BERGISCHE UNIVERSITÄT WUPPERTAL

Fakultät für Maschinenbau und Sicherheitstechnik

Masterthesis

im Studiengang **Qualitätsingenieurwesen**

beim Fachgebiet für **Verkehrssicherheit und Zuverlässigkeit**

zur Erreichung des akademischen Grades

Master of Science

Thema:	**Zuverlässigkeitstechnik in der Industrie 4.0 unter Anwendung maschineller Lernalgorithmen zur Klassifikation von Fehlerzuständen**
Autor:	**Frederik Alexander Illtz**
MatNr:	1831121
Bearbeitungszeitraum:	**18.05.2023** bis **16.08.2023**
Betreuer*in:	Raphael Korbmacher, M. Sc.
Erstprüfer*in:	Jun.-Prof. Dr. Antoine Tordeux
Zweitprüfer*in:	Raphael Korbmacher, M. Sc.

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die von mir eingereichte Abschlussarbeit (Bachelor-/Master-Thesis) selbstständig verfasst und keine andere als die angegebene Quellen und Hilfsmittel benutzt sowie Stellen der Abschlussarbeit, die anderen Werken dem Wortlaut oder Sinn nach entnommen wurden, in jedem Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Ich bin damit einverstanden, dass die Arbeit durch Dritte eingesehen und unter Wahrung urheberrechtlicher Grundsätze zitiert werden darf.

Ort und Datum: _____

Unterschrift: _____

Inhaltsverzeichnis

Abkürzungsverzeichnis

CPS Cyber-physische Systeme

EOL End of Life

IOT Internet of Things

KNN K-Nearest-Neighbor

PHM Prognotics and Health Management

RUL Remaining Useful Life

SVM Support Vector Machine

Abbildungsverzeichnis

Tabellenverzeichnis

Zusammenfassung/Abstract

Zusammenfassung Aufgrund der voranschreitende Digitalisierung im Zusammenhang mit der Industrie 4.0 ergeben sich auch für die Instandhaltung neue Möglichkeiten. Durch die Auswertung von historischen oder Echtzeitdaten kann der Zustand einer Maschine durch maschinelle Lernalgorithmen bestimmt und eine Wartung bedarfsgerecht zu einem Zeitpunkt durchgeführt werden, in dem die Maschine nicht verwendet wird. Dadurch kann es gegenüber den bisherigen Instandhaltungsformen zu einer Kostenreduzierung kommen.

Das Ziel dieser Masterarbeit ist die Klassifizierung von Fehlerzuständen mithilfe von maschinellen Lernalgorithmen. Dazu werden die Algorithmen: K-Nearest-Neighbor, logistische Regression, Entscheidungsbaum, Random Forest, Support Vector Machine und ein neuronales Netz dahingehend optimiert, anhand der Vibrationsdaten einer Maschine vier Zustände richtig zu klassifizieren. Die Optimierung erfolgt zum einen durch die Anpassung der Parameter der einzelnen Lernalgorithmen und zum anderen durch die Auswahl der besten Anzahl von Merkmalen sowie der Merkmale selbst.

Um mithilfe der maschinellen Lernalgorithmen Modelle für die Klassifizierung der Zustände zu erstellen, wurde zunächst eine exploriative Datenanalyse, in der die Datensätze und Auffälligkeiten beschrieben werden, durchgeführt. Anschließend werden die Daten vor verarbeitet, um diese zu bereinigen und Merkmale aus den Daten abzuleiten. Darauf folgend werden die Algorithmen angepasst, um die optimalen Parameter zu ermitteln, bevor die Klassifikationsergebnisse in der Ergebnisdarstellung beschrieben werden.

Durch die Anpassung der Algorithmen konnten durch den Random Forest und die Support Vector Machine Genauigkeiten bei der Klassifizierung der Testdaten von bis zu 98,31 % erzielen werden. Auch die anderen Algorithmen erreichten mit 97,19 % beziehungsweise 97,89 % bei dem neuronalen Netz gute Klassifikationsergebnisse.

Abstract Advancing digitalization in connection with Industry 4.0 is also opening up new opportunities for maintenance. By evaluating historical or real-time data, the condition of a machine can be determined by machine learning algorithms and maintenance can be carried out as needed at a time when the machine is not being used. This can lead to a reduction in costs compared to previous forms of maintenance.

The goal of this master thesis is the classification of fault conditions with the help of machine learning algorithms. For this purpose the algorithms: K-Nearest-Neighbor, logistic regression, decision tree, random forest, support vector machine and a neural network are optimized to correctly classify four states based on the vibration data of a machine. The optimization was done firstly by adjusting the parameters of each learning algorithm and secondly by selecting the best number of features as well as the features themselves.

In order to use machine learning algorithms to build models for classifying the states, exploratory data analysis describing the datasets and anomalies was first performed. Then, the data are pre-processed to clean them and features are derived from the data. Following this, the algorithms are adapted to determine the optimal parameters before the classification results are described in the results presentation.

By adapting the algorithms, the Random Forest and Support Vector Machine were able to achieve accuracies of up to 98.31 % in classifying the test data. The other algorithms also achieved good classification results with 97.19 % and 97.89 % for the neural network.

1. Einleitung

In diesem Abschnitt wird das Thema dieser Masterthesis beschrieben und die Relevanz des Themas herausgestellt. Anschließend werden die Zielsetzung und die verwendeten Methoden dargelegt, bevor der Aufbau dieser Thesis erläutert wird.

1.1. Beschreibung des Themas

Seitdem der Begriff ”Industrie 4.0” erstmals im Jahr 2011 auf der Hannover Messe vorgestellt und damit die vierte Industrielle Revolution angekündigt wurde, sind die Investitionen, wie in Abbildung 1.1 dargestellt ist, in diesem Bereich in den letzten Jahren stetig gestiegen [15, 1]. Hinter dem Begriff Industrie 4.0 verbergen sich eine Vielzahl von

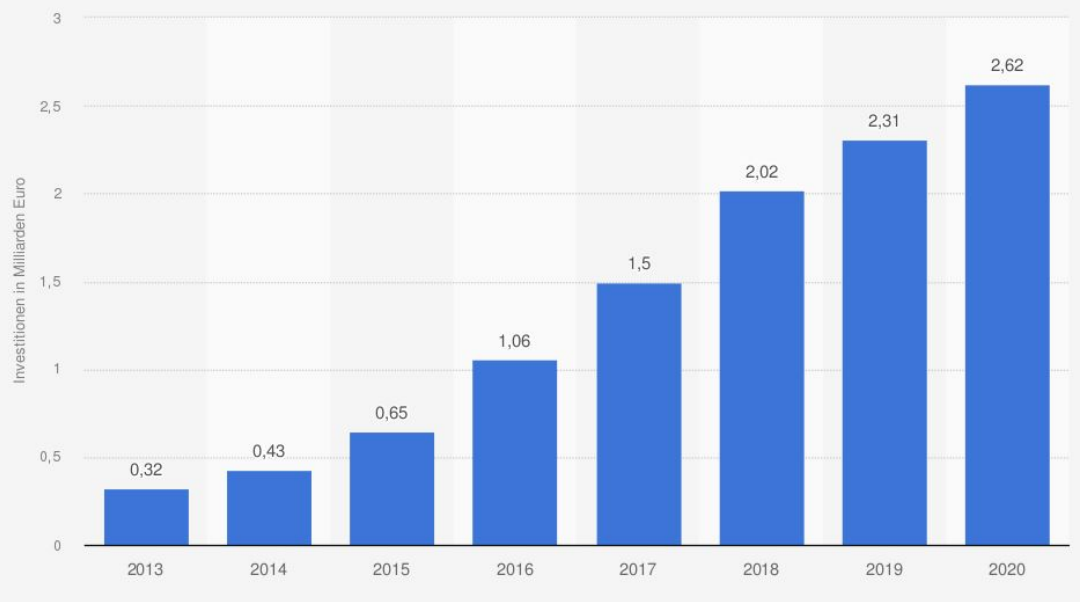

Abbildung 1.1.: Investitionen in die Industrie 4.0 in Deutschland in den Jahren 2013 bis 2020 [1]

Anwendungsmöglichkeiten, doch zählt die vorausschauende Instandhaltung zu einer der greifbarsten Anwendungen in der Industrie 4.0 [15, 16]. Bereits im Jahr 2019 gaben 37 % von 323 befragten Unternehmen aus dem Maschinen- und Anlagenbau sowie der Elektro- und Automobilindustrie an, die Fälligkeit von Wartungsarbeiten automatisiert zu verfolgen und anzeigen zu lassen. Weitere Unternehmen verwenden vorausschauende Wartungsanwendungen beispielsweise dazu, Verschleiß rechtzeitig zu erkennen oder die

Qualität der Erzeugnisse zu verbessern. Lediglich 38 % der befragten Unternehmen nutzen keine Anwendungen der vorausschauenden Wartung [17].

1.2. Zielsetzung und Methoden der Masterthesis

Das Ziel dieser Masterthesis ist die korrekte Klassifizierung von Fehlerzuständen eines Beispieldatensatzes mittels maschineller Lernalgorithmen. Dazu wird der Datensatz vorbereitet und anschließend werden Modelle anhand verschiedener maschineller Lernalgorithmen erstellt, deren Parameter dahingehend optimiert werden, dass die Modelle Fehlerzustände mit einer hohen Genauigkeit und Genrealisierbarkeit klassifizieren.

1.3. Aufbau der Masterthesis

Die Masterthesis gliedert sich in sieben Teile, beginnend mit der Einleitung, in der die Relevanz des Themas sowie die Zielsetzung dieser Thesis herausgestellt wird.

Im zweitem Teil dieser Arbeit werden die Grundlagen der Themen Industrie 4.0, Zuverlässigkeitstechnik in der Industrie 4.0 sowie des maschinellen Lernens beschrieben. Dabei werden unter anderem die verschiedenen Instandhaltungsstrategien und die Möglichkeiten, diese mit den Entwicklungen der Industrie 4.0 zu verknüpfen, erläutert. Des Weiteren werden in diesem Teil die maschinellen Lernalgorithmen vorgestellt, welche im vierten Teil angewendet werden.

Der dritte Teil dieser Arbeit beschäftigt sich mit der grundsätzlichen Vorgehensweise bei der maschinellen Datenanalyse und erläutert die vier Schritte: ”Datenerfassung”, ”Datenvorverarbeitung”, ”Datenanalyse” und ”Ergebnisdarstellung und -interpretation”. Dabei werden die verschiedenen Methoden, die in diesen Schritten zur Anwendung kommen können, erläutert.

Auf den Teilen zwei und drei aufbauend wird im vierten Teil eine Datenanalyse anhand eines Beispieldatensatzes einer Standbohrmaschine vorgenommen, mit dem Ziel, die Fehlerzustände mit den verschiedenen maschinellen Lernalgorithmen korrekt zu klassifizieren.

Anschließend werden im fünften Teil die Ergebnisse der Klassifikation ausgewertet.

Im sechsten Teil folgt eine Diskussion der Ergebnisse, in der diese kritisch bewertet und mögliche Verbesserungspotentiale herausgestellt werden.

Abschließen wird im siebten Teil dieser Thesis ein Fazit gezogen, in dem die Ergebnisse zusammengefasst werden.

2. Theoretische Grundlagen

In diesem Kapitel werden die theoretischen Grundlagen erläutert. Dazu wird zunächst die Industrie 4.0 mit ihren Neuerungen und Potentialen beschrieben. Anschließend werden die in der Masterthesis verwendeten Methoden der prädiktiven Instandhaltung beschrieben. Abschließend wird erläutert, wie ein Modell zur prädiktiven Instandhaltung grundsätzlich aufgebaut werden kann.

2.1. Industrie 4.0

Der Begriff Industrie 4.0 beschreibt die vierte industrielle Revolution. Diese zeichnet sich durch die Informatisierung der Produktion aus [18]. Dadurch sollen Informationen aus verschiedenen Quellen dazu verwendet werden, Entscheidungen zu treffen. Die Informatisierung der Produktion erfolgt über sogenannte CPS, welche aus den drei Ebenen:

- Physikalische Objekte,
- Datenspeicher und
- Dienstsystemen

bestehen [15].

Hinter dem Begriff Industrie 4.0 steht die digitale Agenda der Bundesregierung, welche einen Technologiesprung hin zur Informatisierung der Produktion erzielen soll. Damit folgt die Industrie 4.0 auf die drei industriellen Revolutionen:

1. Mechanisierung,
2. Massenproduktion und
3. Einsatz von Elektronik und IT,

welche in Abbildung 2.1 dargestellt sind [15].

Das Ziel der Industrie 4.0 ist, die Qualität und Wettbewerbsfähigkeit im internationalen Vergleich zu verbessern [19]. Dazu müssen nach Manzei et al. digitale Informationsquellen intern und extern dynamisch miteinander verknüpft, die daraus generierten Daten automatisch analysiert und weitergegeben werden, um die Vorbereitung und Steuerung von Prozessen innerhalb der Wertschöpfungskette bedarfsgerecht zu gestalten. Damit

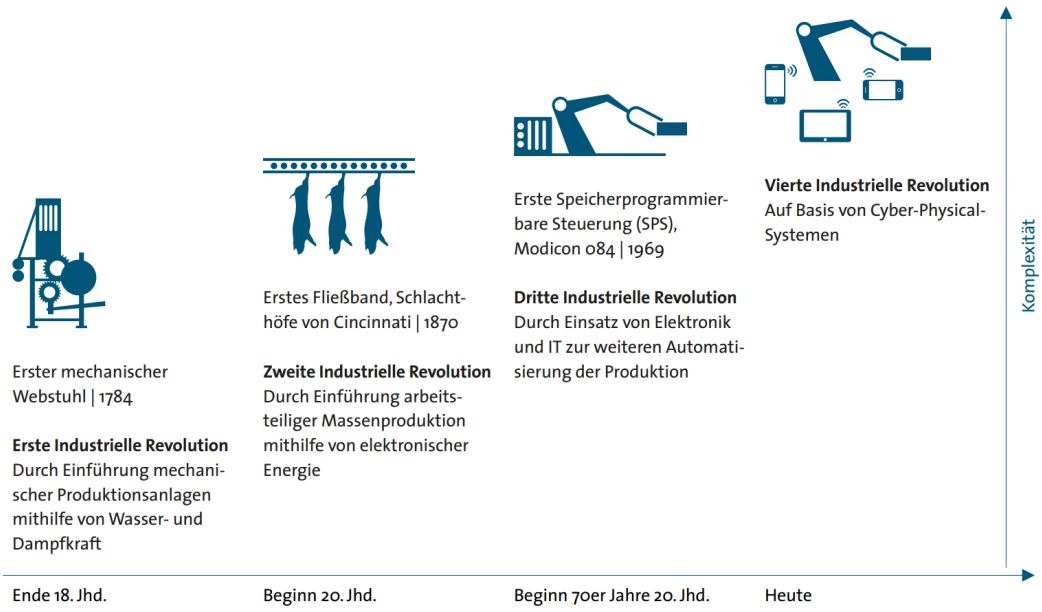

Abbildung 2.1.: Die vier Stufen industrieller Revolutionen [2, S. 10]

sollen Wertschöpfungsketten schneller, kostengünstiger, effizienter, ressourcenschonender, flexibler und kundenorientierter werden [15]. Die Verknüpfung von Informationsquellen zur Steuerung der Prozesse durch physische Objekte erfolgt durch den Einsatz eines CPSs. Ein CPS zeichnet sich durch die folgenden Eigenschaften aus, wobei nicht jede Eigenschaft in jedem CPS vorhanden sein muss:

- In einem CPS sind Sensoren zur Aufnahme von Informationen und Aktuatoren, welche eine Interaktion mit der Umwelt ermöglichen sowie eine Verbindung zum Internet eingebettet. Dadurch kann eine Verbindung zwischen der physischen und virtuellen Ebene geschaffen werden.
- Ein CPS kann langfristig lernen und damit autonom agieren, wodurch ein CPS in der gleichen Situation ein unterschiedliches Verhalten aufweisen kann.
- Durch die Vernetzung mit anderen CPS ist es möglich, ein komplexes und kooperatives Verhalten zu entwickeln.
- Neben der Kooperation mit anderen CPS besteht die Möglichkeit, dass ein CPS mit Menschen kooperiert, indem beide beispielsweise über Text-, Spracheingaben oder weitere Methoden miteinander kommunizieren. Das Ziel ist eine Kooperation, durch die komplexe Aufgaben arbeitsteilig gelöst werden.
- Ein CPS kann aus mehreren unterschiedlichen Untersystemen bestehen, die unabhängig voneinander entwickelt wurden [15].

