Hochschule Ruhr West
UNIVERSITY OF APPLIED SCIENCES

Hochschule Ruhr West

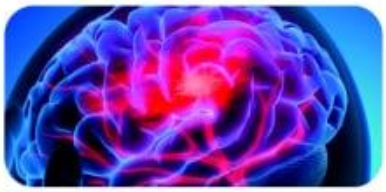
Studiengang Wirtschaftsinformatik

Studienort Bottrop

Sommersemester 2022

Bachelorarbeit

Vergleich von Logistischer Regression und Deep Learning bei der Vorhersage von Schlaganfällen



Quelle: Kaggle (Stroke Prediction)

Erstgutachterin: Prof. Dr. Anne Stockem Novo

Zweitgutachter: Prof. Dr. Fatih Gedikli

Hossein Amiri-Hormozaki
Matrikelnummer: 10009152
Wirtschaftsinformatik

Abgegeben am 24.06.2022

Abstract

Die Bachelorarbeit befasst sich mit dem Vergleich von Logistischer Regression und Deep Learning bei der Vorhersage von Schlaganfällen hinsichtlich der Frage, ob in einer binären Klassifikationsaufgabe die komplexe und aufwändige Methode des Deep Learnings sich bei Anwendung auf kleine tabellarische Datensätze bewährt oder ob Logistische Regression als Basismodell des Maschinellen Lernens effizienter ist. Methodisch werden folgende Schritte ausgeführt: Beschreibung beider Modelle, Durchführung der Datenvorbereitung unter Verwendung des „Stroke Prediction Dataset“ von Kaggle, Implementierung beider Methoden mit dem gleichen angepassten Datensatz. Der abschließende Vergleich benennt als Fazit die Unterschiede in den Ergebnissen und Voraussetzungen für den sinnvollen Einsatz beider Methoden. Eine Schlussfolgerung angesichts der geringen Ergebnisunterschiede hinsichtlich der Prognosegenauigkeit der Schlaganfallrisikos dürfte sein, dass Deep Learning, um ökonomisch sinnvoll bei tabellarischen in kleineren Datensätzen eingesetzt zu werden, aktuell noch nicht genügend bessere Ergebnisse vorweist.

Abstract

This bachelor thesis deals with the comparison of logistic regression and deep learning in the prediction of strokes with regard to the question whether the complex and time-consuming method of deep learning proves its worth in a binary classification task when applied to small tabular data sets or whether logistic regression as a basic model of the machine learning is superior. Methodologically, the following steps are carried out: description of both models, execution of data preparation using Kaggle's *Stroke Prediction Dataset*, implementation of both methods with the same adjusted dataset. The concluding comparison names the differences in the results and prerequisites for the sensible use of both methods as a conclusion. Comparing the small differences in results with regard to the accuracy of forecasting the risk of stroke, one conclusion could be that deep learning does not yet show sufficiently better results to be used in an economically sensible manner.

Danksagung

Ich danke Frau Prof. Dr. Anne Stockem Novo für die Unterstützung bei der Themenfestlegung und hilfreicher Rückmeldung bei der Erstellung der Arbeit.

Inhaltsverzeichnis

1 EINLEITUNG 1

1.1 MOTIVATION 1
1.2 PROBLEMDARSTELLUNG 2
1.3 WAS SOLL BEIM *STROKE PREDICTION DATASET* VORHERGESAGT WERDEN? 3
1.4 AUFBAU DER ARBEIT 3

2 GRUNDLAGEN 4

2.1 MACHINE LEARNING 4
2.2 SUPERVISED LEARNING 5
2.2.1 Regressionsmodelle und Klassifikationsmodelle 5
2.2.2 Unterschied zwischen Klassifikation und Regression 6
2.3 LOGISTISCHE REGRESSION 7
2.3.1 Beschreibung 7
2.3.2 Implementierung der Logistischen Regression 9
2.4 DEEP LEARNING 10
2.4.1 Beschreibung 10
2.4.2 Architekturen 10
2.4.3 Aktivierungsfunktion 12
2.4.4 Implementierung von TensorFlow 13
2.4.5 Aufbau eines Models mit tf. Keras 13
2.5 VERGLEICH VON MACHINE LEARNING UND DEEP LEARNING 16

3 MODELLIERUNG 18

3.1 DATENSATZ 18
3.1.1 Allgemeine Beschreibung der Daten 18
3.1.2 Input und Output Variablen 19
3.1.3 Data Preprocessing 19
3.1.4 EDA 22
3.1.5 Data splitting (train, dev, test) 27
3.1.6 Normalization 28
3.2 TRAINING 29
3.2.1 Loss function 29
3.2.1.1 Hyperparameter (Regularisierung) 30
3.2.1.2 Trainingsverlauf (Learning curve) 30
3.2.2 Deep Learning (TensorFlow) 31
3.2.2.1 Hyperparameter 31
3.2.2.2 Trainingsverlauf (learning curve) 32
3.2.2.3 Regularisierung 32

4 EVALUATION 33

4.2 ALLGEMEINES ZU METRIKEN 33
4.3 ROC-AUC CURVE, SCHWELLENWERT, CONFUSION MATRIX 33

5 FAZIT 37

5.1 WAS KANN VERBESSERT WERDEN? 38
5.2 AUSBLICK 38

6 LITERATURVERZEICHNIS 39

Abbildungsverzeichnis

Abbildung 1: Vergleich logistische Regression und lineare Regression 8
Abbildung 2: Sigmoidfunktion, die einer S-Kurve ähnelt und für jede Eingabe einen begrenzten Ausgabewert im Intervall [0,1] liefert. 9
Abbildung 3: Skizzenhafte Darstellung eines Künstlichen Neuronalen Netzes. 11
Abbildung 4: Verarbeitung der Daten in einem Neuron 11
Abbildung 5: Graphen von Aktivierungsfunktionen 12
Abbildung 6: Vergleich der Personen mit (1) und ohne (0) Schlaganfälle 22
Abbildung 7: Darstellung des Durchschnitts von BMI nach Personengruppen der Spalte work_type 23
Abbildung 8: Darstellung der Verteilung von BMI nach Alter 24
Abbildung 9: Anzahl an Samples in den unterschiedlichen Kategorien der Spalte work_type 25
Abbildung 10: Anzahl an Samples in den unterschiedlichen Kategorien der Spalte smoking_status 25
Abbildung 11: Unterscheidung nach Schlaganfall und kein Schlaganfall bezogen auf work_type und bmi 26
Abbildung 12: Der Bereich möglicher Verlustwerte bei wahrheitsgemäßer Beobachtung 29
Abbildung 13: Logistic Regression classifier 30
Abbildung 14: Zusammenfassung des Modells mit der Tensorflow Library 31
Abbildung 15: Trainingsverlauf des künstlichen neuronalen Netzwerkes mit Angabe des Loss an den Trainingsdaten und den Validierungsdaten 32
Abbildung 16: True-Positive-Rate im Verhältnis zur False-Positive-Rate bei Logistischer Regression mit Angabe der unterschiedlichen Schwellenwerte nach Aufrunden der Wahrscheinlichkeiten. 33
Abbildung 17: True-Positive-Rate im Verhältnis zur False-Positive-Rate bei Neuronalem Netz mit Angabe der unterschiedlichen Schwellenwerte nach Aufrunden der Wahrscheinlichkeiten 34
Abbildung 18: Precision und Recall auf einem Graph 35
Abbildung 19: Zwei Konfusion Matrizen der Modelle Logistische Regression und KNN im Vergleich 36

Tabellenverzeichnis

Tabelle 1: Verwendeter Datensatz 18
Tabelle 2: Spaltenübersicht mit zugehörigen Datentypen 21
Tabelle 3: Durchschnittswerte bei BMI 23
Tabelle 4: Unterschiedlichen Mustern beim Verlauf der Lernkurve 28
Tabelle 5: Vergleich der Logistischen Regression und Deep Learning anhand der Metriken 35

1 Einleitung

1.1 Motivation

Die Methoden des maschinellen Lernens und der künstlichen Intelligenz etablieren sich langsam, aber sicher im Leben der Menschen, unter anderem im medizinischen Alltag. Zukünftig werden sie Ärzte und Ärztinnen bei Diagnose und Therapie unterstützen und so die Behandlung zum Wohl der Patienten und Patientinnen verbessern können. Es ist daher wichtig, sich mit diesem Thema auseinanderzusetzen und ein Grundverständnis dafür zu entwickeln.

Zu wissen, ob Patienten und Patientinnen ein Risiko laufen, eine schwere Krankheit zu bekommen, ist für sie sowie für den Mediziner und die Medizinerin von hoher Bedeutung. Es stellt sich die Frage, welche datengesteuerten Vorhersagealgorithmen bei der Früherkennung und Behandlung von Krankheiten effizient eingesetzt werden können. Logistische Regression und Deep Learning bieten sich für die Untersuchung dieser Frage an.

Deep Learning ist eine sich rasant entwickelnde Methode der Künstlichen Intelligenz, die in vielen Aufgabenstellungen zum Einsatz kommen kann.

Logistische Regression ist bei binärer Klassifikation ein Basismodell des Machine Learning und häufig erste Wahl bei Problemlösungsverfahren. Beides, Deep Learning und Logistische Regression sind sich in ihren mathematischen Komponenten ähnlich.

Gegenüber Logistischer Regression, die mit wenigen Hyperparametern durch schnelles Training besticht, erfordert Deep Learning auf Grund seiner Komplexität eine umfangreiche Optimierung beim Training[1]. Bewährt hat sich Deep Learning im medizinischen Bereich unter anderem in der Bilderkennung. Eignet sich Deep Learning auch bei Fragestellungen, die auf tabellarischen Daten beruhen, beispielsweise bei Prognosen wie Schlaganfällen auf der Basis von heterogenen Patientendaten?[2] Ist Deep Learning hier einfacheren Methoden wie der Logistischen Regression in

[1] Weiß, Medizinische Statistik, 2018, S. 516f.
[2] Borisov, et al., Deep Neural Networks and Tabular Data, 2022

seiner Leistungsfähigkeit sogar überlegen? Wenn ja, in welchen Fällen und unter welchen Bedingungen? Dies ist die Leitfrage in dieser Arbeit.

1.2 Problemdarstellung

Eine bisher verwendete Methode ist Machine Learning. Machine Learning verwendet leistungsstarke Algorithmen, mit deren Hilfe komplexe Beziehungen zwischen einer großen Zahl von Variablen zu Vorhersagen gesundheitlicher Risiken führen können[3].

Inzwischen wird eine weitere Methode des Machine Learning eingesetzt, das so genannte Deep Neural Network. Es beruht auf einem Deep Learning-Ansatz und unterscheidet sich vom Machine Learning-Ansatz durch die Verwendung eines neuronalen Netzes.

Diese Arbeit befasst sich mit der Frage, was die beiden unterschiedlichen Systeme leisten und wann welches System vorteilhaft ist. Im Bereich Machine Learning werden die Methoden der Logistischen Regression und des Deep Learning verwendet. Logistische Regression ist eine bekannte Methode, unter anderem zur binären Klassifikation. Das Modell der Logistischen Regression wird häufig als Baseline Modell bei einer binären Klassifikation genutzt. Das bedeutet, dass es als einfaches und schnelles Modell einen guten ersten Eindruck verschafft und eine zufriedenstellende Performance liefert. Darauf aufbauend können dann weitere und spezielle Modelle implementiert werden, welche jeweils gegen die Baseline gemessen werden. Deep Learning nutzt neuronale Netze sowie große Datenmengen. Die Architektur ist inspiriert von der Funktionsweise des menschlichen Gehirns und ermöglicht eigene Prognosen oder Entscheidungen. Das Gesundheitsproblem, das in dieser Arbeit exemplarisch für den Vergleich zu Grunde gelegt wird, ist der Schlaganfall. Laut der Weltgesundheitsorganisation (WHO) ist Schlaganfall die zweithäufigste Todesursache weltweit und für etwa 11 % aller Todesfälle verantwortlich. Daher ist es sinnvoll, für dieses Krankheitsbild einen

[3] Hung, Chen, Lai, Lin, & Lee, Comparing Deep Neural Network and Other Machine Learning Algorithms for Stroke Prediction in a Large-Scale Population-Based Electronic Medical Claims Database, 2017, S. 3110ff.

Vergleich der Leistungen beider Methoden durchzuführen. Dazu wird ein Datensatz von Kaggle genutzt, um ein Schlaganfallrisiko vorherzusagen[4]. Abschließend wird erörtert, was bei beiden Methoden eventuell verbessert werden könnte und in welche Richtung sich hierbei die zukünftige Praxis entwickeln dürfte.

1.3 Was soll beim *Stroke Prediction Dataset* vorhergesagt werden?

Der oben erwähnte Datensatz wird verwendet, um vorherzusagen, wie hoch die Wahrscheinlichkeit ist, dass eine Person einen Schlaganfall erleiden könnte. Zwölf Eingabeparameter werden hierbei verwendet. Dazu zählen unter anderem das Geschlecht, das Alter, der Arbeitsstatus und der Raucherstatus. Jede Zeile in den Daten enthält relevante Informationen über die konkrete Einzelperson.

1.4 Aufbau der Arbeit

Kapitel 2: „Grundlagen“ enthält die Beschreibung, Leistung und die Anwendungsmöglichkeiten beider Methoden.
Kapitel 3: „Modellierung“ besteht aus zwei Hauptteilen, dem Datensatz und dem Training. Um ein Training sinnvoll und Erfolg versprechend durchführen zu können, ist zuerst die Erstellung, Auswahl, Prüfung und Eignung der Datensätze wesentlich. Mit den identischen Datensätzen trainiert das Modell in den beiden Methoden Logistische Regression sowie Deep Learning.
Kapitel 4: „Evaluation“ befasst sich mit den Ergebnissen beider Methoden, um feststellen zu können, was deren jeweilige Leistungen sind. Der Prüfung werden verschiedene Metriken zu Grunde gelegt. Im letzten Schritt des Vergleichs beider Methoden wird das Kriterium der Genauigkeit der Vorhersage geprüft.
Kapitel 5: „Fazit“ versucht eine Beurteilung der Frage, wann sich welche Methode besonders gut eignet. Unter der Fragestellung „Was kann verbessert werden?“ schließt die Arbeit mit einem Ausblick ab.

