eines künstlichen neuronalen Netzes und einer polynomiellen Regression

Maurice Ahrens & Corinne Pretz

12. Juli 2022

uni-siegen.de

Inhaltsübersicht

1

Einleitung

Einleitung

- Motivation:
  - Jährlich rund 150.000 bis 200.000 Unfälle mit Personenschaden
  - Möglichkeit zur Verringerung der Unfälle durch Unfall-Vorhersage
- Zielsetzung:
  - Interaktive Anwendung entwickeln
  - Gibt Nutzer eine Vorhersage zur Wahrscheinlichkeit für Anzahl der Unfälle zu Zeitpunkt und Ort
- Methode:
  - Entwicklung eines künstlichen neuronalen Netzes zu Vorhersage
  - Entwicklung einer polynomiellen Regression als Benchmark-Vorhersage

2

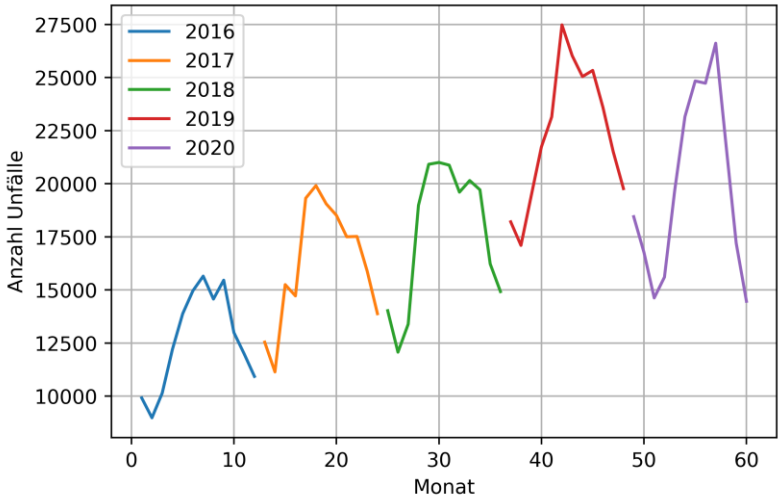
Daten

Unfallatlas

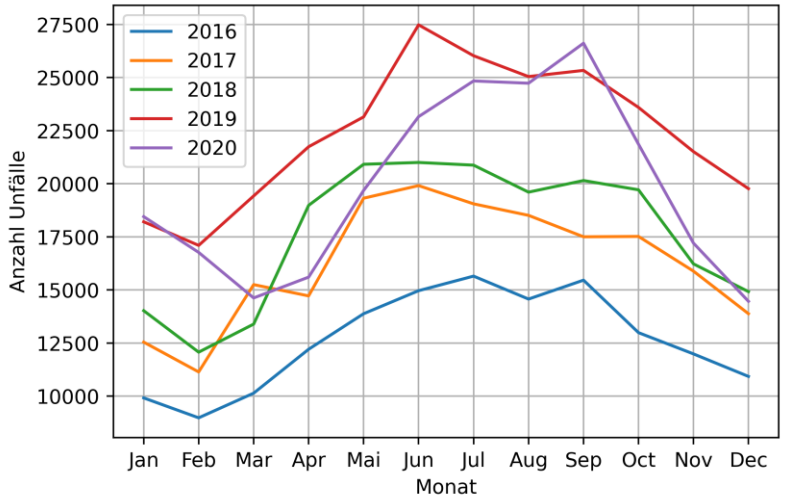
- Quelle: Statistische Ämter des Bundes und der Länder
- Enthält jährlich polizeilich aufgenommene Unfälle des Straßenverkehrs mit Personenschaden
- Aktueller Erfassungszeitraum: 2016 – 2020
- Stärke: Informationen zu Unfallkategorie, -art, -typ, Lichtverhältnisse, Straßenzustand, Position des Unfalls und beteiligte Verkehrsmittel
- Schwäche: Keine Informationen zum genauen Unfallzeitpunkt
- Änderungen:
  - Beschränkung auf Mainz und Wiesbaden
  - Datensatz um keine Unfallzeitpunkte erweitert
  - Für die Vorhersage unbrauchbare Attribute entfernt

Analyse des Unfallatlas

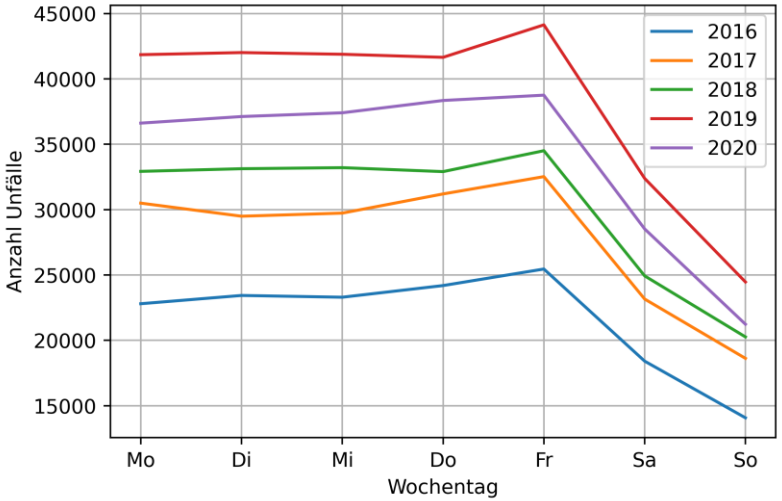
Kapitel 2: Daten



Unfälle von 2016 bis 2020 in Deutschland



Unfälle von 2016 bis 2020 in Deutschland im Monat



Unfälle von 2016 bis 2020 in Deutschland am Wochentag

Temperatur und Niederschlag

- Quellen: Wetterstation Mainz-Lerchenberg und Wetterstation Wiesbaden-Auringen
- Lufttemperatur:
  - Temperatur in stündlichen Abständen
  - Angabe in 2 Meter Höhe und Grad Celsius
  - Zusätzlich: Relative Luftfeuchtigkeit und Qualitätsniveau der Angaben
- Niederschlagshöhe:
  - Niederschlag in 10 Minuten Abständen
  - Angabe in Millimeter
  - Zusätzlich: Niederschlagsform, Qualität der Angaben und binärer Wert für Niederschlag
- Änderung:
  - Lediglich Informationen zu stündlicher Lufttemperatur und Niederschlagshöhe verwendet
  - Mittelwerte der Wochentage in einem Monat berechnet