Um diese Eigenschaften in einem System zu vereinen, werden mehrere Schnittstellen und Schichten innerhalb eines CPS benötigt, welche in Abbildung 2.2 abgebildet sind.

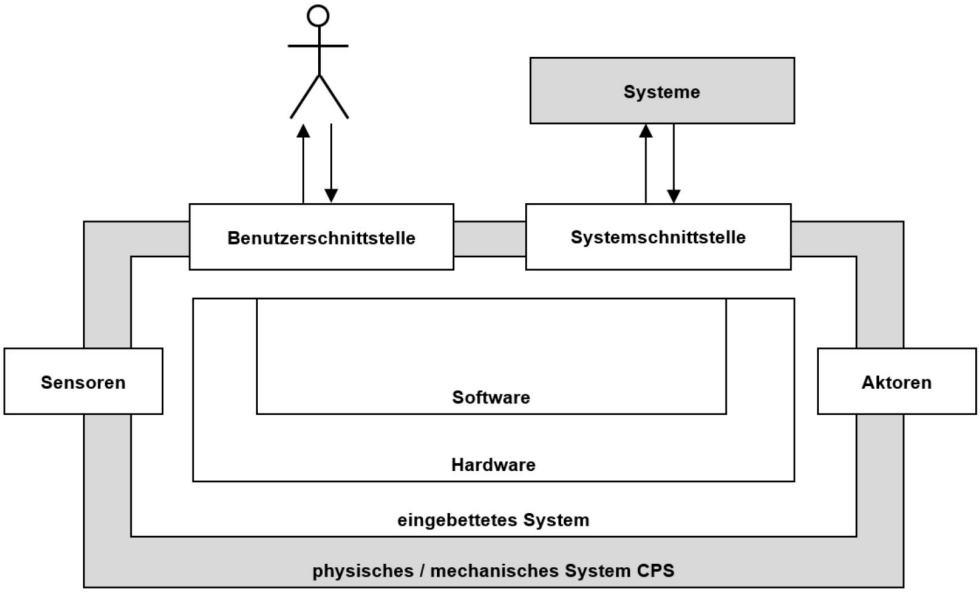

Abbildung 2.2.: Komponenten eines CPS [3, S. 39]

Um die Eigenschaften eines CPSs in den bisher vorrangig passiv eingesetzten Systemen zu ermöglichen, müssen diese um ein eingebettetes System erweitert werden. Dazu zählt ein Mikrocontroller, welcher durch eine Software die empfangenen Daten analysiert, den Status des Objektes überwacht, Entscheidungen trifft und diese ausführt. Des Weiteren werden Sensoren benötigt, durch die das eingebettete System Daten über das Umfeld erhält. Durch Aktuatoren können die zuvor getroffenen Entscheidungen umgesetzt sowie visuelle und akustische Informationen an Personen übermitteln werden [2].

Die Vernetzung der CPS erfolgt über das Internet of Things (IOT). Das IOT benötigt eine standardisierte Kommunikation, um Daten aus verschiedenen Produktions- und IT-Systemen zusammenfließen zu lassen und eine Kooperation innerhalb der Wertschöpfungskette zu ermöglichen [3].

Eine weitere Komponente, welche zur Bewältigung der Komplexität dient, ist Big Data. Diese kann durch die Identifikation von Beziehungen, Mustern und Trends innerhalb der Daten optimale Entscheidungen für die Synchronisation der Wertschöpfungsprozesse treffen. Die Daten können beispielsweise aus dem Produktionsprozess, der Materialbewegung oder den Produktionsressourcen stammen [3].

„Big Data bezeichnet Daten, die in so großer Menge, Vielfalt, Komplexität oder Geschwindigkeit anfallen, dass diese nicht mit konventionellen Methoden der Datenverarbeitung verarbeitet und ausgewertet werden können.“ [3, S. 48]

Big Data kann innerhalb der Industrie 4.0 für die Überwachung der Abläufe und Ergebnisse in der Wertschöpfungskette, für die Vorhersage von Ereignissen und Zuständen in der Produktion und zur Bewertung von verschiedenen Handlungsmöglichkeiten für die Planung und Steuerung verwendet werden [3]. Abbildung 2.3 verdeutlicht den Zusam-

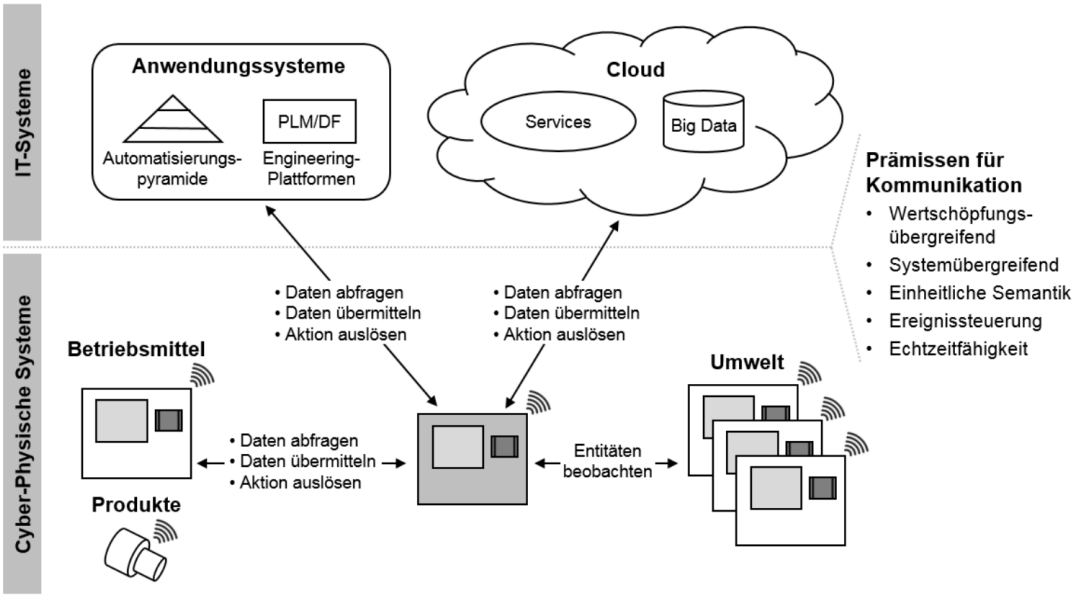

Abbildung 2.3.: Interaktion und Kommunikation in einen Industrie-4.0-Netzwerk [4, S. 86]

menhang zwischen den verschiedenen CPS sowie die Verbindung zum IOT, worüber Daten abgefragt, übermittelt und Aktionen ausgelöst werden.

2.2. Zuverlässigkeitstechnik in der Industrie 4.0

Durch den vermehrten Einsatz von Sensoren im Rahmen von CPS und neuen Technologien wie dem IOT in der Industrie 4.0 besteht die Möglichkeit, große Datenmengen während des Betriebs von Maschinen zu sammeln [20, 5]. Die Technologien der Industrie 4.0 ermöglichen darüber hinaus die Verbindung zwischen den Maschinen und einem Computernetzwerk. Darin können die Daten einem intelligenten Datenmanagement zugeführt und analysiert werden. Durch das kontinuierliche Überwachen und Sammeln von Maschinendaten wird eine Zustandsüberwachung ermöglicht, die in einer Echtzeit-

diagnose Aufschluss über den Maschinenzustand gibt und einen möglichen Ausfall der Maschine erkennen kann. Dadurch wird die prädiktive Instandhaltung ermöglicht [20].

Die Methoden der Zustandsüberwachung und prädiktiven Instandhaltung werden im Prognotics and Health Management (PHM)-Ansatz zusammengeführt. Während der klassische Zuverlässigkeitsbegriff als ”Wahrscheinlichkeit dafür, daß eine Einheit während einer definierten Zeitdauer unter angegebenen Funktions- und Umgebungsbedingungen nicht ausfällt” [21, S. 2] definiert ist, umfasst im Gegensatz dazu das PHM die Zustandsschätzung, die Prognose zukünftig auftretender Ereignisse, die Prognose der Lebensdauer sowie die Optimierung des Systems [5].

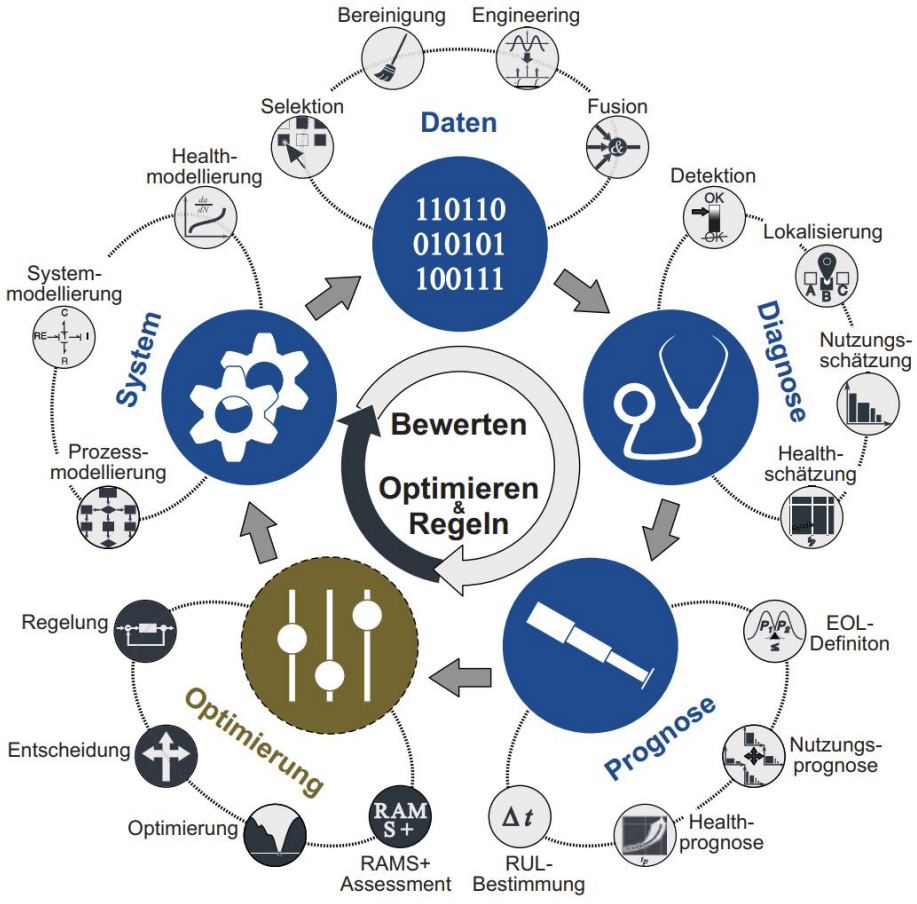

Abbildung 2.4.: Bestandteile eines PHM-Systems [5, S. 498]

Das in Abbildung 2.4 dargestellte PHM-System umfasst fünf Teilbereiche. Bei dem ersten Teilbereich System wird das betrachtete System mittels eines physikalischen oder mathematischen Modells abgebildet, um das vorliegende System und dessen Schädigungsmechanismen zu verstehen und den momentanen Zustand des Systems abzu-

schätzen. Unter dem Bereich Daten werden die Tätigkeiten zusammengefasst, bei denen die relevanten Daten der Sensoren zum Aufbau eines Modells gesammelt, bereinigt und transformiert werden. Im zweiten Schritt werden die Daten zur Diagnose des momentanen Gesundheitszustandes verwendet und ein Healthmodell aufgebaut. Bei der Prognose werden die Daten dazu verwendet, um Kennzahlen wie Health, Remaining Useful Life (RUL) oder End of Life (EOL) zu ermitteln. Health stellt den aktuellen Gesundheitszustand eines Systems dar. Die RUL gibt die Restlebensdauer eines Systems an, indem die verbleibende Zeit bis zu einem Ausfall bei den gegebenen Bedingungen prognostiziert wird. Die Kennzahl EOL beschreibt einen Schwellenwert, bei dem das Lebensdauerende einer Maschine erreicht wird. Dabei kann es sich um einen Ausfall oder den Verlust der Funktionalität eines Systems handeln. Der Bereich der Optimierung dient der Erhöhung der Verfügbarkeit durch die Anpassung der Instandhaltungsstrategie sowie einen Eingriff in das Betriebsverhalten eines Systems [5].

2.2.1. Instandhaltungsstrategien

Das Ziel einer Instandhaltung ist, mit geringen Kosten eine hohe Anlagenverfügbarkeit unter Berücksichtigung gesetzlicher, sicherheitstechnischer, technischer, produktionsrelevanter und wirtschaftlicher Gesichtspunkte zu erreichen. Im Rahmen der Instandhaltungsstrategie sollen unterschiedliche Instandhaltungsziele erreicht werden, wodurch sich unterschiedliche Ansätze ergeben, die sich in drei Kategorien einteilen lassen [22]:

- Korrektive Instandhaltung: Bei der korrektiven Instandhaltung werden Instandhaltungsmaßnahmen erst durchgeführt, nachdem der Fehler aufgetreten ist. Dieser Ansatz ist einfach umzusetzen, ist aber durch die ungeplanten Ausfallzeiten und die hohen Kosten für die Reparaturen in der Regel mit deutlich höheren Kosten verbunden als andere Instandhaltungsstrategien [23].
- Präventive Instandhaltung: Bei diesem Ansatz werden Instandhaltungsmaßnahmen nach einer festgelegten Zeit oder einer bestimmten Anzahl von Prozesswiederholungen durchgeführt. Dadurch werden Ausfälle verhindert, jedoch häufig Reparaturen durchgeführt, welche gegebenenfalls nicht nötig gewesen wären, was zu einer ineffizienten Nutzung von Ressourcen und höheren Betriebskosten führt [23].
- Prädiktive Instandhaltung: Dieser Instandhaltungsansatz bietet die Möglichkeit, den Zustand eines Systems zu schätzen und Ausfälle frühzeitig zu erkennen. Dadurch

können bedarfsgerecht Maßnahmen ergriffen werden, um einen Ausfall zu verhindern [23].

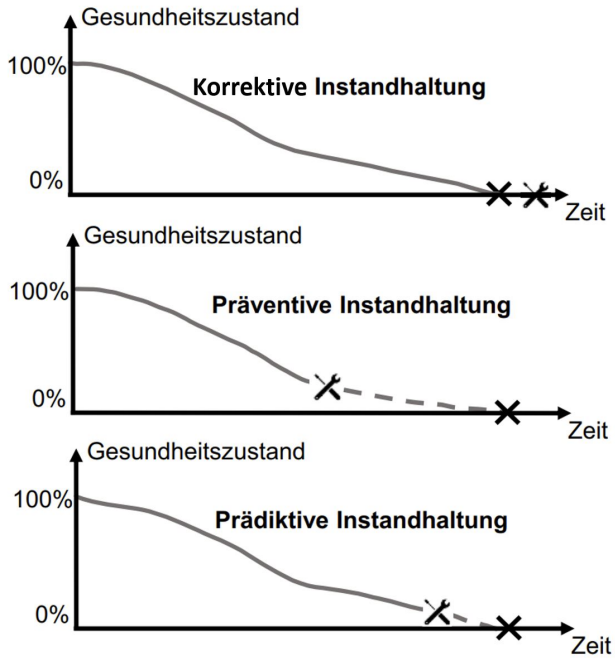

Abbildung 2.5.: In Anlehnung: Zustandsverlauf von Komponenten bei unterschiedlichen Instandhaltungsstrategien [6, S. 442]

Die Industrie 4.0 bietet durch den Einsatz von CPS eine große Datenmenge, welche, wenn sie vernetzt und zentral abrufbar ist, die Möglichkeit bietet, diese mit maschinellen Lernverfahren zu analysieren. Damit besteht die Möglichkeit, beispielsweise den aktuellen Betriebszustand zu ermitteln, woraus Erkenntnisse zu möglichen Wartungen erlangt werden können [9].