[4] fedesoriano, Stroke Prediction Dataset, 2021

2 Grundlagen

In diesem Kapitel wird Machine Learning im Einzelnen beschrieben, insbesondere Supervised Learning, dabei die Lineare Regression, die Logistische Regression sowie Deep Learning. Dabei werden die spezifischen Merkmale und Leistungen beschrieben.

2.1 Machine Learning

Machine Learning ist ein Teilgebiet der künstlichen Intelligenz. Hier sind enorme technologische Fortschritte erreicht worden. Um aus großen Datenmengen und komplexen Zusammenhängen Erkenntnisse zu gewinnen, erweist sich Machine Learning als unverzichtbar. Der Begriff Machine Learning bezeichnet Methoden, die Zusammenhänge in bestehenden Datensätzen erkennen, um darauf basierend mit Hilfe von Lernprozessen Vorhersagen zu treffen. Entsprechend definiert Mitchell: „*Ein Computerprogramm soll aus der Erfahrung E in Bezug auf eine Klasse von Aufgaben T und ein Leistungsmaß P lernen, wenn sich seine Leistung bei Aufgaben in T, gemessen an P, mit der Erfahrung E verbessert*“[5].

Allgemein gesagt, gibt es drei verschiedene Ansätze des Machine Learning:

1. Supervised Learning („überwachtes Lernen“)
2. Unsupervised Learning („unüberwachtes Lernen“)
3. Reinforced Learning („Bestärkendes Lernen“)

Diese Arbeit beschäftigt sich mit Supervised Learning. Dessen Konzepte werden deshalb detailliert vorgestellt. Der wichtigste Unterschied zwischen Supervised Learning und Unsupervised Learning besteht darin, dass bei Supervised Learning die Labels vorgegeben sind, nach denen der Algorithmus die Daten bearbeiten soll, während bei Unsupervised Learning der Algorithmus die Daten nach selbstständig gewählten Labels bearbeitet. Gängige Methoden des Unsupervised Learning sind Clustering und Faktorenanalyse.[6].

[5] Welsch, Eitle, & Buxmann, 2018, S. 370ff.
[6] Grunert, Machine Learning und Neuronale Netze, 2021, S. 11ff.

2.2 Supervised Learning

Wie oben dargestellt, werden die gesuchten Eigenschaften, das heißt Labels, dem Algorithmus vorgegeben, nach denen er die Datensätze bearbeiten und ordnen soll. Mit Hilfe von Trainingsdaten und Testdaten wird ein Modell entwickelt, das, angewendet auf zukünftige, nicht gekennzeichnete Daten, Vorhersagen treffen kann.

Als Beispiel dient die Ermittlung eines Hagelschadens für einen Versicherer. Dort wird eine große Anzahl von Fotos von Hagelschäden eingegeben und mit ihnen als Trainingsdaten die spätere Erkennung von wirklichen Hagelschäden trainiert. An diesem Beispiel ist zu erkennen, wie bei Überwachtem Lernen aus gekennzeichneten Trainingsdaten ein Modell entwickelt wird, das bei künftigen Fotos, also nicht gekennzeichneten Daten, richtige Vorhersagen treffen kann.

Die Algorithmen des Überwachten Lernens werden unter der Bedingung verwendet, dass die Einflussparameter sowie die Eingabe- und Ausgabedaten bekannt sind[7]. Es geht darum herauszufinden, welche Beziehung die Eingabe- und Ausgabeparameter zu einander haben.

Der Algorithmus lernt, indem er Eingaben erhält, dazu die korrekten Ausgaben. Beim Trainingsdurchgang vergleicht er seine Ausgaben mit den vorhandenen korrekten Ausgaben, und auf diese Weise ist es ihm möglich, im Testdurchgang noch vorhandene Fehler zu entdecken und das Modell zu modifizieren. Dabei verwendet das Überwachte Lernen verschiedene Methoden, beispielsweise Klassifizierung, Regression oder Gradientenverstärkung, um die Werte neuer Daten korrekt vorherzusagen[8].

2.2.1 Regressionsmodelle und Klassifikationsmodelle

Regressionsmodelle und Klassifikationsmodelle sind zwei Verfahren des Überwachten Lernens.

Ein wichtiger, da häufig benutzter Anwendungsbereich des Überwachten Lernens erfolgt mit Hilfe von Regressionsmodellen. Regression wird die Methode genannt, mit deren Hilfe Zusammenhänge zwischen zwei oder

[7] Trauth, Bergs, & Prinz, Monetarisierung von technischen Daten, 2021, S. 623
[8] Ongsulee, 2017, Artificial intelligence, machine learning and deep learning, S. 2

mehreren Variablen untersucht werden. Der Begriff Regression ist so zu verstehen, dass untersucht wird, inwieweit sich ein Merkmal auf ein anderes stützt. Die einfachste Regressionsmethode ist die lineare Regression. Dies ist ein statistisches Verfahren. Datenpunkte werden durch eine Gerade beschrieben. Bei Eingabe noch nicht bekannter Daten sind Vorhersagen möglich, indem der Algorithmus die abhängigen Werte (y: Label) auf der Basis der unabhängigen Werte (X: Variable) ermittelt. Das Ziel ist es, eine metrische Größe wie beispielsweise den Kaufpreis eines Hauses anhand der Lage und Ausstattungsinformationen zu prognostizieren. Ein weiteres Beispiel, das bei Umfragen eine Rolle spielt, ist die Frage, ob und inwieweit Lebenszufriedenheit auf das Einkommen zurückzuführen ist. Dies ist ein Regressionsmodell mit metrischen Zielgrößen[9].

Bei der Klassifikation teilt der Algorithmus Beobachtungen anhand von Eigenschaften in vorher festgelegte Klassen ein. Diese Arbeit beschränkt sich auf nur zwei Klassen, die so genannte binäre Klassifikation. Die beiden Klassen werden häufig „positiv“ und „negativ“ genannt. Um klassifizieren zu können, muss auch eine Schranke für die Wahrscheinlichkeit angegeben werden, etwa 50%. Alle Beobachtungen mit einer Wahrscheinlichkeit von 50 % oder mehr werden der Klasse „positiv“ und alle Beobachtungen mit einer Wahrscheinlichkeit von weniger als 50 % der Klasse „negativ“ zugewiesen. Hier wird die Angabe von 50% nur als Richtwert genannt. Als Grenze kann auch ein anderer Prozentsatz angemessen sein, um ein angemesseneres Ergebnis zu erhalten[10].

2.2.2 Unterschied zwischen Klassifikation und Regression

Klassifikation und Regression unterscheiden sich im Typ des Ergebnisses. In dem Fall der Klassifikation ist das Ergebnis immer diskret, repräsentiert also entweder unterschiedliche Kategorien (0=Fußgänger, 1=Auto, 2=Straßenschild) oder auch binär, also 0=kein Spam, 1=Spam, während die Methode der Regression reelle Zahlenwerte ausgibt. Soll festgestellt werden, ob eine Person Corona positiv ist oder nicht, wird die

[9] Tausendpfund, Fortgeschrittene Analyseverfahren, 2020, S. 8ff.
[10] Kalisch & Meier, Logistische Regression, 2021, S. 43f.

Klassifikationsmethode verwendet. Soll jedoch der Preis eines Hauses nach Kriterien vorhergesagt werden, so muss das Ergebnis ein Zahlenwert sein. Hier kommt die Regressionsmethode zum Einsatz.
Die Klassifikation wird vor allem bei qualitativen Fragestellungen verwendet, während Lineare Regressionsmodelle vorwiegend bei quantitativen Fragestellungen benutzt werden. In dieser Arbeit wird die Prognose des Risikos eines Schlaganfalls mit Hilfe eines Logistischen Regressionsmodells untersucht, da es sich um eine qualitative Fragestellung handelt[11].

2.3 Logistische Regression

2.3.1 Beschreibung

Ein Regressionsmodell mit binären Zielgrößen gehört zu den Klassifikationsmodellen. An dieser Stelle wird die Logistische Regression als ein Modell des Klassifikationsmodells dargestellt. Dabei geht es darum, die Daten in zwei Gruppen einzuteilen, also Ja oder Nein, trifft zu oder trifft nicht zu. In der Medizin kann die Antwort lauten, dass der vorhandene Tumor entweder bösartig (Ja) oder gutartig (Nein) ist. Mathematisch kann diese binäre Zielgröße als $Y \in \{0,1\}$ modelliert werden. Daraus geht hervor, dass Y nur zwei Werte annehmen kann, im genannten Beispiel also gutartig oder bösartig. Logistische Regression klassifiziert die Daten nach 0 oder 1 und bestätigt oder widerlegt im Ergebnis ein Risiko[12].
Die Bezeichnung der Variablen bei der Logistischen Regression ist nicht einheitlich. Je nach Zusammenhang wird die abhängige Variable als Y-Variable, Gruppierungsvariable oder Response-Variable bezeichnet. Auch für die unabhängige Variable gibt es unterschiedliche Bezeichnungen: X-Variablen, Einflussgrößen, erklärende Variablen, Prädiktoren, Kovariaten oder Features.
Im Kern besteht der Unterschied zwischen Linearer und Logistischer Regression darin, dass Logistische Regression bei binären Zielgrößen

[11] Hude, Predictive Analytics und Data Mining, 2020, S. 5f.
[12] Hude, Predictive Analytics und Data Mining, 2020, S. 125f.

eingesetzt wird, nicht jedoch die Lineare Regression, wo es das Ziel ist, Datenpunkte entlang der Schätzgeraden zu finden[13].

Im nachfolgenden Schaubild liegen erkennbar keine Datenpunkte entlang der Schätzgeraden der Linearen Regression. Dagegen erfasst die Kurve der Logistischen Regression viele Datenpunkte und eignet sich als binäre Klassifikationsmethode besser, um vorherzusagen, ab welcher Tumorgröße ein Tumor voraussichtlich bösartig sein könnte.

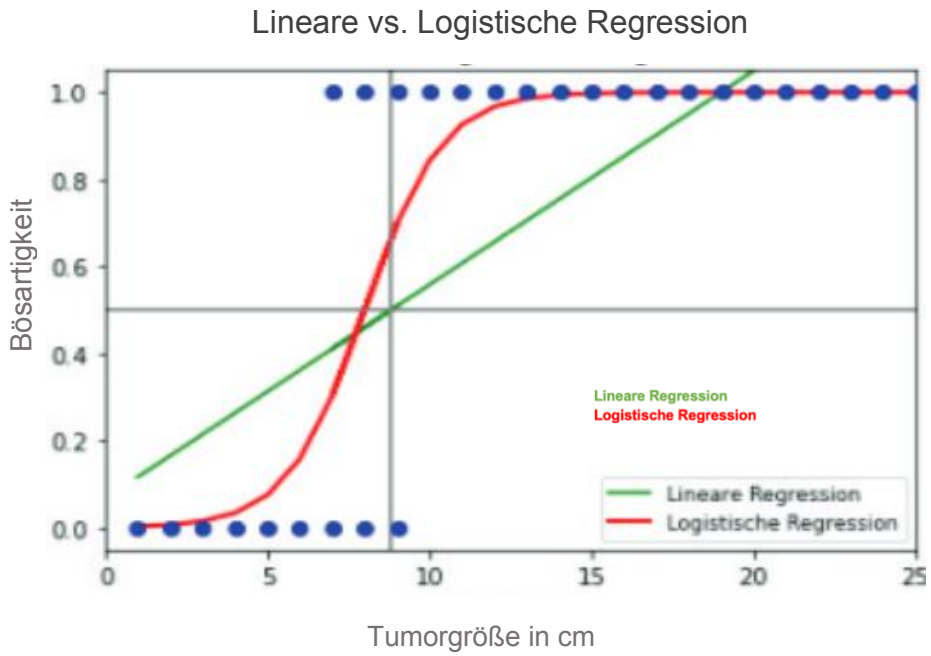


Abbildung 1: Vergleich logistische Regression und lineare Regression bei der Tumorvorhersage

Quelle: Grunert, 2021, S. 207

Daraus folgt, dass die Logistische Regression sich bei Fragestellungen, bei denen es um die Entscheidung Ja oder Nein geht, als geeignete Methode erweist.

[13] Grunert, Machine Learning und Neuronale Netze, 2021, S. 200-208

2.3.2 Implementierung der Logistischen Regression

Eine spezielle *Sigmoid-Funktion* ermöglicht es, Wahrscheinlichkeiten der Ausgabeklasse vorherzusagen. Bei der Verwendung einer *Sigmoid-Funktion* entsteht ein S-förmiger Graph.

Die logistische Funktion, die im Folgenden beschrieben wird, ist ein Spezialfall der *Sigmoid-Funktion*. In diesem Fall liegen die Werte immer zwischen 0 und 1, denn es gilt 0 <= P(t) <=1, wobei P(t) die Wahrscheinlichkeit für das Eintreten eines bestimmten Ereignisses t ist.

Die mathematische Formel dieser logistischen Funktion lautet:

$$P(y = 1) = \frac{1}{1 + e^{-t}}$$

- e = Eulersche Zahl, sie ist die Basis des natürlichen Logarithmus
- t = Logit (Die negative Potenz von *e* steht für ein lineares Regressionsmodell)
- P(y=1) = Wahrscheinlichkeit, dass y = 1

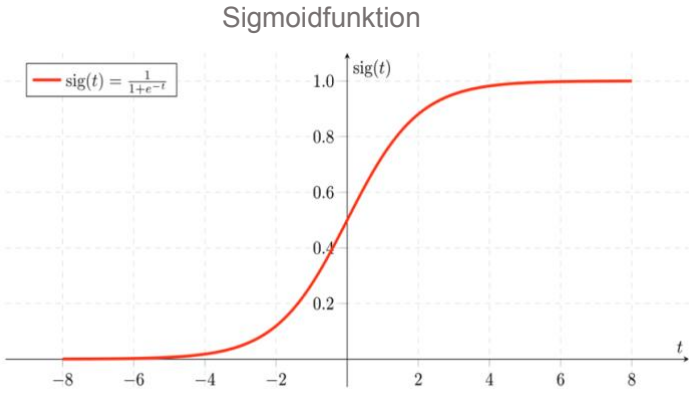


Abbildung 2: Sigmoidfunktion, die einer S-Kurve ähnelt und für jede Eingabe einen begrenzten Ausgabewert im Intervall [0,1] liefert

Quelle: Grunert, 2021, S. 206

Wenn t gegen unendlich geht, wird Y gleich 1 vorhergesagt. Wenn t einen negativ unendlichen Wert annimmt, wird Y gleich 0 vorhergesagt.