3

Polynomielle Regression

Theorie

- Polynomfunktion = Funktion, bestehend aus Summe beliebig vieler Potenzfunktionen mit natürlichen Exponenten
- Form:

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_2 x^2 + a_1 x + a_0$$

- n = Grad der Funktion
- $n \in \mathbb{N}$
- a = Koeffizient
- $a_n, a_{n-1}, \ldots, a_2, a_1, a_0 \in \mathbb{R}$
- $a_n \neq 0$

Theorie

- Je höher der Grad, desto mehr Steigungsänderungen
- Verlauf der Unfallzahlen unterliegt ständigen Schwankungen
- Funktion auf Unfallverlauf anpassen
- Weitere Verlauf der Funktion = Vorhersage

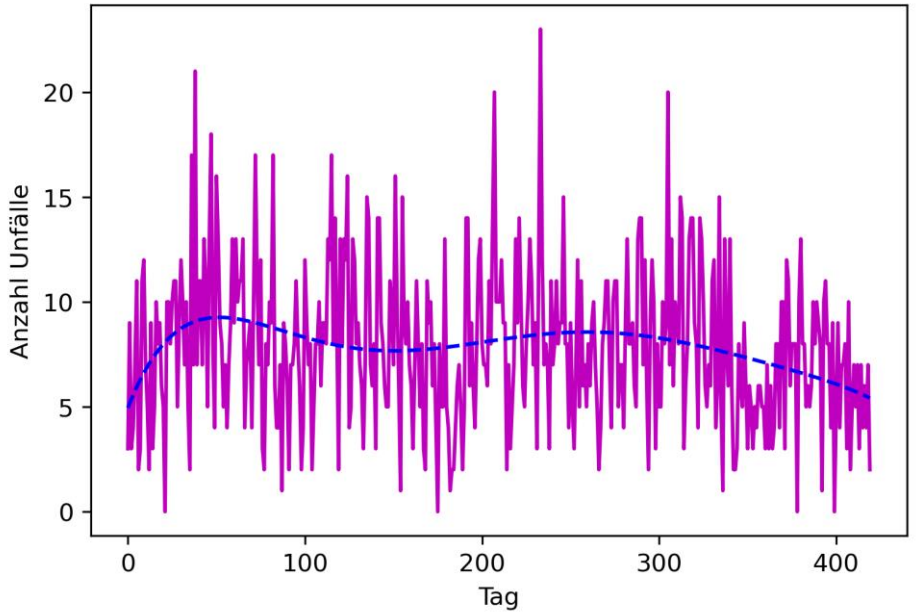
- Wahl eines geeigneten Grades ist wichtig
- Grad zu niedrig = Funktion passt sich Unfallverlauf zu oberflächlich an
  - **Underfitting-Problem**
- Grad zu hoch = Funktion passt sich Unfallverlauf zu streng an
  - **Overfitting-Problem**

Implementierung

- Zeitreihe als Input:

Tag	1	2	3	…	420
Anzahl Unfälle	3	9	3	…	2

- Polynomielle Funktion auf den Daten trainieren lassen:



Ergebnisse

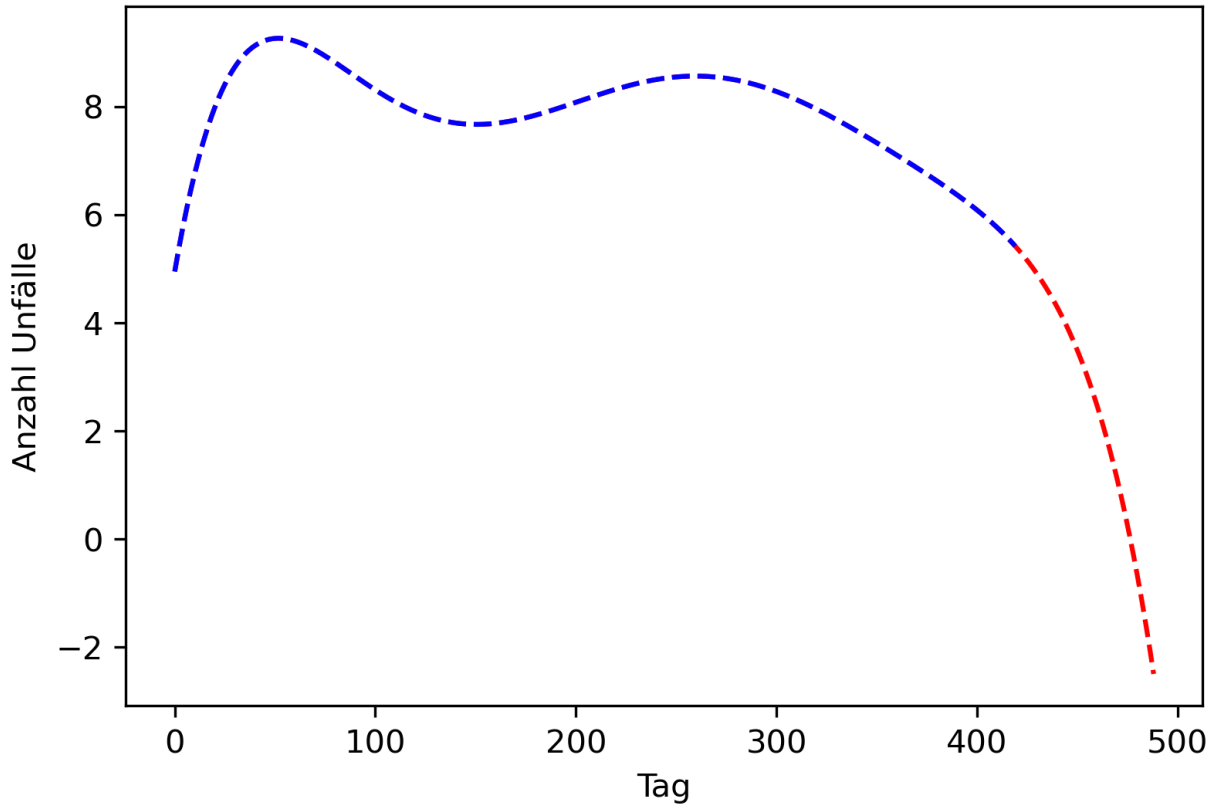
- Accuracy Score = 5.6 %
- Macro f1-Score = 0.086 %
- Micro f1-Score = 0.952 %
- Weighted f1-Score = 0.018 %