2.2.2. Zustandsüberwachung

Die Ziele der Zustandsüberwachung reichen von dem Schutz vor fatalen Schäden oder unerwarteten Maschinenausfällen, der Sicherung der Produktion, der Prozessoptimierung und Lebensdauererhöhung bis hin zur Qualitätskontrolle [7]. Dazu werden durchgehend Messdaten gesammelt und ausgewertet. Außerdem ermöglicht das Sammeln von Daten eine Diagnose über große Distanzen. Sollte sich der Zustand einer Maschine oder Anlage verändern und problematisch werden, können die Informationen der Zustandsüberwachung für die prädiktive Instandhaltung genutzt werden [9].

Die Zustandsüberwachung einer Maschine lässt sich durch verschiedene Methoden

realisieren. Eine der wichtigsten Methoden ist die Schwingungsüberwachung. Da der Betrieb von Maschinen und Anlagen mit dem Auftreten von Schwingungen verbunden ist, wird die Schwingungsüberwachung angewendet. Voraussetzung ist die Möglichkeit, diese mit elektrischen Methoden wie einem Schwingungswächter zu messen, angewendet. Schwingungen können bei der Messung in gute Schwingungen eingeteilt werden, die auch im besten Zustand auftreten und teilweise gewollt sind. Wird die Schwingung von drehzahlvariablen Maschinen gemessen, können bei unterschiedlichen Drehzahlen verschiedene Schwingungsamplituden auftreten. Dabei sind die Gutzustände für den jeweiligen Anwendungsfall zu spezifizieren. Schon bei kleinen Fehlern oder Impferektionen treten zusätzliche Quellen von Schwingungen auf, anhand derer der mechanische Zustand einer Maschine ermittelt werden kann. Abbildung 2.6 verdeutlicht den typischen Verlauf vom Auftreten einer Störung bis hin zum Ausfall, sollte der Schaden nicht rechtzeitig entdeckt werden [7].

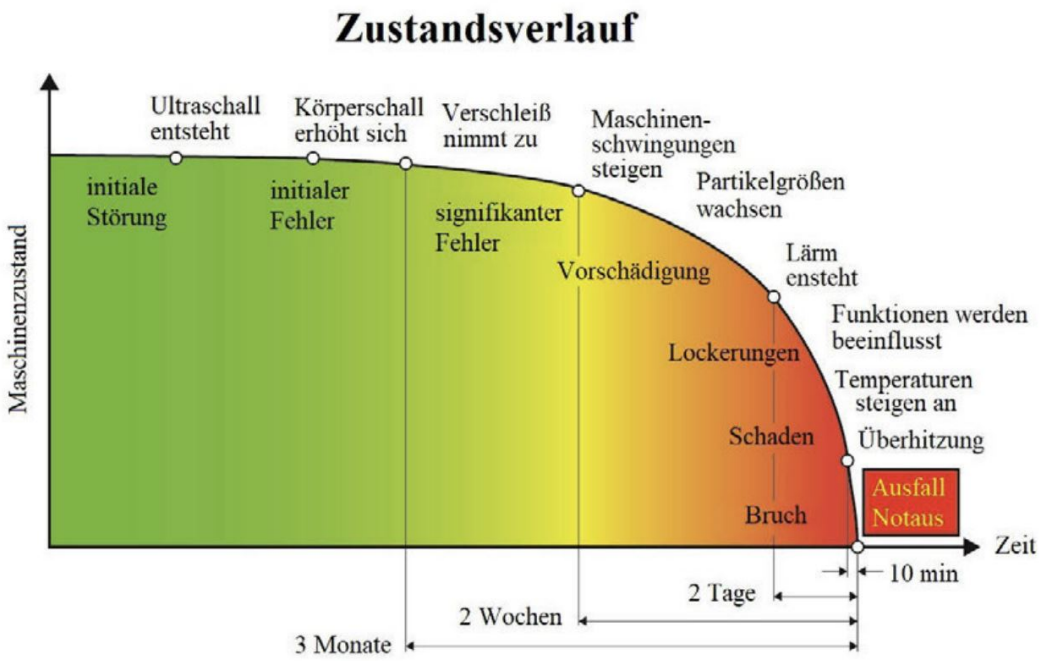

Abbildung 2.6.: Typischer Verlauf der Maschinenhistorie bis zum Schaden [7, S. 28]

Weil Schwingungen bereits im ordnungsgemäßen Zustand einer Maschine auftreten, können bereits kleinste Veränderungen ein Indikator für einen sich anbahnenden Fehler sein und ein Ausfall allein durch die Schwingungsüberwachung verhindert werden [7].

2.2.3. Prädiktive Instandhaltung

Bei der prädiktiven Instandhaltung werden die Daten der Zustandsüberwachung genutzt, um den Zustand der Maschine dauerhaft in einem Optimalzustand zu halten [7]. Dies geschieht, indem durch den Einsatz von maschinellen Lernalgorithmen Muster identifiziert werden, die es ermöglichen, Ausfälle vorherzusagen [24]. Im Gegensatz zur korrektiven und präventiven Instandhaltung erfolgt bei der präventiven Instandhaltung eine Instandhaltungsmaßnahme aufgrund des festgestellten Maschinenzustandes, anhand der Prognose der RUL oder der Diagnose von sich anbahnenden Fehlerzuständen [20, 9, 25]. Dadurch können Instandhaltungsmaßnahmen in einen Zeitraum gelegt werden, in dem keine Produktion stattfindet und ein Produktionsausfall vermieden werden [9]. Damit die prädiktive Instandhaltung effektiv umgesetzt werden kann, müssen die Fehler rechtzeitig identifiziert werden, um eine Wartung planen und umsetzen zu können. Wird ein Fehler zu früh oder falsch angezeigt, führt das dazu, dass eine funktionsfähige Komponente getauscht wird [20].

2.3. Maschinelles Lernen

Beim maschinellen Lernen geht es darum, automatisch Muster in einem Datensatz zu erkennen und anhand der identifizierten Muster Regeln abzuleiten, welche anschließend für unbekannte Daten genutzt werden [9].

Zur Erkennung der Muster können zwei grundsätzliche Verfahren verwendet werden. Das erste Verfahren ist das überwachte Lernen, bei dem die korrekten Ausgabewerte während des Lernprozesses bekannt sind. Im zweiten Verfahren, dem unüberwachten Lernen, sind die korrekten Ausgabewerte nicht bekannt [9].

Im Rahmen der prädiktiven Datenanalyse werden überwachte Lernverfahren verwendet. Bei der Verwendung von überwachten Lernmodellen wird das Vorgehen, wie in Abbildung 2.7 dargestellt ist, in zwei Schritte unterteilt. Im ersten Schritt wird ein Vorhersagemodell aus den Beziehungen zwischen den Merkmalen und den Ausgabewerten anhand von historischen Daten erstellt. Diese historischen Daten werden als Trainingsdaten bezeichnet. Im zweiten Schritt wird das zuvor erstellte Vorhersagemodell dazu verwendet, Vorhersagen für neue Daten zu treffen [8].

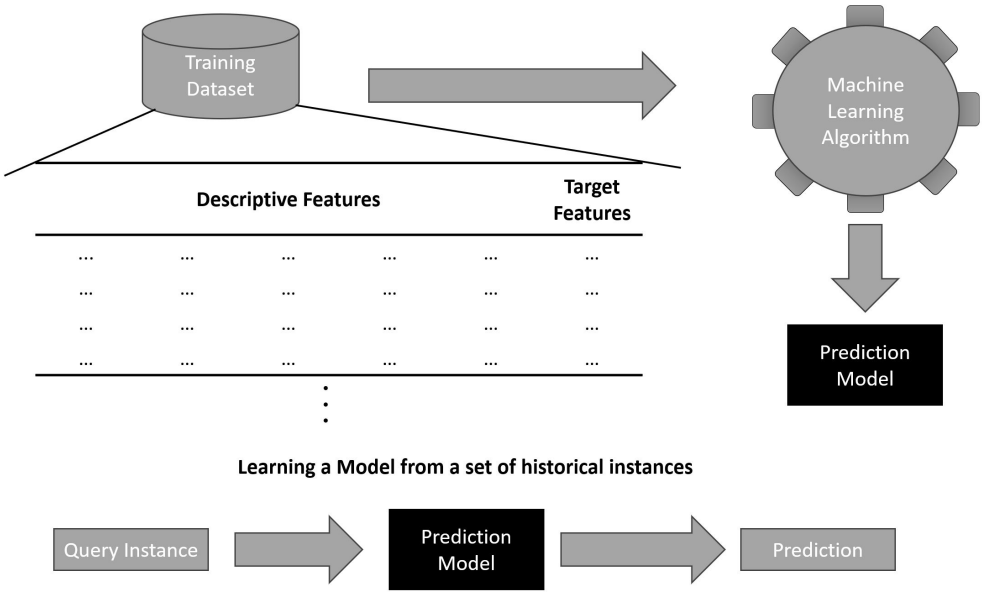

Abbildung 2.7.: Die zwei Schritte des überwachten maschinellen Lernens [8, S. 3]

Ein überwachtes Lernverfahren unterscheidet sich dahingehend, ob es sich bei den Ausgabewerten um diskrete oder kontinuierliche Ausgabewerte handelt. Sind die Ausgabewerte diskret wird, wie in Abbildung 2.8 dargestellt, eine Klassifikation durchgeführt. Das Ziel einer Klassifikation ist die Erkennung von Mustern in den Eingabedaten, um unbekannte Eingabedaten der richtigen Klasse zuzuordnen [9].

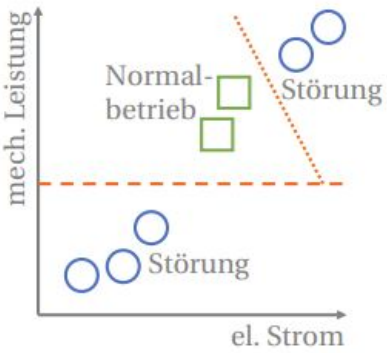

Abbildung 2.8.: Grafische Darstellung einer Klassifikation [9, S. 12]

Liegen kontinuierliche Ausgabedaten vor, wird eine Regression verwendet. Dabei soll innerhalb der Eingabedaten ein Muster identifiziert werden, durch das anschließend die korrekten Ausgabewerte bei unbekannten Eingabedaten ausgegeben werden können. Dazu kann, wie in Abbildung 2.9 zu sehen ist, eine Regressionskurve erstellt werden, durch die Ausgabewerte bei bislang unbekannten Eingabewerten bestimmt werden kön-

nen [9].

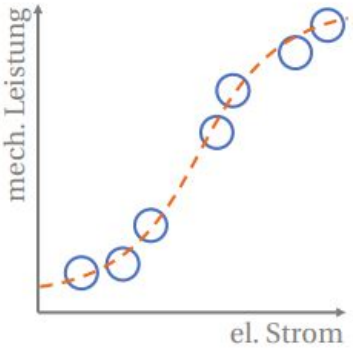

Abbildung 2.9.: Grafische Darstellung einer Regression [9, S. 12]

Das Ziel eines unüberwachten Lernverfahrens ist die Gruppierung von ähnlichen Datenpunkten, wie in Abbildung 2.10 dargestellt ist. Durch die Gruppierung der Datenpunkte können große Datensätze vorverarbeitet werden, um beispielsweise in einem zweiten Schritt eine Klassifikation durchzuführen [9].

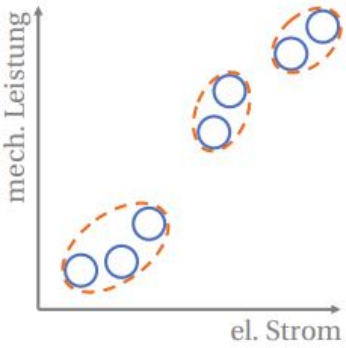

Abbildung 2.10.: Grafische Darstellung Clustering [9, S. 12]

2.3.1. K-Nearest-Neighbor

Bei dem KNN handelt es sich um ein Black-Box-Verfahren, da innerhalb der Daten kein Muster erkannt wird, anhand dessen eine Prognose für neue Daten erstellt wird. Dieses Verfahren speichert zunächst die Trainingsdaten. Sollen neue Eingabedaten klassifiziert werden, wird die Klasse des nächstgelegenen Nachbarn bestimmt. Die nächsten Nachbarn werden anhand der Distanz der Features von den Eingabedaten zu den Trainingsdaten bestimmt [26]. Die Distanz zwischen den Eingabedaten und den bereits vorhandenen

Trainingsdaten kann durch den euklidischen Abstand:

$$d(x, y) = ||x - y|| = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{1}$$

bestimmt werden [11].

Anhand der Klassen der Trainingsdaten wird durch eine Mehrheitsentscheidung bestimmt, welcher Klasse der neue Eingabewert zugeordnet wird. Die Anzahl der Nachbarn, welche für die Mehrheitsentscheidung berücksichtigt wird, wird durch den Parameter k festgelegt. Sollte keine Mehrheitsentscheidung möglich sein, da eine Gleichheit vorliegt, wird die Klasse entweder durch eine Zufallsentscheidung bestimmt oder keine Klasse zugewiesen [26]. Abbildung 2.11 veranschaulicht eine Mehrheitsentscheidung für eine Klassifizierung mit drei Klassen und einem k von 5.