Bei der Implementierung wird von der *Sklearn-Bibliothek* das Modell *Logistic Regression* importiert. Dann wird der Zusammenhang zwischen X -Variable und Y-Variable mit Hilfe der Methode *fit()* festgestellt. In dieser Arbeit wird beispielsweise die *Sigmoid-Funktion* als Aktivierungsfunktion der Output-Layer des Künstlichen Neuronalen Netzes verwendet, um im Zuge der binären Klassifikation eine Wahrscheinlichkeit zwischen 0 und 1 ausgeben zu können[14].

2.4 Deep Learning

2.4.1 Beschreibung

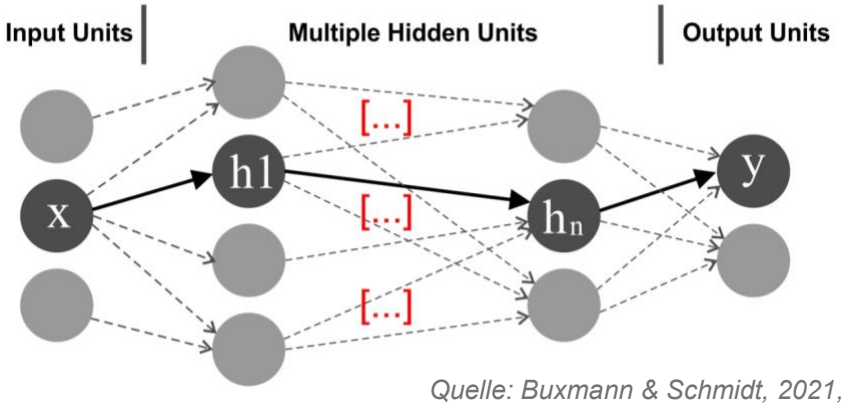
Deep Learning als ein Teilbereich der Künstlichen Intelligenz arbeitet mit tiefen neuronalen Netzen. Sie bestehen - vergleichbar mit dem menschlichen Gehirn – aus Knoten (Neuronen beziehungsweise Units) und Kanten (Synapsen). Anwendungsbereiche sind neben dem in dieser Arbeit untersuchten Bereich der Medizin außerdem die Bereiche des Marketings, des Kundenservice, des Vertriebs sowie autonomes Fahren und Personalwesen. Neuronale Netze werden überwiegend bei Supervised Learning verwendet. In diesem Fall muss es eine Zielgröße geben. Neuronale Netze können im Supervised Learning für Klassifikations- und für Regressionsprobleme zum Einsatz kommen. Unter den zahlreichen *Python* Frameworks werden *TensorFlow*, *Keras*, *Sklearn* oder *Pytorch* am häufigsten benutzt[15].

2.4.2 Architekturen

In der folgenden Darstellung eines Künstlichen Neuronalen Netzes sind die drei Typen von Units zu erkennen. Die Input-Units erhalten die Eingabedaten, in der Darstellung als X bezeichnet. Dies können unter anderem Pixel, Texte oder Gesundheitsdaten wie Blutwerte sein. Output-Units y sind die Ergebnisse, die das Künstliche Neuronale Netz durch Bearbeitung der Eingabedaten ausgibt. Diese Bearbeitung findet in den

[14] Grunert, Machine Learning und Neuronale Netze, 2021, S. 200-20
[15] Grunert, Machine Learning und Neuronale Netze, 2021, S. 306-310

Hidden-Units (Hidden-Layers) statt, deren Anzahl bei der Programmierung bestimmt wird (h1 … h_n).



Quelle: Buxmann & Schmidt, 2021, S. 15

Abbildung 3: Skizzenhafte Darstellung eines Künstlichen Neuronalen Netzes.

Die Eingabedaten, das heißt, die Signale werden über Verbindungen von einem Neuron zu einem oder mehreren Neuronen der nächsten Schicht weitergeleitet, die sie als Eingabewerte registrieren. Um zu erreichen, dass wichtige von unwichtigen Informationen getrennt werden, werden die Werte gewichtet, damit der jeweilige Wert in den folgenden Arbeitsschritten die richtige Bedeutung erhält. Die darauf basierende Ausgabe des Neurons wird als Ergebnis durch die Aktivierungsfunktion ermittelt. Sowohl Menschen als auch der Algorithmus können Gewichtungen festlegen. Die Veränderung von Gewichten erfolgt auf der Basis von Lernregeln[16].

Das folgende Schema veranschaulicht die Vorgehensweise von Künstlichen Neuronalen Netzen. Die drei Inputwerte -1, 4 und 0 werden mit ihren jeweiligen Gewichten multipliziert, aufaddiert und abschließend einer Aktivierungsfunktion übergeben.

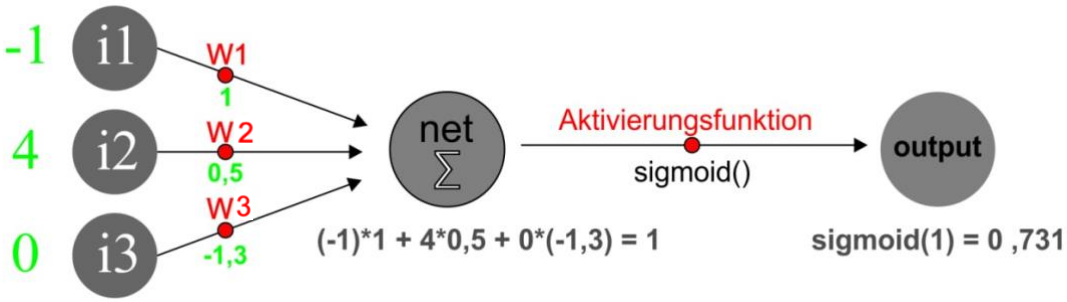


Abbildung 4: Verarbeitung der Daten in einem Neuron *Quelle: Grunert, 2021, S. 311*

[16] Buxmann & Schmidt, Künstliche Intelligenz, 2021, S. 14ff.

Die entscheidenden Faktoren bei neuronalen Netzen sind die Anzahl der Hidden Layers, die Menge der Neuronen pro Layer und die Aktivierungsfunktion. Von wichtigen Verfahren bei Deep Learning - *Feed Forward Neural Networks*, *Convolutional Neural Networks* und *Recurrent Neural Networks* - wird in dieser Arbeit *Feed Forward Neural Networks* verwendet und daher kurz vorgestellt. In *Feed Forward Neural Networks* werden die Informationen immer nur zur nächsten Schicht weitergegeben. Das heißt, der Informationsfluss geht nur in einer Richtung, von der Eingabe zur Ausgabe[17].

2.4.3 Aktivierungsfunktion

Die Aktivierungsfunktion dient der Bestimmung des Wertes, zu dem das künstliche Neuron aktiviert werden soll. Es gibt verschiedene Aktivierungsfunktionen, die unterschiedliche Auswirkungen auf den Ausgabewert eines Neurons haben. In der Arbeit werden folgende Formen der Aktivierungsfunktion verwendet:

- Die *Sigmoidfunktion*, eine sehr viel genutzte Funktion, wandelt alle Eingaben in einen Ausgabewert zwischen 0 und 1 um.
- Die *ReLU-Funktion* gibt auf positive Eingabewerte die gleichen Ausgabewerte zurück. Eingabewerte unter 0 werden als 0 angezeigt.

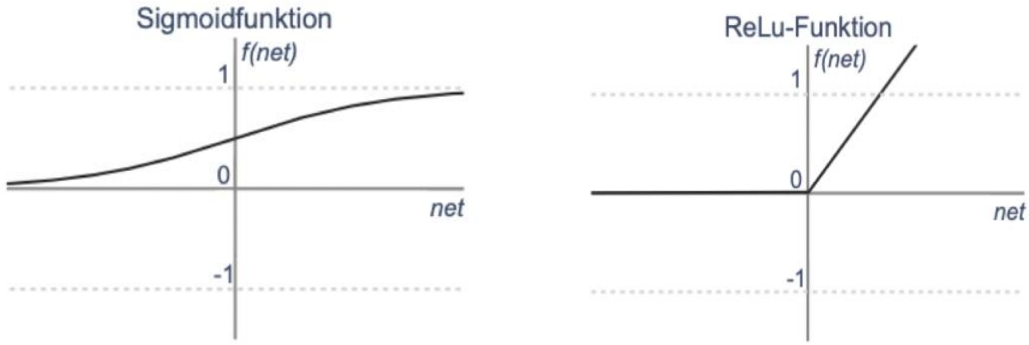


Abbildung 5: Graphen von Aktivierungsfunktionen

Quelle: Frick, et al., Data Science, 2021, S. 229

[17] Grunert, Machine Learning und Neuronale Netze, 2021, S. 310f.

Unter den weiteren Aktivierungsfunktionen sind die Identitäts- beziehungsweise lineare Funktion, die *Tanh-Funktion* sowie die *Softmax*-Aktivierungsfunktion bedeutsam, welche innerhalb dieser Arbeit jedoch nicht von Relevanz sind[18].

2.4.4 Implementierung von *TensorFlow*

In dieser Arbeit wird bei Neuronalen Netzen die sehr verbreitete Bibliothek *TensorFlow* benutzt. *TensorFlow* als eine Open Source Library kann zahlreiche Architekturen von künstlichen neuronalen Netzwerken implementieren und eignet sich besonders bei mathematischen Operationen. Mit Hilfe von *TensorFlow* kann der Aufbau neuronaler Netze für maschinelles Lernen einfacher entwickelt werden. Nicht umsonst nutzt *Google TensorFlow* für beispielweise seine *Google Suche*, *Google Fotos* sowie für die Spracherkennung[19].

Um komplexe Deep Learning Anwendungen, wie beispielsweise Text- oder Bilderkennung zu entwickeln, eignet sich *TensorFlow* besonders gut. Die beiden API-Levels *Estimator-API* und *Keras* haben die Eigenschaft, viele komplexe Details von *Tensorflow* mit einer Abstraktionsschicht zu überlagern, wodurch die Bedienung leichter fällt. Dies ermöglicht, Künstliche Neuronale Netze einfach und flexibel zu entwickeln. Seit *TensorFlow* 2.0 ist Keras ein Teil von *High-Level-API* in *TensorFlow* und vereinfacht seine Nutzung. In dieser Arbeit wird *Keras* verwendet[20].

2.4.5 Aufbau eines Models mit tf. Keras

Zuerst wird das *API* ausgewählt, das verwendet werden soll. In dieser Arbeit wird *Sequential-API* benutzt. Damit können Modelle nacheinander aus verschiedenen parametrisierbaren Schichten aufgebaut werden. Dann wird mittels *model.add ()* der einzelne Layer hinzugefügt. Im Folgenden werden einige Layers genauer beschrieben, die im weiteren Verlauf der Arbeit wichtig sind[21].

[18] Frick, et al., Data Science, 2021, S. 227ff.
[19] Grunert, Machine Learning und Neuronale Netze, 2021, S. 312f.
[20] Frick, et al., Data Science, 2021, S. 218f.
[21] Grunert, Machine Learning und Neuronale Netze, 2021, S. 316ff.

Der Layer *Dropout ()* wird verwendet, um dem Problem der Überanpassung entgegenzuwirken. In diesem Fall werden Neuronen während des Trainings zufällig mit einer zuvor pro Epoche festgelegten Wahrscheinlichkeit *p* abgeschaltet. Das bedeutet, dass bei einer Wahrscheinlichkeit von beispielsweise *p = 0,5* nur *50 %* der Neuronen einer Schicht aktiv sind und während des Trainings trainiert werden. Dabei wird für jede Epoche immer wieder neu ermittelt, um welche Neuronen es sich genau handelt. Infolgedessen werden Neuronen keine zu spezifischen Merkmale lernen, da ein Neuron nicht immer zusammen mit einem anderen Neuron aktiv sein muss. So kann ein mögliches *Overfitting* verhindert werden[22].
Der ebenfalls verwendete Layer *Dense* steht dabei für die Schicht, in der jedes Neuron mit jedem Neuron aus der nächsten Schicht verknüpft ist, wobei die Zahl in Klammern angibt, wie viele Neuronen in der jeweiligen Schicht vorhanden sind.
Das hier vorgestellte und verwendete Modell *Sequential* soll dann unter Nutzung der *compile()* Methode konfiguriert und anschließend trainiert werden.

Die verwendeten Parameter sind:

- *Optimizer*: Als *Optimizer* wird ein Algorithmus bezeichnet, dessen Funktion es ist, die Genauigkeit zu erhöhen. In dieser Arbeit wird *Adam*[23] verwendet.
- *Loss*: Dieser Algorithmus hat die Aufgabe, im Prozess des Trainings den Fehler des Modells auszugeben. Dadurch wird bei jeder Iteration der Abstand zwischen den Zielen und den gezeigten Werten angegeben.
- *Metrics*: Darunter ist eine Liste von Metriken zu verstehen, die den Trainingsprozess überwachen. Dabei ist zu beachten, dass für verschiedene Probleme verschiedene Metriken benötigt werden[24].

[22] Shen, Tian, Liu, Xu, & Tao, Continuous Dropout, 2018, S. 3926
[23] Die Adam-Optimierung ist ein stochastisches Gradientenabstiegsverfahren, das auf einer adaptiven Schätzung von Momenten erster und zweiter Ordnung basiert. Quelle
[24] Grunert, 2021, Machine Learning und Neuronale Netze, S. 316ff.