- Ergebnisse der Vorhersage:

Tage	1	ab 9	ab 25	ab 36	ab 45	ab 52	ab 63
Anzahl Unfälle	5	4	3	2	1	0	-1

Ergebnisse

- Vorhersage für 70 Tage:

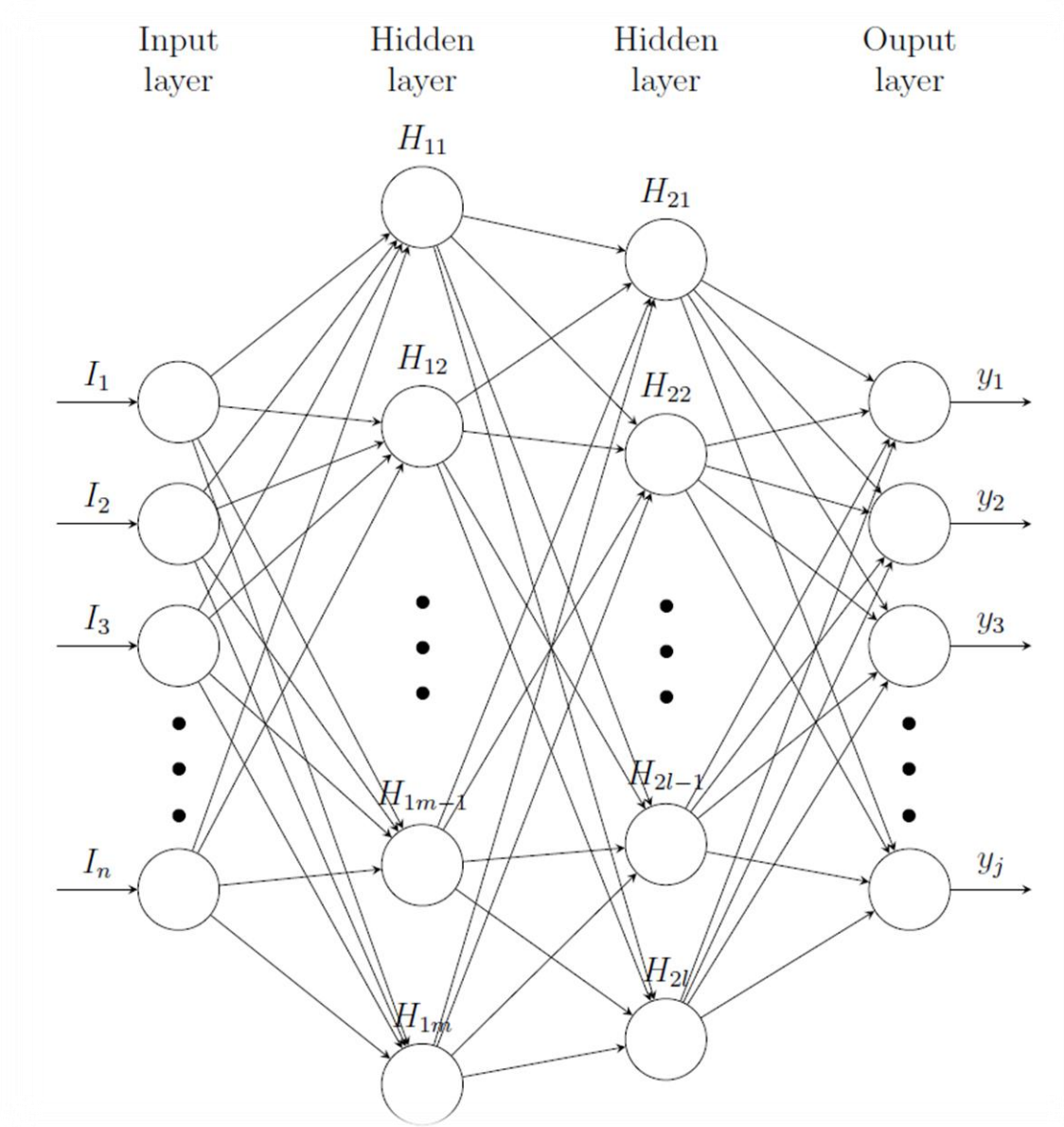


4

Künstliche neuronale Netze

Theorie – Aufbau eines KNNs

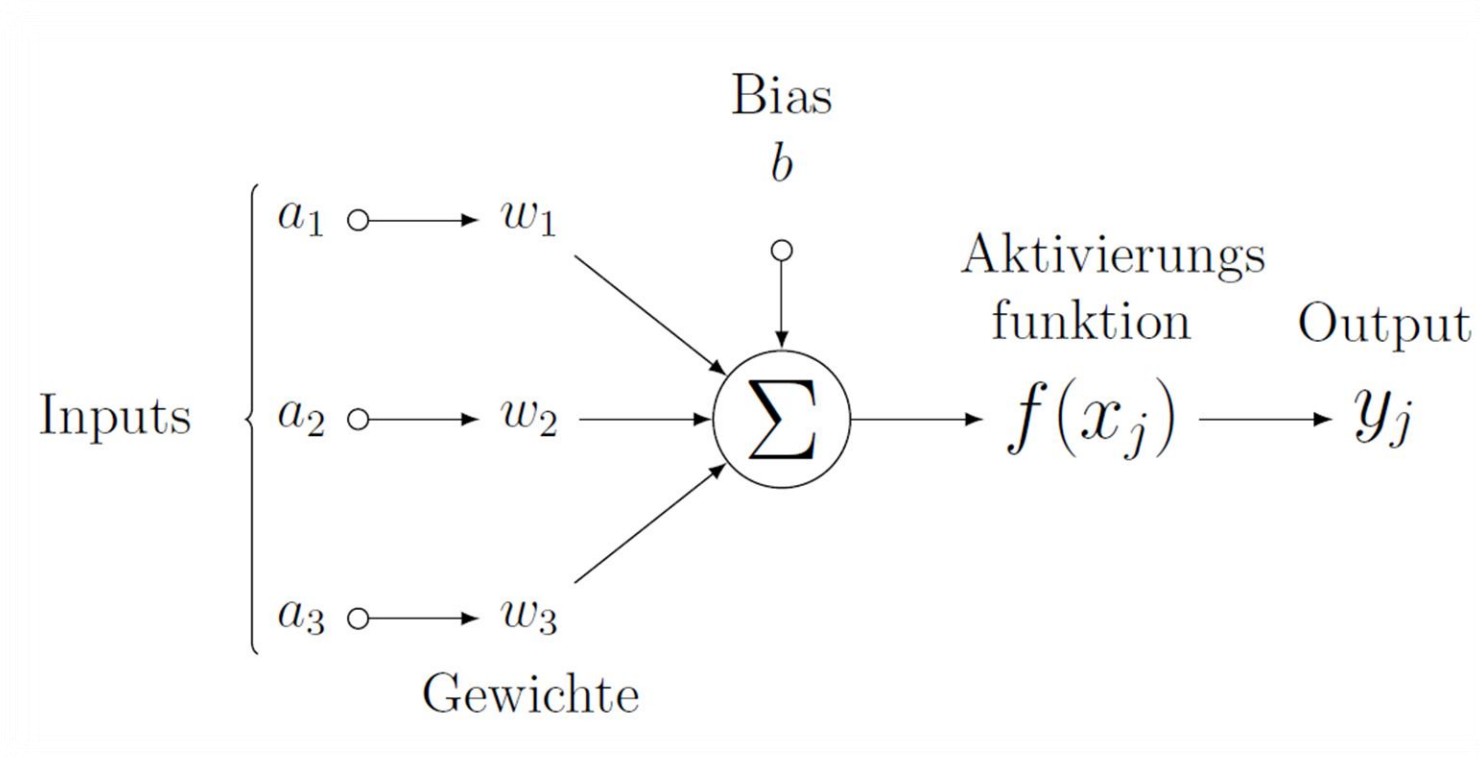
- Wird dem Machine Learning zugeordnet
- Aufbau orientiert sich an der Biologie
- KNNs bestehen aus mehreren Schichten (engl. Layer)
  - Inputlayer
  - Hiddenlayer (eins oder mehr (Deep Learning))
  - Outputlayer
