FernUniversität in Hagen
Fakultät für Mathematik und Informatik

AIG
Artificial Intelligence Group

Annotation von Zeitreihendaten mit Methoden des maschinellen Lernens

Masterarbeit

zur Erlangung des Grades *Master of Science (M.Sc.)*
im Studiengang Praktische Informatik

vorgelegt von
Fabian Nesper

Erstgutachter:	Prof. Dr. Matthias Thimm Artificial Intelligence Group
Betreuer:	Dr.-Ing. Ioannis Moisidis Mercedes-Benz AG

Erklärung

Ich erkläre, dass ich die Masterarbeit selbstständig und ohne unzulässige Inanspruchnahme Dritter verfasst habe. Ich habe dabei nur die angegebenen Quellen und Hilfsmittel verwendet und die aus diesen wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht. Die Versicherung selbstständiger Arbeit gilt auch für enthaltene Zeichnungen, Skizzen oder graphische Darstellungen. Die Masterarbeit wurde bisher in gleicher oder ähnlicher Form weder derselben noch einer anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht. Mit der Abgabe der elektronischen Fassung der endgültigen Version der Masterarbeit nehme ich zur Kenntnis, dass diese mit Hilfe eines Plagiatserkennungsdienstes auf enthaltene Plagiate geprüft werden kann und ausschließlich für Prüfungszwecke gespeichert wird.

Der Veröffentlichung dieser Arbeit auf der Webseite des Lehrgebiets Künstliche Intelligenz und damit dem freien Zugang zu dieser Arbeit stimme ich ausdrücklich zu.

Für diese Arbeit erstellte Software wurde quelloffen verfügbar gemacht, ein entsprechender Link zu den Quellen ist in dieser Arbeit enthalten. Gleiches gilt für angefallene Forschungsdaten.

Stuttgart, 30.11.2025 *Fabian Vesper*

(Ort, Datum) (Unterschrift)

Zusammenfassung

Die wachsende Menge unannotierter Zeitreihendaten in industriellen Anwendungen führt zu einem hohen manuellen Annotation Aufwand und erschwert den Einsatz überwachter Lernverfahren. Da manuelle Annotationen zeitintensiv sind und Kosten verursachen, unterstützen aktuelle Ansätze den Prozess durch Modelle und grafische Benutzeroberflächen, doch der Einfluss automatisierter Trainingsphase ist bislang kaum untersucht.

Diese Arbeit adressiert die Frage, ob regelbasierte Strategien und modellgestützte Vorhersagen den manuellen *Labeling*-Aufwand verringern und in welchem Umfang ein Modell mit autonomer Trainingsphase leistungsfähig bleibt.

Hierfür wurde ein zweiphasiger iterativer modellgestützter Prozess entwickelt, in dem Regeln aus Trainingsdaten abgeleitet und auf Testdaten geprüft wurden. Zudem wurden Fehlvorhersagen erfasst und die Modellleistung bewertet, ergänzt durch einen Vergleich zwischen einem Modell mit autonomen Trainingsphasen und einem vollständig überwachten Modell.

Die Ergebnisse zeigen, dass Regeln und modellgestützte Vorhersagen den manuellen Aufwand reduzieren können, wobei die Wahl geeigneter Regeln entscheidend ist. Ein Modell mit autonomen Trainingsphasen bleibt stabil, erreicht jedoch keinen deutlichen Lernfortschritt. Besonders zuverlässig sind Trainingsstrategien, die vollständige Dateien vorhersagen und das Modell anschließend neu initialisieren und erneut trainieren.

Ein vollständiger Verzicht auf korrekte *Labels* erscheint nicht sinnvoll, doch zukünftig könnte untersucht werden, ob hohe Vorhersagewahrscheinlichkeiten genutzt werden können, um selektiv *Labels* zu übernehmen und den manuellen Aufwand weiter zu verringern.

Abstract

The growing volume of unannotated time series data in industrial applications leads to substantial manual annotation efforts and complicates the use of supervised learning methods. Since manual annotations are time consuming and costly, current approaches employ models and graphical unser interfaces to support the process, althourgh the impact of automated training phases has so far received little attention.

This work addresses whether rule based strategies and model assisted predictions can reduce the manual labeling effort and to what extent a model with an autonomous training phase remains effective.

For this purpose, a two phase iterative model assisted process was developed in which rules were derived from training data and evaluated on test data. In each iteration, incorrect predictions were recorded and the model performance was assessed, followed by a comparison between a model with autonomous training phases and a fully supervised model.

The results show that rules and model assisted predictions can reduce manual effort, although the choice of suitable rules is crucial. A model with autonomous training phases remains stable but exhibits no clear learning progress. The most reliable outcomes are achieved by strategies that predict complete files and then reinitialize and retrain the model.

A complete omission of correct labels does not appear feasible, yet future research may investigate whether high prediction probabilities can be leveraged to selectively accept labels and further reduce manual effort.

Inhaltsverzeichnis

1 Einleitung

Die kontinuierlich steigende Verfügbarkeit großer Datenmengen in industriellen Anwendungen hat in den vergangenen Jahren dazu geführt, dass Methoden des maschinellen Lernens in immer mehr Bereichen eingesetzt werden. Dies betrifft unter anderem die Medizin [11, 46, 50], das Internet der Dinge (*Internet of Things*, IoT) [38], autonome Fahrfunktionen [7, 24, 32] und Finanzanwendungen [48]. In all diesen Feldern spielen umfangreiche Zeitreihendaten eine zentrale Rolle, da sie es ermöglichen, Zustände über die Zeit hinweg zu analysieren und Entscheidungen auf Grundlage der Daten zu unterstützen.

Mit dem wachsenden Einsatz datenverarbeitender Systeme rücken grundlegende Herausforderungen in den Vordergrund. Zu diesen zählen sehr große Datenmengen, die häufig automatisiert und in hoher Geschwindigkeit erzeugt werden [38]. Viele Anwendungen arbeiten zudem mit unausgewogenen Zeitreihen, in denen relevante Ereignisse nur selten auftreten und durch stark überrepräsentierte Normalzustände überlagert werden. Dies zeigt sich beispielsweise in der medizinischen Anfallserkennung [50] oder bei der Detektion von kritischen Verkehrssituationen [49], in denen nur ein kleiner Teil der Daten tatsächliche Zielereignisse umfasst. Gleichzeitig liegen Zeitreihen in vielen industriellen Bereichen überwiegend ohne Annotationen vor, wodurch der Einsatz klassischer überwachter Lernmethoden erschwert wird und große manuelle Aufwände entstehen [38].

Um den Aufwand bei der Annotation von Zeitreihendaten zu verringern, wurden in verschiedenen Domänen graphische Benutzeroberflächen entwickelt, die eine effiziente und qualitätsgesicherte Datenprüfung ermöglichen. Dazu zählt die MaD GUI, die als generisches und domänenunabhängiges Werkzeug für die Annotation und Analyse von Zeitreihen entwickelt wurde und sowohl manuelle Annotationen als auch die Einbindung eigener Algorithmen unterstützt [35]. In der medizinischen Forschung wurden spezialisierte Werkzeuge entwickelt, darunter die GUI von Deberneh und Sadygov [11], das der visuellen Validierung und Qualitätsprüfung von Zeitreihen des *Proteinturnover* dient. Ergänzend hierzu wurde mit PatchSorter [46] ein auf *Deep-Learning* basierendes Tool für die digitale Pathologie entwickelt, das die effiziente gruppenbasierte Annotation großer Mengen histologischer Objekte ermöglicht.

Parallel zu diesen domänenübergreifenden Entwicklungen wächst in der Automobilbranche die Relevanz datenbasierter Methoden besonders stark. Moderne Fahrzeuge umfassen zahlreiche Steuergeräte und Sensoren, die permanent umfangreiche Messdaten austauschen. Mit zunehmender Vernetzung und dem Fortschritt in Richtung automatisierter und autonomer Fahrfunktionen steigt die Menge der erzeugten Daten weiter an, was neue Möglichkeiten der Analyse, aber auch neue Anforderungen an Datenverarbeitung und *Labeling* schafft [7].

Gleichzeitig hat sich gezeigt, dass rekurrente neuronale Netze wie das *Long Short-Term Memory* (LSTM) Modell besonders gut zur Verarbeitung zeitlicher Abhängigkeiten geeignet sind. Diese Modelle erfassen längerfristige Muster und Trends, wodurch sie in zahlreichen Zeitreihenanwendungen erfolgreich eingesetzt werden [48].

Eine aktuelle Studie hat untersucht, wie sich menschliche Eingriffe in den *Labeling*-Prozess auf die Modellgüte auswirken. Die simulationsbasierte Untersuchung von Wang et al. [47] analysiert, unter welchen Bedingungen interaktive *Label*-Korrekturen zu Leistungssteigerungen von Modellen führen und wie sich der Aufwand menschlicher Korrekturen zu den erzielbaren Verbesserungen verhält. Die Ergebnisse zeigen, dass zusätzliche Korrekturen nicht in jedem Fall zu proportionalen Leistungsgewinnen führen und dass der Nutzen mit zunehmendem Umfang an Korrekturen allmählich abnimmt. Gleichzeitig wird ein optimaler Bereich beschrieben, in dem das Verhältnis von Aufwand und Qualitätsgewinn günstig ist. Die Studie verdeutlicht damit, dass selbst ein idealisierten Szenarien mit perfekten Korrekturen eine systematische Betrachtung des Kosten Nutzen Verhältnisses notwendig bleibt.

Auf Grundlage der von Wang et al. [47] dargestellten Ergebnissen wird deutlich, dass Verfahren erforderlich sind, die den manuellen Aufwand bei der Annotation von Zeitreihendatensätze gezielt reduzieren und zugleich die Auswirkungen potenziell fehlerhafter *Labels* auf die Modellgüte transparent machen. Vor diesem Hintergrund entsteht die zentrale Fragestellung dieser Arbeit, wie sich ein iterativer, modellgestützter *Labeling*-Prozess gestalten lässt, der sowohl automatische Vorhersagen als auch nachträgliche menschliche Korrekturen einbindet und dabei den manuellen Aufwand deutlich verringert. Die Arbeit untersucht daher, in welchem Umfang ein Modell ab einer bestimmten Generalisierung auf Testdaten weitertrainiert werden kann, ohne das alle Vorhersagen manuell überprüft werden müssen, und welche Effekte dies auf die Qualität der erzeugten *Labels* hat.

2 Theoretischer Hintergrund

In diesem Kapitel werden grundlegende Konzepte eingeführt, die für das Verständnis der vorliegenden Arbeit zentral sind. Den Auftakt bildet die Einordnung der Lernparadigmen des Maschinellen Lernens. Es folgen Datenvorverarbeitungsschritte, die die Qualität von Modellvorhersagen beeinflussen. Darauf aufbauend werden *Labeling*-Strategien vorgestellt, wie manuelles-, regelbasiertes *Labeling*, modellgestütztes *Labeling* und *Active-Learning* (Aktives Lernen). Es folgt ein Überblick über das Thema *Deep-Learning* und abschließend über verschiedene Evaluationsmetriken.

2.1 Methoden des Maschinellen Lernens

Der folgende Abschnitt stellt drei Lernformen des maschinellen Lernens vor. Die Einteilung erfolgt anhand der Verfügbarkeit von Zielwerten (Labels) und der Art des Lernziels. Es wird zwischen unüberwachtem (*unsupervised*), überwachtem (*supervised*) und teilüberwachtem (*semi-supervised*) Lernen unterschieden.

2.1.1 Unüberwachtes Lernen

Unüberwachtes Lernen beschreibt ein Verfahren, bei der ein Algorithmus ohne vorgegebene Zielvariable aus unbeschrifteten Daten (keine *Labels*) eigenständig Muster, Strukturen oder Zusammenhänge erkennt. Ziel ist es, verborgene Strukturen innerhalb der Daten explorativ zu identifizieren und dadurch Ähnlichkeiten, Gruppenbildungen oder latente Repräsentationen aufzudecken, ohne dass dem Lernprozess bekannte Ausgabewerte als Orientierung vorgegeben werden.

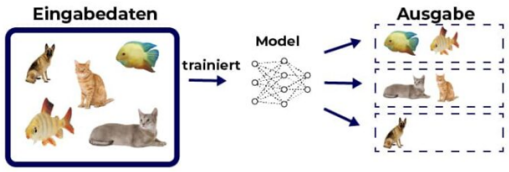

Abbildung 1: Beispielhafte Darstellung des unüberwachten Lernens [10].

Ein anschauliches Beispiel für unüberwachtes Lernen ist in Abbildung 1 dargestellt. Das Modell erhält ausschließlich Eingabedaten ohne zugehörige Zielwerte und versucht selbständig, Ähnlichkeiten oder Muster in den Daten zu erkennen. In diesem Fall werden die Objekte nicht durch vorgegebenen Klassen beschrieben, sondern das Modell gruppiert ähnliche Elemente aufgrund gemeinsamer Merkmale, wie Form oder Farbe, in eigene Cluster. Auf diese Weise können Zusammenhänge innerhalb der Daten erkannt werden, ohne dass eine explizite Beschriftung erforderlich ist.

Eine Methode des unüberwachten Lernen ist das *Clustering*, also die Gruppierung von Datenpunkten anhand von Ähnlichkeit oder Dichte. Partitionierende Ansätze wie *k-means* minimieren die Summe der quadrierten Abstände zu Clusterzentren und erfordern die Wahl der Clusterzahl *k*. Dichtebasierte Verfahren wie *DBSCAN* identifizieren stark besiedelte Regionen und markieren Ausreißer. Hierarchische Verfahren erzeugen Dendrogramme, während modellbasierte Ansätze wie *Gaussian Mixture Models* (GMM) [14] weiche Clusterzuweisungen erlauben. Weiche Zuordnungen liefern für jeden Datenpunkt Wahrscheinlichkeiten über mögliche Cluster. Beim Clustering von Zeitreihen müssen jedoch zeitliche Abhängigkeiten berücksichtigt werden, da benachbarte Werte häufig korreliert sind und eine reine euklidische Distanz oft keine sinnvollen Ergebnisse liefert.

Die Stärken unüberwachter Verfahren liegen in der Unabhängigkeit von *Labels* und der Fähigkeit, unbekannte Strukturen in großen Datensätzen sichtbar zu machen.

2.1.2 Überwachtes Lernen

Überwachtes Lernen liegt vor, wenn für jedes Eingabeobjekt $x_i \in X$ ein zugehöriger Zielwert $y_i \in Y$ existiert. Ziel des Lernprozesses ist es, eine Abbildungsfunktion

$$f : X \to Y$$

zu erlernen, die die Eingabedaten X möglichst genau auf die zugehörigen Ausgabewerte Y abbildet. Abhängig von der Art der Zielvariable y wird zwischen Klassifikation und Regression unterschieden. Bei der Klassifikation nimmt y diskrete Werte an, wie $y \in \{0, 1\}$ oder $y \in \{Katze, Hund\}$. Bei der Regression stellt y eine stetige Zielgröße dar, etwa Temperatur- oder Gewichtsangabe.

Im Gegensatz zum unüberwachten Lernen besitzen die Vorhersagen eine semantische Bedeutung, da sie an bekannte *Labels* gekoppelt sind. Für eine objektive Beurteilung des Lernverfahrens werden die Daten in Trainings- und Testmengen aufgeteilt. Der Trainingsdatensatz dient dem Lernen der Zuordnungsfunktion, während der Testdatensatz zur Bewertung der Modellgüte verwendet wird. Bei Zeitreihen ist dabei zwingend auf die zeitliche Ordnung zu achten, um das Einfließen zukünftiger Informationen zu verhindern.

Ein Beispiel für ein nichtparametrisches Verfahren ist *k-Nearest Neighbors* (kNN) [31]. Diese Methode ist instanzbasiert, dass bedeutet sie besitzt keine eigentliche Trainingsphase. Stattdessen werden die Trainingsdaten gespeichert und bei einer Vorhersage werden die k nächsten Nachbarn anhand eines Distanzmaßes bestimmt. In der Klassifikation entscheidet in der Regel die Mehrheitsklasse, gegebenenfalls distanzgewichtet. In der Regression ist es der gewichtete Mittelwert (*Mean*) der Zielwerte. Die Wahl von k ist entscheidend, denn kleine *k* Werte reagieren stark auf Rauschen und neigen zu *Overfitting* (Überanpassung), während große Werte

lokale Unterschiede glätten und insbesondere bei unausgeglichenen Klassen die Mehrheitsklasse bevorzugen. *Overfitting* bezeichnet den Zustand, in dem ein Modell sich zu stark an die Trainingsdaten anpasst und dabei Muster lernt, die nicht auf neue, unbekannte Daten übertragbar sind. Das Gegenstück dazu ist *Underfitting* (Unteranpassung), bei dem das Modell die zugrunde liegenden Zusammenhänge der Trainingsdaten nicht ausreichend erfasst und dadurch auch auf den Trainingsdaten nur eine geringe Leistung zeigt.

2.1.3 Teilüberwachtes Lernen

Teilüberwachtes Lernen kombiniert gelabelte mit ungelabelten Daten. Ziel ist es, den Beschriftungsaufwand zu reduzieren und dennoch eine gute Generalisierungsfähigkeit zu erreichen. Die Generalisierungsfähigkeit beschreibt die Fähigkeit eines Modells, auch auf bisher unbekannten Daten, korrekte Vorhersagen zu treffen, anstatt lediglich die Trainingsdaten auswendig zu lernen [17].

Typischerweise wird zunächst ein Modell auf den gelabelten Daten trainiert, das anschließend Vorhersagen auf den nicht annotierten Daten erzeugt. Diese Vorhersagen dienen als sogenannte *Pseudo-Labels*. *Pseudo-Labels* sind Modellvorhersagen, die wie echte *Labels* behandelt werden, deren Korrektheit jedoch nicht vollständig garantiert ist. Nur Daten mit hoher Modellkonfidenz werden in das weitere Training übernommen. Dieses Vorgehen eignet sich insbesondere, wenn die Annotation von Daten kostenintensiv oder zeitaufwendig ist, während gleichzeitig große Mengen ungelabelter Daten vorliegen. Dadurch lässt sich die effektive Trainingsmenge erweitern und die Modellleistung verbessern, ohne den menschlichen Aufwand proportional zu erhöhen.

2.2 Labeling-Strategien und ein Auswahlverfahren

Labeling-Strategien umfassen Verfahren, mit denen ungekennzeichnete Daten verlässlich mit Zielwerten versehen werden. Ziel ist ein hochwertiger und ausreichend großer Datensatz für die nachgelagerte Nutzung, wie zum Beispiel für die Analyse. Es werden drei verschiedene Ansätze betrachtet. Beim manuellen *Labeling* beurteilt ein Mensch die Daten anhand von Domänenwissen. Regelbasiertes *Labeling* leitet aus Beispielen mathematische Regeln ab oder nutzt Fachwissen. Modellgestütztes *Labeling* verwendet ein trainiertes Modell. Zusätzlich wird eine Auswahlstrategie vorgestellt. *Active-Learning* steuert die Auswahl der zu annotierenden Daten und priorisiert bestimmte Daten.

2.2.1 Manuelles Labeling

Manuelles *Labeling* bezeichnet die Vergabe von Klassen oder Zielwerten durch eine oder mehrere fachkundige Personen. Ein typisches Beispiel findet sich in der Bildklassifikation. Fachkundige Annotatoren sichten eine große Menge von Bildern und

ordnen jedem Bild ein oder mehrere *Labels* zu, etwa „Hund“ oder „Katze“. Grundlage hierfür sind eindeutige Definitionskataloge, die festlegen, welche visuellen Merkmale zur jeweiligen Klasse gehören. In komplexeren Szenarien erfolgt zusätzlich eine regionenbasierte Annotation, bei der Objekte innerhalb des Bildes durch Rahmen oder Masken abgegrenzt und mit spezifischen Klassen versehen werden. Auch bei Audiodaten spielt die manuelle Annotation eine zentrale Rolle. Dabei markieren Expertinnen und Experten in einer Zeitachse jene Intervalle, in denen bestimmte Ereignisse auftreten, etwa das Starten eines Motors, das Sprechen einer Person oder das Auftreten eines Fehlgeräusches. Diese Intervalle werden anschließend mit entsprechenden Kennzeichnungen versehen. Die Aufgabe erfordert hohe Konzentration, da die Unterscheidung zwischen ähnlichen Klangmustern oder Hintergrundgeräuschen häufig nur mit Fachwissen und Erfahrung möglich ist.

Die manuelle Beschriftung ist äußerst zeitaufwendig, da die durchschnittliche Annotationszeit pro Datensatz stark von Faktoren wie dem verwendeten Tool, der Anzahl der zu annotierenden Elemente und der Datenqualität abhängt. Nach Angaben von [42] kann das *Labeln* von rund 100.000 Bildern mit jeweils fünf Anmerkungen bis zu 1.5000 Arbeitsstunden in Anspruch nehmen.

Annotatoren müssen geschult und auf gemeinsame Kriterien abgestimmt werden, damit die Entscheidungen konsistent ausfallen. Die Qualität wird fortlaufend überwacht, zum Beispiel durch Doppelannotation ausgewählter Abschnitte und die Messung der Übereinstimmung zwischen Annotatoren. Geeignete Kennzahlen sind zum Beispiel die prozentuale Übereinstimmung, Cohen Kappa [16, 33] für zwei Annotatoren und Fleiss Kappa [33] für mehrere Annotatoren.

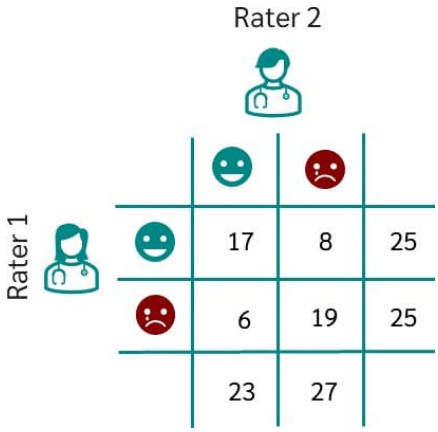

Abbildung 2: Kreuztabelle zur Berechnung des Cohen-Kappa-Koeffizienten [16]. Sie zeigt die Übereinstimmung zweier Bewerter bei der Klassifikation von Personen als glücklich oder traurig.

Mit dem Cohen-Kappa-Koeffizienten kann die Übereinstimmung von Bewertungen auf nominalem Skalenniveau quantifiziert werden. Er ist ein statistisches Maß zur Bestimmung des Grades der Übereinstimmung zwischen zwei Bewertern, die dieselben Objekte in Kategorien einordnen [16]. Angenommen, Personen sollen anhand ihres Gesichtsausdrucks als „glücklich“ (grünes Symbol) und „traurig“ (rotes Symbol) bewertet werden. Zwei Fachexperten (Rater) beurteilen dieselben 50 Personen, woraus sich die in Abbildung 2 dargestellte Kreuztabelle ergibt. Sie zeigt die Häufigkeiten der Übereinstimmungen und Abweichungen zwischen den beiden Ratern:

- Rater 1 glücklich und Rater 2 glücklich: Beide Rater bewerteten dieselben 17 Personen als glücklich.
- Rater 1 glücklich und Rater 2 traurig: Rater 1 beurteilte 8 Personen als glücklich, die von Rater 2 als traurig eingestuft wurden.
- Rater 1 traurig und Rater 2 glücklich: Rater 1 beurteilte 6 Personen als traurig, die von Rater 2 als glücklich eingestuft wurden.
- Rater 1 traurig und Rater 2 traurig: Beide Rater bewerteten dieselben 19 Personen als traurig.
- Randwerte: Rater 1 bewertete 25 Personen als glücklich und 25 als traurig, während Rater 2 23 Personen als glücklich und 27 als traurig kennzeichnete.

Mit diesen Werten kann der Cohen-Kappa-Koeffizient k berechnet werden:

$$k = \frac{p_o - p_e}{1 - p_e}$$

Der Anteil der tatsächlichen Übereinstimmungen p_o beträgt:

$$p_o = \frac{17 + 19}{50} = 0.72$$

Die Wahrscheinlichkeit einer zufälligen Übereinstimmung p_e ergibt sich zu:

$$p_o = \frac{25}{50} \times \frac{23}{50} + \frac{25}{50} \times \frac{27}{50} = 0.5$$

Daraus folgt:

$$k = \frac{0.72 - 0.5}{1 - 0.5} = 0.44$$

Zur Einordnung des berechneten Wertes kann die Klassifikation nach Landis und Koch (1977) [29] herangezogen werden. Sie dient als Orientierung zur qualitativen Bewertung der Stärke der Übereinstimmung zwischen zwei Bewertern. Der im Beispiel berechnete Wert von $k = 0.44$ fällt in den moderaten Bereich und weist auf eine mittlere Übereinstimmung zwischen den Experten hin.

Die Berechnung von Kennzahlen wie Cohen-Kappa, bildet eine wichtige Grundlage für die Qualitätssicherung des manuellen *Labeling*-Prozesses. Sie ermöglicht, die Konsistenz zwischen Annotatoren objektiv zu bewerten und gezielt zu verbessern. Weichen die Übereinstimmungswerte stark voneinander ab, deutet das auf Unklarheiten im Leitfaden oder auf Interpretationsspielräume hin. In solchen Fällen erfolgt eine Anpassung der Richtlinien, die Klärung typischer Grenzfälle und die Überarbeitung von Beispielen im Definitionskatalog. Eine validierte Menge an Referenzfällen dient als verbindlicher Maßstab und wird regelmäßig erweitert.

In der praktischen Umsetzung werden im Allgemeinen zwei Organisationsformen unterschieden.

Bei der Einzelannotation werden die Daten unabhängig voneinander durch einzelne Annotatoren bearbeitet. Jeder Annotator erhält dabei klar definierte Abschnitte, Zeitfenster oder Datensegmente, die nach einem Leitfaden zu kennzeichnen sind. Diese Organisationsform ermöglicht eine hohe Parallelisierung und damit eine deutlich größere Bearbeitungsgeschwindigkeit, da mehrere Personen gleichzeitig an unterschiedlichen Teilen des Datensatzes arbeiten können. Sie ist insbesondere in frühen Phase oder bei großen Datenmengen effizient, wenn eine erste breite Grundannotation erforderlich ist. Zur Sicherung der Konsistenz ist jedoch eine regelmäßige Kalibrierung notwendig. Ausgewählte Abschnitte werden dabei doppelt annotiert und die Ergebnisse werden beispielsweise mit dem Cohen-Kappa-Koeffizienten verglichen. Abweichungen dienen der Überprüfung der Verständlichkeit der Richtlinien und fließen in die Anpassung ein. Ein Nachteil ist, dass ohne Rücksprache individuelle Interpretationen stärker variieren können. Dem wirken jedoch regelmäßige Sitzungen entgegen, in denen Uneinheitlichkeiten besprochen und ein gemeinsames Verständnis der Labeldefinitionen sichergestellt werden.