Wichtige Metriken sind unter anderem *accuracy*, *precision*, *recall* und *f1*[25]. *Accuracy* ist relativ gut zu verstehen und ist meist die erste Wahl, um die Leistung eines Algorithmus bei Klassifizierungsproblemen zu bewerten. Sie kann als das Verhältnis von genau klassifizierten Datenelementen zur Gesamtzahl der Beobachtungen beschrieben werden.
Accuracy wird wie folgt definiert:

$$accuracy = \frac{TP+TN}{TP+FP+FN+TN}$$

Trotz der weit verbreiteten Verwendbarkeit ist *accuracy* in einigen Situationen nicht die geeignetste Leistungsmetrik, insbesondere in den Fällen, in denen Zielvariablenklassen im Datensatz unausgewogen sind.

Precision versucht folgende Frage zu beantworten: Welcher Anteil der vorhergesagten Positiven ist wirklich positiv?
Precision wird wie folgt definiert:

$$precision = \frac{TP}{TP + FP}$$

Ein weiteres sehr nützliches Maß ist *recall*, das eine andere Frage beantwortet: Welcher Anteil der tatsächlichen Positive wird richtig klassifiziert?
Mathematisch ist *Recall* wie folgt definiert[26]:

$$recall = \frac{TP}{TP + FN}$$

[25] Der F1-Score kann als harmonisches Mittel aus Precision und Recall interpretiert werden, wobei ein F1-Score seinen besten Wert bei 1 und seinen schlechtesten Wert bei 0 erreicht. Quelle

[26] Vakili, Ghamsari, & Rezaei, Performance Analysis and Comparison of Machine and Deep Learning Algorithms for IoT Data Classification, 2020

Nun beginnt das Training des kompilierten Modells mit den Trainingsdaten unter Einsatz der Methode *fit()*.
Die Parameter der *fit()*-Methode sind:

- *Epoch*: Eine Iteration wird *epoch* genannt, wobei die Trainingsphase mehrere Epochen umfasst. Diese werden weiter in Batches unterteilt.
- *Batch_size*: Die Daten einer Epoche werden in kleinere Einheiten gruppiert, die *Batch_size* genannt werden. Diese kleineren Einheiten arbeiten unabhängig voneinander bis zur nächsten Epoche.
- *Validation_data*: Um zu überprüfen, wie gut das Modell funktioniert, wird ein Validierungsdatensatz verwendet. Im Ergebnis wird nach jeder Epoche der Loss und die Metriken für den Validierungsdatensatz angezeigt.

Um nun zu überprüfen, wie gut das Modell funktioniert, wird ein Validierungsdatensatz verwendet. Im Ergebnis wird nach jeder Epoche der *Loss* und die Metriken für den Validierungsdatensatz angezeigt. Der nächste Schritt ist die Evaluierung des Modells. Die Evaluierungsmethoden *model.evaluate()* zur Evaluierung des Modells sowie *model.predict()* zur Evaluierung des Ergebnisses werden verwendet[27].

2.5 Vergleich von Machine Learning und Deep Learning

Nach der Beschreibung von Machine Learning und Deep Learning ist es sinnvoll, die Besonderheiten der jeweiligen Methoden noch einmal systematisch zusammenzufassen.
Ein zentraler Unterschied zwischen beiden liegt in der Art und Weise wie Informationen verarbeitet werden. Machine Learning arbeitet mit strukturierten Daten nach mathematischen Verfahren. Dabei kommt dem Menschen eine aktive Rolle zu. Dagegen besteht die Aufgabe beim Deep Learning darin, die Informationen einzugeben, die das Programm autonom in mehreren Schichten bearbeitet. Insofern ist das Deep Learning ein

[27] Grunert, Machine Learning und Neuronale Netze , 2021, S. 320ff.

selbstlernendes System, das mit Rohdaten besser als Machine Learning arbeitet.

Machine Learning wird verwendet, um Muster in Daten nach Definition des Ziels zu erkennen. Im Unterschied dazu wird Deep Learning verwendet, wenn unstrukturierte Daten wie Texte, Bilder, Töne und Videos verarbeitet werden, um Muster zu erkennen. Bei komplexen Aufgabestellungen ist Deep Learning besser geeignet, weil es schneller Lösungen präsentieren kann[28].

Daraus folgt, dass sich die Hardwarevoraussetzungen bei Machine Learning und Deep Learning unterscheiden. Während bei Machine Learning mit relativ kleinen Mengen an verarbeiteten Informationen kleinere *Processing Units* genügen, erfordert Deep Learning wegen großer Datenmengen und einem hohen Grad an Parallelität der Verarbeitung große leistungsstarke Computer mit Grafikprozessoren (GPUs). Die immer komplexer werdenden Neuronalen Netzte erfordern immer größere Verarbeitungskapazitäten, was zu einem entscheidenden Kostenfaktor werden kann. Auch die Unsicherheit, wie lange es dauert, ein Deep Learning Netzwerk zu trainieren, stellt einen relevanten Kostenfaktor dar[29].

[28] Kumar Chauhan & Singh, A Review on Conventional Machine Learning vs Deep Learning , 2018, S. 347

[29] Justus, Brennan, Bonner, & McGough, Predicting the Computational Cost of Deep Learning Models, 2018, S. 3874ff.

3 Modellierung

3.1 Datensatz

3.1.1 Allgemeine Beschreibung der Daten

Bei dem verwendeten Datensatz handelt es sich um einen medizinischen Datensatz mit unterschiedlichen Eigenschaften über den Gesundheitszustand verschiedener Personen. Der Datensatz enthält beispielsweise allgemeine Informationen wie Alter und Geschlecht, aber auch speziellere Datenpunkte wie den Blutzuckerspiegel oder den Body Mass Index (BMI). Als Zielvariable gilt die Spalte “stroke”, in der entweder eine 1 oder eine 0 stehen kann. Der Wert der Zielvariable ist binär und sagt aus, ob die betrachtete Person mit den dazugehörigen Daten einen Schlaganfall hatte. Im Allgemeinen liegt der Datensatz als CSV-Datei vor und enthält 5110 Zeilen und 12 Spalten. Der Datensatz wurde auf der Seite kaggle.com unter diesem link veröffentlicht und steht zu freien Verfügung.

Spaltenbezeichnung	Datentyp	Beispielwerte
id	Integer	9046, 51676, 31112, ...
gender	String	Male, Female, Other
age	Float	67.0, 61.0, 49.0, ...
hypertension	Integer	0, 1
heart_disease	Integer	0, 1
ever_married	String	Yes, No
work_type	String	Private, Self-employed, children, ...
Residence_type	String	Urban, Rural
avg_glucose_level	Float	93.88, 73.00, 95.57, ...
bmi	Float	38, 37, 22, 30, ...
smoking_status	String	never smoked, smokes, Unknown, ...
stroke	Integer	0, 1

Tabelle 1: Verwendeter Datensatz

3.1.2 Input und Output Variablen

Da es sich um ein Problem des Überwachten Lernens handelt, sind im Datensatz Input-Variablen X und eine Output-Variable y enthalten. Zu den Input-Variablen gehören die 10 Spalten *gender, age, hypertension*, *heart_disease, ever_married, work_type, Residence_type*, *avg_glucose_level, bmi* und *smoking_status*. Die Output-Variable y, die für das Modell als Zielvariable verwendet wird, ist die Spalte *stroke*. Diese Spalte ist von Anfang an binär kodiert und enthält eine 1, wenn der Patient einen Schlaganfall hatte, und 0 wenn der Patient keinen Schlaganfall hatte. Die Spalte „id“ enthält für jeden Patienten einen eigene Identifikationsnummer. Diese „id“ wird weder während des Trainings als Teil der Input-Variablen benötigt, noch für die anschließende Evaluation oder Test-Phase. Deshalb wird diese Spalte zu Beginn der *Date-Preprocessing-Pipeline* bereits aus dem Datensatz entfernt. Bei der Zielvariablen gibt es eine unbalancierte Verteilung. Während im gesamten Datensatz mit 5110 Datensamples ganze 4861 Personen vorhanden sind, welche keinen Schlaganfall erlitten haben, stehen dagegen mit 249 nur wenige Personen, welche einen Schlaganfall erlitten haben. Dieses Ungleichgewicht muss im weiteren Prozess der Datenverarbeitung und beim Training des Modells beachtet werden.

3.1.3 Data Preprocessing

Wie im vorherigen Kapitel bereits angesprochen, wird zu Beginn der Datenvorverarbeitung die Spalte “id” aus dem Datensatz entfernt, da diese im weiteren Prozess der Pipeline nicht mehr benötigt wird. Bei der ersten Analyse ist aufgefallen, dass es im Datensatz Zeilen mit fehlenden Werten gibt. In der Spalte “bmi” gibt es 201 Zeilen, die keinen Wert enthalten. Eine weitere Unregelmäßigkeit ist es, dass es in der Spalte “gender” eine einzige Zeile unter den 5110 Datensamples gibt, welche nicht die Ausprägung “male” oder “female” enthält, sondern den Wert “other”. Da die Daten im weiteren Verlauf noch kodiert werden müssen und eine zusätzliche Spalte “other” im *feature-vector* den Datensatz enorm vergrößern würde, wird der Wert “other” aus der Spalte “gender” entfernt.

Damit gibt es ausschließlich fehlende Werte in den Spalten “gender” und “bmi”. Im späteren Verlauf der Datenvorverarbeitung wird der *KNNImputer* aus der Python library *Sklearn* genutzt, um die fehlenden Werte zu ersetzen. Dabei werden auf Grundlage des Künstlichen Neuronalen Netzes mit einer *Distanz-Metrik* die ähnlichsten Datensamples im Datensatz identifiziert und deren Durchschnitt in der jeweiligen Spalte als Wert für den fehlenden Wert ersetzt. Als nächstes werden die jeweiligen Spalten einzelnen Variablentypen zugeordnet, um ein systematisches Vorgehen bei der Kodierung der jeweiligen Werte zu haben.

Bei den qualitativen Variablentypen handelt es sich um Ausprägungen in den Daten, mit denen keine arithmetischen Operationen möglich sind. Desweiteren ist in dem qualitativen Variablentyp des nominalen Typs keine Reihenfolge enthalten. Bei den quantitativen Variablentypen gibt es die Merkmalsausprägung deskriptiv und kontinuierlich. Bei den deskriptiven Variablentypen gibt es nur ganze Zahlen, keine Zwischenwerte, und die Daten bewegen sich in einem groben, aber bestimmten Bereich. Beispielsweise ist das Alter ganzzahlig in dem Datensatz beschrieben und kann nicht niedriger als 0 sein. Auch ist nach aktuellen Erkenntnissen anzunehmen, dass ein Alter ab 100 immer unwahrscheinlicher wird und ein Alter über 200 Jahre aktuell nicht möglich. Bei der kontinuierlichen Spalte “avg_glucose_level” gibt es auch Zwischenwerte hinter dem Komma. Daher gibt es in der Theorie unendlich viele Möglichkeiten an Merkmalsausprägungen. Bei den qualitativen binären Variablentypen liegen die Daten zum Teil schon in richtiger Form vor und müssen gegebenenfalls nur noch in die richtigen Datentypen gebracht werden. Beispielsweise muss aus “yes” eine 1 und aus “no” eine 0 werden. Im Grunde waren die Spalten aber schon in der richtigen Form, nur der Datentyp war nicht ideal[30].

[30] Kaliyadan & Kulkarni, Types of Variables, Descriptive Statistics, and Sample Size, 2019, S. 82ff.

Spaltenbezeichnung	Variablentyp	Methode	Spaltenbezeichnung kodiert
gender	Qualitativ - Binär	OrdinalEncoder	gender
age	Quantitativ - Deskriptiv	keine	age
hypertension	Qualitativ - Binär	keine	hypertension
heart_disease	Qualitativ - Binär	keine	heart_disease
ever_married	Qualitativ - Binär	OrdinalEncoder	ever_married
work_type	Qualitativ -Nominal	OneHotEncoder	work_type_1, work_type_2, work_type_3, work_type_4, work_type_5
Residence_type	Qualitativ - Binär	OrdinalEncoder	Residence_type
avg_glucose_level	Quantitativ - kontinuierlich	keine	avg_glucose_level
bmi	Quantitativ - Deskriptiv	keine	bmi
smoking_status	Qualitativ - Nominal	OneHotEncoder	smoking_status_1, smoking_status_2, smoking_status_3, smoking_status_4
stroke	Qualitativ - Binär	keine	stroke

Tabelle 1: Spaltenübersicht mit zugehörigen Datentypen

3.1.4 EDA

Bei der EDA wird als erstes geschaut, wie viele Personen im Datensatz einen Schlaganfall hatten und dabei das im vorherigen Kapitel bereits angesprochene Ungleichgewicht der Klassen identifiziert.

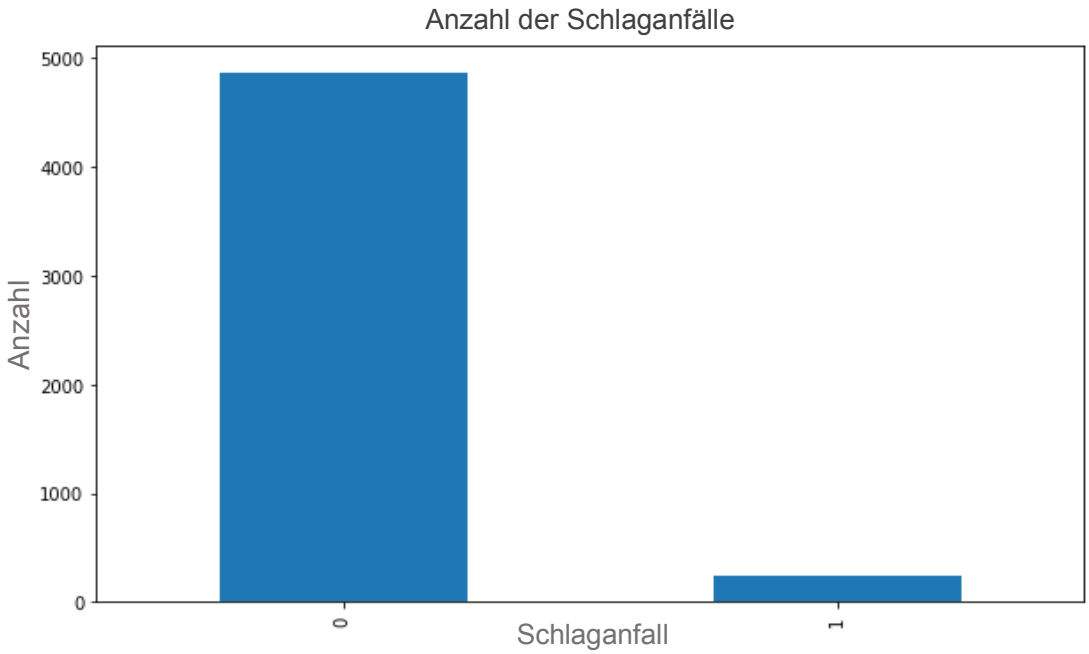


Abbildung 6: Vergleich der Personen mit (1) und ohne (0) Schlaganfälle

Es kommt gerade bei medizinischen Datensätzen häufig vor, dass es bei den Merkmalsausprägungen der Zielvariable ein Ungleichgewicht gibt. Dabei muss auch hervorgehoben werden, dass die Gewichtung zwischen den Ausprägungen der Zielvariable nicht gleich ist. So ist es meistens weniger schlimm, fälschlicherweise eine Diagnose zu geben für eine Krankheit, die nicht besteht, anstatt eine schwere Krankheit als nicht vorhanden zu diagnostizieren, also Gesundheit zu attestieren[31].

