BACHELORARBEIT

Entwicklung eines künstlichen neuronalen Netzes zur Feststellung einer genetischen Prädisposition zum Alkoholismus anhand von EEG-Messungen

vorgelegt am 12. März 2024

Alexej Nikulin

ErstprüferIn: Prof. Dr. Sabine Schumann
ZweitprüferIn: Prof. Dr. Larissa Putzar

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Department Medientechnik
Finkenau 35
20081 Hamburg

Zusammenfassung

Alkoholismus ist eine Erkrankung, die nicht nur den Betroffenen selbst, sondern auch ihren Mitmenschen, insbesondere ihren Nachkommen, Schaden zufügt. Eine genetische Veranlagung zur Entwicklung dieser Sucht ist vererbbar und kann mithilfe von EEG-Messungen vor Ausbruch der Krankheit festgestellt werden. Die Automatisierung dieses Prozesses könnte das medizinische Personal entlasten und die Auswertung von mehr Messungen in kürzerer Zeit ermöglichen. Zur Realisierung dieses Lösungsansatzes beschäftigt sich diese Bachelorarbeit mit der Entwicklung eines künstlichen neuronalen Netzes zur Untersuchung des Vorhandenseins von Veranlagungen zum Alkoholismus bei Testpersonen. Ziel der Arbeit ist es, ein automatisiertes System zu konstruieren, das auf Basis von EEG-Messungen eine genetische Prädisposition zur Entwicklung von Alkoholabhängigkeit mit hoher Wahrscheinlichkeit erkennen kann. Die hierfür verwendeten Daten wurden von der University of California, Irvine bereitgestellt und anhand des KDD-Prozesses für das Training des Modells vorverarbeitet. Für den Aufbau des neuronalen Netzes wurde die Feedforward-Architektur verwendet und dessen Hyperparameter mithilfe der Python-Bibliothek Optuna optimiert. Im Laufe des Umsetzungsprozesses wurden mehrere Architektur- und Hyperparameteranpassungen ausprobiert u.a. das Einbauen von Oversampling sowie von Cross-Validation. Insgesamt erreichte das effizienteste Modell eine Genauigkeit von 86% und einen Alkoholiker-Recall von 85%.

Abstract

Alcoholism is a disease that not only harms those affected, but also their fellow human beings, especially their offspring. A genetic predisposition to developing this addiction is hereditary and can be detected using EEG measurements before the onset of the disease. Automating this process could relieve the burden on medical staff and enable more measurements to be analysed in less time. To realise this approach, this bachelor thesis deals with the development of an artificial neural network to investigate the presence of predispositions to alcoholism in test subjects. The aim of the thesis is to construct an automated system that can recognise a genetic predisposition to the development of alcohol dependence with a high probability on the basis of EEG measurements. The data used for this purpose was provided by the University of California, Irvine and pre-processed using the KDD process for training the model. The feedforward architecture was used to build the neural network and its hyperparameters were optimised using the Python library Optuna. In the course of the training process, several architecture and hyperparameter adjustments were tried out, including the adding of oversampling and cross-validation. Overall, the most efficient model achieved an accuracy of 86% and an alcohol recall of 85%.

Inhaltsverzeichnis

Abkürzungsverzeichnis

AAC American Addiction Center

AACAP American Academy of Child and Adolescent Psychiatry

CDC Centers for Disease Control and Prevention

cGAN Conditional Generative Adversarial Network

CNN Convolutional Neural Network

EEG Elektroenzephalografie

FN False Negative

FNN Feedforward Neural Network

FP False Positive

GAN Generative Adversarial Network

GUI Graphical User Interface

IQR Inter Quantile Range

KDD Knowledge Discovery in Databases

KI Künstliche Intelligenz

KNN Künstliches Neuronales Netz

LSTM Long Short-Term Memory

ML Maschinelles Lernen

PCA Principal Component Analysis

RNN Recurrent Neural Network

SCCN Swartz Center for Computational Neuroscience

SOM Self-organizing map

TN True Negative

TP True Positive

WHO World Health Organization

Abbildungsverzeichnis

1 Einleitung

Im Folgenden wird das Thema der Arbeit erläutert und die Relevanz ihrer Bearbeitung dargelegt. Außerdem wird die Struktur der Thesis festgelegt.

1.1 Motivation

Künstliche Intelligenz (KI) ist ein Begriff, der besonders in den letzten Jahren immer mehr an Aufmerksamkeit in der Weltbevölkerung gewonnen hat [Con]. Dieser Terminus wurde jedoch bereits in den 1950er-Jahren geprägt. Bei der als Geburtsstunde der KI geltenden Darthmouth-Konferenz wurde eingehend über die Programmierung von Computern zur Simulation menschlichen Verhaltens und zur Automatisierung von Arbeitsabläufen diskutiert [Klon]. Noch bevor der Ausdruck begründet wurde, zeigte Alan Turing 1936 mit der bekannten Turing-Maschine, dass es möglich ist, mathematische Berechnungen durch die Verkettung von Einzelschritten, einem Algorithmus, von Maschinen ausführen zu lassen [Klon]. Im Laufe der Zeit entwickelten sich die fassbaren Gebiete von ersten dame-spielenden Maschinen (1952) über erste Chatbots mit Eliza (1966) bis hin zu Systemen, die Schachweltmeister besiegen können (1997) [Kar 2023]. Dank beachtlicher technischer Fortschritte wurden inzwischen Maschinen und Technologien entwickelt, die bestimmte alltägliche Prozesse für Menschen schnell und effizient erleichtern [Blo 2023]. Dabei ist mithilfe von KI-Systemen weit mehr möglich, als die Schulhausaufgaben mit ChatGPT zu erledigen oder Smartphones per Gesichtserkennung zu entsperren.
Eines der größten Potenziale liegt hierbei in der Medizin. Schon jetzt werden solche Mechanismen testweise in Krankenhäusern nicht nur zur Diagnose von Krebszellen verwendet und dies teilweise auch erfolgreicher als spezialisierte Radiologen [Kil 2020], auch bei präzisen Operationen wurden schon erste Versuche mit Techniken dieser Art effizient durchgeführt [Maas 2017]. Hierbei reichen die Anwendungsmöglichkeiten von der Überwachung pflegebedürftiger PatientInnen, über Behandlungsvorschläge bis hin zur sofortigen Erkennung eines möglichen Herzstillstandes in Notsituationen [Gib 2020]. Diese Bereiche der heutigen Wissenschaft haben bereits in den 70er-Jahren begonnen, von den Anwendungsvorteilen dieser Art zu profitieren und tun dies mit beachtlichen Fortschritten bis hinein in die heutige Zeit [Ced 2023]. Die betroffenen PatientInnen machen sich dadurch diese Progresse ebenfalls zunutze. Einer dieser Bereiche befasst sich mit dem Thema 'Prävention'. Von besonderer Bedeutung sind dabei die Themengebiete, die Menschen wichtige gesundheitliche Erkenntnisse liefern und mögliche Störungen wie die Entwicklung einer Alkoholsucht verhindern könnten.
Alkoholabhängige Menschen haben einen großen negativen Einfluss auf ihre Mitmenschen, besonders auf ihre Kinder.

Laut mehreren Organisationen und Zentren, wie der World Health Organization (WHO), dem American Addiction Center (AAC) und den Centers for Disease Control and Prevention (CDC), kann exzessiver Alkoholkonsum nicht nur zu familiären Konflikten führen, sondern auch lebenslange psychische Spuren bei ihren Nachkommen hinterlassen [Sha 2023]. Die dabei entstehenden Konfliktpunkte sind sehr vielfältig und äußern sich u.a. durch [Sha 2023]:

- **Eheliche Konflikte**
- **Untreue**
- **Häusliche Gewalt**
- **Finanzielle Instabilität**
- **Ungeplante Schwangerschaft**
- **Stress**
- **Eifersucht**
- **Negativen Einfluss auf Kinder**
- **Scheidung**

Der Einfluss auf Kinder ist hierbei ein Faktor, der als ein eigenständiger Punkt behandelt werden sollte. Nach Angaben der American Academy of Child and Adolescent Psychiatry (AACAP) hat jeder fünfte erwachsene Amerikaner in seiner Jugend mit einem alkoholabhängigen Verwandten zusammengelebt [AACAP 2019]. Der frühe Kontakt mit einer alkoholabhängigen Person kann das Risiko erhöhen, dass das Kind eine ähnliche oder sogar intensivere Beziehung zum Alkohol entwickelt [AACAP 2019]. Hierbei spielt eine große Rolle, dass solche Neigungen zu Suchtverhalten, wie Alkoholismus, genetisch vererbt werden können [T 2020]. Generell ist die Wahrscheinlichkeit, dass Kinder von Personen, die Alkohol missbrauchen, ebenfalls exzessiv Alkohol konsumieren, deutlich höher als bei Kindern von Personen, die keine Alkoholsucht aufweisen [Sha 2023].
Bei der Frage nach den Möglichkeiten, eine solche genetisch bedingte Prädisposition festzustellen, erweisen sich Elektroenzephalografie (EEG)-Messungen als empfindlich für das Vorhandensein von Alkoholismus oder einer familiären Vorgeschichte von Alkoholismus [PoRaKaJoPaBe 2004, Seite 999]. Diese Marker repräsentieren die genetische Anfälligkeit für Alkoholismus bei nicht betroffenen Verwandten von betroffenen Personen [PoRaKaJoPaBe 2004, Seite 1010].

Genau diese Erkenntnis des möglichen Bestehens einer solchen Veranlagung, sowie ein daraus resultierender möglicher Verzicht auf Alkohol, könnte Menschen und ihre Familien vor einem derartigen Schicksal bewahren, ob in physischer, psychischer oder sozialer Hinsicht.
Ein automatisiertes System, das in der Lage ist, das Vorhandensein dieser Disposition präzise und schnell zu erkennen, würde zum Erreichen dieses Ziels enorm beitragen. Es würde nicht nur die EEG-Messergebnisse mit hoher Wahrscheinlichkeit und Sicherheit auswerten, sondern auch das medizinische Fachpersonal bei der Auswertung der Messungen erheblich entlasten und eine größere Anzahl von Probanden in kürzerer Zeit bearbeiten können.
Darüber hinaus könnte ein derartiges System, durch die Verarbeitung vieler Testpersonen, Risikogruppen identifizieren und gezieltere Interventionen ermöglichen.
Da Alkoholismus auch erhebliche Auswirkungen auf die Arbeitsproduktivität [Gal 2024] hat, könnte die frühzeitige Erkennung und Intervention dazu beitragen, die sozioökonomische Belastung zu reduzieren.
Diese Argumente verdeutlichen, dass ein automatisiertes System zur Erkennung einer Veranlagung zum Alkoholismus anhand von EEG-Messungen einen bedeutenden Beitrag zur öffentlichen Gesundheit und Ökonomie leisten kann.
Aus diesen Gründen beschäftigt sich diese Arbeit mit der Entwicklung eines automatisierten Systems zur Erkennung einer solchen Disposition.

1.2 Struktur der Arbeit

Zu Beginn werden die für das Verständnis der Arbeit notwendigen Grundlagen eingeführt und erläutert. Im darauffolgenden Kapitel wird das Konzept vorgestellt, welches die Basis für den Aufbau des Trainingsprozesses bilden soll. In Kapitel 4 erfolgt die Umsetzung des vorgestellten Konzeptes, welches unter gewissen Umständen zur besseren Ergebniserzielung angepasst werden könnte. In Kapitel 5 werden die Ergebnisse des Trainingsprozesses ausgewertet und evaluiert sowie der Gesamtprozess zusammengefasst. Abschließend wird ein Ausblick auf mögliche nachfolgende Arbeiten gegeben.

2 Grundbegriffe

In diesem Kapitel werden die grundlegenden Definitionen der, für diese Arbeit, wichtigen Bereiche erklärt und beleuchtet. Zum größten Teil betreffen diese die Themengebiete der KI, des Alkoholismus und der EEG.

2.1 KI

Zunächst werden die wichtigsten Grundlagen zum Verständnis der Funktionalitäten, der Vorbereitung und des Trainingsprozesses von KI-Systemen, sowie derer Verwendung in dieser Arbeit, erläutert.

2.1.1 Data-Mining-Prozess

Struktur spielt eine wichtige Rolle bei der Verarbeitung großer Datensätze, auf denen das Training eines automatisierten Systems basiert. Ein vordefinierter Prozess erleichtert nicht nur die Vorbereitung, sondern auch den Aufbau des Trainingsprozesses [Sim 2023]. Der Begriff *Data Mining* wird in Kapitel 2.1.3 ausführlicher erläutert, für den Moment ist es ausreichend zu wissen, dass dieser Begriff sowohl den Gesamtprozess der Sammlung, Verarbeitung und Auswertung von Daten als auch den Einzelschritt der Erkennung von Mustern und Beziehungen in Datensätzen umfasst [GilSteHu 2021]. Im Folgenden soll ein bestimmter Data-Mining-Prozess kurz vorgestellt und etwas näher betrachtet werden.

2.1.1.1 KDD

Der *Knowledge Discovery in Databases (KDD)*-Prozess bietet eine Schrittfolge zur Analyse, Verarbeitung und Anwendung von Daten. In ihm sind die wichtigsten Punkte zusammengefasst, die zur effektiven Herausarbeitung von Zusammenhängen und Mustern in Datensätzen und deren Anwendung auf die jeweilige Problemstellung beitragen. Der Prozess umfasst im Wesentlichen folgende Schritte (Abbildung 1) [FaPiSm 1997, Seite 42]:

Zunächst wird bei der *Datenselektion* ein Datensatz erstellt oder ausgewählt, der für die Lösung der Problem- und Fragestellung verwendet wird. Anschließend folgt die *Datenvorverarbeitung*, die zeitlich den größten Aufwand erfordert. Hier werden die ausgewählten Daten analysiert und ggf. bereinigt, um sie für die weiteren Schritte des KDD-Prozesses vorzubereiten. Bei der *Datentransformation* werden die Daten mittels Reduktions- und Transformationsmethoden umgewandelt und für den Trainingsprozess aufbereitet. Beim *Data Mining* wird danach mithilfe eines ausgewählten Modells nach Mustern und Zusammenhängen in den verwendeten Daten gesucht. Das Modell wird dann auf diese Faktoren trainiert, um es später auf neue Daten anwenden zu können. Abschließend werden die Ergebnisse dieser Schritte, sowie die identifizierten Datenmuster, einer *Interpretation und Evaluation* unterzogen.

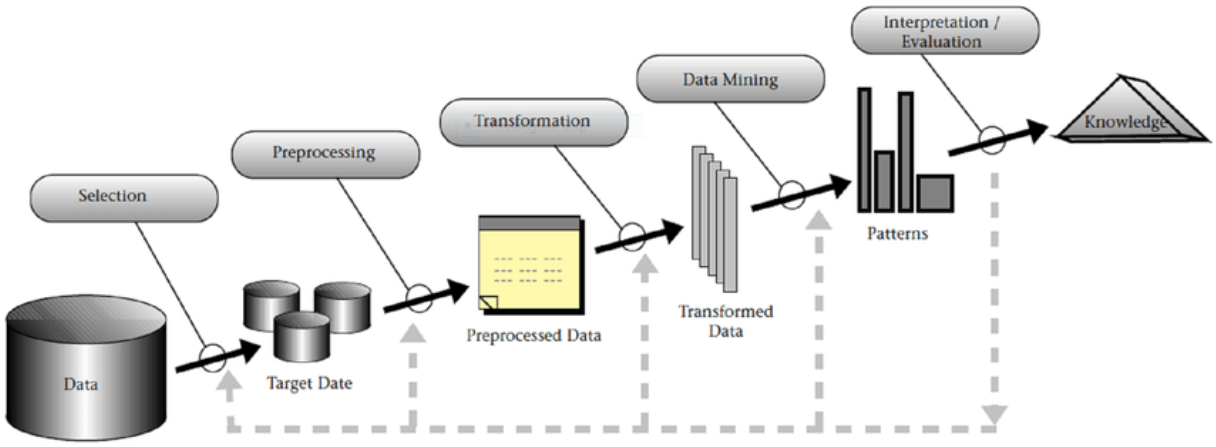


Abbildung 1: Die Schritte des KDD-Prozesses [FaPiSm 1997, Seite 41]

Im Folgenden werden für dieses Thema anwendbare Methoden für die einzelnen Schritte des KDD-Prozesses vorgestellt. Da nicht alle Schritte in diesem Fall spezifische Definitionen erfordern, wird in diesem Kapitel nicht auf alle Vorgänge im Detail eingegangen.
Nach der Auswahl eines Datensatzes stellt sich die Frage, wie diese Daten vor dem Training aufbereitet werden sollen. Für diesen Prozess kann eine Programmiersprache und -umgebung ausgewählt werden. Die Festlegung dieser kann auch im Schritt des Data Minings erfolgen, jedoch soll sie schon hier erwähnt werden, um im späteren Verlauf bestimmte Funktionen der gewählten Sprache referenzieren zu können. Welche Möglichkeiten der Vorverarbeitung zur Verfügung stehen, wird im Folgenden dargelegt.

2.1.2 Datenvorverarbeitung

2.1.2.1 Programmiersprache und -umgebung

Generell gibt es eine Vielzahl von Programmiersprachen und -umgebungen, die für das behandelte Thema geeignet sind. Im Folgenden werden einige kurz vorgestellt:

- **Python** [Py] ist eine der am häufigsten verwendeten Programmiersprachen im Bereich der KI [Zav 2023]. Sie ist sehr vielseitig und findet Anwendung in verschiedenen Bereichen wie Webentwicklung, Datenanalyse und maschinelles Lernen, das in Kapitel 2.1.3.1 näher erläutert wird, sowie in der wissenschaftlichen Forschung und Automatisierung. Python bietet eine Vielzahl von Bibliotheken und Frameworks für spezielle Anwendungen u.a. für das Training automatisierter Systeme [Wod 2023]. Dazu gehören auch Bibliotheken für die Vorverarbeitung von Daten wie beispielsweise *pandas* [Pan] und *NumPy* [Num], sowie zur visuellen Darstellung mathematischer Berechnungen wie *matplotlib* [Mat]. Bei den Frameworks für Data Mining und maschinelles Lernen sind vor allem *TensorFlow* und *PyTorch* hervorzuheben.

TensorFlow [Tf] ist ein führendes Deep-Learning-Framework von Google. Es verwendet statische Berechnungsgraphen und bietet eine umfangreiche Unterstützung für maschinelles Lernen, einschließlich umfassender Tools für Modelltraining, Datenvisualisierung und mathematische Berechnungen in verschiedenen Umgebungen [Boe]. Es ist auch mit anderen Sprachen, wie z.B. R kompatibel und integriert die Bibliothek *Keras* [Ker], welche eine benutzerfreundliche Schnittstelle für die Konstruktion und das Training neuronaler Netze bereitstellt. Dies ermöglicht ein schnelles Prototyping und ist aufgrund seiner Einfachheit sowohl für Anfänger als auch für erfahrene Entwickler geeignet.

PyTorch [PyTo] ist ein weiteres Deep-Learning-Framework mit dynamischen Berechnungsgraphen. Es wurde von Facebook entwickelt und legt den Schwerpunkt auf intuitive Benutzeroberflächen und eine flexible Herangehensweise. PyTorch wird für seine Nutzerfreundlichkeit und die wachsende akademische Community geschätzt [Nvi].

- **R** [R] ist eine Sprache, die speziell für statistische Analysen entwickelt wurde. Sie stellt eine breite Auswahl an Paketen und Funktionen für Datenmanipulation, Visualisierung und statistische Modellierung zur Verfügung [BreCuLiWi 2022]. Mit einer starken Akzeptanz in der wissenschaftlichen Gemeinschaft wird R häufig für Forschung, Datenanalyse und die Erstellung statistischer Modelle verwendet.

- **Visuelle Analyse- und Workflow-Tools:** Es besteht auch die Möglichkeit, Plattformen und Tools zu verwenden, die eine grafische Benutzeroberfläche, auch *Graphical User Interface (GUI)* genannt, für die Analyse von Daten und maschinelles Lernen bieten, ohne dass traditionelle Programmierung erforderlich ist.

  Ein Beispiel hierfür ist *RapidMiner* [Rap]. RapidMiner verwendet eine benutzerfreundliche GUI zum Erstellen von Workflows, die es dem Benutzer erlaubt, Datenanalyse- und maschinelle Lernprozesse grafisch zu modellieren. Es ermöglicht das Verbinden von Modulen in einem visuellen Flussdiagramm, um Daten zu laden, zu transformieren, zu analysieren und Modelle zu erstellen [Arn 2021].

  Alternativ kann *KNIME* [Knime] verwendet werden. Es ähnelt RapidMiner, bietet aber eine breitere Auswahl an Plugins und Integrationen für verschiedene Datenquellen und Analysemethoden. KNIME ist für seine Flexibilität und Erweiterbarkeit bekannt.

Nach der Entscheidung für eine Programmiersprache bzw. -umgebung müssen die Daten mit dieser vorverarbeitet und ggf. bereinigt werden. Wie dies umgesetzt werden kann, wird im Folgenden behandelt.

2.1.2.2 Ausbalancierung des Datensatzes

Zunächst muss darauf geachtet werden, dass das Verhältnis der Datenverteilung auf die beiden Klassen des Datensatzes nicht zu groß ist, z.B. kleiner als 4:1 [Kra 2016]. Dies würde eine gleichermaßen effektive Vorhersagemöglichkeit unabhängig von der Klasse begünstigen. Sollte eine solche Aufteilung unumgänglich sein, könnte entweder mit *Over-* oder *Undersampling* gegengesteuert werden.

Beim *Oversampling* werden dem Datensatz Kopien von Instanzen der Minderheitsklasse hinzugefügt, was auch als Überstichprobe bezeichnet wird. Der Vorteil dieser Methode besteht darin, dass im Gegensatz zum Undersampling kein Informationsverlust entsteht. Allerdings kann sich das Risiko eines *Overfittings*, das in Kapitel 2.1.3.4 näher erläutert wird, erhöhen, da die Daten der Minderheitsklasse wiederholt werden [Pau 2018]. *Undersampling* beschreibt das Entfernen von Instanzen aus der Mehrheitsklasse, auch Unterstichprobe genannt. Einerseits kann dadurch die Laufzeit des Modells durch Reduktion der Trainingsdaten verbessert werden, andererseits können dadurch nützliche Informationen verloren gehen [Pau 2018].

2.1.2.3 Umgehen mit leeren und fehlerhaften Daten

Es kann durchaus vorkommen, dass einige Daten im Datensatz leer sind oder fehlerhafte Werte wie NULL oder ERR enthalten. Eine der größten Auswirkungen von fehlenden Daten ist, dass sie die Ergebnisse von maschinellen Lernverfahren verzerren oder die Genauigkeit des Modells verringern können. Der richtige Umgang mit fehlenden Werten ist daher sehr wichtig [Naw 2022]. Dazu gibt es mehrere Möglichkeiten, von denen zwei hier kurz vorgestellt werden:

Zunächst ist das *Dropping* eine übliche Methode. Dabei werden die fehlenden Zeilen, Spalten etc. einfach aus dem Datensatz entfernt. Dies sollte jedoch mit großer Vorsicht erfolgen. Wenn die Datensatzinformationen wertvoll sind oder der Trainingsdatensatz nur eine geringe Anzahl von Datensätzen enthält, kann das Löschen von Zeilen negative Auswirkungen auf die Analyse haben. Löschmethoden funktionieren gut, wenn die Art der fehlenden Daten völlig zufällig ist [Naw 2022]. Es sollte auch beachtet werden, wie groß die Menge an fehlenden Werten ist. Wenn es sich um einen sehr kleinen Teil des Datensatzes handelt, ist das Dropping unproblematisch.

Darüber hinaus gibt es noch das *Imputing*. Es gibt viele Ansätze für die Imputation, die in der Regel vom jeweiligen Problem abhängen. Wenn die Merkmale numerisch sind, ist die gebräuchlichste Methode, einfache Ansätze wie Mittelwert oder Median zu verwenden [Naw 2022]. Wenn es jedoch Ausreißer gibt, ist der Mittelwert nicht geeignet und es sollte der Medianwert verwendet werden. Der Median ist der Wert, der in der Mitte einer sortierten Datenreihe liegt, wenn die Werte in aufsteigender oder absteigender Reihenfolge angeordnet sind. Im Gegensatz zum Mittelwert ist er weniger anfällig für Ausreißer, da er nicht auf allen Werten basiert, sondern auf der Position in der sortierten Liste.

2.1.2.4 Feature Selection

Feature Selection bezeichnet den Prozess der Auswahl der relevantesten Merkmale oder Variablen aus einem Datensatz, um u.a. die Modellleistung zu verbessern und die Laufzeit zu optimieren.
Dabei muss für jedes einzelne potenzielle Merkmal entschieden werden, welchen Informationsgehalt es, in diesem Fall für den Patientenbefund, trägt. Dies kann zum einen durch die in Kapitel 2.1.2.6 vorgestellte Korrelationsanalyse untersucht werden, zum anderen durch das Plotten der Beziehungen einzelner Variationen innerhalb der untersuchten Merkmale zu den Spannungswerten sowie durch die Ausgabe relevanter Werte, wie z.B. den Durchschnittswert (mean).

2.1.2.5 Bereinigung von Ausreißern

Innerhalb des Datensatzes kann es passieren, dass Werteausreißer in den Daten vorkommen. Das bedeutet, dass einzelne Werte sehr viel kleiner oder größer sind als die Durchschnittswerte der Wertesätze [Bon 2023]. Unbehandelt können diese zu teilweise massiven Einbußen in der Modellperformance führen, da diese Werte beim Training als korrekt und gültig interpretiert werden, obwohl sie in Wirklichkeit fehlerhaft sind [Bon 2023]. Zwei der möglichen Techniken, um solche Werte zu finden, sind *Z-Score* und *Inter Quantile Range (IQR)*.
Zunächst ist zu erwähnen, dass das Verfahren davon abhängt, ob der Datensatz bzw. die analysierten Werte eine *Normalverteilung* aufweisen. Dies bedeutet, dass die Verteilung um den Durchschnittswert des Satzes herum mehr oder weniger gleichförmig ist und keine Verzerrung nach links oder rechts aufweist (Abbildung 2).

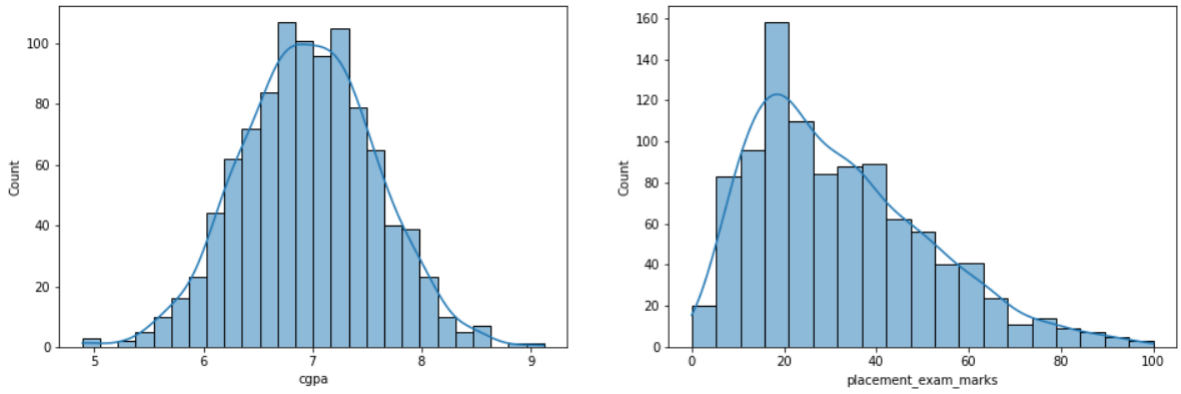


Abbildung 2: Beispiel einer Normalverteilung (links), sowie einer rechtsverzerrten Verteilung (rechts) [Gul 2022]

Im Falle einer Normalverteilung kann Z-Score verwendet werden. Er ist aber in der Weise eingeschränkt, dass bei einer links- oder rechtsverzerrten Verteilung eher IQR verwendet werden sollte [Gul 2022].

Für den Z-Score müssen zunächst der *Durchschnittswert* des Datensatzes, sowie die *Standardabweichung* berechnet werden. Der Durchschnittswert wird durch die Division der summierten Spannungswerte durch die Summe der Messungen ermittelt. Die Standardabweichung misst wie weit ein typischer Messwert im Datensatz von dem Durchschnittswert entfernt ist [W3]. Sie ist die Wurzel aus der Summe der quadrierten von den Datensatzwerten abgezogenen Mittelwerte, geteilt durch die Summe aller Werte.

Der Z-Score bestimmt den Multiplikationsfaktor der Standardabweichung und somit den akzeptablen Wertebereich. Wäre der Z-Score z.B. 1, so würden alle Werte, die im Bereich *Durchschnittswert +/- 1*Standardabweichung* liegen, als korrekt und valide bewertet werden. Alle anderen wären in diesem Beispiel Ausreißerwerte.

Nachdem diese identifiziert wurden, müssten sie entweder getrimmt oder gekappt werden [Gul 2022]. Beim *Trimming* werden alle Ausreißerwerte aus dem Datensatz entfernt. Beim *Capping* wird diesen Werten der maximale oder minimale Grenzwert zugewiesen, je nachdem ob sie stark über oder unter dem Durchschnittswert liegen [Gul 2022].

Bei der Verwendung des IQR werden ebenfalls Trimming und Capping verwendet, jedoch ist der Weg zum Auffinden der Extremwerte etwas anders. Zur Berechnung des IQR muss das *25.*, sowie das *75. Perzentil* der Wertemenge berechnet werden [Gul]. Perzentil beschreibt dabei den Wert, unter dem der festgelegte Satzanteil liegt z.B. beschreibt das 75. Perzentil den Wert unter dem 75% aller Werte liegen. Um die Grenzwerte für die Suche nach Ausreißern zu bestimmen, werden folgende Formeln angewendet:

IQR = 75. Perzentil - 25. Perzentil

GrenzwertMin = 25. Perzentil - 1.5 * IQR

GrenzwertMax = 75. Perzentil + 1.5 * IQR

Nun müssen nur noch die Grenzwerte auf alle Datensatzwerte angewendet werden, um die Ausreißer zu bestimmen. Diese können dann ebenfalls getrimmt oder gekappt werden.