- Schichten bestehen aus einer festgelegten Anzahl an künstlichen Neuronen



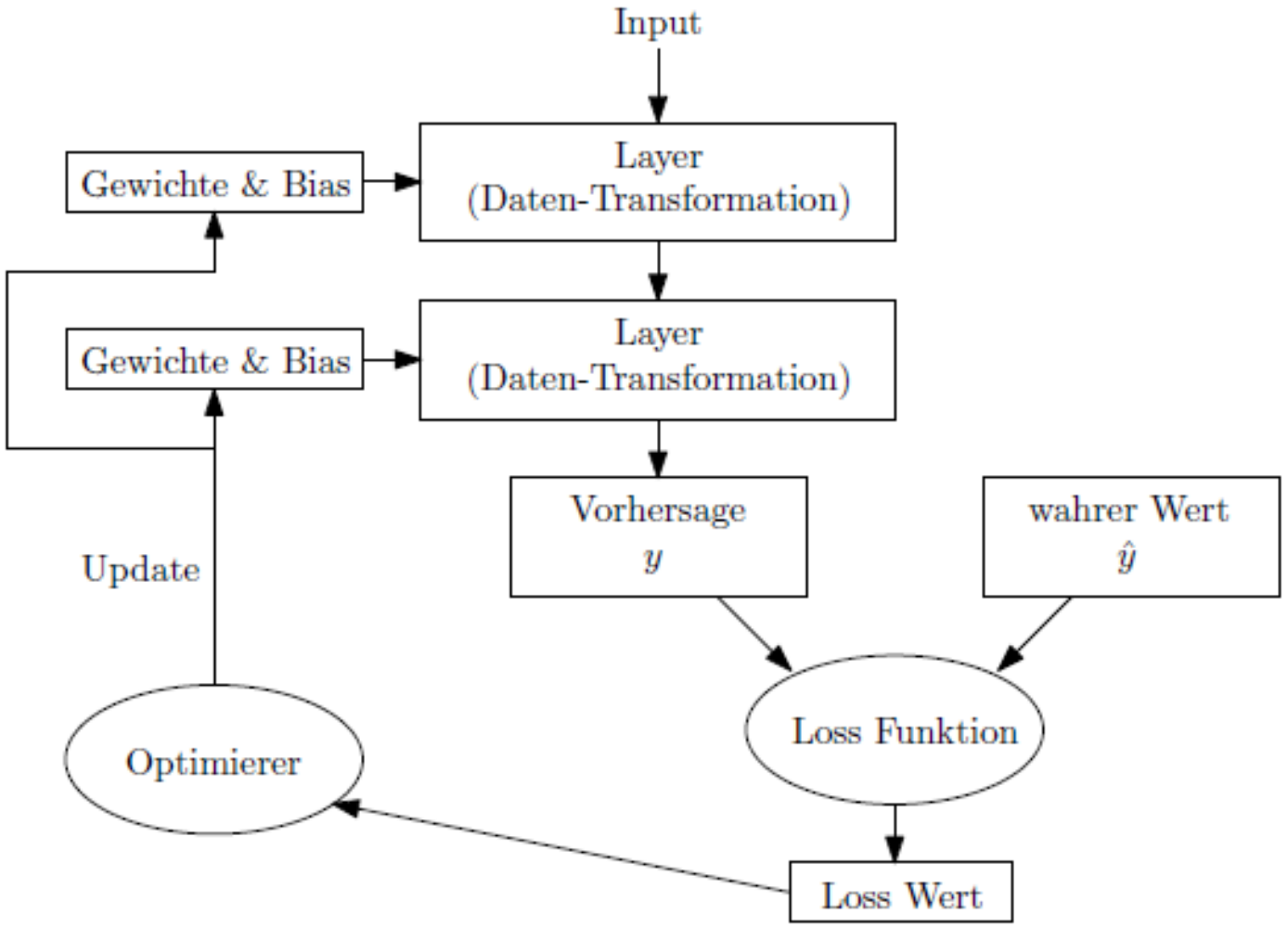
Theorie – Datenfluss eines KNNs

1. Inputlayer bekommt unveränderte Trainingsdaten
2. Weitergabe der Werte an den ersten Hiddenlayer
3. Werte gehen in die „ Aktivierung “ der künstlichen Neuronen ein
4. Berechnung der Aktivierung: $f\left(x_j\right) = f\left(\sum_{i \in I} w_i a_i + b\right) = y_j \qquad \forall j \in J$
5. Aktivierung der künstlichen Neuronen dient als Input für den nächsten Layer