In einer gemeinsamen Runde werden Abschnitte synchron begutachtet, strittige Fälle diskutiert und ein Konsens festgehalten. Dies erhöht die Konsistenz und verbessert den Leitfaden, ist jedoch langsamer und bindet mehr Fachpersonen. Es besteht die Gefahr ausufernder Diskussionen und verzögerter Entscheidungen. Dem wirkt eine strukturierte Moderation mit klarer Agenda, festen Zeitgrenzen und verbindlichen Entscheidungsregeln entgegen. In beiden Formen unterstützen eine benutzerfreundliche Oberfläche, eine saubere Protokollierung mit Zeitstempel und Bearbeiter sowie die Möglichkeit, Unsicherheiten zu markieren und Entscheidungen später zu verifizieren.

Der zentrale Vorteil des manuellen *Labelings* liegt in sorgfältig begründeten Kennzeichnungen, die auch seltene Muster zuverlässig erfassen und später nachvollziehbar sind. Grenzen ergeben sich durch den hohen Zeit- und Kostenaufwand, da jeder Datenpunkt oder jedes Fenster einzeln betrachtet werden muss und Daten häufig schneller entstehen als sie gelabelt werden können. Hinzu kommen Ermüdung

und mögliche Unterschiede zwischen Annotatoren, was Konzentration, Pausenplanung, klare Zuständigkeiten und eine kontinuierliche Qualitätssicherung erforderlich macht. Diese Nachteile lassen sich abmildern durch kurze Annotationseinheiten, regelmäßige Abstimmungstermine und verbindliche Regeln für Konfliktlösung.

2.2.2 Regelbasiertes Labeling

Regelbasiertes *Labeling* weist Daten mithilfe expliziter „Wenn Dann“ Regeln vorläufige Zielwerte zu. Die Regeln entstehen aus Domänenwissen und aus Kennzahlen, die das normale Verhalten beschreiben. Schwellen können beispielsweise mit Quantil, Mittelwert, Minimum und Maximum bestimmt werden. Damit die Regeln robuste Muster erfassen, berücksichtigen sie verschiedene Darstellungen der Daten wie Rohwerte, Beträge der Rohwerte, Gradienten und Beträge der Gradienten. Ergänzend können Fensterstatistiken wie der Durchschnitt, Varianz oder der Anteil überschrittener Werte verwendet werden, um einen lokalen Kontext abzubilden. In der praktischen Anwendung werden die Regeln an einem Referenzdatensatz entworfen, visualisiert und iterativ verfeinert. Die so erzeugten Zuordnungen sind bewusst nicht endgültig. Sie dienen als sogenannte *Pseudo-Labels* und müssen in einem nächsten Schritt durch manuelles *Labeling* geprüft und bei Bedarf korrigiert werden. Auf diese Weise liefert das regelbasierte Verfahren schnelle Vorschläge, während die finale Verantwortung beim Menschen bleibt.

Als Beispiel kann eine Regel verwendet werden, die Referenzwerte, wie den Durchschnittswert des Normalzustandes $\bar{x}_{Normal}$ und Fehlerzustand $\bar{x}_{Fehler}$ verwendet. Es wird überprüft, ob ein Wert x näher am Normal- oder am Fehlerzustand liegt. Dafür wird der Abstand des Wertes x zu beiden Referenzwerten mit der Formel

$$d = \min(|x - \bar{x}_{Normal}|, |x - \bar{x}_{Fehler}|)$$

bestimmt. Liegt der Wert näher am Fehlerdurchschnitt, wird der Abschnitt als anomal gekennzeichnet, andernfalls als normal. Durch die Wahl geeigneter Referenzwerte und einer Schwelle für die Distanz, lässt sich die Empfindlichkeit der Regel steuern und an verschiedene Datensätze anpassen.

Der Vorteil liegt in Transparenz, Nachvollziehbarkeit und sofortiger Verfügbarkeit von Vorschlägen. Die Regeln sind leicht zu erklären, benötigen kein Training und lassen sich zügig auf große Datenmengen anwenden. Grenzen entstehen durch begrenzte Flexibilität gegenüber neuen Mustern, Empfindlichkeit gegenüber *Data Drift*, also Veränderungen in den zugrunde liegenden Datenverteilungen, die im Trainingsprozess nicht berücksichtigt wurden, wodurch sich die Modellleistung im Zeitverlauf verringern kann [5], sowie durch die eingeschränkte Abbildung von Wechselwirkungen zwischen Variablen. Regelbasiertes *Labeling* sollte daher nicht allein verwendet werden, sondern im Zusammenspiel mit manueller Prüfung. Eine gute Praxis umfasst klare Definitionen der Regeln, Dokumentation von Gültigkeitsbereichen sowie eine regelmäßige Anpassung der Regeln mit den neu erworbenen

Daten. Zudem ist es sinnvoll, die Trefferrate, den Anteil an Fehlalarmen und die Abdeckung zu überwachen und strittige Fälle systematisch in das Set der Referenzbeispiele aufzunehmen. So entsteht ein verlässlicher Vorschlagsmechanismus, der die nachgelagerte manuelle Annotation wirksam entlastet und eine solide Grundlage für weitere Analysen schafft.

2.2.3 Modellgestütztes Labeling

Modellgestütztes *Labeling* umfasst zwei Arten des *Labelings*, den unüberwachten und überwachten Fall.

Im unüberwachten Fall werden keine Zielwerte benötigt. Das Modell versucht eine Struktur in den Daten zu lernen und Gruppen zu definieren, deren Bedeutung erst durch nachgelagerte Interpretation festgelegt wird. Jedoch kann die Anzahl der Gruppen unterschiedlich zu den wahren Klassen sein. Geeignete Verfahren sind *Support Vector Machines* (SVM) [45] und *Clustering* [44] im Merkmalsraum.

Im überwachten Fall stehen bestätigte *Labels* bereit. Das Modell wird mit den bereits gelabelten Daten trainiert. Anschließend kann das Modell für jede Zeile der Daten eine Wahrscheinlichkeit ausgeben, die über einen Schwellenwert in *Labels* überführt wird. Die möglichen Labels entsprechen den im Training verwendeten *Labels*. Solange die Vorhersagen vom Modell nicht gut genug sind, müssen diese mithilfe von manuellem *Labeling* korrigiert werden. Die korrigierten *Labels* fließen regelmäßig in das Training zurück.

Unüberwachte Verfahren sind schnell einsatzbereit und liefern frühe Vorschläge, auch ohne Annotationen, jedoch können die Vorschläge deutlich abweichen und haben mitunter nur begrenzten Nutzen, weshalb eine manuelle Prüfung nötig bleibt. Überwachte Verfahren nutzen vollständige Daten, also mit *Labels* und lernen daraus. Die Qualität der Vorhersagen ist jedoch nur so gut, wie die Qualität der Trainingsdaten und reagieren empfindlich auf *Data Drift*.

In der Praxis kann mit einem unüberwachten Verfahren gestartet werden, um *Pseudo-Labels* zu gewinnen, auch wenn diese nicht den wahren *Labels* entsprechen. Die *Pseudo-Labels* werden vom Menschen korrigiert und ein überwachtes Modell trainiert. Dann kann das überwachte Verfahren verwendet werden und das Modell wird fortlaufend weitertrainiert.

2.2.4 Active-Learning

Active Learning stellt eine Auswahlstrategie im Rahmen der *Labeling*-Prozesse dar und adressiert die hohen Kosten und den Zeitaufwand manueller Beschriftung. Das Ziel besteht darin, aus einem großen Pool ungelabelter Daten genau jene Abschnitte zu identifizieren, die den größten Informationsgewinn für das Modell liefern.

Active-Learning generiert selbst keine *Labels*, sondern steuert die Auswahl der zu annotierenden Beispiele, während die tatsächliche Beschriftung durch einen Fachexperten oder ein Modell erfolgt.

Der Ablauf wiederholt sich in Zyklen, dargestellt in Abbildung 3. Zunächst wählt das Modell unter Anwendung von *Query-Strategien* (Abfragestrategien) eine Teilmenge aus, die für die Annotation am informativsten ist. Diese Daten werden anschließend manuell gelabelt und in den Trainingssatz aufgenommen. Das Modell wird daraufhin erneut trainiert und der Prozess beginnt von vorn.

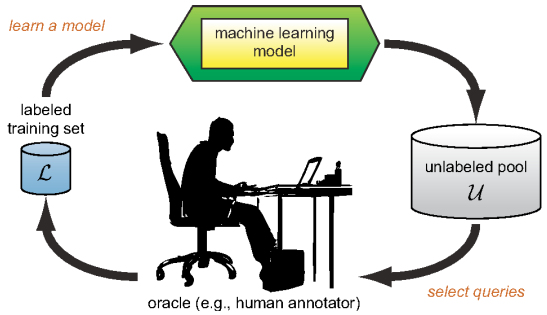

Abbildung 3: Schematische Darstellung des *Active-Learning*-Zyklus [41].

Query-Strategien [6] bewerten jeden Kandidaten nach dem erwarteten Informationsgewinn. Unsicherheitsbasierte Verfahren bevorzugen Datenpunkte, bei denen das Modell eine geringe Vorhersagezuversicht besitzt. Andere Ansätze, wie *Query by Committee* (QBC), messen die Uneinigkeit mehrerer Modelle und bevorzugen Daten, bei denen sie stark voneinander abweichen. Diversitätsbasierte Strategien stellen sicher, dass ausgewählte Beispiele repräsentativ für verschiedene Regionen des Merkmalsraums sind.

Active-Learning ergänzt klassische *Labeling*-Strategien, indem es den menschlichen Aufwand auf die informativsten Beispiele konzentriert. Die Kombination aus regelbasierten, modellgestützten und aktiven Auswahlmechanismen kann zu einem effizienten Kreislauf führen. Regeln markieren auffällige Abschnitte, *Active-Learning* priorisiert die aussagekräftigsten Beispiele, und manuell geprüfte *Labels* verbessern das Modell fortlaufend. Dadurch sinkt der Gesamtaufwand der Annotation, während die Qualität der Trainingsdaten und die Leistungsfähigkeit des Modells zunehmen [20].

2.3 Datenaufbereitung und Merkmalsbildung

Bevor Daten für Analysen oder für Modelltraining verwendet werden, müssen sie korrekt aufbereitet sein. Inkonsistenzen, Duplikate oder Lücken sind zu bereinigen, Einheiten zu harmonisieren und Zeitstempel zu prüfen. Dieser Abschnitt erläutert die Bausteine zur Sicherung der Konsistenz sowie Verfahren, mit denen Daten skaliert und erweitert werden, damit sie verlässlich weiterverwendet werden können. Das Kapitel erläutert die Datenbereinigung, Skalierung, Merkmalsbildung, Datenerweiterung und Datensatzaufteilung jeweils für sich.

2.3.1 Datenbereinigung

Datenbereinigung stellt Qualität und Konsistenz von Daten sicher und umfasst den Umgang mit fehlenden Werten, die Harmonisierung von Einheiten, das Entfernen von Duplikaten sowie die Prüfung von Zeitstempel und Takt in Zeitreihen.

Fehlende Werte treten in Zeitreihen in unterschiedlichen Formen auf, etwa als einzelne Ausfälle, als zusammenhängende Blöcke, als fehlende Zeitstempel oder als gleichzeitige Ausfälle mehrerer Variablen. Der Umgang damit richtet sich nach Ursache und Ausdehnung der Lücken. Ein bloßes Ignorieren ist nur dann vertretbar, wenn der nachgelagerte Algorithmus fehlende Einträge ausdrücklich verarbeiten kann.

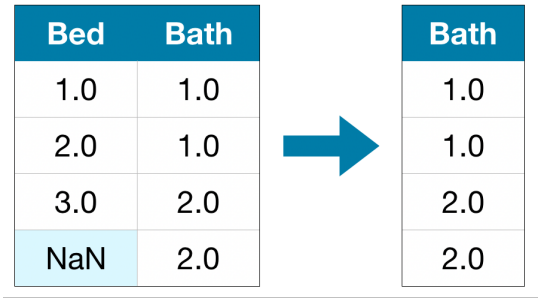

Abbildung 4: Imputation fehlender Werte durch Ersetzen mit dem Mittelwert [8].

Das vollständige Löschen betroffener Beobachtungen ist bei Daten ohne Zeitbezug oft unkritisch, führt bei Zeitreihen jedoch zu unterbrochenen Verläufen und zu inkonsistenten Abständen zwischen Zeitpunkten. In solchen Fällen ist zu prüfen, ob die Reihe segmentiert und die Auswertung auf zusammenhängende Abschnitte beschränkt werden sollte. Alternativ kommen Verfahren der Imputation in Betracht.

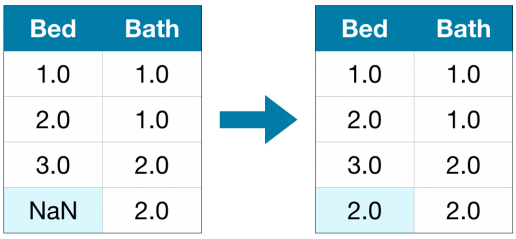

Abbildung 5: Erweiterte Imputation mit zusätzlicher Spalte zur Kennzeichnung fehlender Werte [8].

Kurze Lücken werden innerhalb eines engen Kontextfensters durch lokale Verfahren gefüllt, zum Beispiel durch lineare Interpolation, durch einen gleitenden Median oder durch das Fortschreiben des letzten plausiblen Wertes, bei langsam variierenden Signalen. Beim Auffüllen von längeren Lücken sollte berücksichtigt werden, dass künstliche Muster entstehen können.

Abbildung 4 bis 6 zeigen konkrete Beispiele für den Umgang mit fehlenden Werten. In Abbildung 4 wird die Variable „Bed“ entfernt, da sie einen fehlenden Eintrag enthält und damit keine verlässliche Information liefert. Abbildung 5 zeigt die Variante, bei der der fehlende Wert durch den Mittelwert

$$\bar{x}_{Bed} = \frac{1.0 + 2.0 + 3.0}{3} = 2$$

ersetzt wird. In Abbildung 6 wird der fehlende Wert durch eine Schätzung ergänzt und mit einer zusätzlichen Spalte „Bed_was_missing“ gekennzeichnet, um die Transparenz zu erhalten. Die Methoden unterscheiden sich in ihrer Komplexität und ihrem Einfluss auf die Datenqualität, bieten jedoch eine praxisnahe Darstellung der Datenbereinigung.

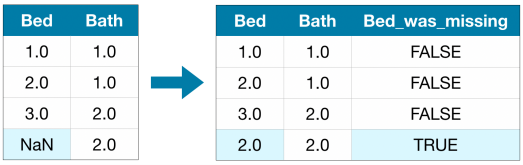

Abbildung 6: Beispiel für die Behandlung fehlender Werte durch Imputation mit zusätzlicher Kennzeichnung. Fehlende Werte in der Spalte *Bed* werden durch den Mittelwert ersetzt und gleichzeitig durch eine Indikatorspalte *Bed_was_missing* markiert, um Informationsverlust zu vermeiden [8].

Bei der Konsistenzprüfung werden Einheiten vereinheitlicht. Unterschiedliche Einheiten können sonst zu Pseudoausreißern führen. Ein Beispiel ist eine Messreihe in Zentimeter, in der einzelne Einträge fälschlich in Meter vorliegen. Ein Wertpaar wie

100 gefolgt von 1.5 Zentimeter wirkt auffällig, obwohl eigentlich 1.5 Meter gemeint sind. Solche Fälle werden durch einheitliche Einheiten und eine nachvollziehbare Umrechnung behoben, zum Beispiel von Umdrehungen pro Minute auf Umdrehungen pro Sekunde.

Auch kategoriale Merkmale müssen vereinheitlicht werden. Verschiedene Schreibweisen für dieselbe Ausprägung erschweren die spätere Auswertung und können zu stillen Fehlern führen. Tabelle 1 zeigt ein Beispiel, in dem sowohl das Merkmal Geschlecht als auch die numerischen Angaben unterschiedlich codiert sind. Die Angaben „Male“, „Männlich“ und „M“ beschreiben zwar dieselbe Ausprägung, unterscheiden sich jedoch in ihrer Schreibweise. Ähnlich weisen die Werte für Zeit und Strecke unterschiedliche Einheiten auf, etwa Sekunden, Minuten oder Meter und Kilometer. Solche Inkonsistenzen führen bei der automatisierten Verarbeitung zu Fehlinterpretationen und müssen vor der Analyse harmonisiert werden, beispielsweise durch einheitliche Umrechnungen und standardisierte Bezeichnungen.

Geschlecht	**Zeit**	**Strecke**
Male	12.4s	100 m
Männlich	0.15 min	0.1 km
M	13.5 s	0.1 km
Female	14.1 s	100 m
Weiblich	0.24 min	0.2 km
F	0.20 min	200 m

Tabelle 1: Daten mit den Spalten Geschlecht, Zeit und Strecke.

Ein weiteres Problem sind Duplikate. Eine Messung gilt als Duplikat, wenn sie dieselbe fachliche Identität wie eine bereits gespeicherte Beobachtung besitzt. In Zeitreihen wird diese Identität durch die Kombination aus Messungen und Zeitstempel bestimmt. Sind Schlüssel und Messwerte vollständig gleich, liegt ein exaktes Duplikat vor und einer der Einträge wird entfernt. Stimmen die Schlüssel überein, unterscheiden sich jedoch die Messwerte, handelt es sich um ein Duplikat mit Konflikt, das nach definierten Regeln zusammengeführt oder alternativ durch Verwerfen eines Eintrags bereinigt wird. Bei Daten ohne Zeitbezug wird die Identität über einen fachlich definierten Verbundschlüssel festgelegt, zum Beispiel eine Kombination aus Objektkennung und Datum.

Person	Anzahl der Bonbons
Peter	5
Sandy	20
Sandy	20
Joseph	3
Ahmed	12

Tabelle 2: Tabelle zur Darstellung von Duplikaten [9].

Tabelle 2 zeigt ein einfaches Beispiel für Duplikate in einem Datensatz. Die Person „Sandy“ taucht mit identischen Werten zweimal auf und stellt ein exaktes Duplikat dar. In der Praxis wird einer der beiden Einträge entfernt, um die Datenredundanz zu verringern und die Integrität des Datensatzes zu sichern. Wird das Duplikat nicht gelöscht, können die statische Auswertung und das Training eines Modells verfälscht werden.

2.3.2 Skalierung

Eine Skalierung passt Werte unterschiedlicher Größenordnungen an einen vergleichbaren Maßstab an. Sie erleichtert Optimierung und Vergleich, stabilisiert numerische Verfahren und verhindert, dass einzelne Merkmale aufgrund ihrer Skala den Lernprozess dominieren. Skalierung ist insbesondere für abstandsbasierte Methoden und modellgestützte Verfahren mit Gradienten von Bedeutung. Formal lässt sich Skalierung als Abbildung:

$$x_i \rightarrow x_i'$$

beschreiben, bei der eine Transformation auf jedes Merkmal angewendet wird, um die Werte in einen definierten Bereich oder auf eine standardisierte Skala zu bringen.

Ein Verfahren ist die Min-Max-Normalisierung [1], bei der jeder Wert x_i durch lineare Transformation in den Bereich $[0, 1)$ überführt wird:

$$x_i' = \frac{x_i - x_{min}}{x_{max} - x_{min}}.$$

x_{min} und x_{max} das Minimum und Maximum der betrachteten Merkmalswerte.

Ein weiteres Verfahren ist die Standardisierung, bei der die Werte zentriert und durch ihre Standardabweichung geteilt werden:

$$x_i' = \frac{x_i - \mu}{\delta}. \tag{1}$$

Wobei μ den Mittelwert und δ die Standardabweichung des Merkmals darstellen. Durch diese Transformation erhält das Merkmal einen Mittelwert von null und eine Varianz von eins.

Für Datensätze mit Ausreißern empfiehlt sich eine robuste Skalierung, bei der Median und *Median Absolute Deviation* (MAD) anstelle von Mittelwert und Standardabweichung verwendet werden:

$$x_i' = \frac{x_i - Median(x)}{MAD(x)}.$$

Dieses Verfahren reduziert den Einfluss extremer Werte und ist insbesondere für heterogene Zeitreihen geeignet.

Die Bestimmung der Skalierungsparameter erfolgt grundsätzlich auf den Trainingsdaten und wird unverändert auf Test- und Neudaten angewendet, um *Data Leckage* (Datenlecks) zu vermeiden. *Data Leckage* treten auf, wenn Informationen aus den Testdaten unbeabsichtigt in den Trainingsprozess einfließen. Dadurch entsteht eine künstlich erhöhte Modellleistung, die in der realen Anwendung nicht reproduzierbar ist. Eine strikte Trennung der Datensätze und die ausschließliche Anpassung der Skalierung an die Trainingsdaten verhindern diesen Effekt.

2.3.3 Feature Engineering

Feature Engineering bezeichnet die gezielte Ableitung und Transformation von Merkmalen aus Rohdaten. Es werden Informationen in einer Form bereitgestellt, die Lernverfahren und analytische Auswertungen wirksam nutzen können. Dazu werden neue Größen erzeugt und vorhandene Werte in zusätzliche Repräsentationen überführt.

Dies findet Anwendung, wenn Rohdaten allein nicht ausreichen, um Klassen sauber zu trennen oder Anomalien verlässlich zu erkennen. Geeignete Merkmale erhöhen die Trennbarkeit, stabilisieren Modelle gegenüber Ausreißern und *Data Drift* und können sogar den Bedarf an gelabelten Daten senken.

Es werden Ableitungen gebildet, etwa der Gradient, sowie Beträge der Rohwerte und der Gradienten, um Richtungswechsel und Stärke von Änderungen besser abzubilden. Fensterbasierte Statistiken erfassen statistische Merkmale wie Mittelwert, Median, Varianz, Minimum, Maximum und Quantile in einem festen Intervall. Zusätzlich kann die Anzahl der Vorzeichenwechsel oder Lauflängen in positiver oder negativer Richtung gezählt werden.

Aus einem ursprünglichen Merkmal x_{raw} können durch Transformation folgende Merkmale entstehen:

$$x_{raw_{abs}} = |x_{raw}|,$$

$$x_{grad} = x_{raw,i+1} - x_{raw,i},$$

$$x_{grad_{abs}} = |x_{grad}|.$$

Der Ausdruck $x_{raw_{abs}}$ beschreibt den Betrag des Rohsignals, $|x_{grad}|$ den zeitlichen Gradienten, also die Änderung des Wertes zwischen zwei aufeinanderfolgenden Zeitpunkten und $x_{grad_{abs}}$ den Betrag dieser Änderung.

Neben den erwähnten Vorteilen, dass durch die Merkmalserweiterung mehr Informationen abgeleitet werden, entstehen erhöhter Speicherbedarf und Rechenzeit, da aus einem Merkmal mehrere Merkmale werden können.

2.3.4 Klassen-/Label-Balancierung

Nicht allein die Menge der Daten entscheidet über den Erfolg, sondern ihre effektive Nutzung. Ein starkes Ungleichgewicht der Klassen führt dazu, dass ein Modell mit hoher scheinbarer Genauigkeit, die dominante Klasse bevorzugt und seltene Ereignisse kaum erkennt.

Für das folgende Beispiel bezeichnet Klasse 0 normale Zeilen und Klasse 1 fehlerhafte Zeilen. Angenommen, ein Datensatz enthält 900 normale und 100 fehlerhafte Zeilen. Ein Modell, das stets die Klasse 0 vorhersagt, erreicht eine scheinbare Genauigkeit von 90%, weil 900 von 1000 Entscheidungen korrekt sind. Für die Fehlerklasse 1, ist die Leistung jedoch gleich Null. Der *Recall* (Rückruf) der positiven Klasse beträgt $\frac{0}{100} = 0$. Die *Precision* (Präzision) ist mangels positiver Vorhersagen nicht definiert und wird in der Praxis als Null gesetzt.

Daraus folgt, dass beim Training und bei der Bewertung das Klassenverhältnis stets berücksichtigt werden muss. Informative Metriken sind *Precision* und *Recall*. Vor jeder Trainingsrunde ist zu prüfen, ob beide Klassen in ausreichender Zahl vertreten sind. Ist dies nicht der Fall, wird das Training zurückgestellt, bis zusätzliche Daten vorliegen und das Gleichgewicht gegeben ist. Die Testdaten werden einmalig festgelegt und bleiben unverändert. Das Klassenverhältnis wird bei der Zusammenstellung des Testsatzes berücksichtigt.

2.3.5 Datenerweiterung

Aufbauend auf Unterkapitel 2.3.4 zur Klassen- und *Label*-Balancierung, wird hier nicht erneut auf die Ursache des Ungleichgewichts eingegangen. Im Fokus steht, dass bei einem Ungleichgewicht zwischen Normalzuständen und Fehlerzuständen nur wenige effektive Trainingsbeispiele vorliegen. Durch Datenerweiterung werden aus bestehenden Beobachtungen plausible Varianten erzeugt oder synthetische Beispiele generiert, die zusätzlich für das Training genutzt werden. Dadurch wächst die Menge der Trainingsdaten, das Modell sieht mehr Varianz und die Vorhersagegüte kann steigen. Wichtig ist, dass die *Label*-Semantik erhalten bleibt und erweiterte Beispiele ausschließlich im Training eingesetzt werden, während die Testdaten unverändert bleiben. Bei Zeitreihen ist der zeitliche Kontext zentral. Erzeugte Beobachtungen müssen den Verlauf und die Reihenfolge der Ereignisse realistisch wiedergeben.

Als Arten der Datenerweiterung bewähren sich Skalierung, kontrollierte Störungen und synthetische Datenerzeugung. Skalierung und Störung teilen sich die Multiplikation mit einem Faktor α. Es sind einfache und wirksame Formen der Datenerweiterung. Gegeben ist ein Datensatz $d(x)$, erzeugt wird eine Variante

$$d'(x) = \alpha \times d(x),$$

wobei α ein Faktor nahe Eins ist. Bei der Skalierung ist der Faktor für den ganzen Datensatz konstant. Bei der Störung hingegen ist der Faktor variabel und kann sich über die Zeit hinweg verändern. Die Operationen verändern die Amplitude, lassen Form, Reihenfolge und zeitliche Struktur der Muster unverändert und können somit reale Kalibrierabweichungen oder leichte Verschiebungen abbilden. Die Wahl des Faktors erfolgt für definierte Bereiche zufällig nahe um eins, damit die Abweichung der Werte realistisch bleibt. Größere Faktoren erhöhen das Risiko unplausibler Werte. Bei multivariaten Daten mit physikalischen Kopplungen muss derselbe Faktor auf alle Merkmale angewandt werden, damit die Relationen erhalten bleiben.

Eine weitere Störung ist die additive Störung. Es wird ein Wert ϵ auf den vollständigen Datensatz addiert, jedoch kann sich der Wert, wie bereits bei der multiplikativen Störung auch, über die Zeit leicht verändern. Der Wert von der Störung befindet sich dabei nahe der Null.