2.1.2.6 Korrelationsanalyse

Bei der *Pearson Correlation* werden einzelne Features auf ihre Korrelation untereinander untersucht. Korrelation bedeutet, dass die Werte im Laufe der Zeit die gleichen Richtungsänderungen aufweisen. Bei der Pearson Correlation wird diese in einem Wertebereich von -1 bis 1 gemessen [Yem 2019]. Korrelieren zwei Werte stark miteinander, so ist die Korrelation positiv. Weisen die Veränderungen der untersuchten Werte im Zeitverlauf entgegengesetzte Richtungen auf, so ist die Korrelation negativ.

Bei der Untersuchung der Features auf Korrelation wird eine Korrelationsmatrix erstellt, die die Korrelationskoeffizienten zwischen den jeweiligen Features anzeigt. Wenn zwei Merkmale stark miteinander korrelieren, so kann eines davon als redundant angesehen und verworfen werden. Auf diese Weise wird der Datensatz bereinigt, ohne dass dabei wichtige Informationen verloren gehen [Yem 2019].
Hierbei können Features mit einem Korrelationskoeffizienten größer als 0,9 als sehr stark korrelierend, zwischen 0,7 und 0,9 als stark korrelierend, zwischen 0,5 und 0,7 als mäßig korrelierend und zwischen 0,3 und 0,5 als schwach korrelierend bezeichnet werden [Cal 2005].

Nachdem die Daten vorverarbeitet und ggf. transformiert wurden, kann mit dem eigentlichen Training des Modells begonnen werden. Welche Faktoren dabei zu beachten und festzulegen sind, wird im Folgenden behandelt.

2.1.3 Data Mining

Aus dem Prozess des Data Minings ergibt sich die Möglichkeit durch die Untersuchung von großen Datensätzen unbekannte Muster, Beziehungen oder sogar Anomalien festzustellen [Mar]. Hierbei können verschiedene Verfahren eingesetzt werden. Zu den bekanntesten gehören u.a. folgende:
Bei der *Klassifizierung* werden Objekte aufgrund bestimmter Gemeinsamkeiten in vordefinierte Klassen eingeteilt. Dies führt zu einer besseren Kategorisierung der Daten [Twin 2023]. Zu diesem Verfahren gehören u.a. Decision Trees, Support Vector Machines und Naive Bayes. Eine *binäre Klassifizierung* beschreibt dabei die Unterscheidung zwischen zwei Klassen.
Das *Clustering* ähnelt dem Klassifizieren, gruppiert jedoch Objekte anhand von Merkmalen, die sie von anderen Objekten unterscheiden. Eine beliebte Methode ist hierbei der k-Means-Algorithmus [Har 2018].
Regressionen sind für die Verarbeitung und Vorhersage von Fällen mit numerischen Werten spezifiziert. Dabei werden stetige Werte analysiert und auf Korrelation mit verschiedenen Variablen überprüft. Lineare und logistische Regression sind hierbei die bekanntesten Verfahren [Har 2018].
Beim *Forecasting* werden Prognosen anhand von Daten aus der Vergangenheit, sowie der Gegenwart, getroffen. Dabei werden diese Daten analysiert und auf verborgene Muster überprüft [Syd 2021].
Darüber hinaus gibt es weitere wichtige Bereiche, wie z.B. neuronale Netze, die ebenfalls zu den Klassifikationsverfahren gehören und auf die in Kapitel 2.1.3.2 näher eingegangen wird.

Beim Data Mining spielt die Automatisierung eine entscheidende Rolle. Durch maschinelles Lernen können entwickelte Datenverarbeitungstechniken nahtlos auf neue Daten angewendet werden.

Im folgenden Abschnitt wird näher darauf eingegangen, wie maschinelles Lernen die Effizienz und Anwendbarkeit von Data Mining weiter erhöht.

2.1.3.1 Maschinelles Lernen

Maschinelles Lernen (ML) befasst sich mit der Anwendung von Daten und Algorithmen, um menschliches Lernen zu simulieren und die Genauigkeit von Analysen schrittweise zu verbessern. Mithilfe statistischer Methoden werden Algorithmen trainiert, um Klassifikationsentscheidungen oder Vorhersagen zu treffen und wertvolle Erkenntnisse in Data-Mining-Projekten zu gewinnen [IbmMl]. Um die Dimension dieser Lernprozesse zu verstehen, ist es wichtig, zwischen einigen verschiedenen Arten von ML zu unterscheiden:

Beim *überwachten Lernen (supervised learning)* werden gelabelte Daten zum Training übergeben. Das bedeutet, dass das Ergebnis einer Auswertung dieser einzelnen Daten für das Training zur Verfügung steht. Anhand dieser lernt das System die Zusammenhänge zwischen Ein- und Ausgabe der Trainingsdaten und kann diese danach auf neue Daten anwenden. Meist wird diese Lernart in Bereichen eingesetzt, in denen ein bestimmtes Klassifikationsergebnis als Output ausgegeben werden soll [IbmSl].

Bei dem *unüberwachten Lernen (unsupervised learning)* erhält das System gar keine Informationen zu den jeweiligen Auswertungsergebnissen der Daten. Die Aufgabe des kognitiven Systems wäre es, bestimmte neue Muster und Ähnlichkeiten zu identifizieren [IbmUsl].

Das *semi-überwachte Lernen (semi-supervised learning)* beinhaltet sowohl das überwachte, als auch das unüberwachte Lernen in sich. Dabei besteht der Trainingsdatensatz aus einer kleinen Stichprobe gelabelter und einer großen Menge ungelabelter Daten [Alt 2022].

Das *bestärkende Lernen (reinforcement learning)* wird beim Training angewendet, indem das System durch wiederholte Trial-and-Error-Versuche und positives bzw. negatives Feedback neue Erkenntnisse zur Bewältigung der Aufgabe gewinnt [Geek].

Ein weiteres Verfahren, das auf dem Modell menschlicher Nervenzellen beruht, basiert auf den sogenannten neuronalen Netze. Sie spielen eine entscheidende Rolle beim Deep Learning, das in Kapitel 2.1.3.3 erläutert wird. Im Folgenden soll auf die Funktionsweise von neuronalen Netzen eingegangen werden.

2.1.3.2 Neuronale Netze

Der Name und die Struktur neuronaler Netze sind vom menschlichen Gehirn inspiriert und ahmen die Signalverarbeitung zwischen biologischen Gehirnneuronen nach [IbmKnn].

Künstliche neuronale Netze (KNNs) können dabei verschiedene Architekturen aufweisen.

Zu den am häufigsten verwendeten und wichtigsten gehören u.a. *Feedforward (FNN), Recurrent (RNN), Long Short-Term Memory (LSTM), Generative Adversarial (GAN), Conditional GAN (cGAN), Self-organizing map (SOM)* und *Convolutional (CNN)*.
Auf die Architektur und Funktionsweise von FNNs, RNNs und CNNs wird im Folgenden etwas näher eingegangen.

- **Feedforward neural networks:** FNNs bestehen aus mehreren miteinander verbundenen Schichten (Abbildung 3).

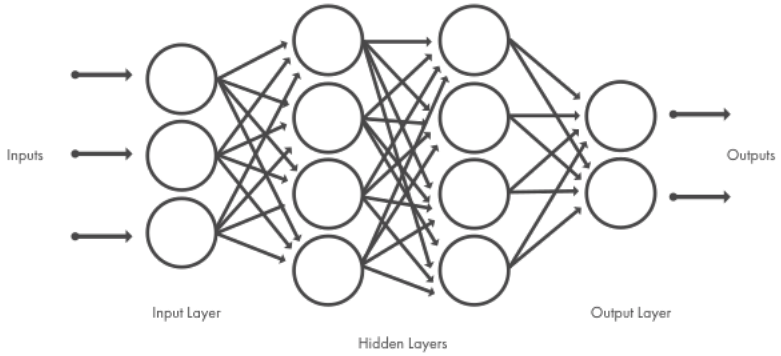


Abbildung 3: Eine vereinfachte Darstellung eines FNNs [Math]

In die Eingabeschicht (Input Layer) werden die Daten eingegeben, die das neuronale Netz verarbeiten soll [Dee]. Zusätzlich wird ein weiterer Wert, der Bias, eingespeist. Er kann Einfluss auf die Aktivierungsfunktion nehmen, indem er sie staucht, streckt, verschiebt usw. (Abbildung 4).

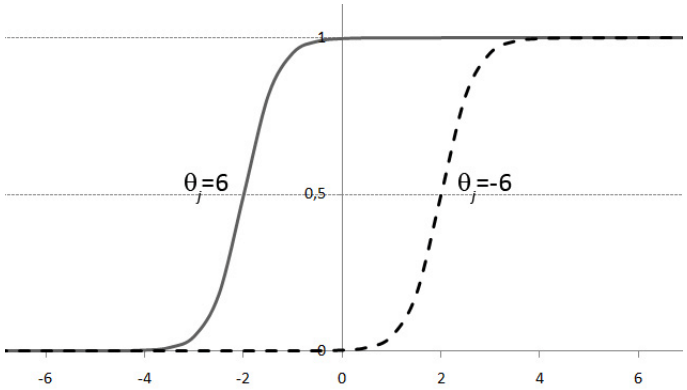


Abbildung 4: Ein Beispiel für den Einfluss eines Bias [Lac 2016]

Um die unterschiedliche Wichtigkeit der Eingaben zu berücksichtigen, werden alle Inputs mit einem individuellen Gewicht multipliziert. Je größer der Einfluss auf das Ergebnis, desto höher das Gewicht. Während des Trainings passt das System seine Gewichtungen und Bias-Werte so an, dass für alle Inputs die gewünschten Outputs berechnet werden [Dee].

Die verborgenen Schichten (Hidden Layer) sind für die Analyse der Eingabedaten zuständig. Der Name leitet sich von der Tatsache ab, dass der/die BenutzerIn keinen Zugriff auf die von diesen Schichten errechneten internen Werte besitzt [Dee].

Die Ausgabeschicht enthält die aus dem berechneten Ergebnis herausgenommene Antwort auf die jeweilige Fragestellung. Die von den vorangegangen Schichten berechneten Werte werden zuvor summiert an eine Aktivierungsfunktion weitergegeben, die den an die nächste Instanz weiterzugebenden Wert berechnet (Abbildung 5) [Dee].

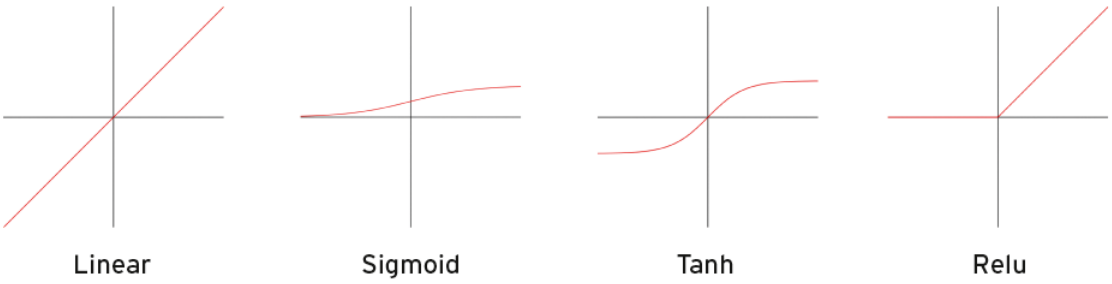


Abbildung 5: Einige Beispiele von Aktivierungsfunktionen [Hin 2016]

Darüber hinaus können weitere Layer hinzugefügt werden, um das Training zu stabilisieren. So werden beispielsweise *Dropout Layer* verwendet, um Overfitting zu reduzieren, indem zufällig ausgewählte Neuronen während des Trainings deaktiviert werden. Dies vermindert Redundanzen und stärkt die Robustheit des KNNs [Data 2023]. Weitere Schichten, wie z.B. die *Batch Normalization Layer*, normalisieren die Aktivierungen der vorherigen Schicht, was zu stabileren Lernprozessen und schnellerer Konvergenz führen kann [Batch].

- **Recurrent neural networks:** RNNs sind ähnlich aufgebaut wie FNNs mit dem Unterschied, dass sie die Ausgabe einer Schicht speichern und wieder in die Eingabe derselben Schicht einspeisen, was die Namensgebung dieses NNs erklärt (Abbildung 6). Bei der Entscheidungsfindung berücksichtigt das RNN, im Gegensatz zum FNN, nicht nur die aktuelle Eingabe, sondern auch die zuvor empfangenen. Einfache RNNs weisen ein eher kurzzeitiges Gedächtnis (Short-Term-Memory) auf, dessen Kapazität aber durch Erweiterungen wie LSTM erhöht werden kann [Don 2023].

- **Convolutional neural networks:** CNNs sind auf die Verarbeitung von Daten mit gitterartigen Topologien, wie z.B. Bildern, spezialisiert. Dabei lernt das CNN anhand bestimmter Muster, die es dann in den Testdaten erkennt. Mit jeder Schicht steigt die erlernbare Komplexität der Bilder. Sie sind so angeordnet, dass zunächst einfachere Muster (Linien, Kurven etc.) und später komplexere Muster (Gesichter, Objekte etc.) erkannt werden [Mis 2020].

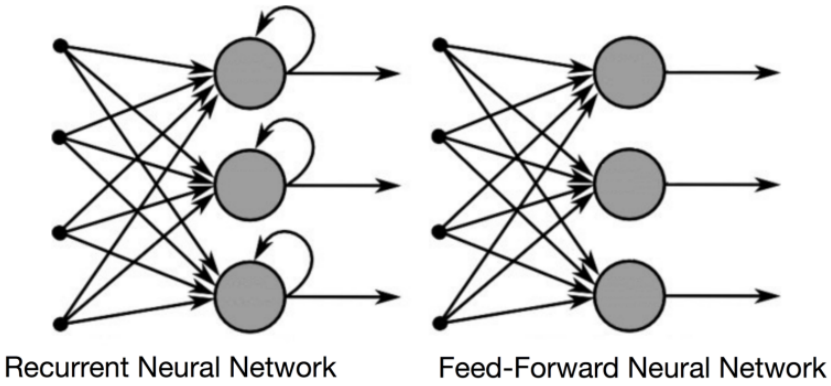


Abbildung 6: Unterschied zwischen einem RNN und einem FNN [Don 2023]

Jedes künstliche Neuron ist mit einem anderen verbunden, was bedeutet, dass die Ausgabe eines Neurons an die Eingabeschicht eines anderen Neurons weitergeleitet wird. Wenn jedes Neuron einer Layer mit jedem Neuron einer anderen Layer verbunden ist, wird dies als *Fully Connected Layer* bezeichnet. Beinhaltet ein neuronales Netzwerk nur Fully Connected Layer, so ist dieses ein *Fully connected neural network* [Unzu 2022].
Komplexe Netze bestehen aus mehreren miteinander verbundenen Neuronen, die teilweise hunderte von verborgenen Schichten zur Berechnung einer Ausgabe besitzen.
In Bezug auf die Anzahl der Layer ist es für Deep-Learning-Verfahren sehr üblich, mehrere Hidden Layer zu verwenden. Dies wird nun im Folgenden näher erläutert.

2.1.3.3 Deep Learning

Deep Learning ist ein besonderer Teilbereich von ML, der sich in einigen Punkten von klassischen ML-Algorithmen unterscheidet. Um diese Unterschiede auf einen Nenner zu bringen, kann man sagen, dass klassische ML-Algorithmen auf einfachen Data-Mining-Strukturen wie linearer Regression oder Entscheidungsbäumen basieren, während bei Deep Learning ausschließlich neuronale Netze verwendet werden. Der Einsatz von neuronalen Netzen erfordert nur wenig menschliche Interaktion, da sie bei der Analyse von Daten automatisch Zusammenhänge erkennen und während des Trainingsprozesses aus ihren eigenen Fehlern lernen. Sie sind zwar komplexer in der Einrichtung, erfordern aber danach nur noch minimale Eingriffe. Im Gegensatz dazu erfordern ältere ML-Algorithmen mehr ständiges menschliches Eingreifen, um Ergebnisse zu erzielen [Mid 2021]. Durch diese Automatisierung werden jedoch im Umkehrschluss viel größere Datensätze, sowie leistungsstarke Grafikkarten, zum Trainieren des Modells benötigt [Cou 2023].

In Bezug auf die Layerarchitektur gilt, dass Deep Learning nur dann besteht, wenn mehrere Hidden Layer verwendet werden. Im Wesentlichen wird bei zwei oder mehr verborgenen Schichten von Deep Learning gesprochen [Pra 2022].
Insgesamt ermöglicht Deep Learning, Probleme anzugehen, die in der Vergangenheit nur mit begrenztem Erfolg gelöst werden konnten [Wol 2023].
Nach der Entscheidung für ein bestimmtes Vorgehen muss der fertige Datensatz für das Training aufgeteilt werden. Dies wird im Folgenden beschrieben.

2.1.3.4 Datensatzaufteilung

Für das Training kann der gesamte Datensatz in zwei bzw. drei Teildatensätze aufgeteilt werden. Der größte Teil wird für das Training selbst verwendet, er wird als Trainingsdatensatz bezeichnet [Shah 2017]. Ein wesentlich kleinerer Teil wird für das Testen des trainierten Modells beiseite gelegt. Dieses Testset repräsentiert neue Daten, die zur prädiktiven Analyse des Modells dienen [Shah 2017]. Häufig wird ein weiterer Teildatensatz verwendet, der Validierungsdatensatz. Dieser kommt während des Trainings zum Einsatz und wird für eine unverzerrte Beurteilung des Prozesses anhand des Trainingssatzes während der Parameteranpassung verwendet [Shah 2017]. Durch das Anzeigen der Vorhersagen während des Trainings kann *Over- bzw. Underfitting* leicht erkannt werden.
Overfitting bezeichnet hierbei eine zu starke Anpassung an die Trainingsdaten, sodass das Modell eine hohe Trefferwahrscheinlichkeit bei Daten aus dem Trainingsset, jedoch eine schlechte Performance für neue Daten aufweist [Gee].
Bei Underfitting ist zusätzlich die Leistung beim Trainingssatz nicht gut, da das Modell z.B. zu einfach aufgebaut ist, um ggf. komplexe Muster im Datensatz zu erkennen [Gee].
Weiterhin kann auch ein *random-state* gesetzt werden. Er steuert die Zufälligkeit bei der Aufteilung der Daten [Mod 2022]. Dieser Parameter ermöglicht die Reproduzierbarkeit der Verteilung, wenn er auf einen bestimmten Wert gesetzt wird. Das bedeutet, dass unabhängig davon, wie oft der Code ausgeführt wird, die Verteilung der Daten immer gleich bleibt, solange random-state den gleichen Wert hat [Mod 2022].
Bezüglich der Aufteilung des Datensatzes hat sich eine prozentuale Rate von 60-80% des Gesamtdatensatzes für das Trainingsset und jeweils 10-20% für das Test- und Validierungsset als praxisüblich etabliert [Bah 2021].
In diesem Zusammenhang ist es auch möglich *cross-validation* zur Evaluierung des Modells zu verwenden. Mit ihr können Modelle anhand einer begrenzten Datenstichprobe besser beurteilt werden [Bro 2023].

Das Verfahren besitzt einen Parameter namens k, der sich auf die Anzahl der Gruppen bezieht, in die eine gegebene Datenstichprobe aufgeteilt werden soll [Bro 2023]. Daher leitet sich auch der technische Name *k-fold cross-validation* ab. Sie ist eine beliebte Methode, weil sie im Allgemeinen zu einer weniger verzerrten oder weniger optimistischen Schätzung der Modellfähigkeit führt als andere Methoden, wie z.B. eine einfache Aufteilung in Trainings- und Testsatz [Bro 2023]. Der Datensatz wird zufällig gemischt und in k-Gruppen aufgeteilt. Eine Gruppe wird als Testdatensatz verwendet, die übrigen bilden den Trainingsdatensatz.
Das Modell wird mit dem neuen Trainingssatz trainiert und anhand des neuen Testsatzes evaluiert. Dieser Prozess wird k-Mal wiederholt, sodass jede Gruppe einmal als Testsatz und k-1-mal im Trainingsdatensatz vorkommt [Bro 2023].
Eine wichtige Erweiterung beinhaltet die *stratified k-fold cross-validation*. Sie funktioniert genauso wie die ursprüngliche Version mit dem Unterschied, dass die Verteilung der Daten auf die beiden Klassen mit einbezogen wird. Bei der normalen k-fold kann es vorkommen, dass ein großer Teil der größeren Klasse und nur ein kleiner Teil der kleiner repräsentierten Klasse in den Trainingsdatensatz aufgenommen wird, da hier die Stichproben zufällig verteilt werden [Geeks]. Wenn jedoch die prozentuale Verteilung der Daten berücksichtigt wird, kann dieses Szenario vermieden werden, da nicht 80% des Gesamtdatensatzes verwendet werden, sondern 80% der ersten und 80% der zweiten Klasse [Geeks].

Beim Trainieren eines Modells müssen verschiedene Hyperparameter beachtet werden. Hyperparameter sind Einstellungen oder Konfigurationen, die vor dem eigentlichen Training eines maschinellen Lernalgorithmus festgelegt werden. Im Gegensatz zu den Parametern, die während des Trainings selbst gelernt werden, sind Hyperparameter nicht Teil des Lernprozesses des Modells. Sie werden festgelegt, um die Struktur oder das Verhalten des Modells zu steuern und zu optimieren. Jeder einzelne hat eine bestimmte Bedeutung und einen gewissen Effekt auf die Leistung des Modells nach dem Training. Im Folgenden wird beschrieben, wie das FNN konfiguriert wird und welche Parameter wie eingestellt werden.

2.1.3.5 Layerarchitektur

Der Datensatz muss zunächst in Eingabe- und Ausgabedaten unterteilt werden. An den Eingabedaten lernt das Modell und anhand der Ausgabedaten stellt es die Verbindung zu der späteren Prognose her und erkennt Zusammenhänge zwischen ihnen.
Die einzelnen Layer können nach den im Kapitel 2.1.3.2 beschriebenen Architekturen aufgebaut werden, hier ist lediglich die Unterscheidung zwischen Inputlayer, Hiddenlayer und Outputlayer als einzelne Variablen wichtig.

Bei den Hiddenlayern kann zusätzlich die Anzahl der Neuronen in dieser Schicht, sowie die Aktivierungsfunktion, die auch für die Outputschicht eingestellt werden muss, festgelegt werden.

2.1.3.6 L1-/L2-Regularisierung

Auch innerhalb der Layer können Maßnahmen bei auftretenden Problemen, wie Overfitting, getroffen werden. So dient *Regularisierung* dazu, bei steigender Modellkomplexität diese wieder zu reduzieren, um das Risiko von Overfitting zu verringern. Hierbei könnte z.B. ein zusätzlicher Term zu der Verlustfunktion des Modells, die in Kapitel 2.1.3.8 näher beleuchtet wird, hinzugefügt werden, der die Größe der Gewichte des Modells einschränkt [Nag 2017].
Wichtige Arten davon sind die *L1-* und *L2-Regularisierungen* [Tya 2021]. Der Hauptunterschied liegt in dem Term, der zur Verlustfunktion addiert wird. Bei L1 ist es die Summe der absoluten Werte der jeweiligen Gewichtskoeffizienten. Durch die L1-Regularisierung werden viele der Gewichte zu null gemacht, wodurch das Modell spärlicher wird. Das bedeutet, dass das Modell nur eine begrenzte Anzahl von Merkmalen verwendet, was zu einer Art Feature Selection führen kann [Tya 2021]. Bei L2 wird die Summe der Quadrate der Gewichtskoeffizienten verwendet. Im Gegensatz zur L1-Regularisierung werden die Gewichte hier nicht auf null reduziert, sondern nur verkleinert [Tya 2021].
Häufig wird auch eine Kombination aus L1- und L2-Regularisierung verwendet. Dieses Prozedere ist auch als *Elastic-Net* bekannt [John 2019].

2.1.3.7 Optimizer

Weiterhin muss ein Optimizer für das Training gesetzt werden. Der Optimizer ist ein Algorithmus, mit dem die Gewichte des neuronalen Netzes während des Trainings angepasst werden [Gup 2023]. Hierbei kann man zwischen adaptiven und stochastischen Optimizern unterscheiden.
Im Gegensatz zu Algorithmen mit fester Lernrate passen adaptive Optimierer die Lernrate in Abhängigkeit von verschiedenen Faktoren dynamisch an [Gup 2023]. Sie verwenden Informationen über die Gradienten vergangener Iterationen, um die Lernrate zu modifizieren. Dadurch kann diese für jedes Gewicht anhand der Aktualisierungshäufigkeit angepasst werden [Gup 2023].
Einer von den adaptiven Optimizern ist *Adagrad*. Er passt die Lernrate für jedes Gewicht basierend auf der Häufigkeit des Updates an. Gewichte, die häufiger aktualisiert werden, erhalten eine kleinere Lernrate, während Gewichte, die seltener aktualisiert werden, eine höhere Lernrate erhalten [Oppe].
RMSprop ist ein weiterer Optimizer und verwendet adaptive Lernraten, um die Lernrate für jedes Gewicht individuell anzupassen, basierend auf dem durchschnittlichen quadratischen Gradienten der vergangenen Iterationen [Oppe].

Schließlich sei noch *Adam* erwähnt. Er berechnet individuelle adaptive Lernraten für jedes Gewicht, indem er Schätzungen des ersten und zweiten Moments der Gradienten verwendet.
Er kombiniert das Konzept des Momentums (wie bei RMSprop) mit der Idee adaptiver Lernraten (wie bei Adagrad) [Oppe].

2.1.3.8 Loss functions

Darüber hinaus gibt es noch die Verlustfunktion (Loss function). Sie misst den Unterschied zwischen den tatsächlichen und den vorhergesagten Werten und soll während des Trainingsprozesses minimiert werden. Verschiedene Verlustfunktionen eignen sich für unterschiedliche Problemstellungen. Einige Beispiele, die bei einer binären Klassifikation verwendet werden können, wären folgende:
Die *Binary Cross-Entropy Loss* berechnet einen Score, der die durchschnittliche Differenz zwischen der tatsächlichen und der vorhergesagten Wahrscheinlichkeitsverteilung zusammenfasst. Ein perfekter Wert für die Kreuzentropie ist 0. Mathematisch gesehen ist sie die bevorzugte Verlustfunktion im Rahmen der Inferenzmethode der maximalen Wahrscheinlichkeit (maximum likelihood). Sie ist die Verlustfunktion, die zuerst ausgewertet werden sollte und nur dann geändert wird, wenn es einen guten Grund dafür gibt [Bro 2020].
Eine Alternative zur Kreuzentropie ist der *Hinge Loss*. Er ist für binäre Klassifikationsprobleme gedacht, bei der die Zielwerte in der Menge {-1, 1} liegen. Der Hinge Loss weist mehr Fehler zu, wenn es einen Vorzeichenunterschied zwischen den tatsächlichen und den vorhergesagten Klassenwerten gibt [Bro 2020].
Der Hinge Loss hat viele Erweiterungen. Eine beliebte Erweiterung ist der *Squared Hinge Loss*, bei der einfach das Quadrat des Hinge Losses berechnet wird. Dadurch wird die Oberfläche der Fehlerfunktion geglättet und die numerische Arbeit mit ihr erleichtert [Bro 2020].

2.1.3.9 Batch-Size

Zu weiteren wichtigen Trainingsparametern gehört die Batch-Size. Sie bestimmt die Anzahl der Trainingsdatensätze, die in einer Iteration (Epoche) verarbeitet werden. Eine größere Batch-Size bedeutet, dass mehr Daten gleichzeitig verarbeitet werden, während eine kleinere Batch-Größe weniger Daten in jeder Epoche verwendet.

Nach Abschluss des Trainingsprozesses müssen die Ergebnisse analysiert und bewertet werden. Hierfür stehen einige Analysewerkzeuge zur Verfügung, die im Folgenden vorgestellt werden.

2.1.4 Interpretation und Evaluation

Eine Analyse des trainierten Modells ist unerlässlich, um Faktoren, wie Over- und Underfitting auszuschließen und der Leistungsfähigkeit des Modells auf den Grund zu gehen. Zu diesem Zweck können folgende Tools und Parameter verwendet werden:
Eine *Confusion Matrix* ist eine Tabelle, die die Leistung eines Klassifikationsmodells darstellt. Sie zeigt die Anzahl der korrekten und falschen Vorhersagen, indem sie die tatsächlichen und vorhergesagten Werte für jede Klasse des Modells vergleicht. Bei einer binären Klassifikation gibt es vier Felder: True Positive (TP), False Positive (FP), True Negative (TN) und False Negative (FN) [Nar 2018].
Die *Accuracy* ist ein Maß dafür, wie viele Vorhersagen eines Modells richtig waren, gemessen als Anteil der richtigen Vorhersagen an der Gesamtzahl der Vorhersagen (Formel: (TP + TN)/(TP + TN + FP + FN)) [Nar 2018].
Die *Precision* gibt an, wie viele der als positiv vorhergesagten Fälle tatsächlich positiv sind (Formel: TP/(TP + FP)) [Nar 2018].
Der *Recall* gibt an, wie viele der tatsächlich positiven Fälle vom Modell als positiv identifiziert wurden (Formel: TP/(TP + FN)) [Nar 2018].
Ein *Classification Report* ist eine Zusammenfassung der Leistung eines Klassifikationsmodells, die die Precision, den Recall und andere wichtige Metriken für jede Klasse im Modell anzeigt.
Nach den Grundlagen des maschinellen Lernens sowie des Trainingsprozesses eines neuronalen Netzwerkes soll nun kurz auf die genetische Vererbung einer Alkoholismus-Veranlagung sowie auf die Verwendung der EEG zur Erkennung dieser geblickt werden, um alle wichtigen Themenbereiche dieser Arbeit abzudecken.