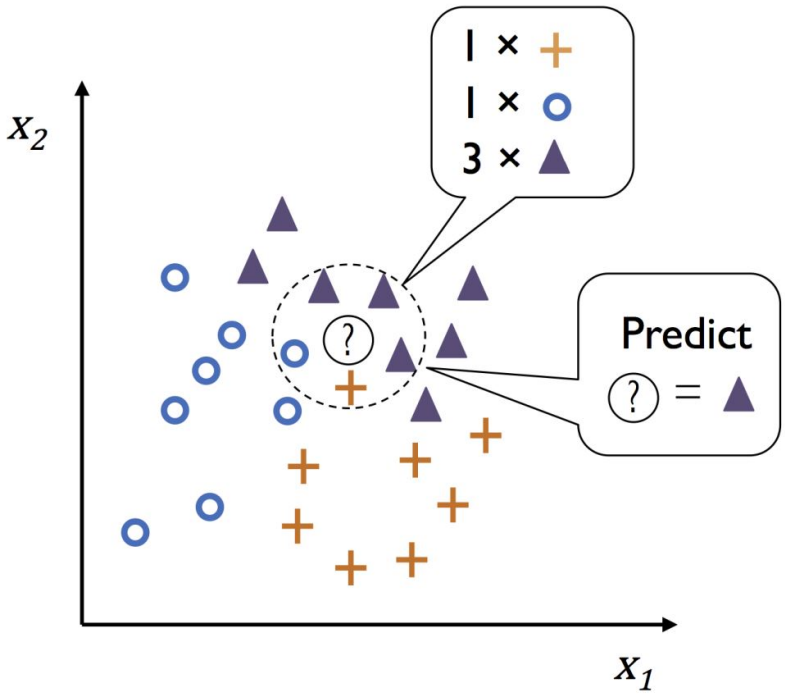

Abbildung 2.11.: Illustration von KNN für ein 3-Klassen-Problem mit k=5. [10, S. 7]

2.3.2. Logistische Regression

Die logistische Regression dient der Klassifizierung von Daten hinsichtlich der binären Zielgröße $Y \in 0, 1$. Bei der logistischen Regression wird zunächst eine gewichtete Sum-

me der Eingabemerkmale mit:

$$p = h_\theta(\mathbf{x}) = \sigma(\mathbf{x^T}\theta) = \sigma(\mathbf{x_1}\theta_\mathbf{1} + ... + \mathbf{x_n}\theta_\mathbf{n}) \tag{2}$$

berechnet. Dabei ist x ein Eingabewert und θ ein Gewicht. Anschließend wird das Ergebnis mittels der logistischen Funktion

$$\sigma(t) = \frac{1}{1 + e^{-t}} \tag{3}$$

berechnet. Die logistische Funktion ist eine Sigmoid-Funktion, die das Ergebnis der Eingabesumme in Zahlen zwischen 0 und 1 transformiert. Mit der transformierten Wahrscheinlichkeit von p kann das logistische Regressionsmodell eine Vorhersage bezüglich der Klassifizierung treffen:

$$y = \begin{cases} 0 \quad \text{bei} \quad p < 0,5, \\ 1 \quad \text{bei} \quad \geq 0,5. \end{cases} \tag{4}$$

Das Training der logistischen Regression erfolgt durch die Anpassung des Gewichtungsvektors θ [12]. Mit der Kostenfunktion

$$\mathbf{J}(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}log(p^i) + (1 - y^i)log(1 - p^i)] \tag{5}$$

lassen sich die durchschnittlichen Kosten über sämtliche Trainingsdatenpunkte berechnen. Diese wird als Log Loss bezeichnet. Da es keine bekannte Formel gibt, mit der ein θ berechnet werden kann, welches die Kostenfunktion minimiert, kann diese Minimierung über einen Optimierungsalgorithmus wie dem Gradientenverfahren erfolgen. Für die Anwendung des Gradientenverfahrens wird die partielle Ableitung der Gleichung 4 benötigt. Die Ableitung nach dem Modellparameter θ_j ergibt die in der Gleichung dargestellt partielle Ableitung der Kostenfunktion

$$\frac{\partial}{\partial\theta_j}\mathbf{J}(\theta) = \frac{1}{m}\sum_{i=1}^{m}(\sigma(\theta^\mathbf{T}\mathbf{x^i}) - y^i)x_j^i. \tag{6}$$

Mit dieser Formel wird für jeden Datenpunkt der Vorhersagefehler berechnet, welcher mit dem Merkmalswert j multipliziert wird, um abschließend den Mittelwert aller Trainingspunkte zu bestimmen. Der daraus entstandene Gradientenvektor kann im Batch-Gradientenverfahren genutzt werden [12]. Da die logistische Regression nur zwischen

zwei Zielgrößen unterscheiden kann, werden die binären Werte wie folgt definiert:

- 1: normaler Zustand und
- 0: fehlerhafter Zustand.

Wird die logistische Regression nicht für eine binäre Klassifikation, sondern für eine Klassifikation mit mehreren Klassen verwendet, kann dies zum einem über das Verfahren "Einer gegen alle", bei dem für jede Klasse ein Klassifikator gegen alle anderen Klassen erstellt wird, zum anderen über das multinominale Logit-Modell, erfolgen [12, 27, 28].

Bei dem multinominalen Logit-Modell ist y_i eine Zielgröße mit mehr als zwei Klassen und $\mathbf{W}$ eine Matrix, in der jeder Zeilenvektor $\mathbf{W}_k$ einer Klasse k entspricht. Die Klassenwahrscheinlich $\hat{p}_k$ wird durch

$$\hat{p}_k(\mathbf{x_i}) = \frac{e^{(\mathbf{x_i}\mathbf{W_k}+\mathbf{W_{0,k}})}}{\sum_{l=0}^{K-1} e^{\mathbf{x_i}\mathbf{W_l}+\mathbf{W_{0,l}}}} \tag{7}$$

ermittelt. Das Modell wird durch:

$$\min_{\mathbf{W}} -C \sum_{i=1}^{n} \sum_{k=0}^{K-1} [y_i = k] log(\hat{p}_k(\mathbf{x}_i)) + r(\mathbf{W}) \tag{8}$$

hinsichtlich $\mathbf{W}$ optimiert [28].

2.3.3. Entscheidungsbaum

Für die Erstellung eines Entscheidungsbaums werden Daten benötigt, die über ein Label verfügen [26]. Damit gehört dieser Lernalgorithmus zu den überwachten Lernverfahren. Ein Entscheidungsbaum besteht aus einem Wurzelknoten, der den Beginn des Entscheidungsbaums darstellt und in dem alle Daten abgebildet sind [26]. Ausgehend vom Wurzelknoten wird ein Entscheidungsbaum solange in interne Entscheidungsknoten unterteilt, bis ein Blattknoten erreicht wird, welcher einen Ausgabewert darstellt [11]. Bei der Unterteilung an einem internen Knoten wird immer nur ein Feature betrachtet. Die Auswahl des Features, welches für die Aufteilung verwendet wird, kann beispielsweise durch die Entropie

$$\eta(t) = -\sum_{i=1}^{k} (p_i * log_2(p_i(t))) \tag{9}$$

oder den Gini-Index

$$g(t) = 1 - \sum_{i=1}^{k} (p_i(t)^2) \tag{10}$$

erfolgen. $\eta(t)$ stellt den Datensatz an einem Knoten dar. i stellt die Klassen dar, welche sich im Datensatz $\eta(t)$ befinden. Wie hoch die Wahrscheinlichkeit ist, dass ein Objekt in die Klasse i fällt, wird durch p_i ausgedrückt. Ist die Wahrscheinlichkeit $p_i = 1$, dass ein Datenpunkt der Klasse i an einem Knoten vorliegt und die Wahrscheinlichkeit für eine Datenpunkt der Klasse j $p_j = 0$, ergibt sich ein Wert für die Entropie und den Gini-Index von 0. Damit liegt eine völlige Homogenität vor. Das Ziel bei der Auswahl eines Features, welches für die Aufteilung eines Knotens verwendet wird, ist die Heterogenität zu reduzieren. Wenn ein interner Entscheidungsknoten nicht weiter unterteilt werden kann, handelt es sich um einen Blattknoten. Dieser Blattknoten stellt eine Klasse dar. Sollte ein Blattknoten nicht ausschließlich aus einer Klasse bestehen, wird dem Blattknoten die Klasse zugeordnet, welche am häufigsten vorhanden ist [26].

Abbildung 2.12 stellt einen Entscheidungsbaum dar, welcher die Eingabedaten in zwei Klassen unterteilt.

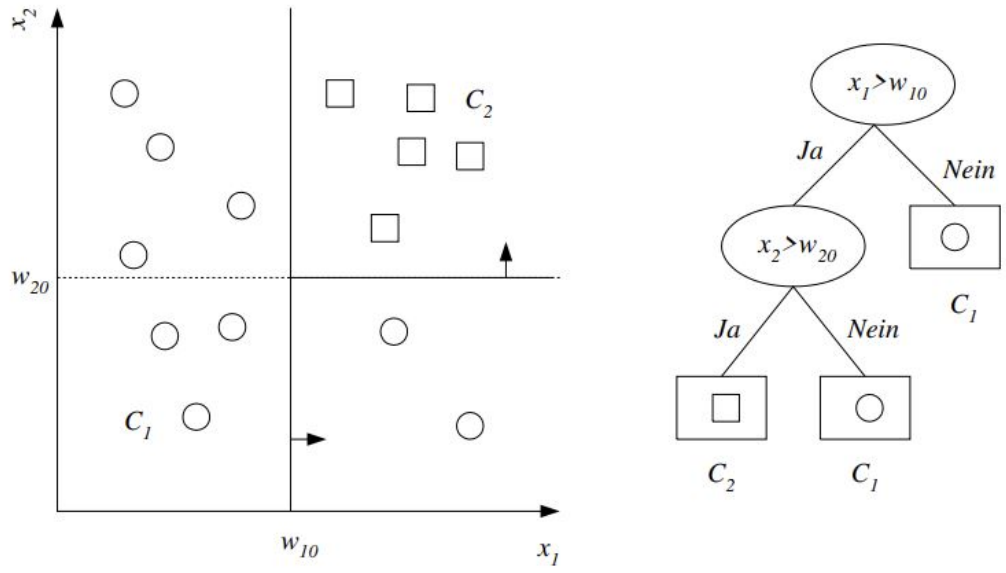

Abbildung 2.12.: Beispiel eines Entscheidungsbaums [11, S. 228]

Um einer Überanpassung bei einem Entscheidungsbaum zu begegnen, wird dieser zurechtgestutzt. Dabei wird verhindert, dass bei einem Entscheidungsbaum zu viele Verästelungen erstellt werden. Dadurch wird die Komplexität des Baumes reduziert und unsinnige Verästelungen reduziert, wodurch der Entscheidungsbaum transparenter

wird. Dieses Verfahren wird als Pruning bezeichnet und kann in Pre- und Post-Pruning unterteilt werden. Wird bei der Erstellung eines Entscheidungsbaums die Pre-Pruning Methode verwendet, wird durch den Einsatz eines Stopp-Kriteriums die maximale Tiefe des Baumes, eine Mindestverbesserungsrate oder eine Mindestgröße der Blattknoten bereits bei der Aufstellung vermieden, dass sich der Baum zu stark verästelt. Das Post-Pruning wird nach der Erstellung eines Entscheidungsbaums verwendet. Dabei werden Entscheidungsknoten durch Blattknoten ersetzt, wodurch die Komplexität verringert wird [29].

2.3.4. Random Forest

Der Random Forest gehört zu den Ensemble-Methoden. Bei einer Ensemble-Methode werden die Ergebnisse mehrerer Prädiktoren verwendet, um eine Vorhersage zu treffen. Beim Hard-Voting-Klassifikator wird dabei, wie in Abbildung 2.13, die Vorhersage vom Ensemble getroffen, die am häufigsten ausgewählt wird [12]. Im Fall des Random

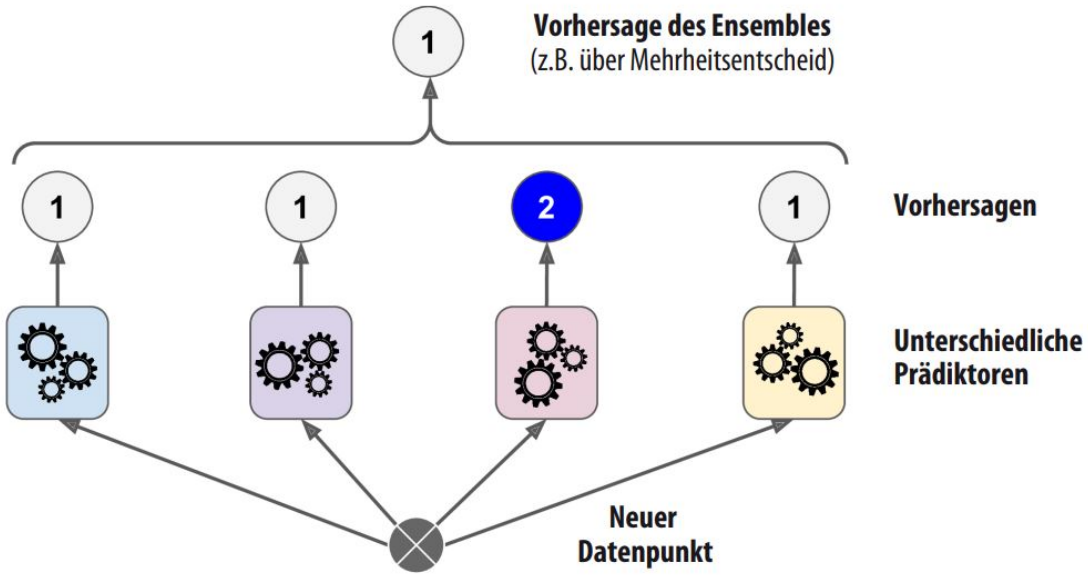

Abbildung 2.13.: Vorhersagen eines Hard-Voting-Klassifikators [12, S. 192]

Forest wird eine Vielzahl verschiedener Entscheidungsbäume erstellt. Damit sich die Entscheidungsbäume unterscheiden, werden diese aus einer zufälligen Teilmenge des Trainingsdatensatzes trainiert, wobei es sich um eine zufällige Stichprobe mit zurücklegen handelt. Dieses Verfahren wird als Bagging bezeichnet [12]. Ein weiteres Zufallselement, das bei einem Random-Forest zum Einsatz kommt, ist die Berücksichtigung einer re-

duzierten und zufälligen Auswahl von Features bei der Erstellung des jeweilig nächsten Knotens [9]. Durch die Zurückhaltung von Informationen bei der Erstellung der jeweiligen Entscheidungsbäume sinkt die Leistungsfähigkeit der einzelnen Bäume, jedoch wird ein Overfitting dadurch verhindert [9].

2.3.5. Support Vector Machine

Die SVM dient grundsätzlich zur Trennung von zwei Klassen durch eine Hyperebene [30]. Damit die SVM eine bessere Generalisierbarkeit erreicht, soll der Abstand der Hyperebene zu den Instanzen, wie in Abbildung 2.14, maximal sein. Dieser Abstand wird als Margin bezeichnet [11]. Um die Instanzen r^t beider Klassen voneinander zu trennen,

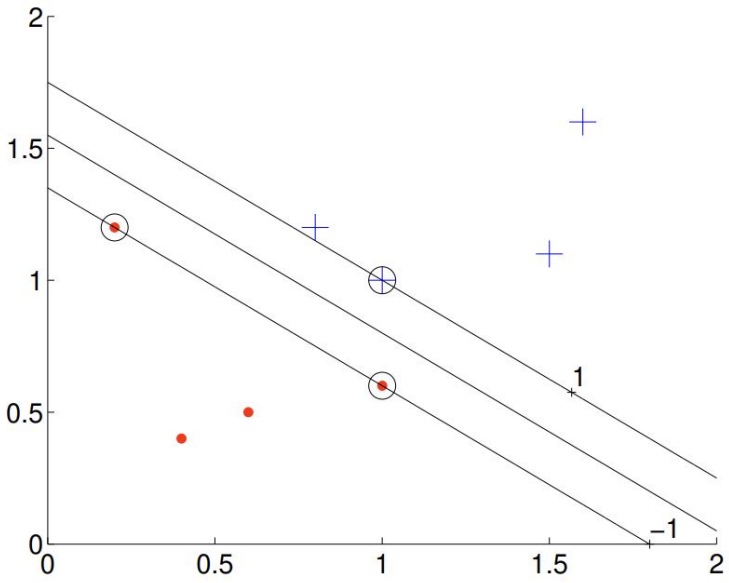

Abbildung 2.14.: Zweiklassenproblem einer SVM [11, S. 414]

werden die Klassen mit dem Label -1 und +1 versehen. Anhand der Bedingungen

$$\begin{aligned} \mathbf{w^T} * \mathbf{x^t} + w_0 &\geq +1, \\ \mathbf{w^T} * \mathbf{x^t} + w_0 &\leq -1, \end{aligned} \tag{11}$$

welche zu

$$r^t(\mathbf{w^T} * \mathbf{x^t} + w_0) \geq +1 \tag{12}$$

umformuliert werden können, wird sichergestellt, dass sich die Instanzen auf der richtigen Seite der Hyperebene befinden. Die Maximierung des Margin erfolgt durch die Minimie-

rung des folgenden Problems:

$$min\frac{1}{2}||\mathbf{w}||^2 \quad \text{unter} \quad r^t(\mathbf{w^T} * \mathbf{x^t} + w_0) \geq +1, \forall t. \tag{13}$$

Die Lösung des Problems aus Gleichung 13 erfolgt unter Verwendung der Lagrange-Multiplikatoren sowie der Karush-Kuhn-Tucker-Bedingungen, woraus sich die Gleichung:

$$L_d = \frac{1}{2}\sum_{t=1}\sum_{s=1} a^t a^s r^t r^s (\mathbf{x^t})^\mathbf{T}\mathbf{x^s} + \sum t_{t=1} a^t \tag{14}$$

welche bezüglich a^t unter der Nebenbedingung:

$$\sum_t a^t r^t = 0 \quad a^t \geq 0, \forall t \tag{15}$$

maximiert wird, ergibt [11].