Theorie - Datenfluss eines KNNs



Theorie - Trainingsalgorithmus



Implementierung - Daten

$$\#Jahre + \#Monate + \#Wochentage + \#Stunden = 10080$$

	Insgesamt	Trainingsdaten	Validierungsdaten	Testdaten
# Instanzen	10080	7257	806	2016
rel. Anteil	1	0.72	0.08	0.2

Wiesbaden

# Unfälle	0	1	2	3	4	5	6
# Instanzen	6413	2483	879	222	61	18	4
rel. Häufigkeit	0.63	0.25	0.09	0.022	0.006	0.002	0.0004

Mainz

# Unfälle	0	1	2	3	4	5
# Instanzen	7528	1936	495	98	22	1
rel. Häufigkeit	0.75	0.19	0.05	0.001	0.002	0.0001

Bei unbalancierten Daten ist das Bewertungsmaß Accuracy nicht besonders aussagekräftig

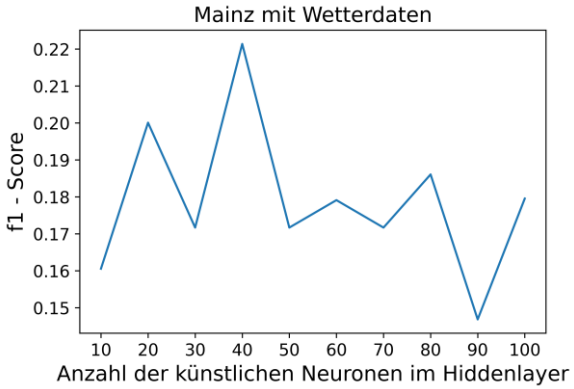
Implementierung – Accuracy vs. F1-Score

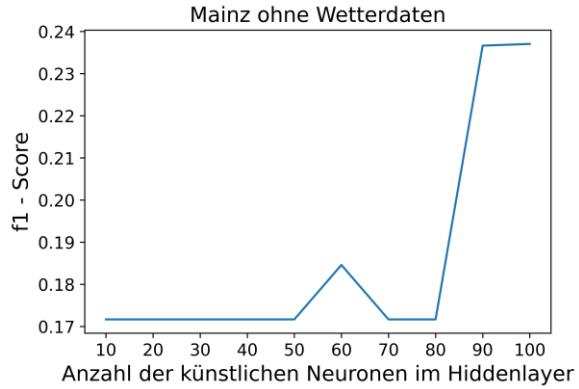
- Accuracy: $$A = \frac{\#\ richtig\ vorhergesagter\ Lösungen}{\#\ der\ Lösungen\ insgesamt}$$
- Recall (Sensitivität): $$Recall = \frac{\#richtig\ positiv}{\#richtig\ positiv + \#falsch\ negativ}$$
- Precision: $$Precision = \frac{\#richtig\ positiv}{\#richtig\ positiv + \#falsch\ positiv}$$
- F1-Score: $$f1 - Score = 2\ \frac{Recall * Precision}{Recall + Precision}$$

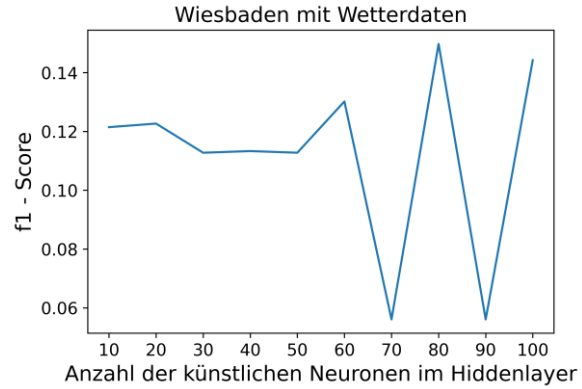
F1-Score berücksichtigt die Fehlerart und nicht nur die Anzahl der Fehler