Da der BMI in der Regel ein guter und einfacher Indikator für den allgemeinen Körperbau des Patienten ist, werden die Durchschnittswerte bei diesem Feature analysiert. Dabei wird zuerst der globale Durchschnitt über alle Samples der Dateien berechnet, und im weiteren Schritt wird in unterschiedliche Gruppen geclustert.

[31] Belarouci & Amine Chikh, Medical imbalanced data classification, 2017, S. 116ff.

mean bmi	work_type
global 28.893236911794666 male 28.64793635007459 female 29.065757680358992	Govt_job 30.522063 Never_worked 25.545455 Private 30.304625 Self-employed 30.211871 children 20.038003

Tabelle 2: Durchschnittswerte bei BMI

Dabei kann ein geringer Unterschied von 1.46 % erkannt werden: Die Gruppe der Frauen hat mit 29.07 gegenüber den Männern mit 28.65 einen leicht höheren BMI. Bei den verschiedenen Klassen in der Spalte *work_type* gibt es deutlichere Unterschiede. Mit 20.04 stehen Kinder alleine durch die zu erwartenden deutlichen anderen körperlichen Ausprägungen an niedrigster Stelle. Zwischen den Arbeitenden in den Kategorien (*Govt_job, Private und Self-employed*) gibt es keine signifikanten Unterschiede. Wird diese Gruppe jedoch zusammengefasst, fällt eine deutliche Differenz zu nicht arbeitenden Personen mit einem durchschnittlichen BMI von 25.55 auf.

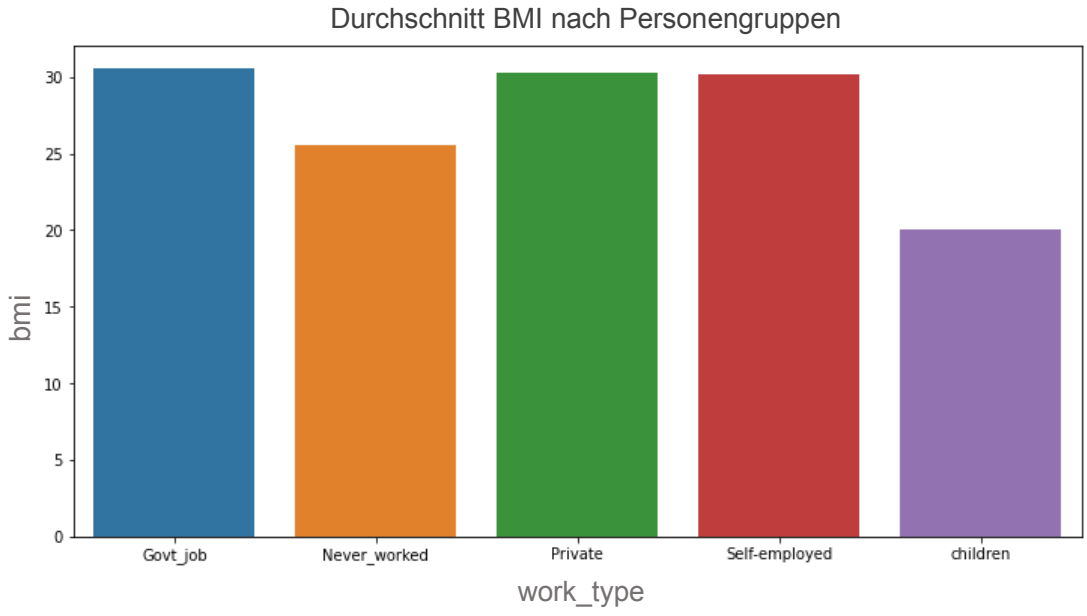


Abbildung 7: Darstellung des Durchschnitts von BMI nach Personengruppen der Spalte work_type

Bei heterogenen Daten ist es außerdem wichtig, wechselseitige Beziehungen und Korrelationen zwischen den einzelnen Werten zu identifizieren. Dafür wird eine Korrelationsmatrix generiert, welche die Korrelation zwischen den jeweiligen Features misst und als Korrelationswert ausgibt. Dabei wird bei einem Korrelationswert von über 0 von einer positiven Korrelation gesprochen. Das heißt, wenn Feature 1 steigt, steigt auch das Feature 2 an. Sollte der Korrelationswert unter 0 sein, wird von einer negativen Korrelation gesprochen. Das bedeutet, wenn Feature 1 abfällt, sinkt auch Feature 2. Die höchste Korrelation mit einem Korrelationswert von 0.33 ist zwischen den Variablen aus der Spalte “age” und “bmi” zu erkennen. Der leicht erhöhte BMI bei fortschreitendem Alter deckt sich auch mit einer Statistik über die Verteilung des BMI nach Altersgruppen aus dem Jahre 2017 (Link). Dieser leichte Anstieg konnte auch mit einem Histogramm über die Häufigkeitsverteilung der Werte für BMI nach Alter aufgezeigt werden.

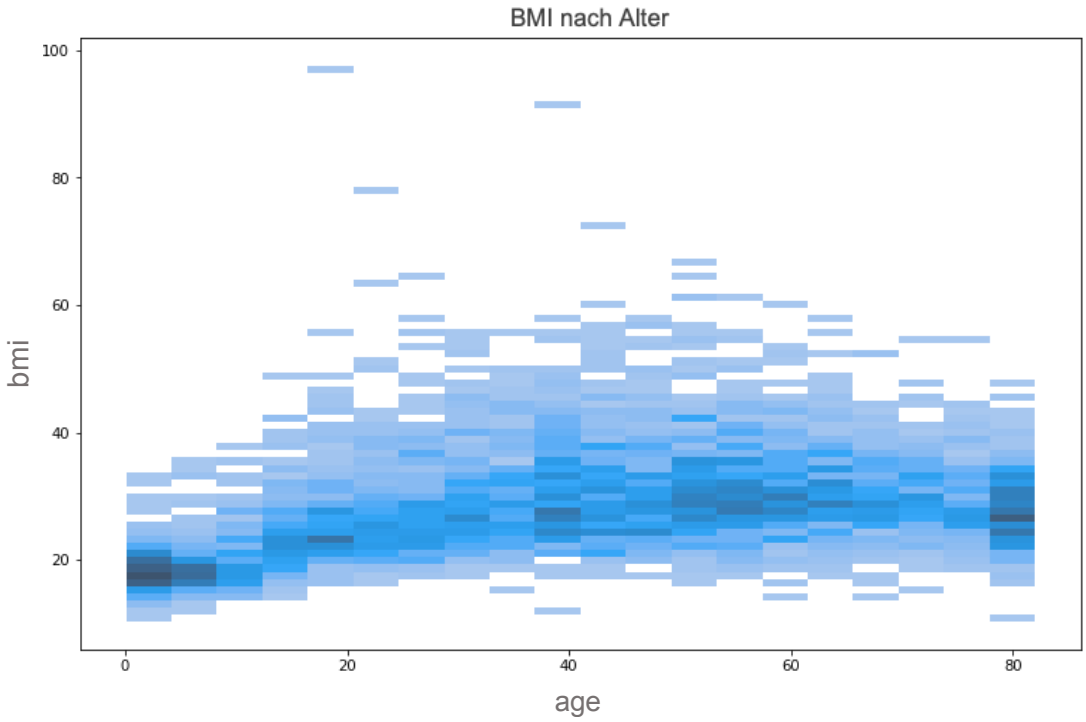


Abbildung 8: Darstellung der Verteilung von BMI nach Alter

Bei den Werten aus der Spalte “work_type” konnte aber auch festgestellt werden, dass die jeweiligen Kategorien nicht gleich verteilt sind. Mehr als 2500 Datensamples fallen in die Kategorie “private”. Die Kategorien “self-

employed", "children" und "govt_job" haben alle mehr als 500 Datensamples, und mit 22 Datensätzen ist die Kategorie "never_worked" weit unter dem Durchschnitt vertreten.

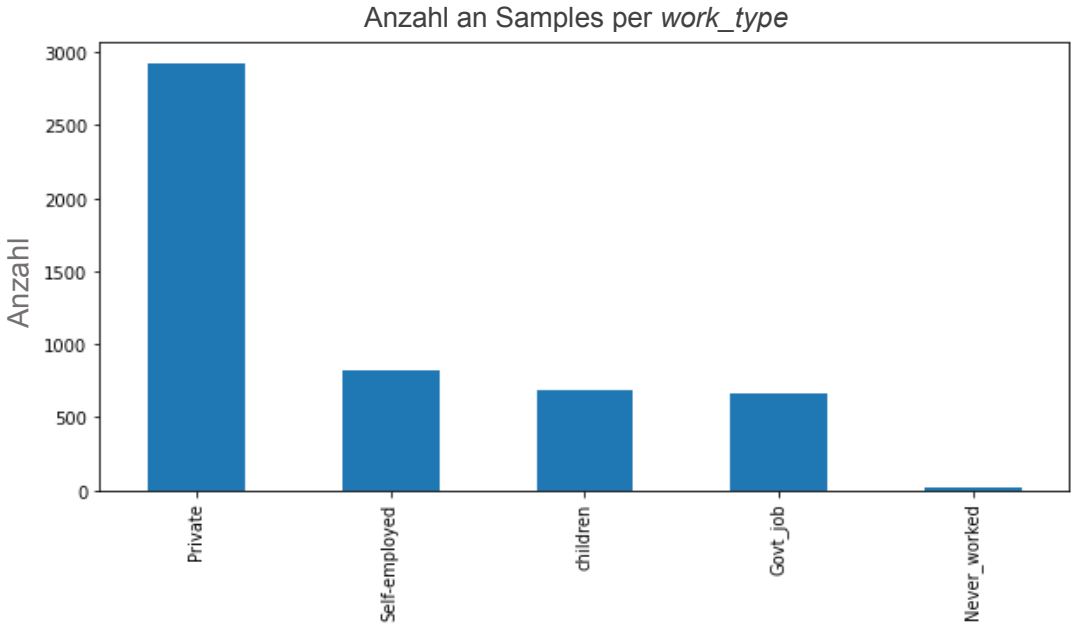


Abbildung 9: Anzahl an Samples in den unterschiedlichen Kategorien der Spalte work_type

Bei den Kategorien in der Spalte "smoking status" überwiegen die Datensamples mit der Kategorie "never_smoked" mit einem Wert von über 1750. Die Kategorien "formely_smoked" und "smokes" sind niedriger, aber ähnlich groß. Dazwischen liegt die Kategorie „Unknown" mit 1500 Samples.

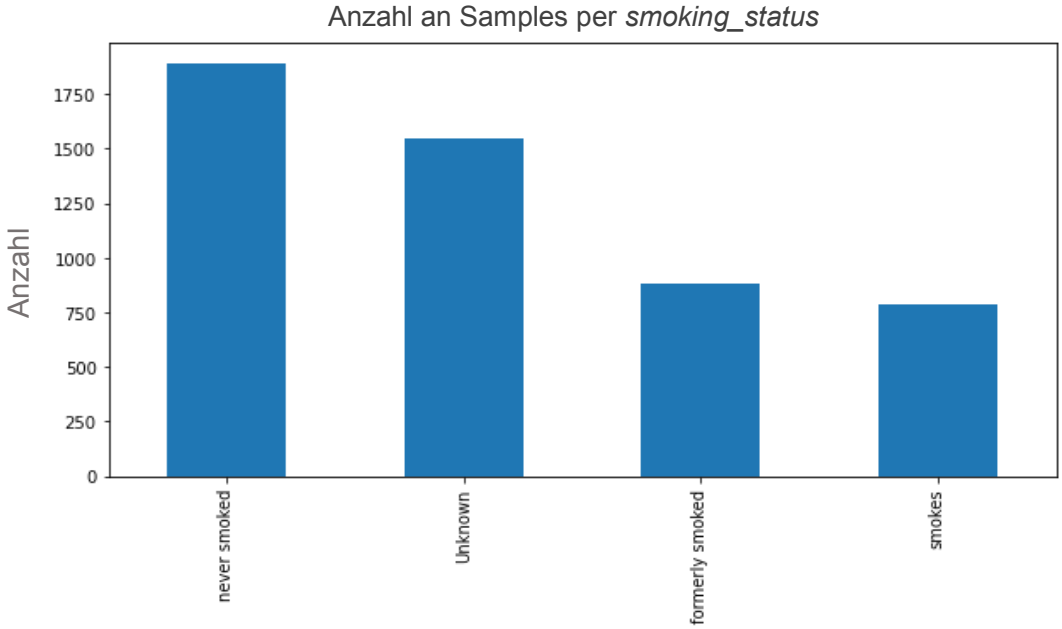


Abbildung 10: Anzahl an Samples in den unterschiedlichen Kategorien der Spalte smoking_status

Bei der Betrachtung der Zielvariablen und in Bezug auf die unterschiedlichen Kategorien bei der Spalte “work_type” fällt auf, dass es bei der Zielvariablen nicht nur ein globales Ungleichgewicht in dem Datensatz gibt, sondern auch das Vorkommen eines Schlaganfalls in den Kategorien “children” und “never_worked” kaum bis gar nicht vorkommt. Bei den Datensamples der Kategorie “children” gibt es lediglich 2 Samples, welche als stroke=1 eingeordnet sind. Bei der Kategorie “never_worked” befindet sich kein Sample mit der Zielvariable stroke=1.