2.2 Alkoholismus

Das Risiko einer Alkoholabhängigkeit wird stark von genetischen Faktoren und der familiären Vorgeschichte von Alkoholismus beeinflusst. Nach [Cot 1979] wurde festgestellt, dass bei alkoholabhängigen Personen die Wahrscheinlichkeit, dass ein Elternteil ebenfalls alkoholabhängig war/ist, 4-6 Mal höher ist, als bei Personen, die keinen Alkohol konsumieren [PoRaKaJoPaBe 2004, Seite 994].
Im Jahr 2012 wurden wichtige Forschungsergebnisse zur Genetik von Alkoholmissbrauch und -abhängigkeit veröffentlicht. Medizinische Forscher fanden heraus, dass elf Genpaare in gewisser Weise mit einem erhöhten Risiko für exzessiven Alkoholkonsum und der Entwicklung zwanghafter Verhaltensweisen im Zusammenhang mit Alkohol in Verbindung stehen [Alc] [MoGoMaAn 2012].
All diese genetischen Faktoren lassen sich sehr gut mit EEG-Messungen untersuchen lassen. Dies wird im nächsten Kapitel erläutert.

2.3 EEG

2.3.1 Definition

Bei der EEG handelt es sich um eine Messmethode, mit der u.a. eine mögliche Veranlagung zum Alkoholismus festgestellt werden kann. Bei der Durchführung dieses Tests wird die elektrische Aktivität des Gehirns mit Hilfe kleiner, auf der Kopfhaut angebrachten, Elektroden gemessen. Die Kommunikation zwischen den Gehirnzellen erfolgt über elektrische Impulse. Diese Aktivität kann in Form von Wellenlinien auf einer EEG-Aufzeichnung abgelesen werden. Die daraus resultierende grafische Darstellung wird als Elektroenzephalogramm bezeichnet. Es ist eine der wichtigsten Untersuchungen in der Diagnostik. Mit ihrer Hilfe ist es möglich, aus eventuell festgestellten Aktivitätsveränderungen im Gehirn auf bestimmte Hirnleistungsstörungen zu schließen [May 2022].
Das Ruhe-Elektroenzephalogramm erfasst die kontinuierliche spontane elektrische Aktivität des Gehirns einer Person. Diese wird üblicherweise in verschiedene Frequenzbereiche unterteilt, nämlich Delta (1,0-3,0 Hz), Theta (3,5-7,5 Hz), Alpha (8-11,5 Hz), Beta (12-28 Hz) und Gamma (28,5-50 Hz). Jeder Frequenzbereich spiegelt eine andere Gehirnaktivität wider. Bei gesunden Erwachsenen überwiegen im wachen Ruhe-Elektroenzephalogramm mittlere (8-13 Hz) und schnelle (14-30 Hz) Frequenzen [PoRaKaJoPaBe 2004, Seite 995].
Der wichtigste Faktor ist, dass es in allen Frequenzbereichen stabil ist und eine hohe Vererbbarkeit aufweist: Delta 76%, Theta 89%, Alpha 89%, Beta 86% [PoRaKaJoPaBe 2004, Seite 995].
Diese Vererbbarkeit ist der Schlüssel zur Untersuchung von Zusammenhängen zwischen EEG-Messungen von Alkoholikern und ihren Nachkommen, wie im nächsten Abschnitt erläutert wird.

2.3.2 Anwendung von EEG zur Untersuchung von Alkoholsucht-Neigungen

EEG-Messungen bieten eine hohe Empfindlichkeit, um die unmittelbaren und langfristigen Auswirkungen von Alkohol auf das Gehirn zu erfassen. Studien deuten darauf hin, dass bestimmte atypische elektrophysiologische Merkmale bei Alkoholikern genetisch bedingt sind und eine Vorstufe zur Entwicklung von Alkoholismus darstellen [PoRaKaJoPaBe 2004, Seite 994]. Da solche Messungen weniger komplex, vererbbarer und näher an der Genaktivität sind als klinische Diagnosen, stellen sie ein wertvolles Instrument zur Identifizierung von Genen dar, die an der Prädisposition zur Entwicklung von Alkoholismus und verwandten Störungen beteiligt sind [PoRaKaJoPaBe 2004, Seite 994].
EEG-Messungen von Menschen im wachen, aktiven Zustand, bei denen Betawellen des Gehirns gemessen werden können, bilden den Grundbaustein der Studie, die dieser Arbeit zugrunde liegt (s. Kapitel 3.2.2).

Der Beta-Rhythmus zeigt eine weiträumige Verteilung über die Kopfhaut und hat sich als bedeutender neurophysiologischer Indikator für die Untersuchung der Prädisposition zum Alkoholismus etabliert. Dabei wurde eine Assoziation der Betawellen mit einigen der oben erwähnten Gene festgestellt. Dieses Ergebnis wurde in mehreren unabhängigen Studien bestätigt, wie z.B. [CoGeHeNeKr 2004] [PoRaKaJoPaBe 2004, Seite 996].

Es ist ebenfalls gut dokumentiert, dass Alkoholiker eine erhöhte Beta-Aktivität aufweisen, insbesondere im Ruhezustand. Verschiedene Studien berichten auch über eine erhöhte Beta-Aktivität im Elektroenzephalogramm bei Familienangehörigen von Alkoholikern. [GaMeVoPoScIt 1982] stellten eine erhöhte Beta-Aktivität bei männlichen Probanden mit Alkoholismus in der Familie fest. [PoEaGa 1995] berichteten über erhöhte Beta-Aktivität bei Personen mit positiver Familienanamnese im Vergleich zu Personen mit negativer Familienanamnese. In einer späteren Studie von [FiJu 1999] wurde eine erhöhte relative Beta-Aktivität in den frontalen Arealen festgestellt [PoRaKaJoPaBe 2004, Seite 997].

Zusammenfassend lässt sich sagen, dass Studien, die über die Zusammensetzung und Charakteristika des Ruhe-Elektroenzephalogramm berichten, auf eine Erhöhung des Beta-Bandes als primäres charakteristisches Merkmal bei Alkoholikern und Hochrisikopersonen hinweisen.

3 Konzept

Nachdem die Problemstellung dieser Bachelorthesis identifiziert und ein Verständnis für den Anwendungsbereich und das relevante Vorwissen entwickelt wurde, werden nun in diesem Abschnitt die Ideen und Möglichkeiten der Modellumsetzung betrachtet und die wichtigsten Entscheidungspunkte herausgearbeitet. Dabei soll nach dem in Kapitel 2.1.1.1 beschriebenen KDD-Prozess vorgegangen werden.

3.1 Zielsetzung

Das Ziel dieser Arbeit besteht darin, ein zuverlässiges Modell zu entwickeln, mit dem das Vorhandensein einer genetischen Prädisposition für Alkoholismus anhand von EEG-Messungen beurteilt werden kann. Dabei müssen die in Kapitel 2.3.2 beschriebenen Zusammenhänge zwischen EEG-Messungen und solchen Veranlagungen berücksichtigt werden. Der Fokus liegt also auf der Beziehung zwischen den gemessenen Spannungswerten und dem Befund der untersuchten Person. Das fertige Modell soll Menschen dabei helfen, eine mögliche Veranlagung zum Alkoholismus festzustellen, sowie anhand dieses Ergebnisses ggf. den eigenen Konsum zu hinterfragen und somit eine u.a. durch Unkenntnis dieser vorhandenen Veranlagung hervorgerufene Alkoholabhängigkeit zu vermeiden. Des Weiteren soll bei der Auswahl des Trainingsalgorithmus, sowie der Architektur, von anderen wissenschaftlichen Arbeiten abgewichen werden, um weitere Trainingsmöglichkeiten in diesem Bereich zu untersuchen. Zusammenfassend ergeben sich unter Berücksichtigung der genannten Aspekte und der in den nächsten Unterkapiteln getroffenen Entscheidungen folgende Forschungsfragen, welche nach der Umsetzung des finalen Modells beantwortet werden sollen:

1. **Wie effizient erweist sich ein FNN zur Untersuchung des Bestehens einer Veranlagung zum Alkoholismus anhand von EEG-Messungen?**

2. **Welche Kombination von Layerarchitekturen und Hyperparametern erweist sich als am treffsichersten zum Trainieren eines FNNs zur Untersuchung des Bestehens einer Veranlagung zum Alkoholismus anhand von EEG-Messungen?**

3.2 Datenselektion

Zunächst muss ein Zieldatensatz gefunden bzw. entwickelt werden. An diesem soll dann das Modelltraining, sowie die anschließende Informationsextraktion, durchgeführt werden.

3.2.1 Auswahlkriterien des Datensatzes

Da in diesem Fall für die Erfüllung der Zielsetzung eine binäre Klassifikation (s. Kapitel 2.1.3), nämlich 'Veranlagung vorhanden' bzw. 'nicht vorhanden', erforderlich ist, wird ein gelabelter Datensatz benötigt. Dieser würde dem System die Spannungswerte bzw. die bei der Messung berücksichtigten Faktoren liefern und das System sollte daraus ableiten, ob die Person eine Veranlagung zum Alkoholismus hat oder nicht.

3.2.2 Auswahl des Datensatzes

Da das gewählte Thema sehr spezifisch ist, ist die Auswahl an möglich verwendbaren Datensätzen begrenzt. Insgesamt wurde nur ein einziger öffentlich zugänglicher Datensatz gefunden, der die beschriebenen Kriterien erfüllt. Dieser ist auch der einzige dieser Art, der in wissenschaftlichen Arbeiten zu diesem Thema verwendet wird [LiWu 2021] und in der Übersicht des Swartz Center for Computational Neuroscience (SCCN) zu frei verfügbaren EEG-Datensätzen in Bezug auf dieses Thema aufgeführt ist [Sccn 2023].

Für diese Arbeit wurde daher der Datensatz *EEG Database* ausgewählt, der 1999 von der University of California, Irvine, zur Verfügung gestellt wurde. Diese Daten stammen aus einer Studie zur Untersuchung von Zusammenhängen in EEG-Messungen in Bezug auf die genetische Veranlagung zum Alkoholismus unter der Leitung von Prof. Dr. Henri Begleiter vom State University of New York Health Center. Die Gesamtzahl der Messfolgen beläuft sich auf 11.074 Dateien (7.033 von Alkoholikern, 4.041 von Kontrollpersonen). Sie enthalten Messungen von 64 auf der Kopfhaut der Versuchspersonen angebrachten Elektroden. Diese wurden pro Untersuchung eine Sekunde lang mit 256 Hz (3,9-Millisekunden-Periode) abgetastet. Zuvor wurden die Probanden in zwei Gruppen, Alkoholiker und Kontrollpersonen, aufgeteilt. Bei jeder Messung wurde den Versuchspersonen entweder ein (in den Tabellen der Messergebnisse mit S1 bezeichnet) oder zwei Reize (S2; Stimuli) dargeboten. Hierbei handelte es sich um Bilder von Objekten, die aus dem Bildsatz von Snodgrass und Vanderwart aus dem Jahre 1980 stammen. Diese Bilder bestehen aus schwarz-weißen Strichzeichnungen, die nach bestimmten Regeln erstellt wurden [SnoVa 1980]. Bei Messungen mit zwei Bildern konnten diese entweder identisch oder unterschiedlich sein.

Dieses Studienkonzept hat den großen Vorteil, dass es Faktoren berücksichtigt, die mit der Feststellung einer Suchtveranlagung zusammenhängen. So wurden die Messungen an Personen im Beta-Zustand durchgeführt, welcher, wie bereits in Kapitel 2.3.2 erläutert, ein wichtiger Marker für die Feststellung einer solchen Veranlagung ist. Dies wurde durch die Konzentration auf die schnell wechselnden Bilder begünstigt. Der komplette Datensatz kann im Literaturabschnitt unter [lcs] eingesehen werden.

Im Folgenden sind Beispielplots einer Kontrollperson und einer alkoholabhängigen Person abgebildet. Die Diagramme zeigen die Zeit, Spannung, sowie die Elektrodenkanäle an und bilden einen Mittelwert von zehn Einzelreizversuchen (Abbildung 7).

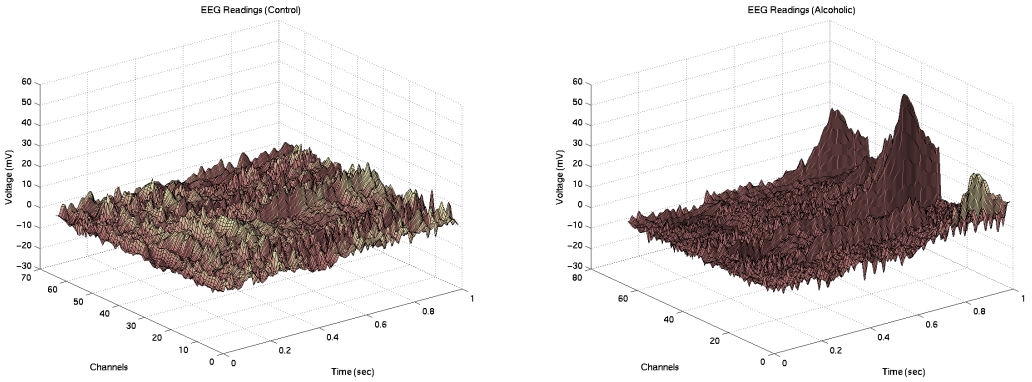


Abbildung 7: Mittelwertplots von zehn Einzelreizversuchen einer Kontrollperson (links) und einer alkoholabhängigen Person (rechts) [Ics]

Jeder der 122 Probanden absolvierte insgesamt 120 Messepochen, bei denen jeweils verschiedene Bilder gezeigt wurden. Die Elektrodenpositionen richteten sich an die *Standard Electrode Position Nomenclature* der American Electroencephalographic Association von 1991 [ShaChaLeLuNuPi 1991].
Jeder Versuch wurde vom Herausgeber in einer separaten Datei gespeichert und im folgenden Format dargestellt:

co2a0000364.rd

$120 Trials, 64 Chans, 416 Samples, 368 post_s tim Samples$

3.906000 msecs uV

S1 obj , trial 0

FP1 chan 0

0 FP1 0 -8.921

0 FP1 1 -8,433

0 FP1 2 -2.574

0 FP1 3 5.239

0 FP1 4 11,587

0 FP1 5 14,028

...

Die ersten vier Zeilen bilden die Kopfdaten. Zeile 1 enthält die Kennung des Probanden, wobei der vierte Buchstabe angibt, ob es sich um einen Alkoholiker (a) oder eine Kontrollperson (c) handelt.

Zeile 4 gibt die verwendete Bildoption an: ein einzelnes Bild (S1obj), zwei übereinstimmende Bilder (S2match) oder zwei unterschiedliche Bilder (S2nomatch). In Zeile 5 beginnen die Daten des entsprechenden Messkanals. Die vier Datenspalten stellen die Versuchsnummer (beginnend bei 0), die Elektrodenposition (Kanal), die Probennummer (0-255) sowie den Impulswert (in Mikrovolt) dar [Ics].

3.3 Datenvorverarbeitung

Die gesammelten Daten müssen gemäß dem in Kapitel 2.1.1.1 beschriebenen KDD-Prozess vorverarbeitet werden. Zu den grundlegenden Schritten gehören u.a. das Sammeln der für die Modellumsetzung notwendigen Informationen, ggf. das Entfernen von Ausreißern sowie die Entscheidung über Strategien zum Umgang mit fehlenden Datenfeldern. Zudem kann im Vorfeld eine Programmiersprache für die Umsetzung der einzelnen KDD-Schritte ausgewählt werden.

3.3.1 Programmiersprache, Frameworks und Umgebung

Um in den nachfolgenden Abschnitten eine Referenz für die Verwendung der jeweiligen Bibliotheken zu haben, wird die Wahl einer Programmiersprache schon in diesem Kapitel getroffen. In diesem Fall wurde sich für Python entschieden, da der Bereitstellung von effizienten und spezifizierten Bibliotheken für das Training eines automatisierten Systems [Cou], sowie einer zeitsparenden Verarbeitung der Datenmenge [Int], eine deutliche Priorität zugewiesen wird. Darüber hinaus können alle Schritte des KDD-Prozesses auf praktische Weise in Python umgesetzt werden. So kann das Einlesen der Daten, sowie die Datenbereinigung, mithilfe von Pandas und NumPy, die Visualisierung dieser mittels matplotlib bzw. Seaborn [Sea], und das Trainieren des Modells mit Tensorflow/Keras und scikit-learn [Sci] realisiert werden.
Auch R und RapidMiner, die in Kapitel 2.1.2.1 beschrieben wurden, wären sinnvolle Alternativen gewesen. Zwar eignet sich R besonders gut für statistische Analysen und Visualisierungen, ist aber im Vergleich zu Python in Bezug auf die Programmierung und die dazu gehörenden Features, auch im Bereich des maschinellen Lernens, etwas weniger vielseitig [Luna 2022] [Int]. GUI-basierte Werkzeuge wie RapidMiner ermöglichen durch Drag-and-Drop-Funktionen und vorgefertigte Module zwar ein schnelles Prototyping, können aber bei der Konstruktion analytischer Werkzeuge etwas weniger robust und präzise sein [Diaz 2021].
Bezüglich des Frameworks ähneln sich die beiden in Kapitel 2.1.2.1 vorgestellten Optionen in ihren Funktionalitäten. TensorFlow bietet eine deutlich bessere Ergebnisvisualisierung, was das Debugging und die Analyse der Resultate, erheblich vereinfacht [Ter 2023].
Zudem verfügt es über eine umfangreichere Auswahl an Bibliotheken, insbesondere im Hinblick auf vordefinierte Modellarchitekturen, was den Modellerstellungsprozess wesentlich vereinfacht [Boe].

PyTorch zeichnet sich durch seine Schnelligkeit und die Möglichkeit aus, die Berechnungsgrafik zur Laufzeit anzupassen und so den Code zu debuggen, was bei TensorFlow nicht möglich ist [Boe].
Angesichts des behandelten Themas und der Konstruktion eines automatisierten Systems, würden vordefinierte Modelle den Prozess der Architekturimplementierung erheblich beschleunigen, was gegenüber einer allgemein schnelleren Verarbeitung bevorzugt wurde. Aus diesem Grund wurde TensorFlow gewählt, obwohl beide Optionen anwendbar wären.
Die Ausführung des Codes soll in Jupyter Notebook erfolgen, da diese Plattform den Vorteil bietet, dass mit den entsprechenden Python-Bibliotheken sehr einfach übersichtliche 3D-Plots und Tabellen, die zur Analyse des Datensatzes benötigt werden, visualisiert und Modelle trainiert werden können. Außerdem kann der Code in Zeilen organisiert werden, sodass Teile davon umgeschrieben und wiederverwendet werden können [Jup].
Nun soll darüber entschieden werden, wie die Daten vorverarbeitet werden sollen.

3.3.2 Ausbalancierung des Datensatzes

Für das Training soll der vollständige Datensatz mit allen 122 Probanden verwendet werden, von denen jeweils eine Anzahl zwischen 30 und 120 Versuchen gespeichert wurde. Somit wurden nicht alle Messfolgen vom Herausgeber des Datensatzes zur Verfügung gestellt. Die beschriebene Datenverteilung beträgt ca. 63%/37% und entspricht damit einer Verteilung von ca. 1,7:1, was als akzeptabel angesehen wird. Somit soll zunächst kein Over- oder Undersampling durchgeführt werden. In Kapitel 3.5.5 werden jedoch Fälle beschrieben, in denen die Anwendung dieser Methoden weiterhin sinnvoll wäre.
Spezifische Einschränkungen des Datensatzes sind nicht erforderlich, da alle Messungen für die zu bearbeitenden Fragestellungen relevant sind und keine Beschränkung auf z.B. eine bestimmte Bildkondition oder einen bestimmten Kanal erfordern. Daher wurde entschieden, alle validen Daten zu verwenden, anstatt Teilmengen oder Samples des gesamten Datensatzes zu nehmen.

3.3.3 Umgehen mit leeren und fehlerhaften Daten

Wie mit leeren Daten umgegangen werden soll, kann erst bei der Umsetzung genau entschieden werden. Es wurde jedoch beschlossen, dass, wenn der leere/fehlende Anteil für den Datensatz nicht relevant ist, z.B. weniger als 1% des gesamten Datensatzes ausmacht, er aus dem endgültigen Datensatz entfernt werden sollte. Ist der Anteil größer, sollte der Mittelwert oder der Median, je nachdem, ob Ausreißer vorhanden sind, zur Vervollständigung der Datenfelder angewendet werden.

3.3.4 Feature Selection

Bei den Features werden zunächst alle möglichen Merkmale, die in den Dateien erwähnt werden, aufgenommen. Der erste Schritt besteht darin, zu prüfen, ob die Features nur einen einzigartigen Wert in ihrer Spalte haben. Dies kann mit der NumPy-Funktion *unique()* festgestellt werden. Ist dies der Fall, werden sie nicht weiter berücksichtigt. Hierbei ist zu beachten, dass eine Korrelationsanalyse der verbliebenen Merkmale in diesem Fall theoretisch nicht durchführbar wäre, da für die Komponenten, die z.B. eine zeitliche Abfolge darstellen, weder auf der Quellenseite des Datensatzes, noch in den zugehörigen Arbeiten, eine medizinisch validierte Aussage über den Befund des Patienten festzuhalten ist und diese somit innerhalb der Spannungswerte gruppiert werden könnten. Dadurch würden sie letztendlich nicht als einzelne Features in den endgültigen Datensatz aufgenommen werden. Da diese Entscheidung jedoch nur mutmaßlich getroffen werden kann, wurde sicherheitshalber beschlossen, beide Datensatzversionen, mit und ohne Gruppierung, zu verwenden, um keine möglich relevanten Faktoren auszulassen. Hierbei ist jedoch anzumerken, dass die Zeilen im nicht-gruppierten Datensatz jeweils ein Sample eines bestimmten Kanals repräsentieren würden. Somit würden auch im späteren Testsatz einzelne Samples als Daten vorliegen. Es ist jedoch sehr unwahrscheinlich, dass man anhand eines einzigen Spannungswertes auf den Befund schließen kann. Sollte somit, während des ersten Trainings festgestellt werden, dass der nicht-gruppierte Datensatz eine viel schlechtere Performance oder, aufgrund der Auflistung einzelner Spannungswerte in jeweils einer Zeile, einen viel höheren zeitlichen Aufwand als der gruppierte Datensatz aufweist, wird dieser verworfen und es wird nur mit dem gruppierten Datensatz weitergearbeitet. Natürlich wäre es vorteilhafter, den Kanal bzw. die Probenummer als einzelne Features aufzunehmen, im ungünstigen Fall sind diese Informationen jedoch im gruppierten Datensatz mitvertreten, da das Modell auch an der Reihenfolge der Spalten lernt und die Spannungswerte in der richtigen Reihenfolge nach Kanal/Probenummer gruppiert werden.
Somit werden zwei Modelle mit den jeweiligen Datensätzen parallel trainiert. Darauf wird in Kapitel 3.5.5 näher eingegangen. Abgesehen von diesen Merkmalen weisen die anderen Features, wie in Kapitel 3.4 erläutert wird, in ihren Werten Label-Codierungen auf, was somit eine Verwendung der Korrelationsanalyse ausschließen würde. Die restlichen Features sollen durch Plots in Bezug auf die Spannungswerte auf Zusammenhänge untersucht werden. Auf diese Weise werden die relevantesten Merkmale ausgewählt, die den größten Beitrag zur Vorhersagekraft des Modells leisten.

3.3.5 Bereinigung von Ausreißern

Da die Spannungswerte das aussagekräftigste Feature für den Patientenbefund darstellen, sollten alle Ausreißer, unabhängig von ihrer Anzahl, bereinigt werden.

Dadurch wird sichergestellt, dass alle Werte des Datensatzes in einem definierten und nicht zu großen Wertebereich liegen.
Anhand der Verteilung der Daten soll im späteren Verlauf entschieden werden, ob mit dem Z-Score oder dem IQR gearbeitet werden soll. Bei der Extremwertbereinigung soll das Capping angewendet werden, um zu verhindern, dass eine ggf. hohe Anzahl an Ausreißer-Werten die Gesamtzahl der Daten stark beeinträchtigt.
Wichtig zu beachten ist, dass die Bereinigung separat für die Alkoholiker- und die Kontrollgruppe durchgeführt werden muss, da diese, durch die in Kapitel 2.3.2 beschriebenen Zusammenhänge, unterschiedliche Wertebereiche aufweisen könnten.
Da die Spannungswerte in beiden Datensatzversionen gleich sind, kann die Bereinigung nur einmal durchgeführt und muss nicht separat auf beide Datensätze angewendet werden.

3.3.6 Korrelationsanalyse

Die Korrelationsanalyse sollte bei dem gruppierten Datensatz auf die Spannungswerte untereinander angewendet werden, um mögliche irrelevante Spalten finden zu können. Bei dem nicht-gruppierten Datensatz kann die Korrelationsanalyse sowohl auf die Probenummern als auch auf die Kanäle in Bezug auf die Spannungswerte eingesetzt werden. Dadurch kann die Datenmenge unter Umständen erheblich reduziert werden, ohne dass die Qualität der Vorhersage wesentlich beeinträchtigt wird. Hierbei wird zunächst als Regel festgelegt, dass bei einem hohen Korrelationskoeffizienten von 0,975 eine der betroffenen Spalten entfernt wird. Es wurde ein sehr hoher Koeffizient gewählt, um nicht zu viele Daten auf einmal löschen zu müssen und um die Unterschiede in den Ergebnissen bei Verwendung verschiedener Koeffizienten besser untersuchen zu können. Es ist dabei wichtig zu verstehen, dass Korrelation nicht notwendigerweise Kausalität bedeutet. Die Tatsache, dass zwei Variablen miteinander korrelieren, bedeutet nicht automatisch, dass die eine Variable die andere verursacht. Es könnte eine verborgene Variable geben, die beide beeinflusst, oder es könnte sich um reinen Zufall handeln. Da jedoch die Untersuchung der Beziehungen zwischen allen einzelnen Spannungsspalten auf Kausalität in diesem Fall einen zu hohen Zeitaufwand beanspruchen würde, wird dies nicht durchgeführt.

3.4 Datentransformation

In diesem Schritt sollen die nützlichen Merkmale zur Darstellung der Daten herausgearbeitet werden. Durch Dimensionsreduktion oder Transformationsmethoden kann die effektive Anzahl der betrachteten Variablen reduziert und es können ggf. invariante Darstellungen für die Daten gefunden werden.

Da der KDD-Prozess interaktiv ist [FaPiSm 1997, Seite 42] und somit auch Anpassungen erlaubt sind, werden einige Transformationen, die für den Schritt der Datenvorverarbeitung notwendig sind, vorgezogen.

Da die Dateien im .tar.gz-Format vorliegen, müssen sie extrahiert und entsprechend transformiert werden. Nach der Extraktion der einzelnen Messdateien sollen die Daten zunächst in einer gemeinsamen .csv-Datei zusammengefasst werden. Dies hat den Vorteil, dass Spalten für einzelne Features und Labels leicht angelegt werden können, was die Zuordnung von Merkmalen zu Labels für das überwachte Lernen erleichtert. Außerdem können Datenbereinigungen effizienter durchgeführt werden, da alle Daten in einer gemeinsamen Datei gespeichert sind [Csv 2023]. Hierbei ist es wichtig zu beachten, dass die Werte, die als Ausgabewerte vorgesehen sind, hier die Befundklassifikation, in einer eigenen Spalte stehen sollen. So kann bei der Aufteilung des Datensatzes sehr einfach zwischen Eingabe- und Ausgabedaten unterschieden werden. Mithilfe von Python und ihrer Pandas-Bibliothek können diese auch sehr leicht ausgelesen und angezeigt werden.

Da die Datentabelle unkomprimiert und unbearbeitet Größen im zweistelligen Gigabyte-Bereich erreichen würde, ist es wichtig, nur solche Komponenten für das Training zu verwenden, die auch Rückschlüsse auf das Messergebnis zulassen, was bei der Feature Selection im Schritt der Datenvorverarbeitung geschieht. Dabei ist zu beachten, dass Features, die keinen Einfluss auf den Befund haben, aus dem Datensatz entfernt werden. Bei dem gruppierten Datensatz sollte jeder Wert innerhalb einer Messung in einer eigenen Spalte gespeichert werden. Dies stellt sicher, dass jede Zeile die gleiche Anzahl von Werten enthält und erleichtert zudem die Korrelationsanalyse zwischen den Spannungswerten. Somit würde jede Zeile der Datensatztabelle jeweils einen vollständigen Versuch darstellen.

Darüber hinaus sollen die Zeilenwerte einen möglichst einheitlichen Datentyp besitzen, da dies nicht nur die Implementierung und das Debugging von ML-Modellen erheblich erleichtert, sondern u.a. auch eine effizientere Speichernutzung ermöglicht. Da die meisten Features keine numerischen Abstände oder Sequenzen bilden, können sie als nominale Daten vorliegen. Da jedoch die Spannungswerte, die als Fließkommazahlen gespeichert sind, das wichtigste Element des Datensatzes bilden, sollten die restlichen Spaltenwerte, der Homogenität und der strukturierteren Darstellung innerhalb des Datensatzes halber, ebenfalls numerisch dargestellt werden. Somit soll an diesen eine einfache *Label-Codierung* durchgeführt werden. So kann z.B. die Bildkondition als 0 für S1obj, 1 für S2match und 2 für S2nomatch gespeichert werden.

Eine Normalisierung der Daten kann in diesem Fall nur für die Spannungswerte durchgeführt werden. Dies ist jedoch nicht notwendig, da alle Spannungswerte die gleiche Einheit besitzen und eine sehr große Wertespannweite durch die Behandlung der Ausreißer mit IQR verhindert wird. Eine Normalisierung würde daher keine großen Änderungen in der Leistungsfähigkeit des Modells bewirken.

