Diplomarbeit

Extraktion und Standardisierung des Raucherstatus aus freitextlicher-klinischer Routinedokumentation unter Verwendung von Methoden des maschinellen Lernens

eingereicht von

Anto KNEZOVIC

zur Erlangung des akademischen Grades

Doktor der gesamten Heilkunde
(Dr. med. univ.)

an der

Medizinischen Universität Graz

ausgeführt am
Institut für Medizinische Informatik, Statistik und Dokumentation
an der Medizinische Universität Graz

unter Anleitung von

Ass.-Prof. Dipl.-Ing. Dr.scient.med. Markus Kreuzthaler
Univ.-Prof. Dr. med. Stefan Schulz

Graz, 27. April 2023

Eidesstattliche Erklärung

Ich erkläre ehrenwörtlich, dass ich die vorliegende Arbeit selbständig angefertigt und abgefasst, und jene Personen und Institutionen, die am Zustandekommen der Forschungsdaten beteiligt waren, namentlich genannt habe. Andere als die angegebenen Quellen habe ich nicht verwendet und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen habe ich als solche kenntlich gemacht. Die Arbeit an der Diplomarbeit und daraus entstandener Publikationen wurde gemäß den Regeln der „Good Scientific Practice“ durchgeführt.

Graz, 27th April 2023 Knezovic Anto

Danksagung

Diese Arbeit wurde im Jahr 2023 am Institut für Medizinische Informatik, Statistik und Dokumentation an der Medizinischen Universität Graz verfasst.

Mein größter Dank gilt meinen beiden Betreuern Ass.-Prof. Dipl.-Ing. Dr.scient.med. Markus Kreuzthaler und Univ.-Prof. Dr. med. Stefan Schulz. Vor allem die engmaschige Motivation und Hilfestellung bei Bedarf von ersterem haben größtmöglich zur Fertigstellung der Arbeit beigetragen. Jemanden zu haben, der stets Zeit für eine Kontrolle oder Erklärung hat, ist von unschätzbarem Wert. Ohne diese Hilfe hätte diese Arbeit nie entstehen können.

Weiters möchte ich mich bei Edmund Weitz, Keith Galli, Joshua Starmer und Grant Sanderson bedanken, welche unabhängig voneinander Ressourcen zum Lernen und Wiederholen von hochschulmathematischen Themen und Lehrvideos zu maschinellem Lernen frei zur Verfügung gestellt haben. Diese Ressourcen haben das Festigen der notwendigen Grundlagen stark vereinfacht.

Mein Dank gilt auch Alexandra Elbakyan, deren Beitrag zur Wissenschaft nicht hoch genug gewertet werden kann.

Zu guter Letzt möchte ich mich noch bei meiner Familie bedanken, welche mich im Laufe des gesamten Studiums als zweiten Bildungsweg stets unterstützt hat.

Graz, am 27. April 2023 Knezovic Anto

Inhaltsverzeichnis

Abkürzungsverzeichnis

CIS Clinical Information System. 2
CNN Convolutional Neural Network. 15, 38
COPD Chronic obstructive pulmonary disease. 1

EHR Electronic Health Records. 2, 39

FAIR Findable Accessible Interoperable Reusable. 40
FHIR Fast Healthcare Interoperability Resource. 40

HL7 Health Level 7 Standards Organization. 40

KI künstliche Intelligenz. v, 3, 4
KNN künstliches neuronales Netzwerk. v, vi, vii, vi, 4, 6, 8, 15, 16, 28, 32, 33, 34, 35, 36, 37

LSTM Long Short-Term Memory. v, vi, vii, vi, v, 6, 8, 17, 28, 34, 35, 36, 37, 38

ML Machine Learning. v, 3, 4, 5, 7, 11, 18, 21, 22, 28, 38

NLP Natural Language Processing. 18, 40

RNN rekurrentes neuronales Netz. 6, 8, 15, 17

SMOTE Synthetic Minority Oversampling Technique. v, vii, vi, v, 27, 28, 30, 31, 32, 34, 37
SNOMED CT Systematized Nomenclature of Medicine Clinical Terms. vii, 26, 40
SVM Support Vector Machine. v, vii, vi, v, 6, 8, 9, 10, 27, 28, 30, 31, 37, 38

TF-IDF Term Frequency-Inverse Document Frequency. 19
TTS transdermales therapeutisches System. 25

UIMA Unstructured Information Management Architecture. 40

Abbildungsverzeichnis

Tabellenverzeichnis

Zusammenfassung

Einleitung. Der Raucherstatus von Patientinnen und Patienten hat einen großen Einfluss auf Pathogenese und Salutogenese. Die richtige Klassifikation des Status ist von großem Wert, sowohl für die Behandlung von Patientinnen und Patienten, als auch die Datenauswertung für retrospektive epidemiologische Studien.

Methoden. Die Arbeit besteht darin, drei Methoden des maschinellen Lernens darin zu vergleichen, klinische Routinedokumentation (Arztbriefe) anhand des Raucherstatus zu klassifizieren. Der verwendete Datensatz wurde von zwei Fachpersonen je einer von 6 Klassen zugewiesen. Mit diesen Daten wurde jeweils eine SVM, ein Feedforward KNN und ein KNN mit einer LSTM-Architektur trainiert und ausgewertet.

Ergebnisse. Die Interrater-Reliabilität der beiden Fachpersonen ergab ein Cohens Kappa von $\kappa = 89,97\%$. Die SVM erreichte sowohl ohne als auch mit SMOTE einen gewichteten F_1-Score von 88%. Denselben Wert hat auch das Feedforward KNN erreicht. Das KNN mit einer LSTM-Architektur erreichte als bestes in dieser Arbeit einen gewichteten F_1-Score von 92%.

Diskussion. Die Performance des LSTM-Modells ist gut vergleichbar mit verwandten Arbeiten. Präzisierung erfordert noch die Klassifizierung seltener Klassen. Auch besteht ein gewisser Selektionsbias, da die Daten nur von bestimmten Kliniken erhoben wurden. Inhärente Fehlerfortpflanzung in den Arztbriefen und unterschiedliche Notationsgewohnheiten für den Raucherstatus ergeben weitere Unschärfen, welche durch größere und diversere Datensätze minimiert werden können. Eine Verwendung für retrospektive, epidemiologische Studien oder transformerbasierte Modelle könnte bei Verbesserung der Performance in Betracht gezogen werden.

Abstract

Introduction. The smoking status of patients has major impact on pathogenesis and salutogenesis. The correct classification of the smoking status therefore has significant value, both for the treatment of patients and for data analysis for retrospective epidemiological studies.

Methods. This thesis consists of training three machine learning methods to classify routine clinical documentation (discharge summaries formulated as doctor's letters) based on smoking status. The data set used was annotated by two experts as belonging to one of 6 classes. A SVM, a Feedforward ANN (artificial neural network) and an ANN with LSTM-architecture were each trained and evaluated with these data.

Results. The inter-rater reliability of the two experts resulted in a Cohen's kappa of $\kappa = 89,97\%$. The SVM achieved a weighted F_1-score of 88% both without and with SMOTE. The Feedforward ANN scored 88% too while the ANN with LSTM-architecture achieved a weighted F_1-score of 92%

Discussion. The performance of the LSTM model is well comparable with related work. More precision is still required for classification of rare classes. There is also a certain selection bias, since the data was only collected from specific clinics. Inherent error propagation in the discharge summaries and different notation habits for the smoking status result in further uncertainties, which can be minimized by larger and more diverse data sets. Use for retrospective epidemiological studies or transformer-based models could be considered if performance is improved.

Kapitel 1

Einleitung

1.1 Motivation

1.1.1 Rauchen und seine Folgen

Die gesundheitlichen Effekte des Rauchens sind ausführlich erforscht. Dazu zählen unter anderem systemische Folgen wie Atherosklerose und COPD [1], dermatologische Auswirkungen wie schlechte Wundheilung, beschleunigte Hautalterung, Psoriasis, Haarverlust, orale Schleimhautkarzinome und Plattenepithelkarzinome [2], Einflüsse auf die Knochengesundheit wie Osteoporose und Frakturen [3] bzw. Frakturheilung [4].

Die größten Effekte des Rauchens sind Beiträge zur Entstehung von Lungenkarzinomen [5] und Karzinomen des Urogenitaltraktes sowie der Erhöhung der Komplikationsrate anderer Nephropathien [6]. Eine präzise Identifikation des Raucherstatus bei Patientinnen und Patienten kann folglich bei der Prävention und Sekundärprävention der oben erwähnten Folgen helfen.

1.1.2 Elektronische Datenerfassung

Clinical Information Systems (CISs) sind essenziell für die Datenpersistierung in der Medizin. Die akkurate Klassifizierung von Patientinnen und Patienten und Fällen ist entscheidend für die Erkennung von Faktoren in der Krankheitsentstehung und entsprechender Prävention. Als Beispiel sei hier die Erkennung von Personen mit Nikotinabusus erwähnt. Deren Klassifizierung enthält so viele Ungenauigkeiten, dass 80% der Daten Fehler aufweisen, wie beispielsweise fehlende Angaben zur Anzahl der Pack-Years (42,7%), veraltete Daten (25,1%) oder fehlende Angaben zur Dauer des Status ohne Nikotinabusus (17,4%) [7].

Botsis et al. haben eruiert, dass viele Datensätze von Pankreaskarzinompatientinnen und -patienten unvollständige Diagnosemarker haben [8]. Als Beispiele seien hier die fehlende Familienanamnese (39%) oder der fehlende Nikotin- und Alkoholstatus (27-29%) erwähnt. Aber auch histologische Parameter wie die Mitoseanzahl (21%) und Differenzierung (38%) waren teilweise nicht vorhanden. Wenngleich nicht jeder diagnostische Test oder Parameter die gleiche Gewichtung in der Diagnostik hat, so haben solche fehlenden Daten und Ungenauigkeiten womöglich einen negativen Einfluss auf die Behandlung. In der Arbeit von Botsis et al. werden Patientinnen und Patienten erwähnt, welche widersprechende Diagnosen gleichzeitig erhalten haben (Diabetes Typ 1 und Typ 2). Sicherlich jedoch beeinflussen diese Ungenauigkeiten die sekundäre Nutzung der Daten wie z.B. die retrospektive wissenschaftliche Datenauswertung durch einen systematischen Fehler (Bias) zugunsten der diagnostisch verwendeten Diagnosemarker.

Die Tatsache, dass Ärztinnen und Ärzte in der Nutzung von elektronischen Aufzeichnungen (Electronic Health Records (EHRs)) viele Texte kopieren, trägt auch zur Fortpflanzung von Fehlern bei [9].

Nicht nur das Kopieren von EHRs führt zur Vermehrung von Fehlern, sondern auch die Diskrepanz der Wahrnehmung zwischen medizinischem Personal und Patientin bzw. Patient. Thebault et al. haben entdeckt, dass die Einteilung des Raucherstatus im Mittel nur zu 60,3% zwischen Klinikpersonal und Patientin bzw. Patient übereinstimmt. Hierbei sind erhebliche Unterschiede zwischen einzelnen ärztlichen Personen zu beobachten, welche von 41,2% Übereinstimmung im zehnten Perzentil und 77,8% Übereinstimmung im neunzigsten Perzentil reichen [10].

1.2 Grundlagen

1.2.1 Künstliche Intelligenz

Künstliche Intelligenz (KI) ist schwierig zu definieren, da bereits der Begriff der Intelligenz nicht einfach fassbar ist. Russell und Norvig haben in ihrem Lehrbuch [11] acht Definitionen für KI von unterschiedlichen Personen und mit unterschiedlichem Schwerpunkt genannt, welche in Tab. 1.1 dargestellt sind. Die Schwerpunkte sind eingeteilt in Denken und Handeln, jeweils menschlich oder rational.

Menschliches Denken	Menschliches Handeln
„Das spannende, neuartige Unterfangen, Computern das Denken beizubringen, ... Maschinen mit Verstand im wahrsten Sinne des Wortes.“ (Haugeland, 1985 [12])	„Die Studie mentaler Fähigkeiten durch die Nutzung programmiertechnischer Modelle.“ (Charniak, 1985 [13])
„[Die Automatisierung von] Aktivitäten, die wir dem menschlichen Denken zuordnen, Aktivitäten wie beispielsweise Entscheidungsfindung, Problemlösung, Lernen ...“ (Bellman, 1978 [14])	„Das Studium derjenigen mathematischen Formalismen, die es ermöglichen, wahrzunehmen, logisch zu schließen und zu agieren.“ (Winston, 1992 [15])
Rationales Denken	**Rationales Handeln**
„Die Kunst, Maschinen zu schaffen, die Funktionen erfüllen, die, werden sie von Menschen ausgeführt, der Intelligenz bedürfen.“ (Kurzweil, 1990 [16])	„Computerintelligenz ist die Studie des Entwurfs intelligenter Agenten.“ (Poole et al., 1998 [17])
„Das Studium des Problems, Computer dazu zu bringen, Dinge zu tun, bei denen ihnen momentan der Mensch noch überlegen ist.“ (Rich und Knight, 1991 [18])	„KI ... beschäftigt sich mit intelligentem Verhalten in künstlichen Maschinen.“ (Nilsson, 1998 [19])

Tabelle 1.1: Acht Definitionen für den Begriff “künstliche Intelligenz“. Tabelle entnommen aus dem Lehrbuch “Künstliche Intelligenz: ein moderner Ansatz“ [11].

Im Rahmen der rechnergestützten KI-Entwicklung in den letzten Jahrzehnten hat sich ein Begriff etabliert, bei der vor allem die Methodik im Vordergrund steht: Machine Learning (ML) (deutsch: Maschinelles Lernen).

1.2.2 Maschinelles Lernen

Das ML ist eine Unterdisziplin der KI und beschreibt Methoden, die für lernende Maschinen notwendig sind. Die Übersicht der Begriffe "künstliche Intelligenz", "Machine Learning" und "künstliche neuronale Netzwerke" bzw. "Deep Learning" ist in Abb. 1.1 dargestellt.

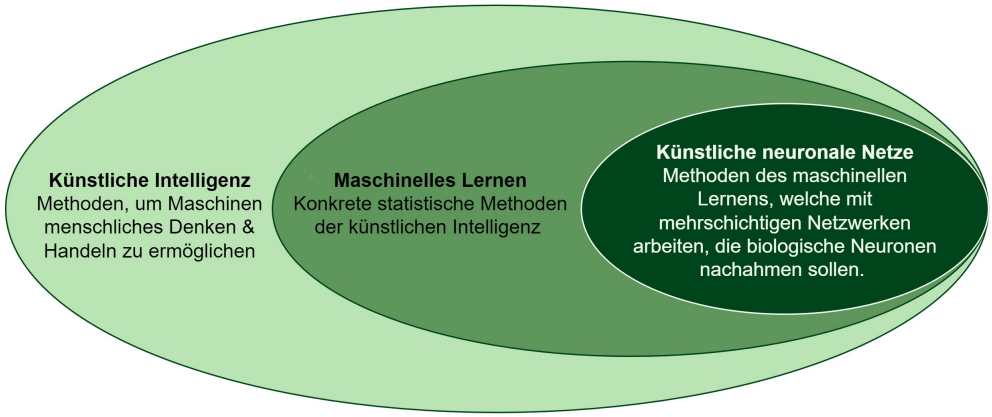

Abbildung 1.1: Zuordnung der Begriffe "künstliche Intelligenz", "Machine Learning" und "künstliche neuronale Netzwerke" in einem Venn-Diagramm [20].

ML lässt sich je nach Fragestellung und Datenlage unterteilen in: Supervised Learning (deutsch: Überwachtes Lernen), Unsupervised Learning (deutsch: Unüberwachtes Lernen) und Reinforcement Learning (deutsch: Verstärkendes Lernen).

1.2.2.1 Supervised Learning

Beim Supervised Learning besitzen die Eingabedaten sogenannte Labels, um sie zu kategorisieren. Das Modell lernt anhand dieser Eingabedaten, neue und ihm unbekannte Datenpunkte auf eine möglichst ähnliche Weise einem Label zuzuordnen (siehe Abb. 1.2). Diese Einteilung kann diskret oder kontinuierlich sein. Bei diskreten Einteilungen handelt es sich um ein Klassifikationsproblem, z.B. Nichtraucherin oder Raucher. Ist die Einteilung kontinuierlich, so hat man ein Regressionsproblem vorliegen, z.B. die Wahrscheinlichkeit an Prostatakrebs zu erkranken.

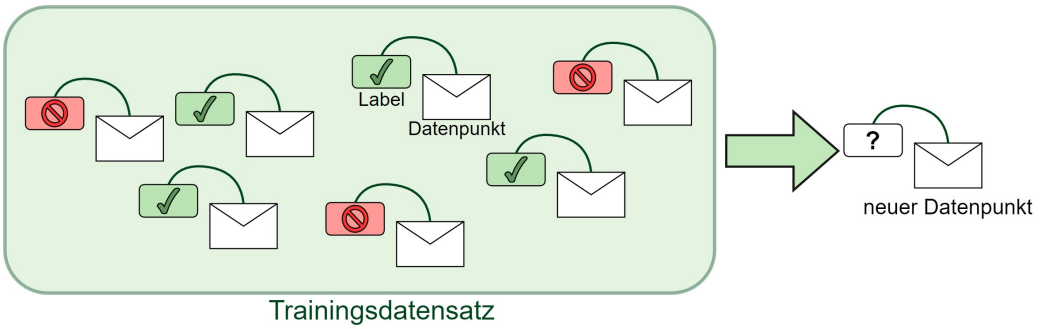

Abbildung 1.2: Beispiel für Supervised Learning: Erkennung von Spam für E-Mails. Bild aus: [21].

Mathematisch ausgedrückt, handelt es sich bei Klassifikation und Regression um zwei Arten von Abbildungen. Es sei $X \in \mathbb{R}^d$ der Eingaberaum mit der Dimensionalität $d \in \mathbb{N}$, dann nennt man die Abbildung

$$X \to Y \in \mathbb{N}^{d_2}$$

eine Klassifikation und

$$X \to Y \in \mathbb{R}^{d_2}$$

eine Regression, wobei Y hierbei als Ausgaberaum gilt. In Abb. 1.3 sind diese beiden Varianten grafisch dargestellt.