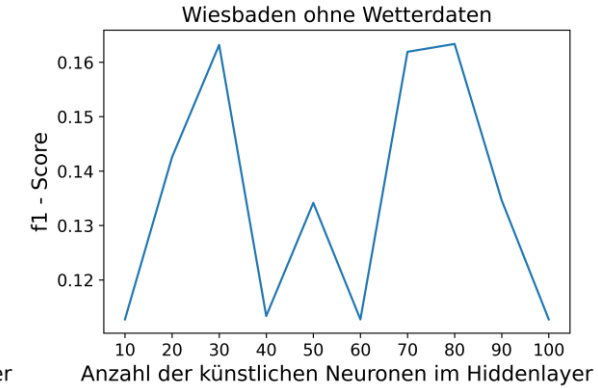
Implementierung – Aufbau des KNNs

Erster Durchlauf

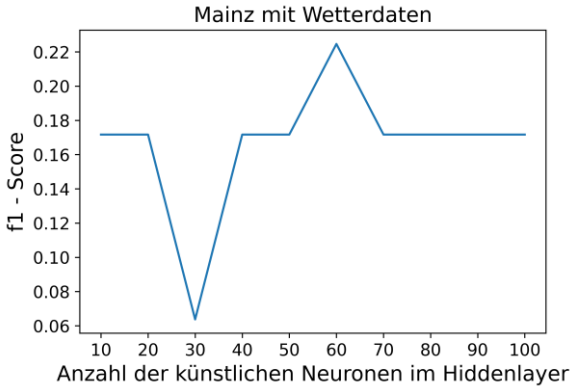


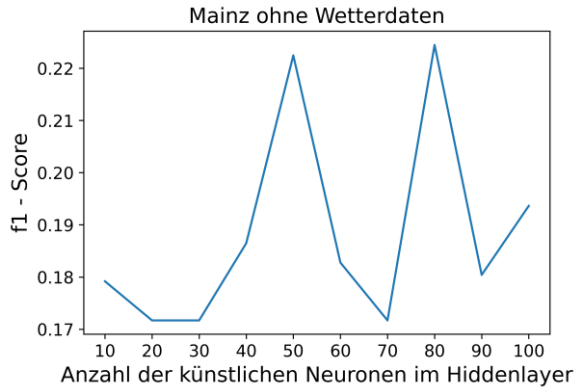


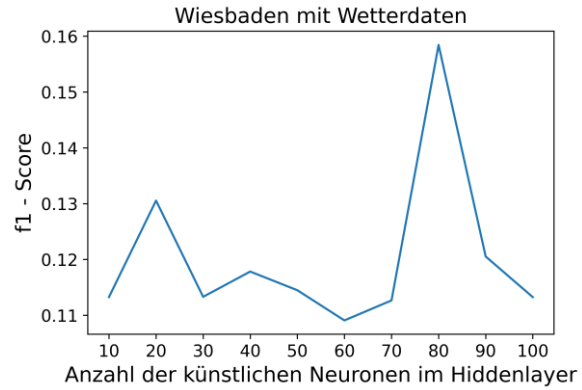


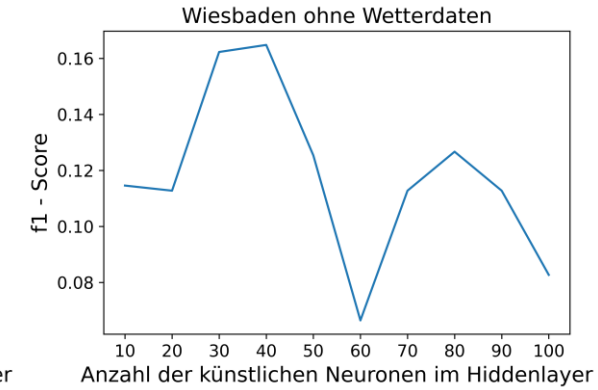


Zweiter Durchlauf



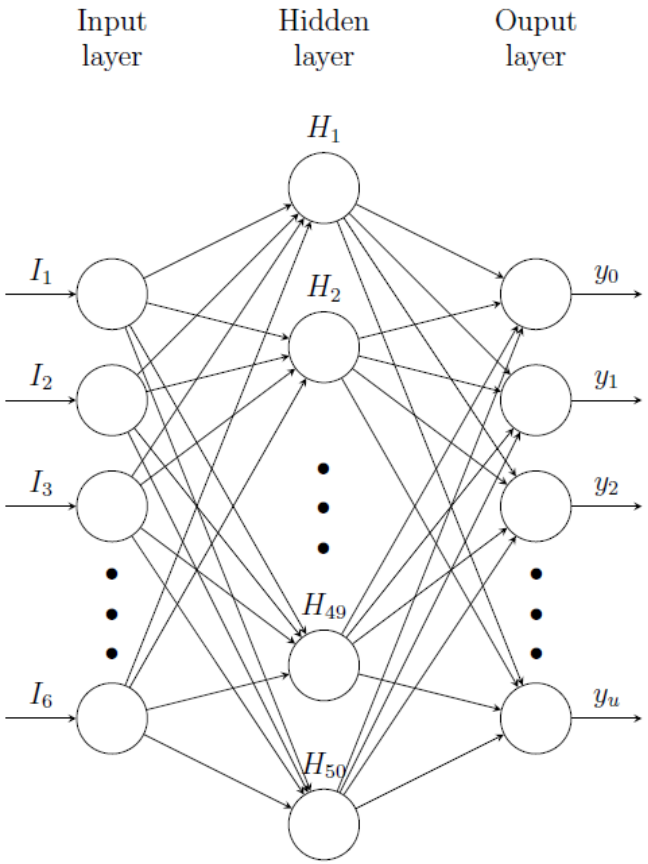




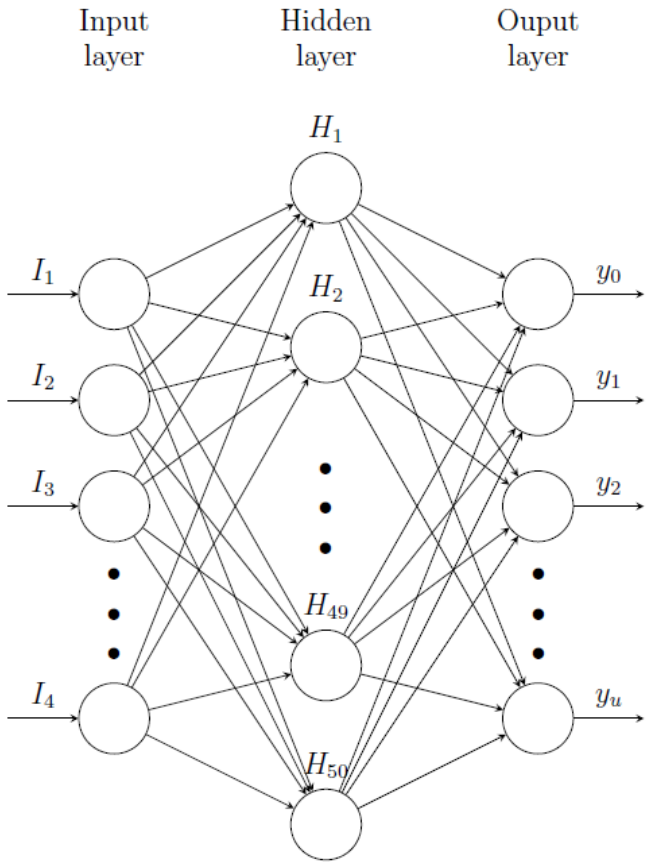


Implementierung – Aufbau der KNNs

mit Wetterdaten

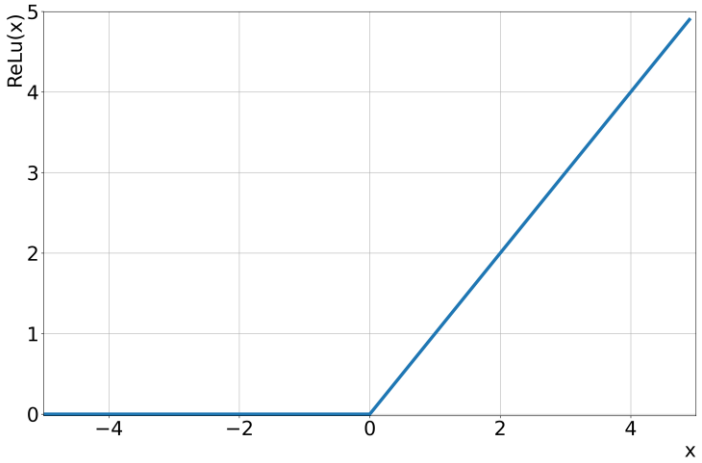


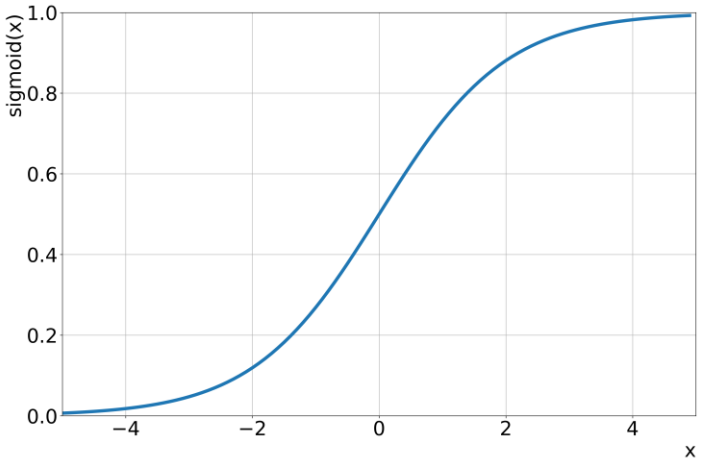
ohne Wetter Daten



Implementierung - weitere Einstellungen des KNNs

- Aktivierungsfunktionen
  Hiddenlayer - „ReLu“ Funktion