Synthetische Datenerweiterung erzeugt, unter anderem mit Modellen, neue Daten mit plausibler Dynamik, die die Konsistenz der Eigenschaften wahren. Verschiedene Methoden, mit denen Daten erzeugt werden können, sind statistische Methoden, *Generative Adversarial Networks* (GANs) [18, 34] oder *Variational Autoencoder* (VAEs) [26, 34]. Statistische Methoden eignen sich, wenn Verteilungen und Korrelationen bekannt sind und erlauben, durch verteilungsbasierte Simulation sowie Inter- und Extrapolation, die Erzeugung neuer Zeitreihenwerte. Mit der Interpolation wird versucht eine stetige Funktion zu definieren, um einen Bereich diskreter Werte zu beschreiben. Extrapolation betrachtet den Bereich darüber hinaus. GANs bestehen aus einem Generator und einem Diskriminator. Der Generator erzeugt synthetische Daten und der Diskriminator versucht echte von künstlichen Daten zu unterscheiden. Beide werden gemeinsam trainiert, bis die generierten Beispiele hinreichend realistisch sind. Die letzte Methode der *Variational Autoencoder* komprimiert Daten mit einem *Encoder* und kehrt das Ganze mit einem *Decoder* um, um neue Daten zu erzeugen [21].

Neben dem Zweck, Datenerweiterung zu verwenden, um die effektive Menge an Trainingsdaten zu erhöhen, wird sie auch eingesetzt, um den Einsatz echter personenbezogener Daten zu vermeiden und den Datenschutz zu stärken. Damit trägt sie wesentlich zur Entwicklung robuster, datenschutzkonformer und genalisierungsfähiger Modelle bei.

2.3.6 Datensatz Aufteilung

Ein Datensatz wird in Trainingssatz und Testsatz aufgeteilt. Der Trainingssatz dient zum Anpassen der Modellparameter. Die Auswahl von Hyperparameter erfolgt ausschließlich auf einer aus dem Trainingssatz abgetrennten Validierungsmenge. Der Testsatz bleibt unverändert und dient nur der abschließenden Beurteilung der Leistung außerhalb der Optimierung.

Für Trainingssatz und Testsatz ist ein angemessenes Klassenverhältnis wichtig. Ein stark unausgewogener Testsatz führt zu wenig aussagekräftigen Kennzahlen. So kann eine stets negative Vorhersage eine hohe scheinbare Genauigkeit erzeugen, ohne die positive Klasse zu erkennen. Ist der Trainingssatz unausgewogen, stehen effektiv wenige Sequenzen zur Verfügung, die für das Training verwendet werden können.

Bei Daten ohne zeitliche Abhängigkeit ist die Aufteilung von Trainingssatz und Testsatz einfacher. Für Datensätze mit gleich vielen Fehlern und Normalzuständen, kann jeweils die gleiche Anzahl an Beispielen gewählt werden, sodass das Klassenverhältnis erhalten bleibt. Bei Zeitreihen impliziert ein ausgewogenes Klassenverhältnis jedoch nicht automatisch eine ausreichende Menge nutzbarer Trainingssequenzen. Die Auswahl erfolgt nicht zufällig über einzelne Punkte, sondern in zusammenhängenden Intervallen, damit der zeitliche Verlauf erlernt wird. Fehlerintervalle sollten komplett enthalten sein und nicht über die Grenze zwischen Training und Test hinausreichen. Nach der Sequenzbildung dürfen die erzeugten Sequenzen im Trainingssatz beliebig vertauscht werden, solange die Ordnung innerhalb einer Sequenz erhalten bleibt. Dies ersetzt jedoch nicht die zeitliche Aufteilung des Rohsignals und löst das Aufteilungsproblem nicht, da übergreifende Fenster weiterhin ausgeschlossen werden müssen.

2.3.7 Principal Component Analysis

Die *Principal Component Analysis* (PCA, Hauptkomponentenanalyse) [43] ist ein statistisches Verfahren zur Reduktion der Dimensionalität. Sie basiert auf einer orthogonalen Transformation, durch die eine Menge korrelierter Variablen in ein neues Koordinatensystem unkorrelierter Variablen, die sogenannten Hauptkomponenten, überführt wird. Ziel ist es, den größten Teil der Varianz der ursprünglichen Daten, mit möglichst wenigen Hauptkomponenten zu erfassen.

f1	f2	f3	f4
1	2	3	4
5	5	6	7
1	4	2	3
5	3	2	1
8	1	2	2

Tabelle 3: Ausgangstabelle für die Hauptkomponentenanalyse [43].

Zur Veranschaulichung der Hauptkomponentenanalyse dient die in Tabelle 3 dargestellte Datenmatrix mit den Merkmalen f_1 bis f_4 und fünf Beobachtungen. Zunächst werden alle Merkmale standardisiert, indem der Mittelwert jeder Spalte subtrahiert und durch die jeweilige Standardabweichung geteilt wird, siehe Gleichung (1). Dadurch erhält jedes Merkmal einen Mittelwert Null und Varianz Eins. Ohne diesen Schritt würden Variablen mit größerem Wertebereich die Analyse dominieren. Die verwendeten Mittelwerte und Standardabweichungen sind in Tabelle 4 dargestellt.

	f1	f2	f3	f4
$\boldsymbol{\mu}$	4	3	3	3.4
$\boldsymbol{\delta}$	3	1.58114	1.73205	2.30217

Tabelle 4: Mittelwerte und Standardabweichungen der Merkmale f_1 bis f_4 [43].

f1	f2	f3	f4
-1	-0.63246	0	0.26062
0.33333	1.26491	1.73205	1.56374
-1	0.63246	-0.57735	-0.17375
0.33333	0	-0.57735	-1.04249
1.33333	-1.26491	-0.57735	-0.60812

Tabelle 5: Standardisierte Datenmatrix X nach Mittelwertzentrierung und Skalierung der Merkmale f_1 bis f_4.

Aus der standardisierten Datenmatrix $X \in \mathbb{R}^{5\times 4}$, dargestellt in Tabelle 5, wird anschließend die Kovarianzmatrix C gebildet.

Sie beschreibt die gemeinsame Streuung der Merkmale und bildet die Grundlage für die Ermittlung der Hauptkomponenten. Für zwei Merkmale x und y ergibt sich die Kovarianz nach folgender Formel:

$$Cov(x,y) = \frac{\sum_{i=1}^{1}(x_i - \bar{x})(y_i - \bar{y})}{N}.$$

Die Kovarianzmatrix C fasst die Varianzen und Kovarianzen aller Merkmale in einer symmetrischen Matrix zusammen. Die Diagonalelemente enthalten die Varianz jedes Merkmals, während die Nebendiagonalelemente die Kovarianzen zwischen den jeweiligen Merkmalspaaren darstellen. Die Ergebnisse der Kovarianzmatrix *C* für die Merkmale f_1 bis f_4 sind in Tabelle 6 dargestellt.

	f1	**f2**	**f3**	**f4**
f1	0.8	-0.25298	0.03849	-0.14479
f2	-0.25298	0.8	0.51121	0.4945
f3	0.03849	0.51121	0.8	0.75236
f4	-0.14479	0.4945	0.75236	0.8

Tabelle 6: Berechnete Kovarianzmatrix C aus der standardisierten Datenmatrix. Die Diagonalelemente enthalten die Varianzen der Merkmale f_1 bis f_4, während die Nebendiagonalelemente deren paarweise Kovarianzen darstellen.

Eine große positive Kovarianz zwischen zwei Merkmalen bedeutet, dass sie gemeinsam ansteigen, während eine negative Kovarianz auf gegenläufige Veränderungen hinweist. Werte nahe null deuten auf eine geringe oder keine lineare Abhängigkeit hin. Im nächsten Schritt werden aus der Kovarianzmatrix die Eigenwerte und die zugehörigen Eigenvektoren bestimmt. Sei A eine quadratische Matrix, also die Kovarianzmatrix, ν ein Vektor und λ ein Skalar, sodass gilt

$$A \times \nu = \lambda \times \nu,$$

dann wird λ als Eigenwert bezeichnet, der dem Eigenvektor ν von A zugeordnet ist. Durch Umstellen der Gleichung erhält man

$$(A - \lambda \times I) \times \nu = 0,$$

wobei I die Einheitsmatrix ist. Diese Gleichung besitzt nur dann eine nichttriviale Lösung, also

$$\nu \neq 0,$$

wenn die Determinante von

$$(A - \lambda \times I)$$

gleich Null ist

$$det(A - \lambda \times I) = 0.$$

Setzt man für A die Kovarianzmatrix ein, ergibt sich eine Matrix, bei der von jedem Diagonalelement der Wert λ subtrahiert wird, dargestellt in Tabelle 7.

	f1	f2	f3	f4
f1	0.8 - λ	-0.25298	0.03849	-0.14479
f2	-0.25298	0.8 - λ	0.51121	0.4945
f3	0.03849	0.51121	0.8 - λ	0.75236
f4	-0.14479	0.4945	0.75236	0.8 - λ

Tabelle 7: Matrix $(C - \lambda \times I)$ zur Berechnung der Eigenwerte der Kovarianzmatrix.

Die berechneten Eigenwerte λ_i sind aus der Lösung der charakteristischen Gleichung

$$det(C - \lambda_i \times I) = 0,$$

$$\lambda_1 = 2.51569234, \lambda_2 = 1.0652885, \lambda_3 = 0.39388704, \lambda_4 = 0.02503121.$$

Anschließend wird die Gleichung

$$(C - \lambda \times I) \times v = 0$$

mit den verschiedenen λ Werten gelöst, um die Eigenvektoren zu erhalten, die als 4×4 Matrix dargestellt werden, siehe Tabelle 8.

e1	**e2**	**e3**	**e4**
0.161960	-0.917059	-0.307071	0.196162
-0.524048	0.206922	-0.817319	0.120610
-0.585896	-0.320539	0.188250	-0.720099
-0.596547	-0.115935	0.449733	0.654547

Tabelle 8: Berechnete Eigenvektormatrix E der Kovarianzmatrix C. Jede Spalte e_i stellt den Eigenvektor dar, der dem Eigenwert λ_i entspricht.

Im nächsten Schritt werden die Eigenvektoren in absteigender Reihenfolge der zugehörigen Eigenwerte sortiert. Die erste Hauptkomponente entspricht dem größten Eigenwert und erklärt den größten Anteil der Gesamtvarianz, während die folgenden Hauptkomponenten jeweils abnehmende Varianzanteile darstellen. In dem Beispiel sind die Eigenwerte bereits in absteigender Reihenfolge angeordnet, daher ist keine zusätzliche Sortierung erforderlich. Als letzten Schritt werden die k größten Eigenwerte und die zugehörigen Eigenvektoren ausgewählt, um die Dimension des Datensatzes zu reduzieren. Die Anzahl k wird so gewählt, dass die ausgewählten Hauptkomponenten einen ausreichend hohen Anteil der Gesamtvarianz abdecken. Die zugehörigen Eigenvektoren bilden gemeinsam die Transformationsmatrix V_k, die aus den ersten k Spalten der vollständigen Eigenvektormatrix besteht. Für das Beispiel werden die beiden größten Eigenwerte und die zugehörigen Eigenvektoren gewählt, sodass sich eine 4×2-Matrix ergibt:

$$V_k = \begin{bmatrix} 0.161960 & -0.917059 \\ -0.524048 & 0.206922 \\ -0.585896 & -0.320539 \\ -0.596547 & -0.115935 \end{bmatrix}$$

Die transformierte Datenmatrix Z ergibt sich aus der Multiplikation der standardisierten Datenmatrix X mit der Transformationsmatrix V_k:

$$Z = X \times V_k$$

$$Z = \begin{bmatrix} 0.014003 & 0.755975 \\ -2.556534 & -0.780432 \\ -0.051480 & 1.253135 \\ 1.014150 & 0.000239 \\ -1.579861 & -1.228917 \end{bmatrix}$$

In der Programmiersprache Python steht mit der Bibliothek *Scikit-learn* [40] eine direkte Implementierung der PCA zur Verfügung. Darüber hinaus bietet die Bibliothek weitere Verfahren zur Dimensionsreduktion, wie *t-distributed Stochastic Neighbor Embedding* (t-SNE), *Random Projections* und *Feature Agglomeration* [39].

2.3.8 Reproduzierbarkeit

Reproduzierbarkeit bedeutet, dass bei gleichen Eingaben und gleichen Einstellungen die gleichen Ergebnisse entstehen. Reproduzierbarkeit als Fähigkeit, ein Experiment unter gleichen Bedingungen mit gleichen Ergebnissen zu wiederholen, ist entscheidend für die Validierung wissenschaftlicher Erkenntnisse und die Vertrauenswürdigkeit von Forschungsergebnissen. Alle Schritte des Entwicklungsprozesses müssen jederzeit nachvollziehbar sein und erneut ausgeführt werden können. Zufall ist zulässig, aber als Pseudozufall mit festen Startwerten, damit identische Läufe identische Resultate erzeugen. Somit werden Zufallsquellen kontrolliert. Startwerte für Zufallsgeneratoren in den verwendeten Bibliotheken werden gesetzt. Das betrifft zum Beispiel NumPy und TensorFlow.

2.4 Deep-Learning

Deep-Learning umfasst Methoden, die auf tiefen neuronalen Netzen basieren und sich besonders für die Modellierung komplexer Muster in Zeitreihendaten eignen. Der folgende Abschnitt führt zunächst in die grundlegenden Strukturen neuronaler Netze ein und beschreibt Architekturen wie *Convolutional Neural Networks* (CNN) und *Recurrent Neural Networks* (RNN), die für sequenzielle Daten von zentraler Bedeutung sind. Darüber hinaus werden Optimierungsverfahren vorgestellt, die ein effizientes Training von Modellen ermöglichen.

2.4.1 Neuronale Netze

Künstliche neuronale Netze (*Artifical Neural Networks*, ANN) bilden die Grundlage moderner *Deep-Learning*-Modelle. Sie bestehen aus einer Vielzahl, miteinander verbundenen Recheneinheiten, den sogenannten Neuronen. Diese Neuronen sind in Schichten (*Layer*) angeordnet und durch gewichtete Verbindungen miteinander

verbunden. Ein typisches neuronales Netz setzt sich aus einer Eingabeschicht (*Input Layer*), einer variablen Anzahl an verdeckten Schichten (*Hidden Layer*) und einer Ausgabeschicht (*Output Layer*) zusammen.

Abbildung 7 zeigt schematisch den Aufbau eines mehrschichtigen neuronalen Netzes. Die Eingabeschicht nimmt die Rohdaten x entgegen und leitet sie an die verborgenen Schichten weiter. In den *Hidden Layers* erfolgt die eigentliche Merkmalsextraktion mithilfe einer Aktivierungsfunktion $g(\cdot)$, wodurch jede Schicht zunehmend abstrakte Repräsentationen der Eingaben erzeugt. Die Ausgabeschicht berechnet schließlich das Ergebnis $\hat{y}$ in Form einer Klassenzuordnung oder einer numerischen Vorhersage.

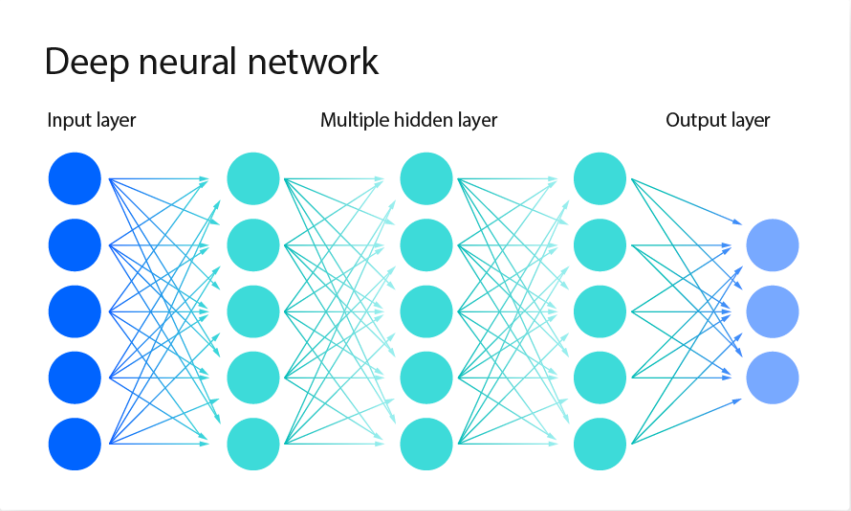

Abbildung 7: Aufbau eines neuronalen Netzes [22]. Es besteht aus einer Eingabeschicht, mehreren verdeckten Schichten und einer Ausgabeschicht.

Jede Verbindung zwischen zwei Neuronen ist durch ein Gewicht (*Weight*) charakterisiert, das die Stärke des Einflusses einer Eingabe auf die nachfolgende Schicht bestimmt. Zusätzlich wird jeder Neuronenausgabe ein *Bias*-Term hinzugefügt, um eine flexible Verschiebung der Aktivierung zu ermöglichen. Das Bias erweitert die lineare Kombination der Eingaben um einen konstanten Wert und ermöglicht es, dem Modell Aktivierungen unabhängig von den Eingabewerten zu verschieben. Dadurch kann das Neuron auch dann aktiv werden, wenn alle Eingaben null sind, was die Flexibilität und Anpassungsfähigkeit des Modells deutlich erhöht.

Der Aktivierungswert eines Neurons ergibt sich aus der gewichteten Summe der Eingaben und des Bias, das anschließend durch eine Aktivierungsfunktion transformiert

wird. Formal kann dieser Zusammenhang wie folgt beschrieben werden:

$$\hat{y} = g(W^T x + b)$$

mit

$$W \in \mathbb{R}^{m \times n}, x \in \mathbb{R}^{m \times 1}, b \in \mathbb{R}^{n \times 1}, \hat{y} \in \mathbb{R}^{n \times 1}.$$

W bezeichnet die Gewichtsmatrix, x den Eingangsvektor, b den *Bias*-Vektor und $g(\cdot)$ eine Aktivierungsfunktion, die eine nichtlineare Abbildung realisiert.

Die Aktivierungsfunktionen spielen eine zentrale Rolle, da sie bestimmen, welche Art von Zusammenhängen ein neuronales Netz modellieren kann. Sie ermöglichen es, nichtlineare Beziehungen zwischen Eingaben und Ausgaben abzubilden und dadurch komplexe Muster und Abhängigkeiten in den Daten zu modellieren. Gängige Funktionen sind die Sigmoid-Funktion, die Werte zwischen 0 und 1 abbildet, die tanh-Funktion, die den Bereich auf -1 bis 1 skaliert und die *Rectified Linear Unit* (ReLU), die negative Eingaben auf null setzt und positive Eingaben unverändert weitergibt.

Die Wahl der Aktivierungsfunktion ist entscheidend für die Leistungsfähigkeit des Modells und hängt stark von der zugrunde liegenden Problemstellung ab. Während Sigmoid- und Tanh-Funktionen häufig in kleineren oder probabilistischen Modellen verwendet werden, eignen sich ReLU-Varianten insbesondere für tiefe Netze, da sie eine effizientere Gradientenweitergabe ermöglichen und Trainingsstabilität fördern.

In der Ausgabeschicht werden je nach Problemstellung unterschiedliche Funktionen verwendet. Bei Klassifikationsaufgaben mit mehreren Klassen kommt häufig die Softmax-Funktion zum Einsatz. Sie wandelt die Ausgaben des Netzwerkes in Wahrscheinlichkeiten um, die sich zu 1 summieren [2] und ermöglicht damit eine eindeutige Klassenzuweisung. Die Softmax-Funktion für eine Ausgabekomponente i ist definiert als

$$Softmax(x_i) = \frac{exp(x_i)}{\sum_j exp(x_j)}.$$

Damit kann die Wahrscheinlichkeit jeder Klasse direkt interpretiert werden, was insbesondere bei *Multi-Class*-Klassifikationen nützlich ist.

Zur Bewertung der Modellgüte wird eine Verlustfunktion (*Loss*) verwendet, die den Unterschied zwischen der Vorhersage des Modells und dem tatsächlichen Zielwert misst. Sie dient als Optimierungskriterium während des Trainings. In Klassifikationsaufgaben wird häufig die *Binary Cross-Entropy* oder *Categorical Cross-Entropy* verwendet, während bei Regressionsproblemen Funktionen wie der *Mean Squared Error (MSE)* üblich sind. Der Optimierungsprozess passt die Gewichte und *Biases* schrittweise an, um die *Loss Function* zu minimieren und die Vorhersagegenauigkeit des Modells zu erhöhen.

Neben der Schichttiefe unterscheiden sich neuronale Netze auch hinsichtlich der Beziehung zwischen Eingabe- und Ausgabesequenzen. Abbildung 8 veranschaulicht die verschiedenen Strukturtypen, die abhängig von der Aufgabenstellung eingesetzt werden. *One-to-One*-Modelle bilden eine einzelne Eingabe auf eine einzelne Ausgabe ab, während *many-to-one*-Modelle mehrere Eingaben verarbeiten, um eine zusammenfassende Ausgabe zu erzeugen. Die verschiedenen Strukturen werden je nach Datenbeschaffenheit und Zielsetzung des Modells verwendet.

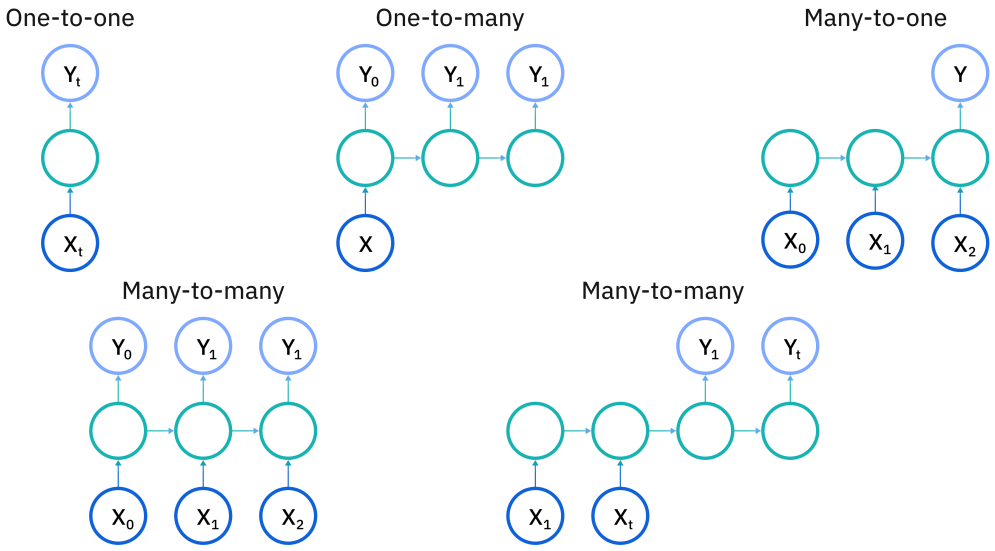

Abbildung 8: Eingabe- und Ausgabestrukturen neuronaler Netze mit unterschiedlicher Sequenzbeziehung [23].

One-to-one-Modelle sind konzeptuell einfach und effizient, eignen sich aber nur für statische Zuordnungsprobleme wie Klassifikation oder Regressionen mit einzelnen Eingabewerten. *One-to-many*-Modelle ermöglichen die Generierung von Ausgabesequenzen aus einem Eingangswert und finden Anwendung in der Text- oder Sprachgeneration. *Many-to-one*-Modelle fassen mehrere Zeitschritte oder Merkmale zu einer Gesamtentscheidung zusammen und sind damit typisch für Klassifikationsaufgaben auf Zeitreihen. *Many-to-many*-Modelle bieten die größte Flexibilität, da sie Sequenzen beliebiger Länge verarbeiten und gleichzeitig Ausgaben für jeden Zeitschritt erzeugen können. Dies ermöglicht den Einsatz in Aufgaben wie Sprachübersetzung oder Zeitreihenprognose, geht jedoch mit höherem Rechenaufwand einher.

2.4.2 Architekturen tiefer neuronaler Netze

Tiefe neuronale Netze (*Deep Neural Networks*, DNN) bezeichnen mehrschichtige Architekturen, die aus einer Vielzahl von hintereinander geschalteten Verarbeitungsebenen bestehen. Sie ermöglichen es, komplexe nichtlineare Abbildungen zwischen Eingabe- und Ausgabedaten zu lernen und bilden die Grundlage zahlreicher moderner *Deep-Learning*-Verfahren. Innerhalb dieser Klasse lassen sich verschiedene spezialisierte Architekturen unterscheiden, die sich hinsichtlich der Art der Datenverarbeitung und ihrer internen Struktur voneinander abgrenzen. Zu den verbreitetsten gehören *Convolutional Neural Networks* und *Recurrent Neural Networks*.

CNN wurden primär für die Verarbeitung räumlich strukturierter Daten entwickelt [12], bei denen die Nachbarschaft einzelner Werte eine Bedeutung besitzt. Ursprünglich wurden sie in der Bildverarbeitung eingesetzt, jedoch eignen sie sich auch für eindimensionale Signale, wie Sensormessungen oder Zeitreihen. Als *Feed-Forward Neural Network* (FFNN) verarbeiten sie Informationen ausschließlich in eine Richtung, von der Eingabe- zur Ausgabeschicht.

Die grundlegende Operation eines CNN ist die Faltung (*Convolution*). Dabei wird ein Gewichtsvektor, der sogenannte Kernel oder Filter, über das Eingabesignal verschoben. Für jede Position wird das Skalarprodukt zwischen Kernel und überlappendem Eingabebereich berechnet.

Abbildung 9 zeigt eine eindimensionale Faltung mit einem Kernel der Größe 3. Das Eingangssignal wird schrittweise mit dem Filter multipliziert, wobei an jeder Position die gewichtete Summe der überlappenden Werte entsteht. Durch diese Verfahren werden lokale Muster erkannt, wie etwa Anstiege, Spitzen oder wiederkehrende Signalformen.

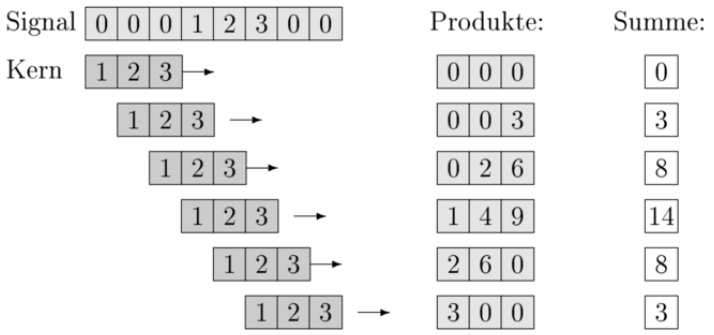

Abbildung 9: Beispiel einer eindimensionalen Faltung [3]. Der Kernel wird über das Signal verschoben und die gewichteten Produkte werden aufsummiert.

Durch das Stapeln mehrerer Faltungsschichten können zunehmend abstrakte Merkmale extrahiert werden, wie in klassischen neuronalen Netzen, jedoch mit deutlich weniger Parametern. CNNs sind daher besonders effizient und robust gegenüber Verschiebungen im Eingangssignal. In der Verarbeitung von Zeitreihen werden sie häufig eingesetzt, um lokale Muster zu erfassen, bevor rekurrente Schichten den zeitlichen Kontext modellieren.

RNNs sind darauf ausgelegt, zeitliche oder sequentielle Abhängigkeiten zu modellieren. Im Gegensatz zu klassischen FFNN enthalten sie rekurrente Verbindungen, über die der *hidden state* eines Neurons an nachfolgende Zeitschritte weitergegeben wird. Dadurch kann das Netzwerk Informationen aus vergangenen Eingaben berücksichtigen und kontextabhängige Beziehungen innerhalb einer Sequenz erfassen. Diese Eigenschaft macht RNNs besonders geeignet für Anwendungen in Zeitreihenanalyse, Sprachmodellierung, maschinelle Übersetzung oder Anomalieerkennung.