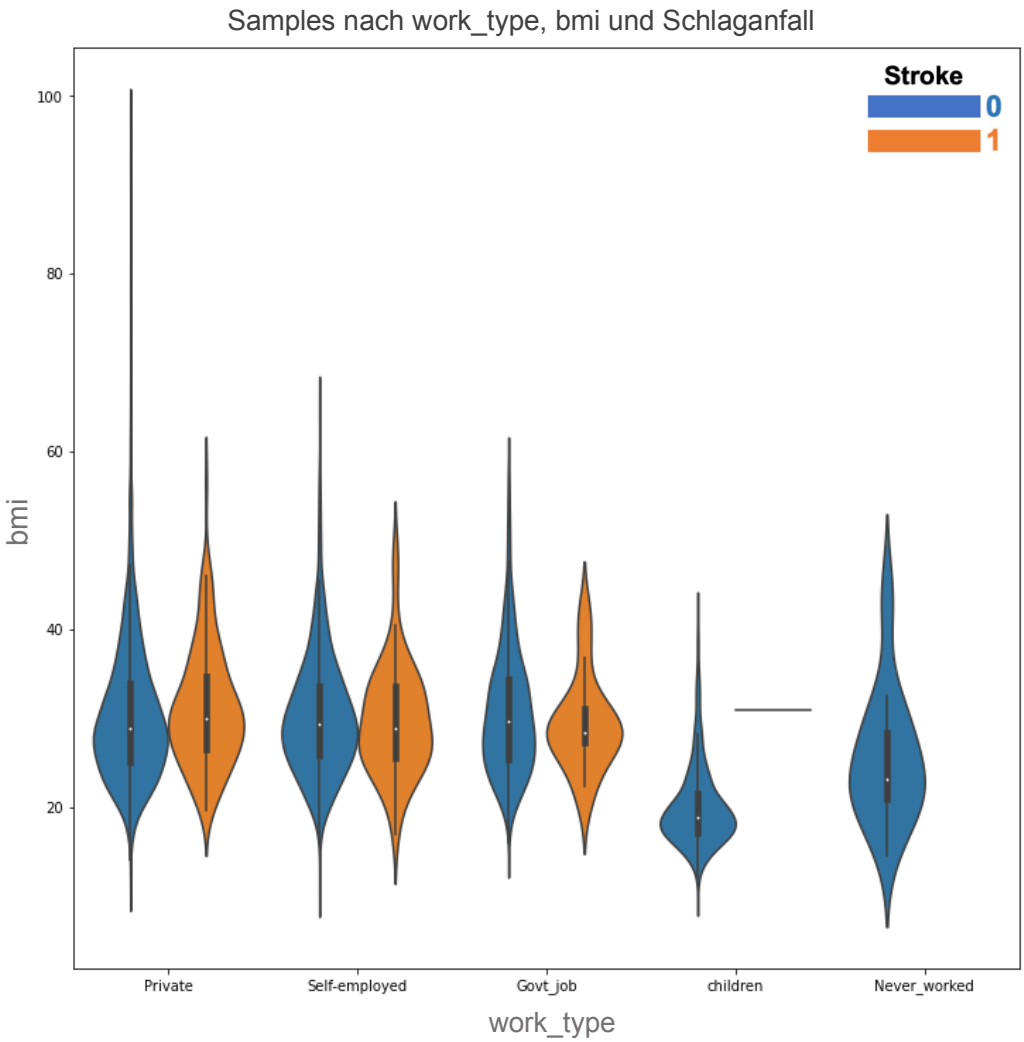


Abbildung 11: Unterscheidung nach Schlaganfall und kein Schlaganfall bezogen auf work_type und bmi

Trotz fehlender Samples mit der Kategorie “*never_worked*” lässt sich daraus nicht ableiten, dass weniger Stroke-Fälle in dieser Gruppe auftreten. Es könnten auch zu wenig Beispiele in den Daten vorhanden sein. Die

Gruppe der “never_worked” könnte nicht ausreichend repräsentativ für die Gesamtmenge der “*never_worked*” zu sein.

Da das Arbeiten mit einem unbalancierten Datensatz die Performance der verwendeten Machine Learning Modelle negativ beeinflussen kann, werden die Daten im Vorfeld mit einer *Oversampling-Technik* bearbeitet. Dabei werden die Klassen, die im Datensatz nur gering vertreten sind - im Beispiel dieser Arbeit ist das die Klasse stroke = 1, also alle Fälle, die einen Schlaganfall erlitten haben - synthetisch vergrößert. Dabei wird versucht, aus den vorhandenen Daten und deren Verteilung neue Datensamples zu generieren, die aufgrund der Verteilung der einzelnen *features* als realistisches Bild in den Datensatz hineinpassen. Dafür wird die Methode *SMOTE - Synthetic Minority Oversampling Technique* - verwendet. Sie sucht zwei ähnliche Datensamples, generiert zwischen ihnen eine Linie und erzeugt dann eine bestimmte Anzahl an Datensamples auf dieser Linie. Diese unterscheiden sich numerisch von den zwei realen ausgewählten Datensamples, jedoch besitzen sie von den features her eine starke Ähnlichkeit zu diesen. Für die *SMOTE* Methode wird die Python Library *imblearn* verwendet[32].

3.1.5 Data splitting (train, dev, test)

Damit die zwei unterschiedlichen Modelle (Machine Learning und Deep Learning) sowohl während des Trainings evaluiert werden können als auch nach dem Training an einem völlig unbekannten Datensatz die Performance getestet werden kann, wird der vorhandene Datensatz in drei unterschiedliche Datensätze aufgeteilt. Dabei wird eine Split-Verteilung von 90 % Trainings-Daten und 10 % Test-Daten gewählt. Während des Trainings wird das Development-Set automatisch mit einem Verhältnis von 10% zum Trainingsdatensatz von der Python Library *Tensorflow keras* erzeugt. Dies wird vor allem für die Darstellung der Lernkurve des künstlichen neuronalen Netzwerkes benötigt. Diese Lernkurve ermöglicht die Identifikation von *High Bias* und *High Variance* Problemen. Dabei

[32] Yu, & Zhou, Survey of Imbalanced Data Methodologies, 2021, S. 1f

beschreiben die Probleme *High Variance* und *High Bias* unterschiedliche Ausprägungen bei den Problemen, die vorherrschen können, wenn ein künstliches neuronales Netzwerk trainiert wird. Dies ermöglicht eine systematische Evaluierung bei der Optimierung des Trainings und des Modells. Die folgende Tabelle zeigt die formale Definition der unterschiedlichen Muster und grundsätzliche Schritte, die bei der Behebung helfen können[33].

	High Bias	**High Variance "Overfitting"**	**Optimal**
Train-Error	Hoch	Niedrig	Niedrig
Test-Error	Hoch	Hoch	Niedrig
Mögliche Schritte	Komplexeres Modell, länger trainieren	Mehr Daten, mehr Regularisierung	_____

Tabelle 3: Unterschiedliche Muster beim Verlauf der Lernkurve

3.1.6 Normalization

Um das Training zu beschleunigen und um die unterschiedliche numerische Verteilung der jeweiligen Spalten aufzulösen, werden die Daten in einem weiteren Schritt normalisiert. Dafür wird der *StandardScaler* der *Sklearn* Library verwendet. Dieser funktioniert, indem für jede Spalte jeweils die Differenz zum Durchschnittswert ins Verhältnis zur Standardabweichung in der jeweiligen Spalte gebracht wird. Die formale Definition für den *StandardScaler* ist $x' = {}^{x-\mu}$. Danach haben die Spalten jeweils einen Durchschnitt von 0 und eine Standardabweichung von 1. Dabei ist es wichtig, die Informationen über den erhaltenen Wert des Durchschnitts μ und den Wert der Standardabweichung σ nur aus den Trainings-Daten zu erlernen und diese dann auf die Test-Daten anzuwenden. Würden diese Werte aus dem globalen Datensatz berechnet werden, würden dadurch Informationen aus den Trainings-Daten in die Testphase des Modells leaken. Dadurch kann die Performance des Trainings verfälscht werden, weil Informationen über Verteilung und Beständigkeit der Daten dem Modell bereits während der Trainingsphase bekannt wären. Mit den erlernten

[33] Dar, Muthukumar, & Baraniuk, An Overview of the Theory of Overparameterized Machine Learning, 2021, S. 12ff.

Werten aus den Trainingsdaten für den *StandardScaler* werden beide Datensätze daraufhin normalisiert[34]

3.2 Training

3.2.1 Loss function

Die Kostenfunktion gibt den jeweiligen Fehler des Modells an den Daten aus. Grundlegend ist es das Ziel, während der Trainingsphase diese Kostenfunktion zu minimieren, um einen möglichst kleinen Error zu haben. Bei dem in dieser Arbeit zu lösenden Problem der Vorhersage des Schlaganfallrisikos handelt es sich um eine binäre Klassifikation. Das bedeutet, dass die Vorhersage entweder eine 1 oder eine 0 ist. Für binäre Klassifikation wird standardmäßig die Kostenfunktion *Binary Cross Entropy* genutzt. Diese wird auch *Log Loss* genannt, da der Fehler aus einer logarithmischen Kurve von 0 bis 1 dargestellt wird. Die formale Definition für die Kostenfunktion *Binary Cross Entropy* ist[35]:

$$L_{Log}\,(y, p) = -\,(y\ log(p) + (1 - y)\ log\ (1 - p))$$

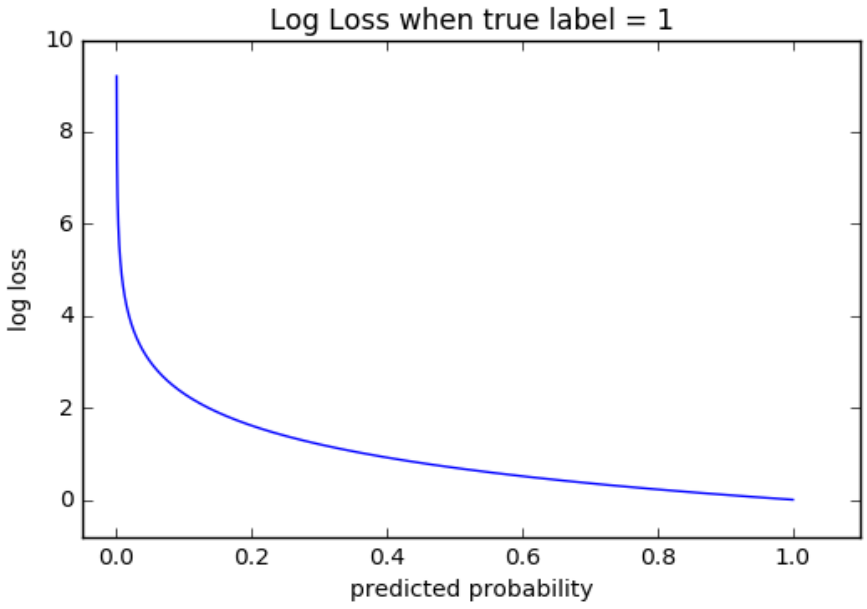


Abbildung 12: Der Bereich möglicher Verlustwerte bei wahrheitsgemäßer Beobachtung

[34] N, G, & M, Data Wrangling and Data Leakage in Machine Learning for Healthcare, 2018, S. 553ff.
[35] Machine Learning Glossary, 2022

Logistische Regression

3.2.1.1 Hyperparameter (Regularisierung)

Das Modell der Logistischen Regression wird mit den Standardeinstellungen der Machine Learning Library implementiert. Dabei wird standardmäßig die L2-Regularisierung genutzt und 100 Iterationen bei der Optimierung des Modells durchgeführt[36]:

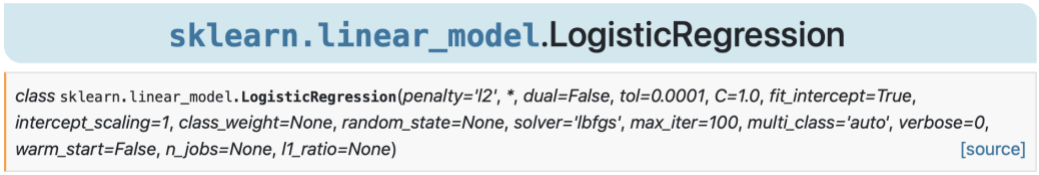


Abbildung 13: Logistic Regression classifier

3.2.1.2 Trainingsverlauf (Learning curve)

Die Performance der Logistischen Regression und des Deep Learning Modells ist nicht alleine davon abhängig, welche Wahrscheinlichkeiten durch die *Sigmoid-Funktion* am Ende ausgegeben werden. Der nächste Schritt ist die weitere Verarbeitung dieser kontinuierlichen Werte. Bei einer hier vorliegenden binären Klassifikation macht es einen großen Unterschied, wo der Schwellenwert bei der Wahrscheinlichkeit ist, der darüber entscheidet, ob die Wahrscheinlichkeit als 1 oder als 0 interpretiert werden soll. Standardmäßig ist dieser Schwellenwert bei 0.5, was auch dem mathematischen Rundungsverfahren gleicht. Um einen optimialen Schwellenwert zu finden, wird in dieser Arbeit mit einer *ROC-Curve* gearbeitet. Von dieser lässt sich die Performance des Modells, gegliedert nach den unterschiedlichen Schwellenwerten, ablesen. Entsprechend wird die TPR - True Positive Rate gegen die FPR - False Positive Rate angezeigt.