  Outputlayer – „Softmax“ Funktion (multiclass multilabel problem)
- Optimierer „Adaptive Moment Estimation“ (Adam)
- Lossfunktion „sparse catergorical-crossentropy“
- Trainierte Epochen = 200



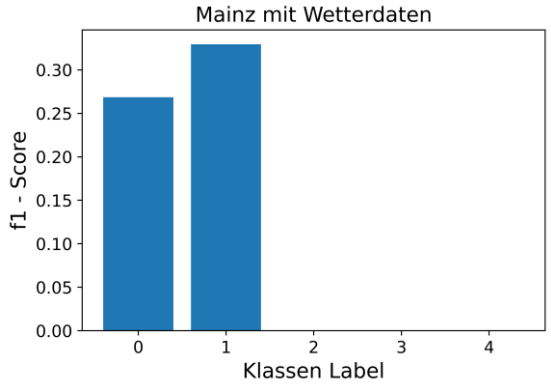


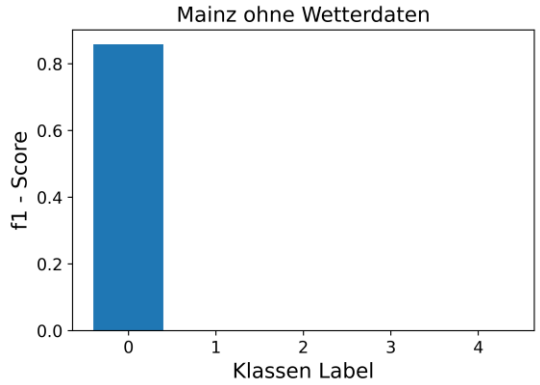
Beispielhafte Vorhersage eine KNNs für die Stadt Mainz

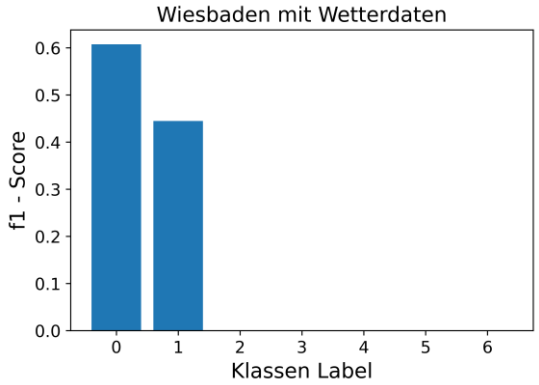
betrachtetes Output-Neuron (Label)	0	1	2	3	4	5
Output des KNN	0.7532	0.2455	0.0001	0.0012	0	0
Gewünschter Output	1	0	0	0	0	0

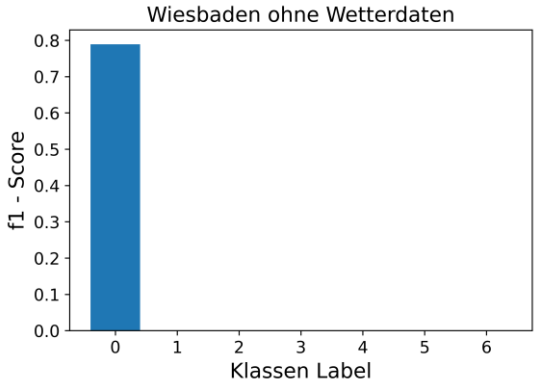
Ergebnisse – Training der KNNs und testen auf den Testdatensätzen