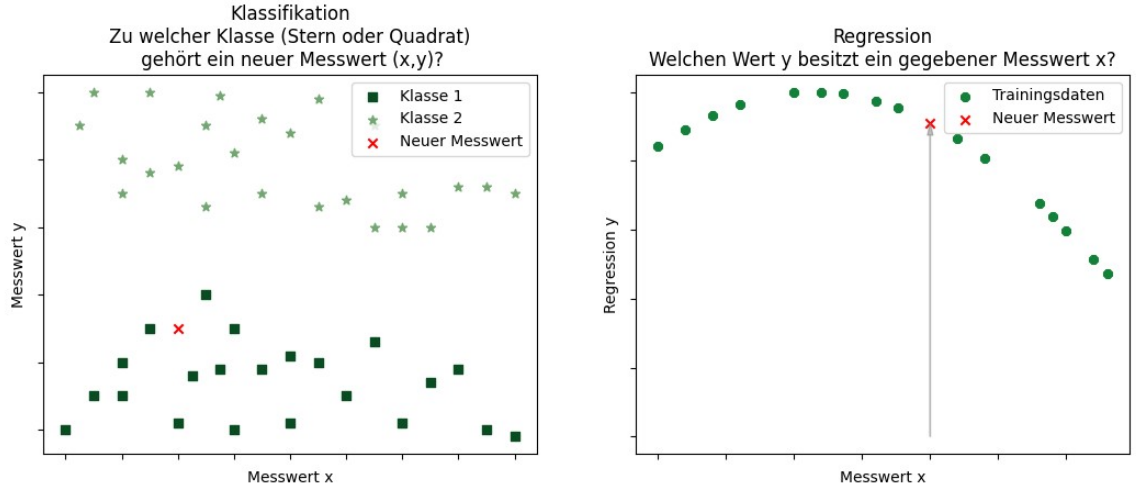

Abbildung 1.3: Schematische Darstellung zweier Arten des ML. Links: Klassifikation. Bei gegebenen Werten x, y soll das Modell Klasse 1 (Quadrat) oder Klasse 2 (Stern) ausgeben. Rechts: Regression. Bei gegebenem Eingabewert x soll das Modell anhand der Daten den passenden Wert y bestimmen.

Um diese Aufgabe zu bearbeiten, wurden unterschiedliche Algorithmen entwickelt. Die wichtigsten seien hier aufgelistet [21]: k-nächste-Nachbarn, lineare Regression, logistische Regression, Support Vector Machines *(SVM)*, Entscheidungsbäume und Random Forests, künstliche neuronale Netzwerke *(KNN)* bzw. rekurrente neuronale Netze *(RNN)*. Die Zuordnung eines Raucherstatus anhand von freitextlicher, klinischer Routinedokumentation wurde in der vorliegenden Arbeit über ein Multiclass-Klassifikationsproblem abgebildet und anhand dreier ausgewählter Modellansätze verglichen.

1.2.2.2 Unsupervised Learning

Im Gegensatz zum Supervised Learning arbeitet Unsupervised Learning ohne Labels. Dabei versucht der Algorithmus selbst Muster in den Daten zu erkennen und diese zu sogenannten Clustern zusammenzufassen. Auch Komprimierungsaufgaben lassen sich mittels Unsupervised Learning durchführen.

1.2.2.3 Reinforcement Learning

Reinforcement Learning benutzt einen Belohnungsmechanismus, um bestimmte Entscheidungen zu verstärken oder abzuschwächen. Dabei handelt es sich im Normalfall um Problemstellungen des Handelns, wie beispielsweise das Spielen eines Spiels oder das autonome Fahren eines Fahrzeugs.

Unsupervised Learning und Reinforcement Learning kommen in dieser Arbeit nicht zum Einsatz, weshalb hier nur die grundsätzlichen Definitionen angegeben sind.

1.3 Zielsetzung

Das Ziel dieser Arbeit ist es, drei Modelle des maschinellen Lernens für die Identifikation des Raucherstatus anhand von Auszügen aus Arztbriefen zu trainieren. Der Datensatz für dieses Training wurde vom Institut für Medizinische Informatik, Statistik und Dokumentation der Medizinischen Universität Graz in de-identifizierter Form zur Verfügung gestellt. Die drei Modelle sind eine SVM, ein KNN auf Basis eines Feedforward neuronalen Netzes und ein RNN anhand einer LSTM-Architektur, welche mit Keras [22] und Scikit-learn [23] erstellt wurden.

1.4 Gliederung

Die Arbeit gliedert sich wie folgt: Kapitel 1 beschreibt die Motivation und Relevanz für die Arbeit und enthält zudem auch die Grundlagen des ML.

Kapitel 2 schildert die Methodik und die statistische Auswertung. Die Erläuterung der verwendeten Modelle, statistischen Maße und die Beschreibung des Datensatzes sind in diesem Kapitel zu finden.

Kapitel 3 ist der Auswertung gewidmet. Die Ergebnisse der statistischen Auswertung werden hier dargestellt.

Kapitel 4 interpretiert und diskutiert die Ergebnisse und vergleicht diese mit verwandten Arbeiten. Zudem werden Limitationen und Fehler der Arbeit erörtert. Ein Ausblick für die weitere Verwendung findet sich ebenfalls in diesem Kapitel.

Kapitel 2

Methoden und Daten

Für die vorliegende Arbeit wurden drei unterschiedliche Methoden des maschinellen Lernens miteinander verglichen: Eine SVM, die als Baseline verwendet wurde. Ein KNN mit einem Feedforward Aufbau und ein Sequenzmodell modelliert als LSTM-Architektur als Beispiel eines RNN. Diese Modelle sind geläufig in der Verarbeitung natürlicher textueller Sprache, unterscheiden sich jedoch grundsätzlich in ihrem methodischen Aufbau, weswegen sie in dieser Arbeit im Kontext der Problemstellung einem Vergleich unterzogen wurden.

Der Datensatz an sich wurde zuerst anhand der Interrater-Reliabilität (siehe Abschnitt Interrater Reliabilität) validiert. Dafür wurde eine zweite Fachperson beauftragt, die Klassifizierung für 20% der Daten vorzunehmen und die Übereinstimmung der beiden Annotatoren als Cohens Kappa κ (Gl. 2.7) berechnet. Auch eine Confusion-Matrix für die Ergebnisse der beiden Annotatoren wurde erstellt (siehe Abb. 3.1).

Für die maschinellen Modelle wurde der Datensatz iterativ in einen Trainingsdatensatz zum Optimieren des Modells und einen Testdatensatz zum Überprüfen des Erfolgs aufgeteilt. Diese Aufteilung erfolgte mittels Cross-Validation (siehe Kap. Cross-Validation auf Seite 21). Als Leistungskennzahlen wurden die im Kapitel Trainings- und Evaluierungsmethoden aufgelisteten Methoden verwendet. Die Ergebnisse sind in den Abschnitten 3.2, 3.3 und 3.4 zu finden.

2.1 Support Vector Machines

2.1.1 Hard Margin

SVMs sind ein Methodenspektrum zur Klassifikation von Daten, welche unter Supervised Learning zusammenzufassen sind. Diese Methodik wurde erstmals 1974 von Vapnik und Chervonenkis publiziert [24, 25].

Sei $X = (\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)$ ein Datensatz von n Datenpunkten (2-Tupel), wobei gilt, dass $\mathbf{x}_i \in \mathbb{R}^d$ mit $d \in \mathbb{R}$ den Datensatz i und y_i die Klasse des Datensatzes repräsentiert. Bei zwei Klassen beispielsweise kann y_i die Werte $\{1, -1\}$ annehmen. Nun lässt sich eine affine Hyperebene zwischen die beiden Klassen legen. Im zweidimensionalen ist diese Ebene eine Gerade. Allgemein hat sie die Dimension $d-1$, wobei d die Dimension der Eingabedaten ist. Die Hyperebene erfüllt folgende Gleichung

$$\langle \mathbf{w}^T, \mathbf{x}_i \rangle = \sum_{j=1}^{n} w_j x_{ij} = \beta$$

wobei $\mathbf{w}^T$ ein gewichteter Normalvektor zu $\mathbf{x}_i$ ist und $\beta \in \mathbb{R}$ ein Parameter.

Die trennende Hyperebene wird dabei so gelegt, dass ihr Abstand zu den Support Vektoren (deutsch: Stützvektoren, siehe Abb. 2.1) maximiert wird. Hat man einen zweidimensionalen Datensatz, so stellt diese Gleichung eine Gerade dar, die beide Klassen trennt. So eine Hyperebene mit ihren Support Vektoren ist in Abb. 2.1 grafisch dargestellt. Die Parameter $\mathbf{w}^T$ und β definieren damit die Klassifizierungsabbildung

$$\mathbf{x}_i \longmapsto \text{sgn}(\langle \mathbf{w}^T, \mathbf{x}_i \rangle - \beta)$$

wobei die Funktion $\text{sgn}(x)$ das Vorzeichen des Arguments x zurückgibt.

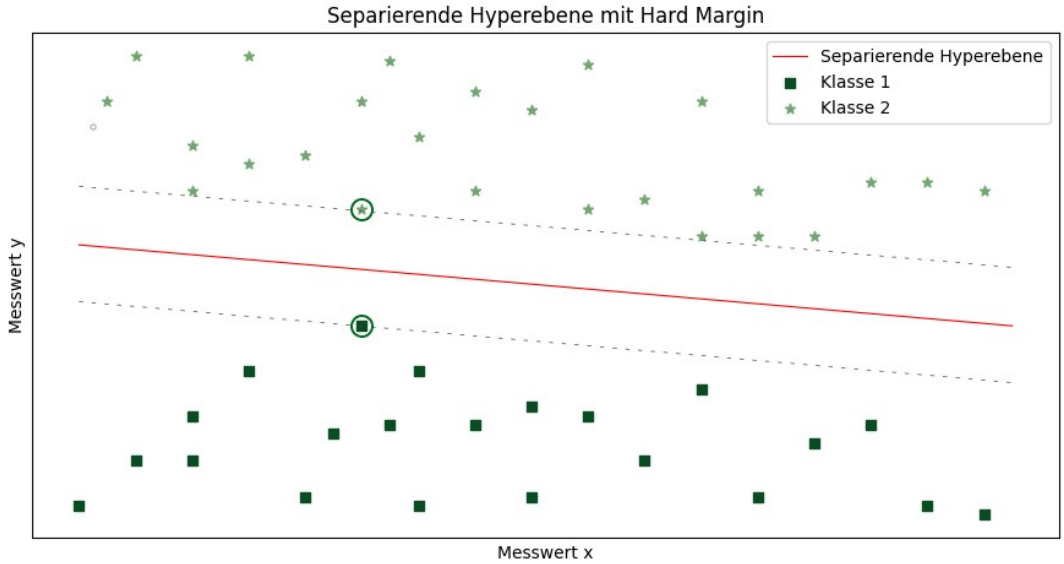

Abbildung 2.1: Separierende Hyperebene in Rot dargestellt. Jeder Datenpunkt (x, y) über der roten Linie wird als Klasse 1 (Stern) klassifiziert, jeder Datenpunkt darunter wird als Klasse 2 (Quadrat) erkannt. Dabei versucht die SVM die trennende Gerade mit maximalem Abstand zu den Stützvektoren (Kreis) zu legen. Im dargestellten Beispiel handelt es sich um linear separierbare Daten, da kein Stern unter der Hyperebene und kein Quadrat über der Hyperebene steht. Folglich lässt sich ein Hard Margin realisieren [21, 26].

Bei linear separierbaren Daten (wie in Abb. 2.1) lässt sich das anschaulich umsetzen. Das nennt man einen Hard Margin. "Linear separabel" nennt man zwei Mengen $M_1 \subset \mathbb{R}^n$ und $M_2 \subset \mathbb{R}^n$, wenn reelle Zahlen $w_1, ..., w_n, \beta$ existieren, mit

$$\sum_{i=1}^{n} w_i x_i > \beta \text{ für alle } \mathbf{x} \in M_1 \quad \text{und} \quad \sum_{i=1}^{n} w_i x_i \leq \beta \text{ für alle } \mathbf{x} \in M_2$$

Dabei nennt man den Parameter β die Schwelle [26, 27].

2.1.2 Soft Margin

Kommen Ausreißer im Datensatz vor, so sind die Daten nicht mehr linear separierbar und ein Hard Margin ist nicht mehr möglich. Es müssen Fehler erlaubt werden. Hier benötigt man nun einen Soft Margin, welcher von Cortes und Vapnik eingeführt wurde [28].

Beim Soft Margin werden Fehlannotationen erlaubt, diese jedoch mittels Hinge Loss Funktional minimiert. Dabei werden alle falsch klassifizierten Datenpunkte mit einem Fehler versehen, welcher größer ausfällt, je weiter entfernt der Datenpunkt von der Hyperebene liegt. Die Summe dieser Werte wird minimiert, was letztendlich den Gesamtfehler minimiert:

$$\min \frac{1}{n} \sum_{i=1}^{N} \left(\langle \mathbf{w}^T, \mathbf{x}_i \rangle - \beta \right)$$

wobei N die Anzahl an fehlerhaften Annotationen ist. Grafisch dargestellt ist das Prinzip in Abb. 2.2.

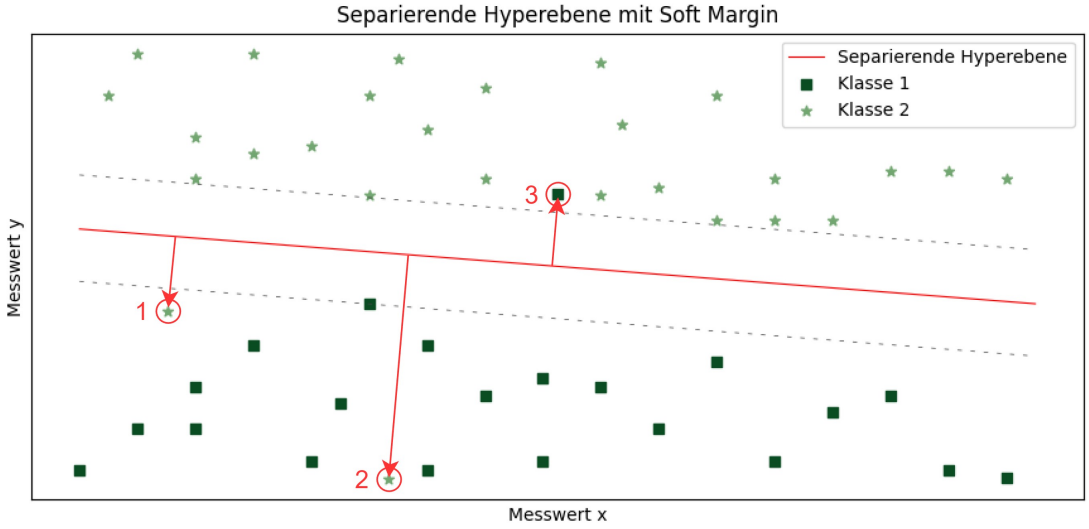

Abbildung 2.2: Prinzip eines Soft Margins: Die rot eingekreisten Datenpunkte wurden falsch klassifiziert. Dabei trägt Datenpunkt 2 stärker zur Fehlersumme bei als die Datenpunkte 1 und 3.

2.2 Neuronale Netze

Neuronale Netze sind mathematische Strukturen, welche im Rahmen des Machine Learning versuchen, die Funktionsweise eukaryotischer Nervensysteme nachzuahmen. Die Bausteine sind die namensgebenden Neuronen.

2.2.1 Biologische Neuronen

Das Nervensystem von Eukaryoten ist eine Ansammlung von Nervenzellen, von denen die Neuronen die wichtigsten Vertreter sind. Ein Neuron ist die kleinste funktionelle Einheit des Nervengewebes. Es handelt sich dabei um Nervenzellen, welche aus einem Zellkörper (Soma oder Perikaryon), Dendriten (eingehende Leitungsbahnen) und einem Axon (ausgehende Leitungsbahn) besteht. Ein solches Neuron ist in Abb. 2.3 dargestellt.

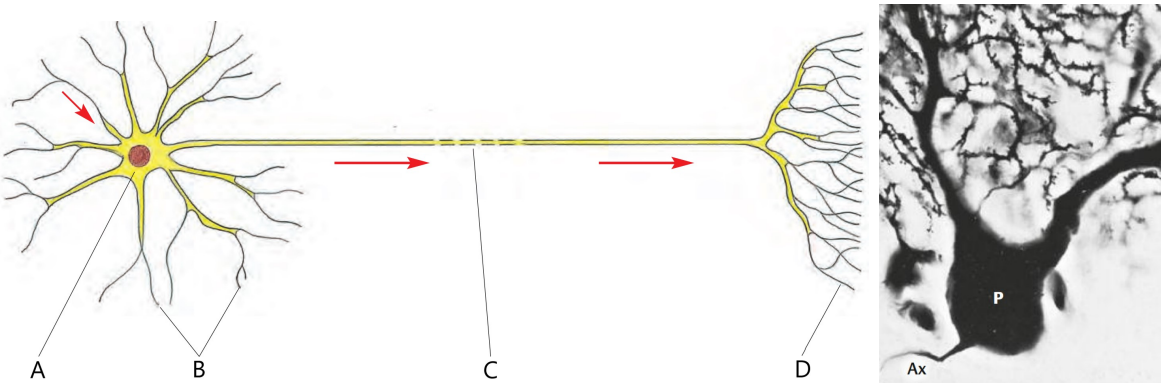

Abbildung 2.3:
Links: Schema eines Neurons. A: Perikaryon (Zellkörper), B: Dendriten, C: Axon (Länge: weniger als 1 mm bis mehr als 1 m), D: Endäste des Axons, rote Pfeile: Richtung der Signalweiterleitung. Die Endäste des Axons enden auf den Dendriten oder dem Zellkörper anderer Neuronen oder anderer Zelltypen, wie Muskel oder Drüsenzellen [29]. Rechts: Reales Neuron. P: Perikaryon, Ax: Axon [30]

Die Dendriten fungieren hierbei als Rezeptoren, welche eingehende Signale anderer Neuronen an den Zellkörper übergeben. Diese Signale kommen über die Endäste der Axone der jeweils vorgeschalteten Neurone an die Dendriten. Die Schnittstelle von Axon auf Dendrit nennt man Synapse. Das aufgenommene Signal wird im Perikaryon verarbeitet und das Ergebnis über das eigene Axon weiterleitet zum nächsten Neuron. Axone stellen also das projizierende Element der Neuronen dar. Sie sind dafür mit Scheiden aus Myelin ummantelt, welche für eine schnellere Weiterleitung sorgen. Bekommt so ein Dendrit über die Synapse einen Impuls, der exzitatorisch genug ist, so depolarisiert das Neuron und initiiert ein Aktionspotential. Dieses Aktionspotential läuft das Axon entlang und führt wiederum zu einer Transmitterfreisetzung an der nächsten Synapse. Schaltet man genügend dieser funktionellen Einheiten zu einem großen Netzwerk zusammen, so erhält man die Basis für das menschliche Nervensystem [31, 32].