3.5 Data Mining

Hier erfolgt die Auswahl eines Data-Mining-Algorithmus, sowie der Methoden, die für die Suche nach Datenmustern verwendet werden sollen. Dieser Prozess umfasst die Entscheidung über geeignete Modelle und Parameter sowie die Abstimmung einer bestimmten Data-Mining-Methode mit den allgemeinen Kriterien des KDD-Prozesses.

3.5.1 Klassifikationsmethode

In Bezug auf das anzuwendende Klassifikationsverfahren, können neuronale Netze besser mit komplexen Merkmalen umgehen als einige klassische Verfahren [Pai 2023], was im vorliegenden Fall aufgrund der in Kapitel 3.2.2 dargestellten vielfältigen Zusammenhänge zwischen den verschiedenen Merkmalen von Vorteil ist. Darüber hinaus sind sie sehr einfach auf metrische Daten anzuwenden und können in diesem Fall auch auf nominale Daten angewendet werden, da diese bei der Datentransformation durch die Labelkodierung numerisch dargestellt werden. Auf der anderen Seite können klassische Verfahren leichter mit einer begrenzten Menge an Daten arbeiten [Har 2018].
Im Prinzip sind beide Optionen auf das Problem anwendbar, jedoch kann in so einem spezifischen Gebiet nicht ausgeschlossen werden, dass die Beziehungen in dem Datensatz, die zu einer Ergebnisführung beitragen, eine große Komplexität aufweisen, weshalb NNs eine sicherere Variante wären und auch für das weitere Vorgehen ausgewählt werden. Darüber hinaus wurden in vergleichbaren Arbeiten viele klassische Algorithmen [KamArdPan 2020] [BurAhBa 2021] verwendet, während eine bestimmte Art von NNs, wie in Kapitel 3.7 gezeigt wird, auf diesem Gebiet bisher weitgehend unerforscht geblieben ist.

3.5.2 KNN-Architektur

Hinsichtlich der konkreten Architektur des NNs soll die in Kapitel 2.1.3.2 vorgestellte Feedforward-Architektur verwendet werden. FNNs eignen sich gut für binäre Klassifikationsaufgaben mit kontinuierlichen Features [Kum 2020]. Auch gegenüber anderen Architekturen, wie CNNs, sind FNNs nicht auf räumliche Strukturen, wie Bilder, angewiesen und können für ein breiteres Aufgabenspektrum eingesetzt werden. Darüber hinaus eignen sich FNNs gut für die Verarbeitung von kontinuierlichen Fließkomma-Daten, da sie die natürliche Darstellung von reellen Zahlen unterstützen, ohne die Notwendigkeit der Datenkodierung oder -transformation. Sie sind bei der Verarbeitung der Daten auch schneller als RNNs [Lin]. Dies wäre in diesem Fall von Vorteil, da hierdurch nicht nur eine größere Anzahl an Messungen in kürzerer Zeit verarbeitet werden könnte, was das medizinische Personal entlasten würde, sondern auch das Endergebnis schneller bei den PatientInnen ankommen und bei ihnen für Gewissheit sorgen würde.

Obwohl FNNs in Bezug auf Sequenzdaten und Messungen in zeitlicher Abfolge gegenüber RNNs im Nachteil sind und RNNs auch häufiger für diese Problemstellungen eingesetzt werden [Pra 2020], wurde sich gegen diese entschieden, da die zeitlichen Zusammenhänge in den Spannungsmessungen in diesem Fall keine besondere Bedeutung aufweisen und die Vorhersagen hauptsächlich von den Messwerten an sich abhängen. Obwohl FNNs aufgrund der einfacheren Trainingsstruktur anfälliger für Overfitting sind [Lin], kann dies u.a. durch die Verwendung von Cross Validation sehr gut kompensiert werden. Darüber hinaus sind FNNs auch einfacher und möglicherweise weniger kostenintensiv zu trainieren als RNNs [Pra 2020]. Dies ist für den vorliegenden Fall ebenfalls von Vorteil, da bei der Menge an Modellanpassungen, die in Kapitel 3.5.5 näher vorgestellt werden, der Aspekt der Zeitersparnis eine wichtige Rolle spielt. Für diesen speziellen Fall werden auch keine Informationen aus früheren Iterationen benötigt.
In Python können FNNs mit den Bibliotheken Tensorflow/Keras, sowie scikit-learn, implementiert werden.

3.5.3 Datensatzaufteilung

Auf Grundlage der in Kapitel 2.1.3.4 beschriebenen Verteilungsquoten, wurde sich für die gängigste Option [Bah 2021], die 80-10-10-Aufteilung bzw. 90-10 bei cross-validation, entschieden, da es generell keine perfekte Aufteilung für einen bestimmten Fall gibt. Falls beim Training eine schlechte Performanz bestehen sollte, wird jedoch offen gehalten, weitere Aufteilungsraten zu testen. Stratified k-fold cross-validation soll hierbei ebenfalls verwendet werden, da sie weniger verzerrte/optimistische Ergebnisse für ein Modell liefert. Allerdings wird sie aufgrund der viel größeren zeitlichen Anteilnahme erst beim bis zu dem Zeitpunkt genauesten Modell eingesetzt. Für die Bestimmung des k-Wertes gibt es ebenfalls keine definierte Regel [Shi 2022]. Für das Ausgangsmodell wurde ein Wert von 9 gewählt, damit eine 90-10 bzw. 80-10-10-Aufteilung weiterhin garantiert werden kann.

Für den Aufbau des neuronalen Netzes sollen verschiedene Kombinationen aus Layeranzahl, Neuronenanzahl in den einzelnen Layern, Aktivierungsfunktionen, Optimizern, Loss-Funktionen, sowie weiteren Hyperparametern getestet werden. Hierbei sollen möglichst viele Hyperparameter auf ihren Einfluss auf die Trefferwahrscheinlichkeit getestet und für jeden ein Optimum gefunden werden.

3.5.4 Konfiguration des FNNs

3.5.4.1 Optimierung der FNN-Architektur

Um die optimalen Werte möglichst zeitsparend zu finden, soll eine Optimierungsfunktion verwendet werden. Hierfür wird die Python-Bibliothek Optuna [Opt] implementiert.
Sie kann für das automatische Tuning von Hyperparametern von ML-Modellen verwendet werden. Optuna verwendet eine Optimierungsmethode, um die besten Hyperparameter-Konfigurationen für ein Modell zu finden. Dadurch können Modelle optimiert und ihre Leistung verbessert werden, ohne dass Parameter manuell ausprobiert werden müssen. Sie ist dadurch sehr nützlich, um Zeit und Ressourcen bei der Modellentwicklung zu sparen.

3.5.4.2 Layerarchitektur

In die Inputschicht werden die in den vorherigen Schritten als Eingabedaten verwendeten Features eingegeben. Die genaue Anzahl der Eingangsneuronen kann erst durch die Feature Selection in der Datenvorverarbeitung genau bestimmt werden, es ist jedoch davon auszugehen, dass bei einer Gruppierung der Spannungswerte, sowie der Darstellung jedes Wertes in einer eigenen Spalte, die Anzahl der Eingänge vor der Datenreduktion bei min. *256 Samples/Kanal * 64 Kanäle = 16384 Spannungswerten* liegen wird. Beim nicht-gruppierten Datensatz würde die Größe der Inputschicht im einstelligen Bereich liegen, da nur ein einzelner Spannungswert pro Zeile aufgelistet wird.
Die einzelnen Parameter der Hidden Layer bilden den Kern des FNNs. Generell soll mithilfe von der Bibliothek Keras ein *Sequential-Model* gebaut werden. Sequential ist eine Klasse in Keras, die es ermöglicht, sequentielle Modelle zu erstellen, in denen die Schichten linear aufeinanderfolgen [Fch]. Diese Schichten werden *Dense-Layer* genannt [Tut]. Mit Optuna soll die Anzahl der Dense-Layer, die Anzahl der Units innerhalb der Dense-Layer, sowie die einzelnen Aktivierungsfunktionen analysiert und die Ergebnisse, die zur bestmöglichen Accuracy führen, herausgegeben werden.
Bei der Outputschicht soll die Wahrscheinlichkeit für das Eintreten der jeweiligen Klassen ausgegeben werden können. Hierzu gibt es einmal die Möglichkeit zwei Outputs mit der *Softmax*-Aktivierungsfunktion einzubauen. Die Softmax-Funktion sorgt dafür, dass die Ausgabewerte zwischen 0 und 1 liegen und sich zu 1 summieren. Dies ermöglicht eine Wahrscheinlichkeitsverteilung über die verschiedenen Klassen. Die zweite Möglichkeit besteht darin, ein Ausgangsneuron mit der *Sigmoid*-Aktivierungsfunktion zu verwenden. Dabei wird die Ausgabe ebenfalls zwischen 0 und 1 skaliert, um die Wahrscheinlichkeit für das Eintreten einer bestimmten Klasse darzustellen. Für diesen Fall soll nur ein Ausgangsneuron verwendet werden, da beide Varianten ähnlich sind, jedoch aufgrund der einfacheren Struktur weniger Gewichte trainiert werden müssen und der Trainingsprozess somit schneller voranschreiten würde [Bro 2022].

Hinzu soll sich für die Umsetzung offen gehalten werden, Dropout- und Batch-Normalization-Layer einzubauen. Dies wird in Kapitel 3.5.5 thematisiert.

3.5.4.3 L1-/L2-Regularisierung

Da die Hauptaufgabe von Regularisierung darin besteht, Overfitting zu lösen [Tya 2021], sollen L1 bzw. L2 erst dann implementiert werden, wenn tatsächlich starkes Overfitting festgestellt werden sollte. Generell würde aber in diesem Fall, sowohl L1, L2, als auch die Kombination von beidem ausprobiert werden, da beide ihre Vorteile in diesem Modell bieten könnten.

3.5.4.4 Optimizer

Bei der Frage nach dem Optimizer kommen verschiedene Möglichkeiten auf. Adaptive Optimizer sind hier generell besser geeignet, da die Lernrate automatisch angepasst wird, im Gegensatz zu solchen, die auf klassischen stochastischen Algorithmen basieren. Bei der konkreten Auswahl soll mit dem Adam-Optimizer [Adam] gearbeitet werden, da dieser für viele Anwendungsfälle, statistisch betrachtet, die geringsten Trainingskosten aufweist und zudem zeitsparend ist (Abbildung 8) [Gup 2023].

Optimizer	Epoch 1 Val accuracy | Val loss	Epoch 5 Val accuracy | Val loss	Epoch 10 Val accuracy | Val loss	Total Time
Adadelta	.4612 | 2.2474	.7776 | 1.6943	.8375 | 0.9026	8:02 min
Adagrad	.8411 | .7804	.9133 | .3194	.9286 | 0.2519	7:33 min
Adam	.9772 | .0701	.9884 | .0344	.9908 | .0297	7:20 min
RMSprop	.9783 | .0712	.9846 | .0484	.9857 | .0501	10:01 min
SGD with momentum	.9168 | .2929	.9585 | .1421	.9697 | .1008	7:04 min
SGD	.9124 | .3157	.9569 | 1451	.9693 | .1040	6:42 min

Abbildung 8: Übersicht über die Accuracy und den Zeitverbrauch von Optimizern [Gup 2023]

3.5.4.5 Loss functions

Für binäre Klassifizierungsprobleme gibt es, wie bereits in Kapitel 2.1.3.8 erläutert, mehrere Möglichkeiten für Verlustfunktionen. Aufgrund der dort beschriebenen Vorteile wird die Binary Cross-Entropy Loss Function verwendet. Darüber hinaus zeigt sie im Vergleich zu den anderen Alternativen bessere Ergebnisse sowohl bei der Entwicklung des Verlustes als auch bei der Genauigkeit während des Trainings (Abbildungen 9-11) [Bro 2020].

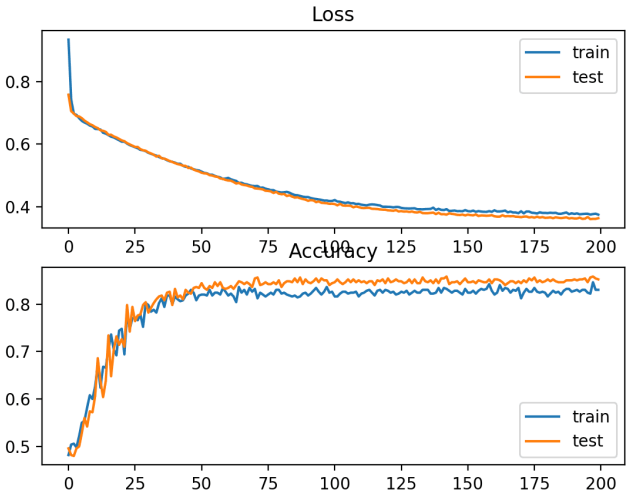


Abbildung 9: Loss und Accuracy bei Verwendung der Binary Cross-Entropy Loss function [Bro 2020]

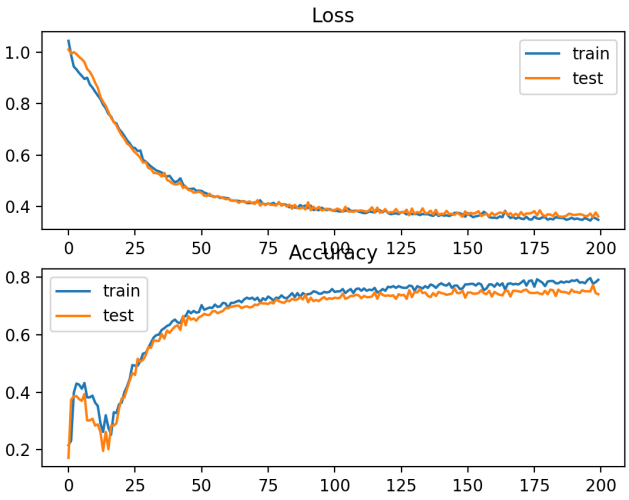


Abbildung 10: Loss und Accuracy bei Verwendung der Hinge Loss function [Bro 2020]

3.5.4.6 Batch-Size

Die Wahl der Batch-Größe hat Auswirkungen auf das Training und die Leistung des Modells. Größere Batches sind schneller zu verarbeiten, während kleinere eine viel geringere Fehleranfälligkeit zur Folge haben [Tha 2023]. Eine übliche Variante für eine Batchgröße ist ein Wert, der in der Mitte liegt, z.B. 32.

3.5.5 Trainingsablauf

Trainiert wird auf einem Privatcomputer mit einer Nvidia GeForce RTX 4070 Ti 12GB GPU.

Als Erstes sollen die beiden Modelle mit den zunächst beschlossenen Konfigurationen trainiert werden.

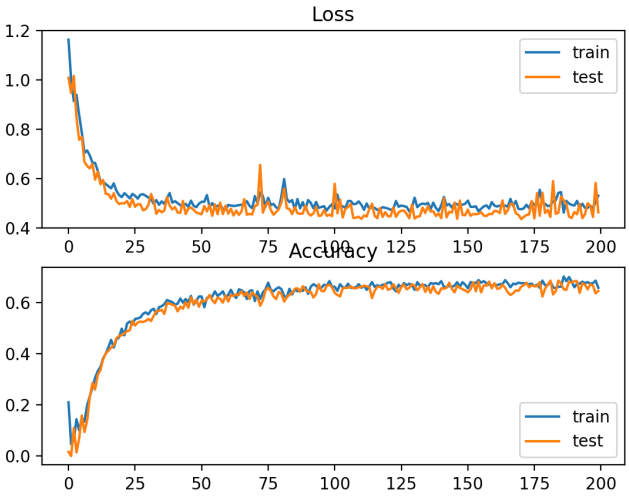


Abbildung 11: Loss und Accuracy bei Verwendung der Squared Hinge Loss function [Bro 2020]

Dies liefert einen Überblick darüber, in welchem Bereich die Genauigkeit der Vorhersagen liegt. Mit Hilfe von Optuna soll versucht werden, ein Optimum für die einzelnen Hyperparameter beider Modelle zu finden.

Wie in Kapitel 3.3.4 erwähnt, wird der nicht-gruppierte Datensatz bei einer, im Vergleich zum gruppierten Datensatz, viel schlechteren Accuracy oder einem viel höheren zeitlichen Anspruch, verworfen.

Durch Anpassungen der beschriebenen Methoden/Parameter soll versucht werden, eine bessere Performance als beim ersten Versuch zu erreichen. Durch die Konstruktion der ersten Modelle kann bei diesem Schritt überprüft werden, wie effektiv sich die angewandten Änderungen auf die Genauigkeit auswirken. Sollten die Ergebnisse nicht zufriedenstellend sein, sollen weitere Parameter im Laufe des Umsetzungsprozesses angepasst werden, um eine höchstmögliche Accuracy zu erreichen. Nach den bereits begründeten Ausschlüssen einiger Anpassungen kommen hierbei noch folgende infrage:

- **Einsatz von Over- bzw. Undersampling**
- **Anpassung des Umgangs mit leeren Daten (abhängig vom Ergebnis der Datenbereinigung)**
- **Ändern der Features**
- **Verwendung anderer Korrelationskoeffizienten**
- **Einbauen von Cross-Validation beim bislang genauesten Modell**
- **Ändern der Datensatzaufteilung**
- **Einfügen einer Dropout-Layer**

- **Einbauen einer Batch-Normalization-Layer**
- **Verwendung von L1-/L2-Regularisierungen**
- **Ändern der Batch-Size**

Es könnten jedoch, abhängig von den Beobachtungen bei der Umsetzung, weitere Anpassungsentscheidungen getroffen werden. Welche Anpassungen letztendlich umgesetzt werden, soll anhand der Analysetools des trainierten Modells entschieden werden. So kann z.B. Over- und Undersampling angewendet werden, wenn die False Positive- bzw. False Negative-Rate in der Confusion Matrix hoch ist.
Zusätzlich soll dem Training eine *EarlyStopping*-Funktion als Callback mitgegeben werden. Sie überprüft die Entwicklung des Verlustes beim Validierungsset und bricht das Training ab, wenn dieser sich nicht innerhalb von fünf Epochen verbessern sollte. Dadurch kann sehr viel Zeit für überschüssiges Training beim Prozess gespart werden.

3.6 Interpretation und Evaluation

Die trainierten Modelle sollen mit den in Kapitel 2.1.4 vorgestellten Analysetools untersucht und aus den Ergebnissen Verbesserungsmöglichkeiten abgeleitet werden. Nach Überprüfung der meisten Faktoren soll im Endergebnis das Modell mit der höchsten Genauigkeit ausgewählt werden.
Dabei ist zu beachten, dass dem Recall der Alkoholikergruppe, also wie viele der tatsächlichen Alkoholismus-Gefährdeten auch als gefährdet erkannt wurden, eine besondere Bedeutung zugewiesen und gegenüber der Precision priorisiert wird. Es wäre viel fataler, wenn ein Modell eine Person, die die Veranlagung hat, als ungefährdet bestimmen würde als umgekehrt.

Um eine Einzigartigkeit der Thesis zu gewährleisten, soll sich im Folgenden mit vergleichbaren Arbeiten auseinandergesetzt, sowie mögliche Umsetzungsvarianten, im Vergleich zu diesen, aufgezeigt werden.

3.7 Auseinandersetzung mit vergleichbaren Arbeiten

Die meisten wissenschaftlichen Arbeiten, die sich mit einem KI-System im Bereich des Alkoholismus und der EEG beschäftigen, verwenden CNN-Architekturen. Dabei werden aus den Messdaten erstellte 2D-/3D-Plots für das Training verwendet [MuQaZa 2021]. In dieser Arbeit soll ein anderer Ansatz, nämlich die FNN-Architektur, auf seine Effizienz hin untersucht werden.
Diese ist, in Bezug auf diese Thematik, nur bedingt erforscht, es konnten lediglich zwei kurze Dokumentationen gefunden werden, die ein ähnliches Vorgehen hatten. Bei ihnen wurden jedoch fest definierte Modellarchitekturen mit Anpassungen einiger Hyperparameter verwendet [Tho] [FaSiKaWa 2020].

Hier soll mithilfe von Optuna die bestmögliche Architektur gefunden werden, wofür hunderte von einzelnen Modellen trainiert und dazu, falls nötig, weitere Anpassungen, wie z.B. Over- und Undersampling getestet werden.
Auch für das generelle Thema wird ein anderer Ansatz verwendet. In den veröffentlichten Arbeiten liegt der Schwerpunkt meist auf dem Vorhandensein von Alkoholismus bei Personen [LiWu 2021]. In dieser Arbeit spielt der Aspekt der Vererbung eine wesentliche Rolle. Durch die Untersuchung von Nachkommen von Alkoholikern auf eine Veranlagung und den möglichen Nachweis einer solchen Veranlagung sollen die Betroffenen gewarnt und somit einer möglichen Entwicklung einer Alkoholabhängigkeit vorgebeugt werden. Eine Erwähnung oder Berücksichtigung dieses Aspektes findet sich in den genannten Arbeiten nicht.
Eine kurze Analyse der verwendeten Spannungswerte zur Absicherung der in Kapitel 2.3.2 beschriebenen Zusammenhänge wird in anderen Arbeiten ebenfalls nicht thematisiert.
Abschließend sind Arbeiten dieser Art in der Regel in englischer Sprache verfasst, eine deutschsprachige ist, wenn überhaupt, nur begrenzt zu finden.

4 Umsetzung

In diesem Kapitel sollen die in Kapitel 3 herausgearbeiteten Konzeptaspekte umgesetzt werden. Dabei wird in den einzelnen Umsetzungsschritten tiefer ins Detail gegangen und ein Überblick über die angewendeten Techniken gegeben. Hierbei wird, wie in Kapitel 3, nach dem KDD-Prozess vorgegangen.

4.1 Bereitstellung der Python-Codes

Die einzelnen in Python geschriebenen Jupyter Notebook-Dateien können dem folgenden Link entnommen werden: https://drive.google.com/drive/folders/1BrdeN1X_y5Jq8QsVOQArdKlm8wrOTJu5?usp=sharing

4.2 Datenvorverarbeitung/Datentransformation

Da der zu verwendende Datensatz bereits in Kapitel 3.2.2 vorgestellt wurde, durch die Verwendung aller validen Daten kein Aufteilen des Datensatzes benötigt wird und die Datenselektion somit abgeschlossen ist, kann direkt mit der Datenvorverarbeitung bzw. mit der Datentransformation begonnen werden.
Wie in Kapitel 3.4 bereits erwähnt, sollen die einzelnen Messungen aus den .tar.gz-Dateien, in denen die Messdateien zunächst vorliegen, extrahiert und in eine neu erstellte .csv-Datei geschrieben werden (Code 4.1). Somit werden von der Reihenfolge her die Schritte der Datentransformation mit denen der -vorverarbeitung kombiniert.
Nach diesem Schritt kann der Prozess mit dem Auffinden von fehlerhaften Daten sowie der Feature Selection fortgeführt werden.

4.2.1 Umgehen mit leeren und fehlerhaften Daten

Damit der genaue Anteil an fehlerhaften Daten berechnet werden kann, muss zunächst herausgefunden werden, wie viele Messungen der Datensatz insgesamt beinhaltet.
Generell wurden im Extraktionsschritt 11.074 Messdateien gefunden, wo von 17 keine Daten enthielten. Dies wurde auch auf der Quellseite des Datensatzes kommuniziert [Ics]. In jeder der 11.057 validen Dateien wurde mit jeweils 64 Kanälen und hierbei jeweils 256 Proben gemessen. Die Gesamtzahl an Einzelmessungen beläuft sich somit auf 11.057 * 64 * 256 = **181.157.888**.
Von dieser Gesamtzahl muss der fehlerhafte Anteil ermittelt werden (Code 4.2). Wie auf der Quellseite vermerkt, sind die fehlerhaften Daten in der Bildkondition mit einer *err*-Zeichenfolge gekennzeichnet. Dadurch kann diese Anzahl und somit der Anteil leicht ermittelt werden:

Gesamtanzahl der Datensatzzeilen: **181.157.888**

Anzahl der gespeicherten Nomatch-Err-Messwerte: **573.440**

Anzahl der gespeicherten Match-Err-Messwerte: **983.040**

Anzahl der gesamten Err-Messwerte: **1.556.480**

Anteil der Err-Messwerte zur Gesamtanzahl: **0,859%**

Da der fehlerhafte Anteil weniger als 1% des Gesamtdatensatzes darstellt, werden diese Dateien, nach Kapitel 3.3.3, entfernt und für das weitere Vorgehen nicht beachtet. Nach der Bereinigung besteht eine Gesamtmessanzahl von **179.601.408**.

4.2.2 Feature Selection

Nach Kapitel 3.3.4 sollen für die Feature Selection zunächst alle in den Dateien erwähnten Merkmale in die .csv-Datei aufgenommen werden. Als erste Version der einzelnen Dateienverarbeitungen ergibt sich folgende speicherlastige Tabelle (Code 4.2, Abbildung 12):

	trial	channel	sample_no	voltage	label	total_trials	total_channels	total_samples	epoch_time	matching_condition
0	0	FP1	0	-8.921	co2a0000364.rd	120 trials	64 chans	416 samples 368 post_stim samples	3.906000 msecs uV	S1 obj
1	0	FP1	1	-8.433	co2a0000364.rd	120 trials	64 chans	416 samples 368 post_stim samples	3.906000 msecs uV	S1 obj
2	0	FP1	2	-2.574	co2a0000364.rd	120 trials	64 chans	416 samples 368 post_stim samples	3.906000 msecs uV	S1 obj
3	0	FP1	3	5.239	co2a0000364.rd	120 trials	64 chans	416 samples 368 post_stim samples	3.906000 msecs uV	S1 obj
4	0	FP1	4	11.587	co2a0000364.rd	120 trials	64 chans	416 samples 368 post_stim samples	3.906000 msecs uV	S1 obj

Abbildung 12: Struktur der großen Datentabelle [eigene Darstellung]

Zunächst müssen alle Features entfernt werden, die nur einen einzigartigen Wert besitzen. Durch die Python-Funktion *unique()* konnte festgestellt werden, dass die Spalten *total_trials*, *total_channels*, *total_samples* und *epoch_time* in jeder ihrer Zeilen jeweilig denselben Wert haben. Somit können diese entfernt werden (Abbildung 13).

	trial	channel	sample_no	voltage	label	matching_condition
0	0	FP1	0	-8.921	co2a0000364.rd	S1 obj
1	0	FP1	1	-8.433	co2a0000364.rd	S1 obj
2	0	FP1	2	-2.574	co2a0000364.rd	S1 obj
3	0	FP1	3	5.239	co2a0000364.rd	S1 obj
4	0	FP1	4	11.587	co2a0000364.rd	S1 obj

Abbildung 13: Struktur der gefilterten Datentabelle [eigene Darstellung]

Für den späteren Output unserer Trainingsdaten wird der Patientenstatus anhand des vierten Zeichens des jeweiligen Dateinamens ausgelesen (a = Alkoholiker, c = Kontrollperson).
Da im späteren Verlauf auch ein gruppierter Datensatz erstellt werden soll, wird eine weitere Spalte mit der Patientennummer angelegt. Diese orientiert sich an die einzigartigen Werte des Labels.
Wie in Kapitel 3.4 festgelegt, sollen die Werte, die momentan als *strings* vorliegen in *int*-Werte umgewandelt werden. Dies betrifft die Kanal-, die Bildkonditions- sowie die Befundsspalte (Abbildung 14).

	trial	channel	sample_no	voltage	label	matching_condition	label_group	patient_number
0	0	0	0	-8.921	co2a0000364.rd	0	1	0
1	0	0	1	-8.433	co2a0000364.rd	0	1	0
2	0	0	2	-2.574	co2a0000364.rd	0	1	0
3	0	0	3	5.239	co2a0000364.rd	0	1	0
4	0	0	4	11.587	co2a0000364.rd	0	1	0

Abbildung 14: Struktur der gefilterten Datentabelle samt Patientenstatus und -nummer [eigene Darstellung]

Da das Label für das Training unbrauchbar ist, kann dieses aus dem Datensatz entfernt werden. Die Patienten- bzw. Versuchsnummer haben zwar keine Aussage über den Befund, werden aber später noch für die Gruppierung des Datensatzes gebraucht. Nach diesem Schritt werden sie aus dem endgültigen Datensatz entfernt. Somit stellt sich der vorläufige nicht-gruppierte Datensatz mit folgender Struktur heraus (Abbildung 15):

	patient_number	trial	matching_condition	channel	sample_no	voltage	label_group
0	0	0	0	0	0	-8.921	1
1	0	0	0	0	1	-8.433	1
2	0	0	0	0	2	-2.574	1
3	0	0	0	0	3	5.239	1
4	0	0	0	0	4	11.587	1

Abbildung 15: Struktur der gefilterten Datentabelle ohne das Label [eigene Darstellung]

Da der Schritt der Ausreißerbereinigung auf einen Datensatz viel zeitsparender anzuwenden wäre, als auf zwei, wird dieser Schritt auf dem nicht-gruppierten Datensatz durchgeführt. Der genaue Ablauf wird in Kapitel 4.2.3 beschrieben (Code 4.3).
Das einzige Feature, dass ohne die Korrelationsanalyse auf die Sinnhaftigkeit dessen Verwendung untersucht werden muss, ist die Bildkondition. Da sie labelkodiert ist, wird sie nach der Bereinigung durch die Generierung von Plots analysiert (Code 4.4).

Falls die Plots der einzelnen Bildkonditionen starke Unterschiede in den Spannungsverläufen aufweisen sollten, werden sie in die endgültigen Datensätze aufgenommen, da sie somit einen Einfluss auf die Spannunngswerte besitzen.

Hierbei werden die Plots nur mit der Alkoholikergruppe gezeichnet, da sie die festzustellende Gruppe darstellen. Durch die große Anzahl an einzelnen Messungen innerhalb der Versuche werden zufällige Komplettmessungen der jeweiligen Bildkonditionen geplottet und verglichen (Abbildung 16).