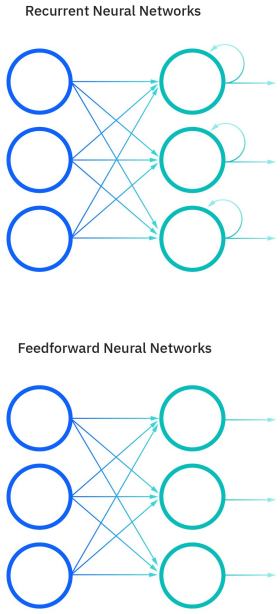

Abbildung 10: Vergleich zwischen *Feed-Forward Neural Network* und *Recurrent Neural Network* [23]. FFNN verarbeiten Informationen ausschließlich in eine Richtung, während RNN über Rückkopplungen verfügen, die frühere Zustände in die aktuelle Berechnung einbeziehen.

Abbildung 10 veranschaulicht den strukturellen Unterschied zwischen einem klassischen *Feedforward Neural Network* und einem *Recurrent Neural Network*. RNN besitzen zusätzlich Rückkopplungen innerhalb der versteckten Schichten. Durch die Rückkopplung kann die Ausgabe eines Neurons nicht nur an die nächste Schicht weitergegeben werden, sondern auch als Eingabe für dasselbe Neuron oder andere Neuronen derselben oder einer vorherigen Schicht dienen. Dadurch wird ein interner Zustand aufgebaut, der Informationen über vergangene Eingaben enthält und das Modell befähigt, Abhängigkeiten über die Zeit zu erfassen. Der Gewichtsparameter, zur Steuerung der Rückkopplung, bleibt über alle Zeitschritte hinweg identisch.

Ein bekanntes Problem klassischer RNNs ist der sogenannte Gradientenverfall (*Vanishing Gradient Problem*) [2]. Bei langen Sequenzen werden die Gradienten während des Trainings über viele Zeitschritte zurückpropagiert. Durch wiederholte Multiplikation kleiner Ableitungswerte tendieren diese gegen null, wodurch das Netzwerk kaum noch in der Lage ist, langfristige Abhängigkeiten zu erlernen. Umgekehrt können große Ableitungen zu explodierenden Gradienten (*Exploding Gradient Problem*) [2] führen, was das Training instabil macht.

Das *Long Short-Term Memory* ist eine Erweiterung klassischer rekurrenter neuronaler Netze und adressiert das Problem des Gradientenverfalls und der instabilen Trainingsdynamik in einfachen RNNs. Es führt eine explizite Speicherzelle ein, die Informationen über längere Zeiträume hinweg beibehalten kann. Der Informationsfluss in der Zelle wird über drei *Gates* gesteuert, das *Input-Gate*, *Forget-Gate* und *Output-Gate*. Diese Mechanismen regulieren, welche Informationen hinzugefügt, gelöscht oder weitergegeben werden, wodurch relevante Merkmale über viele Zeitschritte hinweg erhalten bleiben.

Das *Input-Gate* kontrolliert, welche neuen Informationen in die Speicherzelle übernommen werden. Es gewichtet den aktuellen Eingang relativ zum bestehenden Zellzustand, sodass nur relevante Signale aufgenommen werden. Das *Forget-Gate* entscheidet, welche Teile des bisherigen Zellzustands gelöscht oder beibehalten werden, was eine flexible Anpassung an neue Kontexte ermöglicht. Das *Output-Gate* bestimmt schließlich, welcher Anteil des internen Zustands in den *hidden state* überführt wird und somit zur nächsten Vorhersage beiträgt.

Durch die beschriebene additive Struktur des Speicherpfads wird der Gradient über viele Schritte hinweg stabil gehalten. Das ermöglicht es dem LSTM, langfristige Abhängigkeiten zu modellieren, ohne dass die Lernrate oder der Gradientenfluss zusammenbricht.

Eine häufig verwendete Erweiterung ist das bidirektionale LSTM (Bi-LSTM). Dabei wird die Eingabesequenz sowohl in Vorwärts- als auch in Rückwärtsrichtung verarbeitet. Die resultierenden Zustände werden kombiniert, sodass das Modell zu

jedem Zeitpunkt über Informationen aus Vergangenheit und Zukunft verfügt [15]. Dies verbessert insbesondere bei Aufgaben wie Sprachmodellierung, Textanalyse oder Anomalieerkennung die Kontextrepräsentation. Allerdings erfordert das Bi-LSTM den Zugriff auf die gesamte Sequenz, wodurch es für Echtzeitvorhersagen nur eingeschränkt geeignet ist. Die doppelte Verarbeitung führt außerdem zu erhöhtem Rechen- und Speicherbedarf.

Weitere Varianten erweitern das LSTM um zusätzliche Mechanismen, die die Informationsverarbeitung verfeinern. *Attention*-LSTM ergänzt die Architektur um einen Aufmerksamkeitsmechanismus, der die Relevanz einzelner Schritte bewertet und es dem Modell ermöglicht, sich gezielt auf informative Abschnitte der Eingabe zu konzentrieren. *Multi-Head*-LSTMs verwenden mehrere parallele LSTM-Köpfe, deren Ausgaben kombiniert werden [25], um unterschiedliche Abhängigkeitsmuster simultan zu erfassen. Beide Varianten erhöhen die Ausdrucksfähigkeit des Modells, erfordern jedoch meist eine größere Datenmenge und längere Trainingszeiten.

2.4.3 Optimierungsverfahren

Die Leistung neuronaler Netze hängt in hohem Maße von der Wahl und Abstimmung ihrer Parameter ab. Diese Parameter bestimmen, wie effektiv das Modell während des Trainings Muster in den Daten erkennen und verallgemeinern kann. Dabei wird zwischen trainierbaren Modellparametern, wie den Gewichten und *Bias* und nicht-trainierbaren Hyperparametern unterschieden.

Zu den zentralen Hyperparametern gehören die *Batch*-Größe, die Anzahl der Neuronen pro Schicht, die Fenstergröße bei sequenziellen Modellen, die Lernrate sowie die Anzahl der Epochen. Jede dieser Größen beeinflusst das Lernverhalten des Modells auf unterschiedliche Weise.

Die *Batch*-Größe bestimmt, wie viele Trainingsbeispiele in einem Schritt zur Aktualisierung der Gewichte verwendet werden. Eine große *Batch*-Größe kann zu einer schnelleren Konvergenz, aber geringerer Generalisierungsfähigkeit führen, während eine kleine *Batch*-Größe zu instabileren Gradienten und einem schwankenden Lernverlauf führt.

Die Anzahl der Neuronen pro Schicht (*Units*) legt die Kapazität des Modells fest. Eine höhere Anzahl ermöglicht komplexere Abbildungen, erhöht jedoch auch die Gefahr des *Overfittings*.

Die Fenstergröße (*window size*) definiert bei sequenziellen Modellen den zeitlichen Kontext, der bei der Verarbeitung von Zeitreihen berücksichtigt wird und beeinflusst somit die Fähigkeit des Modells, Abhängigkeiten über mehrere Zeitschritte zu erkennen.

Die Lernrate (*learning rate*) steuert die Schrittweite, mit der die Gewichte des Modells während des Trainings angepasst werden. Eine zu hohe Lernrate kann zu Divergenz führen, während eine zu niedrige Lernrate eine sehr langsame Konvergenz bewirkt.

Die Anzahl der Epochen bestimmt, wie oft der gesamte Trainingsdatensatz durchlaufen wird. Eine zu geringe Anzahl kann zu *Underfitting* führen, während eine zu hohe Anzahl *Overfitting* begünstigt.

Die optimale Kombination der Hyperparameter hängt stark von Datensatz, Modellarchitektur und Trainingsziel ab, weshalb systematische Suchverfahren eingesetzt werden. Typische Ansätze sind *Grid Search*, *Random Search* oder adaptive Verfahren wie der *Hyperband-Tuner* [30], die mehrere Parameterkombinationen automatisiert evaluieren. *Grid Search* testet alle Parameterkombinationen innerhalb eines definierten Wertebereichs, *Random Search* wählt stichprobenartig Kombinationen aus. Der *Hyperband-Tuner* kombiniert den zufälligen Ansatz mit einer dynamischen Ressourcenverteilung, indem vielversprechende Modellkonfigurationen priorisiert und weniger erfolgreiche frühzeitig verworfen werden. Unabhängig vom gewählten Verfahren werden verschiedene Kombinationen der Hyperparameter getestet und anhand einer definierten Optimierungsfunktion, häufig der Validierungsverlust oder einer Zielmetrik, siehe Gleichungen (2) bis (7), bewertet. Die Konfiguration mit dem besten Ergebnis wird anschließend als optimale Parametereinstellung übernommen.

Ein weiterer Aspekt der Modelloptimierung ist die Steuerung des Trainingsverlaufs. Neuronale Netze erreichen nicht zwingend am Ende des Trainingszyklus ihre beste Leistung, weshalb Verfahren wie *EarlyStopping* [36] zum Einsatz kommen. Das Training wird beendet, sobald sich die Modellleistung auf den Validierungsdaten über eine festgelegte Anzahl von Epochen nicht weiter verbessert. Auf diese Weise lässt sich *Overfitting* vermeiden und die Trainingszeit reduzieren.

Zur weiteren Stabilisierung und Feinsteuerung des Lernprozesses kann zusätzlich *ReduceLROnPlateau* [51] verwendet werden. Dieses Verfahren überwacht den Verlauf der Validierungsmetrik und reduziert automatisch die Lernrate, wenn sich die Leistung über mehrere Epochen hinweg nicht verbessert. Dadurch werden zu große Gewichtsanpassungen vermieden und eine feinere Konvergenz nahe des Optimums ermöglicht.

Gemeinsam tragen die Optimierungsverfahren dazu bei, die Trainingsdauer zu verkürzen, *Overfitting* zu vermeiden und die Generalisierungsfähigkeit des Modells zu verbessern.

2.5 Evaluationsmetriken

	Positive	**Negative**
Positive	True Positive (TP)	False Positive (FP)
Negative	False Negative (FN)	True Negative (TN)

Tabelle 9: Konfusionsmatrix zur Ableitung der Evaluationsmetriken.

Zur Beurteilung der Leistungsfähigkeit eines Modells werden verschiedene Metriken eingesetzt. Grundlage bildet die Konfusionsmatrix, die die Anzahl an korrekt und inkorrekt klassifizierter Instanzen in den Kategorien *True Positive* (TP), *True Negative* (TN), *False Positive* (FP) und *False Negative* (FN) darstellt. In Tabelle 9 ist eine Konfusionsmatrix mit beschriebenen Kategorien dargestellt. Mit den Kategoriewerten aus der Konfusionsmatrix kann die *Accuracy* (Genauigkeit), *Precision*, *Recall*, *Specificity* (Spezifität) und der $F1$-*Score* wie folgt berechnet werden:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4}$$

$$\text{Specificity} = \frac{TN}{TP + FN} \tag{5}$$

$$F1 - \text{Score} = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{6}$$

$$F_\beta\text{-Score} = (1 + \beta^2) \times \frac{Precision \times Recall}{(\beta^2 \times Precision) + Recall} \tag{7}$$

Die *Accuracy* beschreibt den Anteil korrekt klassifizierter Instanzen an der Gesamtmenge. Bei unausgeglichenen Klassen ist die Metrik nicht aussagekräftig, da eine Mehrheitsklasse die Genauigkeit stark verzerren kann. Ist das Verhältnis der Klassen stark unausgeglichen und macht die Mehrheitsklasse etwa 90% des Datensatzes aus, erscheint die Genauigkeit trotzdem hoch, wenn ausschließlich die dominierende Klasse vorhergesagt wird, da nur rund 10% der Instanzen falsch klassifiziert werden.

Die *Precision* gibt den Anteil korrekt positiver Vorhersagen an allen vom Modell als positiv vorhergesagten Instanzen an. Es wird die Verlässlichkeit positiver Vorhersagen gemessen.

Der *Recall* quantifiziert den Anteil korrekt erkannter positiver Instanzen an allen

tatsächlich positiven Fällen. Die Metrik ist entscheidend, wenn es wichtiger ist, möglichst viele positive Instanzen zu erfassen.

Die *Precision* und der *Recall* sind von dem Problem, das die *Accuracy* hat nicht betroffen und liefern trotz einer Mehrheitsklasse aussagekräftige Aussagen.

Die *Specificity* ist das Gegenstück zum *Recall*. Sie misst den Anteil tatsächlich negativer Instanzen, die korrekt als negativ erkannt werden.

Der $F1$-*Score* ist das harmonische Mittel aus der *Precision* und dem *Recall* und stellt ein ausgewogenes Maß dar, wenn sowohl eine hohe *Precision* als auch ein hoher *Recall* wichtig sind. *Precision* und *Recall* sind dabei gleich gewichtet.

Der F_β-*Score* ist die allgemeine Variante vom $F1$-*Score* und erlaubt es, *Precision* und *Recall* unterschiedlich stark zu gewichten. Wird der Wert von β auf 1 gesetzt, sind beide gleich gewichtet. Der $F1$-*Score* stellt somit den Spezialfall des F_β-*Scores* mit $\beta = 1$ dar. Ein Wert kleiner als 1 betont die *Precision* stärker, ein Wert größer als 1 den *Recall*.

3 Verwandte Arbeiten

In diesem Kapitel werden bestehende Forschungsarbeiten vorgestellt, die sich mit der Automatisierung von *Labeling*-Prozessen, der Unterstützung durch maschinelles Lernen und der schrittweisen Reduktion des manuellen Annotierungsaufwands beschäftigen.

3.1 Segment Anything

Das *Segment Anything Model* (SAM) wurde von Kirillov et al. (2003) [27] als universelles Modell zur automatischen Bildsegmentierung entwickelt. Ziel der Arbeit war es, ein sogenanntes *Foundation Model* zu schaffen, das in der Lage ist, beliebige Objekte in Bildern zu erkennen und zu segmentieren, ohne für jede spezifische Aufgabe neu trainiert werden zu müssen. Um den dafür erforderlichen Trainingsdatensatz zu erzeugen, wurde ein modellgestützter *Labeling*-Prozess eingeführt, der sich in drei aufeinanderfolgenden Phasen gliedert.

In der ersten Phase, der *Assisted Manual Stage*, unterstützten vom Modell generierte Maskenvorschläge menschliche Annotatoren bei der manuellen Segmentierung. In der anschließenden *Semi-Automatic Stage*, erzeugt das Modell Masken mit hoher Konfidenz automatisch, während nur unsichere Vorhersagen überprüft und bei Bedarf korrigiert werden. Der Übergang zur *Fully Automatic Stage* erfolgte erst, nachdem die automatisch erzeugten Masken eine nahezu menschliche Genauigkeit erreichten. Dazu wurde die Übereinstimmung zwischen automatischen und manuell korrigierten Masken über die Kennzahl *Intersection over Union* (IoU) bewertet. Dabei zeigte sich, dass 94% der automatisch erzeugten Masken eine Überlappung von über 90% mit den durch Menschen erstellten Masken aufwiesen, was dem typischen Grad menschlicher Übereinstimmung entsprach [19, 28].

Nach dieser Qualitätsprüfung wurde das Modell in der *Fully Automatic Stage* ohne menschliche Beteiligung eingesetzt und generierte sämtliche Masken selbstständig. Durch diesen iterativen Prozess gelang es den Autoren, den Datensatz SA-1B zu erstellen, der über eine Milliarde Segmentierungsmasken umfasst. Die erzielte Maskenqualität war dabei nahezu identisch mit den von Menschen erstellten Annotationen. Das Projekt zeigt, wie durch die Kombination aus menschlicher Interaktion und maschinellem Lernen, der manuelle *Labeling*-Aufwand schrittweise reduziert und schließlich vollständig automatisiert werden kann.

3.2 MaD GUI

Das von Ollenschläger et al. vorgestellte MaD GUI [35] ist ein *Open-Source*-Werkzeug zur Annotation und Analyse von Zeitreihendaten, das insbesondere auf Adaptierbarkeit, Benutzerfreundlichkeit und positive Nutzererfahrung ausgelegt ist. Das

System basiert auf einer flexiblen Plugin-Architektur, die es ermöglicht, unterschiedliche Datenformate zu laden, Label-Strukturen zu definieren sowie Algorithmen zur Verarbeitung oder automatisierten Annotation einzubinden. Ergänzend setzt MaD GUI auf ein *Model-View-Controller*-Konzept, das Dateninhalt, Visualisierung und Interaktion klar voneinander trennt und damit die Erweiterung des Systems unterstützt.

Abbildung 11 zeigt das Hauptfenster der MaD GUI und verdeutlicht die gleichzeitige Visualisierung verschiedener Annotationsarten, beispielsweise Aktivitäten wie „Jump“ oder „Walk“ sowie darunterliegende Schrittsegmente.

Abbildung 11: Hauptfenster der MaD GUI mit geöffneter Seitenleiste, für Plugin-Auswahl und aktiven Annotationsmodus. Im Beispiel werden im oberen Bereich des Plots Aktivitäten markiert, während darunter einzelne Schrittabschnitte hervorgehoben sind [35].

Ein zentrales Element der Arbeit ist die Durchführung einer zweistufigen Studie, in der sowohl Entwickler als auch medizinische Fachanwender das System evaluierten. Die im Rahmen der Studie ermittelten Arbeitszeiten zeigten, dass Entwickler zur Umsetzung typischer Aufgaben wie Datenimport oder Algorithmusintegration deutlich weniger Zeit benötigen als mit dem Vergleichssystem PALMS [13]. Zusätzlich wurde MaD GUI von Fachanwendern im Rahmen des *System Usability Scale* (SUS) und des *User Experience Questionnaire* (UEQ) als intuitiv und leicht erlernbar bewertet.

Die Studie verdeutlicht, dass MaD GUI eine flexible und praxistaugliche Lösung

für die Annotation von Zeitreihendaten bietet. Durch die modulare Architektur und die, in der Studie bestätigte Bedienbarkeit, eignet sich das Werkzeug sowohl für Entwickler als auch für medizinische Fachanwender.

4 Methodik

4.1 Datensatz

Der in dieser Arbeit verwendete Datensatz ist der **Indoor-Outdoor Detection (IOD) Dataset**, der im Rahmen der Studie *Deep Learning-based Multivariate Time Series Classification for Indoor-Outdoor Detection* [4] erstellt wurde. Er basiert auf realen Sensormessungen, die über einen Zeitraum von sechs Monaten mit unterschiedlichen Smartphones aufgezeichnet wurden. Ziel war die Klassifikation von Umgebungen in *indoor* und *outdoor* Szenarien auf Grundlage multivariater Zeitreihen. Es wurde sich bewusst für den Einsatz multivariater Zeitreihen entschieden, da das Unternehmen das Ziel verfolgt, seine Modellierung auf solche Daten auszurichten.

Der Datensatz umfasst insgesamt acht Merkmale, die verschiedene Umwelteinflüsse und Gerätesensoren abbilden. Alle Messungen wurden mit einer Abtastrate von 1 Hz erhoben. Die Merkmale setzen sich wie folgt zusammen:

- **RSRP**: Signalstärke des Mobilfunks
- **RSRQ**: Signalqualitätsmessung
- **Mag**: magnetische Feldstärke
- **Light**: Lichtintensität
- **Acc**: Beschleunigung
- **Sound**: Geräuschintensität
- **Proximity**: Näherungssensor zur Überprüfung der Zuverlässigkeit der Lichtmessung
- **Daytime**: binäres Merkmal zur Unterscheidung zwischen Tag- und Nachtphasen

Zusätzlich zu den acht Features sind zwei weitere Spalten im Datensatz enthalten:

- **New_Recording**: markiert den Beginn einer neuen Aufzeichnung
- **IO**: enthält den Zielwert für die Klassifikation, dabei gilt $IO = 1$ für Indoor- und $IO = 0$ für Outdoor-Umgebungen

Der gesamte Datensatz umfasst $1.473.287$ Zeilen. Anhand der Spalte „new_recording“ konnten insgesamt 84 Aufzeichnungen identifiziert werden, die jeweils den Beginn einer neuen Messsequenz markieren. Vier Aufzeichnungen mit weniger als 500 Zeitschritten wurden verworfen, sodass 80 Aufzeichnungen für die weitere Verarbeitung berücksichtigt wurden. Jede Aufzeichnung wurde als separate CSV-Datei gespeichert, um eine klare Trennung zwischen verschiedenen Aufnahmesitzungen

zu gewährleisten und zu verhindern, dass zeitlich unabhängige Sequenzen im Modell als fortlaufend interpretiert werden. Anschließend wurden die 80 CSV-Dateien nochmals in Trainings- und Testanteile aufgeteilt, wobei die ersten 80% der Zeitschritte für das Training und die verbleibenden 20% für das Testen verwendet wurden. Auf diese Weise entstand ein repräsentativer Trainings- und Testdatensatz, der die unterschiedlichen Messbedingungen und Szenarien abdeckt.

Kennzahlen/Datensatz	Originaldatensatz	Training	Test
Gesamtzeilen	1.473.287	1.164.482	291.163
Anzahl Aufzeichnungen	84	80	80
Mean	–	14556	3639
Standardabweichung	–	9244	2311
Median	–	12.771	3.193
Min	–	2.684	671
Max	–	45.332	11.333

Tabelle 10: Statistische Kennzahlen der Aufzeichnungslängen vom Original-, Trainings- und Testdatensatz.

Tabelle 4.1 fasst die statistischen Kennzahlen der Aufzeichnungslängen für den vollständigen Datensatz sowie für die Trainings- und Testdaten zusammen. Die Werte zeigen, dass die Länge der Aufzeichnungen erheblich variiert. Im Trainingsdatensatz beträgt der Mittelwert der Aufzeichnungslängen 14556 Zeitschritte. Die Standardabweichung von 9244 Zeitschritten weist darauf hin, dass viele Aufzeichnungen deutlich vom Mittelwert abweichen. Der Median liegt bei 12771 Zeitschritten. Die Aufzeichnungen reichen von 2684 bis 45332 Zeitschritten und verdeutlichen die ausgeprägte Streuung innerhalb des Trainingsdatensatzes. Auch der Testdatensatz zeigt eine erkennbare Variabilität. Er besitzt einen Mittelwert von 3639 Zeitschritten und eine Standardabweichung von 2311 Zeitschritten. Der Median beträgt 3193 Zeitschritte und die Aufzeichnungen umfassen Längen zwischen 671 und 11333 Zeitschritten. Dies bestätigt, dass die Aufzeichnungen im Testdatensatz insgesamt kürzer, zugleich aber ebenfalls stark unterschiedliche Längen besitzen.

Eine Datenbereinigung war nicht erforderlich, da im Originalpaper [4] bereits eine umfassende Vorverarbeitung durchgeführt wurde. Dabei wurden Ausreißer im oberen Perzentil entfernt, die auf fehlerhafte Sensormessungen zurückzuführen waren. Zudem wurden die Sensorwerte mittels Min-Max-Normalisierung in einen einheitlichen Wertebereich transformiert.

4.2 Prozess

Im Rahmen dieser Abschlussarbeit wird ein Prozess für das modellgestützte *Labeling* beschrieben, der im Unternehmen entwickelt und in dieser Arbeit als Tool umgesetzt wurde. Der Prozess dient als methodische Grundlage zur Unterstützung der

manuellen Annotation, sowie zur iterativen Verbesserung eines Modells. Er bildet den vollständigen Ablauf von der initialen Erstellung gelabelter Trainingsdaten bis hin zur fortlaufenden Erweiterung und Optimierung des Modells durch neue Daten ab. Phase 1 und Phase 2 sind in Abbildung 12 dargestellt. Der Prozess gliedert sich in zwei aufeinander aufbauende Phasen.

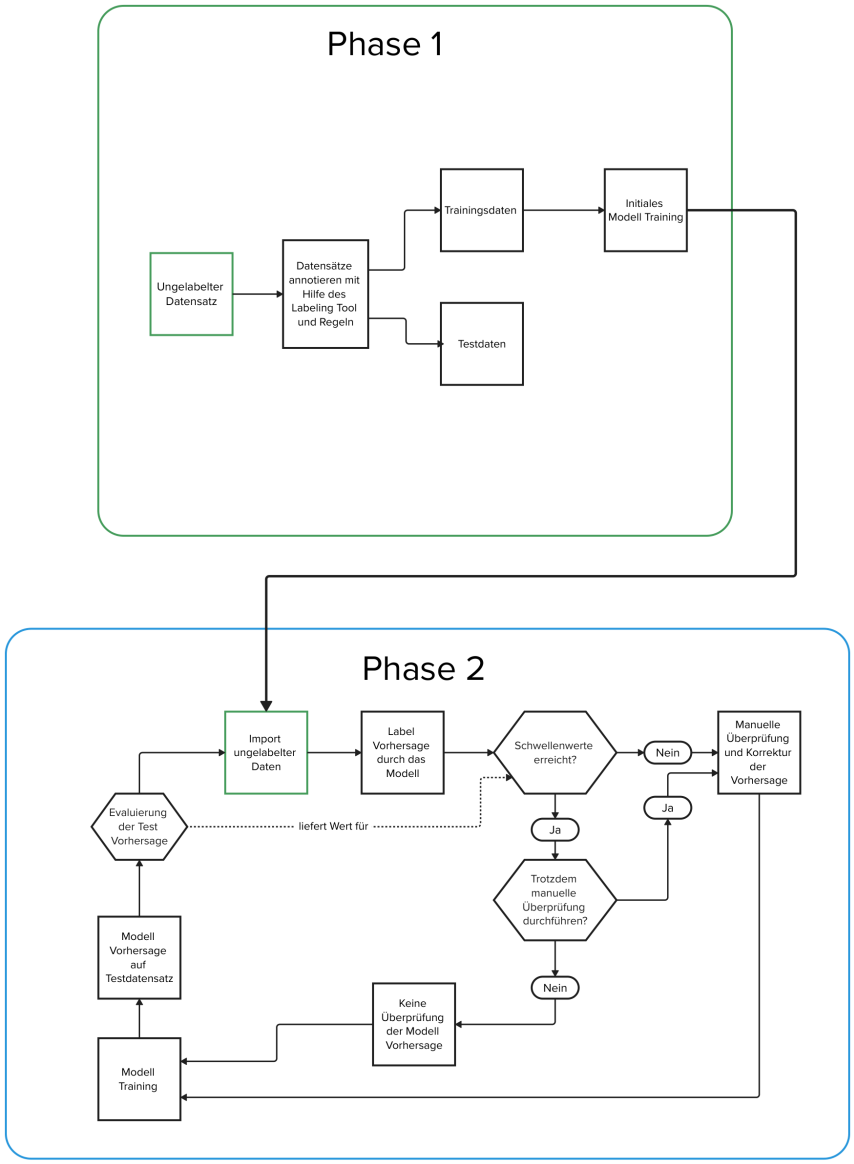

Abbildung 12: Zweiphasiger Prozess für das modellgestützte *Labeling*.