2.2.2 Künstliche Neuronen

Um die Funktion eines biologischen Neurons mathematisch zu modellieren, haben McCulloch und Pitts 1943 die McCulloch-Pitts-Zelle entworfen [33]. Dabei handelt es sich um ein mathematisches Modell, welches nur binäre Signale aufnehmen und ausgeben konnte.

Heutzutage bedient man sich des Modells, welches in Abb. 2.4 schematisch dargestellt und in Gl. 2.1 mathematisch formalisiert ist. So ein zweilagiges Modell wird "Perzeptron"genannt, welches häufig Synonym mit dem Begriff "künstliches Neuron"verwendet wird. Dieses kann nicht mehr nur binäre Eingaben verarbeiten. Damit allein lassen sich lineare Klassifikationsaufgaben für eine Differenzierung zwischen zwei Klassen lösen [27, 34].

Ein Perzeptron ist eine Abbildung mit der Vorschrift

$$P(x) = \begin{cases} 1 & \text{falls } \sum_{i=0}^{n} w_i \cdot x_i + b > 0 \\ 0 & \text{sonst} \end{cases} \tag{2.1}$$

wobei hier $\mathbf{w} = (w_1, ..., w_n) \in \mathbb{R}^n$ ein Gewichtsvektor[1], $\mathbf{x} = (x_1, ..., x_n) \in \mathbb{R}^n$ ein Eingabevektor und b die Schwelle sind. Diese Abbildung stellt wieder eine Hyperebene wie in Kap. 2.1 dar.

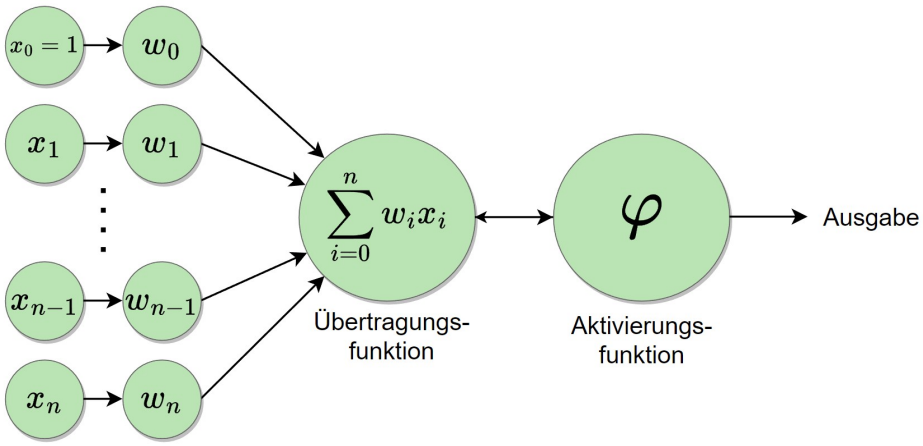

Abbildung 2.4: Schema eines künstlichen Neurons, genannt Perzeptron. Die x_i stellen hierbei die Komponenten des Eingabevektors $\mathbf{x}$ dar, w_i die Gewichtungen. Die Übertragungsfunktion ist das Skalarprodukt der beiden, dessen Ergebnis als Eingabewert für die Aktivierungsfunktion φ dient [35].

[1] Die Gewichtsvektoren sind gewöhnlicherweise in einer Matrix zusammengefasst, wodurch sich zwei Indizes ergeben. Aufgrund von besserer Lesbarkeit wird auf diese Notation verzichtet und stets jeweils nur ein Gewichtsvektor der Matrix $\mathbf{w}_i = (w_{i1}, ..., w_{in})$ als $\mathbf{w} = (w_1, ..., w_n)$ bezeichnet.

Die Eingabekomponenten x_i des Eingabevektors $\mathbf{x}$ sind hierbei analog zum eingehenden Signal über die Dendriten des biologischen Neurons. Jedoch wird vor der Verarbeitung des Signals jede Komponente mit w_i gewichtet und erst als Übertragungsfunktion

$$\langle \mathbf{w}, \mathbf{x} \rangle = \sum_{i=0}^{n} w_i \cdot x_i + b \tag{2.2}$$

an die Aktivierungsfunktion φ übergeben. Das b ist der Schwellwert.

Wie in der Übertragungsfunktion (Gl. 2.2) zu erkennen ist, beginnt der Index i mit 0. Als x_0 wird hierbei die 1 definiert. Die Gewichte w_i modellieren die Stärken der jeweiligen Verbindung. Diese werden während dem Training adaptiert, um das Ergebnis zu optimieren.

Die Aktivierungsfunktion φ bekommt als Argument den Wert der Übertragungsfunktion und ermittelt daraus die "Stärke"der Aktivierung des Perzeptrons.

$$\varphi = \varphi\left(\langle \mathbf{w}, \mathbf{x} \rangle\right) \tag{2.3}$$

Als Aktivierungsfunktion lassen sich diverse Funktionen verwenden. Zu den häufigsten gehören ReLU (Rectified linear unit), die Sigmoidfunktion, der Tangens Hyperbolicus und die Stufenfunktion. Diese sind in Abb. 2.5 dargestellt.

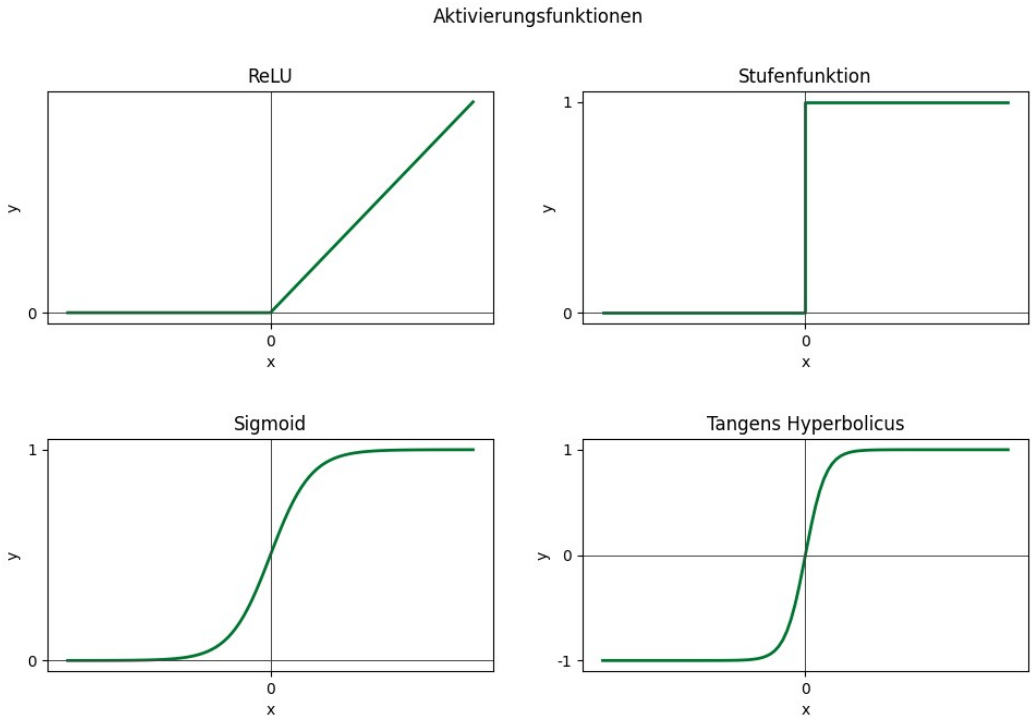

Abbildung 2.5: Vier häufige Beispiele für Aktivierungsfunktionen.

2.2.3 Neuronale Netzwerke

Nutzt man mehrere Layer (deutsch: Schichten) von Neuronen hintereinander, so erhält man ein künstliches neuronales Netzwerk (KNN). So ein Modell ist in Abb. 2.6 schematisch dargestellt.

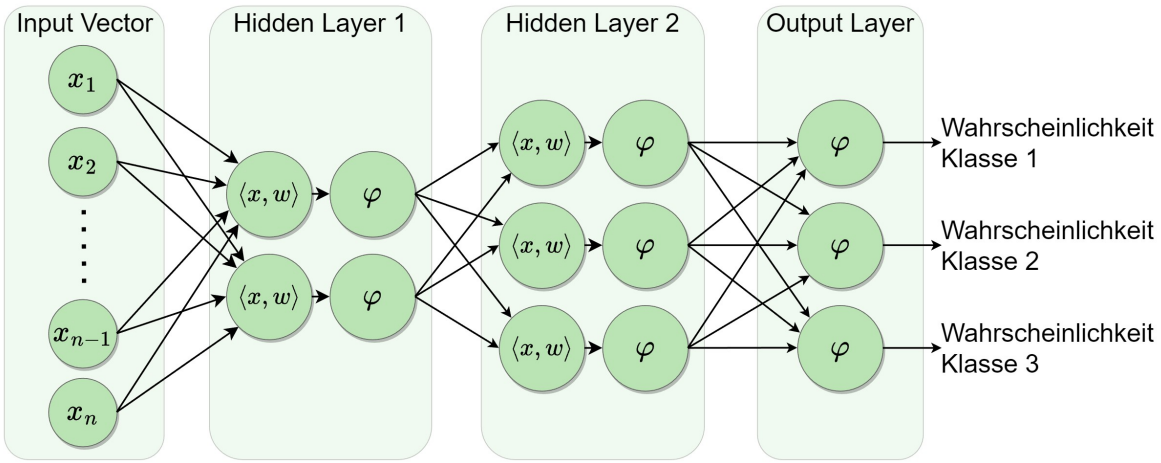

Abbildung 2.6: Tiefes neuronales Netzwerk mit 2 Hidden Layers (deutsch: verborgene Schichten). Der Vektor $\mathbf{x}$ enthält die Eingabedaten, der Output Layer gibt die Wahrscheinlichkeit für die Klassen aus. Das Skalarprodukt $\langle \mathbf{x}, \mathbf{w} \rangle$ wird pro Layer an die Aktivierungsfunktion φ gegeben und damit in jedem Hidden Layer erneut gewichtet.

Dabei werden nach jeder Aktivierungsfunktion die Ausgabewerte derjenigen an den nächsten Hidden Layer weitergegeben, der wiederum eigene Gewichte w und eine eigene Aktivierungsfunktion φ besitzt. Bei mehreren Layern nennt man das Netzwerk ein Deep Neural Network (deutsch: tiefes neuronales Netzwerk) [21].

Kommt es dabei von einer Schicht zur nächsten nicht zur Rückkopplung, so bezeichnet man das Modell als Feedforward Neural Network (deutsch: Feedforward KNN). Ist jedoch eine Rückkopplung vorhanden, so wird das mit dem Begriff Recurrent Neural Network (deutsch: rekurrentes neuronales Netz (RNN)) bezeichnet (siehe Kap. 2.2.4). Wenn das neuronale Netzwerk mit Faltungen (englisch: Convolution) arbeitet, so nennt man es Convolutional Neural Network (CNN).

Gradientenabstieg

Die Lernfähigkeit erhält ein KNN mithilfe des Gradient Descent (deutsch: Gradientenabstieg). Dabei handelt es sich um eine numerische Optimierungsmethode, bei welcher eine Loss Function (deutsch: Fehlerfunktion) iterativ minimiert wird. Beim Supervised Learning nutzt man als Gradientenverfahren die Backpropagation (deutsch: Fehlerrückführung). Ein Beispiel für eine Loss Function F wäre die euklidische Norm

$$F(\mathbf{y}, \mathbf{y}') = \sqrt{\sum_{i=1}^{n} (y_i - y_i')^2} \tag{2.4}$$

wobei $\mathbf{y}$ der vom Modell berechnete Wert und $\mathbf{y}'$ der richtige Wert (also das Label im Datensatz) darstellt [27]. Das Ergebnis des Gradientenabstiegs

$$-\nabla F(\mathbf{y}, \mathbf{y}')$$

dient dazu, die Gewichtsvektoren $\mathbf{w}$ zu aktualisieren. Da der Gradient die Richtung des steilsten Anstieges darstellt, nutzt man das negative Vorzeichen, um einen "Abstieg" zum Fehlerminimum zu erreichen. Dadurch reduziert man schrittweise den Fehler. Folglich "lernt" das Netzwerk hinzu.

In dieser Arbeit wird als Loss Function die Cross Entropy C (deutsch: Kreuzentropie) verwendet:

$$C = -\sum_{i=1}^{N} p(i) \cdot \log(p(j)) \tag{2.5}$$

wobei N die Anzahl an Klassen, $p(x)$ die Wahrscheinlichkeit von x, i die richtigen Labels und j die vorhergesagten Labels darstellen.

2.2.4 Long Short-Term Memory

In Modellen, mit denen Sprache verarbeitet werden soll, werden üblicherweise neuronale Netze mit Rückkopplung (RNN) verwendet. Ein Sonderfall stellt hierbei das Long Short-Term Memory-Netz (LSTM) dar, welches sogenannte LSTM-Zellen verwendet, um ein Kurzzeitgedächtnis zu simulieren.

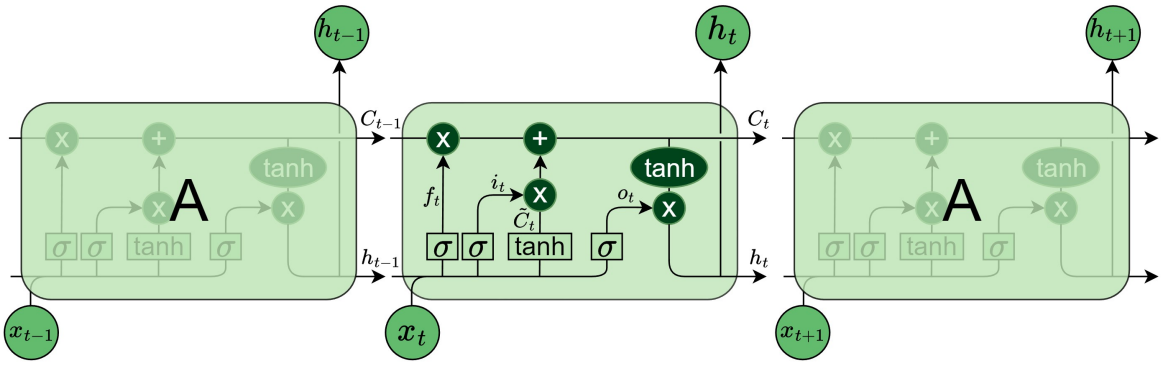

Abbildung 2.7: Drei LSTM-Zellen A mit den Inputvektoren x_i und den Ausgabevektoren h_i [36].

So eine LSTM-Zelle ist in Abb. 2.7 dargestellt. Die obere Linie symbolisiert den "Status" der Zelle, anfangs bezeichnet mit C_{t-1} und nach der Zelle aktualisiert auf C_t. Der Index $t-1$ bezeichnet hierbei den Zustand der vorhergehenden Zelle. Dieser Status wird von der unteren Linie verändert. Diese beinhaltet als Input den aktuellen Vektor x_t und einen Vektor aus der vorherigen Zelle h_{t-1}. Die erste Sigmoid-Funktion (f_t) nimmt diese beiden Eingaben und entscheidet, ob h_{t-1} vergessen werden soll (forget layer). Die Sigmoid-Funktion realisiert das durch eine Ausgabe im Intervall $[0, 1]$ (siehe Abb. 2.5).

Die zwei nächsten Zugänge (i_t und $\tilde{C}_t$) entscheiden darüber, welche neue Information in den Zustand einfließt. Dabei erzeugt eine tanh-Funktion mögliche Werte für den neuen Zustand $\tilde{C}$. Der letzte Zugang (o_t) entscheidet darüber, wie der neue Output h_t aussehen soll. Dieser fließt einerseits in den neuen Zustand mit ein, andererseits wird er auch als eigener Ausgabevektor an die nächste LSTM-Zelle weitergegeben [36–38].

2.3 Natural Language Processing

Unter Natural Language Processing (NLP, deutsch: Natürliche Sprachverarbeitung) versteht man jenen Bereich des Machine Learning, welcher sich mit Verarbeitung von natürlicher Sprache befasst.

2.3.1 Tokenization und Vectorizer

Um Sprache mit mathematischen Methoden aus der linearen Algebra verarbeiten zu können, muss diese zuerst in Vektorform dargestellt werden. Dieser Vorgang umfasst zwei Teilschritte:

Im ersten Schritt wird der Text beispielsweise in Sub-Wörter, Wörter oder kurzen Phrasen zerlegt, wobei dieser Vorgang üblicherweise **Tokenization** (deutsch: Tokenisierung) genannt wird. Danach wird ein Index auf die eindeutigen Tokens erstellt und ist somit vergleichbar mit einem indexierten Vokabular des Datensatzes auf Basis der Tokenisierung.

Der zweite Schritt besteht darin, den vorhandenen Text anhand von Vektoren darzustellen, welche anstatt der Wörter und Phrasen, die im ersten Schritt erstellten Tokens verwenden. Diesen Vorgang nennt man **Vectorization** (deutsch: Vektorisierung) [39]. Algorithmen, welche diese Aufgabe übernehmen, nennt man Vectorizer. Die Dimension der so erzeugten Vektoren entspricht dabei genau der Anzahl an erzeugten eindeutigen Tokens.

Beispielsweise könnte man mit den Tokens {'Wissen':1, 'frei':2, 'ist':3} den Satz "Wissen ist frei"als folgenden Vektor darstellen: [1, 1, 1]. Der Satz "Wissen ist "hingegen würde durch den Vektor [1, 0, 1] dargestellt werden. Die 0 an der Stelle 2 symbolisiert das Fehlen des Wortes "frei ". Diesen Algorithmus nennt man One-Hot-Codierung. Dabei codiert man das Vorkommen eines Wortes mit 1 und das Fehlen mit 0. Es gibt jedoch auch andere Vectorizer [21, 40].

	Wissen	frei	ist
"Wissen ist frei "	1	1	1
"Wissen ist "	1	0	1

Tabelle 2.1: Tabellarische Darstellung der beiden Vektoren in Zeilenform. Die Tokens sind hierbei {'Wissen':1, 'frei':2, 'ist':3 } in dieser Reihenfolge (siehe obere Zeile).