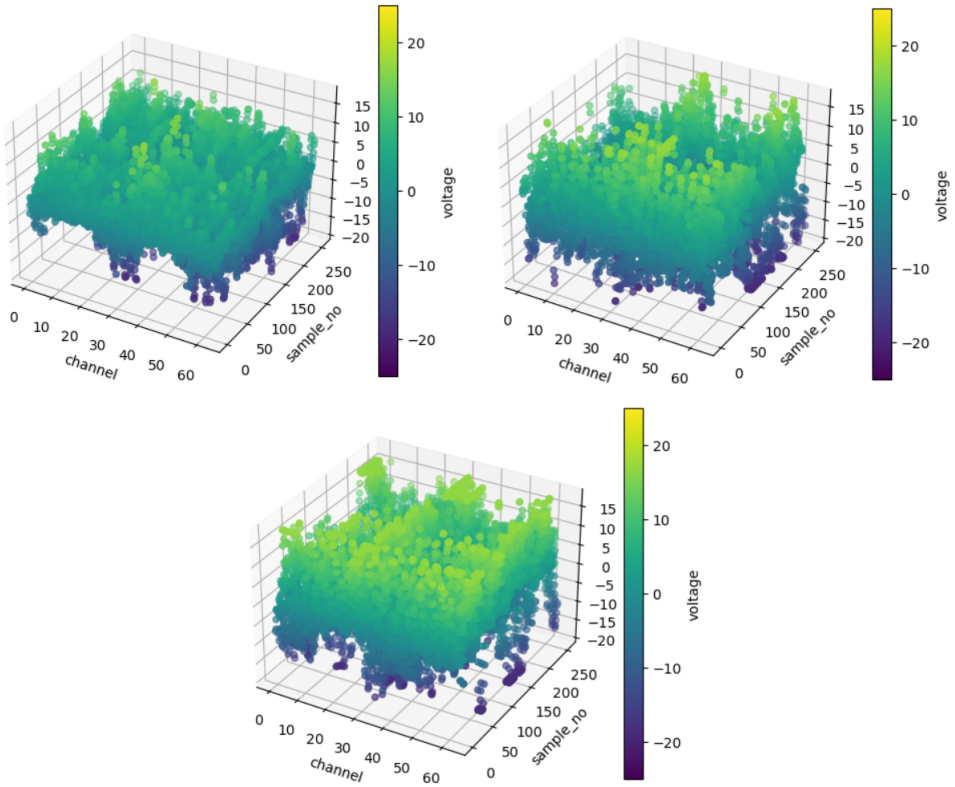


Abbildung 16: Spannungswerte zufälliger Alkoholiker mit Bildkondition S1obj (oben links), S2match (oben rechts) und S2nomatch (unten) [eigene Darstellung]

Wie zu erkennen ist, weisen die Spannungen teilweise sehr starke Unterschiede auf. In anderen Durchläufen ergaben die Plots jedoch andere Ergebnisse. Dies wurde auch bei mehreren Wiederholungen der zufälligen Plots bestätigt. Aus diesem Grund werden die Werte der einzelnen Konditionen zusätzlich mit der Pandas-Funktion *describe()* verglichen (Abbildung 17).

Wie zu sehen ist, besitzen die Durchschnittswerte teilweise große Unterschiede (-0,28uV bei S1obj, -1,85uV bei S2match und -1,48uV bei S2nomatch). Dies lässt darauf schließen, dass die Bildkondition einen Einfluss haben könnte, weshalb entschieden wurde, diese in den Datensätzen beizubehalten.

Als Letztes werden die Spannungswerte in den einzelnen Kanälen und Probenummern gruppiert, um den zweiten Datensatz zu generieren (Code 4.5). Durch diesen Schritt konnte der Speicherbedarf erheblich verringert werden (Abbildung 18).

count	5.680333e+07	count	2.872115e+07	count	2.842624e+07
mean	-2.824219e-01	mean	-1.845342e+00	mean	-1.478801e+00
std	7.715624e+00	std	7.692969e+00	std	7.720935e+00
min	-1.860500e+01	min	-1.860500e+01	min	-1.860500e+01
25%	-4.578000e+00	25%	-6.236000e+00	25%	-5.880000e+00
50%	-7.100000e-02	50%	-1.353000e+00	50%	-1.038000e+00
75%	4.191000e+00	75%	2.655000e+00	75%	3.011000e+00
max	1.679500e+01	max	1.679500e+01	max	1.679500e+01
Name: voltage, dtype: float64		Name: voltage, dtype: float64		Name: voltage, dtype: float64	

Abbildung 17: Statistikwerte von Alkoholikern mit Bildkondition S1obj (links), S2match (mittig) und S2nomatch (rechts) [eigene Darstellung]

Die unregelmäßigen Versuchsnummern sind durch die begrenzte Datenbereitstellung seitens der Quellseite zu erklären, was bereits in Kapitel 3.3.2 thematisiert wurde.

	patient_number	trial	matching_condition	label_group	voltage
0	0	0	0	1	[-8.921, -8.433, -2.574, 5.239, 11.587, 14.028...
1	0	2	0	1	[9.064, 6.622, 4.669, 3.693, 3.693, 3.693, 4.1...
2	0	7	2	1	[7.701, 4.771, 0.376, -3.042, -4.018, -3.042, ...
3	0	9	1	1	[-0.163, -0.163, 0.326, 1.302, 1.302, 0.814, -...
4	0	10	0	1	[5.28, 4.791, 4.303, 4.791, 5.28, 6.744, 7.721...

Abbildung 18: Struktur der gruppierten Datentabelle [eigene Darstellung]

Die Patienten- sowie die Versuchsnummer können nun entfernt werden. Damit garantiert werden kann, dass jede Zeile diesselbe Anzahl an Werten enthält und damit jeder einzelne Spannungswert als Eingangswert vom späteren Modell verarbeitet werden kann, werden zum Schluss die vorliegenden Arrays in einzelne Wertespalten aufgeteilt (Abbildung 19).

	matching_condition	label_group	0	1	2	3	4	5	6	7	...	16374	16375	16376	16377	16378	16379	16380
0	0	1	-8.921	-8.433	-2.574	5.239	11.587	14.028	11.587	6.704	...	16.795	16.795	11.454	4.618	3.153	6.571	12.431
1	0	1	9.064	6.622	4.669	3.693	3.693	3.693	4.181	4.181	...	16.795	13.153	6.805	3.387	3.387	4.364	4.852
2	2	1	7.701	4.771	0.376	-3.042	-4.018	-3.042	-0.600	1.353	...	-9.583	-7.141	-2.747	-0.793	-2.747	-7.141	-10.071
3	1	1	-0.163	-0.163	0.326	1.302	1.302	0.814	-1.628	-4.557	...	-14.394	-14.882	-15.859	-16.836	-18.605	-18.605	-18.605
4	0	1	5.280	4.791	4.303	4.791	5.280	6.744	7.721	8.698	...	-0.448	2.970	7.853	10.295	7.853	4.435	2.970

5 rows × 16386 columns

Abbildung 19: Struktur der aufgeteilten gruppierten Datentabelle [eigene Darstellung]

Da nun jede Zeile einen ganzen Messvorgang repräsentiert, ist es wichtig darauf zu achten, dass die Anzahl der Spannungsspalten 64 * 256 = *16.384* beträgt, was hier der Fall ist. Somit ist der gruppierte Datensatz fertig.

Beim nicht-gruppierten Datensatz müssen lediglich nur noch die Patienten- bzw. die Versuchsnummer entfernt werden, was die endgültige nicht-gruppierte Tabelle zur Folge hat (Abbildung 20).

	matching_condition	channel	sample_no	voltage	label_group
0	0	0	0	-8.921	1
1	0	0	1	-8.433	1
2	0	0	2	-2.574	1
3	0	0	3	5.239	1
4	0	0	4	11.587	1

Abbildung 20: Struktur der vorerst finalen nicht-gruppierten Datentabelle [eigene Darstellung]

4.2.3 Bereinigung von Ausreißern

Wie in Kapitel 3.3.5 erläutert, soll zwischen der Verwendung des Z-Scores und der des IQRs entschieden werden, indem der vorliegende Datensatz auf Verzerrung innerhalb der Spannungswerte überprüft wird (Code 4.3). Dies kann mit der Pandas-Funktion *skew()* untersucht werden (Abbildung 21):

```
Verzerrung bei der Alkoholikergruppe:  0.977
Verzerrung bei der Kontrollgruppe:  2.325
```

Abbildung 21: Die Verzerrungen beider Datensätze [eigene Darstellung]

Wie zu erkennen ist, besitzen die Werte eine teilweise starke Rechtsverzerrung. Deshalb sollen die Ausreißer mithilfe des IQRs ermittelt und gecappt werden. Zuvor soll kurz auf die Richtigkeit der Werte des Datensatzes eingegangen werden.

```
count    1.139507e+08
mean    -1.017154e+00
std      1.004433e+01
min     -5.008850e+02
25%     -5.330000e+00
50%     -6.310000e-01
75%      3.520000e+00
max      5.504050e+02
Name: voltage, dtype: float64
```

```
count    6.565069e+07
mean    -1.190689e+00
std      1.181749e+01
min     -7.001140e+02
25%     -6.582000e+00
50%     -8.340000e-01
75%      4.293000e+00
max      6.801250e+02
Name: voltage, dtype: float64
```

Abbildung 22: Die wichtigsten Statistikwerte des Alkoholiker- (links) sowie des Kontrollgruppendatensatzes (rechts) [eigene Darstellung]

Der Durchschnittswert, also der *mean*, ist bei den Alkoholikern höher, was aufgrund der in Kapitel 2.3.2 beschriebenen Zusammenhänge auch zu erwarten war (Abbildung 22). Damit kann durch diese kurze Analyse der Datensatz generell als valide angesehen werden. Weitere Analysen wären in diesem Fall nicht möglich, da die höhere Durchschnittsspannung der einzige für die Unterschiedsfindung relevante Wert ist, der sich in diesem Datensatz überprüfen ließe.

Durch die Perzentile, sowie die in Kapitel 3.3.5 dargestellte Formel, können nun die IQRs sowie Ober- bzw. Untergrenzen errechnet werden (Abbildung 23):

```
                                    Obergrenze bei der Alkoholikergruppe:  16.795
                                    Untergrenze bei der Alkoholikergruppe:  -18.605
IQR bei der Alkoholikergruppe:  8.85   Obergrenze bei der Kontrollgruppe:  20.606
IQR bei der Kontrollgruppe:  10.875    Untergrenze bei der Kontrollgruppe:  -22.894
```

Abbildung 23: IQRs und Ober-/Untergrenzen der beiden Klassen [eigene Darstellung]

Nun müssen die Ausreißerwerte gecappt werden. Hierbei werden die Werte, die die errechneten Grenzen über- bzw. unterschreiten, mit der Pandas-Funktion *loc()* ermittelt und dann mit den Grenzwerten überschrieben. Die folgende Grafik zeigt, dass die Ausreißerwerte beseitigt werden konnten und alle Werte sich in dem definierten Bereich befinden (Abbildung 24):

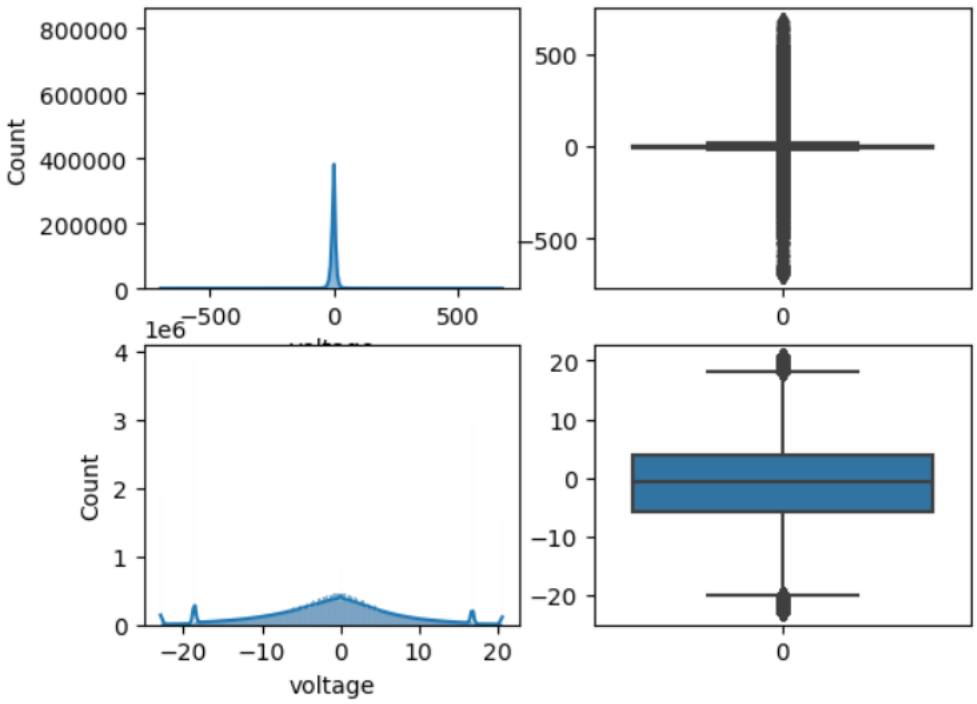


Abbildung 24: Aufteilung der Daten im Grenzbereich vor und nach der Bereinigung [eigene Darstellung]

Zum Schluss soll mit *describe()* nochmal überprüft werden, ob die wissenschaftliche Basis der Spannungswerte weiterhin erhalten geblieben ist (Abbildung 25).
Dies ist, den Durchschnittswerten nach zu urteilen, der Fall, sodass die Vorverarbeitung mit dem nächsten Schritt, der Korrelationsanalyse, fortgesetzt werden kann.

```
count    1.139507e+08          count    6.565069e+07
mean    -9.748038e-01          mean    -1.205355e+00
std      7.743169e+00          std      9.378608e+00
min     -1.860500e+01          min     -2.289400e+01
25%     -5.330000e+00          25%     -6.582000e+00
50%     -6.310000e-01          50%     -8.340000e-01
75%      3.520000e+00          75%      4.293000e+00
max      1.679500e+01          max      2.060600e+01
Name: voltage, dtype: float64  Name: voltage, dtype: float64
```

Abbildung 25: Spannungswerte zufälliger Alkoholiker mit Bildkondition S1obj (oben links), S2match (oben rechts) und S2nomatch (unten) [eigene Darstellung]

4.2.4 Korrelationsanalyse

Nun müssen die Features der beiden Datensätze auf Korrelation in Bezug auf die Spannungswerte überprüft werden. Beim nicht-gruppierten Datensatz stellt sich die Frage, ob zwischen dem Kanal bzw. der Probenummer und den dazugehörigen Spannungswerten eine korrelierende Beziehung besteht (Code 4.6.1). Sollte dies der Fall sein, würde ein hoher Korrelationswert in der Korrelationsmatrix zu sehen sein (Abbildung 26).

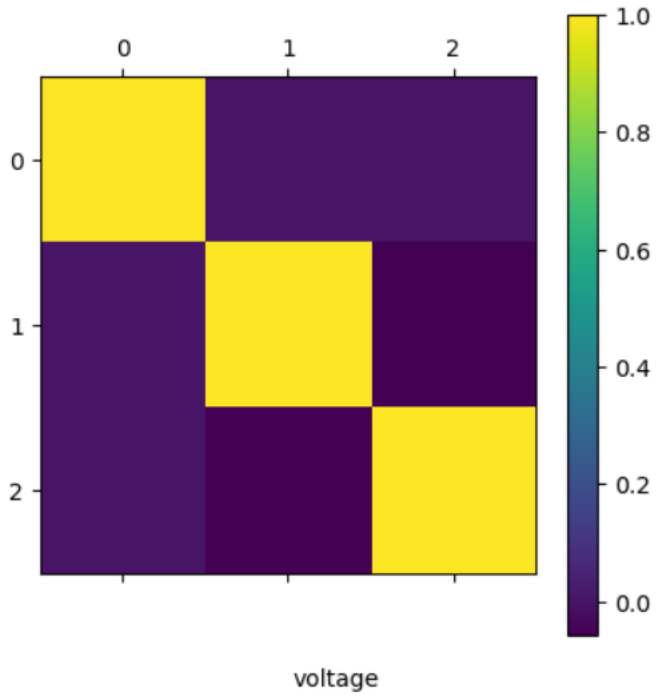
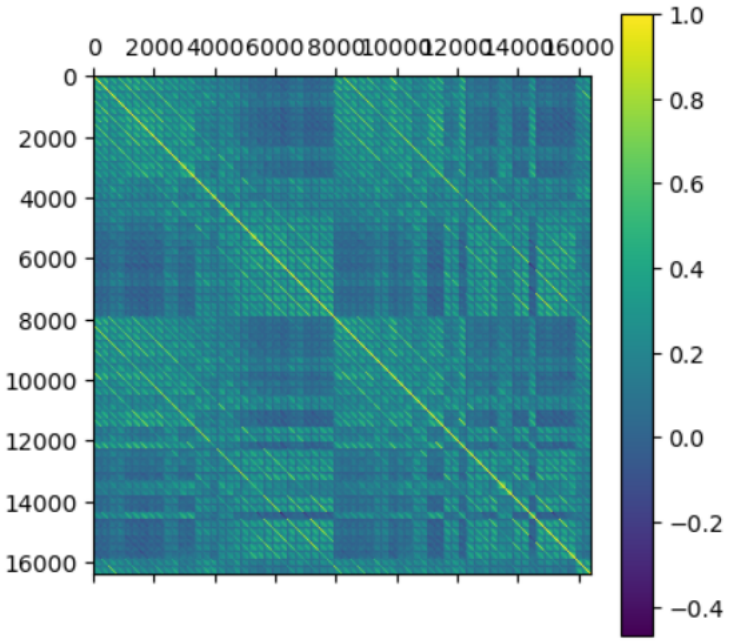


Abbildung 26: Visualisierte Korrelationsmatrix beim nicht-gruppierten Datensatz [eigene Darstellung]

Wie zu erkennen ist, weisen die Spalten untereinander keinerlei Korrelation auf, sodass der Datensatz nicht weiter verändert werden muss.

Beim gruppierten Datensatz werden die gemessenen Spannungswerte untereinander auf Korrelation überprüft (Code 4.6.2).

Hierbei wird, wie in Kapitel 3.3.6 entschieden, ein sehr hoher Korrelationskoeffizient von 0,975 gewählt, um keine wertvollen Informationen zu verlieren. Bei einem derartig hohen Koeffizienten wird diese Gefahr minimiert.




```
Korrelierende Spalten bei einem Koeffizienten von 1:  0
Korrelierende Spalten bei einem Koeffizienten von 0.975:  7909
```

Abbildung 27: Visualisierte Korrelationsmatrix beim gruppierten Datensatz [eigene Darstellung]

Insgesamt weisen 7.909 Spalten einen hohen Korrelationswert auf (Abbildung 27). Diese können nun aus dem Datensatz entfernt werden. Dadurch verbleiben 8.477 Spalten im endgültigen Datensatz. Somit ist der gruppierte Datensatz ebenfalls fürs Training aufbereitet.

4.3 Data Mining

Wie in Kapitel 3.5 bestimmt, soll ein binär klassifizierendes FNN anhand eines überwachten Lernverfahrens trainiert werden. Da hierbei zwei verschiedene Modelle mit zwei unterschiedlichen Datensätzen beschlossen wurden, werden diese auch, unabhängig voneinander beschrieben.

4.3.1 Genereller Trainingsaufbau

Als Erstes werden die Features in einem eigenen Dataframe *X* und die Ausgabedaten, die Befundsklassen, in einem Dataframe *y* gespeichert. Diese sollen nach Kapitel 3.5.3 zunächst in einer einfachen Rate von 80-10-10 aufgeteilt werden. Hierfür wird die *train_test_split()*- Methode von sklearn verwendet. Danach soll das Modell in einer eigens definierten Funktion gebaut werden.

Für jedes Optuna-Trial wird ein neues Modell aufgebaut und mit den gewählten Hyperparametern trainiert. Einige Werte, wie die Anzahl der Eingabeneuronen, leiten sich aus der Struktur des Datensatzes ab, bei einigen müssen individuelle Entscheidungen getroffen werden. Hierbei ist es wichtig einen Kompromiss zwischen dem Bestreben so viele Kombinationen wie möglich zu testen und der zeitlichen Komponente zu finden. Da hierzu keine definierten Normen existieren, wurde sich, unter dem Aspekt des Zusammenspiels beider Faktoren, für folgende Wertebereiche entschieden. Diese können, je nach Bedarf, während des Prozesses weiter angepasst werden:

Inputshape - gruppierter Datensatz: 8.476
Inputshape - nicht-gruppierter Datensatz: 4
Anzahl der Hiddenlayer (Optuna): 1-5
Anzahl der Units pro Hiddenlayer (Optuna): 32-256
Auswahl an Aktivierungsfunktionen (Optuna): ['relu', 'tanh', 'sigmoid']

Bei der Anzahl der Hidden Layer wurde sich für einen Bereich entschieden, bei dem die Möglichkeit besteht auch tiefe neuronale Netze zu testen. Ähnliches gilt auch für die Neuronenanzahl pro Hidden Layer, denn je mehr Neuronen vorliegen, desto genauer könnte theoretisch ein Modell trainiert werden, was aber wiederum auch ein erhöhtes Risiko für Overfitting lässt und einen großen zeitlichen Anspruch besitzen könnte. In Bezug auf die Aktivierungsfunktionen wurden insgesamt drei ausgewählt. *Sigmoid* wird häufig für Klassifikationsaufgaben verwendet [Gup]. *Tanh* ist ähnlich aufgebaut wie Sigmoid mit dem Unterschied, dass sie um 0 zentriert ist. *ReLU* ist eine beliebte Aktivierungsfunktion, die häufig als Allzwecklösung für verschiedene Probleme Anwendung findet [Gup].
Hierbei ist anzumerken, dass das Testen von größeren Wertebereichen und anderen Aktivierungsfunktionen valide Alternativen gewesen wären, in diesem Fall jedoch zwischen den zuvor erwähnten Seiten abgewägt werden musste.
Das Ausgangsneuron wird dann zum Schluss, wie in Kapitel 3.5.4.2 festgelegt, mit einer *Sigmoid*-Funktion versehen.
Nach der Bestimmung der dynamisch wählbaren Hyperparameter durch Optuna wird das Modell mit den in Kapitel 3.5 beschlossenen Konfigurationen aufgebaut und mithilfe des Trainingsdatensatzes trainiert. Auch bei der Entscheidungsfindung bzgl. der Epochenanzahl pro Trainingsdurchlauf sowie der Anzahl der gesamten Modelltrainings musste zwischen der Effektivität und der Zeit ein passender Mittelwert gefunden werden. Da es auch hier keine maßgeschneiderten Regeln gibt und die EarlyStopping-Funktion mit einer hohen Wahrscheinlichkeit häufig aufgerufen wird, wurde zunächst eine sichere Basis von 50 Epochen gewählt.

Damit viele verschiedene Kombinationen getestet werden können, wurde eine Trialanzahl von 200 bestimmt. Das bedeutet, dass pro Trainingsprozess 200 verschiedene Modelle trainiert werden, von welchen das mit den besten Accuracy-Werten, unter Berücksichtigung der Loss-Werte, als Endergebnis ausgewählt wird. Diese Werte könnten jedoch später erhöht werden, je nachdem, wie zeitintensiv das Training verläuft.
Das optimale Modell wird zum Schluss nochmal separat aufgebaut und anhand der in Kapitel 3.6 beschriebenen Analysetools ausgewertet und gespeichert. Anhand dieser Analyse werden dann die Maßnahmen für den nächsten Trainingsprozess getroffen. Der KDD-Schritt der *Interpretation und Evaluation* wird somit nach jedem Training durchgeführt.

4.3.2 Erster Trainingsdurchlauf

Der erste Versuch beim nicht-gruppierten Datensatz musste abgebrochen werden, da bei einer Datenmenge von ca. 180 Millionen Einzeldaten ein einziges Trainingstrial ca. 30 Stunden dauern würde, mit der Vermutung, dass dieses durch das EarlyStopping nach ungefähr 15 Epochen abgebrochen werden würde. Deshalb wurde der Datensatz zunächst bis zu dem Punkt gekürzt, an dem das Training eine akzeptable Zeit aufweisen würde (Code 4.7.1.0). Dies war bei ungefähr 0,5% des Gesamtdatensatzes, also bei 983.040 Messwerten, der Fall. Bei diesem Prozess wurde darauf geachtet, den Datensatz so ausgeglichen, wie möglich, zu gestalten, sodass sowohl die Verteilung auf die beiden Befundsklassen als auch die der Bildkonditionen in den einzelnen Klassen gleich war. Dadurch konnte das Risiko eines Overfittings zu Gunsten von einer der beiden Klassen stark vermindert werden. Außerdem wurde die Anzahl der Trials auf 50 heruntergesetzt, da bei 200 kein zufriedenstellendes zeitliches Ergebnis erreicht werden konnte.
Das gefundene optimale Modell stellte ein tiefes Netz mit drei Hidden Layern dar (Abbildung 28). Das Ergebnis brachte eine Accuracy von 60% und einen Alkoholiker-Recall von 63% bei einem Test-Loss von 0,64 (Code 4.7.1.1). Diese Werte sind auch sowohl in der Confusion-Matrix (Abbildung 29) als auch in den Trainingskurven (Abbildung 30) zu erkennen. Auch die Trainingszeit von ca. 12 Stunden spiegelt kein gutes Zeit-Leistungs-Verhältnis wider. Ein möglicher Grund könnte die geringe Aussagekraft von Einzelwerten zum Befund sein. Auch die enorme Reduzierung des Gesamtdatensatzes lässt eine starke Verschlechterung der Ergebnisse zu.

```
Best Hyperparameters: {'num_hidden_layers': 3, 'num_units_layer_0': 132, 'num_units_layer_1': 62, 'num_units_layer_2': 94, 'act
ivation_0': 'sigmoid', 'activation_1': 'sigmoid', 'activation_2': 'sigmoid'}
```

Abbildung 28: Nicht-Gruppiert: Modellarchitektur [eigene Darstellung]

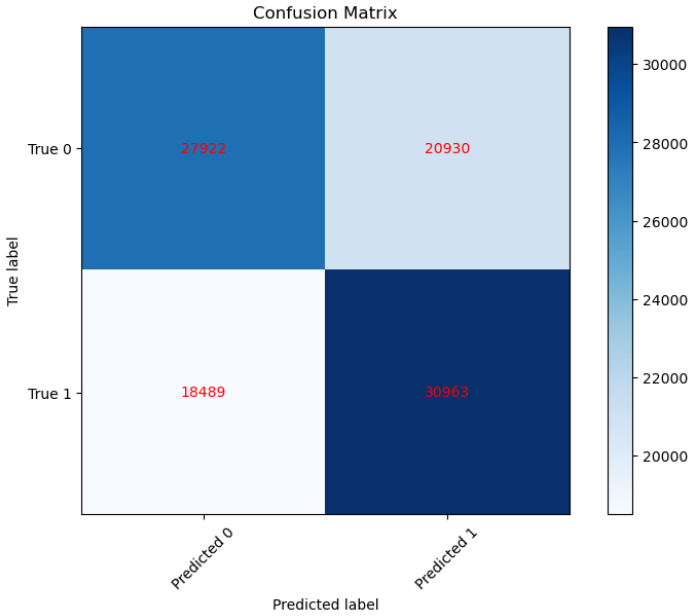


Abbildung 29: Nicht-Gruppiert: Confusion-Matrix [eigene Darstellung]

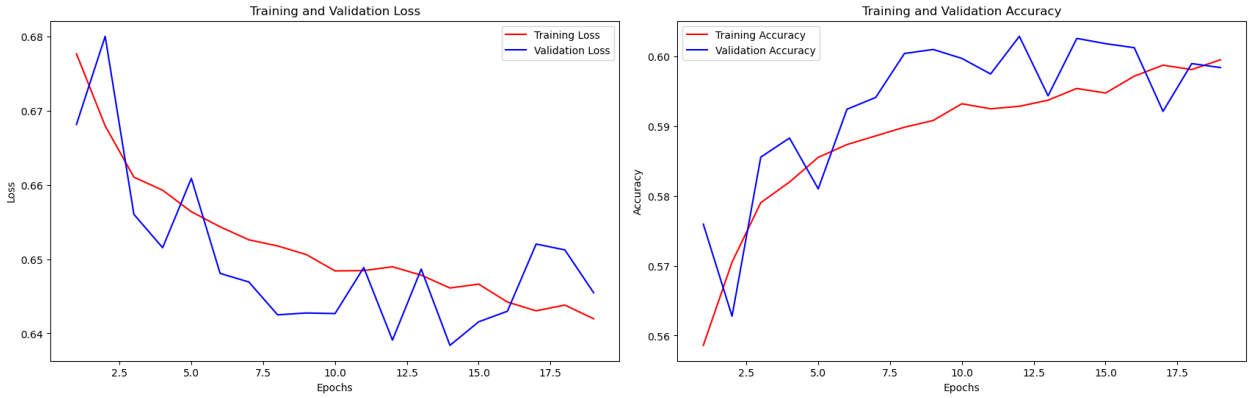


Abbildung 30: Nicht-Gruppiert: Loss (links), Accuracy (rechts) [eigene Darstellung]

Nun soll der gruppierte Datensatz trainiert und dessen Ergebnisse mit denen des nicht-gruppierten verglichen werden.