Phase 1 umfasst die manuelle Annotation eines ausreichend großen Datensatzes oder mehrerer Datensätze durch eine oder mehrere Personen mit fachlicher Expertise. Ziel

ist die Erstellung einer ersten Menge qualitativ hochwertiger *Labels*, die als Basis für die Aufteilung in Trainings- und Testdaten dient. Für die manuelle Annotation wird ein *Labeling*-Tool verwendet, das den Annotierenden über eine interaktive Oberfläche unterstützt und im Kapitel 5 näher beschrieben wird. Ergänzend können regelbasierte *Pseudo-Labels* eingesetzt werden, um den manuellen Aufwand zu reduzieren und den Annotierenden eine erste Orientierung im *Labeling*-Tool zu bieten. Mit den annotierten Daten wird das Modell initial trainiert, wodurch eine Ausgangsbasis für die Generierung von Vorhersagen geschaffen wird. Das resultierende Modell dient als Startpunkt für die nachfolgende Phase 2.

Phase 2 beschreibt einen iterativen Ablauf, in dem das trainierte Modell auf neu eingehende, bislang nicht annotierte Daten angewendet wird. Das Modell erzeugt auf diesen Daten Vorhersagen in Form von *Pseudo-Labels*, die als Vorschläge für die manuelle Überprüfung dienen. Auch in dieser Phase wird das *Labeling*-Tool eingesetzt, um die vom Modell vorgeschlagenen *Labels* zu visualisieren und deren Validierung sowie Korrektur durch Personen mit spezialisiertem Fachwissen zu unterstützen. Die korrigierten *Labels* fließen anschließend in den Prozess ein, um das Modell schrittweise zu verbessern und seine Vorhersagefähigkeit für zukünftige Daten weiter zu stärken. Auf diese Weise soll sich das Modell schrittweise verbessern, während gleichzeitig der manuelle Aufwand mit jeder Iteration reduziert wird.

Nach jedem Trainingszyklus in Phase 2 erfolgt eine Evaluation auf dem Testdatensatz, um die Modellleistung zu überwachen. Basierend auf den erzielten Kennwerten, der *Precision* und *Recall*, wird entschieden, ob das Modell künftig ohne manuelle Überprüfung labeln und trainieren darf. Diese Metriken wurden gewählt, da sie sowohl die Genauigkeit der positiven Vorhersagen als auch die Erkennungsrate der relevanten Klassen erfassen. Wird ein festgelegter Schwellenwert überschritten, kann das Modell seine eigenen Vorhersagen direkt für das weitere Training verwenden. Somit soll ein Übergang von einem überwachten, zu einem zunehmend selbständigen, modellgestützten *Labeling* ermöglicht werden. Der Prozess beschreibt damit einen vollständigen, praxisorientierten Ablauf zur Reduktion des manuellen *Labeling*-Aufwands durch den schrittweisen Einsatz eines Modells. Im Rahmen dieser Arbeit wurde dieser Prozess in Form eines Tools implementiert, das die beschriebenen Schritte softwaregestützt abbildet und die Interaktion zwischen Modell und Annotierenden unterstützt. Eine detaillierte Beschreibung der technischen Umsetzung und der einzelnen Komponenten des Tools erfolgt in Kapitel 5.

4.3 Labeling-Strategien

Das *Labeling* der Daten stellt einen wesentlichen Bestandteil des Prozesses in Kapitel 4.2 dar, da die Qualität der erzeugten *Labels* einen direkten Einfluss auf die Leistungsfähigkeit des Modells hat. Um den manuellen *Labeling*-Aufwand zu reduzieren und gleichzeitig eine hohe *Label*-Qualität zu erreichen, werden in dieser

Arbeit verschiedene Strategien zur Annotation der Daten verwendet. Die Strategien unterscheiden sich hinsichtlich ihres Automatisierungsgrades und werden schrittweise eingesetzt, um den Labelprozess effizient zu gestalten und die Trainingsdaten sukzessive zu verbessern.

4.3.1 Manuelles Labeling

Das manuelle *Labeling* stellt den Ausgangspunkt des gesamten Prozesses dar und bildet die Grundlage für alle nachfolgenden Strategien. Es ist zu Beginn zwingend erforderlich, da ohne eine initiale Annotation durch einen Fachexperten kein Datensatz mit Annotationen zur Verfügung steht. Das manuelle *Labeling* wird nicht nur in der initialen Phase, sondern auch im weiteren Verlauf des Prozesses als Korrektur- und Validierungsschritt eingesetzt, um modellgestützte *Labels* zu überprüfen und gegebenenfalls zu korrigieren.

Manuelles Labeln ist sehr zeitintensiv und kognitiv anspruchsvoll, besonders bei großen Datenmengen. Um die Annotierenden zu entlasten und den Prozess effizienter zu gestalten, wird ein *Labeling*-Tool mit grafischer Benutzeroberfläche eingesetzt. Es bietet eine visuelle Darstellung der Datenverläufe und unterstützt die interaktive Markierung, Anpassung und Überprüfung von *Label*-Intervallen.

4.3.2 Regelbasiertes Labeling

Regelbasiertes *Labeling* nutzt vorhandenes Domänenwissen, um einfache, nachvollziehbare Regeln zu definieren, die auf den unannotierten Datensätz angewendet werden.

Die regelbasiert erzeugten *Pseudo-Labels* dienen als erste Orientierung für die Annotierenden und können potenzielle *Label*-Wechsel markieren. Diese automatisch erzeugten *Labels* müssen anschließend von den Annotierenden überprüft und korrigiert werden, um die *Pseudo-Labels* zu validieren. Dieser Ansatz bietet den Vorteil, dass bereits ohne ein trainiertes Modell eine erste Zuweisung von *Labels* erfolgen kann. Die Qualität der erzeugten *Pseudo-Labels* hängt dabei maßgeblich von der Erfahrung der Fachpersonen sowie davon ab, wie gut sich die Daten über Regeln definieren lassen.

Einen vergleichbaren Ansatz beschreiben Ratner et al. (2016) [37] mit dem Konzept des *Data Programming*. Dabei werden anstelle manueller Annotationen sogenannte *Labeling Functions* definiert, welche heuristische oder regelbasierte Strategien kodieren und automatisch *Labels* für Teilmengen der Daten erzeugen. Dieses Vorgehen ermöglicht es, große Trainingsdatensätze effizient zu erstellen und den manuellen *Labeling*-Aufwand signifikant zu reduzieren. Die Grundidee deckt sich mit dem in dieser Arbeit gewählten regelbasierten *Labeling*, das ebenfalls Domänenwissen in Form von Regeln operationalisiert, um eine erste *Pseudo*-Annotation der Daten zu

erzeugen, die anschließend durch Experten überprüft und korrigiert wird.

Die Umsetzung der regelbasierten Strategien erfolgt über ein Dialogfenster, in dem grundlegende Operatoren wie größer gleich, größer, kleiner gleich und kleiner ausgewählt werden können. Zusätzlich besteht die Möglichkeit, durch das Setzen von Klammern komplexere Bedingungen zu definieren, wodurch sich fachliches Wissen präziser in Regeln übertragen lässt. Die Darstellung der genauen Umsetzung erfolgt in Kapitel 5.

4.3.3 Modellgestütztes Labeling

Das modellgestützte *Labeling* kommt zum Einsatz, sobald ein Modell initial trainiert wurde. Ziel dieser Strategie ist es, mithilfe der Modellvorhersagen automatisch *Pseudo-Labels* zu generieren und dadurch den manuellen Aufwand zu verringern.

Die Qualität der vom Modell erzeugten *Labels* hängt unmittelbar von der Güte des zugrunde liegenden Trainingsdatensatzes sowie von der Modellarchitektur ab. In frühen Trainingsphasen ist die Vorhersagefähigkeit des Modells häufig noch begrenzt, weil es zunächst auf einer kleinen Menge gelabelter Daten basiert und die zugrunde liegenden Muster noch nicht vollständig erfasst hat. Mit zunehmender Menge an verifizierten und in das Training einbezogenen Daten verbessert sich die Generalisierungsfähigkeit des Modells, wodurch stabilere und präzisere Vorhersagen erzielt werden können.

Im Gegensatz zu regelbasierten Ansätzen, die auf festen Heuristiken beruhen, können Modelle ihr Wissen mit jedem Training erweitern und sich an die Charakteristik der Daten anpassen. Dadurch werden die vom Modell erzeugten *Pseudo-Labels* zunehmend verlässlicher, sodass neue Daten bei nachgewiesener Modellstabilität weitgehend automatisch annotiert werden können.

4.4 Modell

Die Wahl der Modellarchitektur orientiert sich an den Eigenschaften des im Kapitel 4.1 beschriebenen Datensatzes, der aus multivariaten Zeitreihen besteht und eine binäre Klassifikation erfordert. Entsprechend wird ein Modell eingesetzt, das zeitliche Abhängigkeiten zwischen den Merkmalen erfassen und beide Klassen zuverlässig unterscheiden kann. Die Wahl fiel auf ein LSTM-Modell, weil der Datensatz auf der Studie von Bakirtzis et al. [4] basiert, in der verschiedene Varianten dieser Architektur untersucht und durchweg zufriedenstellende Ergebnisse erzielt wurden.

In dieser Arbeit wird ein LSTM-Modell eingesetzt, das auf einer einzelnen LSTM-Schicht basiert. Das *Long Short-Term Memory*-Netzwerk eignet sich aufgrund seiner Fähigkeit, zeitliche Abhängigkeiten in Sequenzen zu erfassen, besonders für Klassifikationsaufgaben mit fortlaufenden Messwerten. Das Modell folgt einem sequenzi-

ellen Aufbau mit einer *many-to-many*-Beziehung, bei der jedem Eingabeelement x_i unmittelbar ein Ausgabeelement y_i zugeordnet wird, siehe Abbildung 8. Dadurch wird für jeden Zeitschritt eine separate Klassifikationsentscheidung getroffen. Diese Konfiguration ermöglicht eine präzisere Erfassung von Verlaufsänderungen innerhalb der Sequenzen und befähigt das Modell dazu, Klassenwechsel auch innerhalb einer Sequenz korrekt zu erkennen und entsprechend zu klassifizieren.

Das LSTM-Modell verarbeitet multivariate Zeitreihen, bei denen mehrere Eingangskanäle gleichzeitig berücksichtigt werden. Die internen Speichermechanismen des LSTM ermöglichen es, relevante Informationen über längere Zeiträume hinweg zu behalten und irrelevante Einflüsse zu unterdrücken. Nach jedem Training mit einer Sequenz werden die Zustände zurückgesetzt, um sicherzustellen, dass keine Informationen über Sequenzen hinweg übertragen werden. Dadurch bleiben unabhängige Sequenzen voneinander getrennt und das Modell lernt nur innerhalb jeder Sequenz den relevanten zeitlichen Kontext, ohne unerwünschte Kopplungen zu erzeugen.

Für die Bildung der Sequenzen wurde eine Fenstergröße von 60 Zeitschritten gewählt. Diese orientiert sich an der im Originalpaper [4] verwendeten Größe von 50 und wurde leicht erhöht, um längere zeitliche Abhängigkeiten innerhalb der Signale erfassen zu können.

Die Hyperparameter, darunter die Lernrate, die Anzahl der Einheiten in der LSTM-Schicht sowie die *Batch*-Größe, wurden im Rahmen eines Hyperparameter-Tunings bestimmt. Hierzu wurde die erste in Kapitel 4.1 beschriebene Trainingsdatei verwendet. Die erzeugten Sequenzen wurden anschließend in 80% und 20% Trainings- und Validierungsdaten separiert, auf deren Basis das Hyperparameter-Tuning durchgeführt wurde.

5 Implementierung

In diesem Kapitel wird die praktische Realisierung des in Kapitel 4.2beschriebenen Prozesses dargestellt.

Für die Umsetzung des Prozesses wurde eine grafische Benutzeroberfläche mit der Python-Bibliothek Tkinter implementiert. Sie dient als zentrales Werkzeug zur Steuerung und Durchführung der einzelnen Prozessschritte und ermöglicht eine strukturierte Interaktion während des gesamten Ablaufs. Der vollständige Quellcode zur Verwendung des Tools ist im zugehörigen GitHub-Repository[1] veröffentlicht.

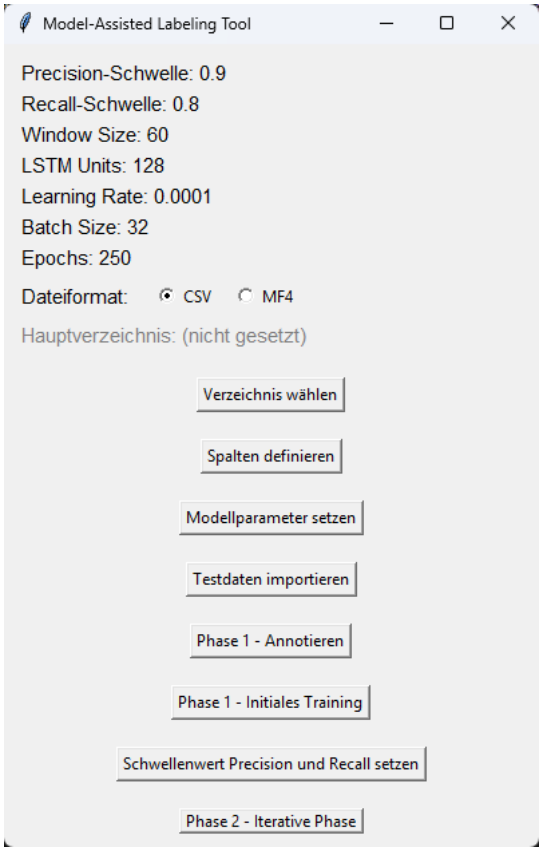

Abbildung 13: Darstellung der grafischen Benutzeroberfläche des Prozesses. Über die Buttons werden die einzelnen Prozessschritte für Phase 1 und Phase 2 ausgeführt.

Über die GUI, dargestellt in Abbildung 13, stehen verschiedene Funktionen in Form von *Buttons* zur Verfügung, die jeweils einen spezifischen Teilprozess initiieren und zusammen Phase 1 und Phase 2 bilden. Die einzelnen Ausführungen werden in den

[1] GitHub: https://github.com/NesperF/masterarbeit-model-assisted-labeling

nachfolgenden Paragraphen erklärt.

Dateiformat Zu Beginn muss das zu verwendende Dateiformat festgelegt werden. In der Benutzeroberfläche stehen hierfür die Optionen „CSV“ und „MF4“ zur Auswahl. Das gewählte Format bestimmt, in welchem Dateiformat die Eingangsdaten eingelesen und verarbeitet werden. Standardmäßig ist das Dateiformat CSV aktiviert. Die Unterstützung des Dateiformats von MF4 ermöglicht zusätzlich die Verarbeitung von Messdaten aus fahrzeugspezifischen Aufzeichnungssystemen.

Verzeichnis wählen Mit dem *Button* „Verzeichnis wählen“ muss ein Ordner als Arbeitsverzeichnis festgelegt werden, der als zentrale Ablage für alle im weiteren Verlauf erzeugten Artefakte dient. Über den *Button* wird ein Dialogfenster geöffnet, in dem der Benutzer ein bestehendes Verzeichnis auswählen oder ein neues Verzeichnis anlegen kann.

Nach der Auswahl wird der Pfad intern gespeichert und in der Benutzeroberfläche angezeigt, um die erfolgreiche Verknüpfung zu bestätigen. Bei der Auswahl eines neuen Verzeichnisses werden die Unterverzeichnisse für die definierten Spalten, das Modell sowie die Trainings- und Testdaten angelegt. Wurde das Verzeichnis bereits zuvor verwendet, werden die vorhandenen Ressourcen geladen.

Spalten definieren Die Ausführung von „Spalten definieren“ erfordert das Auswählen einer Datei, abhängig vom Dateiformat das definiert ist, deren enthaltene Spalten automatisch erkannt werden. Anschließend müssen die Spalten für das „Training“, die „Visualisierung“ und den „Export“ aus den erkannten Spalten ausgewählt sowie die „Zielspalte“ über ein Eingabefeld manuell angegeben werden.

In Abbildung 14 ist die Spaltenauswahl für das „Training“ dargestellt. Die Auswahldialoge für die „Visualisierung“ und den „Export“ besitzen eine identische Struktur und unterscheiden sich lediglich im Fenstertitel. Die blau markierten Spalten definieren diejenigen Attribute, die für das Training des Modells verwendet werden. Die Spalten für die „Visualisierung“ werden für die Darstellung beim manuellen *Labeling* in der graphischen Benutzeroberfläche verwendet. Die „Export“ Spalten definieren die Spalten die nach Abschluss der Annotation von einer Datei gespeichert werden sollen. Die „Zielspalte“ ist diejenige Spalte, die die *Labels* enthält. Das Eingabefeld zur Angabe der „Zielspalte“ ist in Abbildung 15 dargestellt.

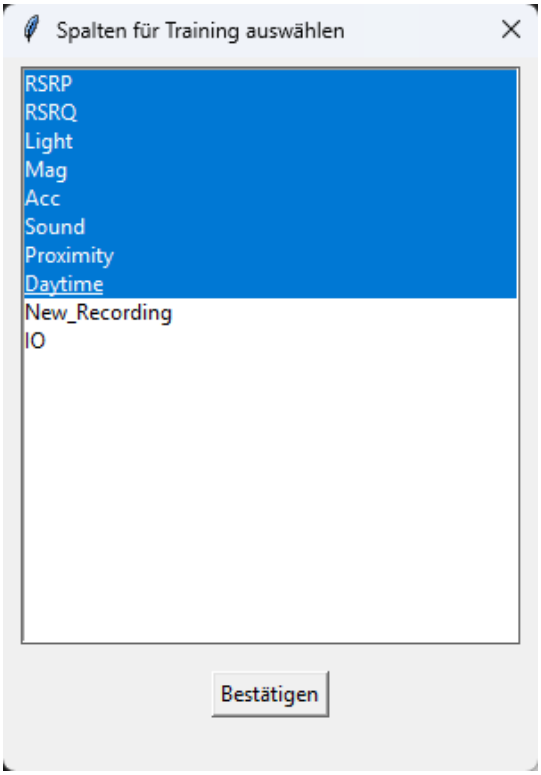

Abbildung 14: Dialogfenster zur Auswahl der Trainingsspalten. Mehrere Spalten können gleichzeitig markiert und für das Training berücksichtigt werden.

Testdaten importieren Über den *Button* „Testdaten importieren“ muss eine Datei ausgewählt werden, die als Testdatensatz dient. Nach dem Aufruf öffnet sich ein Dialogfenster, in dem eine Datei ausgewählt wird. Beim Importvorgang werden die Daten der Datei in Sequenzen umgewandelt und im definierten Verzeichnis abgelegt.

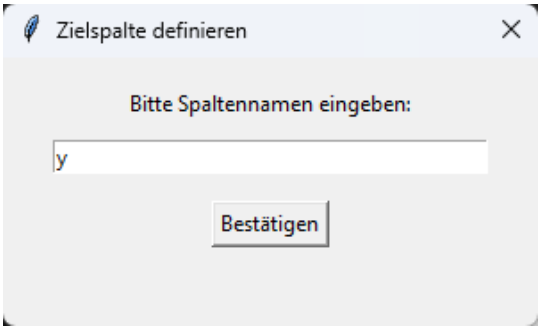

Abbildung 15: Dialogfenster zur Eingabe der Zielspalte, die als *Label*-Variable im Trainingsprozess verwendet wird.

Phase 1 - Annotieren Mit der Ausführung von „Phase 1 - Annotieren“ wird ein manueller *Labeling*-Prozess gestartet, indem eine Datei ausgewählt werden muss, die annotiert werden soll. Bevor die Visualisierung der Daten erfolgt, öffnet sich ein Regelfenster, siehe Abbildung 16, in dem optional Regeln angegeben werden können. Abbildung 16 zeigt beispielhaft den Dialog zur Definition von Regeln, in den mehrere Bedingungen kombiniert und logisch verknüpft werden können.

Im Regelfenster kann mit dem Button „Regel hinzufügen“ eine neue Zeile zur Definition weiterer Regeln erzeugt werden. In oberen Bereich befindet sich das Feld „Ausdruck“, in dem der vollständige logische Ausdruck der aktuell definierten Regeln angezeigt wird. Obwohl es optisch einem Eingabefeld ähnelt, dient es ausschließlich der Darstellung der Regelkombination und kann vom Benutzer nicht direkt bearbeitet werden. Dadurch bleibt die logische Struktur zu der Regeldefinition jederzeit nachvollziehbar.

In den nachfolgenden Zeilen erfolgt die eigentliche Definition: optional beginnend mit einer öffnenden Klammer, gefolgt von der Auswahl einer Spalte die im Schritt „Spalten definieren“ für das „Training“ festgelegt wurde, einem Operator, dem Vergleichswert und einer schließenden Klammer. Werden Klammern oder logische Verknüpfungen nicht korrekt gesetzt, erscheint eine Fehlermeldung, und die Eingabe muss angepasst werden. Mit dem Button „Entfernen“ kann eine bestehende Regel gelöscht werden, während „Abbrechen“ den Regelprozess beendet, ohne eine Regel anzuwenden. Durch die Bestätigung mit „Ok“ werden die definierten Regeln übernommen und auf die Daten angewandt. Anschließend werden die Daten der Datei im *Labeling*-Tool visualisiert.

Das *Labeling*-Tool besteht aus zwei übereinander angeordneten Darstellungsbereichen, dargestellt in Abbildung 18. Im oberen Fenster werden die ausgewählten Sensormesswerte der Kategorie „Visualisierung“ dargestellt, während im unteren Fenster die als „Zielspalte“ festgelegte *Label*-Variable angezeigt wird. Der Wertebereich der y-Achse kann intuitiv durch Klicken und Ziehen mit der linken Maustaste im oberen Fenster angepasst werden, wobei stets alle Zeitschritte sichtbar bleiben. Eine dynamische Anpassung des y-Bereichs ist notwendig, da bei multivariaten Werten häufig stark unterschiedliche Wertebereiche auftreten können. Über den Button „Reset-View“ wird der y-Bereich automatisch auf den globalen Wertebereichs aller Variablen zurückgesetzt. Zur Vergabe eines *Labels* wählt der Benutzer die Funktion „Set Value“ und markiert im oberen Fenster mit gedrückter linker Maustaste ein Intervall, das anschließend grün hervorgehoben wird. Nach dem Loslassen öffnet sich ein Eingabefeld, in das der *Label*-Wert, 0 oder 1, eingetragen wird. Im unteren Fenster erscheint daraufhin eine Linie, die das markierte Intervall entsprechend der gewählten Klasse kennzeichnet. Nach der Annotation eines Abschnitts kann über „Save and Next“ zum nächsten Zeitfenster gewechselt werden. Über dem oberen Fenster wird der Fortschritt im Form der Gesamtanzahl der zu bearbeitenden Dia-

gramme angezeigt. Nach Abschluss aller Abschnitte wird der Benutzer aufgefordert, einen Speicherort zu wählen, an dem die final annotierte Datei im ausgewählten Dateiformat abgelegt wird. Der Speichervorgang ist verpflichtend und kann nicht abgebrochen werden, um den Verlust einer annotierten Datei zu verhindern.

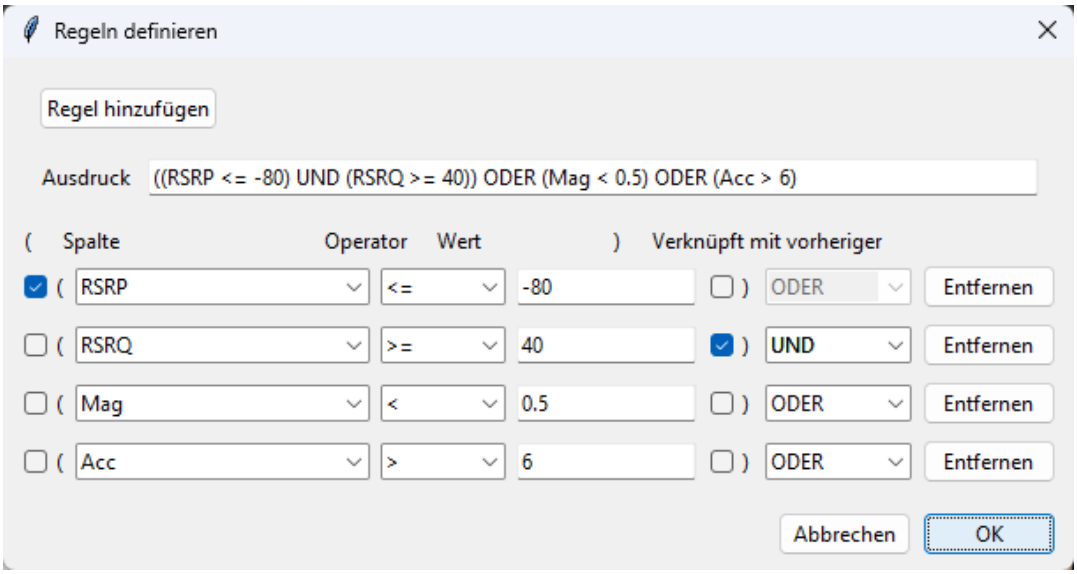

Abbildung 16: Dialogfenster zur Definition von Regeln, mit denen Bedingungen über verschiedene Sensormesswerte formuliert und logisch verknüpft werden können.

Nach erfolgreichem Speichern beginnt der Prozess erneut, wobei der Benutzer die Möglichkeit besitzt, eine weitere Datei zu importieren und zu annotieren. Auf diese Weise können mehrere Dateien in Folge verarbeitet werden, ohne den *Button* erneut drücken zu müssen. Wird keine Datei ausgewählt, wird der Vorgang abgebrochen.

Modellparameter setzen Über die Schaltfläche „Modellparameter setzen“ öffnet sich ein Dialogfenster, in dem die zentralen Parameter des LSTM-Modells festgelegt werden, siehe Abbildung 17. Der Benutzer hat die Möglichkeit, Parameter wie „Window Size“, „LSTM Units“, „Learning Rate“, „Batch Size“ und „Epochs“ manuell anzupassen.

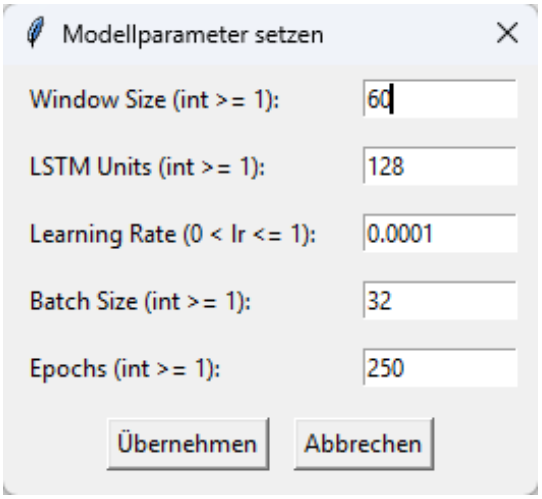

Abbildung 17: Dialogfenster zur Einstellung der Modellparameter. Es können Parameter wie „Window Size“, „LSTM Units“, „Learning Rate“, „Batch Size“ und Anzahl der „Epochs“ festgelegt werden.