[36] Scikit-learn.org, 2022

3.2.2 Deep Learning (*TensorFlow*)

3.2.2.1 Hyperparameter

Bei dem Deep Learning Modell wird der *Optimizer* Adam verwendet. Da die Datensamples nicht viel Speicher verbrauchen, kann eine größere *Batch Size* mit dem Parameter auf 7862 verwendet werden. Damit enthält bereits ein Batch die gesamten Samples des Datensatzes. Das neuronale Netzwerk besteht aus insgesamt 3 Layers. Daraus resultieren 1 Input-Layer, 1 Hidden-Layer und 1 Output-Layer. Der Hidden-Layer besteht aus 16 Units. Er wird beim Forwarding mit der *ReLU* Aktivierungsfunktion aktiviert. Da der Wert nach dem Output-Layer zwischen 0 und 1 sein muss, damit er darauffolgend als Wahrscheinlichkeit für die Klasse 1 oder Gegenwahrscheinlichkeit für die Klasse 0 interpretiert werden kann, wird der Output-Layer mit der Sigmoid-Funktion aktiviert. Während des Trainings wird außerdem die *accuracy* des Modells ausgegeben, um während des Trainings nicht nur die Entwicklung des Trainings- und Validierungs-Loss zu erkennen, sondern eine weitere Metrik zur Hand zu haben. Damit hat das Modell insgesamt 305 lernbare Parameter.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 16)                288