Nach 200 Trainingsversuchen wies beim gruppierten Datensatz das treffsicherste Modell eine Accuracy von 78% und einen Alkoholiker-Recall von 81% auf mit einem Test-Loss von 0,77, welches somit viel besser abschneidet als das des nicht-gruppierten Datensatzes (Code 4.7.2.1). Hierbei beinhaltete das Netzwerk nur eine einzige Hidden Layer (Abbildung 31). Auch die Trainingszeit war viel kürzer. Obwohl dies in der Confusion Matrix nicht deutlich sichtbar ist (Abbildung 32), besteht ein starker Unterschied zwischen den Entwicklungen der Trainings- und Validation-Accuracy, sowie des Losses (Abbildung 33), was auf ein Überanpassungsproblem hindeutet.

```
Best Hyperparameters: {'num_hidden_layers': 1, 'num_units_layer_0': 52, 'activation_0': 'relu'}
```

Abbildung 31: Gruppiert: Modellarchitektur [eigene Darstellung]

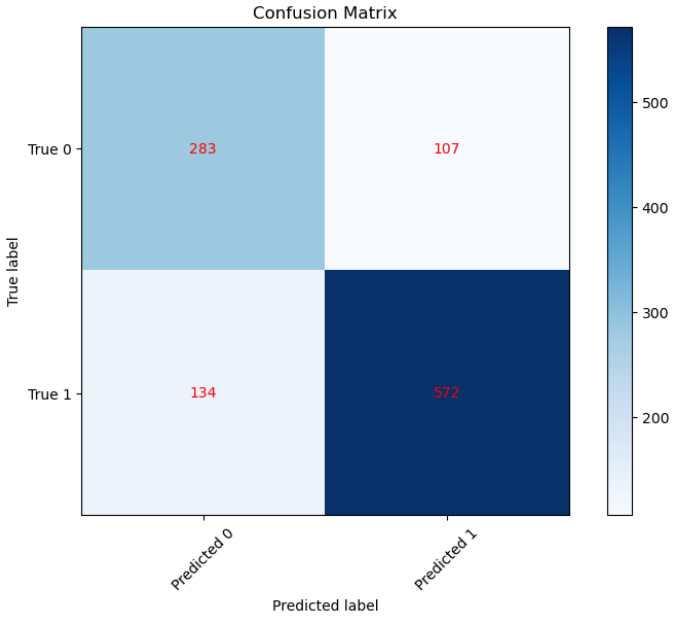


Abbildung 32: Gruppiert: Confusion-Matrix [eigene Darstellung]

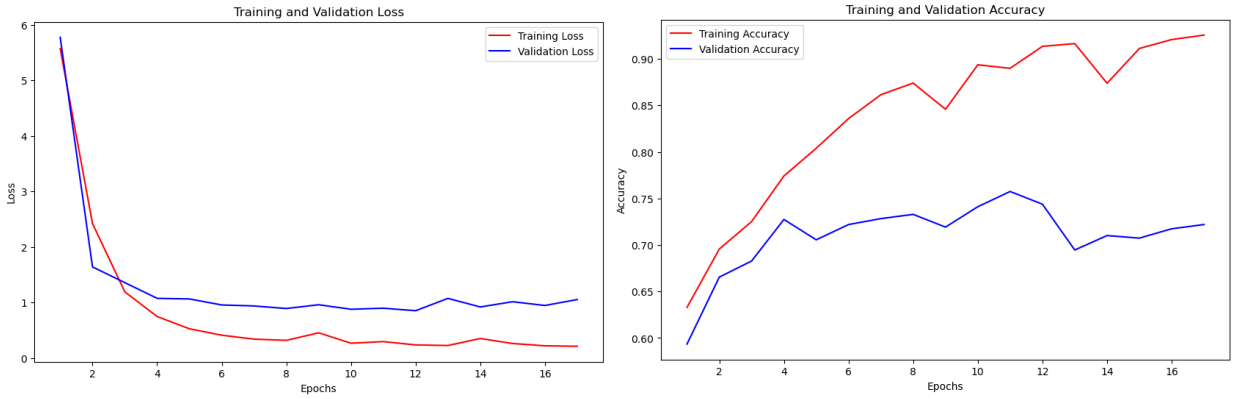


Abbildung 33: Gruppiert: Loss (links), Accuracy (rechts) [eigene Darstellung]

Dies kann u.a. an der unausgeglichenen Datenverteilung zwischen Alkoholikern und Kontrollpersonen liegen, weshalb im nächsten Schritt sowohl Over- als auch Undersampling getestet werden sollen. Hierbei bietet es sich an, beides zu testen, da die Trainingszeit trotz eines möglichen Anstiegs durch das Oversampling sich noch in einem guten Rahmen befinden würde. Beide Verfahren könnten dazu beitragen, das Ungleichgewicht in der Datenverteilung zu adressieren und die Modellleistung zu verbessern. Aus den in Kapitel 3.3.4 geschilderten Gründen wurde sich aufgrund der viel schlechteren Ergebnisse dafür entschieden den nicht-gruppierten Datensatz nicht weiter anzupassen und ab jetzt nur noch mit dem gruppierten Datensatz zu arbeiten.

4.3.3 Over- und Undersampling

Mit dem *RandomOverSampler* bzw. dem *RandomUnderSampler* der *imblearn*-Python-Bibliothek können diese Verfahren einfach eingebaut werden. Mit *fit_resample()* werden die Eingabe- und Ausgabedaten an die gewünschten Verhältnisse angepasst.

Beim Oversampling erwies sich ein Modell mit drei Hidden Layer als am treffsichersten (Code 4.7.2.2; Abbildung 34). Es wurde eine geringe Verbesserung der Accuracy auf 79% sowie eine bessere Verteilung in der Confusion Matrix (Abbildung 35) bei einem Test-Loss von 0,47 und ein ähnlicherer Verlauf der Loss- bzw. Accuracy-Funktionen festgestellt (Abbildung 36). Der Recall verschlechterte sich jedoch auf 74%.

```
Best Hyperparameters: {'num_hidden_layers': 5, 'num_units_layer_0': 116, 'num_units_layer_1': 98, 'num_units_layer_2': 155, 'nu
m_units_layer_3': 187, 'num_units_layer_4': 187, 'activation_0': 'relu', 'activation_1': 'relu', 'activation_2': 'relu', 'activ
ation_3': 'relu', 'activation_4': 'relu'}
```

Abbildung 34: Gruppiert, Oversampling: Modellarchitektur [eigene Darstellung]

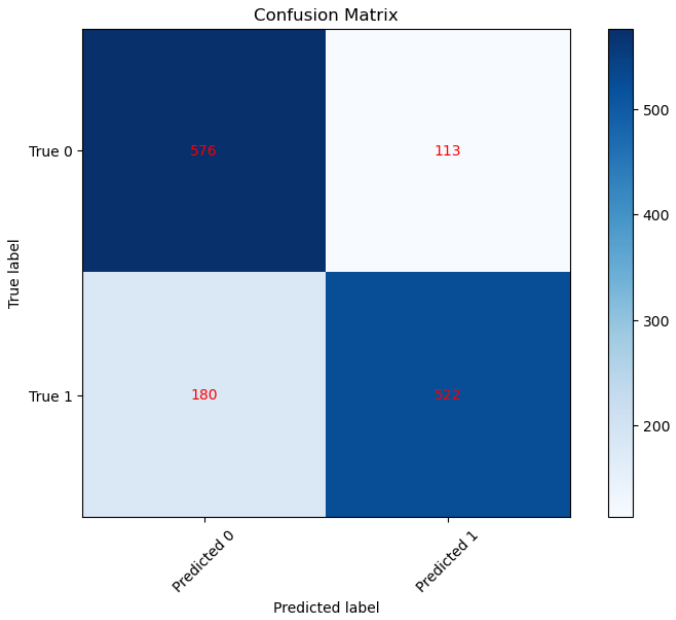


Abbildung 35: Gruppiert, Oversampling: Confusion-Matrix [eigene Darstellung]

Beim Undersampling hatten sich alle Werte bei einer Architektur von einem Hidden Layer (Code 4.7.2.3; Abbildung 37) verschlechtert. Die Accuracy lag bei 71%, der Alkoholiker-Recall bei 69% und der Test-Loss bei 1,43 (Abbildung 38). Dies könnte an einem aus der Datenreduktion folgenden starken Informationsverlust liegen.

Da das Oversampling-Modell fast in allen Werten eine Verbesserung nach sich zog und durch das bessere Handeln von Overfitting näher an der wahren Accuracy liegt, wurde sich dafür entschieden, für die weiteren Anpassungen das overgesamplete Modell zu verwenden.

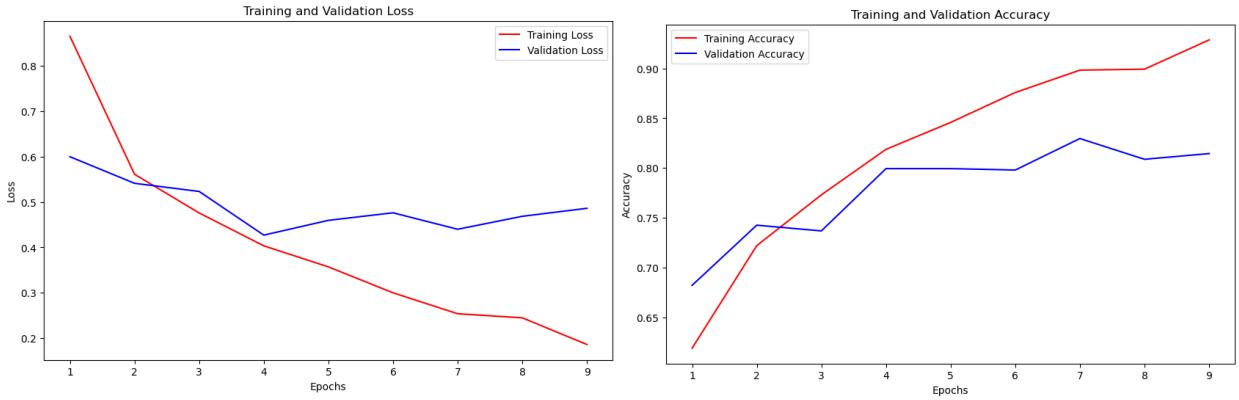


Abbildung 36: Gruppiert, Oversampling: Loss (links), Accuracy (rechts) [eigene Darstellung]

```
Best Hyperparameters: {'num_hidden_layers': 1, 'num_units_layer_0': 225, 'activation_0': 'relu'}
```

Abbildung 37: Gruppiert, Undersampling: Modellarchitektur [eigene Darstellung]

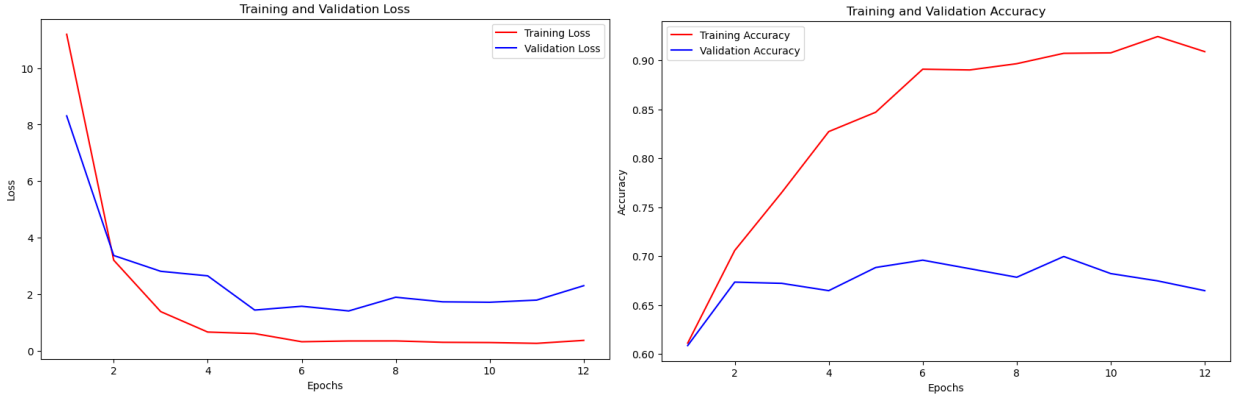


Abbildung 38: Gruppiert, Undersampling: Loss (links), Accuracy (rechts) [eigene Darstellung]

Um einem möglichen Overfitting weiter gegensteuern zu können und eine bessere Annäherung zwischen Trainings- und Validationloss zu erreichen, soll eine Dropout-Layer mit einer Rate von 0.5 nach den Hidden Layer und vor dem Output Layer eingebaut werden (Code 4.7.2.4).

4.3.4 Dropout-Layer

Hierbei ist beim Oversampling-Satz mit einer Architektur von einem Hidden Layer (Abbildung 39) die Accuracy minimal auf 78% gesunken, der Recall ist gleich geblieben und der Test-Loss etwas auf 0,49 gestiegen. Die Verläufe des Trainigsprozesses bzw. die Loss-Werte waren jetzt jedoch fast gleich (Abbildung 40), was dafür spricht, dass das Overfitting minimiert werden konnte. Somit soll die Dropout-Layer weiterhin verwendet werden, um das Risiko eines Overfittings nicht ansteigen zu lassen.
Nun sollen weitere Parameter, sowohl einzeln als auch, abhängig von den Ergebnissen, in Kombinationen angepasst werden, um die nicht optimale Accuracy zu behandeln.

```
Best Hyperparameters: {'num_hidden_layers': 1, 'num_units_layer_0': 36, 'activation_0': 'relu'}
```

Abbildung 39: Gruppiert, Oversampling und Dropout-Layer: Modellarchitektur [eigene Darstellung]

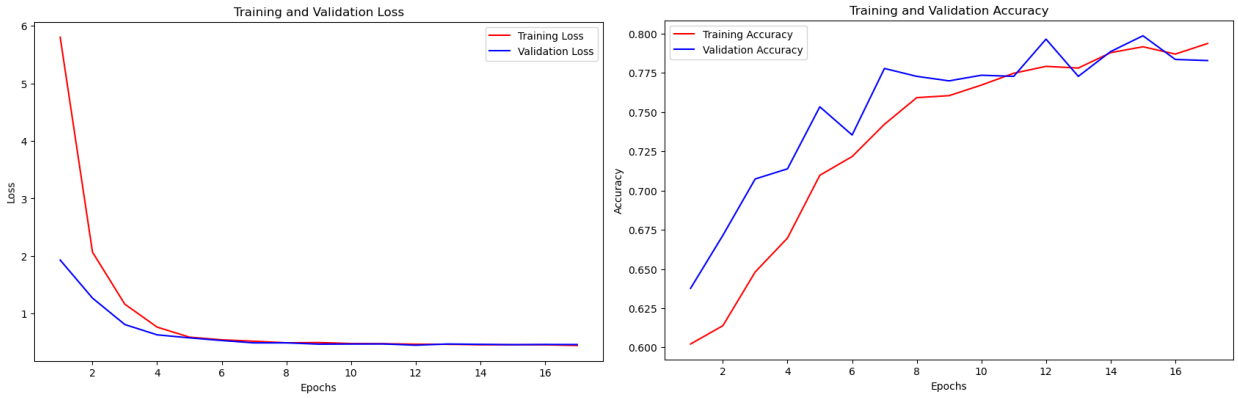


Abbildung 40: Gruppiert, Oversampling und Dropout-Layer: Loss (links), Accuracy (rechts) [eigene Darstellung]

Zunächst wurde beschlossen, die Batch-Size auf 64 bzw. 128 und damit den Informationsgehalt innerhalb eines Batches zu erhöhen, um zu prüfen, ob sich das Modell dadurch verbessern würde.

4.3.5 Erhöhung der Batch-Size

Bei einer Batch-Size von 64 und der effektivsten Architektur mit einem Hidden Layer (Code 4.7.2.5; Abbildung 41) hat sich die Accuracy auf 81% verbessert, der Recall ist dagegen um 1% gesunken. Der Test-Loss war mit 0,48 etwas höher als vorher. Hier konnte auch bemerkt werden, dass eine etwas größere Differenz zwischen Training und Validierung aufgetreten ist (Abbildung 42).

```
Best Hyperparameters: {'num_hidden_layers': 1, 'num_units_layer_0': 74, 'activation_0': 'relu'}
```

Abbildung 41: Gruppiert, Oversampling, Dropout-Layer und Batch-Size von 64: Modellarchitektur [eigene Darstellung]

Die beste gefundene Architektur bei einer Batch-Size von 128 bestand ebenfalls aus nur einem Hidden Layer (Code 4.7.2.6; Abbildung 43). Die Accuracy verbesserte sich nochmals, diesmal auf 83%, der Recall auf ganze 79%. Jedoch entwickelte sich wieder eine größere Differenz in der Entwicklung des Losses bzw. der Accuracy (Abbildung 44) und der Test-Loss stieg auf 0,55 an. Um einen besseren Überblick über den weiteren Progress zu bekommen, wurde auch eine Batch-Size von 256 ausprobiert.

Abermals mit einer Architektur mit einem Hidden Layer (Abbildung 45) blieb die Accuracy gleich, jedoch verschlechterte sich der Recall auf 77% und der Test-Loss auf 0,70 (Code 4.7.2.7). Auch die Differenz zwischen dem Trainings- und Validation-Loss hat sich stark erhöht (Abbildung 46).

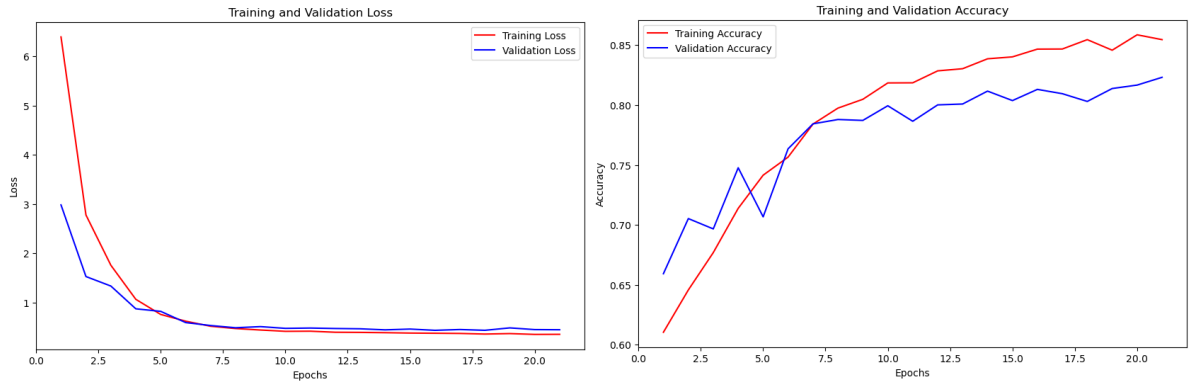


Abbildung 42: Gruppiert, Oversampling, Dropout-Layer und Batch-Size von 64: Loss (links), Accuracy (rechts) [eigene Darstellung]

```
Best Hyperparameters: {'num_hidden_layers': 1, 'num_units_layer_0': 151, 'activation_0': 'relu'}.
```

Abbildung 43: Gruppiert, Oversampling, Dropout-Layer und Batch-Size von 128: Modellarchitektur [eigene Darstellung]

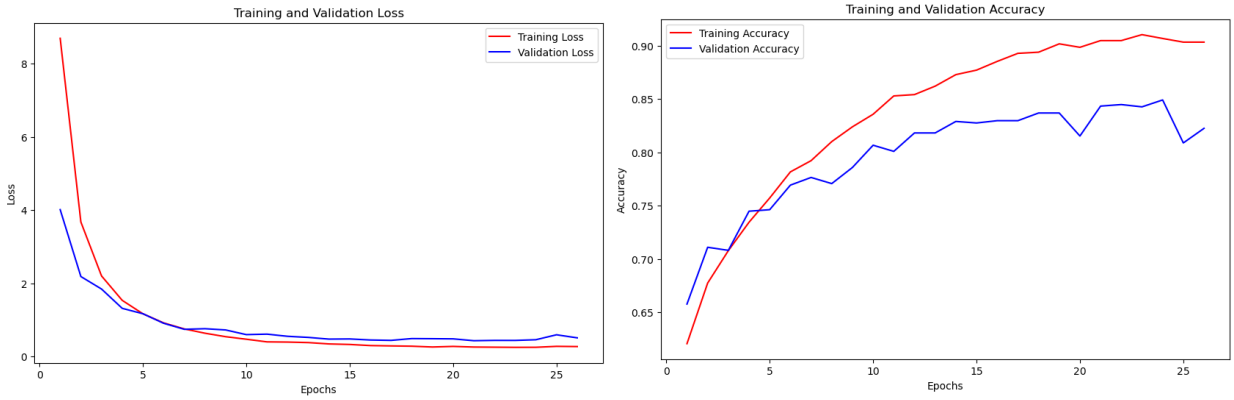


Abbildung 44: Gruppiert, Oversampling, Dropout-Layer und Batch-Size von 128: Loss (links), Accuracy (rechts) [eigene Darstellung]

```
Best Hyperparameters: {'num_hidden_layers': 1, 'num_units_layer_0': 166, 'activation_0': 'relu'}
```

Abbildung 45: Gruppiert, Oversampling, Dropout-Layer und Batch-Size von 256: Modellarchitektur [eigene Darstellung]

Zu diesem Zeitpunkt wird ein mögliches Overfitting akzeptiert, da dieses später mithilfe von Cross Validation und ggf. den L1-/L2-Regularisierungen gelöst werden könnte. Das momentane Hauptziel besteht darin, die Accuracy bzw. den Recall zu verbessern. Somit wurde sich zunächst, aufgrund der besseren erzielten Werte, für eine Batch-Size von 128 entschieden.

Eine weitere Möglichkeit den Informationsgehalt zu erhöhen wäre die Datensatzaufteilung anzupassen. Hierbei könnte eine Veränderung der Aufteilungsrate auf 90-5-5 Verbesserungen nach sich ziehen.

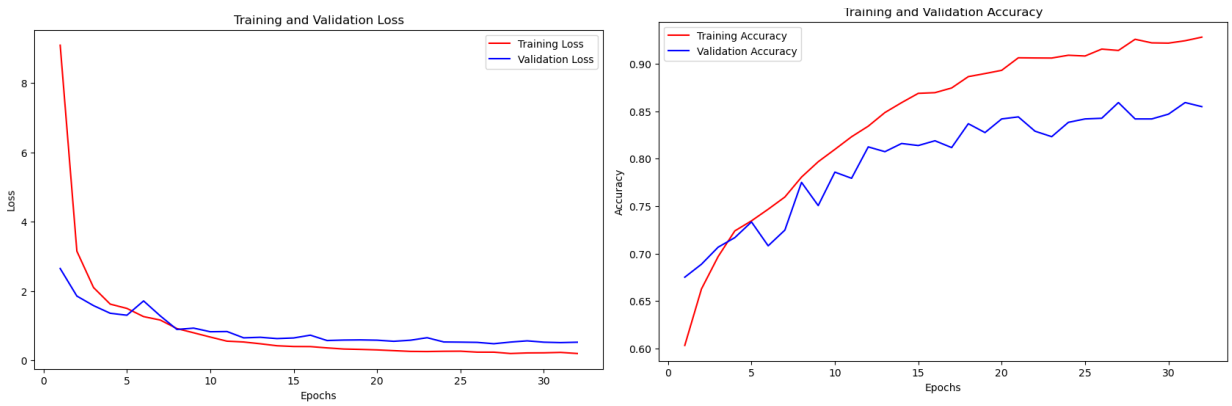


Abbildung 46: Gruppiert, Oversampling, Dropout-Layer und Batch-Size von 256: Loss (links), Accuracy (rechts) [eigene Darstellung]

4.3.6 Veränderung der Datensatzaufteilung

Die Vergrößerung des Trainingssatzes bei einer Architektur mit einem einzigen Hidden Layer (Abbildung 47) brachte wenig Verbesserung weder bei der Accuracy, noch beim Recall (Code 4.7.2.8), jedoch bei dem Test-Loss mit einem Wert von 0,43. Der Recall hat sich sogar um 2% verschlechtert. Auch die Loss-Differenz ist größer geworden (Abbildung 48), weshalb im weiteren Verlauf mit der 80-10-10-Aufteilung weitergearbeitet wird.

```
Best Hyperparameters: {'num_hidden_layers': 1, 'num_units_layer_0': 110, 'activation_0': 'relu'}
```

Abbildung 47: Gruppiert, Oversampling, Dropout-Layer, Batch-Size von 128 und Datenaufteilungsrate von 90-5-5 : Modellarchitektur [eigene Darstellung]

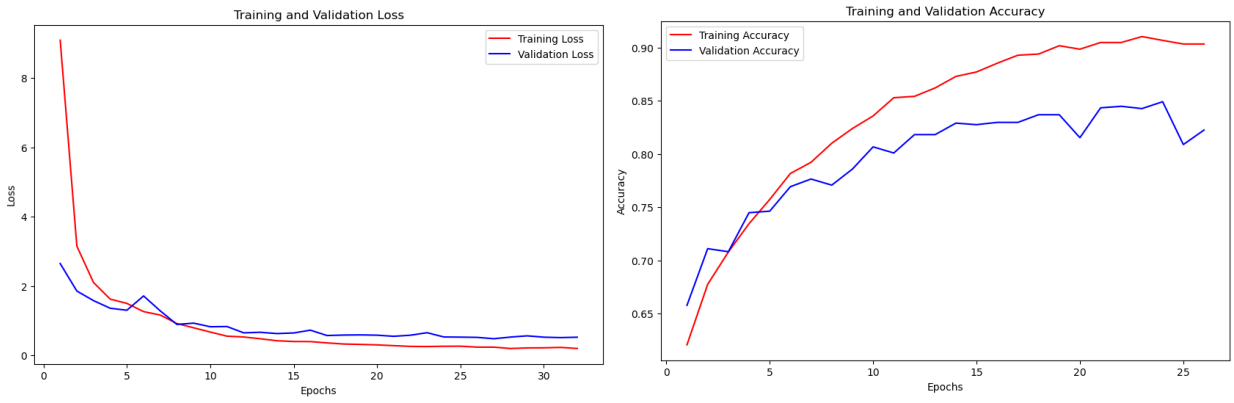


Abbildung 48: Gruppiert, Oversampling, Dropout-Layer, Batch-Size von 128 und Datenaufteilungsrate von 90-5-5 : Accuracy (rechts), Loss (links) [eigene Darstellung]

Da diese Änderung keine Verbesserungen gebracht hat, wurde überlegt, den Datensatz mit dem Korrelationskoeffizienten von 1 zu verwenden.

Da die Spannungswerte in einem sehr begrenzten Bereich liegen und schon kleine Unterschiede große Auswirkungen auf den Befund haben könnten, ist nicht auszuschließen, dass ein Korrelationskoeffizient von 0,975 schon ausreicht, um wichtige Informationen zu entfernen. Auch im Hinblick auf die große Anzahl an bei diesem Schritt entfernten Spalten, besteht diese Möglichkeit. Aus diesem Grund wurde der vor der Korrelationsanalyse vorliegende Datensatz verwendet, da bei einem Korrelationskoeffizienten von 1 keine Spalten entfernt werden müssen [Abbildung 27].

4.3.7 Veränderung des Korrelationskoeffizienten

Durch die Änderung des Korrelationskoeffizienten konnte mit einer Hidden Layer (Abbildung 49) eine weitere Verbesserung der Accuracy auf 84% sowie des Recalls auf 80% festgestellt werden (Code 4.7.2.9). Der Test-Loss stieg auf 0,65 an. Die Kurven haben sich jedoch besser angeglichen (Abbildung 50). Deshalb soll der Datensatz mit dem Korrelationskoeffizienten von 1 beibehalten werden.

```
Best Hyperparameters: {'num_hidden_layers': 1, 'num_units_layer_0': 170, 'activation_0': 'relu'}
```

Abbildung 49: Gruppiert, Oversampling, Dropout-Layer, Batch-Size von 128 und Korrelationskoeffizient von 1: Modellarchitektur [eigene Darstellung]

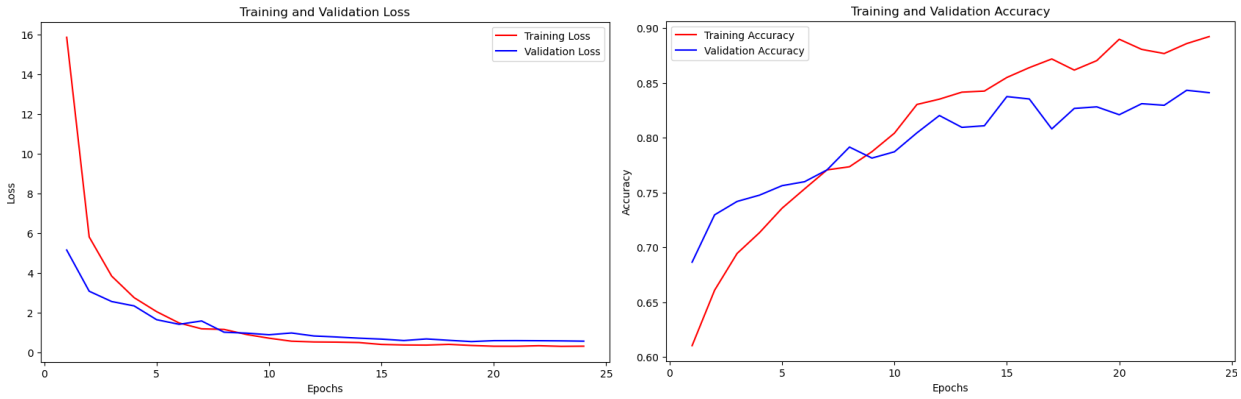


Abbildung 50: Gruppiert, Oversampling, Dropout-Layer, Batch-Size von 128 und Korrelationskoeffizient von 1: Accuracy (rechts), Loss (links) [eigene Darstellung]

Eine weitere Möglichkeit die Werte zu verbessern, wäre denselben Trainingsprozess ohne EarlyStopping durchzuführen. Einerseits würde sich das Risiko des Overfittings erhöhen, jedoch könnte andererseits das Modell besser trainiert und vielleicht auch eine noch effektivere Layerarchitektur gefunden werden. Dies soll im nächsten Schritt ausprobiert werden. Da es theoretisch passieren konnte, dass ein Modelltraining durch das EarlyStopping zu früh gestoppt wurde und sich im späteren Verlauf noch verbessert hätte, wird der komplette Prozess nochmal mit Optuna durchlaufen.