Die Eingabefelder sind mit Gültigkeitsbereichen versehen, um ungültige Werte zu vermeiden. Nach Bestätigung mit „Übernehmen“ werden die angegebenen Parameter gespeichert und für das zukünftige Modelltraining verwendet, während „Abbrechen“ den Vorgang ohne Änderungen beendet. Die Änderungen sind anschließend in der Hauptoberfläche sichtbar und werden dort im oberen Bereich unter den jeweiligen Parameterbezeichnungen angezeigt.

Phase 1 - Initiales Training Nach Abschluss von „Phase 1 - Annotieren“ liegen vollständige annotierte Datensätze vor, von dem einer als Grundlage für den *Button* „Phase 1 - Initiales Training“ dient. In diesem Schritt wird ein annotierter Datensatz importiert, in Sequenzen überführt, gemischt und anschließend für das initiale Modelltraining verwendet. Gleichzeitig wird der verarbeitete Datensatz gespeichert, um die Wiederverwendung in „Phase 2 - Iterative Phase“ sicherzustellen. Abschließend wird das trainierte Modell im definierten Verzeichnis gespeichert, sodass es ebenfalls in „Phase 2 - Iterative Phase“ erneut verwendet wird.

Abbildung 18: *Labeling*-Tool zur Visualisierung und Annotation der Zeitreihendaten. Im oberen Bereich werden die ausgewählten Sensordaten für die Visualisierung dargestellt, im unteren Bereich die Zielspalte.

Schwellenwert Precision und Recall setzen Über die Schaltfläche „Schwellenwert Precision und Recall setzen“ öffnet sich ein Dialogfenster, das in seiner Struktur dem Fenster zum Setzen der Modellparameter entspricht, siehe Abbildung 17. In diesem Dialog werden die Schwellenwerte für die Metriken *Precision* und *Recall* festgelegt. Diese Werte definieren, welche Mindestgüte das Modell auf dem Testdatensatz erreichen muss, um anschließend mit seinen eigenen Vorhersagen weitertrainiert zu werden, ohne dass diese manuell überprüft werden müssen.

Phase 2 - Iterative Phase Die Schritte zu Beginn des *Buttons* „Phase 2 - Iterative Phase“ entsprechen grundsätzlich denen von „Phase 1 - Annotieren“, unterscheiden sich jedoch darin, dass anstelle von Regeln die Modellvorhersagen als *Pseudo-Labels* verwendet werden. Diese werden vom trainierten Modell aus „Phase 1 - Initiales Training“ generiert und in der Benutzeroberfläche visualisiert, wie in Abbildung 18 dargestellt.

Beim erstmaligen Aufruf der „Phase 2 - Iterative Phase“ ist eine manuelle Überprüfung und Korrektur der vom Modell erzeugten *Pseudo-Labels* erforderlich. Die Korrektur erfolgt über das bereits vorgestellte *Labeling*-Tool in Abbildung 18. Nach

dem anschließenden Training wird anhand der Ergebnisse auf dem Testdatensatz überprüft, ob die definierten Schwellenwerte aus „Schwellenwert Precision und Recall setzten“ über- oder unterschritten wurden.

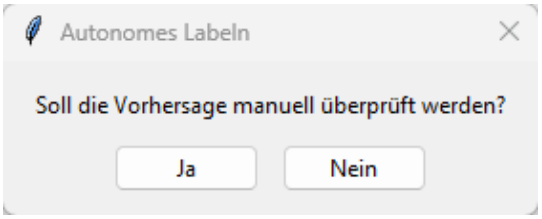

Abbildung 19: Dialogfenster der Phase 2, in dem der Benutzer gefragt wird, ob die vom Modell erzeugten *Pseudo-Labels* manuell überprüft werden sollen, bevor das Training fortgesetzt wird.

Wird die Schwelle unterschritten, bleibt der Prozess bei der nächsten Vorhersage identisch zum bisherigen Ablauf.

Wird die Schwelle hingegen überschritten, erscheint nach der nächsten Vorhersage des Modells das in Abbildung 19 dargestellte Dialogfenster, über das der Benutzer gefragt wird, ob dennoch eine manuelle Überprüfung und Korrektur erfolgen soll. Bei der Auswahl von „Ja“ wird diese wie zuvor über das *Labeling*-Tool durchgeführt. Wird die Option „Nein“ gewählt, trainiert das Modell mit seinen eigenen, unkorrigierten Vorhersagen weiter. Diese Abfrage erscheint bei jeder neuen Vorhersage, solange das Modell die definierten Schwellenwerte für *Precision* und *Recall* überschreitet.

Nach der Bearbeitung einer Datei wird der Benutzer aufgefordert, die Datei zu speichern, wie bereits bei dem *Button* „Phase 1 - Annotieren“. Anschließend folgt die Option, eine neue Datei auszuwählen, um den Prozess fortzusetzen. Erfolgt keine Auswahl, wird die Phase beendet.

6 Evaluierung

In diesem Kapitel wird der entwickelte modellgestützte *Labeling*-Prozess systematisch ausgewertet. Dazu werden verschiedene Trainingsstrategien verglichen, der manuelle *Labeling*-Aufwand über die Iterationen hinweg analysiert und regelbasierte *Labeling*-Strategien anhand statistischer Kennzahlen bewertet.

6.1 Ziel der Evaluierung

Ziel der Evaluierung ist es, den in Kapitel 4.2 beschriebenen *Labeling*-Prozess hinsichtlich seiner Wirksamkeit und Effizienz zu bewerten, in Bezug auf die Reduzierung des manuellen *Labeling*-Aufwands sowie die erzielte Modellleistung. Im Mittelpunkt steht der Vergleich zwischen einem vollständig überwachten Referenzmodell, das ausschließlich mit korrekt annotierten *Labels* trainiert wird, und einer automatisierten Variante, die ab einem definierten Schwellenwert mit ihren eigenen, nicht verifizierten Vorhersagen weitertrainiert. Auf diese Weise wird untersucht, wie sich die Nutzung von Selbstkorrektur auf die Modellgüte auswirkt und in welchem Umfang der manuelle *Labeling*-Aufwand reduziert werden kann. Ergänzend erfolgt eine Analyse regelbasierter *Labeling*-Strategien, um deren Potenzial als Unterstützung für Annotierende zu bestimmen.

6.2 Aufbau der Evaluierung

Die Evaluierung besteht aus zwei voneinander unabhängigen Teilprozessen, die in den Abbildungen 20 und 21 dargestellt sind. Beide Abläufe beginnen jeweils an dem grün markierten Knoten und basieren auf der im Kapitel 4.1 definierten ersten Trainingsdatei.

Im Rahmen dieser Arbeit stand kein Domänenwissen über den Datensatz zur Verfügung und aus Zeitgründen wurde keine manuelle Annotation durchgeführt, weshalb die Evaluierung vollständig automatisiert erfolgt.

Der erste Prozess, die Regel-Evaluierung, dient der quantitativen Untersuchung von regelbasierten *Labeling*-Strategien. Ausgangspunkt ist die erste annotierte Trainingsdatei, aus der die statistischen Kennzahlen Minimum, Maximum, Mittelwert, 95- und 99-Quantil bestimmt werden. Diese Kennzahlen werden anschließend auf dem Testdatensatz aus Kapitel 4.1 ausgewertet, um die Effektivität und die Generalisierbarkeit der definierten Regeln zu bewerten.

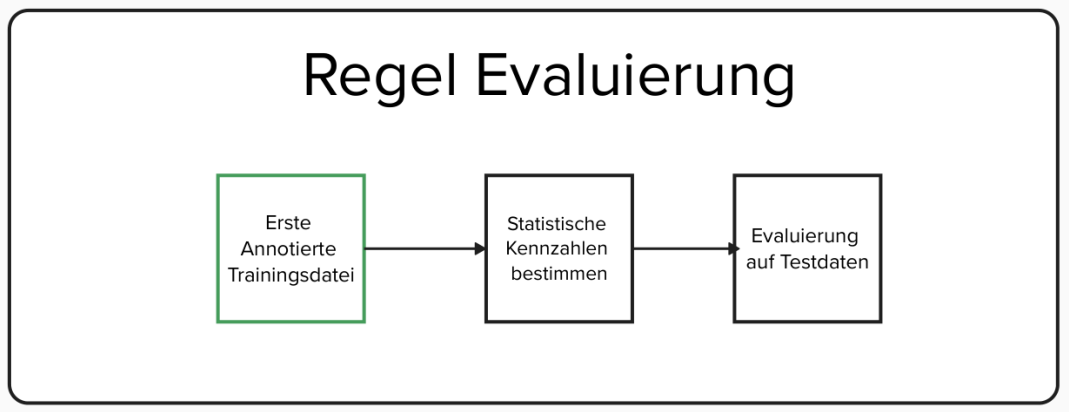

Abbildung 20: Angepasster zweiphasiger Prozess zur Evaluierung der Modellleistung.

Der zweite Prozess stellt den automatisierten Ablauf des modellgestützten *Labeling*-Prozesses dar. Dabei werden die im Trainingsdatensatz enthaltenen Dateien iterativ importiert, vom Modell klassifiziert und abhängig vom erreichten Schwellenwert überprüft oder unbeaufsichtigt weiterverarbeitet. Nach jeder Iteration erfolgt eine Vorhersage auf dem Testdatensatz, um die Modellgüte zu bestimmen und die Entscheidung für die nächste Iteration zu steuern. Wird der definierte Schwellenwert überschritten, entfällt die automatische Überprüfung, sodass das Modell mit seinen eigenen Vorhersagen weitertrainiert.

Zur Bewertung des automatisierten modellgestützten Ansatzes wird ein Überprüft-Modell herangezogen, das in allen Iterationen ausschließlich mit den tatsächlichen, korrekt annotierten *Labels* trainiert wird. Dadurch kann der Einfluss der automatisierten Selbstkorrektur auf die Modellgüte quantifiziert und die Leistungsfähigkeit des automatisierten Trainingspfads im Vergleich zu vollständig überwachten Trainingsabläufen beurteilt werden.

Beide Evaluierungsprozesse zusammen ermöglichen eine ganzheitliche Bewertung des Gesamtsystems. Die Regel-Evaluierung untersucht die Nützlichkeit regelbasierter *Labeling*-Strategien, während die Prozess-Evaluierung den Einfluss des automatisierten Weitertrainings auf den manuellen Aufwand sowie auf die Modellleistung analysiert.

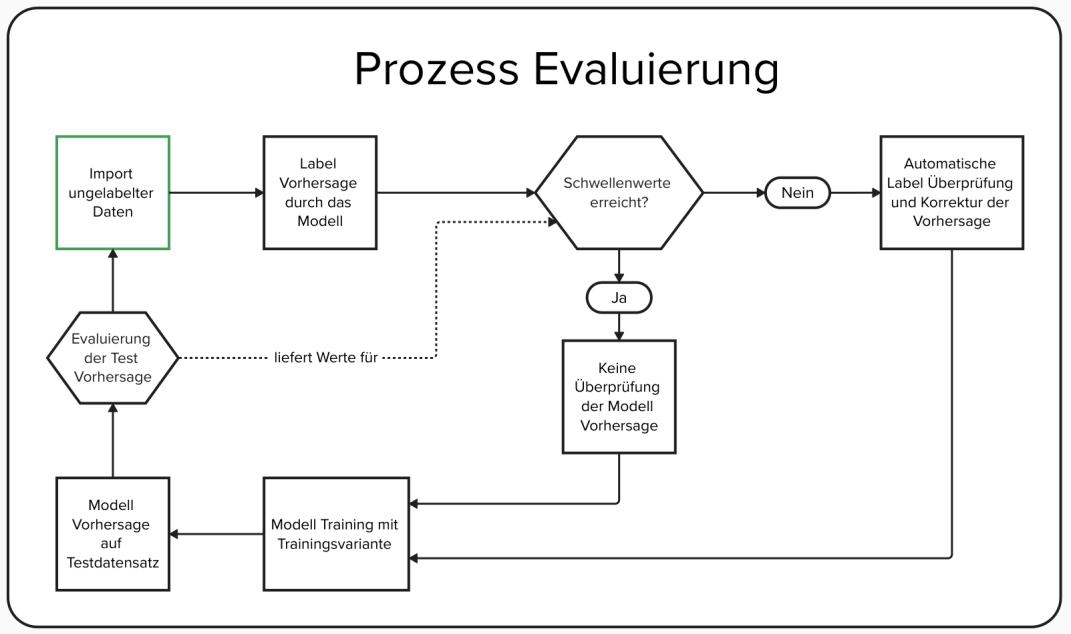

Abbildung 21: Angepasster zweiphasiger Prozess zur Evaluierung der Modellleistung.

6.3 Regelbasierte Evaluationsmethodik

Für die Bewertung regelbasierter *Labeling*-Strategien wurde die erste Trainingsaufzeichnung herangezogen und in zwei Teilmengen unterteilt, die den Klassen $IO = 0$ und $IO = 1$ entsprechen. Für beide Klassen wurden auf Basis sämtlicher Merkmale statistische Kennwerte berechnet, um charakteristische Unterschiede zwischen den Klassen zu identifizieren. Dazu zählen der Mittelwert, das Minimum, das Maximum, sowie das 95- und 99-Quantil:

$$\text{Kennwerte} = \{\mu, \min, \max, Q_{95}, Q_{99}\},$$

wobei μ den Mittelwert und Q_p das p-Quantil eines Merkmals bezeichnet. Aus diesen Kennwerten wurden anschließend die Regeln abgeleitet, die eine Zuordnung neuer Beobachtungen zu den Klassen $IO = 0$ oder $IO = 1$ ermöglichen. Die Definition der Regeln erfolgte nach folgenden Kriterien.

Mittelwertsbasierte Regeln Für jedes Merkmal x wurde der Abstand zum Klassenmittelwert betrachtet. Die Regel wählt die Klasse, deren Mittelwert dem beobachteten Wert x_i näher liegt:

$$R_{\text{mean}}(x_i) = \underset{c \in \{0,1\}}{\arg\min} \; |x_i - \mu_c|. \tag{8}$$

Maximumbasierte Regeln Eine Regel auf Basis des Maximums wurde nur dann definiert, wenn der maximale Wert der positiven Klasse ($IO = 1$) größer ist als der der negativen Klasse ($IO = 0$):

$$\max(x_1) > \max(x_0).$$

In diesem Fall gilt die Entscheidungsregel:

$$R_{\max}(x_i) = \begin{cases} 1, & \text{wenn } x_i > \max(x_0) \\ 0, & \text{sonst.} \end{cases} \tag{9}$$

Minimumbasierte Regeln Analog wurde eine Regel nur dann definiert, wenn der minimale Wert der positiven Klasse ($IO = 1$) kleiner ist als der der negativen Klasse ($IO = 0$):

$$\min(x_1) < \min(x_0).$$

Dann gilt die Entscheidungsregel:

$$R_{\min}(x_i) = \begin{cases} 1, & \text{wenn } x_i < \min(x_0) \\ 0, & \text{sonst.} \end{cases} \tag{10}$$

Quantilbasierte Regeln Für die Quantile $p \in \{0.95, 0.99\}$ wurden Regeln nur dann gebildet, wenn das Quantil der positiven Klasse ($IO = 1$) größer ist als das der negativen Klasse ($IO = 0$):

$$Q_p(x_1) > Q_p(x_0).$$

Ist die Bedingung erfüllt, gilt die Entscheidungsregel:

$$R_{Q_p}(x_i) = \begin{cases} 1, & \text{wenn } x_i > Q_p(x_0) \\ 0, & \text{sonst.} \end{cases} \tag{11}$$

Diese Definitionen gewährleisten, dass Beobachtungen oberhalb des typischen Wertebereichs von $IO = 0$ als wahrscheinlich zu $IO = 1$ gehörend klassifiziert werden.

Nachdem alle Regeln definiert wurden, wurden sie auf den Testdatensatz angewandt. Für jede Regel wurde überprüft, ob die Klassifikation mit dem tatsächlichen Label übereinstimmt. Auf Basis dieser Ergebnisse wurden die Kennzahlen *Precision* und *Recall* berechnet.

6.4 Trainingsstrategien

Im Rahmen der Prozess-Evaluierung werden verschiedene Trainingsstrategien untersucht, um den Einfluss unterschiedlicher Trainingsabläufe auf die Modellgüte und den manuellen *Labeling*-Aufwand zu bewerten. Ziel ist es, zu ermitteln, welche Vorgehensweise eine möglichst stabile Verbesserung der Modellleistung führt und

zugleich den manuellen *Labeling*-Aufwand am effektivsten reduziert. Die Strategien unterscheiden sich darin, ob das Modell nach jeder Vorhersage vollständig neu initialisiert und trainiert wird oder auf Basis seines bisherigen Zustands weitertrainiert wird, sowie darin, ob die Modellvorhersagen auf einer vollständigen Datei oder lediglich auf einen Teilbereich einer Datei basieren.

Es werden insgesamt vier Varianten betrachtet:

Erste: Die Vorhersage erfolgt auf einer vollständigen Datei und das Modell wird unter Einbeziehung der bisherigen Daten neu trainiert. Diese Strategie repräsentiert ein vollständig überwachtes Training, bei dem alle verfügbaren Daten genutzt werden.

Zweite: Die Vorhersage erfolgt auf einer vollständigen Datei, das Modell wird diesmal jedoch weitertrainiert. Dabei werden alte und neue Daten in einem Verhältnis von eins zu zwei berücksichtigt, um den Einfluss der neu hinzugekommenen Informationen zu betonen und den Anpassungsprozess zu beschleunigen. Diese Strategie zielt auf eine fortlaufende Wissensakkumulation ab, birgt jedoch das Risiko des *Catastrophic Forgetting*, also des schrittweisen Verdrängens zuvor gelernter Muster durch neu hinzukommende Informationen.

Dritte: Die Vorhersage erfolgt auf einem Teilbereich einer Datei und das Modell wird anschließend neu trainiert. Auch dadurch soll überprüft werden, ob bereits begrenzte zusätzliche Annotationen ausreichen, um durch ein vollständiges Neutraining eine signifikante Verbesserung der Modellleistung zu erreichen.

Vierte: Die Vorhersage erfolgt auf einem Teilbereich einer Datei, jedoch wie in der zweiten Variante, werden alte und neue Daten in einem Verhältnis von eins zu zwei kombiniert, um neue Muster schneller zu adaptieren und gleichzeitig den Rechenaufwand zu einem vollständigen Neutraining zu verringern.

Beim vollständigen Neutraining werden alle Modellparameter vor jedem Training neu initialisiert, wodurch potenzielle Pfadabhängigkeiten vermieden werden. Das Weitertrainieren hingegen erfolgt auf Basis des zuvor gelernten Zustands und erlaubt eine fortlaufende Anpassung an neue Daten. Beide Verfahren unterscheiden sich hinsichtlich ihres Rechenaufwands und der Art, wie Modellgewichte aktualisiert werden.

Die Trainingsdaten werden vor jedem Training als Sequenzen neu vermischt und anschließend zufällig in Trainings- und Validierungsdaten aufgeteilt, wobei 80% der Sequenzen für das Training und 20% für die Validierung genutzt werden. Die zeitliche Struktur innerhalb der einzelnen Sequenzen bleibt dabei vollständig erhalten, um eine Durchmischung von Zeitschritten zu verhindern und die zeitlichen Abhängigkeiten korrekt abzubilden.

Zusätzlich kommen zur Stabilisierung des Lernprozesses *EarlyStopping* und *ReduceLROnPlateau* zum Einsatz. *EarlyStopping* beendet das Training, sobald sich die Validierungsleistung über mehrere Epochen hinweg nicht verbessert, wodurch *Overfitting* vermieden und Rechenzeit reduziert wird. *ReduceLROnPlateau* senkt die Lernrate adaptiv, wenn keine weitere Verbesserung auftritt, und ermöglicht es dem Modell, ein Optimum präziser zu erreichen. Vor jedem Trainingsdurchlauf wird die Lernrate auf ihren Ausgangswert zurückgesetzt, um eine einheitliche Ausgangsbasis zu gewährleisten und sicherzustellen, dass neue Daten während des Lernprozesses ausreichend Einfluss auf die Gewichtsanpassung erhalten.

6.5 Prozess Evaluationsmethodik

Zur Bewertung der Modellgüte und des manuellen Aufwands werden verschiedene Metriken herangezogen, die sowohl die Vorhersagequalität und den manuellen Aufwand abbilden. Die Analyse stützt sich dabei auf die Kennzahlen *Precision* und *Recall* und den daraus abgeleiteten manuellen *Labeling*-Aufwand.

Die *Precision* beschreibt den Anteil der korrekt als positiv klassifizierten Beobachtungen an allen positiven Vorhersagen und gibt damit an, wie zuverlässig die positiven Klassifikationen des Modells sind. Der *Recall* misst den Anteil der korrekt erkannten positiven Ereignisse an allen tatsächlich positiven Beobachtungen und bewertet somit die Fähigkeit des Modells, alle relevanten Ereignisse zu identifizieren. Beide Metriken werden nach jeder Trainingsiteration auf dem Testdatensatz berechnet, um die Entwicklung der Modellleistung über den gesamten Evaluierungsprozess hinweg zu verfolgen.

Die Entscheidung des Schwellenwertes, ob das Modell ohne Korrektur weitertrainiert, basiert auf festgelegten Grenzwerten der verwendeten Kennzahlen. Die in Tabelle 11 aufgeführten Werte definieren die Bedingungen, unter denen das Modell als hinreichend zuverlässig eingestuft wird. Wird eine *Precision* von mindestens 90% und gleichzeitig ein *Recall* von mindestens 80% erreicht, werden die vom Modell erzeugten Vorhersagen unbeaufsichtigt in den Trainingsprozess einfließen.

Kennzahl	Schwellenwert
Precision	90%
Recall	80%

Tabelle 11: Schwellenwerte für die autonome Weiterverwendung der Modellvorhersagen.

Die Werte aus Tabelle 11 orientieren sich an typischen Übereinstimmungen zwischen menschlichen Annotierenden und gewährleisten, dass das Modell erst dann autonom trainiert, wenn eine vergleichbare Zuverlässigkeit erzielt wird [19, 28].

Zur Berechnung des manuellen *Labeling*-Aufwands wird der Anteil der durch das Modell fehlerhaft klassifizierten Beobachtungen berechnet. Dieser ergibt sich aus der Summe der FP und FN Vorhersagen im Verhältnis zur Summe aller Vorhersagen:

$$A_{\text{Aufwand}} = \frac{FP + FN}{TP + TN + FP + FN} \times 100. \tag{12}$$

Ein niedriger Wert von $A_{Aufwand}$ bedeutet, dass das Modell nur wenige Korrekturen erfordert und den Annotierenden entlastet, während ein hoher Wert auf einen entsprechend größeren manuellen Aufwand hinweist.

Zur Einordnung der Modellleistung werden zusätzlich Referenzwerte für *Precision* und *Recall* bestimmt, die unter optimalen Bedingungen, eine maximal erreichbare Modellfähigkeit darstellen. Die Referenzwerte wurden durch ein Hyperparameter-Tuning bestimmt, bei dem alle verfügbaren Trainingsdateien als Trainingssatz und sämtliche Testdateien als Validierungssatz verwendet wurden, um auf Basis des vollständigen Datensatzes die optimalen Modellparameter zu ermitteln. Anschließend wurde das Modell mit den gefunden Parametern auf allen Trainingsdaten trainiert und auf dem Testdatensatz evaluiert. Tabelle 6.5 zeigt die verwendeten Modellparameter sowie die daraus resultierenden Referenzwerte für *Precision* und *Recall*.

Fenstergröße	**Einheiten**	**Lernrate**	**Batch Größe**	**Ref. Precision**	**Ref. Recall**
60	192	0.001	256	96%	94%

Tabelle 12: Modellparameter und Referenzwerte (Ref.). Fenstergröße gibt die Anzahl der Zeitschritte pro Sequenz an. Die Einheiten beziehen sich auf die Anzahl der LSTM Einheiten. Die Lernrate beschreibt, wie stark die Gewichte im Training angepasst werden. Die Batch-Größe bezeichnet die Anzahl der Sequenzen pro Zeitschritt.

Die Kombination dieser Metriken ermöglicht eine Bewertung des Gesamtsystems. *Precision* und *Recall* dienen der Beurteilung der Modellgüte und der Entscheidungssteuerung im automatisierten *Labeling*-Prozess, während der *Labeling*-Aufwand als Maß für die tatsächliche Reduktion menschlicher Annotationstätigkeit herangezogen wird.

6.6 Ergebnisse

Die folgenden Unterkapitel präsentieren die Ergebnisse der Regel-Evaluierung sowie die Ergebnisse der Prozess-Evaluierung, die sowohl den manuellen *Labeling*-Aufwand als auch die Testergebnisse umfasst.

6.6.1 Regelbasiertes Labeling

Abbildung 22 zeigt die Ergebnisse der Regel-Evaluierung anhand der Metriken *Precision* und *Recall*. Der Name einer Regel, wie beispielsweise „max_light“ bezeichnet die Kombination aus der verwendeten Entscheidungsregel und dem zugehörigen Merkmal.

Die Auswertung zeigt, dass von den getesteten Maximum-Regeln lediglich eine Regel, die „max_light“, die erforderlichen Bedingungen für die Definition erfüllt. Diese Regel erzielt eine hohe *Precision*, identifiziert jedoch weniger als 20% der tatsächlich positiven Instanzen, was sich in einem entsprechend niedrigen *Recall* zeigt.

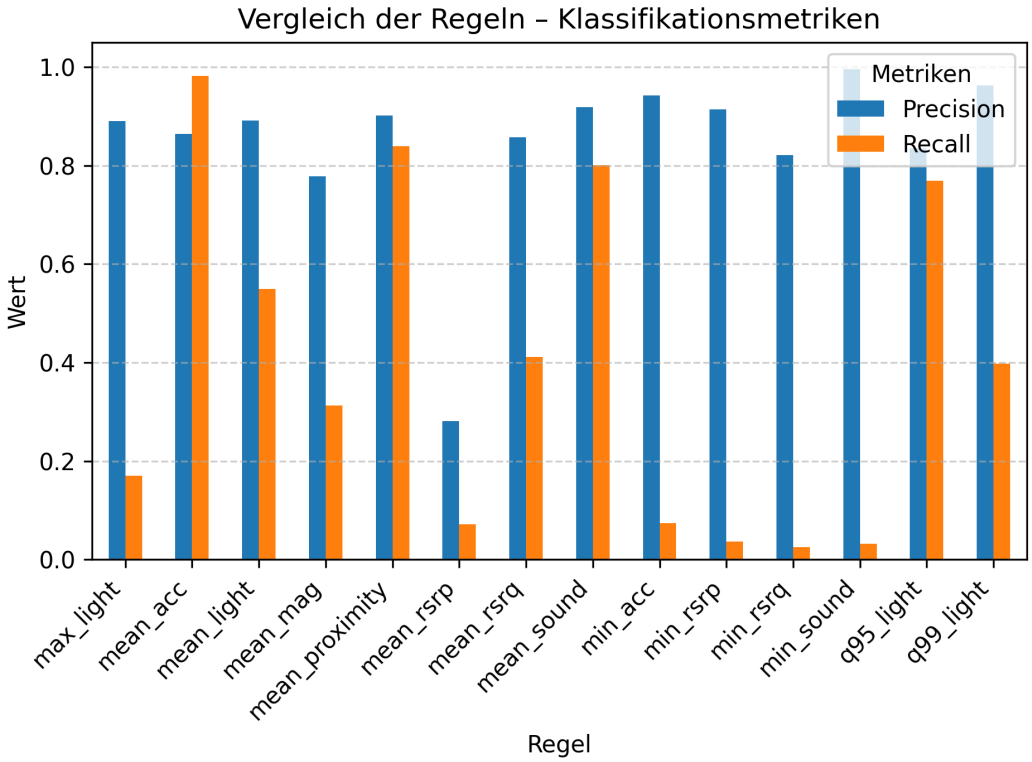

Abbildung 22: Vergleich der definierten Regeln anhand der Metriken *Precision* und *Recall*. Dargestellt sind die für jedes Merkmal abgeleiteten Entscheidungsregeln und deren Klassifikationsgüte auf dem Testdatensatz.