 dropout (Dropout)           (None, 16)                0

 dense_1 (Dense)             (None, 1)                 17

=================================================================
Total params: 305
Trainable params: 305
Non-trainable params: 0
_________________________________________________________________
```

Abbildung 14: Zusammenfassung des Modells mit der Tensorflow Library

3.2.2.2 Trainingsverlauf (learning curve)

Wie bereits im Kapitel Data Splitting angesprochen, ist die Lernkurve signifikant für die Einordnung des Modells während und nach dem Trainingsprozess. An der Lernkuve des verwendeten Modells kann man erkennen, dass sowohl der Trainings- als auch der Validierungs-Loss stetig sinkt und es weder ein *High Bias* noch ein *High Variance* Problem gibt. Kombiniert mit der *accuracy* ist zu erkennen, ob das Modell die erwartete Performance erfüllt. Das Modell wird über 272 Epochen trainiert.

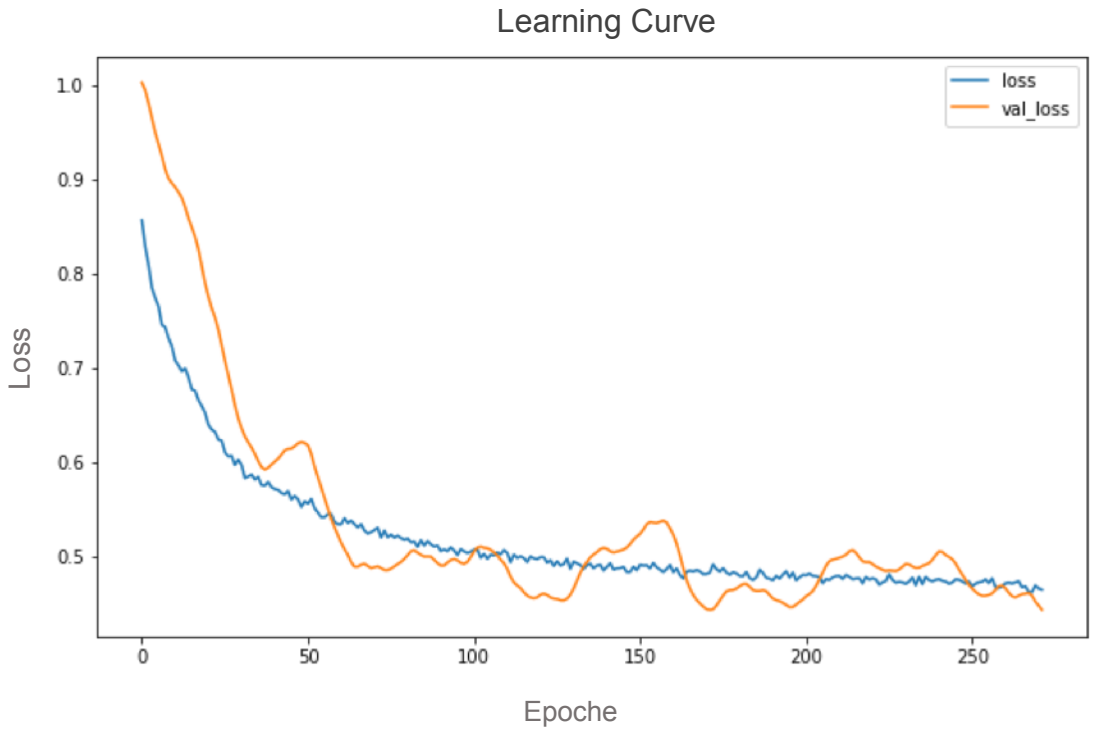


Abbildung 15: Trainingsverlauf des künstlichen neuronalen Netzwerkes mit Angabe des Loss an den Trainingsdaten und den Validierungsdaten

3.2.2.3 Regularisierung

Als Regularisierungsmethode wird in der Netzwerk-Architektur ein *Dropout-Layer* verwendet. Dabei wird ein Parameter von p=0.5 gewählt, damit während des Trainings 50 % aller Aktivierungen vor dem Output-Layer des Modells beim Forwarding ignoriert werden. So wird ein reglarisierender Effekt ausgelöst.

4 Evaluation

4.2 Allgemeines zu Metriken

Weil es vor allem im Bereich der Klassifikationsprobleme im Machine Learning unzählige Metriken gibt, ist es umso wichtiger, sich auf wenige zu fokussieren. Während im Training mit der *accuracy* gearbeitet wird, um die Entwicklung des neuronalen Netzwerkes grob einzuordnen, wird im weiteren Schritt mit der Metrik *Recall* und *Precision* gearbeitet.

4.3 ROC-AUC Curve, Schwellenwert, Confusion Matrix

Aus der ROC-Curve werden für die beiden Modelle die folgenden Parameter abgelesen. Bei der Logistischen Regression steigt die TPR anfangs im Verhältnis zur FPR stark an. Bei ungefähr 0.2 steigt die FPR stärker an, ohne dass die TPR weiter zunimmt.

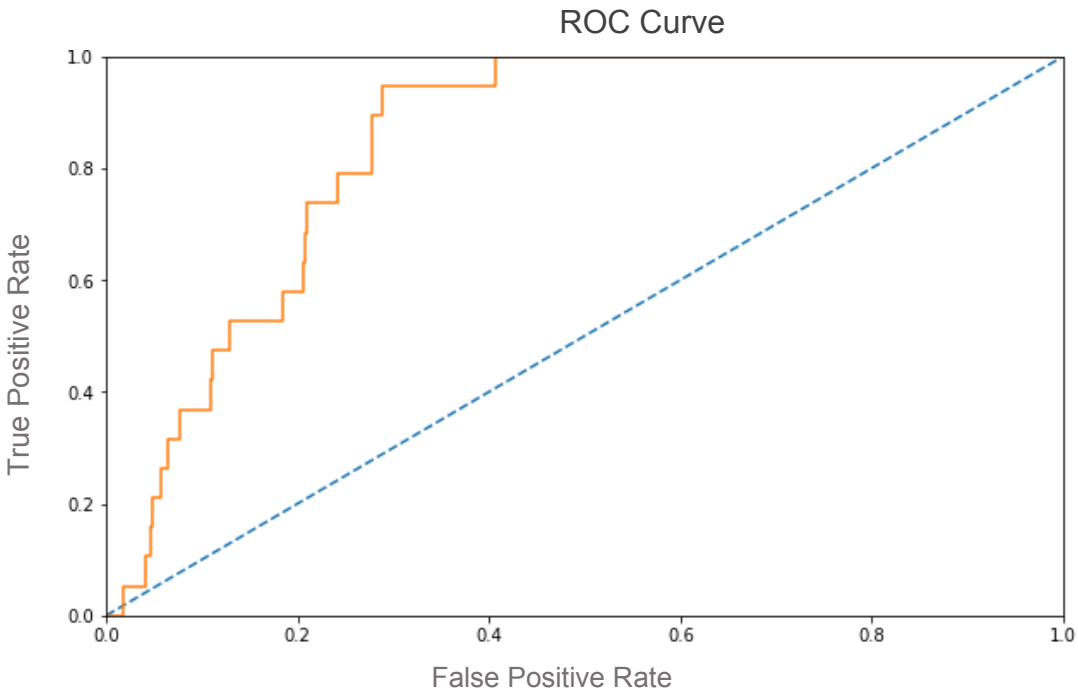


Abbildung 16: True-Positive-Rate im Verhältnis zur False-Positive-Rate bei Logistischer Regression mit Angabe der unterschiedlichen Schwellenwerte nach Aufrunden der Wahrscheinlichkeiten.

Bei dem neuronalen Netzwerk steigt die TPR im Verhältnis zur FPR anfangs stark an. Erst zwischen 0.2 und 0.4 steigt die FPR stärker an, ohne dass die TPR weiter zu nimmt.

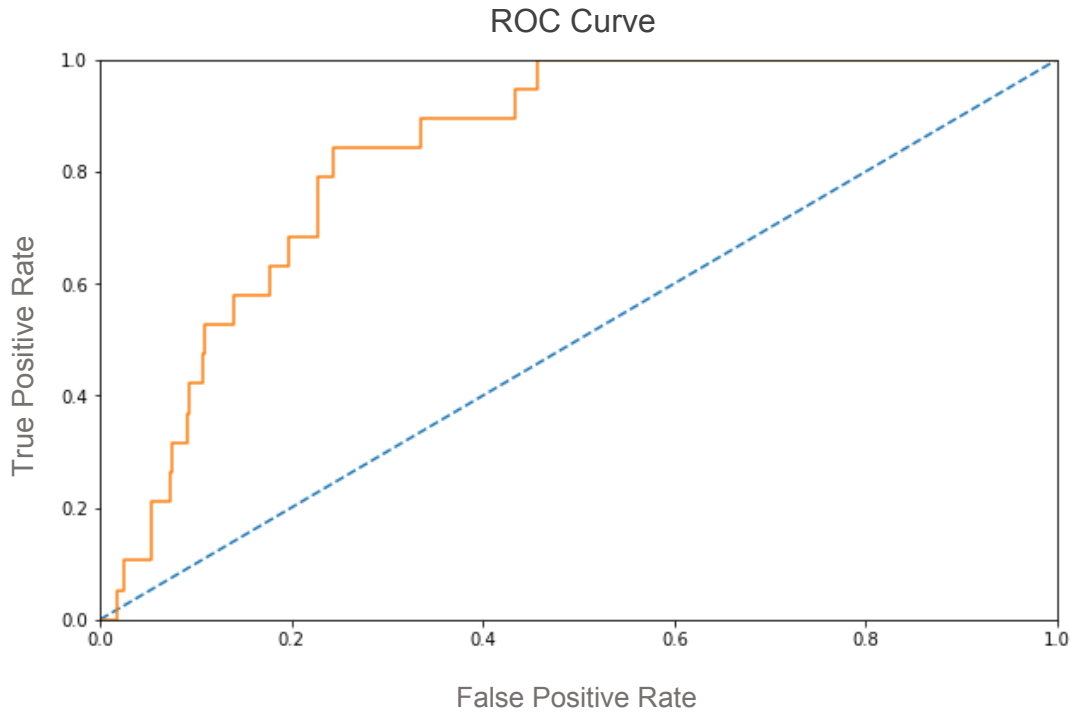


Abbildung 17: True-Positive-Rate im Verhältnis zur False-Positive-Rate bei Neuronalem Netz mit Angabe der unterschiedlichen Schwellenwerte nach Aufrunden der Wahrscheinlichkeiten

Außerdem wird eine *Precision-Recall-Curve* implementiert. Dabei werden die zwei Metriken *Precision* und *Recall* hinsichtlich ihrer Ergebnisse gegeneinander auf einem Graph dargestellt.

Bei der *Precision-Recall-Kurve* ist bei beiden Modellen abzulesen, dass schon bei einer leichten Erhebung über dem Wert 0 aufgerundet werden sollte. Dabei ist die *Precision* zwar niedrig, der *Recall* jedoch hoch. Somit werden die meisten Fälle von tatsächlich vorkommenden Schlaganfällen gefunden. Mit der *Confusion Matrix* können wir den Schwellenwert zusätzlich optimieren. Ziel ist es, hierbei eine gute Mitte zwischen *Precision* und *Recall* zu finden. Das heißt, so viele Schlaganfälle wie möglich zu entdecken und dabei nicht zu viel *Precision*, also falsche Diagnosen für einen Schlaganfall, zu haben.

4.3 Vergleich der beiden Methoden (ML und DL) anhand dieser Metriken

Anhand der unterschiedlichen Evaluierungsmethoden kann festgestellt werden, dass die Logistische Regression nur eine leicht schlechtere Performance als Deep Learning hat. Damit bestätigt sie aktuelle Erfahrungen im Bereich Machine Learning, dass heterogene Datensätze künstliche neuronale Netzwerke vor eine große Herausforderung stellen. Bei den Metriken ist zu sehen, dass beide Modelle bei einem ungefähren *Recall-Wert* von 80 % Abweichungen haben. Dabei zeigt sich das neuronale Netzwerk der Logistischen Regression leicht überlegen.

	Logistische Regression	KNN
Accuracy	0.73	0.76
Precision	0.10	0.12
Recall	0.79	0.84

Tabelle 4: Vergleich der Logistischen Regression und Deep Learning anhand der Metriken

Beim Vergleich beider *Confusion Matrizen* ist zu erkennen, dass beide Modelle bei den positiven Labels, also allen Samples, bei denen ein Schlaganfall (y = 1) vorliegt, im Testdatensatz eine ähnliche Performance haben. Beispielsweise ist das Verhältnis zwischen TP und FN bei beiden Modellen ähnlich, woraus sich schließen lässt, dass beide Modelle mit gleicher Präzision einen Schlaganfall vorhersagen können. Bei den negativen Labels, also allen Datensamples, bei denen kein Schlaganfall vorliegt (y = 0 im Testdatensatz), ist eine größere Differenz zwischen den beiden Modellen zu erkennen. Das Künstliche Neuronale Netz hat 120 FP und 372 TN. Das heißt, es weist immer noch eine hohe Fehlerquote bei den gesunden Patienten auf und kann damit vor allem gesunde Patienten nicht ausreichend gut klassifizieren. Die Logistische Regression hat hingegen 135 FP und 357 TN und damit ungefähr das gleiche Ergebnis wie das Künstliche Neuronale Netz.

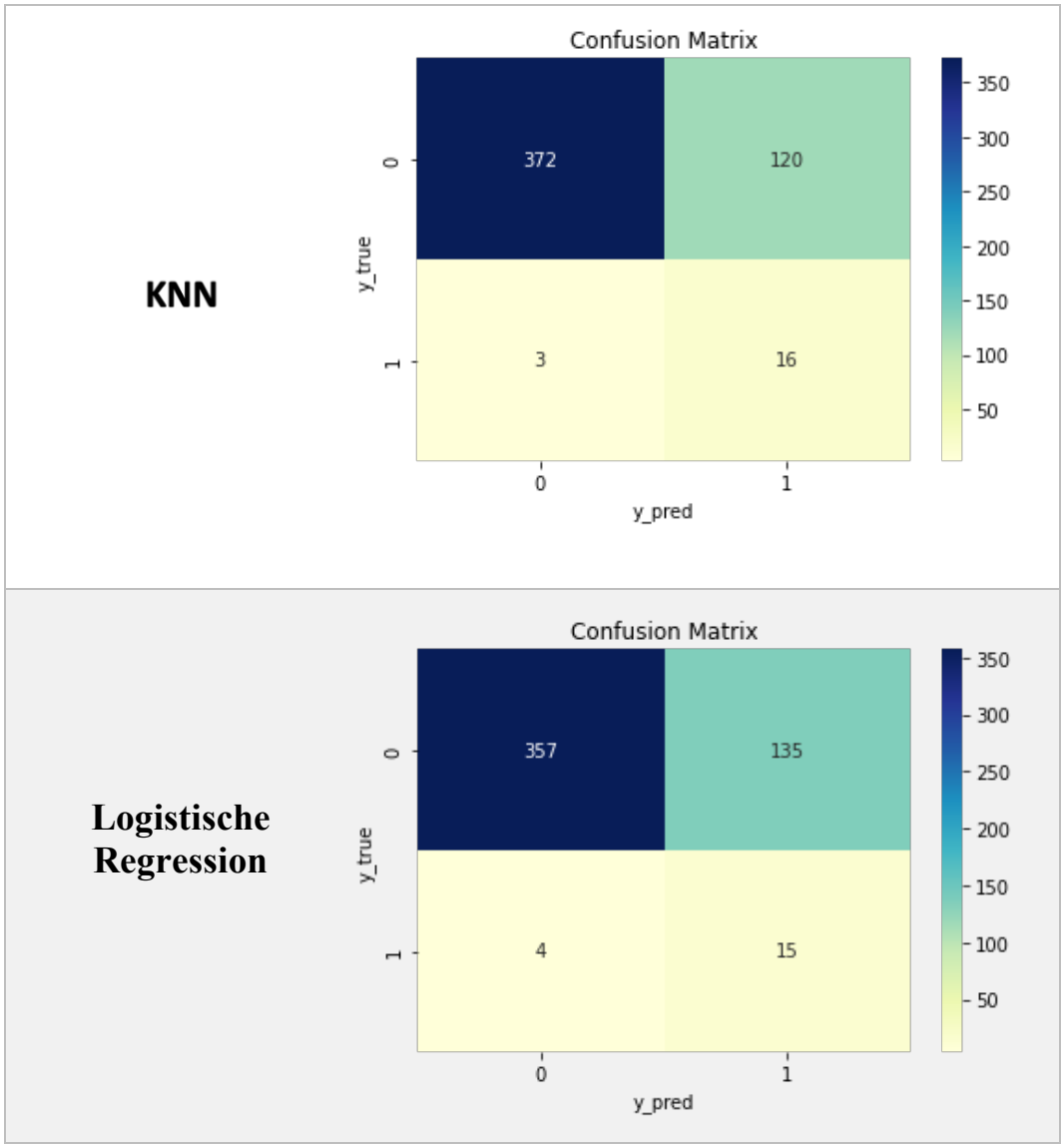


Abbildung 18: Zwei Konfusion Matrizen der Modelle Logistische Regression und KNN im Vergleich

5 Fazit

Als Schlussfolgerung lässt sich aus dieser Arbeit ableiten, dass die hier verwendete klassische Architektur eines künstlichen neuronalen Netzwerkes bei diesen tabellarischen und heterogenen Datensätzen Probleme aufweist. Ein signifikanter Performance-Gewinn des Künstlichen Neuronalen Netzes im Vergleich zu einem klassischen Machine Learning Modell wie der Logistischen Regression ist nicht erkennbar.

Außerdem haben sich in den vergangenen Jahren die Erkenntnisse erhärtet, dass *tree-based models* bei tabellarischen Daten das Mittel der Wahl sind. Deep Learning wird vor allem im Bereich der homogenen Daten wie Bildern, Texten oder Sprache verwendet[37]. Damit hat das neuronale Netzwerk im vorliegenden Projekt nicht nur eine schlechtere Performance als erwartet, sondern ist von der Implementierung her weitaus aufwändiger. Im Gegensatz zur Logistischen Regression hat das künstliche neuronale Netzwerk viel mehr Hyperparameter, und es bestehen sehr viele Einstellungsmöglichkeiten für die Netzwerkarchitektur. Damit steigt auch die Fehleranfälligkeit. Ein weiterer Punkt ist, dass neuronale Netzwerke, vor allem der Bereich Deep Learning, bisher sehr große Datenmengen voraussetzen. Mit einem Datensatz von 5110 Samples wird in dieser Arbeit ein relativ kleiner Datensatz verwendet.

Um die Performance von Deep Learning Modellen auch bei geringen Datenmengen signifikant zu verbessern, müssen neue Wege gefunden werden. Dies ist vor allem im medizinischen Bereich der Fall, wo kleinere Datensätze mit vielen Variablen die Regel sind. Um hier Deep Learning sinnvoll einsetzen zu können, ist es eine Möglichkeit, neue algorithmische Erkenntnisse zu gewinnen. Zusammenfassend lässt sich sagen, dass das hier erzielte Ergebnis, dass ein einfaches Künstliches Neuronales Netzwerk zwar bessere Ergebnisse als eine Logistische Regression erzielt, jedoch diese Differenz nicht groß genug ist, um es vorzuziehen, auch das Ergebnis der Studie von Issitt et al. ist[38].

[37] Borisov, et al., Deep Neural Networks and Tabular Data: A Survey, 2022, S. 3ff.

[38] Issitt, et al., Classification Performance of Neural Networks Versus Logistic Regression Models: Evidence From Healthcare Practice, 2022

5.1 Was kann verbessert werden?

Die jüngsten Erfolge in anderen Bereichen zeigen, wie viel Potential in diesen Modellen steckt. Daher ist es wichtig, auch die Herausforderungen der tabellarischen Daten auf lange Sicht zu meistern. Tabellarische Daten gibt es in allen Lebensbereichen. Viele Daten liegen in dieser Struktur vor. Bei dem vorliegenden Datensatz könnte versucht werden, über Methoden aus dem Bereich *Feature Engineering* weitere *Features* zu entwickeln, welche einen signifikanten Vorteil über die Vorhersage der Zielvariable bilden.

5.2 Ausblick

Wie in publizierten Arbeiten bereits erwähnt, sind weiterführende Entwicklungen im Bereich der hybriden Architekturen zu erwarten. Dabei geht es darum, die klassischen Deep Learning Architekturen mit Komponenten aus den *tree-based* Lernalgorithmen zu koppeln, um damit das optimale Potential aus beiden Bereichen herauszuholen.
Ein anderer Ansatz könnte die Verwendung von Transformern sein, welche durch die eingebauten Aufmerksamkeits-Layers ein besseres *encoding* für die heterogenen *Features* erzeugen könnte.
Da Deep Learning Modelle auf große Datenmengen angewiesen sind, gibt es bereits Möglichkeiten, mit Techniken wie *Synthetic Data* Generation, *GANs* und Data Augmentation den verwendeten Datensatz künstlich zu vergrößern. Kleinere Datensätze sind dann kein Nachteil mehr[39].
Im Unterschied zu Logistischer Regression ist Deep Learning in ständiger Weiterentwicklung begriffen. Perspektivisch dürfte sich Deep Learning daher auch in der hier untersuchten Fragestellung überlegen zeigen.

[39] Borisov, et al., Deep Neural Networks and Tabular Data: A Survey, 2022, S. 9f.

6 Literaturverzeichnis

Belarouci, S., & Amine Chikh, M. (2017). Medical imbalanced data classification. *Advances in Science Technology and Engineering Systems Journal*, 116ff.

Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M., & Kasneci, G. (Feb 2022). *Deep Neural Networks and Tabular Data: A Survey.* Abgerufen am Jun 2022 von arxiv.org: https://arxiv.org/pdf/2110.01889.pdf

Buxmann, P., & Schmidt, H. (2021). *Künstliche Intelligenz.* Berlin: Springer Gabler.

Dar, Y., Muthukumar, V., & Baraniuk, R. (Sep 2021). *A Farewell to the Bias-Variance Tradeoff? An Overview of the Theory of Overparameterized Machine Learning.* Abgerufen am Jun 2022 von arxiv.org: https://arxiv.org/abs/2109.02355.

F, D., L, J., Shao, Y., A, C., & Zeng-Treitler, Q. (2020). Using Explainable Deep Learning and Logistic Regression to Evaluate Complementary and Integrative Health Treatments in Patients with Musculoskeletal Disorders. *International Conference on System Sciences*, (p. 3265). Hawaii.

fedesoriano. (2021). *kaggle*. Von Stroke Prediction Dataset: https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset abgerufen.

Frick, D., Gadatsch, A., Kaufmann, J., Lankes, B., Quix, C., Schmidt, A., & Schmitz, U. (2021). *Data Science.* Wiesbaden: Springer Vieweg .

Grunert, P. (2021). *Machine Learning und Neuronale Netze.* Wydawnictwo: BMU Media GmbH.

Hude, M. v. (2020). Predictive Analytics und Data Mining. In *Regressions- und Klassifikationsfragestellungen* (S. 5f.). Wiesbaden: Springer Vieweg.

Hung, C.-Y., Chen, W.-C., Lai, P.-T., Lin, C.-H., & Lee, C.-C. (2017). Comparing Deep Neural Network and Other Machine Learning Algorithms for Stroke Prediction in a Large-Scale Population-Based Electronic Medical Claims Database. *Annual International Conference - (EMBC).* Jeju, Korea (South): IEEE.

Issitt, R., Cortina-Borja, M., Bryant, W., Bowye, S., Taylor, A., & Sebire, N. (2. Feb 2022). *Classification Performance of Neural Networks Versus Logistic Regression Models: Evidence From Healthcare Practice.* Abgerufen am Jun 2022 von cureus.com: https://www.cureus.com/articles/79255-classification-performance-of-neural-networks-versus-logistic-regression-models-evidence-from-healthcare-practice

Justus, D., Brennan, J., Bonner, S., & McGough, A. (2018). Predicting the Computational Cost of Deep Learning Models. *International Conference on Big Data (Big Data)* (S. 3874ff.). Seattle, WA, USA: IEEE.

Kalisch, M., & Meier, L. (2021). Logistische Regression. In *Klassifikation* (S. 5f.). Wiesbaden: Springer Spektrum.

Kaliyadan , F., & Kulkarni, V. (2019). Types of Variables, Descriptive Statistics, and Sample Size. *Indian Dermatol Online J*, 82ff.

Kumar Chauhan , N., & Singh, K. (2018). A Review on Conventional Machine Learning vs Deep Learning. *International Conference on Computing, Power and Communication Technologies (GUCON)* (S. 347). Greater Noida: IEEE.

Machine Learning Glossary. (2022). Abgerufen am Jun 2022 von Loss Functions: https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html

Masoud Rahmani, A., Yousefpoor, E., Yousefpoor, M. S., Mehmood, Z., Haider, A., Hosseinzadeh, M., & Ali Naqvi, R. (21. November 2021). *Machine Learning (ML) in Medicine.* Von Mathematics: https://www.mdpi.com/2227-7390/9/22/2970 abgerufen

N, S., G, S., & M, B. (Nov 2018). Data Wrangling and Data Leakage in Machine Learning for Healthcare. *International Journal of Emerging Technologies and Innovative Research*.

Ongsulee, P. (2017). Artificial intelligence, machine learning and deep learning. *International Conference on ICT and Knowledge Engineering (ICT&KE)* (S. 2 C.). Bangkok, Thailand: IEEE.

Saitoh, K. (2021). Deep Learning from the Basics. Packt Publishing, Limited.

Shen, X., Tian, X., Liu, T., Xu, F., & Tao, D. (2018). Continuous Dropout. *IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS*, 3926.

scikit-learn.org. (2022). (Scikit-learn, Produzent) Abgerufen am Jun 2022 von sklearn.linear_model.LogisticRegression: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Tausendpfund, M. (2020). Fortgeschrittene Analyseverfahren. In *Was ist eine Regression?* (S. 8 -10). Wiesbaden: Springer VS.

Trauth, D., Bergs, T., & Prinz, W. (2021). In *Monetarisierung von technischen Daten* (S. 623). Berlin: Springer Vieweg.

Vakili, M., Ghamsari, M., & Rezaei, M. (27. January 2020). *Performance Analysis and Comparison of Machine and Deep Learning Algorithms for IoT Data Classification.* Von Arxiv: https://arxiv.org/abs/2001.09636 abgerufen

Weiß, C. (August 2018). Logistische Regression. In *Medizinische Statistik.* Mannheim: Springer Nature.

Welsch, A., Eitle, V., & Buxmann, P. (2018). In *Maschinelles Lernen* (S. 370-372). Wiesbaden: Springer.

Yu, L., & Zhou, N. (Apr 2021). *Survey of Imbalanced Data Methodologies.* Von arxiv.org: https://arxiv.org/abs/2104.02240 abgerufen

Eidesstattliche Erklärung

„Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig, ohne fremde Hilfe und nur unter Benutzung der angegebenen Quellen und Hilfsmittel angefertigt habe. Alle Textstellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Quellen entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit hat in gleicher oder in ähnlicher Form keiner anderen Prüfungsbehörde vorgelegen."

Oberhausen, 24,06,2022

Ort, Datum

Unterschrift