4.3.8 Entfernen des EarlyStoppings

Die Epochenanzahl von 50 wurde erstmal beibehalten, um zu prüfen, wie sich der Loss bzw. die Accuracy rein ohne EarlyStopping entwickelt (Code 4.7.2.10).
Das Training mit einer ähnlichen Architektur wie zuvor (Abbildung 51) und ohne EarlyStopping hat keine deutlichen Verbesserungen gebracht. Die Accuracy verschlechterte sich um 1% und der Recall verbesserte sich um 1%. Die Kurvenverläufe waren hierbei ebenfalls dem vorherigen Versuch sehr ähnlich, nur der Test-Loss sank wenig auf 0,55 (Abbildung 52).

```
Best Hyperparameters: {'num_hidden_layers': 1, 'num_units_layer_0': 71, 'activation_0': 'relu'}
```

Abbildung 51: Gruppiert, Oversampling, Dropout-Layer, Batch-Size von 128, Korrelationskoeffizient von 1 und ohne EarlyStopping: Modellarchitektur [eigene Darstellung]

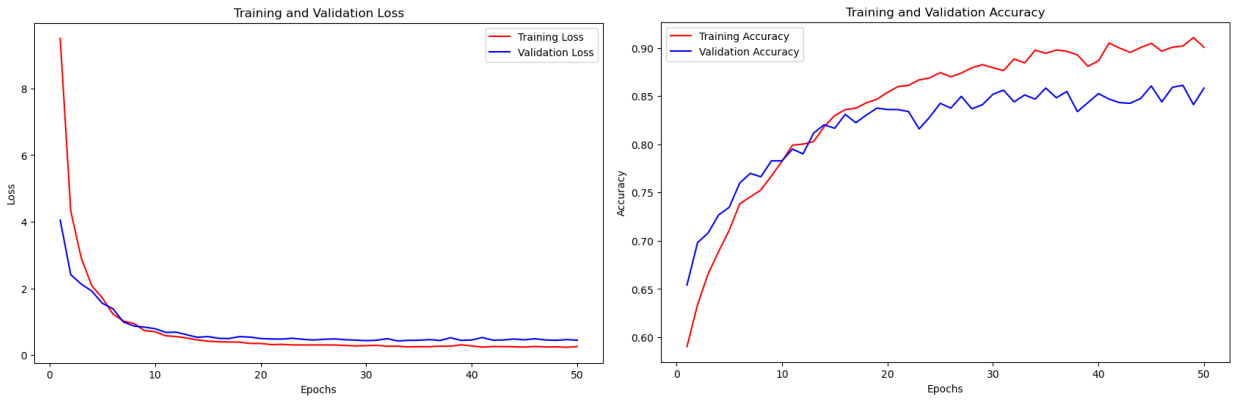


Abbildung 52: Gruppiert, Oversampling, Dropout-Layer, Batch-Size von 128, Korrelationskoeffizient von 1 und ohne EarlyStopping: Accuracy (rechts), Loss (links) [eigene Darstellung]

Um zu überprüfen, ob weiteres Training die Performanz verändert, soll der Prozess nochmal mit einer Epochenanzahl von 100 durchlaufen werden (Code 4.7.2.11).
Hierbei konnte mit einer einzigen Hidden Layer (Abbildung 53) eine starke Verbesserung der Accuracy auf 87% festgestellt werden. Auch der Recall ist auf 82% gestiegen. Der Test-Loss verschlechterte sich jedoch auf 0,60. Die Kurvenverläufe sind dabei ungefähr gleich geblieben (Abbildung 54).

```
Best Hyperparameters: {'num_hidden_layers': 1, 'num_units_layer_0': 52, 'activation_0': 'relu'}
```

Abbildung 53: Gruppiert, Oversampling, Dropout-Layer, Batch-Size von 128, Korrelationskoeffizient von 1 und Epochenanzahl von 100 ohne EarlyStopping: Modellarchitektur [eigene Darstellung]

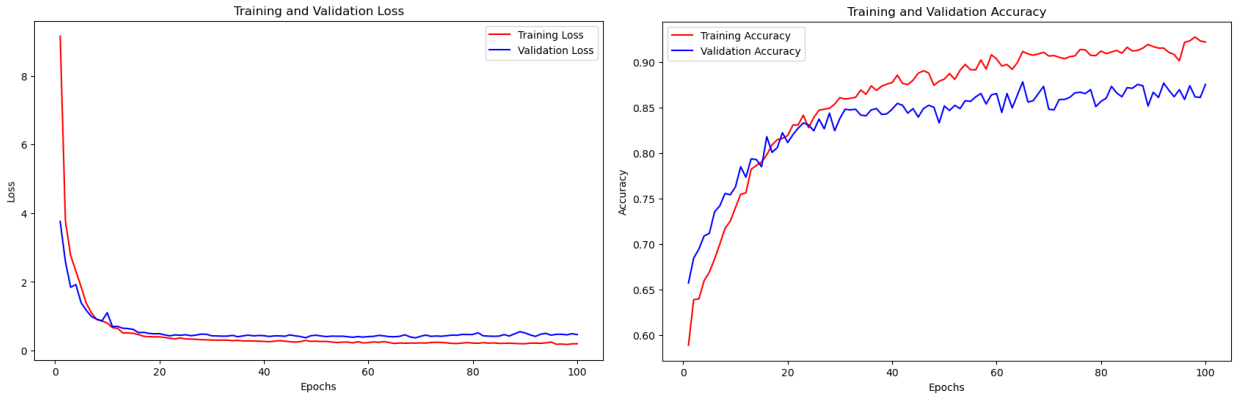


Abbildung 54: Gruppiert, Oversampling, Dropout-Layer, Batch-Size von 128, Korrelationskoeffizient von 1 und Epochenanzahl von 100 ohne EarlyStopping: Accuracy (rechts), Loss (links) [eigene Darstellung]

Nun soll zum Schluss, wie in Kapitel 3.5.3 festgelegt, stratified k-fold cross validation bei der performantesten Architektur- und Hyperparameterkombination angewendet werden.

4.3.9 Cross-Validation

Da das Entfernen des EarlyStoppings zuvor keine Verbesserungen brachte, soll es nun wieder integriert werden.

Für die Cross Validation sollte die Datenaufteilung so aufgebaut werden, dass die 80-10-10-Aufteilung beibehalten wird. Hierfür werden zunächst 10% des Gesamtdatensatzes für das Testset beiseite gelegt. Die übrigen 90% werden für die Cross validation verwendet. Bei einem k-Wert von 9 wird dieser Anteil so gesplittet, dass im Endeffekt die 80-10-10-Aufteilung erhalten bleibt. Zusätzlich wird die *stratify*-Variable bei der Aufteilung des Datensatzes gesetzt, um sicherzustellen, dass die Verteilung der beiden Klassen in den Teildatensätzen ausgeglichen bleibt. Da auch hier die zeitliche Komponente eine große Rolle spielt, wird das Modell pro Fold 10 Epochen lang trainiert, sodass pro Optuna-Trial 90 Epochen Training ohne EarlyStopping eingeplant werden (Code 4.7.2.12).

Hierbei wurde bei fünf Hidden Layern (Abbildung 55) ein sehr starker Unterschied zwischen der Trainings- bzw. Validation- und der Testperformance festgestellt. Die Trainings- und Validation-Accuracies befanden sich im hohen 90-er-Bereich (Abbildung 56), bei dem Testsatz wurde jedoch eine Accuracy von 88% und ein Recall von 80% festgestellt, was nur eine teilweise Werteverbesserung darstellt. Der Test-Loss ist jedoch auf 0,52 gesunken.

```
Best Hyperparameters: {'num_hidden_layers': 5, 'num_units_layer_0': 193, 'num_units_layer_1': 157, 'num_units_layer_2': 213, 'n
um_units_layer_3': 143, 'num_units_layer_4': 158, 'activation_0': 'relu', 'activation_1': 'relu', 'activation_2': 'relu', 'acti
vation_3': 'relu', 'activation_4': 'relu'}
```

Abbildung 55: Gruppiert, Oversampling, Dropout-Layer, Batch-Size von 128, Korrelationskoeffizient von 1, ohne EarlyStopping und mit Cross Validation: Modellarchitektur [eigene Darstellung]

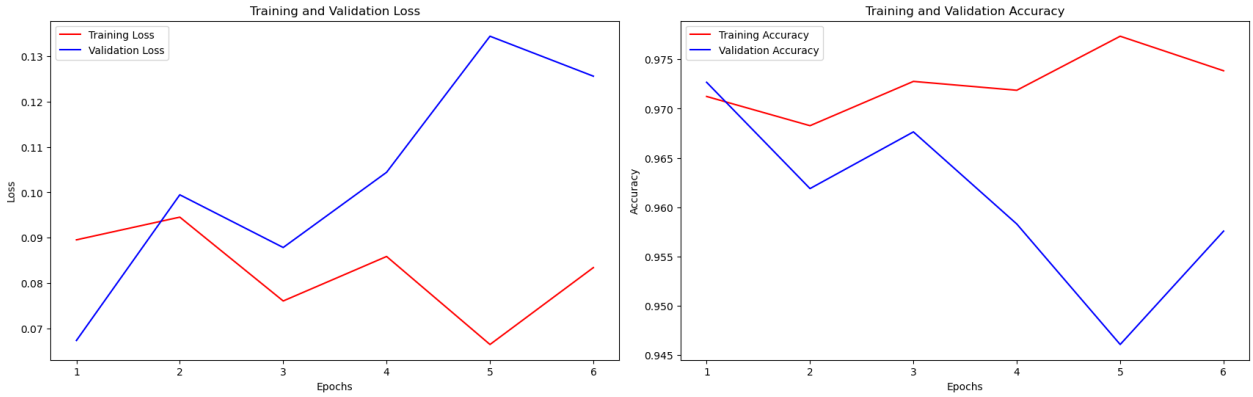


Abbildung 56: Gruppiert, Oversampling, Dropout-Layer, Batch-Size von 128, Korrelationskoeffizient von 1, ohne EarlyStopping und mit Cross Validation: Accuracy (rechts), Loss (links) [eigene Darstellung]

Da der Recall der Kontrollgruppe 96% beträgt und Overfitting somit wahrscheinlich erscheint, soll nun versucht werden, mithilfe der L1- bzw. L2-Regularisierung gegenzusteuern. Die Werte für L1 und L2 sollen ebenfalls von Optuna bestimmt werden. Hierfür wurde eine große Range von $1e^{-6}$ bis $1e^{-2}$ gewählt.

4.3.10 L1-/L2-Regularisierung

Bei der L1-Regularisierung mit drei Hidden Layern (Abbildung 57) hat sich der Test-Loss mit 0,6 etwas verschlechtert. Die Accuracy bzw. der Recall konnten jedoch keine besseren Ergebnisse erzielen (Abbildung 58). Mit 87% bzw. 80% schneidet dieses Modell etwas schlechter als das bislang beste Modell ab (Code 4.7.2.13).

```
Best Hyperparameters: {'num_hidden_layers': 3, 'num_units_layer_0': 179, 'num_units_layer_1': 249, 'num_units_layer_2': 216, 'a
ctivation_0': 'relu', 'activation_1': 'relu', 'activation_2': 'relu', 'l1_regularizer': 1.2121969008883125e-06}
```

Abbildung 57: Gruppiert, Oversampling, Dropout-Layer, Batch-Size von 128, Korrelationskoeffizient von 1, Cross Validation und L1-Regularisierung: Modellarchitektur [eigene Darstellung]

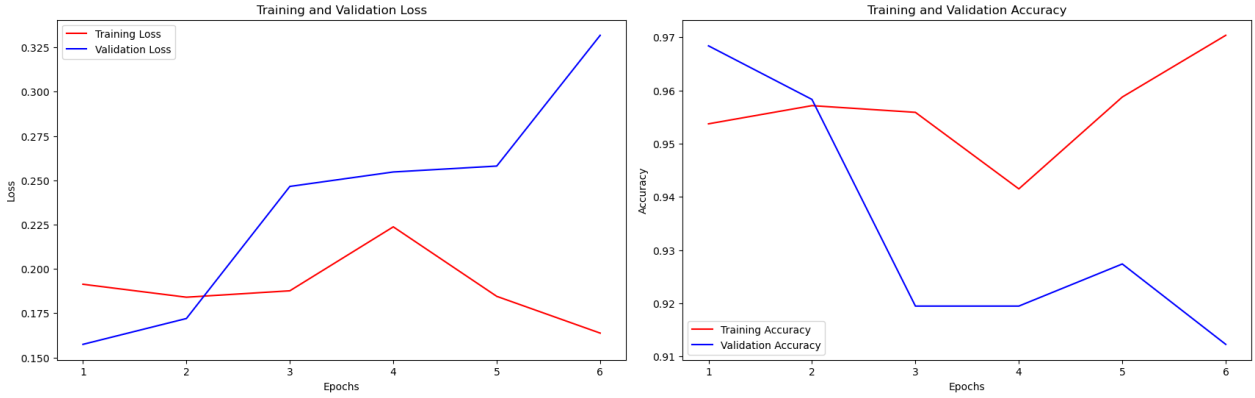


Abbildung 58: Gruppiert, Oversampling, Dropout-Layer, Batch-Size von 128, Korrelationskoeffizient von 1, Cross Validation und L1-Regularisierung: Accuracy (rechts), Loss (links) [eigene Darstellung]

Mit der L2-Regularisierung und einem Hidden Layer (Abbildung 59) konnte der bislang beste Recall-Wert von 85% erreicht werden. Die Accuracy sank zwar minimal auf 86%, der Test-Loss erreichte jedoch mit 0,42 ebenfalls seinen derzeitigen Top-Wert mit. Die Kurvenverläufe erreichen zwar nicht die Werte, die bei der L1-Regularisierung betrachtet werden konnten, in der Gesamtrelation wird dieses Modell jedoch, vor allem durch den stark verbesserten Recall, als das bislang effektivste bewertet (Abbildung 60; Code 4.7.2.14).

```
Best Hyperparameters: {'num_hidden_layers': 1, 'num_units_layer_0': 100, 'activation_0': 'relu', 'l2_regularizer': 7.5273106638
51669e-06}
```

Abbildung 59: Gruppiert, Oversampling, Dropout-Layer, Batch-Size von 128, Korrelationskoeffizient von 1, Cross Validation und L2-Regularisierung: Modellarchitektur [eigene Darstellung]

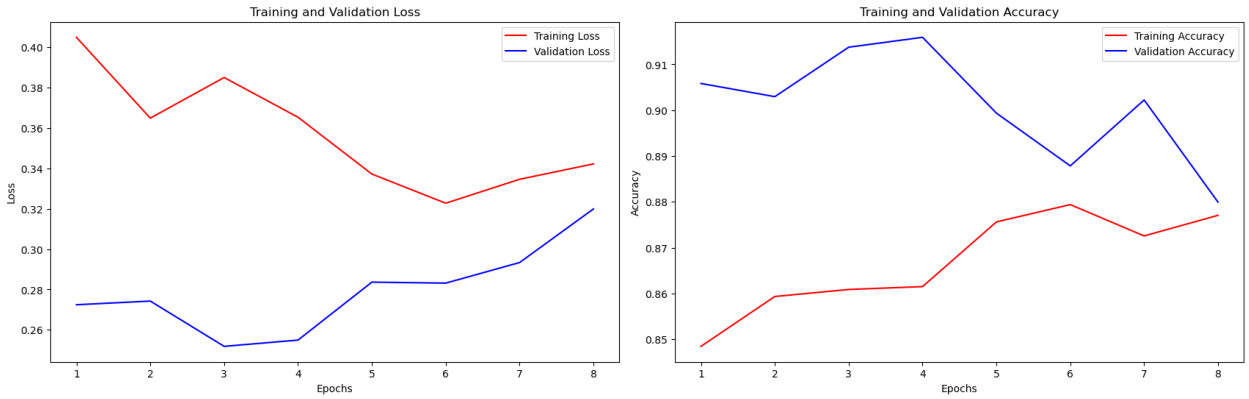


Abbildung 60: Gruppiert, Oversampling, Dropout-Layer, Batch-Size von 128, Korrelationskoeffizient von 1, Cross Validation und L2-Regularisierung: Accuracy (rechts), Loss (links) [eigene Darstellung]

Bei der Kombination von L1 und L2, also einem Elastic-Net, konnten keine Verbesserungen bei fünf Hidden Layer (Abbildung 61) festgestellt werden (Code 4.7.2.15). Die Accuracy lag bei 83%, der Recall bei 78% und der Test-Loss verschlechterte sich auf 0,60. Die Accuracy- und Loss-Verläufe waren jetzt ebenfalls unterschiedlicher (Abbildung 62).

```
Best Hyperparameters: {'num_hidden_layers': 5, 'num_units_layer_0': 32, 'num_units_layer_1': 134, 'num_units_layer_2': 78, 'num
_units_layer_3': 186, 'num_units_layer_4': 248, 'activation_0': 'relu', 'activation_1': 'sigmoid', 'activation_2': 'tanh', 'act
ivation_3': 'tanh', 'activation_4': 'relu', 'l1_regularizer': 0.00019090666989625123, 'l2_regularizer': 0.004295029943608783}
```

Abbildung 61: Gruppiert, Oversampling, Dropout-Layer, Batch-Size von 128, Korrelationskoeffizient von 1, Cross Validation und Elastic-Net-Regularisierung: Modellarchitektur [eigene Darstellung]

Somit ist nach diesem Schritt das Modell mit der einzelnen L2-Regularisierung das bislang effektivste.

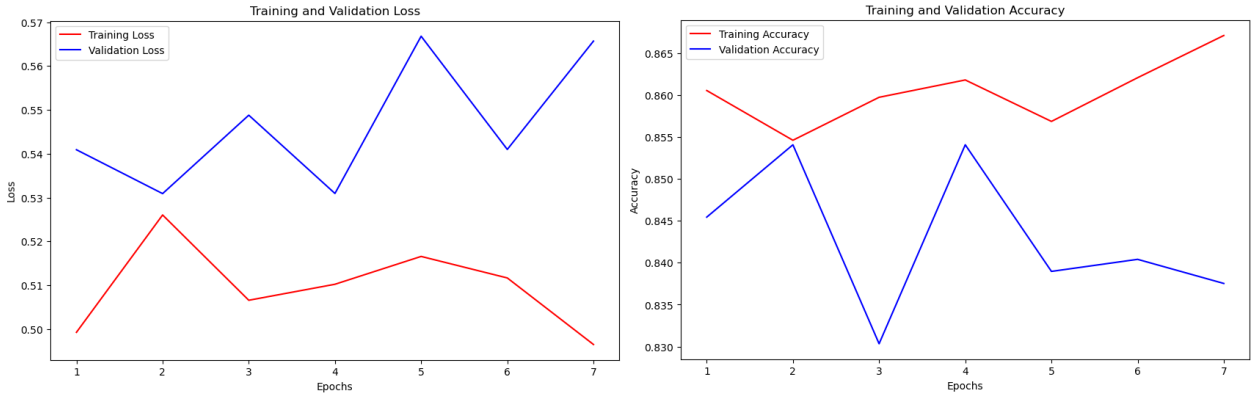


Abbildung 62: Gruppiert, Oversampling, Dropout-Layer, Batch-Size von 128, Korrelationskoeffizient von 1, Cross Validation und Elastic-Net-Regularisierung: Accuracy (rechts), Loss (links) [eigene Darstellung]

4.3.11 Weitere mögliche Anpassungen

Nicht alle Anpassungen, die im Konzept Erwähnung fanden, wurden auch umgesetzt. Dies lag zumeist daran, dass diese in dem gegebenen Kontext keine großen Verbesserungen mit sich bringen würden.
So hätte eine angepasste Behandlung der fehlerhaften/leeren Dateien auch nur, wenn überhaupt, minimale Auswirkungen auf die Accuracy, da deren Anteil, wie in Kapitel 4.2.1 gesehen, weniger als 1% aller Daten beträgt.
Auch das Hinzufügen entfernter Features ist nicht möglich, da die Features, die im Prozess der Datenbereinigung entfernt wurden, rein wissenschaftlich, keine Rolle im Zusammenhang mit dem Befund spielen und somit nicht relevant sind. Selbiges gilt für das Entfernen der Features, da dies weiterer Informationsverlust bedeuten würde.
Schon bei dem verwendeten sehr hohen Koeffizienten von 0,975 wurde etwas weniger als die Hälfte der Spalten entfernt. Eine weitere Reduktion könnte zu einem Informationsverlust führen, da für spezifische Zusammenhänge, wie bei den Spannungswerten, wo es um Änderungen im uV-Bereich geht, nur sehr große Koeffizienten verwendet werden sollten. Wären bei diesem Koeffizienten wenige bis gar keine Spalten entfernt worden, wäre diese Option eine Überlegung wert gewesen.
Da der Bereich der Spannungswerte mithilfe des IQRs stark reduziert wurde, würde auch eine Batch-Normalization-Layer keine genaueren Ergebnisse liefern. Bei Eingaben, die bereits in einem kleinen Bereich liegen, würde sie keine zusätzlichen Vorteile bringen, da ihre Hauptaufgabe darin besteht, die Aktivierungen pro Batch zu normalisieren.
Da somit die meisten Hyperparameter bereits angepasst wurden und weitere Anpassungen nur wenig Ertrag zeigen würden, wurde beschlossen, das Training an diesem Punkt zu beenden.

4.4 Zusammenfassung

4.4.1 Trainingsergebnisse

Schon nach den ersten Trainingsversuchen ergab sich eine Treffwahrscheinlichkeit von 78%. Hierbei war es wichtig, dass das System Alkoholiker mit einer hohen Genauigkeit bestimmen konnte, weshalb der Recall in den Vordergrund gesetzt wurde. Weitere Anpassungen der Vorgänge und Hyperparameter, wie z.B. Oversampling und das Hinzufügen einer Dropout-Layer verminderten das anfangs aufgetretene Overfitting stark. Insbesondere die Erhöhung der Batch-Size auf 128 sowie die L2-Regularisierung haben zu einer starken Verbesserung der Accuracy sowie des Recalls beigetragen. Das effektivste Modell weist einen Accuracy-Wert von 86% auf. Der entsprechende Alkoholiker-Recall beträgt 85%.

4.4.2 Beantwortung der Forschungsfragen

Verglichen mit anderen Arbeiten, die andere Architekturen wie z.B. CNNs verwenden und dabei eine Accuracy teilweise im hohen 90-er%-Bereich erreicht haben, schneidet das hier trainierte Modell schlechter ab. Dadurch, dass viele Hyperparamter-Anpassungen ausprobiert wurden, wäre ein möglicher Grund, dass die Vorverarbeitung des Datensatzes zu keinem optimalen Ergebnis geführt hat. In den erwähnten Arbeiten wurden die Werte teilweise in 2D-Grafiken umgewandelt, was vielleicht für diesen Fall eine bessere Methode wäre. Auch das FNN an sich könnte theoretisch hinterfragt werden. Jedoch ist hier zu erwähnen, dass das trainierte Modell die Ergebnisse der erwähnten Dokumentationen (s. Kapitel 3.7) teilweise übertroffen hat, was somit für eine Verwendung von Optimierungsfunktionen, wie Optuna, zum Auffinden der geeignetsten Architektur spricht. Es kann nicht ausgeschlossen werden, dass die Testung von mehr Kombinationen mit Optuna das Erkennen von besseren Architekturen nach sich ziehen würde. Insgesamt kann ein FNN für die Lösung dieses Problems als geeignet angesehen werden, mit dem Zusatz, dass durch ein ausgedehnteres Training dieser oder ähnlicher Architekturen bessere Ergebnisse erzielt werden könnten.

Die beste Kombination aus den getesteten Architekturen und Hyperparametern weist eine Accuracy von 86% und einen Alkoholiker-Recall von 85% auf (Abbildung 63). Hierbei wurde der gruppierte Datensatz mit Oversampling vergrößert, mithilfe einer Dropout-Layer und einer L2-Regularisierung dem Overfitting gegengesteuert, die Batch-Size auf 128 vergrößert, der Korrelationskoeffizient auf 1 erhöht und das Modell mit Cross Validation trainiert. Optuna ermittelte dabei eine Hidden-Layer-Architektur von einem Layer mit 100 Units, der relu-Aktivierungsfunktion sowie einem L2-Wert von $7,53e^{-6}$ als optimal.

Modellanpassungen	Accuracy	Alkoholiker-Recall	Test-Loss
Nicht-gruppierter Datensatz	60%	63%	0,64
Gruppierter Datensatz	78%	81%	0,77
Gruppierter Datensatz mit Oversampling	79%	74%	0,47
Gruppierter Datensatz mit Undersampling	71%	69%	1,43
Gruppierter Datensatz mit Oversampling und Dropout-Layer	78%	74%	0,49
Gruppierter Datensatz mit Oversampling, Dropout-Layer und Batch-Size von 64	81%	73%	0,48
Gruppierter Datensatz mit Oversampling, Dropout-Layer und Batch-Size von 128	83%	79%	0,55
Gruppierter Datensatz mit Oversampling, Dropout-Layer und Batch-Size von 256	83%	77%	0,7
Gruppierter Datensatz mit Oversampling, Dropout-Layer, Batch-Size von 128 und einer Datensatzaufteilung von 90-5-5	83%	75%	0,43
Gruppierter Datensatz mit Oversampling, Dropout-Layer, Batch-Size von 128 und Korrelationskoeffizienten von 1	84%	80%	0,65
Gruppierter Datensatz mit Oversampling, Dropout-Layer, Batch-Size von 128, Korrelationskoeffizienten von 1 und ohne EarlyStopping	83%	81%	0,55
Gruppierter Datensatz mit Oversampling, Dropout-Layer, Batch-Size von 128, Korrelationskoeffizienten von 1, ohne EarlyStopping, bei einer Epochenanzahl von 100	87%	82%	0,6
Gruppierter Datensatz mit Oversampling, Dropout-Layer, Batch-Size von 128, Korrelationskoeffizienten von 1 und Cross Validation	88%	80%	0,51
Gruppierter Datensatz mit Oversampling, Dropout-Layer Batch-Size von 128, Korrelationskoeffizienten von 1, Cross Validation und L1-Regularisierung	87%	80%	0,6
Gruppierter Datensatz mit Oversampling, Dropout-Layer, Batch-Size von 128, Korrelationskoeffizienten von 1, Cross Validation und L2-Regularisierung	86%	85%	0,42
Gruppierter Datensatz mit Oversampling, Dropout-Layer, Batch-Size von 128, Korrelationskoeffizienten von 1, Cross Validation und Elastic-Net	83%	78%	0,6

Abbildung 63: Resultate der einzelnen Trainingsprozesse [eigene Darstellung]

4.4.3 Ausblick für weitere Arbeiten

Ein genauer Grund für die nicht optimale Treffwahrscheinlichkeit kann nur gemutmaßt werden. Durch ein längeres Training mit Optuna oder durch die Verwendung anderer Optimierungsmechanismen kann nicht ausgeschlossen werden, dass hierbei bessere Ergebnisse erzielt werden könnten. Obwohl die Datenvorverarbeitung nach außen hin stimmig und auch optimiert scheint, kann ebenfalls der Fall bestehen, dass durch andere Techniken, wie z.B. der Principal Component Analysis (PCA) während der Feature Extraction, Daten effizienter und theoretisch mit weniger Informationsverlust bereinigt werden könnten.

Da FNNs mit diesem Datensatz noch nicht ausgiebig trainiert wurden, wäre eine weitere Option, dass andere Architekturen bzw. -erweiterungen wie z.B. CNNs und/oder der Einbau von LSTM besser für diesen speziellen Fall geeignet sind.

5 Fazit

Um das definierte Ziel, die Entwicklung eines Algorithmus zur erfolgreichen Feststellung einer Prädisposition zum Alkoholismus, zu erreichen, wurden zunächst die wichtigsten Begrifflichkeiten und Grundlagen im Bereich der KI, des Alkoholismus sowie des EEG thematisiert.
Darauf aufbauend wurde im 3. Kapitel ein Konzept zur Datenvorverarbeitung sowie zum Training des Modells entworfen. Hierbei wurden Entscheidungen für bzw. gegen die Verwendung von einzelnen Faktoren/Architekturen getroffen, die sich spezifisch auf den behandelten Fall bezogen. Letztendlich wurde sich für die Entwicklung eines binär klassifizierenden FNNs anhand eines überwachten Lernverfahrens entschieden. Mithilfe der Optimierungsfunktion Optuna sollten für die Anzahl der Hidden Layer, die Anzahl derer Neuronen sowie für deren Aktivierungsfunktionen die optimalen Parameter gefunden werden. Hierbei wurden für jede manuelle Anpassung anderer Hyperparamter 200 Optuna-Trials mit jeweils 50 Epochen durchgeführt.
In der Umsetzung des entwickelten Konzeptes entstanden zwei voneinander unabhängige Datensätze, die sich darin unterschieden, dass einer von ihnen den Kanal und die Probenummer der Messungen als einzelne Features berücksichtigte und der andere wiederum nicht, wobei die Spannungswerte beim ersten als Einzelwerte in einer Spalte, beim zweiten gruppiert als einzelne Spalten gespeichert wurden. Im weiteren Verlauf wurden fehlerhafte/leere Dateien durch das Dropping und Ausreißer mithilfe des IQRs entfernt. Mithilfe der Korrelationsanalyse konnten im gruppierten Datensatz redundante Spalten festgestellt und somit ebenfalls aus dem Datensatz gelöscht werden.
Während des ersten Trainingdurchgangs wurde festgestellt, dass der nicht-gruppierte Datensatz, wegen einer sehr großen Trainingszeit, stark gekürzt werden musste. Mit einem Datensatzanteil von 0,5% des Gesamtsatzes schnitt es viel schwächer und zeitintensiver ab als das Modell mit dem gruppierten Datensatz. Aus diesem Grund wurde der nicht-gruppierte Datensatz verworfen.
Mithilfe von Oversampling sowie dem Einfügen einer Dropout-Layer konnte das Overfitting, das beim ursprünglichen Modell beobachtet werden konnte, minimiert werden. Das Testen von Undersampling brachte hierbei viel schlechtere Werte als beim Oversampling. Durch die Erhöhung der Batch-Size auf 128 konnte die Accuracy sowie der Recall der Alkoholikergruppe weiter gesteigert werden. Andere Batch-Größen von 64 und 256 schnitten schlechter ab. Eine Änderung der Datensatzverteilung auf 90-5-5 führte ebenfalls zu schlechteren Ergebnissen. Daraufhin wurde der Korrelationskoeffizient auf 1 erhöht, sodass keine Spalten entfernt wurden. Dies führte zu einer Verbesserung sowohl der Accuracy als auch des Recalls, erklärbar durch einen anscheinend hohen Informationsverlust selbst bei einem Korrelationskoeffizienten von 0,975.