Die Quantil-basierten Regeln liegen ausschließlich für das Merkmal „Light“ vor. Die Regel „q99_light“ erreicht eine nahezu perfekte *Precision*, jedoch bei gleichzeitig rund 40% *Recall*. Die Regel „q95_light“ weist eine kombinierte Ausprägung beider Kennzahlen auf und erzielt somit eine *Precision* von etwa 85% sowie einen *Recall* von

knapp unter 80%.

Die betrachteten Minimum-Regeln erreichen ausnahmslos eine hohe *Precision*, zeigen jedoch durchweg einen sehr niedrigen *Recall* Wert von unter 10%. Dies bedeutet, dass diese Regeln nur einen kleinen Teil der positiven Instanzen erfassen.

7 von insgesamt 14 der definierten Regeln basieren auf dem Mittelwert. Diese Regeln weisen insgesamt eine relativ konstante Ausprägung von *Precision* und *Recall* auf und erreichen im Vergleich aller Regeln, die höchste gemeinsame Ausprägung beider Kennzahlen. Die Merkmale „Acceleration“, „Proximity“ und „Sound“ zeigen die höchsten *Recall*-Werte innerhalb der betrachteten Regeln, während Merkmale wie „RSRP“, „RSRQ“ und „Magnitude“ deutlich geringere *Recall*-Werte aufweisen.

6.6.2 Manueller Labeling-Aufwand

Die Ergebnisse zeigen den manuellen *Labeling*-Aufwand gemäß Gleichung (12). In allen folgenden Abbildungen werden das Autonom-Modell, das ab den in Tabelle 11 definierten Schwellenwerten ohne Überprüfung weitertraininert, dem Überprüft-Modell gegenübergestellt, das in jeder Iteration ausschließlich mit verifizierten *Labels* trainiert wird. Zur besseren Übersichtlichkeit wurden sämtliche Werte blockweise aggregiert. Für jeweils fünf aufeinanderfolgende Iterationen wurden die Minimal- und Maximalwerte der betrachteten Metriken bestimmt und dargestellt. Die gestrichelte Linien kennzeichnet den Minimalwert, während die durchgezogene Linie den Maximalwert innerhalb des jeweiligen Fünferblocks repräsentiert. Somit wird die Spannweite der Schwankungen sichtbar, wodurch sowohl stabile als auch stark variierende Iterationen im Verlauf besser erkennbar werden.

Die Abbildungen 23, 24, 25 und 26 zeigen den Verlauf des manuellen *Labeling*-Aufwands $A_{Aufwand}$ über alle Iterationen hinweg. Für die Abbildungen 23 und 24 entspricht eine Iteration der Vorhersage auf einer vollständigen Datei, während sie für die Abbildungen 25 und 26 der Vorhersage auf einem Teilbereich der Datei entspricht. Das anschließende Weitertraining des Modells wird in den Abbildungen 23 und 25 dargestellt, während in den Abbildungen 24 und 26 eine Neuinitialisierung des Modells mit anschließendem Training auf allen verfügbaren Trainingsdaten erfolgt.

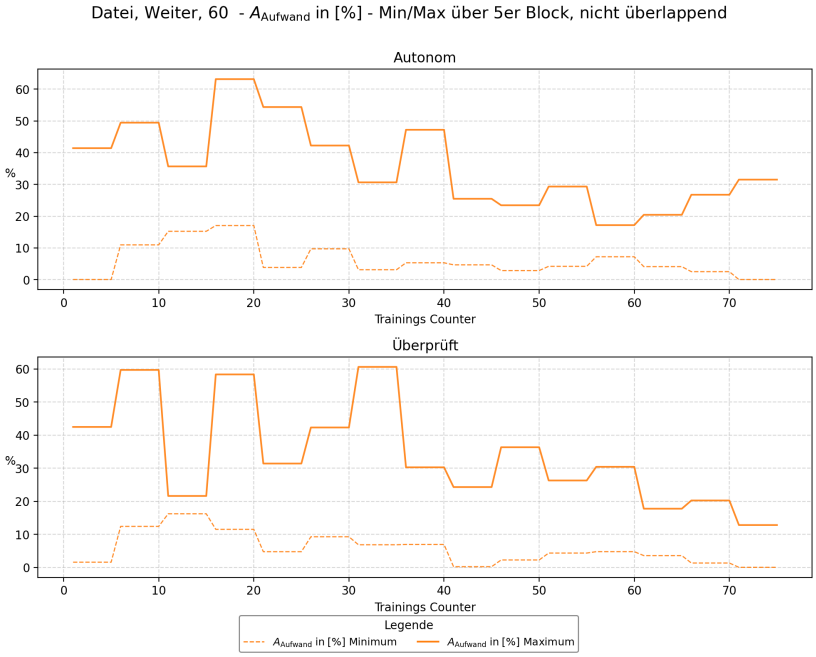

Abbildung 23: Manueller Aufwand ($A_{Aufwand}$) für die Autonom- und Überprüft-Modelle je Trainingsiteration, basierend auf Vorhersagen über eine gesamte Datei und anschließendem Weitertraining. Die Werte werden als Minimum und Maximum über nicht überlappende 5er-Blöcke dargestellt.

In Abbildung 23 liegen für das Autonom-Modell die Werte zwischen 0% und etwa 62%. Die Maximalwerte zeigen im Verlauf Veränderungen, darunter ein Anstieg zwischen dem dritten und dem vierten Block. Der untere Teil der Abbildung zeigt die Werte für das Überprüft-Modell. Auch hier verändern sich die Maximalwerte über den Iterationsverlauf und liegen bis zu fast 40% auseinander und erreichen Werte bis 60%. Die Minimalwerte liegen in allen bis auf drei Blöcken, also 15 Iterationen, im einstelligen Prozentbereich. Der Abstand zwischen Minimalwert und Maximalwert variiert in beiden Modellen über die Iterationen hinweg und zeigt sowohl engere als auch breitere Abstände.

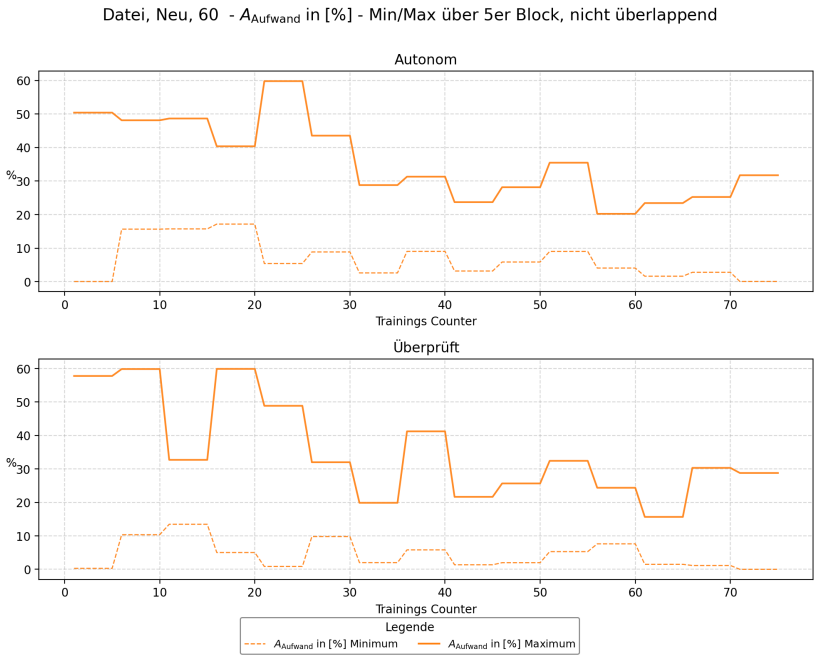

Abbildung 24: Manueller Aufwand ($A_{Aufwand}$) für die Autonom- und Überprüft-Modelle je Trainingsiteration, basierend auf Vorhersagen über eine gesamte Datei und anschließendem Neu-Initialisieren und erneutem Trainieren der Modelle. Die Werte werden als Minimum und Maximum über nicht überlappende 5er-Blöcke dargestellt.

In Abbildung 24 beginnt der Maximalwert bei 50% und endet im Verlauf bei etwa 30%. Vom vierten auf den fünften Block steigt der Maximalwert um 20% auf 60% an, gefolgt von weiteren Änderungen, die einzelne Anstiege von etwa 5% umfassen. Der Minimalwert weist zu Beginn bei beiden Modellen zweimal eine Zunahme auf und nimmt anschließend über die Iterationen hinweg kontinuierlich bis auf 0% ab, wobei kleinere Zwischenanstiegen auftreten. Beim Überprüft-Modell sinkt der Maximalwert vom zweiten zum dritten Block um nahezu 30% und steigt im anschließenden Block wieder an.

Abbildung 25: Manueller Aufwand ($A_{Aufwand}$) für die Autonom- und Überprüft-Modelle je Trainingsiteration, basierend auf Vorhersagen auf einem Teilbereich der Datei und anschließendem Weitertraining. Die Werte werden als Minimum und Maximum über nicht überlappende 5er-Blöcke dargestellt.

In Abbildung 25 liegen die Maximalwerte im oberen Verlauf zwischen 0% und 100%. Die Minimalwerte bewegen sich überwiegend im Bereich zwischen 0% und 25%, während sie beim Überprüft-Modell zwischen 0% und 15% liegen. In beiden Fällen variiert der Abstand von Minimum und Maximum von geringen Abständen bis hin zu 90% Abständen. In einigen Blöcken weisen beide Modelle gleichzeitig einen Minimal- und Maximalwert von 0% auf, während in der Mehrzahl der Blöcke der Minimalwert bei 0% liegt.

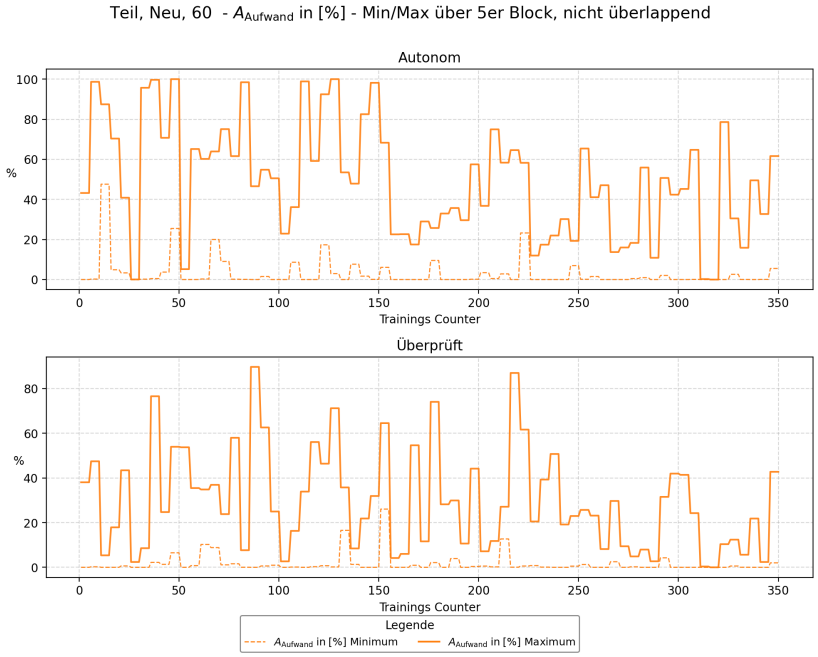

Abbildung 26: Manueller Aufwand ($A_{Aufwand}$) für die Autonom- und Überprüft-Modelle je Trainingsiteration, basierend auf Vorhersagen auf einem Teilbereich der Datei und anschließendem Neu-Initialisieren sowie erneutem Trainieren der Modelle. Die Werte werden als Minimum und Maximum über nicht überlappende 5er-Blöcke dargestellt.

In Abbildung 26 bewegen sich die Maximalwerte im oberen Verlauf zwischen 0% und 100%, wobei der Verlauf mehrfach steigende und sinkende Werte aufweist. Der Minimalwert liegt zwischen 0% und knapp 50% und bleibt bis auf drei Blöcke unter 20%. Beim Überprüft-Modell erreicht der Maximalwert 90% und zeigt ebenfalls mehrfache Änderungen mit steigenden und sinkenden Werten. Die Minimalwerte liegen überwiegend zwischen 0% und 5%, mit einzelnen Anstiegen bis knapp 30%.

6.6.3 Testergebnisse

Die Abbildungen zeigen die Ergebnisse des Autonom-Modell und Überprüft-Modell auf den Testdaten hinsichtlich der *Precision* und *Recall*. Zusätzlich sind die Referenzwerte aus Tabelle 6.5 eingezeichnet. Ergänzend ist ein dritter Verlauf dargestellt, der für jede Iteration ausweist, ob die definierten Schwellenwerte für *Precision* und *Recall* erfüllt wurden. Dieser Verlauf dient der direkten Darstellung der Schwellenüberschreitungen, da durch die blockweise Aggregation, wie im Paragraph zum manuellen *Labeling*-Aufwand beschrieben, die Minimal- und Maximalwerte nicht erkennen lassen, in wie vielen Iterationen die Referenzwerte erreicht wurden und ab

welchem Zeitpunkt das Autonom-Modell ohne Korrektur weitertrainiert.

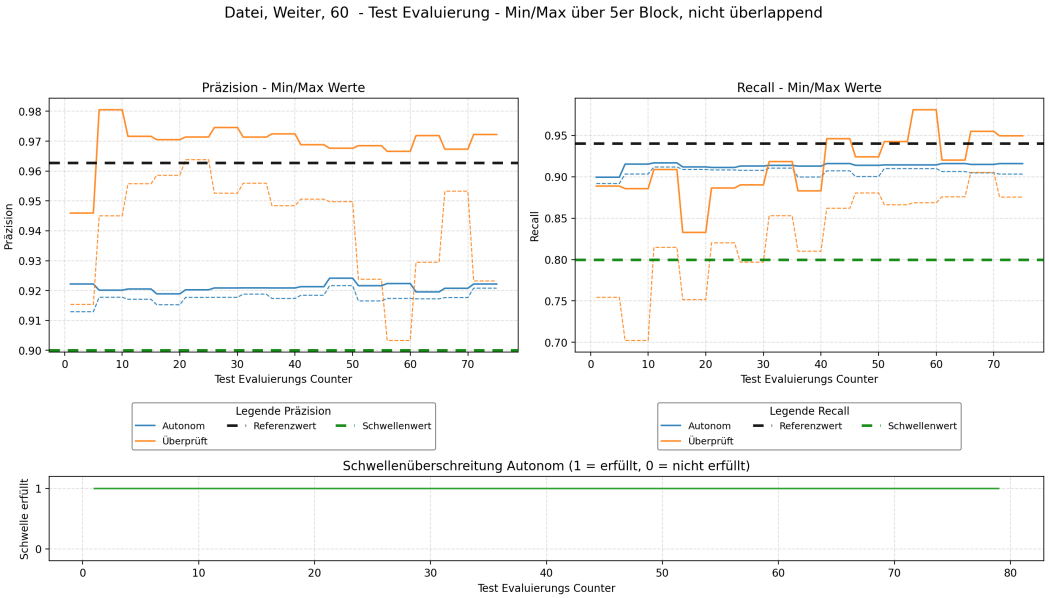

Abbildung 27: Ergebnisse der Testvorhersage für die Autonom- und Überprüft-Modelle nach dem Training mit Vorhersagen über eine gesamte Datei und anschließendem Weitertraining. Die Werte werden als Minimum und Maximum über nicht überlappende 5er-Blöcke dargestellt.

In Abbildung 27 erreicht der Verlauf des Autonom-Modells in allen Iterationen für beide Metriken nicht die festgelegten Referenzwerte. Der Abstand zwischen Minimal- und Maximalwert erreicht einen Abstand von maximal 1%. Die Abstände der Metriken für das Überprüft-Modell betragen für die *Precision* etwa 8% und für den *Recall* bis nahezu 20%. Bei der *Precision* erreicht das Überprüft-Modell im zweiten Block mit dem Maximalwert den Referenzwert. Für den *Recall* liegt der Maximalwert fünfmal über dem Referenzwert und unterschreitet ihn in den übrigen Blöcken. Die Schwellenwertüberschreitung zeigt über alle Iterationen hinweg den Wert 1.

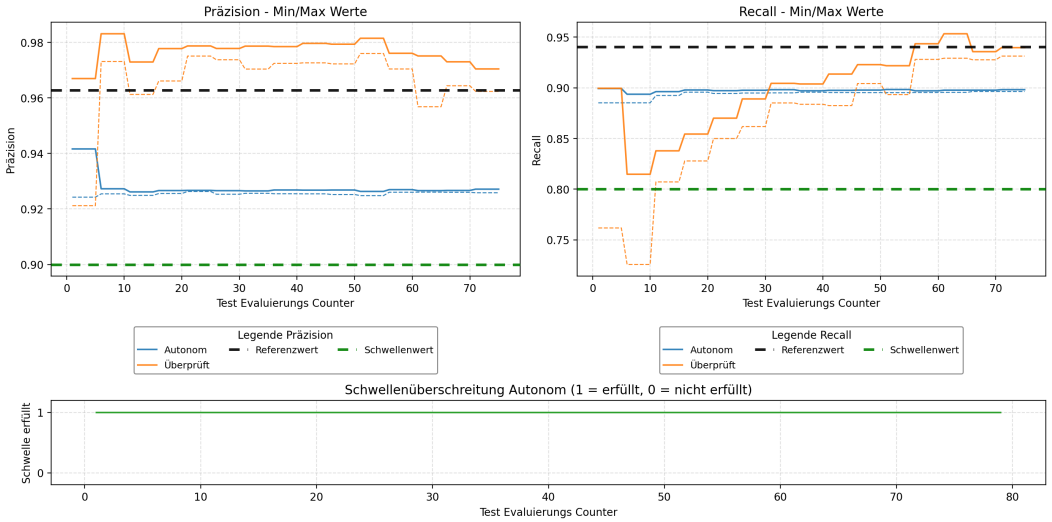

Abbildung 28: Ergebnisse der Testvorhersage für die Autonom- und Überprüft-Modelle nach dem Training mit Vorhersagen über eine gesamte Datei und anschließendem Neu-Initialisieren und erneutem Trainieren der Modelle. Die Werte werden als Minimum und Maximum über nicht überlappende 5er-Blöcke dargestellt.

In Abbildung 28 liegen *Precision* und *Recall* des Autonom-Modells in allen Iterationen unterhalb der Referenzwerte, jedoch durchgehend oberhalb der festgelegten Schwellenwerte. Daher nimmt der Verlauf der Schwellenüberschreitung in jeder Iteration den Wert 1 an. Für das Überprüft-Modell wird der Referenzwert der *Precision* beim Maximalwert in allen Blöcken erreicht, jedoch unterschreitet der Minimalwert dreimal den Wert. Für den *Recall* zeigt sich zu Beginn ein leichter Abfall, gefolgt von einem nahezu kontinuierlichen Anstieg, sodass der Referenzwert ab der 55 Iteration beim Maximalwert für zwei aufeinanderfolgende Blöcke überschritten wird.

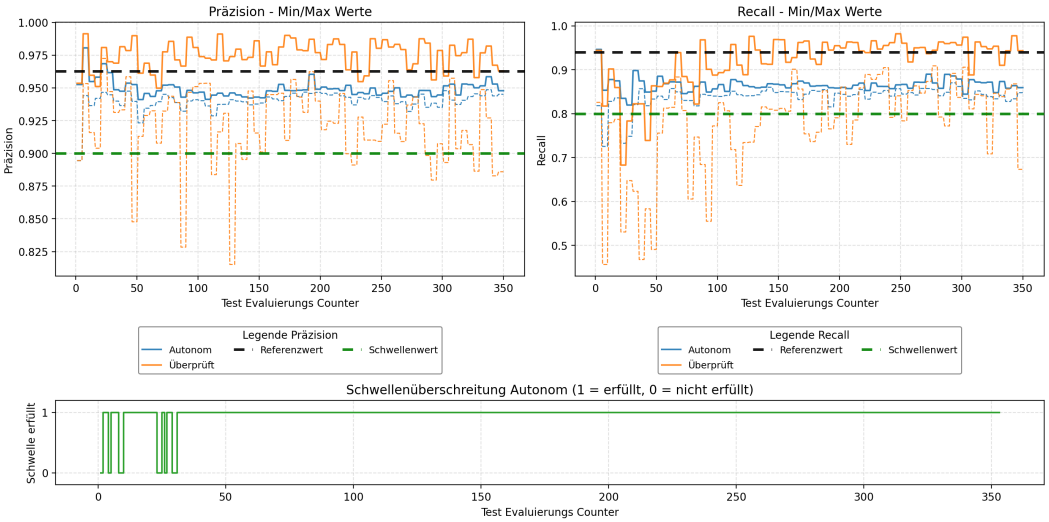

Abbildung 29: Ergebnisse der Testvorhersage für die Autonom- und Überprüft-Modelle nach dem Training mit Vorhersagen auf einem Teilbereich der Datei und anschließendem Weitertraining. Die Werte werden als Minimum und Maximum über nicht überlappende 5er-Blöcke dargestellt.

In Abbildung 29 liegt das Autonom-Modell bei beiden Metriken nahezu über den gesamten Iterationsverlauf knapp unter den Referenzwerten und besitzt einen sehr geringen Abstand zwischen Minimal- und Maximalwert. Das Überprüft-Modell zeigt in den ersten 125 Iterationen wechselnde Abstände, die von etwa 2% bis hin zu größeren Spannweiten von bis zu 15% für die *Precision* und bis zu 40% für den *Recall* reichen. Die Maximalwerte erreichen überwiegend die Referenzwerte für beide Metriken. Der Schwellenverlauf wechselt in den ersten rund 40 Iterationen sechsmal zwischen den Werten 0 und 1 und liegt in den darauffolgenden Iterationen bei 1.

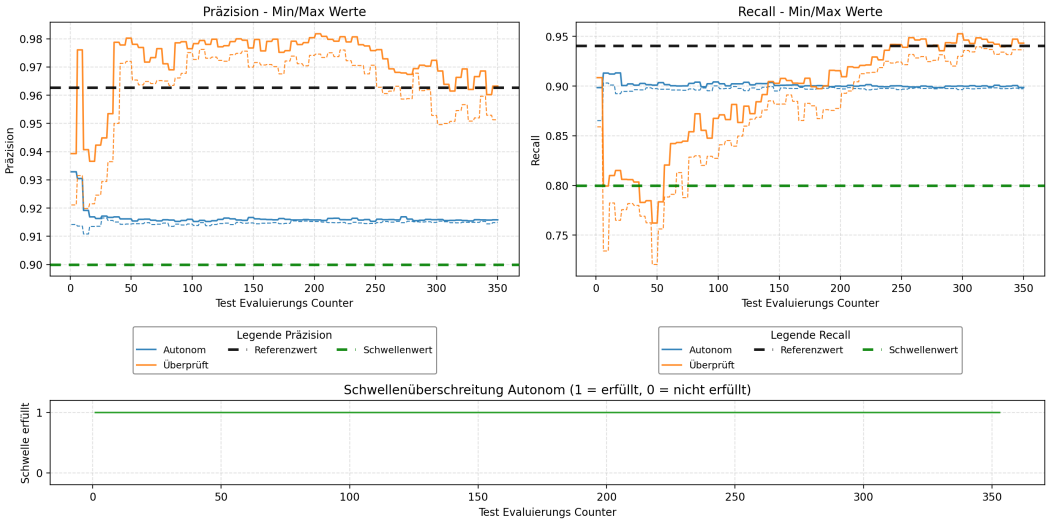

Abbildung 30: Ergebnisse der Testvorhersage für die Autonom- und Überprüft-Modelle nach dem Training mit Vorhersagen auf einem Teilbereich der Datei und anschließendem Neu-Initialisieren und erneutem Trainieren der Modelle. Die Werte werden als Minimum und Maximum über nicht überlappende 5er-Blöcke dargestellt.

In Abbildung 30 erreicht das Autonom-Modell in beiden Verläufen nicht einmal die Referenzwerte, jedoch besitzen beide Metriken einen geringen Abstand der Minimal- und Maximalwerte. Das Überprüft-Modell erreicht in der Iteration 40 bis 45 den Referenzwert für die *Precision* und bleibt dann bis Iteration 255 oberhalb für Minimal- und Maximalwert. Beim *Recall* sinken die Werte zu Beginn, steigen dann stetig an und erreichen den Referenzwert ab Iteration 250 das erste Mal, aber nur für den Maximalwert. Der Abstand der Werte beträgt maximal 7%. Der Schwellenverlauf für das Autonom-Modell liegt ab der ersten Iteration durchgehend bei 1.

7 Diskussion der Ergebnisse

Im folgenden werden die Ergebnisse für die Regel-Evaluierung, den manuellen *Labeling*-Aufwand $A_{Aufwand}$ und der Testergebnisse diskutiert.

7.1 Regelbasiertes Labeling

Die Ergebnisse der regelbasierten Strategien zeigen, dass einfache Entscheidungsregeln, abhängig vom Merkmal, einen Teil der positiven Instanzen zuverlässig identifizieren können. Damit leisten sie einen Beitrag zur Reduzierung des manuellen *Labeling*-Aufwands, da der Annotierende nicht jede Instanz korrigieren muss. Die Leistungsfähigkeit dieser Regeln hängt jedoch stark von der Informationsqualität des jeweiligen Merkmals ab. Merkmale mit geringer Varianz oder überlappenden Klassenbereichen liefern nur eingeschränkte Unterstützung, während strukturierte Merkmale zu stabileren Ergebnissen führen.

Im Gegensatz zur ersten Vorhersage in den Abbildungen der Testergebnissen, die auf einem initial trainierten Modell basiert, wurden die regelbasierten Strategien direkt auf den Testdaten angewendet. Sie dienen damit als Referenz für die erzielbare Vorhersagequalität, ohne zuvor ein Modell trainieren zu müssen. Die Testergebnisse zeigen, dass die Modelle bereits nach dem ersten Training bei vielen Konfigurationen sowohl in *Precision* als auch in *Recall* Werte von über 90% erreichen. Einige Modelle erzielen jedoch insbesondere beim *Recall* niedrigere Ausgangswerte im Bereich von etwa 75%, was die Spannweite der anfänglichen Modellleistung verdeutlicht. Verglichen mit diesen Testergebnissen liefern die regelbasierten Strategien insgesamt geringere und stärker variierende Werte, was ihre begrenzte Fähigkeit zur präzisen Abbildung der Klassenstruktur unterstreicht. Sie können daher als unterstützende Methode dienen, bevor ein Modell trainiert ist, erreichen jedoch nicht die Konsistenz und Genauigkeit der modellgestützten Verfahren.