Da nicht jedes Klassifikationsproblem linear trennbar ist, gibt es eine Vielzahl von Möglichkeiten, die Gleichungen der SVM zu erweitern. Eine Möglichkeit ist die sogenannte Soft-Margin-Trennebene, welche durch den Einsatz einer Schlupfvariablen nach einer Hyperebene sucht, die den geringsten Fehler verursacht. Durch den Einsatz des Kernel-Tricks können nicht lineare Probleme durch eine nicht lineare Transformation in einem neuen Raum gelöst werden. Durch die Erhöhung der Dimension kann ein lineares Modell im neuen Raum erstellt werden, bei dem es sich um ein nicht lineares Modell im Originalraum handelt [11].

Wie eingangs beschrieben, handelt es sich bei der SVM um einen binären Klassifikator. Um diesen Algorithmus bei einem Problem mit mehreren Klassen zu verwenden, gibt es die Strategien:

- Einer gegen alle: Bei dieser Strategie wird jede Klasse gegen alle anderen separiert und ein Entscheidungsscore ermittelt. Abschließend wird die Klasse mit dem höchsten Entscheidungsscore ausgewählt [11, 12].
- Einer gegen einen: In dieser Strategie wird jede Klasse gegen jede andere Klasse einzeln trainiert. Beim Durchlauf durch alle erstellten Klassifikatoren wird letztendlich die Klasse ausgewählt, die am häufigsten vorkommt [12].

2.3.6. Neuronales Netz

Ein neuronales Netz besteht aus einer Vielzahl einzelner künstlicher Neuronen, die auch Perzeptronen genannt werden. Jedes Neuron, siehe Abbildung 2.15, besteht aus mindestens einem Eingang, welcher als x_1 bis x_n bezeichnet wird, dem Neuron mit der dazugehörigen Aktivierungsfunktion und einem Ausgang A [9].

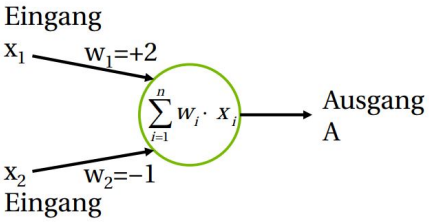

Abbildung 2.15.: Einzelnes künstliches Neuron (Perzeptron) mit Eingängen x_1 und x_2 und einem Ausgang A [9, S. 114]

Wie in Abbildung 2.15 dargestellt, besitzt jeder Eingang ein Gewicht w_1 bis w_n, welches mit den Eingaben in der Aktivierungsfunktion des Neurons multipliziert wird. Die dargestellte Aktivierungsfunktion würde die Werte, ohne diese in der Höhe zu beschränken, an den Ausgang A weiterleiten. Zudem existiert kein Schwellenwert, der zu einer Aktivierung des Neurons führen würde. Aus diesen Gründen muss die Aktivierungsfunktion angepasst werden und beispielsweise die Heaviside-Funktion:

$$A = \begin{cases} 0 = (\sum_{i=1} x_i - b) < 0 \\ 1 = (\sum_{i=1} x_i - b) \geq 0 \end{cases} \tag{16}$$

verwendet werden [9].

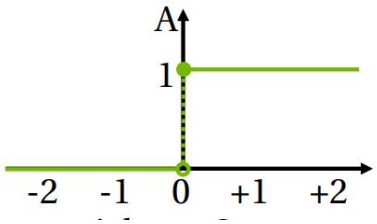

Abbildung 2.16.: Heaviside-Funktion [9, S. 114]

Die Heaviside-Funktion ergibt den Wert 1, sobald die gewichtete Summe der Eingänge den Schwellenwert b überschreitet. Als Konvention kann festgehalten werden, dass ein Ausgang aktiviert wird, sobald die Aktivierungsfunktion den Wert 1 annimmt und nicht

aktiviert wird, wenn das Ergebnis 0 ist [9]. Mithilfe der Heaviside-Funktion kann ein Neuron dazu verwendet werden, den logischen Zusammenhang eines Nicht-Oder-Gatters darzustellen. Tabelle 2.1 stellt den Zusammenhang eines Nicht-Oder-Gatters dar, bei dem der Ausgang 0 ergibt, wenn die beiden Eingänge aktiviert sind. Mithilfe dieses logischen Zusammenhangs ist es möglich, alle logischen Schaltungen der Digitaltechnik durch ein oder mehrere Gatter zu realisieren und damit alle Programme und Algorithmen [9].

Tabelle 2.1.: Eingabe-Ausgabe-Matrix für ein künstliches Neuron mit zwei Eingängen $w_1 = w_2 = -1$ und dem Schwellenwert $b = -1$ [9, S. 115]

Eingang x_1	**Eingang** x_2	**Gewichtete Summe**	**Ausgang A**
0	1	0 $\geq$ -1,5	1
0	0	-1 $\geq$ -1,5	1
1	1	-1 $\geq$ -1,5	1
1	0	-2 < -1,5	0

Durch die Verknüpfung mehrerer Neuronen entsteht ein neuronales Netz. Das Feed-Forward-Netz stellt eine einfache Variante eines neuronalen Netzes dar. Darin werden die einzelnen Neuronen in eine Eingabe-, Verdeckte- und Ausgabeschicht unterteilt. Die Verknüpfungen innerhalb des Netzes verlaufen im Feed-Forward-Netz, wie in Abbildung 2.17 dargestellt, von der Eingabeschicht hin zur Ausgabeschicht [9].

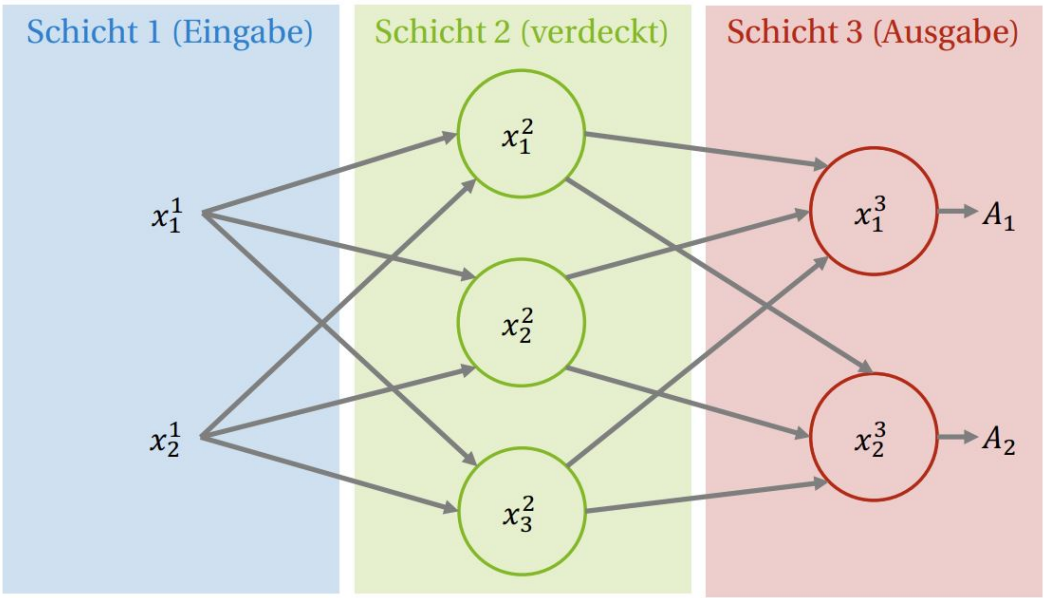

Abbildung 2.17.: Dreischichtiges künstliches neuronales Feed-Forward-Netz bestehend aus Eingabe-, Verdeckt- und Ausgabeschicht [9, S. 114]

Die Eingabeschicht stellt die Anzahl der Eingabeeigenschaften für das neuronale

Netz dar und die Ausgabeschicht die möglichen Klassifizierungen. Mit einem neuronalen Netz können durch die Aktivierung mehrerer Neuronen in der Ausgabeschicht mehrere Klassifikationen in Kombination auftreten. Die Anzahl der verdeckten Schichten sowie die darin enthaltenen Parameter müssen als Hyperparameter vom Benutzer festgelegt werden. Die Hyperparameter müssen abhängig von dem vorliegenden Problem und dessen Komplexität gewählt werden. Werden mehrere verdeckte Schichten verwendet, wird das Modell als Deep Learning bezeichnet [9]. Um ein neuronales Netz zu trainieren, wird eine Aktivierungsfunktion benötigt, die im Gegensatz zur Heaviside-Funktion keine Sprunggrenze aufweist. Stattdessen wird eine Funktion wie die Sigmoid-Funktion verwendet:

$$sig(z) = \frac{1}{1 + e^{-z}} = \frac{e^z}{1 + e^z}, \tag{17}$$

wobei z der gewichteten Summe abzüglich des Schwellenwertes entspricht:

$$z = \sum_{i=1} w_i * x_i - b. \tag{18}$$

Der Vorteil dieser Funktion ist, dass diese stetig, streng monoton steigend ist und dadurch eine zielgerichtete Optimierung ermöglicht wird. Kommt es beispielsweise zur Anpassung eines Gewichts w ändert sich der Ausgabewert. Erfolgt die Änderung des Ausgabewertes in Richtung des korrekten Wertes, kann die Änderung beim nächsten Schritt vorgesetzt werden. Entfernt sich der Ausgabewert vom korrekten Wert, muss die Optimierung in die entgegengesetzte Richtung erfolgen. Bei der Heaviside-Funktion hingegen würde sich der Ausgabewert immer nur dann ändern, wenn die Sprunggrenze überschritten wird. Damit wäre es nicht möglich zu entscheiden, ob die Änderung eines Parameters einen positiven oder negativen Einfluss auf den Ausgabewert hat und in welche Richtung der Wert weiter angepasst werden muss. Ein weiterer Vorteil der Sigmoid-Funktion ist, dass keine eindeutige Zuordnung der Klasse mittels 0 und 1 stattfindet. Damit kann ein neuronales Netz auch für Regressionen verwendet werden [9]. Neben der Sigmoid-Funktion wird häufig die Tangens-hyperbolicus-Funktion sowie die sogenannten ReLU- oder leaky ReLU-Funktion als Aktivierungsfunktion verwendet. Die Tangens-hyperbolicus-Funktion bietet die Möglichkeit, Werte in einem Bereich von -1 bis 1 abzubilden und ist wie folgt definiert:

$$A(z) = tanh(z) = \frac{sinh(z)}{cosh(z)} = \frac{e^z - e^{-z}}{e^z - e^{-z}}. \tag{19}$$

Die ReLU- und leaky ReLU-Funktion haben den Vorteil, dass sie im Gegensatz zur Sigmoid- und Tangens-hyperbolicus-Funktion keine Division und keine Exponentialfunktion und damit keine rechenintensiven mathematischen Operationen verwenden. Die ReLU-Funktion ist mit

$$A = \begin{cases} 0 & \text{für} \quad z < 0 \\ 1 & \text{für} \quad z \geq 0 \end{cases} \tag{20}$$

und die leaky ReLU-Funktion mit

$$A = \begin{cases} 0,01 * zfrz < 0 \\ 1frz \geq 0 \end{cases} \tag{21}$$

beschrieben [9].

Das Training des künstlichen neuronalen Netzes erfolgt durch eine Rückpropagierung. Bei der Rückpropagierung werden alle Modellparameter gleichzeitig angepasst und nach jeder Anpassung die mittlere quadratische Abweichung der berechneten Ausgabewerte von den korrekten Ausgabewerten bestimmt. Die mittlere quadratische Abweichung wird mit

$$E = \frac{1}{2n} \sum_{i=1}^{n} (y_i - A(x_i, w, b))^2 \tag{22}$$

berechnet [9]. Die Anzahl der Datenpunkte, die in einer Optimierungsschleife verwendet werden, werden als Batch bezeichnet. Eine Optimierungsschleife besteht aus sechs Schritten, wobei die Schritte drei bis sechs so lange wiederholt werden, bis sich keine Verbesserung des Fehlers mehr einstellt oder eine definierte Anzahl an Durchläufen erreicht wurden:

1. Berechnung des Fehlers E nach der zufälligen Festlegung aller Modellparameter.
2. Veränderung der Modellparameter w und b um zufällige Werte.
3. Erneute Berechnung des Fehlers E.
4. Berechnung des Einflusses jedes Neurons auf den Fehler E ausgehend von der Ausgabeschicht hin zur Eingabeschicht.
5. Da durch Schritt 4 bekannt ist, welchen Anteil jedes Neuron an dem Fehler E hat, kann bestimmt werden, in welchem Verhältnis die Veränderung des Fehlers E und die Änderungen der Parameter w und b stehen.
6. Anpassung der Modellparameter mithilfe des Gradientenabstiegs [9].

Beim Gradientenabstieg werden die Modellparameter optimiert, um den Fehler E zu minimieren [9]. Dabei ist die Fehlerfunktion E eine differenzierbare Funktion eines Vektors von Variablen, die die Modellparameter darstellen. Der Gradientenvektor

$$\nabla_w E = [\frac{\partial E}{\partial w_1}, ..., \frac{\partial E}{\partial w_n}]^T \tag{23}$$

besteht aus den partiellen Ableitungen der Fehlerfunktion. Der Gradientenabstieg beginnt mit der Wahl eines zufälligen Werts für w. In jeder Schleife, die durchgeführt wird, wird w in entgegengesetzter Richtung zum Gradienten

$$\delta w_i = -\eta \frac{\partial E}{\partial w_i}, \forall i, \tag{24}$$

$$x_i = w_i + \delta x_i \tag{25}$$

aktualisiert [11]. In der Formel 24 beschreibt der Parameter η die Lernrate und nimmt dabei meistens Werte von $\eta << 1$ an [9]. Die Lernrate bestimmt, wie groß die jeweiligen Schritte in eine Richtung sind. Sobald ein Minimum erreicht wurde, ist die Ableitung null, wobei es sein kann, dass lediglich ein lokales Minimum gefunden wurde. Wenn ein Wert für η zu klein gewählt wurde, kann die Konvergenz zu langsam sein, während bei einem zu großen Wert das Minimum übersprungen werden kann [11].

3. Vorgehensweise der Analyse

Die Vorgehensweise zur maschinellen Datenanalyse lässt sich in die in Abbildung 3.1 dargestellten vier Schritte unterteilen.

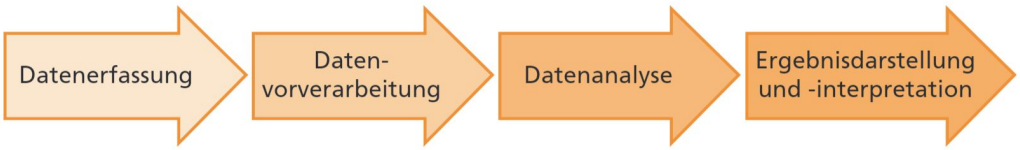

Abbildung 3.1.: Verarbeitungskette der Datenanalyse [13, S. 225]

Zu den vier Schritten zählt die Datenerfassung, in der die relevanten Daten erzeugt oder beschafft werden, sofern diese bereits existieren sollten. Der zweite Schritt ist die Datenvorverarbeitung. Bei der Datenvorverarbeitung werden die Daten bereinigt, transformiert, reduziert oder ggf. integriert und damit die Daten gewonnen, welche für die spätere Analyse maßgeblich sind. Im dritten Schritt werden Informationen durch den Einsatz von maschinellen Lernalgorithmen oder statistischen Methoden aus den Daten ermittelt. Beim letzten Schritt handelt es sich um die Darstellung der Ergebnisse sowie deren Interpretation. Dazu werden die Ergebnisse geprüft, visualisiert und interpretiert, sodass mögliche Zusammenhänge aufgezeigt werden können [13].