Der in dieser Arbeit verwendete Vectorizer ist der TF-IDF-Vectorizer (Term Frequency-Inverse Document Frequency). Die Term Frequency tf des Wortes T ist für den vewendeten TF-IDF-Vectorizer von Scikit-learn schlicht die Häufigkeit des Vorkommens von Wort T[1]. Die Inverse Document Frequency idf(T) des Wortes T wird wie folgt berechnet

$$idf(T) = \begin{cases} \log\left(\frac{1+N}{1+df(T)}\right) + 1 & \text{falls smooth\_idf=true} \\ \log\left(\frac{N}{df(T)}\right) + 1 & \text{falls smooth\_idf=false} \end{cases}$$

wobei N die Anzahl an Dokumenten darstellt und $df(T)$ die Anzahl an Dokumenten, welche T enthalten. Insgesamt erhält man die TF-IDF des Wortes T durch Multiplikation von tf und idf. Siehe Gl. 2.6 [40–42].

$$tfidf(T) = tf(T) \cdot idf(T) \tag{2.6}$$

2.3.2 Embeddings

Man kann Wörter nicht nur auf diskrete Zahlen eines Vokabulars abbilden, sondern auch auf einen kontinuierlichen Vektorraum $\mathbb{R}^n$. Diesen Vorgang nennt man Embedding (deutsch: Einbettung) und er lässt sich auf verschiedene Arten umsetzen. Der Vorteil davon ist, dass Wörter mit ähnlicher Bedeutung bei richtigem Vorgehen nahe beieinander liegen. Diese Nähe lässt sich mit der Metrik des Vektorraums (beispielsweise der Euklidischen Norm im $\mathbb{R}^n$) berechnen. Synonyme werden also idealerweise mit dem gleichen Vektor repräsentiert. Es existieren verschiedene Formen des Embeddings für Wörter. Das erste Verfahren wurde von Mikolov et al. im Jahr 2013 publiziert und nennt sich **word2vec**. Dabei wurde ein neuronales Netz trainiert, die Wörter anhand der vorgehenden und der nachfolgenden Wörter im Vektorraum zu platzieren. Die Wahrscheinlichkeit, dass das gegebene Wort genau an einem bestimmten Ort im Vektorraum dargestellt wird, wird maximiert. Dadurch wurde erreicht, dass Synonyme tatsächlich nahe beieinander landeten und semantisch ähnlich Wörter wie beispielsweise “Frankreich”, “Italien” und “Spanien” in Clustern gruppiert waren [21, 43]. Heute gebräuchliche kontextualisierte Embeddingrepresäntationen können auch Homonyme voneinander unterscheiden [44].

[1] Es gibt verschiedene Möglichkeiten, die tf und idf zu berechnen. Scikit-learn verwendet die in diesem Kapitel erläuterten.

2.4 Trainings- und Evaluierungsmethoden

2.4.1 Interrater Reliabilität

Um die Reliabilität der Labels eines Datensatzes zu überprüfen, kann man eine zweite Person den Datensatz klassifizieren lassen und die Übereinstimmung dieser beiden von Mensch durchgeführten Klassifizierungen mittels Cohens Kappa κ (Gl. 2.7) ermitteln.

$$\kappa = \frac{p_{obs} - p_{exp}}{1 - p_{exp}} \tag{2.7}$$

hierbei ist p_{obs} der prozentual gemessene Übereinstimmungswert der beiden von Menschen (englisch: Rater) erstellen Labels, p_{exp} der Prozentsatz der erwarteten zufälligen Übereinstimmung. Diese hängt von der Anzahl an Klassen ab [45]. Ein Wert für $\kappa = 1$ bedeutet perfekte Übereinstimmung der Rater, bei $\kappa \leq 0$ wurde exakt der zufällige Übereinstimmungswert ($\kappa = 0$) oder sogar weniger ($\kappa < 0$) erreicht.

Fleiss und Chilton [46] schlagen vor, κ-Werte unter $0,4$ als schwach ("poor"), Werte im Bereich von $0,4 - 0,75$ als annehmbar bis gut ("fair to good") und $\kappa > 0,75$ als exzellent ("excellent") anzunehmen.

Bakeman und Gottman [45] erwähnen die Arbeit von Fleiss und Chilton, schlagen aber auch vor κ-Werte unter $0,7$ mit Skepsis zu betrachten ("Our own inclination, based on using kappa with a number of different coding schemes, is to regard kappas less than .7, even when significant, with some concern, but this is only an informal rule of thumb.") .

Landis und Koch [47] kategorisieren κ-Werte von $0 - 0,20$ als leichte ("slight") Übereinstimmung, $0,21 - 0,40$ breits als annehmbar ("fair"), $0,41 - 0,60$ als moderat ("moderate"), $0,61 - 0,80$ als substanziell ("substantial") und $\kappa > 0,80$ als beinahe perfekt ("almost perfect").

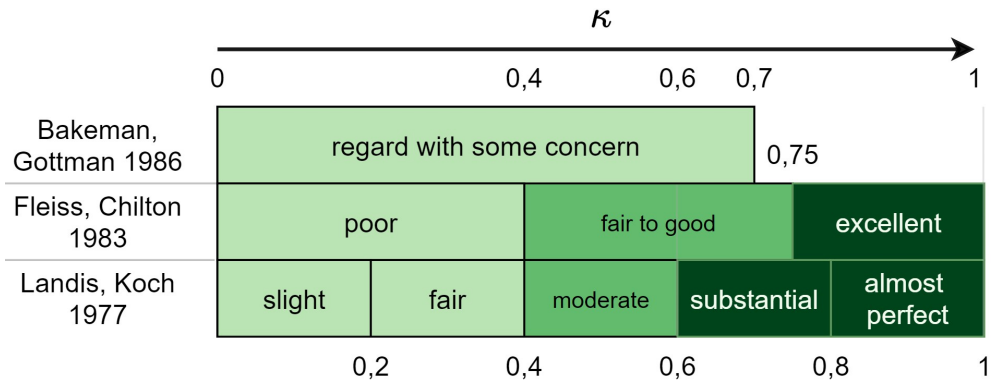

Abbildung 2.8: Interpretationen von κ-Werten verschiedener Publikationen [48].

2.4.2 Cross-Validation

Unter Cross-Validation (deutsch: Kreuzvalidierungsverfahren) versteht man ein statistisches Verfahren zur Validierung eines ML-Modells. Dabei geht man wie folgt vor: Bei einer n-fold Cross-Validation wird der Datensatz in n Teilmengen, sogenannte "folds", unterteilt. Dann wird jede dieser folds einmal als Testdatensatz verwendet, während die restlichen folds als Trainingsdatensatz fungieren. Folglich wird das Modell so n mal trainiert. Als Gesamtfehler wird dann der arithmetische Mittelwert der n einzelnen Fehler angegeben. Das Verfahren ist schematisch in Abb. 2.9 dargestellt.

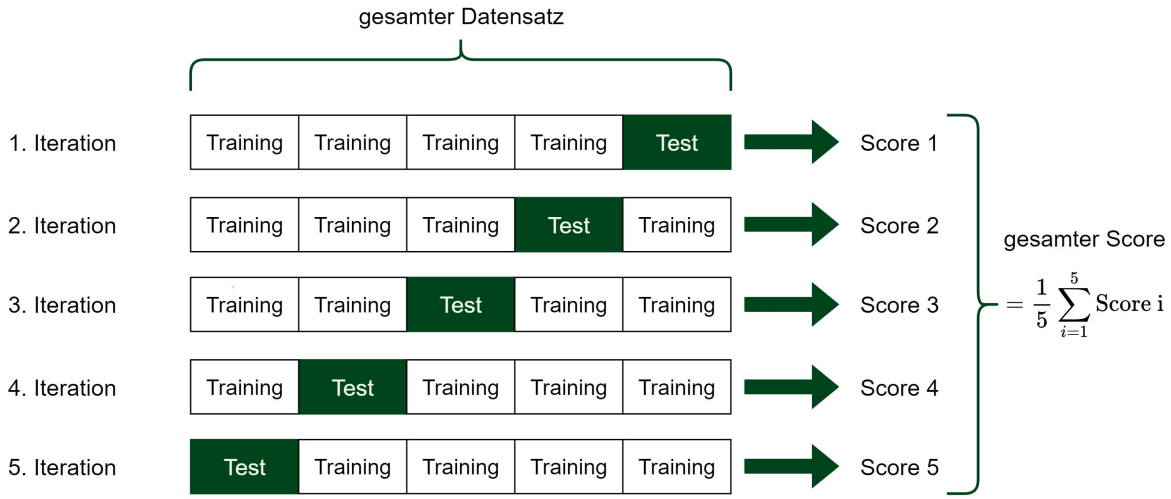

Abbildung 2.9: Schema für eine 5-fold Cross-Validation. Hierbei wird bei jeder Iteration ein anderes Fünftel des Datensatzes als Testdatensatz verwendet und ein eigener Score pro Iteration berechnet. Der gesamte Score berechnet sich dann aus dem Mittelwert der einzelnen Scores. [49].

2.4.3 Accuracy

Ein Score um ML-Modelle bewerten zu können ist die Accuracy A aus Gl. 2.8.

$$A = \frac{TP + TN}{TP + TN + FP + FN} = \frac{\text{True Calls}}{\sum \text{Calls}} \tag{2.8}$$

Hierbei stehen TP für True Positives, TN für True Negatives, FP für False Positives und FN für False Negatives. Die Accuracy beschreibt, wie viele der klassifizierten Messwerte (Calls) richtig klassifiziert wurden. Siehe Abb. 2.10.

2.4.4 Precision

Die Precision P (deutsch: Präzision, Genauigkeit) beschreibt, wie viele positive Werte auch richtige Werte sind. Anders gesagt beschreibt der Wert, wie viele als positiv klassifizierte Messwerte auch wirklich positiv sind. Die entsprechende Gl. 2.9 zeigt die Formel dafür und Abb. 2.10 stellt den Sachverhalt grafisch dar. Die Präzision wird in der Medizin auch "positive predictive value" genannt.

$$P = \frac{TP}{TP + FP} = \frac{TP}{\sum \text{Positive Calls}} \tag{2.9}$$

2.4.5 Recall

Der Recall R (deutsch: Ausbeute) beschreibt, wie viele der richtigen positiven Werte erkannt wurden. Ein hoher Anteil an falsch negativen Messwerten senkt den Recall, da diese eigentlich positiv sind, aber als negativ klassifiziert und daher nicht erkannt wurden. Der Recall wird in der Medizin häufig "Sensitivität" genannt. Siehe Gl. 2.10 und Abb. 2.10.

$$R = \frac{TP}{TP + FN} = \frac{TP}{\sum \text{Positives}} \tag{2.10}$$

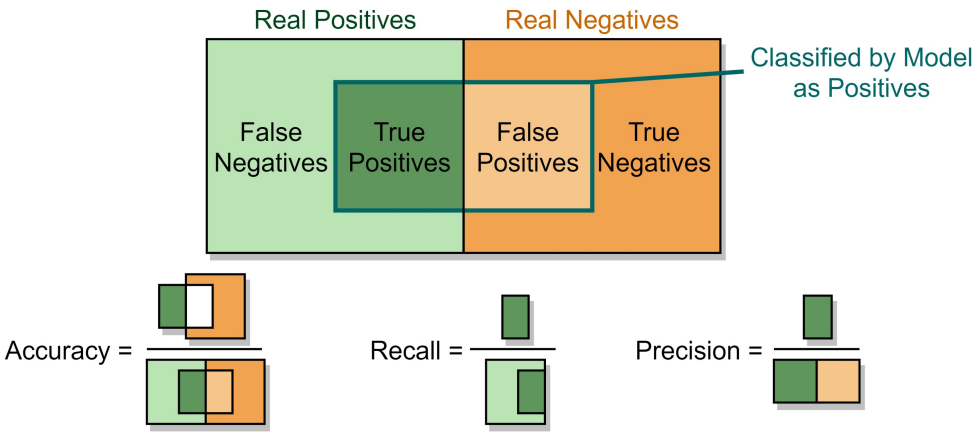

Abbildung 2.10: Grafische Erläuterung von Accuracy, Precision und Recall [50]. Dunklere Felder stehen für richtig klassifizierte, hellere für falsch klassifizierte Daten. Grüne Felder stehen für richtig positive und orange für richtig negative Daten. Accuracy: Wie viele wurden richtig klassifiziert? Recall: Wie viele der richtig Positiven wurden als solche erkannt? Precision: Wie viele der positiv Klassifizierten sind richtig positiv?

2.4.6 F_1-Score

Der F_1-Score ist das harmonische Mittelmaß von Precision P und Recall R für eine Klasse:

$$F_1 = 2\frac{P \cdot R}{P + R} = \frac{TP}{TP + \frac{1}{2}(FN + FP)} \tag{2.11}$$

Man kann nun für jede Klasse einzeln einen F_1-Score berechnen, aber auch zwei Arten von Mittelwerten daraus, einen Macro-Average F_1-Score und einen Micro-Average F_1-Score. Der Macro-Average Ansatz ist der arithmetische Mittelwert der F_1-Scores der einzelnen n Klassen.

$$F_{1-macro} = \frac{1}{n}\sum_{i=1}^{n} F_i \tag{2.12}$$

Beim Micro-Average Ansatz werden die TP, FP und FN der einzelnen Klassen addiert und daraus der F_1-Score berechnet:

$$F_{1-micro} = \frac{TP_{ges}}{TP_{ges} + \frac{1}{2}(FN_{ges} + FP_{ges})} \tag{2.13}$$

Hierbei gilt $TP_{ges} = \sum_{i=1}^{n} TP_i$. Analog werden auch FN_{ges} und FP_{ges} als Summen der jeweiligen Werte über alle Klassen berechnet.

2.4.7 Weighted Average

Der gewichtete Mittelwert $\bar{x}$ (englisch: Weighted Average) ist in Gl. 2.14 dargestellt. Dabei erhalten die Summanden F_i eine Gewichtung proportional zu ihrer Anzahl im Datensatz. Häufigere Klassen fließen dadurch stärker ins gewichtete Mittelmaß ein als seltenere Klassen.

$$\bar{x} = \sum_{i}^{n} \left(F_i \cdot \frac{s_i}{n} \right) \tag{2.14}$$

wobei s_i den Support (deutsch: Häufigkeit) für die i-te Klasse im Datensatz darstellt und n die Anzahl an Datenpunkten insgesamt.

2.4.8 Confusion-Matrix

Um die verwendeten Modelle zu vergleichen, kann man eine Confusion-Matrix (deutsch: Kreuztabelle) verwenden. Dabei trägt man die richtigen Messwerte auf eine Achse auf und die klassifizierten Werte auf die andere Achse. So eine Matrix ist in Abb. 2.11 dargestellt.

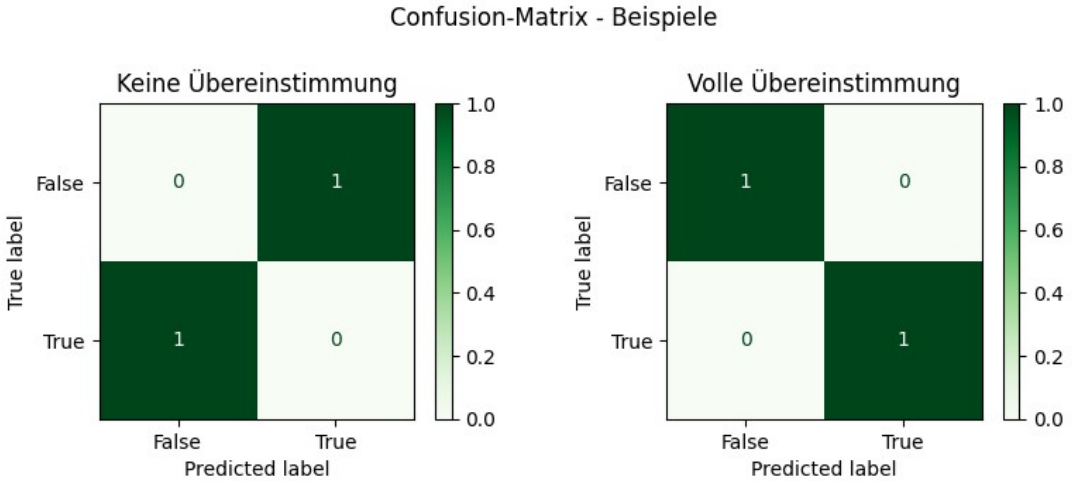

Abbildung 2.11: Beispiel für eine Confusion-Matrix. Links sieht man den Fall, dass die Klassifikation des Modells (Predicted label) in keinem Fall mit dem eigentlichen Label aus dem Datensatz (True label) übereinstimmt. Rechts hingegen stimmen die Klassifikationen des Datensatzes und des Modells überein.

2.5 Datensatz

Extrahiert wurden die Daten aus Texten der Kliniken für Kardiologie, Dermatologie und Onkologie, um verschiedene Fachgebiete in ihrem Sprachgebrauch abzudecken. Die Daten der Onkologie enthielten nur Patientinnen und Patienten mit Kolonkarzinomen. Der Datensatz enthält 7242 Textausschnitte aus de-identifizierten Arztbriefen. Jeder Textausschnitt hatte eine Länge von 200 Zeichen. Die Klassifikation erfolgte in eine von 6 Kategorien durch eine Fachperson. Diese sind "Past Smoker", "Current Smoker", "Current Non-Smoker", "Never Smoker", "Current or Past" und "Unknown". Eine zweite Fachperson hat daraufhin 20% des Datensatzes ebenso annotiert, um die Reliabilität der menschlichen Annotation zu überprüfen (siehe 2.4.1 und 3.1).

Die 6 Klassen wurden aufgrund der vorhandenen Textpassagen ausgewählt, um diese möglichst akkurat abbilden zu können. In Tab. 2.2 sind typische Beispiele für jede Klasse gelistet.

Klasse	Textausschnitt
PAST SMOKER	... Nikotin Abusus: Exraucher ...
CURRENT SMOKER	... chron. Nikotinabusus ...
CURRENT NON-SMOKER	... die Pat. ist Nichtraucherin ...
NEVER SMOKER	... Nikotinanamnese: neg ...
CURRENT OR PAST	... Nikotinell 21µg TTS ...
UNKNOWN	(keine Aussage zu Nikotinabusus im Text)

Tabelle 2.2: Typische Beispiele für die verwendeten Klassen. Die Klasse "Current or Past" beschreibt typischerweise eine Erwähnung von Nikotinell TTS Pflastern, wo jedoch keine weitere Präzisierung des Raucherstatus vorliegt.