Die Evaluierung der regelbasierten Strategien erfolgte auf dem Testdatensatz, sodass die Regeln auf Daten aller Aufzeichnungen angewendet wurden und ihre Generalisierbarkeit beurteilt werden kann. Durch die Verwendung lediglich der letzten 20% jeder Aufzeichnung wirkt sich ein möglicher zeitlicher *Data Drift* innerhalb einzelner Aufzeichnungen nur geringfügig auf die Bewertung aus. Auf neuen, nicht annotierten Daten passen sich die Regeln jedoch nicht eigenständig an Veränderungen der Datenverteilung an. Tritt bei neuen Daten ein *Data Drift* auf kann die Unterstützung der Regeln deutlich beeinträchtigt werden. Modelle können in einem solchen Szenario grundsätzlich durch erneutes Training an veränderte Verteilungen angepasst werden. In den Ergebnissen zum manueller *Labeling*-Aufwand zeigt sich, dass der Aufwand auf neu hinzukommenden Daten in den Abbildungen 25 und 26 deutlich schwankt. In den Abbildungen 23 und 24 treten ebenfalls Schwankungen auf, allerdings in einem deutlich geringeren Ausmaß als bei den Regelergebnissen. Dieser Befund verdeutlicht, dass die Anpassung des Modells an zuvor unbekannten

Daten nicht in jedem Fall gelingt und die Verteilung neuer Daten einen erheblichen Einfluss auf die Vorhersagequalität ausübt. Auf dem Testdatensatz bleibt die Modellvorhersage nach jedem Training stabil, was die Unterschiede zwischen kontrollierten und unveränderten Datenverteilungen verdeutlicht.

Die Regeln basieren ausschließlich auf aus der Datenverteilung abgeleiteten Schwellenwerten. Eine Limitation besteht darin, dass nicht untersucht werden konnte, ob fachlich begründete Entscheidungsregeln zu ähnlichen, besseren oder stabileren Ergebnissen führen würden. Somit bleibt offen, in welchem Umfang die beobachtete Leistungsfähigkeit der Regeln von der Datenstruktur abhängt und ob Domänenwissen eine präzisere Abbildung der Klassen ermöglichen kann.

Zusammenfassend lässt sich festhalten, dass die regelbasierten Strategien im Rahmen dieser Arbeit einen begrenzten, aber nachvollziehbaren Beitrag leisten konnten. Sie übernehmen funktional die Rolle eines datenbasierten Ersatzes für Domänenwissen und verdeutlichen, welches Potenzial gut gewählte Regeln für die Reduzierung des manuellen *Labeling*-Aufwands bieten können.

7.2 Manueller Labeling-Aufwand

Die Ergebnisse des manuellen *Labeling*-Aufwands zeigen deutliche Unterschiede zwischen den Trainingsvarianten, die auf vollständigen Dateien erfolgt und jenen, bei denen nur Teilbereiche einer Datei vorhergesagt werden. Vollständige Dateien enthalten mehr Zeitschritte und damit eine breitere Mischung verschiedener Merkmalszustände. Teilbereiche einer Datei umfassen dagegen nur kurze Ausschnitte und weisen daher weniger repräsentative Abschnitte eines vollständigen Ablaufs auf. Dies führt zu deutlich ausgeprägteren Schwankungen, da kurze Abschnitte stärker von den zuvor gelernten Mustern abweichen können.

Besonders die Varianten mit Teilbereichen weisen sehr starke Veränderungen des Aufwands auf. Die Werte reichen von nahezu keinen erforderlichen Korrekturen bis hin zu sehr hohen Aufwänden, was auf eine deutlich geringere Vorhersagestabilität hinweist. Diese Instabilität tritt unabhängig vom verwendeten Modell auf. Damit zeigt sich, dass die Schwankungen weniger durch das Modell selbst verursacht werden, sondern primär durch die Eigenschaften der eingehenden Daten oder der Trainingsmethode. Wie in den Abbildungen 27 bis 30 mit der Schwellenwertdarstellung sichtbar wird, überschreiten drei der vier Autonom-Modelle die definierten Schwellenwerte bereits ab der ersten Testvorhersage, sodass sie von Beginn an automatisiert weitertrainieren. Dies bedeutet, dass ein großer Teil der Iterationen ohne Korrektur der Vorhersagen erfolgt und Fehler in der autonomen Phase direkt in das Modell zurückfließen, dadurch verstärkt sich die Instabilität zusätzlich. In der Abbildung 29 erfolgen zwar einige Wechsel zwischen überprüftem und autonomen Training, doch der Anteil der überprüften Phasen bleibt insgesamt gering. Damit

basiert der überwiegende Teil der Iterationen auf Vorhersagen, die ohne Korrektur in das Training zurückfließen. Diese Dynamik verstärkt die Schwankungen des Autonom-Modells zusätzlich, da diese in einzelnen Iterationen die Marke von 100% erreicht, wohingegen das Überprüft-Modell maximal etwa 95% aufweist.

Neben den starken Schwankungen zeigen die Ergebnisse in der letzten Iteration, dass die Modelle beim Maximalwert ähnliche Werte aufweisen, jedoch weiterhin leichte Unterschiede bestehen bleiben. Besonders deutlich wird dies in den Abbildungen 23 und 26, in denen ein 20% Unterschied bei den Maximalwerten auffällt. In den Abbildungen 24 und 25 liegen die Werte dagegen sehr nah beieinander und in Abbildung 25 erreicht das Autonom-Modell einen geringeren Maximalwert als das Überprüft-Modell. Dies zeigt, dass selbst in späten Iterationen keine vollständige Konvergenz erreicht wird und dass die Modelle weiterhin unterschiedlich auf Daten reagieren.

Bei Trainingsvarianten, die ihre Vorhersage auf einer vollständigen Datei durchführen, also in Abbildung 23 und 24, kann der Einfluss neuer Aufzeichnungen am besten beobachtet und bewertet werden. Jede Iteration entspricht einer neuen Aufzeichnung, aufgrund der 5er-Aggregation umfasst jeder Block fünf neue Aufzeichnungen. Starke Veränderungen des manuellen *Labeling*-Aufwands lassen darauf schließen, dass die neuen Aufzeichnungen entweder ein zuvor nicht gesehenes Muster aufweisen, welches das Modell bislang nicht gelernt hat, oder dass die Muster sehr ähnlich zu den bisherigen Trainingsdaten sind und somit vom Modell gut klassifiziert werden können.

Ein Vergleich der Abbildungen 23 und 24 zeigen zusätzliche Unterschiede zwischen den Iterationen zehn und elf. Beide Überprüft-Modelle weisen eine deutliche Reduzierung des Aufwands auf, was darauf hindeutet, dass die dort verarbeiteten Aufzeichnungen ähnliche Merkmalsverläufe zu den zuvor verarbeiteten Daten besitzen. Die Autonom-Modelle zeigen in diesem Block hingegen nur eine geringe Abnahme oder sogar eine Zunahme des Aufwands. Dies lässt darauf schließen, dass sich Fehler aus früheren autonomen Iterationen fortpflanzen und das Modell dadurch empfindlicher auf neue Merkmalsverläufe reagiert, während das Überprüft-Modell stabiler bleibt.

Zusammenfassend zeigen die Ergebnisse zum manuellen Aufwand, dass die Stabilität der Modellvorhersagen maßgeblich von den Mustern der eingehenden Daten abhängt. Vorhersagen und Training auf vollständigen Dateien führen zu deutlich gleichmäßigeren Aufwandsverläufen, während Vorhersage und Training auf Teilbereiche starke und teils sprunghafte Veränderungen aufweisen. Das Überprüft-Modell bleibt über den gesamten Verlauf hinweg robuster gegenüber neuen oder abweichenden Mustern, wohingegen das Autonom-Modell aufgrund der Fehlerfortpflanzung stärker schwankt. Insgesamt wird deutlich, dass eine dauerhaft niedrige Reduktion

des manuellen Aufwands schwer zu erreichen ist, insbesondere dann, wenn kontinuierlich neue Daten mit bislang unbekannten Mustern hinzukommen, da selbst korrekt annotierte *Labels* die damit verbundene Veränderungen nicht vollständig ausgleichen können.

7.3 Testergebnisse

Die Analyse der Testergebnisse verdeutlicht nochmals die Unterschiede zwischen den Autonom- und Überprüft-Modellen. Die Autonom-Modelle bleiben in allen Varianten über den gesamten Iterationsverlauf hinweg unter den definierten Referenzwerten. Gleichzeitig weisen sie nur sehr geringe Abstände zwischen Minimal- und Maximalwerten auf, was auf einen stabilen, jedoch konsistenten nicht optimalen Vorhersagezustand hindeutet. Dieses Verhalten lässt darauf schließen, dass die autonomen Iterationen kaum effektive Lernfortschritte erzeugen. Stattdessen stabilisiert das Modell früh erlernte Muster, die durch fehlende korrigierende *Labels* nicht weiter angepasst werden. Fehler in frühen Iterationen können sich somit über den gesamten Prozess hinweg fortsetzen.

Im Gegensatz dazu zeigen die Überprüft-Modelle eine Verbesserung der Vorhersagen im Verlauf der Iterationen. Die Entwicklung ist eine direkte Folge der kontinuierlich bereitgestellten, korrekt annotierten *Labels*, die das Modell iterativ korrigieren und Fehlentscheidungen aus vorherigen Iterationen gezielt reduzieren. Besonders im *Recall* wird ein klarer Lernfortschritt sichtbar. In allen Überprüft-Modellen steigt der *Recall* kontinuierlich an und erreicht in den späteren Iterationen mit dem Maximalwert den definierten Referenzwert, während der Minimalwert zu keinem Zeitpunkt den Referenzwert erreicht. Dies verdeutlicht, dass das Modell mit zunehmender Menge verifizierter *Labels* in der Lage ist, zuvor übersehen positive Instanzen zu erkennen und seine Entscheidungsgrenzen gezielt anzupassen.

Auch die *Precision* der Überprüft-Modelle zeigt insgesamt einen Lernverlauf, wobei sich die Stabilität zwischen den Trainingsvarianten deutlich unterscheidet. In den Varianten Abbildung 28 und 30 bleibt der Abstand zwischen Minimal- und Maximalwert überwiegend eng. Diese geringen Spannweiten deuten darauf hin, dass das Modell in diesen Varianten sehr konsistente Entscheidungen bezüglich der Klassifizierung trifft. Ursächlich hierfür ist die Trainingsstrategie mit Neuinitialisierung, bei der das Modell in jeder Iteration mit allen bislang verfügbaren Trainingsdaten vollständig neu trainiert wird. Dadurch erhält das Modell ein stabiles und gleichbleibend strukturiertes Lernverhalten und der Einfluss einzelner abweichender Trainingsaufzeichnungen wird deutlich reduziert. Fehlklassifikationen aus vorherigen Iterationen können sich nicht fortpflanzen, sodass die Vorhersagen nur eine geringe Varianz aufweist. Im Gegensatz dazu fallen die Spannweiten in Abbildung 29 deutlich stärker aus. Besonders im *Recall* zeigen sich ausgeprägte Einbrüche, bei denen die Werte in einzelnen Iterationen deutlich unter 50% sinken und sehr große Abstände zwischen

Minimal- und Maximalwerten auftreten. Diese starke Varianz weist darauf hin, dass das Modell in dieser Variante besonders sensibel auf Unterschiede in den jeweils neu hinzukommenden Trainingsaufzeichnungen reagiert. Eine mögliche Erklärung dafür ist, dass das Weitertrainieren ohne Neuinitialisierung das Modell stärker an die zuletzt verarbeiteten Muster bindet oder zu einer temporären Abschwächung zuvor gelernter Strukturen führen kann. Denkbare Mechanismen umfassen eine verstärkte Anpassung an einzelne, abweichende Aufzeichnungen oder Effekte inkrementellen Überanpassens. Diese Mechanismen wurden im Rahmen der Arbeit nicht explizit analysiert, daher lässt sich das Verhalten nicht auf eine einzelne Ursache zurückführen. Die Ergebnisse weisen jedoch darauf hin, dass diese Variante insgesamt am stärksten von der Variabilität der Trainingsdaten beeinflusst wird und die geringste Stabilität innerhalb der Überprüft-Modelle aufweist. Die Trainingsvariante in Abbildung 27 zeigt über große Teile der Iterationen ein weitgehendes stabiles Verhalten, weist jedoch gegen Ende einen leichten Anstieg der Spannweite zwischen Minimal- und Maximalwert auf. Eine naheliegende Erklärung für die geringere Variabilität im Vergleich zu Abbildung 29 ist die Nutzung vollständiger Dateien. Diese enthalten längere und vielfältigere Merkmalsverläufe und bilden dadurch ein repräsentativeres Bild des zugrunde liegenden Prozesses ab. Neue Muster innerhalb einer Aufzeichnung wirken sich somit weniger stark auf die Modellanpassung aus als in Varianten, in denen nur kurze Teilbereiche verarbeitet werden.

Zusammengefasst zeigen die Testergebnisse, dass die Überprüft-Modelle signifikante Leistungssteigerungen erzielen und die angestrebten Referenzwerte erreichen. Die Autonom-Modelle bleiben dagegen über alle Varianten hinweg deutlich unterhalb der definierten Referenzwerte und zeigen keine nennenswerten Lernfortschritte. Diese Ergebnisse verdeutlichen, dass für eine verlässliche Vorhersagequalität die Einbeziehung korrekt annotierter *Labels* entscheidend bleibt und dass autonome Trainingsphasen nur dann sinnvoll sind, wenn die eingehenden Daten keine bislang unbekannten Muster enthalten und die Struktur der Daten über die Iterationen hinweg ausreichend stabil bleiben.

8 Fazit

In dieser Arbeit wurde untersucht, in welchem Umfang ein iterativer, modellgestützter *Labeling*-Prozess den manuellen *Labeling*-Aufwand bei der Annotation multivariater Zeitreihen verringern kann und wie sich die Modellleistung verändert, wenn ein Modell ab bestimmten Schwellenwerten auf den Testdaten ohne weitere Korrektur der *Labels* mit verschiedenen Trainingsstrategien weitertrainiert wird, im Vergleich zu einem Modell, das ausschließlich mit korrekten *Labels* trainiert wird. Ausgangspunkt war ein Referenzprozess aus der Automobilindustrie, in dem ein Fachexperte neu anfallende Daten manuell annotieren muss, bevor diese zur Verbesserung eines bestehenden Modells genutzt werden können. Um den Aufwand zu verringern, wurden unterschiedliche *Labeling*-Strategien eingesetzt und in einem angepassten Evaluationsprozess untersucht.

Die regelbasierte Strategien, die Heuristiken wie den Mittelwert als Ersatz für fehlendes Fachwissen genutzt haben, können den manuellen *Labeling*-Aufwand reduzieren, erfordern jedoch eine sorgfältige Wahl der Regeln und können sich nicht an neue Muster anpassen, wodurch ihre Leistung je nach Daten variieren kann. Die modellgestützte Strategie liefert verlässliche Vorhersagen für Muster, die das Modell bereits gelernt hat, führt jedoch zu fehlerhaften Vorhersagen bei neuen Mustern in den Daten, was im Lernprozess zu inkorrekten *Labels* und damit zu einem erhöhten manuellen Korrekturaufwand führen kann.

Die Trainingsstrategien zeigten, dass Vorhersagen auf vollständigen Dateien mit anschließendem Training zu stabileren Ergebnissen führen als Iterationen, in denen nur Teilbereiche einer Datei vorhergesagt und nachtrainiert werden. Es wurde deutlich, dass der Verzicht auf korrekt annotierte *Labels* problematisch ist, da ein Modell, dass mit seine eigenen Vorhersagen weitertrainiert falsche Strukturen verstärken kann, obwohl die Testergebnisse zunächst stabil bleiben. Insgesamt ließ sich kein nachhaltiger Lernprozess bei Modellen feststellen, die überwiegend mit ihren eigenen, nicht überprüften Vorhersagen weitertrainiert wurden.

9 Ausblick

Die vorliegende Arbeit eröffnet mehrere Perspektiven für zukünftige Forschung und Weiterentwicklung. Ein zentraler Ansatzpunkt besteht in einer praxisnäheren Evaluierung des manuellen *Labeling*-Aufwands, da in dieser Arbeit stets das korrekte *Label* verwendet wurde und reale Bedingungen, in denen menschliche Diskussionen, unterschiedliche Interpretationen oder zeitliche Einschränkungen auftreten, dadurch nicht abgebildet wurden. In diesem Zusammenhang wäre es relevant zu untersuchen, inwieweit heuristische Regeln wie Durchschnitt, Maximum, Minimum oder Quantile tatsächlich das Fachwissen eines Annotierenden widerspiegeln, da diese in der vorliegenden Arbeit mangels verfügbarer Expertise lediglich als stellvertretende Entscheidungsregeln eingesetzt wurden.

Darüber hinaus ist eine Evaluierung des eingesetzten *Labeling*-Tools notwendig, um zu prüfen, wie gut annotierende Personen mit der grafischen Benutzeroberfläche zurechtkommen und welche funktionalen Verbesserungen den praktischen Einsatz zusätzlich unterstützen könnten. Eine solche Untersuchung würde nicht nur die Nutzerfreundlichkeit, sondern auch die Effizienz des gesamten *Labeling*-Prozesses weiter präzisieren.

Zudem bieten *Active Learning* Verfahren ein vielversprechendes Potenzial, indem sie gezielt informative Instanzen auswählen und so die Menge der manuell zu annotierenden Daten weiter reduzieren. Ergänzend könnten zukünftige Arbeiten Strategien entwickeln, die den Prozess auf Basis von Modellsicherheiten steuern und nicht auf der Vorhersagequalität auf den Testdaten, um damit eine adaptive Entscheidung darüber zu ermöglichen, ob Vorhersagen überprüft werden müssen, anstatt eine starre autonome Phase einzusetzen.

Literatur

[1] Data Mining: Concepts and Techniques 3rd Edition. *DATA MINING*.

[2] Charu C. Aggarwal. *Neural Networks and Deep Learning: A Textbook*. Springer International Publishing, Cham, 2018.

[3] Manuel Atug. Neuronale netze und faltung. https://www.informatik-aktuell.de/betrieb/kuenstliche-intelligenz/neuronale-netze-und-faltung.html, 2020. Accessed: 10 November 2025.

[4] Stefanos Bakirtzis, Kehai Qiu, Ian Wassell, Marco Fiore, and Jie Zhang. Deep-learning-based multivariate time-series classification for indoor/outdoor detection. *IEEE Internet of Things Journal*, 9(23):24529–24540, 2022.

[5] Johannes Bauer, Stephan Trattnig, Fabian Vieltorf, and Rüdiger Daub. Handling data drift in deep learning-based quality monitoring: evaluating calibration methods using the example of friction stir welding. *Journal of Intelligent Manufacturing*, pages 1–16, 01 2025.

[6] Avishan Bewtra and Christopher Ye. A Survey of Query Strategies for Active Learning.

[7] Yann Cherdo, Benoit Miramond, Alain Pegatoquet, and Alain Vallauri. Unsupervised Anomaly Detection for Cars CAN Sensors Time Series Using Small Recurrent and Convolutional Neural Networks. *Sensors*, 23(11):5013, May 2023.

[8] Alexis Cook. Handling missing values. https://www.kaggle.com/code/alexisbcook/missing-values, 2022. Accessed: 10 November 2025.

[9] DataCamp. Guide to data cleaning in python. https://www.datacamp.com/de/tutorial/guide-to-data-cleaning-in-python, 2024. Accessed: 10 November 2025.

[10] datasolut GmbH. Unsupervised learning. https://datasolut.com/wiki/unsupervised-learning/, 2024. Accessed: 10 November 2025.

[11] Henock M. Deberneh and Rovshan G. Sadygov. Software Tool for Visualization and Validation of Protein Turnover Rates Using Heavy Water Metabolic Labeling and LC-MS. *International Journal of Molecular Sciences*, 23(23):14620, November 2022.

[12] M. Egmont-Petersen, D. De Ridder, and H. Handels. Image processing with neural networks—a review. *Pattern Recognition*, 35(10):2279–2301, October 2002.

[13] Andrejs Fedjajevs, Willemijn Groenendaal, Carlos Agell, and Evelien Hermeling. Platform for analysis and labeling of medical time series. *Sensors*, 20(24), 2020.

[14] Eloi Figueiredo and Elizabeth Cross. Linear approaches to modeling nonlinearities in long-term monitoring of bridges. *Journal of Civil Structural Health Monitoring*, 3(3):187–194, August 2013.

[15] Chaitanya Gandhi, Pradeepta Kumar Sarangi, Merry Saxena, and Ashok Kumar Sahoo. SMS Spam Detection Using Deep Learning Techniques: A Comparative Analysis of DNN Vs LSTM Vs Bi-LSTM. In *2023 International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES)*, pages 189–194, Greater Noida, India, April 2023. IEEE.

[16] Numiqo GmbH. Cohen's kappa – erklärung, berechnung und beispiel. https://numiqo.de/tutorial/cohens-kappa, 2022. Zugriff am 12. Oktober 2025.

[17] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[18] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks, June 2014. arXiv:1406.2661 [stat].

[19] Agrim Gupta, Piotr Dollár, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation, 2019.

[20] Steve Hanneke. *Theoretical foundations of active learning*. PhD thesis, USA, 2009. AAI3362265.

[21] IBM. Was sind synthetische daten? https://www.ibm.com/de-de/think/topics/synthetic-data, 2024. Accessed: 10 November 2025.

[22] IBM. What are neural networks? https://www.ibm.com/think/topics/neural-networks, 2025. Accessed: 10 November 2025.

[23] IBM. What are recurrent neural networks (rnns)? https://www.ibm.com/think/topics/recurrent-neural-networks763338458, 2025. Accessed: 10 November 2025.

[24] Shyr-Long Jeng. Generative Adversarial Network for Synthesizing Multivariate Time-Series Data in Electric Vehicle Driving Scenarios. *Sensors*, 25(3):749, January 2025.

[25] Murad Khan and Yousef Hossni. A comparative analysis of LSTM models aided with attention and squeeze and excitation blocks for activity recognition. *Scientific Reports*, 15(1):3858, January 2025.

[26] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

[27] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything, 2023.

[28] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, Tom Duerig, and Vittorio Ferrari. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *International Journal of Computer Vision*, 128(7):1956–1981, March 2020.

[29] J. Richard Landis and Gary G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174, 1977.

[30] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization.

[31] Ana C. Lorena, Luis F.O. Jacintho, Marinez F. Siqueira, Renato De Giovanni, Lúcia G. Lohmann, André C.P.L.F. De Carvalho, and Missae Yamamoto. Comparing machine learning classifiers in potential distribution modelling. *Expert Systems with Applications*, 38(5):5268–5275, May 2011.

[32] Andrei Mihalea, Robert-Florian Samoilescu, and Adina Magda Florea. Self-Supervised Steering and Path Labeling for Autonomous Driving. *Sensors*, 23(20):8473, October 2023.

[33] Filip Moons and Ellen Vandervieren. Measuring agreement among several raters classifying subjects into one or more (hierarchical) categories: A generalization of Fleiss' kappa. *Behavior Research Methods*, 57(10):287, September 2025.

[34] Alhassan Mumuni, Fuseini Mumuni, and Nana Kobina Gerrar. A survey of synthetic data augmentation methods in machine vision. *Machine Intelligence Research*, 21(5):831–869, March 2024.

[35] Malte Ollenschläger, Arne Küderle, Wolfgang Mehringer, Ann-Kristin Seifer, Jürgen Winkler, Heiko Gaßner, Felix Kluge, and Bjoern M. Eskofier. MaD GUI: An Open-Source Python Package for Annotation and Analysis of Time-Series Data. *Sensors*, 22(15):5849, August 2022.

[36] David C. Plaut, Steven J. Nowlan, and Geoffrey E. Hinton. Experiments on learning by back propagation. 1986.

[37] Alexander Ratner, Christopher De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. Data programming: Creating large training sets, quickly, 2017.

[38] Ujjwal Sachdeva and P. Raghu Vamsi. A Study on Anomaly Detection with Deep Learning Models for IoT Time Series Sensor Data. In *2022 8th International*

Conference on Signal Processing and Communication (ICSC), pages 11–14, Noida, India, December 2022. IEEE.

[39] Scikit-learn Developers. Unsupervised dimensionality reduction. https://scikit-learn.org/stable/modules/unsupervised_reduction, 2024. Accessed: 10 November 2025.

[40] scikit-learn developers. scikit-learn: Machine learning in python. https://scikit-learn.org/stable/, 2025. Accessed: 2025-11-09.

[41] Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52, 07 2010.

[42] Shaip. Understanding the differences between manual and automatic data labeling. https://de.shaip.com/blog/understanding-the-differences-between-manual-automatic-data-labeling/, 2023. Accessed: 10 November 2025.

[43] Abhishek Singh. Understanding principal component analysis (pca) step-by-step. https://medium.com/analytics-vidhya/understanding-principle-component-analysis-pca-step-by-step-e7a4bb4031d9, 2020. Accessed: 10 November 2025.

[44] Paolo Trunfio. *Service-Oriented Distributed Knowledge Discovery*, volume 20121229 of *Chapman & Hall/CRC Data Mining and Knowledge Discovery Series*. Chapman and Hall/CRC, October 2012.

[45] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer New York, New York, NY, 2000.

[46] Cédric Walker, Tasneem Talawalla, Robert Toth, Akhil Ambekar, Kien Rea, Oswin Chamian, Fan Fan, Sabina Berezowska, Sven Rottenberg, Anant Madabhushi, Marie Maillard, Laura Barisoni, Hugo Mark Horlings, and Andrew Janowczyk. PatchSorter: a high throughput deep learning digital pathology tool for object labeling. *npj Digital Medicine*, 7(1):164, June 2024.

[47] Yixuan Wang, Jieqiong Zhao, Jiayi Hong, Ronald G. Askin, and Ross Maciejewski. A Simulation-Based Approach for Quantifying the Impact of Interactive Label Correction for Machine Learning. *IEEE Transactions on Visualization and Computer Graphics*, 31(9):5687–5703, September 2025.

[48] Dingming Wu, Xiaolong Wang, Jingyong Su, Buzhou Tang, and Shaocong Wu. A labeling method for financial time series prediction based on trends. *Entropy*, 22(10), 2020.

[49] Shuhei Yamamoto, Takeshi Kurashima, and Hiroyuki Toda. *Identifying Near-Miss Traffic Incidents in Event Recorder Data*, pages 717–728. 05 2020.

[50] Sungmin You, Baek Hwan Cho, Young-Min Shon, Dae-Won Seo, and In Young Kim. Semi-supervised automatic seizure detection using personalized anomaly detecting variational autoencoder with behind-the-ear EEG. *Computer Methods and Programs in Biomedicine*, 213:106542, January 2022.

[51] Manzil Zaheer, Sashank J. Reddi, Devendra Sachan, Satyen Kale, and Sanjiv Kumar. Adaptive methods for nonconvex optimization. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 9815–9825, Red Hook, NY, USA, 2018. Curran Associates Inc.