3.1. Datenerfassung

Im Zuge der Datenerfassung für die spätere Auswertung müssen zunächst die erforderlichen Daten ermittelt und erfasst werden. Dabei werden die Daten in verschiedenen Datentypen wie beispielsweise transaktionale Daten, Maschinendaten, Netzwerkdaten und Stammdaten unterschieden. Für die prädiktive Instandhaltung von Maschinen stehen dabei vor allem Maschinendaten im Vordergrund. Diese Daten bestehen aus numerischen Daten, Metriken und Messungen. Für die Erhebung dieser Daten werden Sensoren, Kommunikationswege sowie ein Datenerhaltungssystem benötigt [13].

3.2. Datenvorverarbeitung

Die Datenvorverarbeitung spielt eine zentrale Rolle für das Ergebnis der maschinellen Datenanalyse. Um die bestmöglichen Ergebnisse zu erreichen, müssen die Daten geprüft und eventuell verändert werden [13]. Bei der Datenvorverarbeitung erfolgt in der Regel eine:

- Datenbereinigung,
- Datentransformation,
- Datenreduktion sowie die
- Datenintegration [13].

3.2.1. Datenbereinigung

Die Datenbereinigung beinhaltet die Korrektur des Datensatzes von Fehlern und die Ergänzung fehlender Informationen. Dabei werden die Fehler in zwei Klassen unterteilt. Zufällige Fehler sind zum Beispiel Mess- und Übertragungsfehler, Ausreißer oder fehlende Messwerte. Diese Fehler können durch Filterungen der 4-Sigma-Regel oder Löschen der Einträge behoben werden [13, 14]. Bei der 4-Sigma-Regel werden Werte, die außerhalb des Bereichs: $\bar{x} \pm 4s$, also vier Standardabweichungen über und unter dem Mittelwert liegen, als Ausreißer verworfen. Der Mittelwert und die Standardabweichung werden dabei ohne die möglichen Ausreißer berechnet [14].

Systematische Fehler dagegen sind Fehler, die durch eine identifizierbare Ursache permanent die Richtigkeit der erhobenen Daten beeinflussen können. Dazu gehört beispielsweise eine falsche Kalibrierung einer Sensors [13].

3.2.2. Datentransformation

Bei der Datentransformation werden Daten durch Normierung und Standardisierung in eine Form umgewandelt, welche für die Verwendung in maschinellen Lernalgorithmen besser geeignet ist [13].

Wird eine Normierung angewendet, werden die numerischen Werte auf ein Intervall abgebildet [13]. Scikit-Learn enthält für die Normierung die Min-Max-Skalierung, welche mit dem MinMaxScaler Transformer implementiert ist. Diese Skalierung bildet die Werte

in einem Bereich zwischen null und eins ab. [28]

Für die Standardisierung ist in Scikit-Learn ebenfalls ein Transformer implementiert. Durch den StandardScaler werden die Werte so verändert, dass diese mit einer Standardabweichung von eins um den Mittelwert null streuen [28].

Diese Methoden werden bei Werten mit unterschiedlichen Wertebereichen verwendet, da maschinelle Lernalgorithmen nicht gut mit Werten auf unterschiedlichen Skalen arbeiten können [12, 13].

Ein weiterer Teil der Datentransformation ist das Feature-Engineering. Im Rahmen des Feature-Engineering soll eine Teilmenge von Merkmalen identifiziert werden, welche bei der Vorhersage durch die maschinellen Lernalgorithmen relevant ist und zu guten Vorhersageergebnissen führt [31]. Dazu werden statistische Merkmale aus dem Datensatz bestimmt. Mögliche Merkmale, welche von Zhang et al., Khlaief et al und Matzka genannt und im Nachhinein für die maschinellen Lernalgorithmen verwendet werden können, werden im Folgenden vorgestellt [30, 31, 9].

Die ersten Merkmale, welche aus einem Datensatz gewonnen werden können, sind das Maximum:

$$s_1 = max(x_i) \tag{26}$$

und das Minimum:

$$s_2 = min(x_i), \tag{27}$$

welche jeweils den größten und kleinsten Wert in einem Datensatz beschreiben [14]. Die Spannweite gibt den Abstand zwischen dem größten und kleinsten Wert an und wird durch

$$s_3 = max|x_i| - |min(x_i)| \tag{28}$$

ermittelt [14]. Der Mittelwert kann durch:

$$\bar{s_4} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{29}$$

bestimmt werden. Dieser zählt aber zu den empfindlichen Lagemaßen, da Ausreißer einen großen Einfluss auf den Mittelwert ausüben. Ein robusteres Lagemaß gegenüber

dem Mittelwert ist der Median, welcher durch

$$\tilde{s_5} = \begin{cases} x_{(N+1)/2}, \text{ wenn N ungerade ist,} \\ \dfrac{x_{N/2} + x_{(N+1/2)}}{2}, \text{ wenn N gerade ist,} \end{cases} \tag{30}$$

berechnet werden kann [14]. Neben dem Lagemaß kann durch die Varianz:

$$s_6 = \frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})^2 \tag{31}$$

und die Standardabweichung

$$s_7 = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})^2} \tag{32}$$

die Streuung bestimmt werden. Zu den Geometrischen Mittelwerten gehört der quadratische Durchschnitt, welcher durch:

$$s_8 = \frac{1}{N} \sum_{i=1}^{N} x_i^2 \tag{33}$$

bestimmt wird. Die Schiefe wird mittels:

$$s_9 = \frac{1}{N} \sum_{i=1}^{N} \frac{(x_i - s_4)^3}{s_7^3} \tag{34}$$

und die Wölbung wird durch:

$$s_{10} = \frac{1}{N} \sum_{i=1}^{N} \frac{(x_i - s_4)^4}{s_7^4} \tag{35}$$

berechnet. Diese Merkmale geben die in Abbildung 3.2 dargestellte Wölbung und Schiefe einer Verteilung an. Die Wölbung bestimmt, ob eine Verteilung im Verhältnis zur Normalverteilung flacher oder steiler gewölbt ist. Die Schiefe gibt an, ob eine Verteilung links-, rechtssteil oder symmetrisch verteilt ist [14].

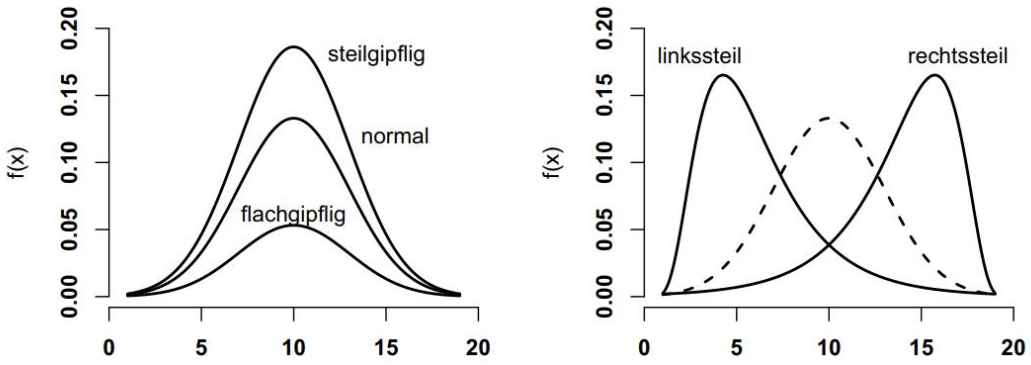

Abbildung 3.2.: Verschiedene Formen einer Verteilung: Exzess (Steilheit) und Schiefe [14, S. 219]

3.2.3. Datenreduktion

Da nicht alle Merkmale den gleichen Beitrag für das Ergebnis der Datenanalyse bieten, wird der Datensatz reduziert. Das hat den Vorteil, dass dadurch zum einen Speicherplatz und zum anderen Rechenleistung gespart werden [13]. Die Auswahl der Merkmale, welche im weiteren Verlauf genutzt werden sollen, kann mit Hilfe der Kovarianz und dem Korrelationskoeffizienten durchgeführt werden [29]. Die Kovarianz wird mit der folgenden Gleichung:

$$\text{Cov}(x, y) = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x}) * (y_i - \bar{y}) \tag{36}$$

ermittelt. Die Kovarianz beschreibt die gemeinsame Schwankung eines Merkmals x und der Zielgröße y um den jeweiligen Mittelwert. Liegt also der Wert des Merkmals und der Zielgröße über oder unter dem Mittelwert und liegen die Werte ähnlich weit über oder unter dem Mittelwert, kann ein Zusammenhang vorliegen. Der Nachteil der Kovarianz liegt aber in der Interpretation des Ergebnisses, da die Werte nicht normiert sind. Durch die Verwendung des Korrelationskoeffizienten kann dieses Problem gelöst werden. Der Korrelationskoeffizient, welcher durch:

$$\text{Kor}(x, y) = \frac{\text{Cov}(x, y)}{\sqrt{\sigma_x^2 * \sigma_y^2}} = \frac{\frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x}) * (y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2 * \sum_{i=1}^{n} (y_i - \bar{y})^2}} \tag{37}$$

berechnet wird, hat den Vorteil, dass dieser auf den Bereich zwischen $\text{Kor}(x, y) \in [-1, 1]$ normiert ist. Eine 1 beschreibt dabei einen perfekten, linear positiven Zusammenhang, während eine -1 einen perfekten, linear negativen Zusammenhang beschreibt. Nimmt der Korrelationskoeffizient den Wert 0 an, so liegt kein linearer Zusammenhang vor. Zwar

können durch die Korrelation nur linear abhängige Daten identifiziert werden, doch können praktische Anwendungen häufig linear angenähert werden, da oftmals ein eingeschränkter Wertebereich bezogen auf die Merkmale aufgenommen wird [29].

Da eine Auswahl eines Merkmals für die spätere Analyse allein durch den Korrelationskoeffizienten dazu führen kann, dass sich kein geeigneter Vektorraum ergibt, kann die Auswahl der Merkmale sequenziell durch einen Greedy-Algorithmus erfolgen. Ein solcher Algorithmus sucht iterativ bei jedem Durchlauf nach der optimalen Auswahl der Merkmale. Dadurch entsteht auf alle Merkmale bezogen nicht das optimale Ergebnis, da nicht alle kombinatorischen Möglichkeiten in Betracht gezogen werden, jedoch ist das Ergebnis in der Praxis gut anwendbar. Zwei Greedy-Algorithmen sind die sequenzielle Rückwärtsauswahl sowie die sequenzielle Vorwärtsauswahl. Für beide Algorithmen gilt die Ausgangssituation, dass ein Merkmalsraum mit d Merkmalen auf $k < d$ verkleinert werden soll. Bei der sequenziellen Rückwärtsauswahl werden solange nach und nach Merkmale aus dem ursprünglichen Merkmalsraum d entfernt, bis eine festgelegte Anzahl von Merkmalen erreicht wird. Bei der sequenziellen Vorwärtsauswahl wird ein leerer Merkmalsraum d in jeder Iterationsschleife solange mit Merkmalen befüllt, bis die vorgegebene Anzahl an Merkmalen dem Merkmalsraum hinzugefügt wird [29].

Eine Methode zur Auswahl von Merkmalen, bei der keine Merkmale aussortiert werden, ist die Hauptkomponentenanalyse. Mittels der Hauptkomponentenanalyse werden die Merkmale von einem Raum, der alle Merkmale umfasst, in einen kleineren Raum projiziert. Die dabei zum Teil neu entstandenen Basisvektoren setzten sich linear aus mehreren Merkmalsvektoren zusammen. Die Zusammensetzung der Merkmalsvektoren erfolgt anhand von Mustern in den Daten, die aufgrund der Korrelationen zwischen den Merkmalen identifiziert werden [29]. Die Berechnung der Hauptkomponenten erfolgt anhand eines Datensatzes, welcher alle Merkmale umfasst und aus k zum Teil miteinander korrelierenden Merkmalen besteht. Für diese eingegebenen Merkmale x_1 bis x_k werden durch die Matrixmultiplikation

$$\begin{bmatrix} w_{11} & \cdots & w_{1k} \\ \vdots & \ddots & \vdots \\ w_{d1} & \cdots & w_{dk} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_k \end{bmatrix} \tag{38}$$

die Gewichte w berechnet, wodurch die Eigenschaften auf einen kleineren Raum mit d reduziert werden. Ein Nachteil bei der Anwendung der Hauptkomponentenanalyse ist, dass die ursprünglich interpretierbaren Daten nach der Kombination nicht mehr interpretierbar sind [9].

3.2.4. Datenintegration

Nachdem die Datenvorverarbeitung abgeschlossen ist, müssen die Daten zu einem Datensatz konsolidiert werden. Das geschieht vor dem Hintergrund, dass die meisten maschinellen Lernalgorithmen nicht in der Lage sind, sich an Daten aus verschiedenen Datenquellen zu bedienen. Bei der Zusammenführung der Daten kann es zu Problemen bei der Identifikation gleicher Objekte kommen sowie Daten redundant auftreten [13].

3.3. Datenanalyse

Die Datenanalyse unterteilt sich in die zwei Schritte:

- explorative Datenanalyse und
- maschinelles Lernen.

Die beiden Schritte unterscheiden sich dahingehend, dass bei der explorativen Datenanalyse manuelle oder semi-automatische Methoden zum Einsatz kommen, um Erkenntnisse aus dem Datensatz zu erhalten, welche im Nachgang für die automatischen Verfahren der maschinellen Lernmethoden verwendet werden können [13].

3.3.1. Explorative Datenanalyse

Bei der explorativen Datenanalyse sollen Informationen durch den Einsatz von statistischen Methoden und Visualisierungstechniken aus den Daten gewonnen werden [13]. Durch die gewonnenen Informationen können Aussagen über die Qualität der vorliegenden Daten getroffen werden [9]. Die explorative Datenanalyse konzentriert sich darauf, Beziehungen zwischen zwei Merkmalen zu untersuchen [8].

Die Beschreibung und Erforschung eines Datensatzes erfolgt mit Hilfe der folgenden Darstellungsarten:

- Tabellen,
- Diagramme oder
- Parameter.

Die verschiedenen Darstellungsarten unterscheiden sich hinsichtlich des Abstraktionsgrades, der Übersichtlichkeit und ihres Informationsgehalts. Eine Tabelle weist einen geringen Abstraktionsgrad auf, da sie Daten in ihrem ursprünglichen Format und Genauigkeit wiedergibt Dadurch verliert diese Darstellungsmethode bei großen Datensätzen an Übersichtlichkeit und ist nur für eine geringe Anzahl an Daten, bei denen der exakte Wert von Bedeutung ist, geeignet. Diagramme setzen die Daten in ein Verhältnis zueinander und stellen damit eine Abstraktion dar. Dadurch besteht die Möglichkeit, sich einen Überblick über die Lage und Verteilung der Daten zu verschaffen. Durch die Darstellung des Datensatzes mittels Parameter erfolgt die höchste Abstraktion. Die Parameter sollen durch eine Kennzahl wie beispielsweise einem Lageparameter einen Überblick über den Datensatz ermöglichen [9].