Erster Durchlauf

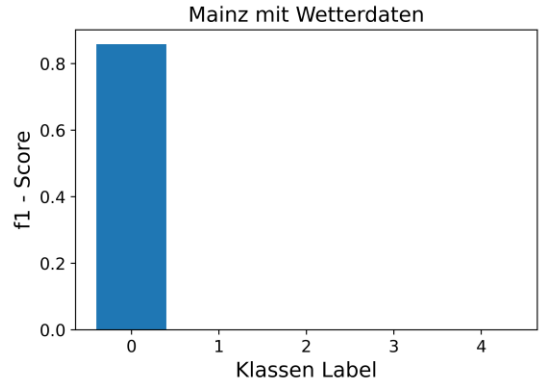


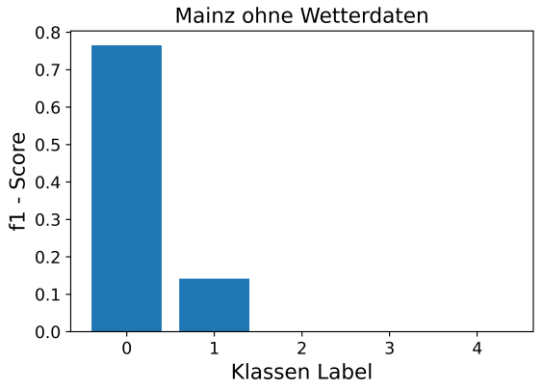


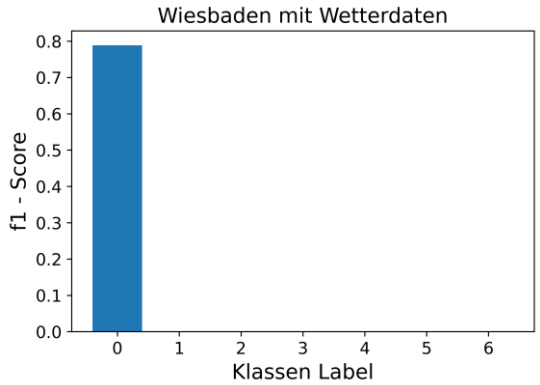


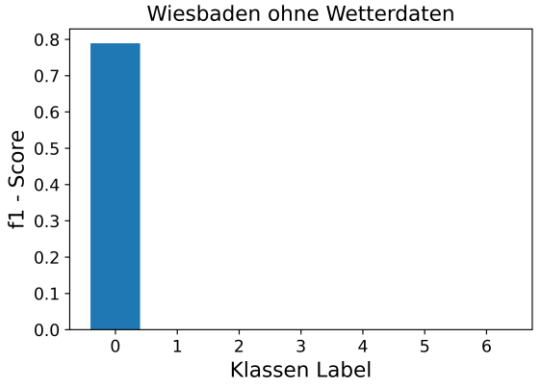


Zweiter Durchlauf





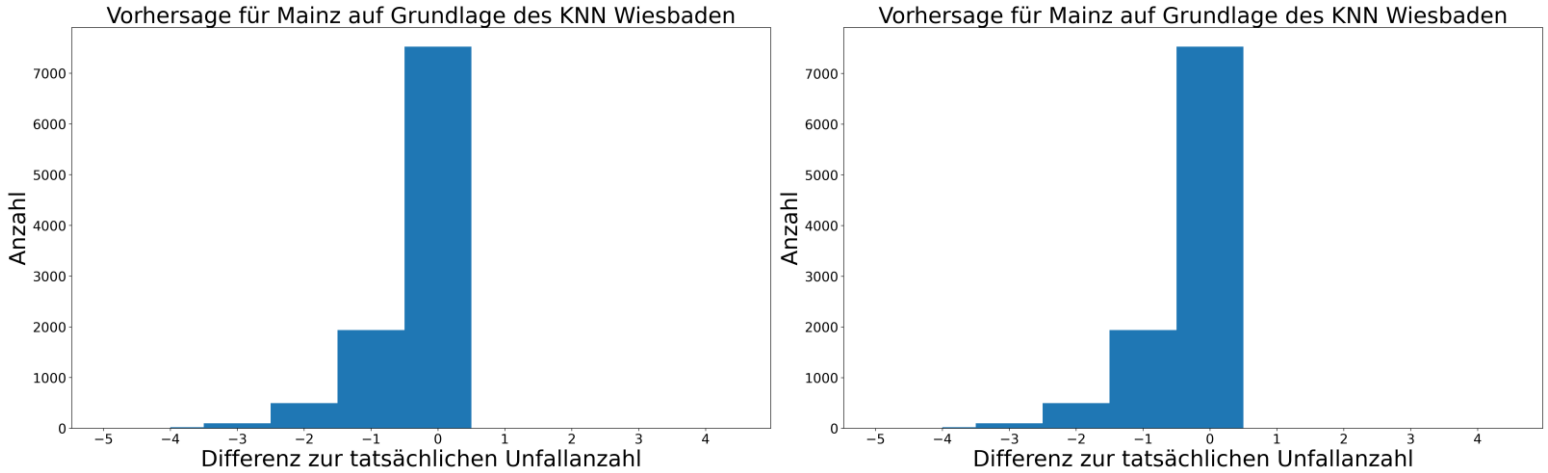




		f1-Score	
Stadt	Wetterdaten	1.Durchlauf	2.Durchlauf
Mainz	✓	0.1196	0.1717
Mainz	×	0.1717	0.1814
Wiesbaden	✓	0.1503	0.1127
Wiesbaden	×	0.1127	0.1127

Ergebnisse

Training mit den Daten einer Stadt und testen auf den Daten der jeweils anderen Stadt



Stadt	Wetterdaten	f1 - Score	accuracy
Mainz	✓	0.14	0.75
Mainz	×	0.14	0.75
Wiesbaden	✓	0.11	0.64
Wiesbaden	×	0.13	0.56

5

Fazit

Fazit

- Zu unbalancierter Datensatz um sichere Vorhersage treffen zu
- Mehr Daten könnten das Problem vielleicht lösen allerdings bleibt vermutlich das Problem der unbalancierten Daten
- Training mit den Daten einer größeren Stadt in der mehr Unfälle geschehen
- Vorhersage der Kategorie des Unfalls oder den involvierten Fahrzeugen könnte Problem des unbalancierten Datensatzes lösen

Quellen

- **3Blue1Brown**. But what is a neural network? https://www.youtube.com/watch?v=aircAruvnKk, 2021. Accessed: 2021-14-07.
- **Bradley, R. A., and Srivastava, S. S.:** Correlation in polynomial regression. The American Statistician 33, 1 (1979), 11–14.
- **Brownlee, J.** Gentle introduction to the adam optimization algorithm for deep learning. https://machinelearningmastery.com/adam-optimization-algorithm-fordeep-learning/, 2017. Accessed: 2022-25-06.
- **Brownlee, J.** A gentle introduction to the rectified linear unit (relu). https://machinelearningmastery.com/rectified-linear-activation-functionfor-deep-learning-neural-networks/, 2019. Accessed: 2022-25-06.
- **Buckland, M., and Gey, F.** The relationship between recall and precision. Journalof the American society for information science 45, 1 (1994), 12–19.
- **Chollet, F.** Deep learning with Python. Simon and Schuster, 2017.
- **Korstanje, J.** The f1 score. https://towardsdatascience.com/the-f1-scorebec2bbc38aa6, 2021. Accessed: 2022-25-06.
- **Ostertagov´a, E.:** Modelling using polynomial regression. Procedia Engineering 48 (2012), 500–506.
- **Strecker, S.** K¨unstliche Neuronale Netze: Aufbau und Funktionsweise. Universit¨atsbibliothek, 2004.
- **Tensorflow.** Module: tf.keras.activations. https://www.tensorflow.org/api docs/python/tf/keras/activations, 2021. Accessed: 2021-14-07.

Vielen Dank für eure Aufmerksamkeit

Maurice Ahrens & Corinne Pretz

Projektseminar SoSe 22