2.5.1 SNOMED

Bei der Auswahl der Klassen besteht eine Ähnlichkeit zur Terminologie von SNOMED CT (Systematized Nomenclature of Medicine Clinical Terms) [51]. Eine mögliche Abbildung der verwendeten Klassen als SNOMED CT Konzepte ist in Tab. 2.3 dargestellt.

Klasse	SCT Vorzugsterm	SCT ID
PAST SMOKER	Ex-smoker	8517006
CURRENT SMOKER	Smoker	77176002
CURRENT NON-SMOKER	Non-smoker[1]	8392000
NEVER SMOKER	Never smoked tobacco	266919005
CURRENT OR PAST	Current or past smoker[2]	410511007, 77176002
UNKNOWN	Tobacco smoking consumption unknown	266927001

Tabelle 2.3: Abbildung der gewählten Klassen in der Terminologie von SNOMED CT.

2.5.2 Unausgewogenheit der Daten

Der Datensatz beinhaltet eine stark asymmetrische Verteilung der Klassenhäufigkeiten. Diese sind in Tab. 2.4 aufgelistet.

Klasse	absolute Häufigkeit	relative Häufigkeit
PAST SMOKER	2182	30,13 %
CURRENT SMOKER	4255	58,75 %
CURRENT NON-SMOKER	27	0,37 %
NEVER SMOKER	432	5,97 %
CURRENT OR PAST	85	1,17 %
UNKNOWN	261	3,60 %

Tabelle 2.4: Absolute und relative Häufigkeiten der einzelnen Klassen im Datensatz.

Durch diese Ungleichverteilung entstehen zwangsweise hohe Ungenauigkeitswerte bei den seltenen Klassen. Je nachdem, ob die wenigen vorhandenen Daten in den Trainings- oder den Testdatensatz fallen, fehlt es entweder an genügend Training für das Modell oder die Validierung für diese seltene Klasse ist unzureichend.

[1] "Current Non-Smoker" wird durch das SNOMED CT-Finding "Non-smoker" abgebildet, da dieses als Tochterklassen sowohl "Never smoked tobacco", als auch "Ex-Smoker" beinhaltet.

[2] Die Klasse "Current or Past" ist in der SNOMED CT kein eigenes Finding, sondern zusammengesetzt aus einem temporal context value und dem Finding "Smoker".

2.5.3 Synthetic Minority Over-sampling Technique

Um das Problem zu mindern, wurde der Datensatz für die SVM einmal mittels Oversampling extrapoliert. Dafür wurde die Methode SMOTE (Synthetic Minority Over-sampling Technique) aus der Bibliothek Imbalanced-learn verwendet. Dabei werden diejenigen Klassen, welche selten vorkommen, künstlich vermehrt, um eine Ausgewogenheit zu erreichen. Diese Methode wurde nur auf den Trainingsdatensatz verwendet, da ansonsten das Testergebnis verfälscht werden würde [52].

Kapitel 3

Ergebnisse und Auswertung

In diesem Kapitel werden die Ergebnisse der trainierten ML-Algorithmen statistisch ausgewertet. In Abschnitt 3.1 wurde initial die Interrater-Reliabilität berechnet. Der Abschnitt 3.2 wird die Leistung der SVM ermitteln. Hierfür wird einmal der Datensatz ohne Veränderung verwendet und zum Vergleich einmal mit SMOTE. Im Anschluss daran wird im Abschnitt 3.3 das Feedforward KNN trainiert und dessen Leistung ausgewertet. Hierbei wird kein SMOTE verwendet. Zuletzt wird in Abschnitt 3.4 die Performanz des KNN mit der LSTM-Architektur bewertet. Auch dieses wurde mit den Daten ohne SMOTE trainiert.

3.1 Interrater-Reliabilität

Die Auswertung der beiden Annotatoren ergab einen κ-Wert von $\kappa = 89,97\%$ bei einer Übereinstimmung $p_{obs} = 91,64\%$ und damit eine signifikante Übereinstimmung der beiden Personen nach den Einteilungen in Abb. 2.8. Der erste Annotator hat dabei den gesamten Datensatz, der zweite Annotator 20% der Daten klassifiziert (nach den Guidelines von O'Connor und Joffe [53]). Die klassenabhängige Übereinstimmung ist in Abb. 3.1 als Confusion-Matrix dargestellt.

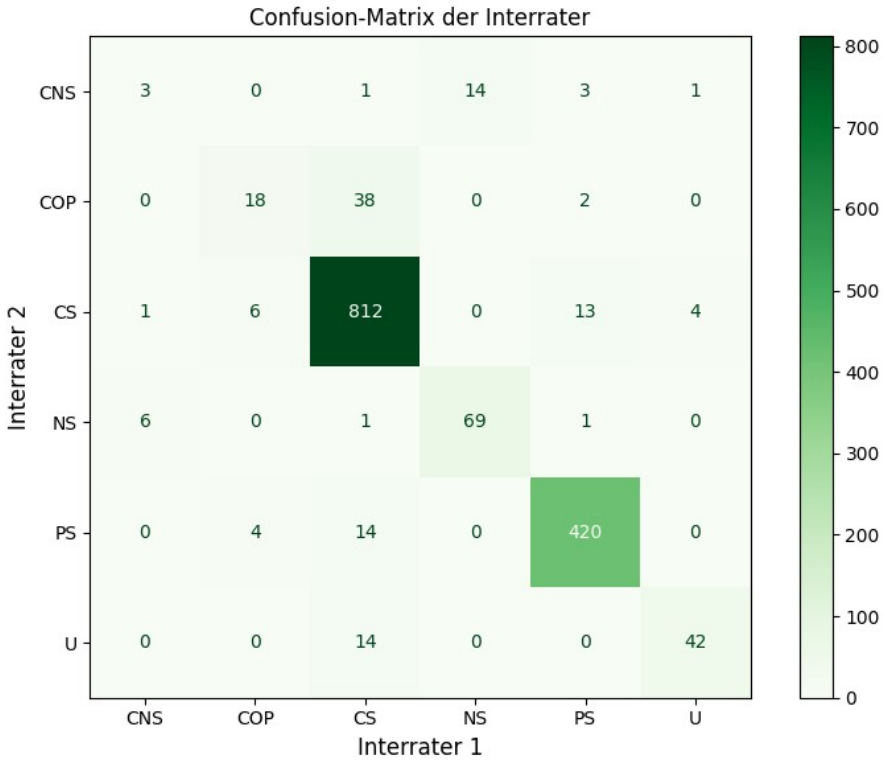

Abbildung 3.1: Die klassenabhängige Übereinstimmung der beiden Annotatoren als Confusion-Matrix. CNS: Current Non-Smoker, COP: Current or Past, CS: Current Smoker, NS: Never Smoker, PS: Past Smoker, U: Unknown.

3.2 Support Vector Machine

Die SVM wurde einmal ohne und einmal mit SMOTE trainiert. Die jeweiligen Confusion-Matrizen sind in den Abb. 3.2 und Abb. 3.3 dargestellt.

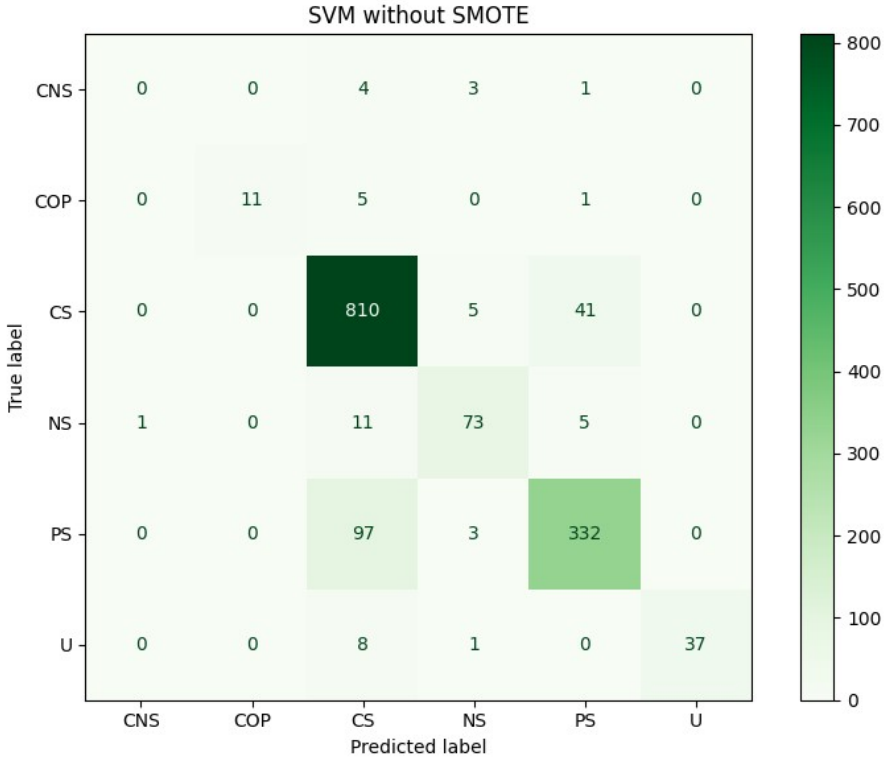

Abbildung 3.2: Confusion-Matrix der SVM ohne SMOTE. CNS: Current Non-Smoker, COP: Current or Past, CS: Current Smoker, NS: Never Smoker, PS: Past Smoker, U: Unknown.

Klasse	Precision	Recall	F_1-Score	Support
CURRENT NON-SMOKER	0.00	0.00	0.00	7
CURRENT OR PAST	1.00	0.41	0.58	22
CURRENT SMOKER	0.88	0.96	0.92	856
NEVER SMOKER	0.82	0.83	0.82	76
PAST SMOKER	0.91	0.79	0.85	432
UNKNOWN	1.00	0.88	0.93	56
Accuracy			0.89	1449
Macro Average	0.77	0.64	0.68	1449
Weighted Average	0.89	0.89	0.88	1449

Tabelle 3.1: Auswertung: SVM-Modell ohne SMOTE. F_1-Scores der einzelnen Klassen.

Abbildung 3.3: Confusion-Matrix der SVM mit SMOTE. CNS: Current Non-Smoker, COP: Current or Past, CS: Current Smoker, NS: Never Smoker, PS: Past Smoker, U: Unknown.

Der $F_{1-macro}$ des SVM-Modells mit SMOTE beträgt 83,00%, der $F_{1-weighted}$ 88,00%. Die F_1-Scores der einzelnen Klassen sind in Tab. 3.2 aufgelistet.

Klasse	Precision	Recall	F_1-Score	Support
CURRENT NON-SMOKER	1.00	0.50	0.67	4
CURRENT OR PAST	0.95	0.64	0.77	28
CURRENT SMOKER	0.91	0.90	0.90	878
NEVER SMOKER	0.81	0.94	0.87	88
PAST SMOKER	0.81	0.82	0.82	401
UNKNOWN	1.00	0.86	0.92	50
Accuracy			0.88	1449
Macro Average	0.91	0.78	0.83	1449
Weighted Average	0.88	0.88	0.88	1449

Tabelle 3.2: Auswertung: SVM-Modell mit SMOTE. F_1-Scores der einzelnen Klassen.

3.3 Neuronales Netz

Das Feedforward KNN wurde ohne SMOTE trainiert. Um die Daten zu vektorisieren, wurden Funktionen von Keras verwendet. Für die Tokenisierung wurde eine maximale Wortanzahl des Vokabulars auf 5000 gesetzt und die Dimensionalität der Embeddingvektoren auf 50.

Das Modell besteht aus einem Embedding Layer, einem GlobalAveragePooling1D-Layer, einem Hidden Layer (Dense Layer mit 256 Neuronen) und einem dense Output Layer mit 6 Neuronen (eines für jede Klasse). Als Aktivierungsfunktion diente die Sigmoid-Funktion (siehe Abb. 2.5) und als Loss Function diente die Categorical Cross Entropy (siehe Gl. 2.5). Alle Parameter können im Anhang A nachgelesen werden.

Das Training dauerte 36 Epochen lang (vorzeitiger Abbruch aufgrund von fehlender Verbesserung). Der Wert der Loss Function bei Epoche 36 betrug 0.36.

In Abb. 3.4 ist die Confusion-Matrix des Feedforward KNN.

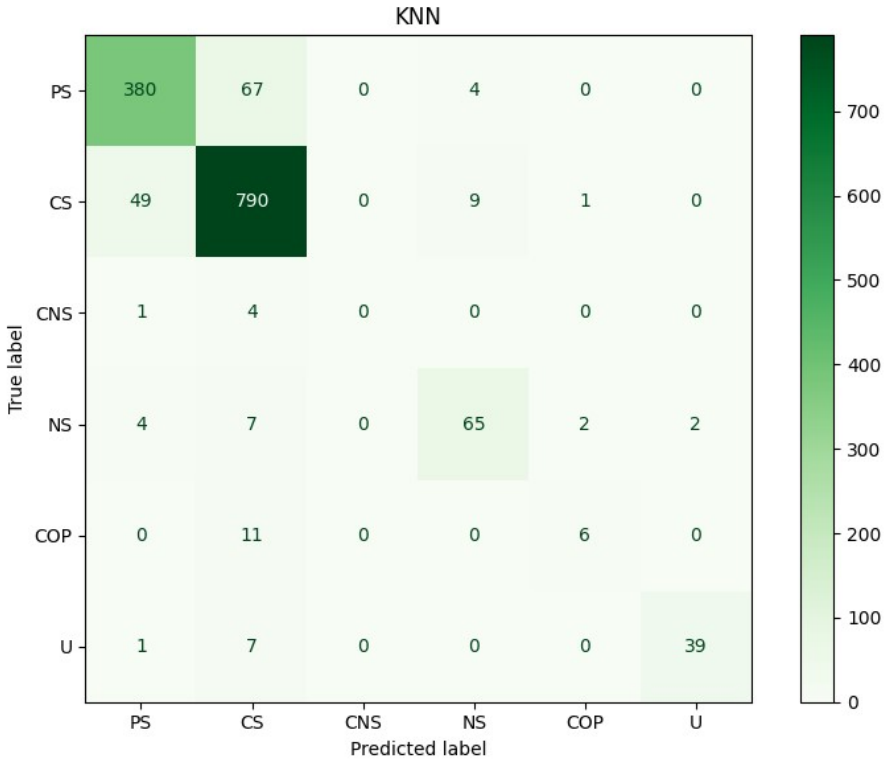

Abbildung 3.4: Confusion-Matrix des Feedforward KNN. CNS: Current Non-Smoker, COP: Current or Past, CS: Current Smoker, NS: Never Smoker, PS: Past Smoker, U: Unknown.

In Tab. 3.3 sind die Scores der einzelnen Klassen aufgelistet:

Klasse	Precision	Recall	F_1-Score	Support
CURRENT NON-SMOKER	0.00	0.00	0.00	5
CURRENT OR PAST	0.67	0.35	0.46	17
CURRENT SMOKER	0.89	0.93	0.91	849
NEVER SMOKER	0.83	0.81	0.82	80
PAST SMOKER	0.87	0.84	0.86	451
UNKNOWN	0.95	0.83	0.89	47
Accuracy			0.88	1449
Macro Average	0.70	0.63	0.66	1449
Weighted Average	0.88	0.88	0.88	1449

Tabelle 3.3: Auswertung: KNN-Modell. F_1-Scores der einzelnen Klassen.

Die Verläufe von Accuracy und Loss Function pro Epoche sind in den Abb. 3.5 und 3.6 dargestellt.

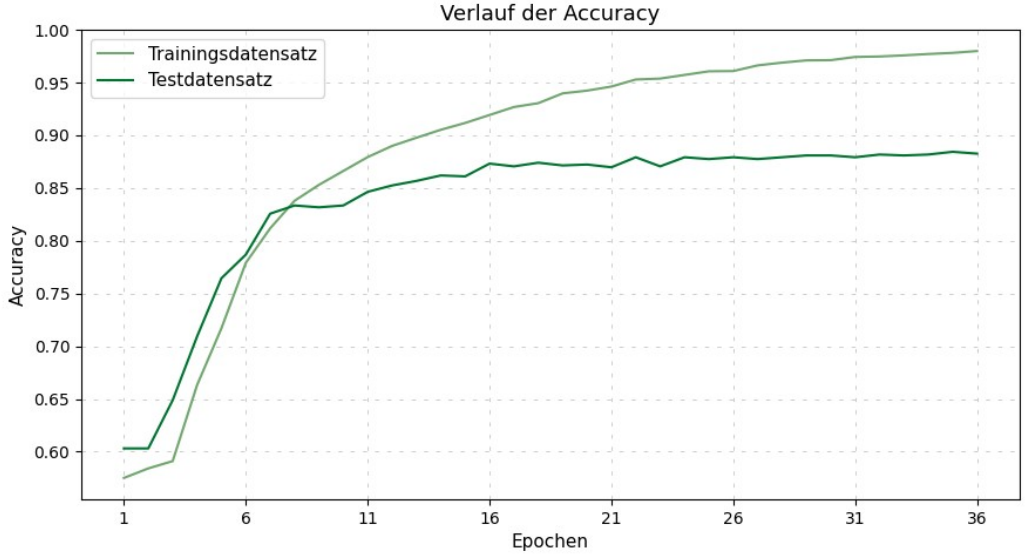

Abbildung 3.5: Verlauf der Accuracy des Feedforward KNN nach Epochen.

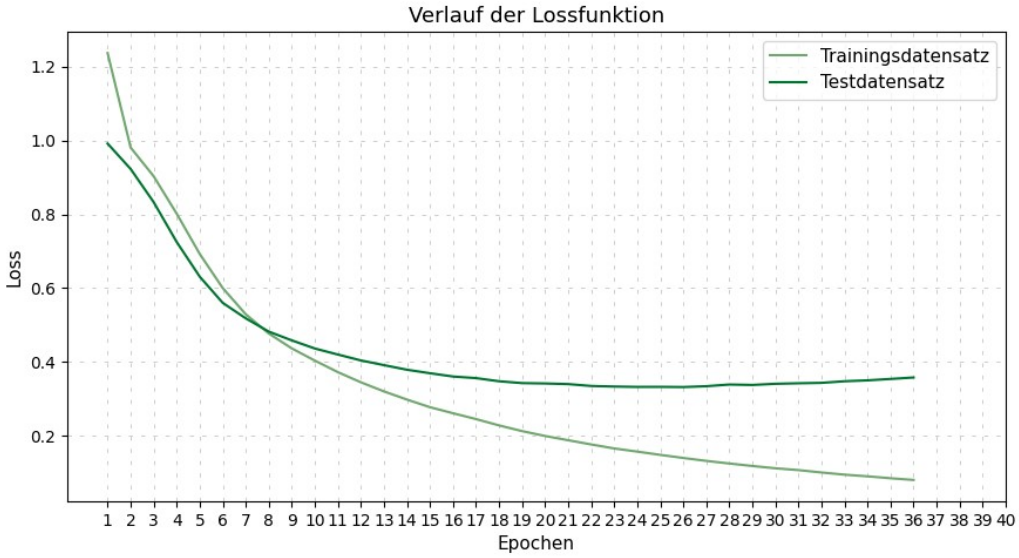

Abbildung 3.6: Verlauf der Loss Function des Feedforward KNN nach Epochen.