Für die explorative Datenanalyse eines Datensatzes eignet sich die Lage, um die zentrale Tendenz einer Eigenschaft eines Datensatzes darzustellen. Die Lage kann mit dem arithmetischen Mittel $\bar{x}$ aus Formel 29 beschrieben werden. Dies ist aber anfällig gegenüber Ausreißern und bietet nur eine generelle Einordnung für die Lage der Daten [9]. Im Gegensatz zum arithmetischen Mittelwert stellt der Median aus Formel 30 einen robusteren Wert dar, bei dem Ausreißer an den Enden des Wertebereichs zu keiner Veränderung des Medians führen [9]. Ein weiterer Lageparameter ist der Modus, bei dem der am häufigsten vorkommende Wert bestimmt wird. Deshalb wird dieser Parameter in der Regel bei diskreten und nicht bei kontinuierlichen Werten verwendet, da diese möglicherweise nur einmal im Datensatz vorkommen [9]. Die Aussagekraft des Lageparameters kann durch die Streuung ermittelt werden. Bei einer stark abweichenden Streuung müssen die Werte genauer betrachtet werden, wohingegen die Lage von Eigenschaften mit einer geringen Streuung belastbar ist [9]. Die Streuung wird anhand der Parameter Varianz und Standardabweichung bestimmt. Die Varianz berechnet sich durch die Formel

$$s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \tag{39}$$

für die Stichprobe aus einer Grundgesamtheit. Die Standardabweichung ist die Wurzel der Varianz. Beide Parameter bilden ein Maß für die Streuung der einzelnen Werte um

das arithmetische Mittel der Eigenschaft. Der Vorteil bei der Verwendung der Standardabweichung liegt darin, dass diese die gleiche Dimension und Einheit wie die einzelnen Werte der Eigenschaft besitzt [32].

Die Visualisierung der Daten ist ein wesentlicher Bestandteil der explorativen Datenanalyse und kann mithilfe eines Histogramms, Boxplot Diagramms oder Streudiagramms erfolgen [8, 9, 13]. Zur Visualisierung von kontinuierlichen Merkmalen eignet sich ein Streudiagramm, bei dem zwei kontinuierliche Merkmale auf jeweils einer Achse abgebildet und damit in eine Beziehung zueinander gesetzt werden. Ein Streudiagramm ermöglicht Rückschlüssen über die Korrelation zwischen zwei Merkmalen. In Abbildung 3.3 werden beispielhaft verschiedene Merkmale miteinander in Beziehung gesetzt. Das erste Streudiagramm (a) stellt eine stark positive Korrelation zwischen den Merkmalen ”Höhe” und ”Gewicht” dar. Aus dem zweiten Diagramm (b) kann eine stark negative Korrelation zwischen den Merkmalen ”Alter” und ”Sponsoren Einnahmen” identifiziert werden. Das letzte Diagramm (c) zwischen den Merkmalen ”Alter” und ”Höhe” weißt keinerlei Korrelation zwischen den Merkmalen auf [8]. Eine Streudiagrammmatrix ist eine Möglichkeit,

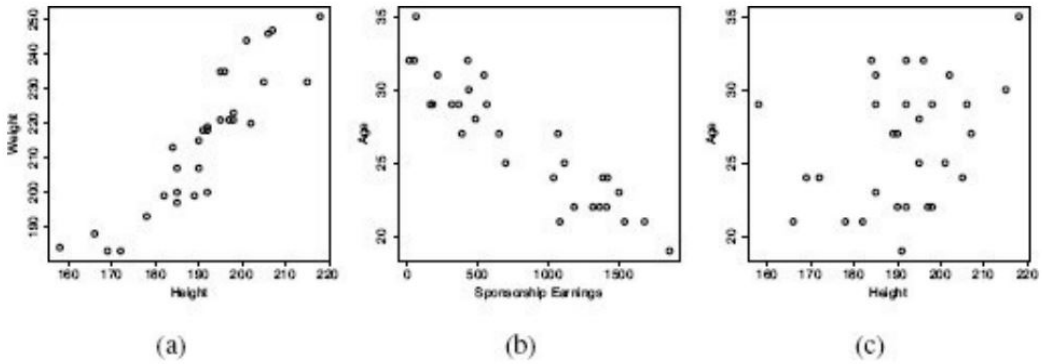

Abbildung 3.3.: Streudiagramm [8, S. 79]

sämtliche kontinuierliche Merkmale gegeneinander abzubilden und schnell eine Übersicht über mögliche Korrelationen zwischen den Merkmalen zu erhalten. Abbildung 3.4 stellt eine solche Matrix für die Beispiele aus Abbildung 3.3 dar. Diese Matrix wird allerdings dadurch begrenzt, dass bei einer zu großen Anzahl von Merkmalen die Übersichtlichkeit verloren geht. Sie sollte daher nur für bis zu acht Merkmalen verwendet werden [8].

Ein weiterer Parameter, der zur Analyse eines Datensatzes verwendet werden kann, ist die in Kapitel 3.2 vorgestellte Korrelation. Um einen Überblick über die Korrelationen

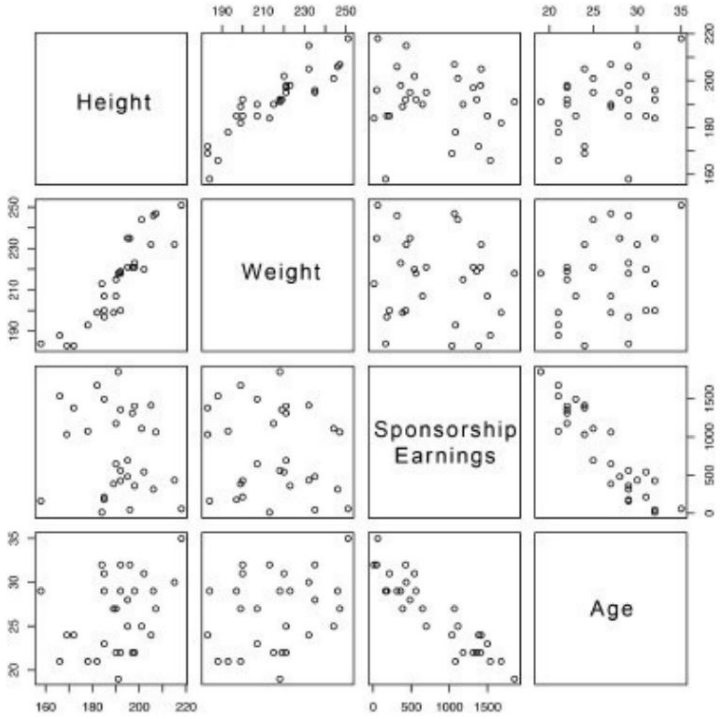

Abbildung 3.4.: Streudiagramm Matrix [8, S. 80]

zwischen den Merkmalen zu erhalten, können diese in einer Korrelationsmatrix, die folgendermaßen aufgebaut ist:

$$\underset{\{a,b,\ldots,z\}}{\text{Korrelationsmatrix}} = \begin{bmatrix} corr(a,a) & corr(a,b) & \cdots & corr(a,z) \\ corr(b,a) & corr(b,b) & \cdots & corr(b,z) \\ \vdots & \vdots & \ddots & \vdots \\ corr(z,a) & corr(z,b) & \cdots & corr(z,z) \end{bmatrix} \tag{40}$$

abgebildet werden [8]. Anhand der Korrelation können lineare Beziehungen zwischen kontinuierlichen Merkmalen identifiziert werden. Des Weiteren können die Beziehungen zwischen Daten mit einem gleichen Korrelationskoeffizienten unterschiedlich sein, wie das in Abbildung 3.5 dargestellte Anscombe-Quartett zeigt. Beim Anscombe-Quartett haben sämtliche Daten nahezu die gleichen statistischen Maßzahlen, weshalb es in der explorativen Datenanalyse entscheidend ist, die Daten zu visualisieren und nicht nur Parameter und Kennzahlen aus dem Datensatz abzuleiten [8].

Die Erkenntnisse der explorativen Datenanalyse werden im Weiteren bei der Anwendung der maschinellen Lernalgorithmen verwendet und können einen Einfluss auf die

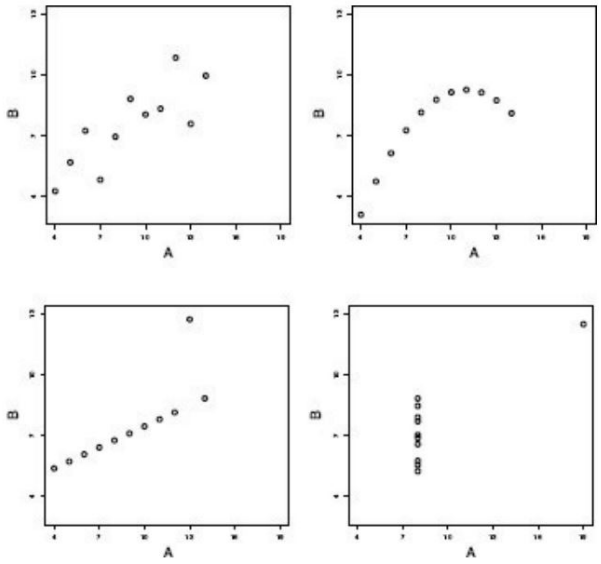

Abbildung 3.5.: Anscombe-Quartett [8, S. 91]

Merkmale haben, die zum Trainieren der maschinellen Lernalgorithmen genutzt werden. Darüber hinaus ist es üblich, eine explorative Datenanalyse bereits in den vorhergegangenen Schritten der Datenanalyse durchzuführen [13].

3.3.2. Maschinelles Lernen

Vor der Anwendung eines maschinellen Lernalgorithmus muss der Datensatz unterteilt werden, um eine Überprüfung des maschinellen Lernmodells zu ermöglichen. Ein Ansatz, den Datensatz zu unterteilen ist, wie in Abbildung 3.6 dargestellt, die Teilung in einen Trainings- und einen Testdatensatz [8]. Der Testdatensatz wird zufällig aus dem Datensatz erstellt. Dazu wird der Datensatz so unterteilt, dass der Trainingsdatensatz 80 % und der Testdatensatz 20 % der ursprünglich vorhandenen Daten enthalten, die im Rahmen des Trainingsprozesses nicht verwendet werden [9, 8]. Nachdem ein Modell aus den Trainingsdaten erstellt wurde, werden den Testdaten dazu verwendet, die Leistung des Modells zu ermitteln. Die Verwendung von für das Modell unbekannten Daten bei der Validierung hat den Vorteil, dass das Modell nicht auf bereits gelernte Daten zurückgreifen kann. Findet die Validierung mit bereits bekannten Daten statt, ist es wahrscheinlich, dass die Leistung des Modells sehr gut ist. Ein weiterer Vorteil ist, dass die Validierungsergebnisse der Testdaten mehr Informationen darüber liefern, ob ein Modell generalisierbar ist [8].

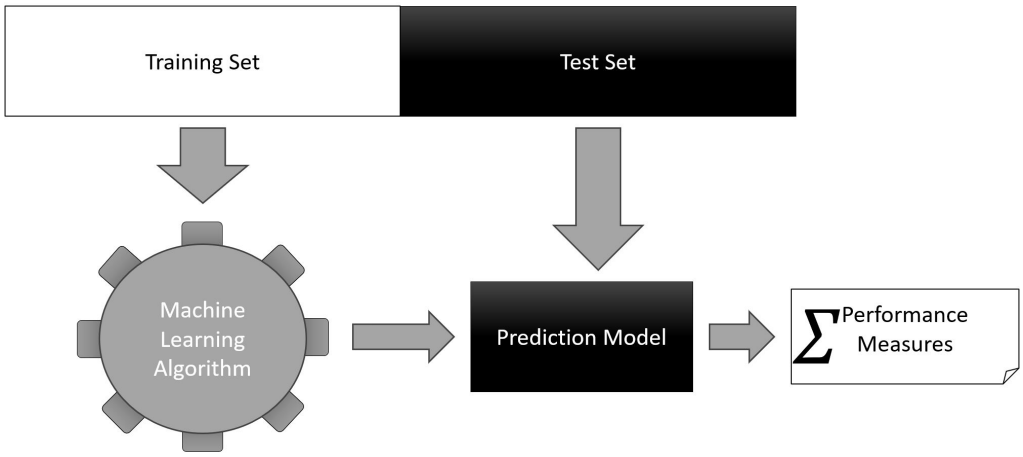

Abbildung 3.6.: Der Prozess der Erstellung und Bewertung eines Modells anhand eines Testdatensatzes. [8, S. 400]

Ein weiterer Ansatz, um die Generalisierung des Modells sicherzustellen, ist die Verwendung eines zusätzlichen Validierungsdatensatzes. Dabei wird der Datensatz zufällig unterteilt in einen Trainingsdatensatz mit 70 % der Daten und in ein Validierungs- und Testdatensatz mit jeweils 15 % der Daten aus dem ursprünglichen Datensatz [9]. Dieser Ansatz wird vor allem verwendet, um Überanpassungen bei iterativen Algorithmen zu vermeiden [8].

Eine Methode, in der alle Daten zum Training des Modells eingeschlossen werden und somit eine mögliche ungünstige Verteilung der Daten auf die erstellten Datensätze verhindert wird, ist die Kreuzvalidierung. Bei einer Kreuzvalidierung wird der Datensatz, wie in Abbildung 3.7 zu sehen ist, zufällig in Unter-Datensätze unterteilt. Anschließend wird fünfmal mit jeweils vier der fünf Unter-Datensätze unabhängig voneinander ein Modell trainiert. Der fünfte Unter-Datensatz dient der Leistungsmessung als Testdatensatz. Danach wird geprüft, ob die Ergebnisse der Leistungsmessung aller Modelle vergleichbar sind. Ist das der Fall, kann davon ausgegangen werden, dass der verwendete Algorithmus generalisierte Modelle für den Datensatz erstellt. Abschließend wird ein weiterer Durchlauf gestartet, in dem alle vorhandenen Daten als Trainingsdaten verwendet werden. Eine Kreuzvalidierung prüft also nicht die Generalisierung des trainierten Modells, sondern ob es mit dem maschinellen Lernalgorithmus möglich ist, auf die bestehenden Daten ein generalisiertes Modell zu erstellen [9].

Nachdem die Daten unterteilt wurden, werden die in Kapitel 2.3 vorgestellten maschinellen Lernmethoden mit den vorliegenden Datensätzen trainiert. Die Analyse des Da-

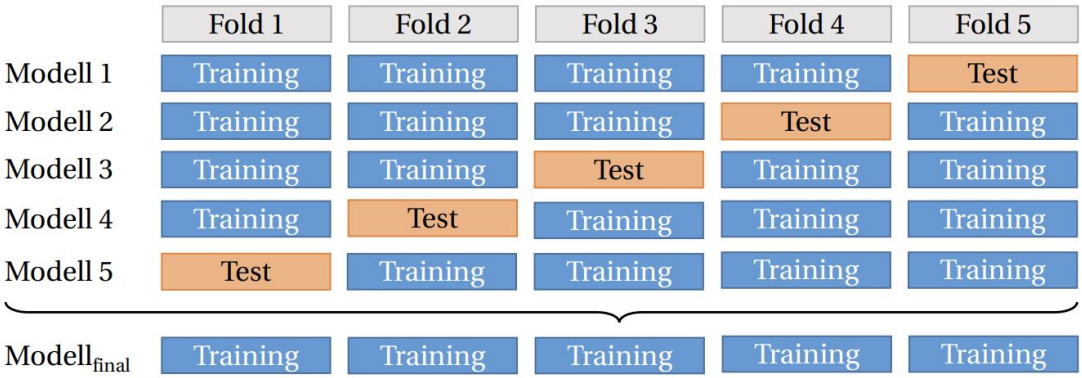

Abbildung 3.7.: Kreuzvalidierung durch Permutation der Verwendung von Unter-Datensätzen (engl. folds) als Trainings- oder Testdatenpunkte. Das finale Modell wird mit allen Trainingsdaten erstellt.[9, S. 156]

tensatzes in dieser Arbeit erfolgt mit der Interpretersprache Python. Außerdem werden die Python-Bibliotheken NumPy, SciPy, Matplotlib, Pandas und scikit-learn verwendet. Für sämtliche maschinellen Lernalgorithmen wird in dieser Arbeit die Bibliothek Scikit-learn verwendet [28]. Diese Bibliothek ermöglicht eine schnelle Implementierung von einer Vielzahl von maschinellen Lernmethoden sowie Methoden zur Datenvorverarbeitung und Ergebnisdarstellung und -evaluierung in Python [33].