Ab diesem Zeitpunkt wurden die meisten implementierbaren und sinnhaften Anpassungen der Architektur bzw. der Hyperparamteter getestet, sodass es nun darum ging, die Werte mithilfe von einem ausgedehnteren Training zu verbessern. Hierbei wurde zunächst das EarlyStopping entfernt, was keine großen Verbesserungen mit sich brachte. Da dies höchstwahrscheinlich durch die zu geringe Epochenanzahl bedingt war, wurde diese auf 100 erhöht. Dies führte zu der bis dahin besten Werteverteilung von 87% Accuracy sowie 82% Recall. Eine Implementation von Cross Validation führte beim Testsatz nur zu einer geringen Verbesserung, konnte jedoch durch die Implementation einer L2-Regularisierung stark verbessert werden. Der Einbau von L1-Regularisierung und Elastic Net brachten hingegen keine Verbesserungen. Das effektivste Modell erreichte einen Accuracy-Wert von 86% und einen entsprechenden Alkoholiker-Recall von 85%.

Insgesamt dient das performanteste gefundene Modell einer durchaus genauen Vorhersage einer Veranlagung zum Entwickeln einer Alkoholsucht, ist jedoch für eine praxisnahe alleinstehende Verwendung im medizinischen Bereich ungeeignet. Da das Modell die Ergebnisse der erwähnten Dokumentationen, die ebenfalls FNNs verwendet haben, teilweise übertrifft, kann davon ausgegangen werden, dass die Verwendung von Optuna oder auch anderen Optimierungsfunktionen eine effektive sowie zeitsparende Suche der besten Hyperparamter-Kombinationen bietet.

Im Vergleich zu Arbeiten, die z.B. CNNs verwenden, sind die hier erzielten Ergebnisse jedoch schlechter. Dies könnte an den unterschiedlichen Ansätzen der Datenvorverarbeitung sowie der Datensatzerstellung liegen, da in diesen gezeigt werden konnte, dass eine grafische Darstellung der verwendeten Messungen sehr hohe Accuracy-Werte erzielen kann.

Obwohl das Thema, wie auch viele andere, die mit einer KI in Verbindung stehen, noch nicht genug ausgereift ist, um eine Analyse vom medizinischen Fachpersonal zu ersetzen, so dient es generell als eine unterstützende Entlastungshilfe bei der Untersuchung und Auswertung von großen Anzahlen an Testergebnissen. Es sei hervorzuheben, dass Modelle, wie diese, selbstverständlich zu jeder Zeit fehleranfällig sein können, weshalb Auswertungen immer von Ärzten und/oder medizinischem Fachpersonal zu überprüfen wären.

Ein KI-System mit einem derartigen Algorithmus kann auch auf andere Suchtarten angewandt werden, die ähnliche Messverhalten und Datensätze ermöglichen, sodass die Entwicklung von künstlichen Analysemodellen in anderen Gebieten sehr ähnlich verlaufen könnte.

Selbstverständlich wächst die Forschung und Weiterentwicklung solcher Systeme stetig voran, sodass neue Algorithmen aufgestellt werden, die wiederum eine höhere Vorhersagewahrscheinlichkeit bzw. eine niedrigere Fehlerquote ermöglichen.

Insgesamt ist das entwickelte Modell als Erfolg zu bewerten und kann somit als Grundlage für genauere KI-Modelle und Algorithmen in diesem und anderen medizinischen Bereichen dienen.

Literaturverzeichnis

[AACAP 2019] Alcohol Use in Families: https://www.aacap.org/AACAP/Families_and_Youth/Facts_for_Families/FFF-Guide/Children-Of-Alcoholics-017.aspx, 05.2019, letzter Zugriff: 27.02.2024

[Adam] Adam: https://keras.io/api/optimizers/adam/, letzter Zugriff: 27.02.2024

[Alc] Is Alcoholism Inherited?: https://alcohol.org/alcoholism/is-it-inherited/, 07.09.2023, letzter Zugriff: 27.02.2024

[Alt 2022] Semi-Supervised Learning, Explained with Examples: https://www.altexsoft.com/blog/semi-supervised-learning/, 18.03.2022, letzter Zugriff: 27.02.2024

[Arn 2021] ARNALDO, Muhammad: Intro to Rapidminer: A No-Code Development Platform for Data Mining (with Case Study): https://www.analyticsvidhya.com/blog/2021/10/intro-to-rapidminer-a-no-code-development-platform-for-data-mining-with-case-study/, 04.10.2021, letzter Zugriff: 27.02.2024

[Bah 2021] BAHETI, Pragati: Train Test Validation Split: How To & Best Practices: https://www.v7labs.com/blog/train-validation-test-set, 13.09.2021, letzter Zugriff: 27.02.2024

[Batch] BatchNormalization layer: https://keras.io/api/layers/normalization_layers/batch_normalization/, letzter Zugriff: 27.02.2024

[Blo 2023] Blockchain Council: Benefits Of Artificial Intelligence In Our Daily Lives: https://www.blockchain-council.org/ai/what-are-the-benefits-of-artificial-intelligence-in-our-daily-lives/, 06.02.2024, letzter Zugriff: 27.02.2024

[Boe] BOESCH, Gaudenz: Pytorch vs Tensorflow: A Head-to-Head Comparison: https://viso.ai/deep-learning/pytorch-vs-tensorflow/, letzter Zugriff: 27.02.2024

[Bon 2023]: BONTHU, Harika: Detecting and Treating Outliers | Treating the odd one out!: https://www.analyticsvidhya.com/blog/2021/05/detecting-and-treating-outliers-treating-the-odd-one-out/, 28.09.2023, letzter Zugriff: 27.02.2024

[BreCuLiWi 2022] BREIMAN, Leo; CUTLER, Adele; LIAW, Andy; WIENER, Matthew: Package 'randomForest': https://cran.r-project.org/web/packages/randomForest/randomForest.pdf, 14.10.2022, letzter Zugriff: 27.02.2024

[Bro 2020] BROWNLEE, Jason: How to Choose Loss Functions When Training Deep Learning Neural Networks: https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/, 25.08.2020, letzter Zugriff: 27.02.2024

[Bro 2022] BROWNLEE, Jason: Binary Classification Tutorial with the Keras Deep Learning Library: https://machinelearningmastery.com/binary-classification-tutorial-with-the-keras-deep-learning-library/, 05.08.2022, letzter Zugriff: 27.02.2024

[Bro 2023] BROWNLEE, Jason: A Gentle Introduction to k-fold Cross-Validation: https://machinelearningmastery.com/k-fold-cross-validation/, 04.10.2023, letzter Zugriff: 27.02.2024

[BurAhBa 2021] BURIRO, Abdul Baseer; AHMED, Bilal; BALOCH, Gulsher; AHMED, Junaid; SHOORANGIZ, Reza; WEDDELL, Stephen J.; JONES, Richard D.: Classification of alcoholic EEG signals using wavelet scattering transform-based features: https://www.sciencedirect.com/science/article/abs/pii/S0010482521007630, 12.2021, letzter Zugriff: 27.02.2024

[Cal 2005] CALKINS, Keith G.: Correlation Coefficients: https://www.andrews.edu/~calkins/math/edrm611/edrm05.htm, 18.07.2005, letzter Zugriff: 27.02.2024

[Ced 2023] Cedars-Sinai Staff: AI's Ascendance in Medicine: A Timeline: https://www.cedars-sinai.org/discoveries/ai-ascendance-in-medicine.html, 20.04.2023, letzter Zugriff: 27.02.2024

[CoGeHeNeKr 2004] COVAULT, Jonathan; GELERNTER, Joel; HESSELBROCK, Victor; NELLISSERY, Maggie; KRANZLER, Henry R.: Allelic and haplotypic association of GABRA2 with alcohol dependence: https://onlinelibrary.wiley.com/doi/full/10.1002/ajmg.b.30091, 06.07.2004, letzter Zugriff: 27.02.2024

[Con] AI is all around us: https://www.consulteer.com/ai-is-all-around-us/, letzter Zugriff: 27.02.2024

[Cot 1979] COTTON, Nancy S.: The familial incidence of alcoholism: a review: https://www.jsad.com/doi/10.15288/jsa.1979.40.89, 1979, letzter Zugriff: 27.02.2024

[Cou] 9 Best Python Libraries for Machine Learning: https://www.coursera.org/articles/python-machine-learning-library, 29.11.2023, letzter Zugriff: 27.02.2024

[Cou 2023] Deep Learning vs. Machine Learning: Beginner's Guide https://www.coursera.org/articles/ai-vs-deep-learning-vs-machine-learning-beginners-guide, 29.11.2023, letzter Zugriff: 27.02.2024

[Csv 2023] CSV explained: https://ai-jobs.net/insights/csv-explained/, 06.12.2023, letzter Zugriff: 27.02.2024

[Data 2023] What is the Dropout Layer?: https://databasecamp.de/en/ml/dropout-layer-en, 05.04.2023, letzter Zugriff: 27.02.2024

[Diaz 2021] DIAZ, Valeria Fonseca: GUI's or coding: Production vs. Operation: https://towardsdatascience.com/guis-or-coding-production-vs-operation-fc1de9e483a8, 08.06.2021, letzter Zugriff: 27.02.2024

[Dee] Feed Forward Neural Network: https://deepai.org/machine-learning-glossary-and-terms/feed-forward-neural-network, letzter Zugriff: 27.02.2024

[Don 2023] DONGES, Niklas; WHITFIELD, Brennan: A Guide to Recurrent Neural Networks: Understanding RNN and LSTM Networks: https://builtin.com/data-science/recurrent-neural-networks-and-lstm, 28.02.2023, letzter Zugriff: 27.02.2024

[FaSiKaWa 2020] FARSI, Leila; SIULY, Siuly; KABIR, Enamul; WANG, Hua: Classification of Alcoholic EEG Signals Using a Deep Learning Method: https://research.usq.edu.au/download/a17650e2d7a7f1059dd748ef52e075e860df3dbd6c7e3b5524fd0c2606141241/973904/09207939.pdf, 12.11.2020, letzter Zugriff: 27.02.2024

[FaPiSm 1997] FAYYAD, Usama; PIEATERSKY-SHAPIRO, Gregory; SMYTH, Padhraic: From Data Mining to Knowledge Discovery in Databases: https://www.kdnuggets.com/gpspubs/aimag-kdd-overview-1996-Fayyad.pdf, 08.01.1997, letzter Zugriff: 27.02.2024

[Fch] FCHOLLET: The Sequential model: https://keras.io/guides/sequential_model/, 25.06.2023, letzter Zugriff: 27.02.2024

[FiJu 1999] FINN, Peter R.; JUSTUS, Alicia: Reduced EEG Alpha Power in the Male and Female Offspring of Alcoholics: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1530-0277.1999.tb04108.x, 30.05.2006, letzter Zugriff: 27.02.2024

[Gal 2024] GALBICSEK, Carol: Alcoholism In The Workplace: https://www.alcoholrehabguide.org/resources/alcoholism-workplace/, 09.01.2024, letzter Zugriff: 27.02.2024

[GaMeVoPoSclt 1982] GABRIELLI JR., William F.; MEDNICK, Sarnoff A.; VOLVAVKA, Jan; POLLOCK, Vicki E.; SCHULSINGER, Fini; ITIL, Turan M.: Electroencephalograms in Children of Alcoholic Fathers: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1469-8986.1982.tb02495.x, 07.1982, letzter Zugriff: 27.02.2024

[Gee] ML | Underfitting and Overfitting: https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/, letzter Zugriff: 27.02.2024

[Geek] Reinforcement learning: https://www.geeksforgeeks.org/what-is-reinforcement-learning/, letzter Zugriff: 27.02.2024

[Geeks] Stratified K Fold Cross Validation: https://www.geeksforgeeks.org/stratified-k-fold-cross-validation/, letzter Zugriff: 27.02.2024

[Gib 2020] GIBBONS, Serenity: 5 Life-Saving Applications Of Artificial Intelligence: https://www.forbes.com/sites/serenitygibbons/2020/08/25/5-life-saving-applications-of-artificial-intelligence/?sh=261424751c58, 25.08.2020, letzter Zugriff: 27.02.2024

[GilSteHu 2021] GILLIS, Alexander S.; STEDMAN, Craig; HUGHES, Adam: data mining: https://www.techtarget.com/searchbusinessanalytics/definition/data-mining, 02.2024, letzter Zugriff: 27.02.2024

[Gold 1993] GOLDMAN, David: Recent developments in alcoholism:genetic transmission: https://pubmed.ncbi.nlm.nih.gov/8234925/, letzter Zugriff: 27.02.2024

[Gul] GULATI, Aman Preet: Dealing with Outliers Using the IQR Method: https://www.analyticsvidhya.com/blog/2022/09/dealing-with-outliers-using-the-iqr-method/, 13.09.2022, letzter Zugriff: 27.02.2024

[Gul 2022] GULATI, Aman Preet: Dealing with outliers using the Z-Score method: https://www.analyticsvidhya.com/blog/2022/08/dealing-with-outliers-using-the-z-score-method/, 02.09.2022, letzter Zugriff: 27.02.2024

[Gup] GUPTA, Dishashree: Fundamentals of Deep Learning – Activation Functions and When to Use Them?: https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/, 03.11.2023, letzter Zugriff: 27.02.2024

[Gup 2023] GUPTA, Ayush: A Comprehensive Guide on Optimizers in Deep Learning: https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/, 23.01.2024, letzter Zugriff: 27.02.2024

[Har 2018] HARLALKA, Rajat: Choosing the Right Machine Learning Algorithm: https://medium.com/hackernoon/choosing-the-right-machine-learning-algorithm-68126944ce1f, 18.06.2018, letzter Zugriff: 27.02.2024

[Hin 2016] HINZE, Chris: Spammende Nutzer automatisiert erkennen (Seite 4): https://www.linux-magazin.de/ausgaben/2016/12/machine-learning/4/, 12.2016, letzter Zugriff: 27.02.2024

[IbmKnn] What is a neural network?. https://www.ibm.com/topics/neural-networks, letzter Zugriff: 27.02.2024

[IbmMl] What is machine learning?: https://www.ibm.com/topics/machine-learning, letzter Zugriff: 27.02.2024

[IbmSl] What is supervised learning?: https://www.ibm.com/topics/supervised-learning, letzter Zugriff: 27.02.2024

[IbmUsl] What is unsupervised learning?: https://www.ibm.com/topics/unsupervised-learning, letzter Zugriff: 27.02.2024

[Ics] EEG Databsse: https://archive.ics.uci.edu/dataset/121/eeg+database, 12.10.1999, letzter Zugriff: 27.02.2024

[Int] Python Vs R: Know The Difference: https://www.interviewbit.com/blog/python-vs-r/, 04.01.2024, letzter Zugriff: 27.02.2024

[John 2019] JOHN, Robert Thas: Regularization in TensorFlow using Keras API: https://johnthas.medium.com/regularization-in-tensorflow-using-keras-api-48aba746ae21, 14.01.2019, letzter Zugriff: 27.02.2024

[Jup] Jupyter Notebook: https://jupyter.org/, letzter Zugriff: 27.02.2024

[KamArdPan 2020] KAMARAJAN, Chella; ARDEKANI, Babak A.; PANDEY, Ashwini K.; CHORLIAN, David B.; KINREICH, Sivan; PANDEY, Gayathri; MEYERS, Jacquelyn L.; ZHANG, Jian; KUANG, Weipeng; STIMUS, Arthur T.; PORJESZ, Bernice: Random Forest Classification of Alcohol Use Disorder Using EEG Source Functional Connectivity, Neuropsychological Functioning, and Impulsivity Measures: https://pubmed.ncbi.nlm.nih.gov/32121585/, 03.2020, letzter Zugriff: 27.02.2024

[Kar 2023] KARJIAN, Ron: The history of artificial intelligence: Complete AI timeline: https://www.techtarget.com/searchenterpriseai/tip/The-history-of-artificial-intelligence-Complete-AI-timeline, 16.08.2023, letzter Zugriff: 27.02.2024

[Kra 2016] KRAWCZYK, Bartosz: Learning from imbalanced data: open challenges and future directions: https://link.springer.com/article/10.1007/s13748-016-0094-0, 22.04.2016, letzter Aufruf: 27.02.2024

[Ker] Keras: https://keras.io/, letzter Zugriff: 27.02.2024

[Kil 2020] KILLOCK, David: AI outperforms radiologists in mammographic screening: https://www.nature.com/articles/s41571-020-0329-7, 21.01.2020, letzter Zugriff: 27.02.2024

[Klon] AI history: the Dartmouth Conference: https://www.klondike.ai/en/ai-history-the-dartmouth-conference/, letzter Zugriff: 27.02.2024

[Knime] KNIME: https://www.knime.com/, letzter Zugriff: 27.02.2024

[Kum 2020] KUMAR, Anupam: A Simple Neural Networks for Binary Classification -Understanding Feed Forward: https://medium.com/afblabs-data-science/a-simple-neural-networks-for-binary-classification-understanding-feed-forward-68c3c0659f78, 19.04.2020, letzter Zugriff: 27.02.2024

[Lac 2016] LACKI, Miroslaw: Intelligent Prediction of Ship Maneuvering: https://www.researchgate.net/figure/Influence-of-bias-coefficient-j-to-value-of-sigmoid-function_fig1_312057335, 09.2016, letzter Zugriff: 27.02.2024

[Lin] What is the difference between a recurrent and feedforward neural network?: https://www.linkedin.com/advice/3/what-difference-between-recurrent-feedforward, letzter Zugriff: 27.02.2024

[LiWu 2021] LI, Houchi; WU, Lei: EEG Classification of Normal and Alcoholic by Deep Learning: https://www.mdpi.com/2076-3425/12/6/778, 14.06.2022, letzter Zugriff: 27.02.2024

[Luna 2022] LUNA, Javier Canales: Python vs R for Data Science: Which Should You Learn?: https://www.datacamp.com/blog/python-vs-r-for-data-science-whats-the-difference, 12.2022, letzter Zugriff: 27.02.2024

[Maas 2017] World's first super-microsurgery operation with 'robot hands': https://www.maastrichtuniversity.nl/news/world%E2%80%99s-first-super-microsurgery-operation-%E2%80%98robot-hands%E2%80%99, 02.10.2017, letzter Zugriff: 27.02.2024

[Mar] MARR, Bernard: What Is The Difference Between Data Mining And Machine Learning?: https://bernardmarr.com/what-is-the-difference-between-data-mining-and-machine-learning/, letzter Zugriff: 27.02.2024

[Mat] Matplotlib: https://matplotlib.org/, letzter Zugriff: 27.02.2024

[Math] What Is a Neural Network? 3 things you need to know: https://www.mathworks.com/discovery/neural-network.html, letzter Zugriff: 27.02.2024

[May 2022] EEG (electroencephalogram): https://www.mayoclinic.org/tests-procedures/eeg/about/pac-20393875, 11.05.2022, letzter Zugriff: 27.02.2024

[Mid 2021] MIDDLETON, Michael: Deep Learning vs. Machine Learning: https://flatironschool.com/blog/deep-learning-vs-machine-learning/, 08.02.2021, letzter Zugriff: 27.02.2024

[Mis 2020] MISHRA, Mayank: Convolutional Neural Networks, Explained: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939, 26.08.2020, letzter Zugriff: 27.02.2024

[Mod 2022] MODASIYA, Kishan: What is random-state?: https://medium.com/mlearning-ai/what-the-heck-is-random-state-24a7a8389f3d, 25.06.2022, letzter Zugriff: 27.02.2024

[MoGoMaAn 2012] MOROZOVA, Tatiana V; GOLDMAN, David; MACKAY, Trudy FC; ANHOLT, Robert RH: The genetic basis of alcoholism: multiple phenotypes, many genes, complex networks: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3334563/, 20.02.2012, letzter Zugriff: 27.02.2024

[MuQaZa 2021] MUKHTAR, Hamid; QAISAR, Saaed Mian; ZAGUIA, Atef: Deep Convolutional Neural Network Regularization for Alcoholism Detection Using EEG Signals: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8402228/pdf/sensors-21-05456.pdf, 13.08.2021, letzter Zugriff: 27.02.2024

[Nag 2017] NAGPAL, Anuja: Over-fitting and Regularization: https://towardsdatascience.com/over-fitting-and-regularization-64d16100f45c, 11.10.2017, letzter Zugriff: 27.02.2024

[Nar 2018] NARKHEDE, Sarang: Understanding Confusion Matrix: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62, 09.05.2018, letzter Zugriff: 27.02.2024

[Nark 2018] NARKHEDE, Sarang: Understanding AUC - ROC Curve: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5, 26.06.2018, letzter Zugriff: 27.02.2024

[Naw 2022] NAWALE, Tejashree: How to deal with Missing Values in Machine Learning: https://medium.com/geekculture/how-to-deal-with-missing-values-in-machine-learning-98e47f025b9c, 17.05.2022, letzter Zugriff: 27.02.2024

[Num] NumPy: https://numpy.org/, letzter Zugriff: 27.02.2024

[Nvi] PyTorch: https://www.nvidia.com/en-us/glossary/data-science/pytorch/, letzter Zugriff: 27.02.2024

[Oppe] OPPERMANN, Artem: Optimization in Deep Learning: AdaGrad, RMSProp, ADAM: https://artemoppermann.com/optimization-in-deep-learning-adagrad-rmsprop-adam/, letzter Zugriff: 27.02.2024

[Opt] Optuna: https://optuna.org/, letzter Zugriff: 27.02.2024

[Pai 2023] PAI, Aravindpai: Analyzing Types of Neural Networks in Deep Learning: https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/, 13.07.2023, letzter Zugriff: 27.02.2024

[Pan] pandas: https://pandas.pydata.org/, letzter Zugriff: 27.02.2024

[Pau 2018] PAUL, Sayak: Diving Deep with Imbalanced Data: https://www.datacamp.com/tutorial/diving-deep-imbalanced-data, 10.2018, letzter Zugriff: 27.02.2024

[PoEaGa 1995] POLLOCK, Vicki E.; EARLEYWINE, Mitchell; GABRIELLI, William F.: Personality and EEG in Older Adults with Alcoholic Relatives: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1530-0277.1995.tb01470.x, 02.1995, letzter Zugriff: 27.02.2024

[PoRaKaJoPaBe 2004] PORJESZ, Bernice; RANGASWAMY, Madhavi; KAMARAJAN, Chella; JONES, Kevin A.; PADMANABHAPILLAI, Ajayan; BEGLEITER, Henri: The utility of neurophysiological markers in the study of alcoholism: https://www.downstate.edu/research/centers-departments/henri-begleiter-neurodynamics-laboratory/_documents/pdf/2005-Porjesz-The%20utility%20of%20neurophysiological%20markers%20in%20the%20study%20of%20alcoholism.pdf, 17.12.2004, letzter Zugriff: 27.02.2024

[Pra 2020] PRATHAP, Prajeesh: Feed-forward and Recurrent Neural Networks: The Future of Machine Learning: https://medium.com/@prajeeshprathap/feed-forward-and-recurrent-neural-networks-the-future-of-machine-learning-8b2c1975f0c5, 20.07.2020, letzter Zugriff: 27.02.2024

[Pra 2022] PRAMODITHA, Rukshan: Two or More Hidden Layers (Deep) Neural Network Architecture: https://medium.com/data-science-365/two-or-more-hidden-layers-deep-neural-network-architecture-9824523ab903, 01.01.2022, letzter Zugriff: 27.02.2024

[Py] Python: https://www.python.org/, letzter Zugriff: 27.02.2024

[PyTo] PyTorch: https://pytorch.org/, letzter Zugriff: 27.02.2024

[R] What is R?: https://www.r-project.org/about.html, letzter Zugriff: 27.02.2024

[Rap] RapidMiner: https://docs.rapidminer.com, letzter Zugriff: 27.02.2024

[Sax 2022] SAXENA, Saurabh: Precision-Recall Curve: https://pub.towardsai.net/precision-recall-curve-26f9e7984add, 03.10.2022, letzter Zugriff: 27.02.2024

[Sci] scikit-learn: https://scikit-learn.org/stable/, letzter Zugriff: 27.02.2024

[Sccn 2023] EEG / ERP data available for free public download: https://sccn.ucsd.edu/~arno/fam2data/publicly_available_EEG_data.html, 2023, letzter Zugriff: 27.02.2024

[Sea] seaborn: statistical data visualization: https://seaborn.pydata.org/, letzter Zugriff: 27.02.2024

[Sha 2023] SHARP, Amelia: The Effects of Alcoholism on Families: How Alcoholism Effects Families: https://americanaddictioncenters.org/alcoholism-treatment/family-marital-problems, 21.02.2024, letzter Zugriff: 27.02.2024

[ShaChaLeLuNuPi 1991] SHARBROUGH, F.; CHATRIAN, G.E.; LESSER, Ronald; LUDERS, H.; NUWER, M.; PICTON, Terence W.: American Electroencephalographic Society guidelines for standard electrode position nomenclature: https://www.researchgate.net/publication/284654024_American_Electroencephalographic_Society_guidelines_for_standard_electrode_position_nomenclature,

01.1999, letzter Zugriff: 27.02.2024

[Shah 2017] SHAH, Tarang: About Train, Validation and Test Sets in Machine Learning: https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7, 06.12.2017, letzter Zugriff: 27.02.2024

[Shi 2022] How to choose the Value of k in K-fold Cross-Validation: https://www.shiksha.com/online-courses/articles/how-to-set-the-value-of-k-in-k-fold-cross-validation/, 14.06.2022, letzter Zugriff: 27.02.2024

[Sim 2023] Simplilearn: What Is Data Mining: Definition, Benefits, Applications, and More: https://www.simplilearn.com/what-is-data-mining-article, 10.08.2023, letzter Zugriff: 27.02.2024

[SnoVa 1980] SNODGRASS, Joan G.; VANDERWART, Mary: A standardized set of 260 pictures: Norms for name agreement, image agreement, familiarity, and visual complexity: https://psycnet.apa.org/record/1981-06756-001, 1980, letzter Zugriff: 27.02.2024

[Syd 2021] SYDORENKO, Iryna: How to Choose the Right Machine Learning Algorithm: A Pragmatic Approach: https://labelyourdata.com/articles/how-to-choose-a-machine-learning-algorithm, 03.05.2021, letzter Zugriff: 27.02.2024

[T 2020] T, Buddy: The Role of Genetics in Alcoholism: https://www.verywellmind.com/alcoholism-is-it-inherited-63171, 16.11.2023, letzter Zugriff: 27.02.2024

[Ter 2023] TERRA, John: Keras vs Tensorflow vs Pytorch: Key Differences Among Deep Learning: https://www.simplilearn.com/keras-vs-tensorflow-vs-pytorch-article, 31.01.2024, letzter Zugriff: 27.02.2024

[Tf] Tensorflow: https://www.tensorflow.org/, letzter Zugriff: 27.02.2024

[Tha 2023] THAKUR, Ayush: What's the Optimal Batch Size to Train a Neural Network?: https://wandb.ai/ayush-thakur/dl-question-bank/reports/What-s-the-Optimal-Batch-Size-to-Train-a-Neural-Network---VmlldzoyMDkyNDU, 09.07.2023, letzter Zugriff: 27.02.2024

[Tho] THOMPSON, Giles: Using Neural Networks to Detect Alcoholism from Simple EEG Features: http://users.cecs.anu.edu.au/~Tom.Gedeon/conf/ABCs2021/paper1/ABCs2021_paper_46.pdf, letzter Zugriff: 27.02.2024

[Tut] Keras - Dense Layer: https://www.tutorialspoint.com/keras/keras_dense_layer.htm, letzter Zugriff: 27.02.2024

[Twin 2023] TWIN, Alexandra: What Is Data Mining? How It Works, Benefits, Techniques, and Examples: https://www.investopedia.com/terms/d/datamining.asp, 23.02.2024, letzter Zugriff: 27.02.2024

[Tya 2021] TYAGI, Neelam: L2 and L1 Regularization in Machine Learning: https://www.analyticssteps.com/blogs/l2-and-l1-regularization-machine-learning, 01.03.2021, letzter Zugriff: 27.02.2024

[Unzu 2022] UNZUETA, Diego: Fully Connected Layer vs. Convolutional Layer: Explained: https://builtin.com/machine-learning/fully-connected-layer, 18.10.2022, letzter Zugriff: 27.02.2024

[W3] Statistics - Standard Deviation: https://www.w3schools.com/statistics/statistics_standard_deviation.php, letzter Zugriff: 27.02.2024

[Wod 2023] WODECKI, Ben: 7 AI Programming Languages You Need to Know: https://aibusiness.com/verticals/7-ai-programming-languages-you-need-to-know, 05.05.2023, letzter Zugriff: 27.02.2023

[Wol 2023] WOLFEWICZ, Arne: Deep Learning vs. Machine Learning – What's The Difference?: https://levity.ai/blog/difference-machine-learning-deep-learning, 15.02.2023, letzter Zugriff: 27.02.2024

[Yem 2019] YEMULWAR, Sangita: Feature Selection Techniques https://medium.com/analytics-vidhya/feature-selection-techniques-2614b3b7efcd, 27.09.2019, letzter Zugriff: 27.02.2024

[Zav 2023] ZAVERIA: What is the Role of Python in Artificial Intelligence?: https://www.analyticsinsight.net/what-is-the-role-of-python-in-artificial-intelligence/, 16.04.2023, letzter Zugriff: 27.02.2024

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorstehende Bachelorthesis mit dem Titel

Entwicklung eines künstlichen neuronalen Netzes zur Feststellung einer genetischen Prädisposition zum Alkoholismus anhand von EEG-Messungen

– bzw. im Falle einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – selbstständig ohne fremde Hilfe gefertigt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quelle kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Ort, Datum, Unterschrift