3.4 Long Short-Term Memory

Das neuronale Netz mit LSTM-Architektur wurde ebenso ohne SMOTE trainiert. Die Parameter für Tokenisierung, maximale Wortanzahl des Vokabulars und maximale Vektordimensionalität waren gleich wie beim Feedforward KNN. Auch Loss Function und Aktivierungsfunktion wurden nicht verändert.

Das Modell besteht aus einem Embedding-Layer, einem LSTM-Layer und ebenfalls einem Dense Layer mit 6 Neuronen (eines für jede Klasse). Die Quellcodes können im Anhang A nachgelesen werden.

Das Training dauerte 20 Epochen lang. Der Wert der Loss Function bei Epoche 20 betrug 0.30.

In Abb. 3.7 ist die Confusion-Matrix des KNN mit LSTM-Architektur dargestellt.

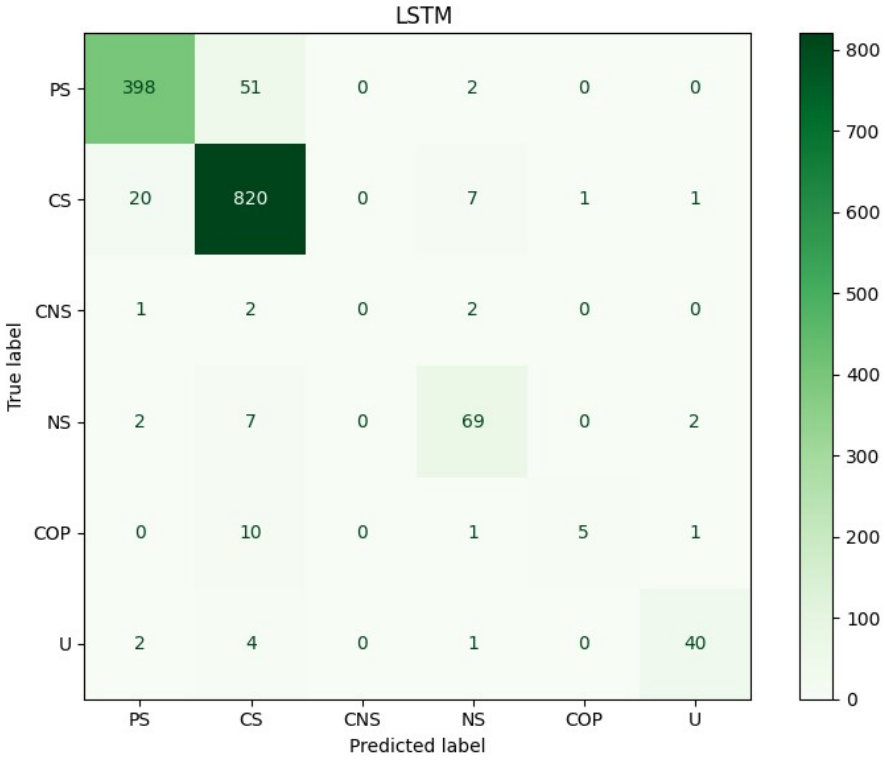

Abbildung 3.7: Confusion-Matrix des KNN mit LSTM-Architektur. CNS: Current Non-Smoker, COP: Current or Past, CS: Current Smoker, NS: Never Smoker, PS: Past Smoker, U: Unknown.

In Tab. 3.4 sind die Scores der einzelnen Klassen aufgelistet:

Klasse	Precision	Recall	F_1-Score	Support
CURRENT NON-SMOKER	0.00	0.00	0.00	5
CURRENT OR PAST	0.83	0.29	0.43	28
CURRENT SMOKER	0.92	0.97	0.94	849
NEVER SMOKER	0.84	0.86	0.85	88
PAST SMOKER	0.94	0.88	0.91	451
UNKNOWN	0.91	0.85	0.88	50
Accuracy			0.92	1449
Macro Average	0.74	0.64	0.67	1449
Weighted Average	0.92	0.92	0.92	1449

Tabelle 3.4: Auswertung: LSTM-Modell. F_1-Scores der einzelnen Klassen.

Die Verläufe von Accuracy und Loss Function pro Epoche sind in den Abb. 3.8 und 3.9 dargestellt.

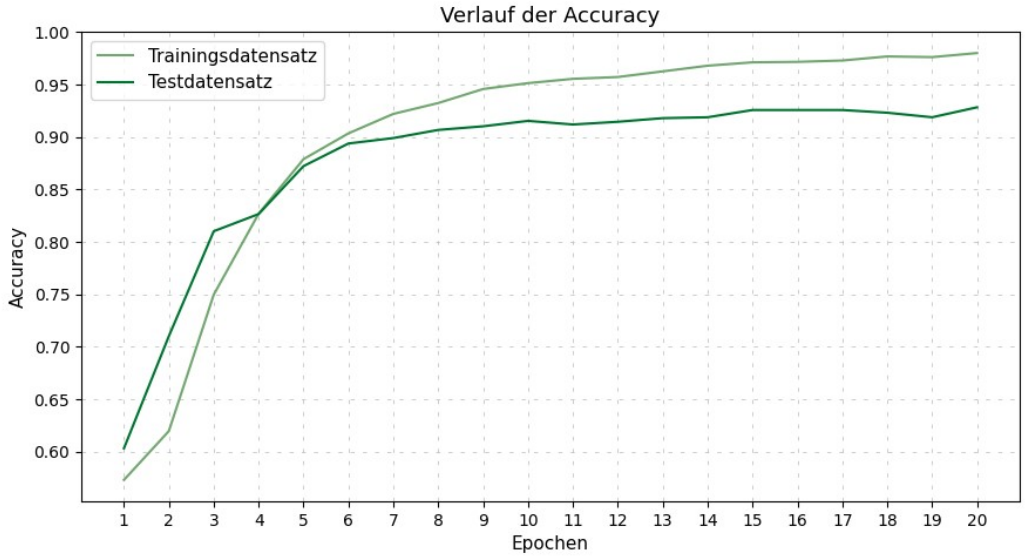

Abbildung 3.8: Verlauf der Accuracy des KNN mit LSTM-Architektur nach Epochen.

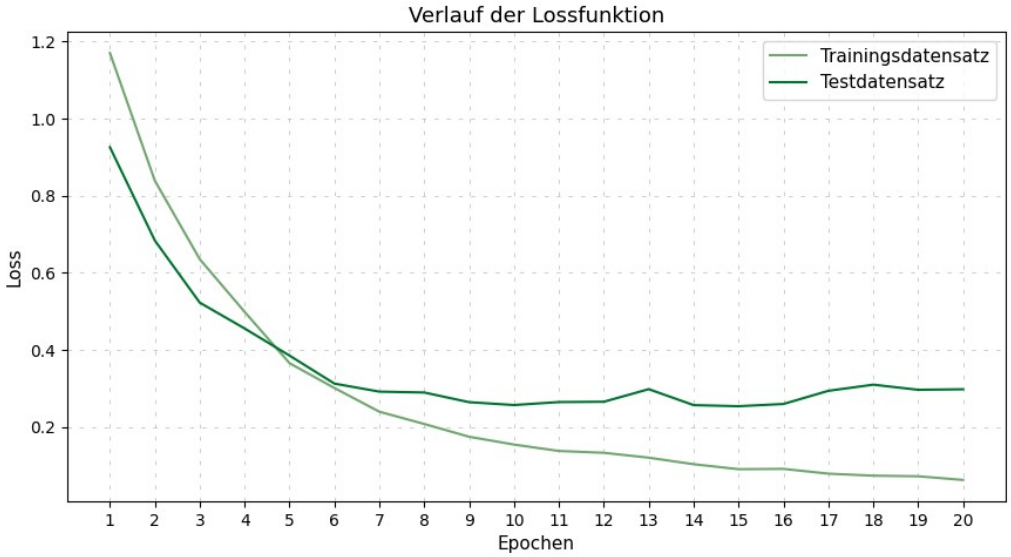

Abbildung 3.9: Verlauf der Loss Function des KNN mit LSTM-Architektur nach Epochen.

Wie man hier erkennen kann, ist das Modell ab etwa 10 Epochen nicht mehr zu verbessern. Sowohl die Loss-Funktion, als auch die Accuracy verändern sich beim Testdatensatz ab dieser Anzahl an Epochen nicht mehr in eine Richtung, sondern pendeln um einen erreichten Wert.

Kapitel 4

Diskussion

4.1 Modellperformance

In Tab. 4.1 sind die Scores $F_{1-macro}$ und $F_{1-weighted}$ der verwendeten Modelle dargestellt. Dabei sieht man, dass das KNN mit LSTM-Architektur beim $F_{1-weighted}$ mit 92% am besten abschneidet.

	$F_{1-macro}$	$F_{1-weighted}$
SVM$_R$	0,68	0,88
SVM$_S$	**0,83**	0,88
KNN	0,66	0,88
LSTM	0,67	**0,92**

Tabelle 4.1: Zusammenfassung der Ergebnisse. SVM$_R$ = SVM ohne SMOTE; SVM$_S$ = SVM mit SMOTE; KNN = Feedforward KNN; LSTM = KNN mit LSTM-Architektur

Die Werte für $F_{1-weighted}$ befinden sich bei den Modellen in einem ähnlichen Bereich, wobei das KNN mit LSTM-Architektur um 4 Prozentpunkte besser abschneidet. Die Sequenzmodell-charakteristik der LSTM-Architektur könnte auf diesem Datensatz einen Vorteil gegenüber der anderen Methodiken haben.

Auffällig ist, dass der $F_{1-macro}$ deutlich niedriger liegt. Der Grund liegt im schlechten Abschneiden seltener Klassen bei diesem Score. Diese werden beim arithmetischen Mittel gleichwertig zu den häufigeren Klassen behandelt, wodurch der Score schlechter ausfällt. Damit erklärt sich auch der höhere Wert für $F_{1-macro}$ bei der SVM mit SMOTE: Da die seltenen Klassen per Oversampling vermehrt wurden, erzielen diese Klassen höhere F_1-Scores und der arithmetische Mittelwert nähert sich dem gewichteten Mittelwert an.

4.1.1 Verwandte Arbeiten

Die Scores ähnlicher Arbeiten sind in Tab. 4.2 dargestellt. Bei Arbeiten, bei denen mehrere ML-Modelle trainiert wurden, wurde jeweils nur das Beste in die Tabelle aufgenommen.

Arbeit	Modell	Klassen	$F_{1-weighted}$	Precision	Recall
Caccamisi et al. [54]	SVM	4	98.1	98.1	98.1
Wang et al. [55]	CNN	5	92.0[1]	93.0[1]	92.0[1]
Figueroa et al. [56]	SVM	2x2[2]	89.1[2]	87.5[2]	91.6[2]
Palmer et al. [57]	SVM	5[3]	90.1[3]	90.9[3]	90.3[3]
Patel et al. [58]	SVM	3	98.0[1]	98.0[1]	98.0[1]
Rajendran et al. [59]	CNN	3	68.0[1]	71.1[1]	69.3[1]
Diese Arbeit	LSTM	6	92.0	92.0	92.0

Tabelle 4.2: Scores verwandter Arbeiten in Prozent.

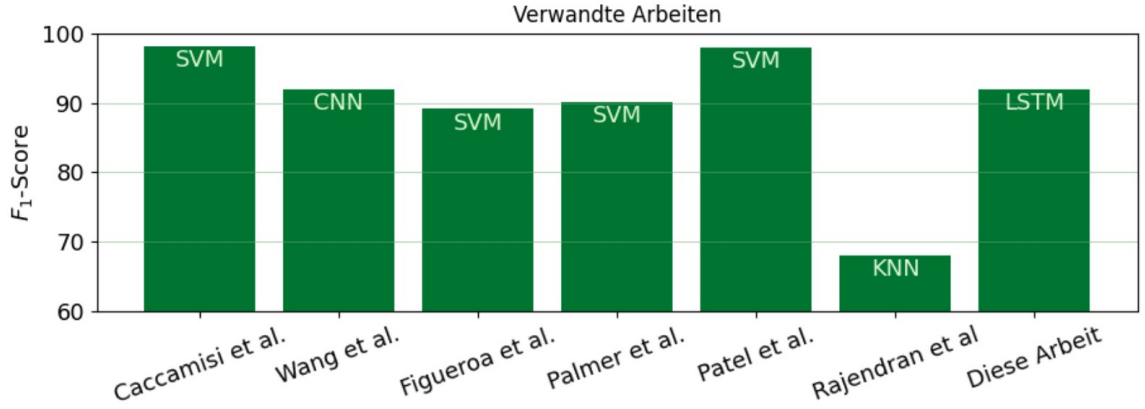

Abbildung 4.1: Grafische Darstellung der F_1-Scores verwandter Arbeiten

4.2 Fehleranalyse

Trotz großer Übereinstimmung der beiden Annotatoren ($p_{obs} = 91,64\%$, $\kappa = 89,97$) ergaben sich einige Abweichungen, welche auf einen fehlenden Standard bei der Dokumentation schließen lassen. Die Auffälligkeit der 38 unterschiedlich manuell klassifizierten "Current or

[1]Keine Angabe, welches Maß (weighted, macro oder micro) berechnet wurde.

[2]Figueroa et al. haben zwei unterschiedliche Klassfikationen mit jeweils zwei Klassen verwendet. Für diese Tabelle wurde ein gewichteter Mittelwert der beiden berechnet.

[3]Palmer et al. haben zwar 6 Klassen verwendet, jedoch wurde eine davon ("Ever") in der Auswertung nicht verwertet. Für diese Tabelle wurde ein gewichteter Mittelwert der einzelnen Scores berechnet.

Past"von Annotator 2 bzw. "Current Smoker" von Annotator 1 (siehe Abb. 3.1) lässt sich nach Kontrolle wie folgt erklären: Befand sich in der Dokumentation die Beschreibung "Risikofaktor: Nikotin", so wurde unterschiedlich bewertet. Diese Formulierung kann einerseits bedeuten, dass der Patient bzw. die Patientin noch Raucher bzw. Raucherin ist oder aber auch, dass das Rauchverhalten in der Vergangenheit liegt.

Eine Auffälligkeit findet sich bei allen untersuchten Modellen für die beiden Klassen "Current Smoker" und "Past Smoker". Diese beiden wurden häufig maschinell verwechselt. Jedoch handelt es sich dabei um die häufigsten Klassen im Datensatz, weshalb auch die Missklassifikationen häufiger ausfallen.

4.3 Limitationen

Die Modelle wurden mit Daten aus EHRs trainiert, weswegen sie auch nur in diesem Kontext anwendbar sind. Abweichungen vom ärztlichen Usus bei der Beschreibung des Nikotinstatus (beispielsweise "Raucher: nein" oder "Raucherin: nein") können nicht von den Modellen erkannt werden, da diese auch im Datensatz nicht vorkommen. Da die Daten nur von bestimmten Kliniken mit bestimmten Krankheiten erhoben wurden (Kardiologie, Dermatologie und Patientinnen und Patienten mit Kolonkarzinom), ist von einem Selektionsbias auszugehen. Je nach medizinischem Fachgebiet werden Notationen des Raucherstatus anders gehandhabt oder überhaupt erst erhoben. Interessant wären hier auch Daten aus einer pulmologischen und einer angiologischen Klinik gewesen, da in diesem ein positiver Raucherstatus aufgrund typischer Folgeerkrankungen des Rauchens stark überrepräsentiert sein dürfte.

Ebenso bestehen Ungenauigkeiten bei der Klassifikation durch ungenaue Dokumentation seitens des medizinischen Personals. Beispielsweise ist aus der Bezeichnung "Nikotin: negativ" nicht ersichtlich, ob es sich um eine Person handelt, die nie geraucht hat oder die in der Vergangenheit geraucht hat. Eine weitere Ungenauigkeit besteht beim Erkennen von Nikotinpflastern oder anderen Formen der Aufnahme von Nikotin. Zusätzlich können auch andere Substanzen als Nikotin geraucht werden, was eine eigene Dokumentation benötigen würde, häufig aber unter "Rauchen" aufgenommen wird. Es besteht die Annahme, dass Abweichungen vom üblichen Rauchverhalten genau beschrieben werden.

Die vermutlich größte Fehlerquelle dürften die Ungenauigkeiten in den Daten selbst sein, welche in Kap. 1.1 beschrieben wurden. Da die Modelle mit eben jenen fehlerhaften Daten trainiert wurden, pflanzen sich diese Fehler auch in die Modellperformance fort.

4.4 Zusammenfassung und Ausblick

Trotz des limitierten Datensatzes konnten bereits gute Ergebnisse bei der Klassifikation des Raucherstatus erzielt werden. Mit einer Verbesserung der Daten durch eine gleichmäßigere Klassenverteilung könnte die Leistung insofern weiter verbessert werden, bis sie eventuell die Werte von menschlichen Experten bei der Einteilung des Raucherstatus aus klinischer-freitextlicher Routinedokumentation in Kategorien internationaler Standards wie SNOMED CT erreichen könnte.

Die Robustheit der Modelle würde wahrscheinlich mithilfe von Daten höherer Diversität ebenfalls verstärkt werden. Das könnten Daten anderer Kliniken, aber auch anderer Länder sein. Die unterschiedlichen Gewohnheiten der Notation wären damit auch im Trainingsdatensatz abgebildet. Die Robustheit könnte in einem nächsten Schritt auch auf der methodischen Basis transformerbasierter Modelle erhöht werden. Hierbei spielt die technische Integration als NLP-Komponente eine wichtige Rolle. Hierfür würde sich Apaches UIMA (Unstructured Information Management Architecture) eignen, um große textuelle Datenmengen systematisch zu analysieren [60].

Gemäß der FAIR-Kriterien (Findable, Accessible, Interoperable, Reusable) wäre eine Betrachtung des Raucherstatus im Kontext von HL7-FHIR [61, 62], in Kombination mit SNOMED CT von Interesse und inwieweit sich das hier gefunden Klassifikationsschema auf internationale Interoperabilitäts- und Kommunikationsstandards in der Medizin abbilden lässt. Eine erste Zuordnung zu SNOMED CT Codes wurde in dieser Arbeit präsentiert. Ein potenzieller Anwendungsfall wäre in weiterer Folge die Unterstützung retrospektiver Studien aus der gewonnenen codierten Information aus klinischer Routinedokumentation, eingebettet in das entsprechende Informationsmodell. Nikotin als relevante Noxe vieler Krankheiten könnte auf diese Weise epidemiologisch und interoperabel, akkurater erforscht werden.

Literaturverzeichnis

[1] Dilyara G. Yanbaeva , Mieke A. Dentener , Eva C. Creutzberg , Geertjan Wesseling , und Emiel F.M. Wouters . Systemic effects of smoking. *Chest*, 131(5):1557–1566, 2007. ISSN 0012-3692. doi: 10.1378/chest.06-2179. URL https://www.sciencedirect.com/science/article/pii/S0012369215316305.

[2] Freiman A., Bird G., Metelitsa A. I., Barankin B., und Lauzon G. J. Cutaneous effects of smoking. *Journal of Cutaneous Medicine and Surgery*, 8(6):415–423, 2004. ISSN 1615-7109. doi: 10.1007/s10227-005-0020-8. URL https://doi.org/10.1007/s10227-005-0020-8.

[3] Wong P. K. K., Christie J. J., und Wark J. D. The effects of smoking on bone health. *Clinical Science*, 113(5):233–241, 2007. ISSN 0143-5221. doi: 10.1042/CS20060173.

[4] A. Sloan , I. Hussain , M. Maqsood , O. Eremin , und M. El-Sheemy . The effects of smoking on fracture healing. *The Surgeon*, 8(2):111–116, 2010. ISSN 1479-666X. doi: 10.1016/j.surge.2009.10.014. URL https://www.sciencedirect.com/science/article/pii/S1479666X09000158.

[5] DOLL R. und HILL A. B. Smoking and carcinoma of the lung; preliminary report. *British medical journal*, 2(4682):739–748, 1950. ISSN 0007-1447. doi: 10.1136/bmj.2.4682.739.

[6] Stephan R. Orth , Eberhard Ritz , und Robert W. Schrier . The renal risks of smoking. *Kidney International*, 51(6):1669–1677, 1997. ISSN 0085-2538. doi: 10.1038/ki.1997.232. URL https://www.sciencedirect.com/science/article/pii/S0085253815600722.

[7] Kukhareva P. V., Caverly T. J., Li H., Katki H. A., Cheung L. C., Reese T. J., Del Fiol G., Hess R., Wetter D. W., Zhang Y., Taft T. Y., Flynn M. C., und Kawamoto K. Inaccuracies in electronic health records smoking data and a potential approach to address resulting underestimation in determining lung cancer screening eligibility. *Journal of the American Medical Informatics Association*, 29(5):779–788, 2022. ISSN 1527-974X. doi: 10.1093/jamia/ocac020.

[8] Botsis T., Hartvigsen G., Chen F., und Weng C. Secondary use of EHR: data quality issues and informatics opportunities. *Summit on Translational Bioinformatics*, 2010:1, 2010.

[9] Hirschtick R. E. Copy-and-Paste. *JAMA*, 295(20):2335–2336, 05 2006. ISSN 0098-7484. doi: 10.1001/jama.295.20.2335. URL https://doi.org/10.1001/jama.295.20.2335.

[10] Thebault J.-L., Falcoff H., Favre M., Noël F., und Rigal L. Patient-physician agreement on tobacco and alcohol consumption: a multilevel analysis of gps' characteristics. *BMC Health Services Research*, 15(1):110, 2015. ISSN 1472-6963. doi: 10.1186/s12913-015-0767-6. URL https://doi.org/10.1186/s12913-015-0767-6.

[11] Russell S. und Norvig P. *Künstliche Intelligenz: Ein moderner Ansatz*. Pearson Deutschland GmbH, 4 edition, 2021. ISBN 978-1-292-40113-3.

[12] Haugeland J. *Artificial Intelligence: The Very Idea*. Cambridge: MIT Press, 1985.

[13] Charniak E. *Introduction to Artificial Intelligence*. ADDISON-WESLEY SERIES IN COMPUTER SCIENCE. Pearson Education, 1985. ISBN 9788131703069. URL https://books.google.at/books?id=HACaS635bYcC.

[14] Bellman R. *An Introduction to Artificial Intelligence: Can Computers Think?* Boyd & Fraser Publishing Company, 1978. ISBN 9780878350667. URL https://books.google.at/books?id=84xQAAAAMAAJ.

[15] Winston P. H. *Artificial Intelligence*. Addison-Wesley, 3 edition, 1992. ISBN 978-0-201-53377-4.

[16] Kurzweil R. *The age of intelligent machines*. MIT Press, 1990.

[17] Poole D., Mackworth A., und Goebel R. *Computational Intelligence: A Logical Approach*. Oxford University Press, 1998.

[18] Rich E. und Knight K. *Artificial Intelligence*. Artificial Intelligence Series. McGraw-Hill, 1991. ISBN 9780071008945. URL https://books.google.at/books?id=6P6jPwAACAAJ.

[19] Nilsson N. J. *Artificial Intelligence: A New Synthesis*. Elsevier, 1998. doi: 10.1016/c2009-0-27773-7. URL https://doi.org/10.1016/c2009-0-27773-7.

[20] Gabo.de . Grafik angelehnt an Venn-Diagramm von Gabo.de, zuletzt aufgerufen am 08.03.2023. URL https://www.gabo.de/wp-content/uploads/2020/01/AI-vs-ML-vs-Deep-Learning_GABO-e1578660361203.png.

[21] Géron A. *Praxiseinstieg Machine Learning mit Scikit-Learn, Keras und TensorFlow: Konzepte, Tools und Techniken für intelligente Systeme.* dpunkt Verlag, 2 edition, 2020. ISBN 9783960091240.

[22] Chollet F. Keras. Eine Machine Learning API für Python, 2015. URL https://keras.io/api/.

[23] Cournapeau D., Brucher M., Pedregosa F., Varoquaux G., und Michel V. Scikit-learn. Eine Machine Learning Bibliothek für Python, 2007. URL https://scikit-learn.org/stable/.

[24] Vapnik V. und Chervonenkis A. Theory of pattern recognition, 1974.

[25] Vapnik V. und Chervonenkis A. *Theorie der Zeichenerkennung.* Akademie-Verlag, 1979.

[26] Richter S. *Statistisches und maschinelles Lernen - Gängige Verfahren im Überblick.* Springer-Spektrum Verlag, 2019. ISBN 978-3-662-59353-0.

[27] Ertel W. *Grundkurs Künstliche Intelligenz: Eine praxisorientierte Einführung.* Springer Vieweg Verlag, 4 edition, 2016. ISBN 978-3-658-13548-5.

[28] Cortes C. und Vapnik V. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995. doi: 10.1007/bf00994018. URL https://doi.org/10.1007/bf00994018.

[29] Alberts B., Johnson A. D., Lewis J., Morgan D., und Raff M. *Molekularbiologie der Zelle.* Wily-VCH Verlag, 6 edition, 2017. ISBN 978-3-527-34072-9.

[30] Lüllmann-Rauch R. und Asan E. *Taschenlehrbuch Histologie.* Thieme Verlag, 5 edition, 2015. ISBN 978-3131292452.

[31] Frotscher M. und Kahle W. *Taschenatlas Anatomie, Band 3: Nervensystem und Sinnesorgane.* Thieme Verlag, 2 edition, 2013. ISBN 978-3-13-492211-0.

[32] Schünke M., Schulte E., Schuhmacher U., Voll M., und Wesker K. *Prometheus Kopf, Hals und Neuroanatomie.* Thieme Verlag, 2 edition, 2009. ISBN 978-3-13-139542-9.

[33] McCulloch W. S. und Pitts W. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, December 1943. doi: 10.1007/bf02478259. URL https://doi.org/10.1007/bf02478259.

[34] Rosenblatt F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958. doi: 10.1037/h0042519. URL https://doi.org/10.1037/h0042519.

[35] Chrislb . Grafik angelehnt: Schema eines künstlichen neurons: Künstliches neuron mit index j, zuletzt aufgerufen am 08.03.2023. GNU Free Documentation License, 2005. URL https://commons.wikimedia.org/wiki/File:ArtificialNeuronModel_deutsch.png.

[36] Olah C. Colah's blog, August 2015. URL https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[37] Hochreiter S. und Schmidhuber J. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, November 1997. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

[38] Rumelhart D. E., Hinton G. E., und Williams R. J. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986. doi: 10.1038/323533a0. URL https://doi.org/10.1038/323533a0.

[39] Carstensen K.-U., Ebert C., Ebert C., Jekat S., Klabunde R., und Langer H. *Computerlinguistik und Sprachtechnologie. Eine Einführung.* Spektrum Akademischer Verlag, 3 edition, 2010. ISBN 978-3-827420237.

[40] Google.com . Textklassifizierung - schritt 3: Daten vorbereiten, zuletzt aufgerufen am 10. Oktober 2022. URL https://developers.google.com/machine-learning/guides/text-classification/step-3.

[41] (BSD License) scikit-learn developers . Scikit-learn documentation: 6.2. feature extraction, 2022. URL https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction.

[42] Sparck Jones K. A statistical interpretation of term specifity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, January 1972. doi: 10.1108/eb026526. URL https://doi.org/10.1108/eb026526.

[43] Mikolov T., Chen K., Corrado G., und Dean J. Efficient estimation of word representations in vector space, 2013. URL https://arxiv.org/abs/1301.3781.

[44] Devlin J., Chang M.-W., Lee K., und Toutanova K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

[45] Bakeman R. und Gottman J. M. *Observing Interaction: An Introduction to Sequential Analysis*. Cambridge University Press, 2 edition, 1997. ISBN 978-0521450089. doi: 10.1017/CBO9780511527685.

[46] Fleiss J. L. und Chilton N. W. The measurement of interexaminer agreement on periodontal disease. *Journal of Periodontal Research*, 18(6):601–606, December 1983. doi: 10.1111/j.1600-0765.1983.tb00397.x. URL https://doi.org/10.1111/j.1600-0765.1983.tb00397.x.

[47] Landis J. R. und Koch G. G. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159, March 1977. doi: 10.2307/2529310. URL https://doi.org/10.2307/2529310.

[48] Youngstrom E. Grafik angelehnt: Comparison of four widely-cited recommendations for labeling levels of inter-rater agreement, zuletzt aufgerufen am 08.03.2023. GNU Free Documentation License, 12.12.2019. URL https://commons.wikimedia.org/wiki/File:Comparison_of_rubrics_for_evaluating_inter-rater_kappa_(and_intra-class_correlation)_coefficients.png.

[49] Niu M., Li Y., Wang C., und Han K. Grafik angelehnt an: Abbildung 4, RFAmyloid: A Web Server for Predicting Amyloid Proteins - Scientific Figure on ResearchGate, , zuletzt aufgerufen am 08.03.2023. *International journal of molecular sciences*, 19, 07 2018. doi: 10.3390/ijms19072071. URL https://www.researchgate.net/figure/Ten-fold-cross-validation-diagram-The-dataset-was-divided-into-ten-parts-and-nine-of_fig1_326465007.

[50] Maleki F., Ovens K., Najafian K., Forghani B., Md C., und Forghani R. Grafik angelehnt: Overview of machine learning part 1 - scientific figure on researchgate. , zuletzt aufgerufen am 05.03.2023. *Neuroimaging Clinics of North America*, 30:e17–e32, 11 2020. doi: 10.1016/j.nic.2020.08.007. URL https://www.researchgate.net/figure/Visualizing-accuracy-recall-aka-sensitivity-and-precision-which-are-the-common_fig3_346129022.

[51] Systematized Nomenclature of Medicine Clinical Terms. SNOMED International, zuletzt aufgerufen am 30.03.2023. URL https://www.snomed.org/?lang=de.

[52] Lemaître G., Nogueira F., und Aridas C. K. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017. URL http://jmlr.org/papers/v18/16-365.html.

[53] O'Connor C. und Joffe H. Intercoder reliability in qualitative research: Debates and practical guidelines. *International Journal of Qualitative Methods*, 19: 160940691989922, January 2020. doi: 10.1177/1609406919899220. URL https://doi.org/10.1177/1609406919899220.

[54] Caccamisi A., Jørgensen L., Dalianis H., und Rosenlund M. Natural language processing and machine learning to enable automatic extraction and classification of patients' smoking status from electronic medical records. *Upsala Journal of Medical Sciences*, 125(4):316–324, July 2020. doi: 10.1080/03009734.2020.1792010. URL https://doi.org/10.1080/03009734.2020.1792010.

[55] Wang Y., Sohn S., Liu S., Shen F., Wang L., Atkinson E. J., Amin S., und Liu H. A clinical text classification paradigm using weak supervision and deep representation. *BMC Medical Informatics and Decision Making*, 19(1), January 2019. doi: 10.1186/s12911-018-0723-6. URL https://doi.org/10.1186/s12911-018-0723-6.

[56] Figueroa R. L., Soto D. A., und Pino E. J. Identifying and extracting patient smoking status information from clinical narrative texts in spanish. In *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, August 2014. doi: 10.1109/embc.2014.6944182. URL https://doi.org/10.1109/embc.2014.6944182.

[57] Palmer E. L., Hassanpour S., Higgins J., Doherty J. A., und Onega T. Building a tobacco user registry by extracting multiple smoking behaviors from clinical notes. *BMC Medical Informatics and Decision Making*, 19(1), July 2019. doi: 10.1186/s12911-019-0863-3. URL https://doi.org/10.1186/s12911-019-0863-3.

[58] Patel J., Siddiqui Z., Krishnan A., und Thyvalikakath T. Leveraging electronic dental record data to classify patients based on their smoking intensity. *Methods of Information in Medicine*, 57(05/06):253–260, November 2018. doi: 10.1055/s-0039-1681088. URL https://doi.org/10.1055/s-0039-1681088.

[59] Rajendran S. und Topaloglu U. Extracting smoking status from electronic health records using nlp and deep learning. *AMIA Joint Summits on Translational Science proceedings. AMIA Joint Summits on Translational Science*, 2020:507–516, 2020. ISSN 2153-4063.

[60] Foundation T. A. S. UIMA. Unstructured Information Management Architecture, 2010. URL https://uima.apache.org/.

[61] V. e. H. D. FHIR. Fast Healthcare Interoperability Resources, 2014. URL https://hl7.de/themen/hl7-fhir-mobile-kommunikation-und-mehr/warum-fhir/.

[62] Wilkinson M. D., Dumontier M., Aalbersberg I. J., Appleton G., Axton M., Baak A., Blomberg N., Boiten J.-W., Silva Santos da L. B., Bourne P. E., Bouwman J., Brookes A. J., Clark T., Crosas M., Dillo I., Dumon O., Edmunds S., Evelo C. T., Finkers R., Gonzalez-Beltran A., Gray A. J., Groth P., Goble C., Grethe J. S., Heringa J., Hoen 't P. A., Hooft R., Kuhn T., Kok R., Kok J., Lusher S. J., Martone M. E., Mons A., Packer A. L., Persson B., Rocca-Serra P., Roos M., Schaik van R., Sansone S.-A., Schultes E., Sengstag T., Slater T., Strawn G., Swertz M. A., Thompson M., Lei van der J., Mulligen van E., Velterop J., Waagmeester A., Wittenburg P., Wolstencroft K., Zhao J., und Mons B. The FAIR guiding principles for scientific data management and stewardship. *Scientific Data*, 3(1), March 2016. doi: 10.1038/sdata.2016.18. URL https://doi.org/10.1038/sdata.2016.18.

Anhang A

Quellcode

A.1 functions.py

```
import os
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import
    CountVectorizer, TfidfVectorizer

# This Class contains 1 text and 1 smokingstatus
# supports the readability of the Data
class Data:
    def __init__(self, text, smokingStatus):
        self.text = text
        self.smokingStatus = smokingStatus

    def get_SmokingStatus(self):
        return self.smokingStatus

    def get_text(self):
        return self.text

    def create_data_array_from_DataClass(self):
        letters = []
```

```
            for i,j in enumerate(self.get_SmokingStatus()):
                letters.append(Data(self.get_text()[i],
                    self.get_SmokingStatus()[i]))
            return letters

    # Reads the Data in argument filename and returns an
        array with the 3 values:
    # [Pandas Dataframe,     Numpy bool Array without column
        1,       Headernames as vector]
    # Delimiter = ;
    # encoding = latin1 (this is necessary to encode german
        charakters like "a)
    def read_csv_semicolon_separated(filename):
        # Pandas DataFrame:
        data_pd = pd.read_csv(filename, delimiter =';',
            encoding='latin1', header=0)

        # Numpy bool Array without column 1, without header
        data_np = data_pd.to_numpy()
        data_np = data_np[:, 1:]
        data_np = data_np == 1

        # Header Array
        header = data_pd.columns.values
        header = header.astype(str)
        return [data_pd, data_np, header]

    # Takes the Data from above and returns 2 Vectors x,y
    # x: First Column of Dataset as String
    # y: Vector, which contains the headername, where the
       row has value 1
    def data_to_xy(pandas_df, numpy_array, header):
        x = pandas_df[header[0]].to_numpy()

        # initialize vector y
        y = np.empty(numpy_array.shape[0])
        y = y.astype(str)
```

```
    # Fill vector y
    for i,j in enumerate(numpy_array):
        a = np.where(numpy_array[i])[0] + 1      # a is
            just the index, where the row i has value
            True
        y[i] = header[a][0]                      # fill
            in the header values to vector y
    return [x,y]

def split_train_test(letters):
    training, test = train_test_split(letters,
        test_size = 0.20)

    x_train = [x.text for x in training]
    y_train = [y.smokingStatus for y in training]

    x_test = [x.text for x in test]
    y_test = [y.smokingStatus for y in test]

    return [x_train, y_train, x_test, y_test]

def convert_vec_to_int(vector):
    for i, v in enumerate(vector):
        if v == np.amax(vector):
            return i+1

def convert_int_to_vec(integer, size):
    vec = np.zeros(size)
    vec[integer-1] = 1
    return vec

def convert_intarr_to_vecarray(vec):
    returnvec = np.zeros([vec.size, 6])
    for n, i in enumerate(vec):
        returnvec[n, i] = 1
    return returnvec
```

A.2 SVM.py

```
#region: Imports
import numpy as np
import random
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import
    CountVectorizer, TfidfVectorizer
from sklearn import svm
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import f1_score
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix,
    ConfusionMatrixDisplay
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report
from imblearn.over_sampling import SMOTE
import pandas as pd
import fasttext
import json
# Own Module
import functions
#endregion

# Path to CSV-File
datafilepath = 'E:\SmokingStatus_GS.CSV'

#region: Data Preparation
##### DATA PREPARATION #######
# Reading the CSV and creating an Array "letters".
# The Data-Class from functions contains the letter and
     the smoking status
[data_pd, data_np, header] = functions.
    read_csv_semicolon_separated(datafilepath)
[x, y] = functions.data_to_xy(data_pd, data_np, header)
                # Data as Numpy-Array
```

```
letters = functions.Data(x, y).
    create_data_array_from_DataClass()
# Data as Array with Class Values
# Example how to access Data:
# Get Text from second Datapoint: letters[2].get_text()
# Get Smoking Status Class from second Datapoint:
    letters[2].get_SmokingStatus()

# Datasets as a whole x- and y-list
x_gesamt = [x.text for x in letters]
y_gesamt = [y.smokingStatus for y in letters]

##### SPLITTING THE DATA INTO TRAINING AND TEST SET
    #####
[x_train_0, y_train, x_test_0, y_test] = functions.
    split_train_test(letters)
# this function is not to be mistaken with
    train_test_split from scikit-learn!
# It is a Custom function defined in the functions
    module

#vectorizer = CountVectorizer()
vectorizer = TfidfVectorizer()
x_train = vectorizer.fit_transform(x_train_0)    # Bags
    of Words vectorization
x_test = vectorizer.transform(x_test_0)          # No
    fit needed here (already fit above), Bags of Words
    vectorization
#endregion

#region: Model Definition, Fit and Score
##### MODEL #####
parameters = {'kernel': ('linear', 'sigmoid', 'rbf', '
    poly'), 'C': (1,4,8,16,32)}
classifier_svm = svm.SVC(kernel='sigmoid', C=1)

classifier_svm.fit(x_train, y_train)
```

```
# Prediction Score of the Model
clf_score = classifier_svm.score(x_test, y_test)
print('Score overall: ' + str(clf_score))
print('---------------')
#endregion

#region: Evaluation Without SMOTE
# F1-Score of the Model
print('F1-Scores without SMOTE:')
classes = header[1:]
f_score = f1_score(y_test, classifier_svm.predict(
    x_test), average=None, labels=classes, zero_division
    =1)
for i,j in enumerate(classes):
     print(str(j) + ': ' + str(f_score[i]))

# Confusion Matrix without SMOTE
disp_labels = ['CNS','COP', 'CS', 'NS', 'PS', 'U']
fig = ConfusionMatrixDisplay.from_predictions(y_test,
    classifier_svm.predict(x_test), display_labels=
    disp_labels)
fig.plot(cmap="Greens")
figu = fig.ax_.get_figure()
figu.set_figwidth(8)
figu.set_figheight(6)
plt.title("SVM without SMOTE")
plt.tight_layout()
plt.show()
print(classification_report(y_test, classifier_svm.
    predict(x_test)))

# Crossvalidation without SMOTE
x_gesamt_vec = vectorizer.fit_transform(x_gesamt)
            # First Vectorize Data
classifier_svm.fit(x_gesamt_vec, y_gesamt)
```

```
scores = cross_val_score(classifier_svm, x_gesamt_vec,
    y_gesamt, scoring="accuracy", cv=5)
print('\nCrossVal-Scores for mean accuracy (which is
    the default Scorer for the SVC Model in Scikit-learn
    ): ')
print(scores)
#endregion

# Resampling with SMOTE
print('\n----------\nNow repeat with SMOTE: \n')
smote = SMOTE()
x_train_resample, y_train_resample = smote.fit_resample
    (x_train, y_train)
x_test_resample = x_test     # Trainset must remain
    unchanged
classifier_svm_SMOTE = svm.SVC(kernel='sigmoid', C=50)
classifier_svm_SMOTE.fit(x_train_resample,
    y_train_resample)
clf_score_SMOTE = classifier_svm_SMOTE.score(
    x_test_resample, y_test)
print('Score overall with SMOTE: ' + str(
    clf_score_SMOTE))
print('---------------')
print('F1-Scores with SMOTE:')
classes = header[1:]
f_score_SMOTE = f1_score(y_test, classifier_svm_SMOTE.
    predict(x_test_resample), average=None, labels=
    classes, zero_division=1)
for i,j in enumerate(classes):
     print(str(j) + ': ' + str(f_score_SMOTE[i]))

# Confusion Matrix with SMOTE
conf_matrix_SMOTE = confusion_matrix(y_test,
    classifier_svm_SMOTE.predict(x_test_resample))
fig = ConfusionMatrixDisplay.from_predictions(y_test,
    classifier_svm_SMOTE.predict(x_test_resample),
    display_labels=disp_labels)
```

```
fig.plot(cmap="Greens")
figu = fig.ax_.get_figure()
figu.set_figwidth(8)
figu.set_figheight(6)
plt.title("SVM with SMOTE")
plt.tight_layout()
plt.show()

print(classification_report(y_test,
    classifier_svm_SMOTE.predict(x_test_resample)))

### Crossvalidation with SMOTE
classifier_svm_SMOTE.fit(x_gesamt_vec, y_gesamt)
scores_SMOTE = cross_val_score(classifier_svm_SMOTE,
    x_gesamt_vec, y_gesamt, scoring="accuracy", cv=5)
print('\nCrossVal-Scores with SMOTE: ')
print(scores_SMOTE)

print("done")
```

A.3 KNN.py

```
#region: Imports
import numpy as np
from sklearn.metrics import accuracy_score,recall_score
    ,precision_score,f1_score, confusion_matrix,
    ConfusionMatrixDisplay, classification_report
from sklearn.model_selection import train_test_split
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from keras.models import Model, Sequential
from keras.layers import RNN, LSTM, Activation, Dense,
    Input, Embedding, Dropout, GlobalAveragePooling1D
from keras.preprocessing.text import Tokenizer
from keras.preprocessing import sequence
from keras.callbacks import EarlyStopping,
    ReduceLROnPlateau
import seaborn as sns
from sklearn import metrics
import time
import functions
#endregion

#region: Definitions
datafilepath = 'E:\SmokingStatus_GS - DataPrep.xlsx'
datacsv1 = 'E:/SmokingStatus/file_name.csv'
datacsv2 = 'E:/SmokingStatus/file_name_lstm.csv'

max_word = 5000      # for Tokenizer. Size of Vocabulary
max_len = 50
epochs = 40

fontsize = 13
fontsize2 = 11
#endregion

#region: Data Preprocessing
```

```
data = pd.read_excel(datafilepath, usecols=[0,8], names
    =['Text','Label'])

#split the data into train and test datas
X_Train, X_test, Y_Train, Y_test = train_test_split(
    data['Text'],data['Label'], test_size=0.2,
    random_state=42)

Encoder = LabelEncoder()
Y_Train = Encoder.fit_transform(Y_Train)
Y_test = Encoder.fit_transform(Y_test)

Y_Train = functions.convert_intarr_to_vecarray(Y_Train)
Y_test_old = Y_test.copy()
Y_test_old = Y_test_old + 1
Y_test = functions.convert_intarr_to_vecarray(Y_test)

# Split the remaining data to train and validation
X_train, X_val, Y_train, Y_val = train_test_split(
    X_Train, Y_Train, test_size=0.2, random_state=1000)

# Tokenize, Vectorize
tok = Tokenizer(max_word)
tok.fit_on_texts(data['Text'])

sequences = tok.texts_to_sequences(X_train)
sequences_matrix = sequence.pad_sequences(sequences,
    maxlen=max_len)

Val_sequences = tok.texts_to_sequences(X_val)
Val_sequences_matrix = sequence.pad_sequences(
    Val_sequences,maxlen=max_len)

test_sequences = tok.texts_to_sequences(X_test)
test_sequences_matrix = sequence.pad_sequences(
    test_sequences,maxlen=max_len)
#endregion
```

```
#region: Model Definition & Compilation
model = Sequential()
model.add(Embedding(max_word, 256, input_length=max_len
    ))
model.add(GlobalAveragePooling1D())
#model.add(Input(shape=sequences_matrix.shape[1]))
model.add(Dense(256))
model.add(Dense(6, input_dim=256, activation="sigmoid",
     name='out_layer'))
model.summary()

model.compile(loss='categorical_crossentropy',optimizer
    ='Adamax',metrics=['accuracy'])
#endregion

#region: Model Fit and Evaluation
earlyStopping = EarlyStopping(monitor='val_loss',
    patience=10, min_delta=0.0001, restore_best_weights=
    True)
reduce_lr_loss = ReduceLROnPlateau(monitor='val_loss',
    factor=0.1, patience=10, verbose=1, min_delta=1e-4,
    mode='min')

start = time.time()
history=model.fit(sequences_matrix, Y_train, batch_size
    =128,epochs=epochs,
          validation_data=(Val_sequences_matrix, Y_val),
             callbacks=[earlyStopping, reduce_lr_loss])
end = time.time()
print(end - start, "seconds - Training Time")

start = time.time()
Evaluation = model.evaluate(test_sequences_matrix,
    Y_test)
end = time.time()
print(end - start, "seconds - Evaluation Time")
```

```
print('Test set\n  Loss: {:0.3f}\n  Accuracy: {:0.3f}'.
    format(Evaluation[0],Evaluation[1]))
#endregion

#region: Model Prediction Testing
start = time.time()
Prediction = model.predict(test_sequences_matrix)
end = time.time()
print(end - start, "seconds - Prediction Time")

# Convert Prediction from one-hot Encoding to Integer
Prediction = Prediction.tolist()
for i, predic in enumerate(Prediction):
    Prediction[i] = functions.convert_vec_to_int(predic
        )
#endregion

#region: Confusion-Matrix
disp_labels = ['PS','CS', 'CNS', 'NS', 'COP', 'U']
fig = ConfusionMatrixDisplay.from_predictions(
    Y_test_old, Prediction, display_labels=disp_labels)
fig.plot(cmap="Greens")
figu = fig.ax_.get_figure()
figu.set_figwidth(8)
figu.set_figheight(6)
plt.title("KNN")
plt.tight_layout()
plt.show()
print(classification_report(Y_test_old, Prediction))
#endregion

color1="#007532"     # Color for Matplotlib Figures
color2="#73a771"     # Color for Matplotlib Figures

#region: Figure: Training and Validation Loss
plt.plot(history.history['loss'], label='
    Trainingsdatensatz', color=color2)
```

```
plt.plot(history.history['val_loss'], label='
    Testdatensatz', color=color1)
fig = plt.gcf()      # get current figure
fig.set_size_inches(10, 5)
plt.title('Verlauf der Lossfunktion', fontsize=fontsize
    )
plt.xlabel('Epochen', fontsize=fontsize2)
plt.xticks(np.arange(0, epochs, step=1), np.arange(1,
    epochs+1, step=1))
plt.ylabel('Loss', fontsize=fontsize2)
plt.legend(fontsize=fontsize2)
plt.grid(color='#BABABA', linewidth=0.5, linestyle=(0,
    (5, 10)))
plt.subplots_adjust(top=0.9, bottom=0.1, right=0.9,
    left=0.1)
plt.show()
#endregion

#region: Figure: Training and Validation Accuracy
plt.plot(history.history['accuracy'], label='
    Trainingsdatensatz', color=color2)
plt.plot(history.history['val_accuracy'], label='
    Testdatensatz', color=color1)
fig = plt.gcf()
fig.set_size_inches(10, 5)
plt.title('Verlauf der Accuracy', fontsize=fontsize)
plt.xlabel('Epochen', fontsize=fontsize2)
plt.xticks(np.arange(0, epochs, step=5), np.arange(1,
    epochs+1, step=5))
plt.ylabel('Accuracy', fontsize=fontsize2)
plt.legend(fontsize=fontsize2)
plt.grid(color='#BABABA', linewidth=0.5, linestyle=(0,
    (5, 10)))
plt.subplots_adjust(top=0.9, bottom=0.1, right=0.9,
    left=0.1)
plt.show()
#endregion
```

```
print("done")
```

A.4 LSTM.py

```
#region: Imports
import numpy as np
from sklearn.metrics import accuracy_score,recall_score
    ,precision_score,f1_score, confusion_matrix,
    ConfusionMatrixDisplay, classification_report
from sklearn.model_selection import train_test_split
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from keras.models import Model, Sequential
from keras.layers import RNN, LSTM, Activation, Dense,
    Input, Embedding
from keras.preprocessing.text import Tokenizer
from keras.preprocessing import sequence
from keras.callbacks import EarlyStopping,
    ReduceLROnPlateau
import seaborn as sns
from sklearn import metrics
import time
import functions
#endregion

#region: Definitions
datafilepath = 'E:\SmokingStatus_GS - DataPrep.xlsx'
datacsv1 = 'E:/SmokingStatus/file_name.csv'
datacsv2 = 'E:/SmokingStatus/file_name_lstm.csv'

max_word = 5000      # for Tokenizer. Size of Vocabulary
max_len = 50
epochs = 20
fontsize = 13
```

```
fontsize2 = 11
#endregion

#region: Data Preprocessing
data = pd.read_excel(datafilepath, usecols=[0,8], names
    =['Text','Label'])

#split the data into train and test datas
X_Train, X_test, Y_Train, Y_test = train_test_split(
    data['Text'],data['Label'], test_size=0.2,
    random_state=42)

Encoder = LabelEncoder()
Y_Train = Encoder.fit_transform(Y_Train)
Y_test = Encoder.fit_transform(Y_test)

Y_Train = functions.convert_intarr_to_vecarray(Y_Train)
Y_test_old = Y_test.copy()
Y_test_old = Y_test_old + 1
Y_test = functions.convert_intarr_to_vecarray(Y_test)

# Split the remaining data to train and validation
X_train, X_val, Y_train, Y_val = train_test_split(
    X_Train, Y_Train, test_size=0.2, random_state=1000)

# Tokenize, Vectorize
tok = Tokenizer(max_word)
tok.fit_on_texts(data['Text'])

sequences = tok.texts_to_sequences(X_train)
sequences_matrix = sequence.pad_sequences(sequences,
    maxlen=max_len)

Val_sequences = tok.texts_to_sequences(X_val)
Val_sequences_matrix = sequence.pad_sequences(
    Val_sequences,maxlen=max_len)
```

```
test_sequences = tok.texts_to_sequences(X_test)
test_sequences_matrix = sequence.pad_sequences(
    test_sequences,maxlen=max_len)
#endregion

#region: Model Definition & Compilation
model = Sequential()
model.add(Embedding(max_word, 256, input_length=max_len
    ))
model.add(LSTM(256, dropout=0.2, recurrent_dropout=0.2)
    )
model.add(Dense(6, activation="sigmoid", name='
    out_layer'))
model.summary()

model.compile(loss='categorical_crossentropy',optimizer
    ='Adamax',metrics=['accuracy'])
#endregion

#region: Model Fit and Evaluation
earlyStopping = EarlyStopping(monitor='val_loss',
    patience=10, min_delta=0.0001, restore_best_weights=
    True)
reduce_lr_loss = ReduceLROnPlateau(monitor='val_loss',
    factor=0.1, patience=7, verbose=1, min_delta=1e-4,
    mode='min')

start = time.time()
history=model.fit(sequences_matrix, Y_train, batch_size
    =128,epochs=epochs,
          validation_data=(Val_sequences_matrix, Y_val),
             callbacks=[earlyStopping, reduce_lr_loss])
end = time.time()
print(end - start, "seconds - Training Time")

start = time.time()
```

```
Evaluation = model.evaluate(test_sequences_matrix,
    Y_test)
end = time.time()
print(end - start, "seconds - Evaluation Time")
print('Test set\n  Loss: {:0.3f}\n  Accuracy: {:0.3f}'.
    format(Evaluation[0], Evaluation[1]))
#endregion

#region: Model Prediction Testing
start = time.time()
Prediction = model.predict(test_sequences_matrix)
end = time.time()
print(end - start, "seconds - Prediction Time")

# Convert Prediction from one-hot Encoding to Integer
Prediction = Prediction.tolist()
for i, predic in enumerate(Prediction):
    Prediction[i] = functions.convert_vec_to_int(predic
        )
#endregion

#region: Confusion-Matrix
disp_labels = ['PS','CS', 'CNS', 'NS', 'COP', 'U']
fig = ConfusionMatrixDisplay.from_predictions(
    Y_test_old, Prediction, display_labels=disp_labels)
fig.plot(cmap="Greens")
figu = fig.ax_.get_figure()
figu.set_figwidth(8)
figu.set_figheight(6)
plt.title("KNN")
plt.tight_layout()
plt.show()
print(classification_report(Y_test_old, Prediction))
#endregion

color1="#007532"     # Color for Matplotlib Figures
color2="#73a771"     # Color for Matplotlib Figures
```

```
#region: Figure: Training and Validation Loss
plt.plot(history.history['loss'], label='
    Trainingsdatensatz', color=color2)
plt.plot(history.history['val_loss'], label='
    Testdatensatz', color=color1)
fig = plt.gcf()      # get current figure
fig.set_size_inches(10, 5)
plt.title('Verlauf der Lossfunktion', fontsize=fontsize
    )
plt.xlabel('Epochen', fontsize=fontsize2)
plt.xticks(np.arange(0, epochs, step=1), np.arange(1,
    epochs+1, step=1))
plt.ylabel('Loss', fontsize=fontsize2)
plt.legend(fontsize=fontsize2)
plt.grid(color='#BABABA', linewidth=0.5, linestyle=(0,
    (5, 10)))
plt.subplots_adjust(top=0.9, bottom=0.1, right=0.9,
    left=0.1)
plt.show()
#endregion

#region: Figure: Training and Validation Accuracy
plt.plot(history.history['accuracy'], label='
    Trainingsdatensatz', color=color2)
plt.plot(history.history['val_accuracy'], label='
    Testdatensatz', color=color1)
fig = plt.gcf()
fig.set_size_inches(10, 5)
plt.title('Verlauf der Accuracy', fontsize=fontsize)
plt.xlabel('Epochen', fontsize=fontsize2)
plt.xticks(np.arange(0, epochs, step=1), np.arange(1,
    epochs+1, step=1))
plt.ylabel('Accuracy', fontsize=fontsize2)
plt.legend(fontsize=fontsize2)
plt.grid(color='#BABABA', linewidth=0.5, linestyle=(0,
    (5, 10)))
```

```
plt.subplots_adjust(top=0.9, bottom=0.1, right=0.9,
    left=0.1)
plt.show()
#endregion
```