3.4. Ergebnisdarstellung und -interpretation

Der letzte Teil der Verarbeitungskette befasst sich mit der Aufbereitung und Präsentation der Ergebnisse der vorhergegangenen Analyse. Dazu müssen alle relevanten Informationen so aufbereitet werden, dass diese in einer leicht zu interpretierenden Form vorliegen. Die Informationen, die eine Ergebnisdarstellung beinhalten muss, sollten die getroffenen Annahmen, die verwendeten Analysewerkzeuge und die Daten, auf deren Grundlage die Ergebnisse erzielt wurden, umfassen. Außerdem sollten die Ergebnisse kritisch hinterfragt werden, ob alle Ziele erfüllt wurden und ob es zu Fehlern gekommen ist bzw. noch Verbesserungspotential vorhanden ist [13, 35].

Eine Möglichkeit, das Ergebnis einer Klassifikation darzustellen und zu interpretieren, ist die in Abbildung 3.8 dargestellte Konfusionsmatrix, die die beiden Klassen ”gut” und ”schlecht” umfasst [13]. Eine Konfusionsmatrix ermöglicht eine Bewertung der Klassifikationsgüte eines maschinellen Lernmodells [13]. Dazu wird die Häufigkeit der möglichen Vorhersageergebnisse für den Testdatensatz berechnet, um das Abschneiden eines Mo-

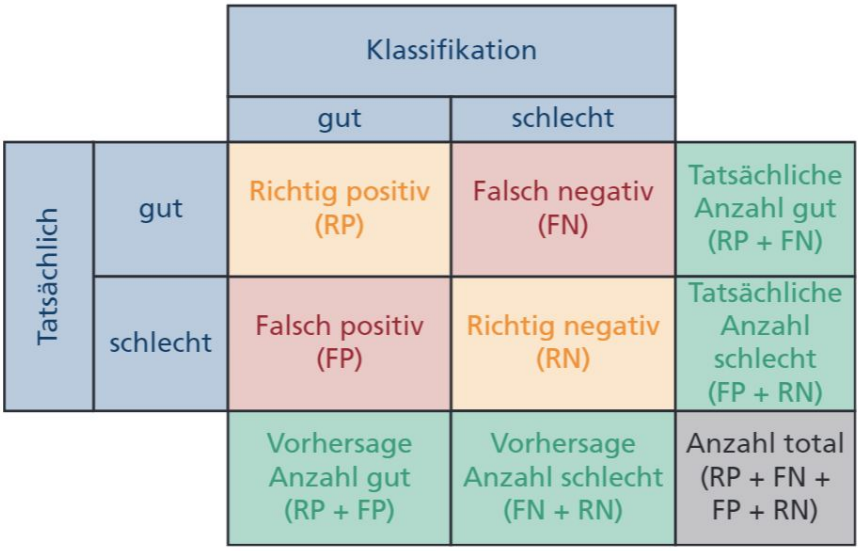

Abbildung 3.8.: Konfusionsmatrix der Klassifizierung [13, S. 255]

dells detailliert zu beschreiben [8]. Bei einer Konfusionsmatrix steht jede Spalte für die vorhergesagte Klasse, während jede Zeile für die tatsächliche Klasse steht. Enthält eine Konfusionsmatrix ausschließlich Werte auf der Hauptdiagonalen, wäre die Klassifikation ohne einen Fehler [12]. Die in Abbildung 3.8 dargestellte Konfusionsmatrix stellt eine binäre Klassifikation dar, anhand derer sich vier Kategorien zur Bewertung der Klassifikationsgüte ergeben [13]:

- Richtig positiv (RP): Ein positiver Eintrag im Testdatensatz, der positiv klassifiziert wurde.
- Richtig negativ (RN): Ein negativer Eintrag im Testdatensatz, der negative klassifiziert wurde.
- Falsch positiv (FP): Ein negativer Eintrag im Testdatensatz, der positiv klassifiziert wurde.
- Falsch negativ (FN): Ein positiver Eintrag im Testdatensatz, der negativ klassifiziert wurde.

Anhand der Konfusionsmatrix können weitere kompakte Qualitätsmaße bestimmt werden. So gibt die

$$\text{Relevanz (engl. Precision)} = \frac{RP}{RP + FP} \tag{41}$$

die Genauigkeit für die positiven Klassifizierungen an [12]. Die Relevanz gibt an, wie oft

eine positive Klassifizierung eines Modells korrekt ist [8]. Die

$$\text{Sensitivität (engl. Recall)} = \frac{RP}{RP + FN} \tag{42}$$

gibt dagegen an, wie viele positive Einträge bei der Klassifikation entdeckt wurden [12]. Sowohl die Relevanz als auch die Sensitivität können Werte zwischen null und eins annehmen, wobei ein höherer Wert in beiden Fällen auf eine bessere Modellbildung hindeutet [8]. Ein weiteres Qualitätsmaß, welches anhand der Konfusionsmatrix bestimmt werden kann, ist der

$$F_1\text{-Score} = \frac{2}{\frac{1}{\text{Relevanz}} + \frac{1}{\text{Sensitivität}}} = \frac{RP}{RP + \frac{FN + FP}{2}}, \tag{43}$$

welcher der harmonische Mittelwert der Relevanz und Sensitivität ist. Dieses Qualitätsmaß eignet sich gut zum Vergleich von Klassifikatoren, da beim harmonischen Mittelwert die niedrigen Mittelwerte ein höheres Gewicht erhalten und damit Klassifikatoren nur einen hohen F_1-Score erhalten, wenn die Relevanz und Sensitivität hoch sind [12]. Des Weiteren kann die

$$\text{Klassifizierungsgenauigkeit} = \frac{RP + RN}{RP + RN + FP + FN} \tag{44}$$

bestimmt werden. Diese liegt ebenfalls im Wertebereich zwischen null und eins, wobei auch bei diesem Wert eine höhere Genauigkeit für ein besseres Modell steht [8]. Die Genauigkeit gibt das Verhältnis der Testdaten an, welche mittels des Modells richtig positiv und richtig negativ, im Gegensatz zu den Testdaten, die falsch positiv und falsch negativ klassifiziert wurden [9].

Anhand der Konfusionsmatrix und den Qualitätsmaßen kann einerseits abgeleitet werden, ob ein Modell generalisierbare, überangepasste, unterangepasste oder zufällige Ergebnisse liefert und andererseits kann die Leistung eines Modells bewertet werden. Von einer Generalisierung eines Modells kann dann ausgegangen werden, wenn die Klassifizierungsgenauigkeiten sowohl für die Trainingsdaten als auch für die Testdaten nahezu identische Ergebnisse liefern. Sind die Ergebnisse bei einem Trainingsdatensatz deutlich besser als die bei dem Testdatensatz, kann von einer Überanpassung ausgegangen werden. Zwar kann der Testdatensatz aus schwierig zu klassifizierenden Daten

bestehen, dennoch führt diese Imbalance zwischen der Genauigkeit des Trainings- und Testdatensatz dazu, dass der Klassifikator keine generalisierbaren Ergebnisse liefert und daher nicht belastbar ist. Eine Unteranpassung liegt dann vor, wenn sowohl die Genauigkeit der Trainingsdaten als auch der Testdaten niedrig ist und das Modell daher nicht gut angepasst ist. Ist das Ergebnis der Klassifizierungsgenauigkeit beim Testdatensatz deutlich besser als das Ergebnis der Trainingsdaten, liegt wahrscheinlich ein Zufallsergebnis vor [9].

4. Anwendungsbeispiel

Die im Kapitel 3 beschriebenen Vorgehensweisen und Methoden werden im Folgenden praktisch angewendet. Im Rahmen eines Anwendungsbeispiels erfolgt mittels eines Datensatzes die Klassifizierung von Fehlerzuständen einer Standbohrmaschine [36]. Dazu wird zunächst der Datensatz beschrieben. Anschließend wird der Datensatz vorverarbeitet, sodass zum einen Merkmale aus dem Datensatz abgeleitet und zum anderen die Merkmale ausgewählt werden, welche sich am besten für die Erstellung eines maschinellen Lernmodells eignen. Anschließend wird anhand der gewonnenen Merkmale eine explorative Datenanalyse durchgeführt, um die ausgewählten Merkmale weiter zu analysieren und die Daten zu beschreiben, bevor damit verschiedene maschinelle Lernalgorithmen trainiert und angepasst werden. Abschließend werden die Klassifizierungsergebnisse der verschiedenen Algorithmen ausgewertet und miteinander verglichen.

4.1. Beschreibung des Datensatzes

Der Datensatz, welcher diesem Anwendungsbeispiel zugrunde liegt, beschreibt insgesamt vier Maschinenzustände einer Standbohrmaschine:

1. Normalzustand
2. Holzsockel, welcher an der Standbohrmaschine befestigt wurde
3. Unwucht, die durch ein Gewicht am Ende des Holzsockels erzeugt wird
4. Unwucht, die durch das Hinzufügen von Gewichten an zwei Enden des Holzsockels erzeugt wird

Die Anzahl der Datenpunkte für die einzelnen Zuständen sind wie in Tabelle 4.1 dargestellt verteilt.

Tabelle 4.1.: Verteilung der Datenpunkte im Datensatz

Maschinenzustand	**Anzahl Datenpunkte**
Normalzustand	10802
Holzsockel	6604
Unwucht mit einem Gewicht	9229
Unwucht mit zwei Gewichten	9193

Der Datensatz besteht zum einen aus Vibrationsdaten, welche mit einer Frequenz

von 100 Hz von der x-, y- und z-Achse erfasst werden und zum anderen aus der Gesamtbeschleunigung, welche mit allen Vibrationssensoren aufgezeichnet wird [36].

Zur weiteren Beschreibung des Datensatz werden die statistischen Kennzahlen der einzelnen Maschinenzustände gebildet und in den Tabellen 4.2, 4.3, 4.4 und 4.5 dargestellt. Die Kennzahlen aus den Tabellen zeigen, dass der Mittelwert und der Median der Vibrationen in den Raumebenen nahe null sind. Im Normalzustand, dessen Kennzahlen in Tabelle 4.2 abgebildet sind, liegt die Streuung der Messpunkte der Raumachsen zwischen 0,61 und 0,95. Die erfassten Messwerte liegen im Bereich zwischen -3,42 und 3,49, wobei es zu solch starken Vibrationen nur auf der z-Achse kommt.

Tabelle 4.2.: Statistische Kennzahlen Normalzustand

Statistische Kennzahl	**ax**	**ay**	**az**	**aT**
Mittelwert	0,01	0,008	-0,12	1,26
Standardabweichung	0,61	0,78	0,95	0,57
Minimal Wert	-2,04	-2.14	-3,42	0,03
25 % Quantil	-0,41	-0,63	-0,74	0,85
50 % Quantil	0,02	-0,02	-0,15	1,16
75 % Quantil	0,46	0,64	0,42	1,57
Maximal Wert	2,1	2,00	3,49	3,51

Verglichen mit den Kennzahlen des Normalzustandes zeigen die Kennzahlen der befestigten Bohrmaschine in Tabelle 4.3, dass die Vibrationen gedämpft werden. Dadurch sinkt die Streuung von 0,95 auf einen maximalen Wert von 0,74 und der Bereich, indem die Messwerte erfasst wurden, erstreckt sind nur noch von -2,51 bis 2,51.

Tabelle 4.3.: Statistische Kennzahlen des Holzsockels

Statistische Kennzahl	**ax**	**ay**	**az**	**aT**
Mittelwert	0,01	-0,003	-0,09	0,92
Standardabweichung	0,44	0,53	0,74	0,43
Minimal Wert	-1,40	-1,69	-2,51	0,04
25 % Quantil	-0,31	-0,43	-0,57	0,63
50 % Quantil	0,02	-0,01	-0,12	0,87
75 % Quantil	0,31	0,42	0,35	1,15
Maximal Wert	1,65	1,52	2,51	2,58

Durch das Anbringen eines Gewichts zum Erzeugen einer Unwucht erhöht sich, wie in Tabelle 4.4 dargestellt ist, die Streuung der erfassten Werte auf maximal 1,23. Ebenso wird der Bereich, in dem Werte erfasst wurden, größer. So ist die Beschleunigung der Vibration auf der z-Achse auf Werte zwischen -4,28 und 4,45 gestiegen beziehungswei-

se gesunken. Aber auch in den anderen Raumebenen ist anhand der Kennzahlen zu beobachten, dass die Unwucht dazu führt, dass die Beschleunigungen stärker werden.

Tabelle 4.4.: Statistische Kennzahlen Unwucht mit einem Gewicht

Statistische Kennzahl	**ax**	**ay**	**az**	**aT**
Mittelwert	0,01	0,01	-0,134	1,64
Standardabweichung	0,67	1,13	1,23	0,75
Minimal Wert	-2,35	-3,18	-4,28	0,089
25 % Quantil	-0,42	-0,81	-0,95	1,08
50 % Quantil	-0,03	-0,01	-0,14	1,55
75 % Quantil	0,42	0,83	0,66	2,12
Maximal Wert	2,66	3,32	4,45	4,64

In Tabelle 4.5 ist der Zustand mit zwei Gewichten, die eine Unwucht erzeugen, dargestellt. Durch das weitere Gewicht erhöhen sich die Vibrationen nochmals und die maximalen sowie minimalen Werte der meisten Achsen werden im Vergleich zur Unwucht mit einem Gewicht erhöht. Ebenso erhöhen sich erneut die Standardabweichungen.

Tabelle 4.5.: Statistische Kennzahlen Unwucht mit zwei Gewichten

Statistische Kennzahl	**ax**	**ay**	**az**	**aT**
Mittelwert	0,01	0,0006	-0,12	1,88
Standardabweichung	0,87	1,31	1,29	0,78
Minimal Wert	-3,12	-3,78	-4,91	0,08
25 % Quantil	-0,59	-0,91	-0,97	1,30
50 % Quantil	-0,01	0,02	-0,05	1,84
75 % Quantil	0,57	1,03	0,76	2,41
Maximal Wert	3,75	3,52	3,82	4,95

Die Mittelwerte sowie der Median der ax, ay und az Sensoren der verschiedenen Zustände nehmen ähnliche Werte zwischen -0,15 und 0,02 an, weshalb sich diese Kennzahlen wahrscheinlich nicht als aussagekräftige Merkmale zur Unterscheidung der Zustände verwenden lassen.

Die Boxplotdiagramme stellen jeweils alle vier Zustände, die wie folgt abgebildet sind:

1. Zustand: Normalzustand,
2. Zustand: Holzsockel,
3. Zustand: Unwucht mit einem Gewicht und
4. Zustand: Unwucht mit zwei Gewichten

für jede Raumachse und die Gesamtbeschleunigung dar.

Die Standardabweichungen der verschiedenen Zustände unterscheiden sich, bis auf die Zustände zwei und drei des Sensors ax und Zustand drei und vier des Sensors az, deutlich hinsichtlich der Standardabweichung. Das führt dazu, dass die 25 % und 75 % Quantile in den Boxplot Diagrammen 4.1, 4.2, 4.3 und 4.4 weiter auseinander liegen. Die Mediane der Beschleunigungen in den jeweiligen Raumebenen unterscheiden sich nur gering und liegen in den Diagrammen jeweils nahe des Werts null. Ebenso zeigen die Diagramme, dass die Streuung durch das Anbringen von zusätzlichen Gewichten steigt.

In den in Abbildung 4.1 dargestellten Boxplot Diagrammen werden die verschiedenen Zustände des ax Sensors aufgezeigt. Darin ist zu sehen, dass der Median jeweils immer nahe des Wertes null liegt. Der gedämpfte Zustand zwei zeigt die geringsten Vibrationen. Das 25 % und 75 % Quantil und die Whisker des ”Normalzustands” und des Zustands drei nehmen ähnliche Werte an, wobei das Maximum und Minimum vom dritten Zustand über beziehungsweise unter dem des Normalzustands liegen. Die Werte des vierten Zustands nehmen in der x-Achse die höchsten beziehungsweise niedrigsten Werte an.

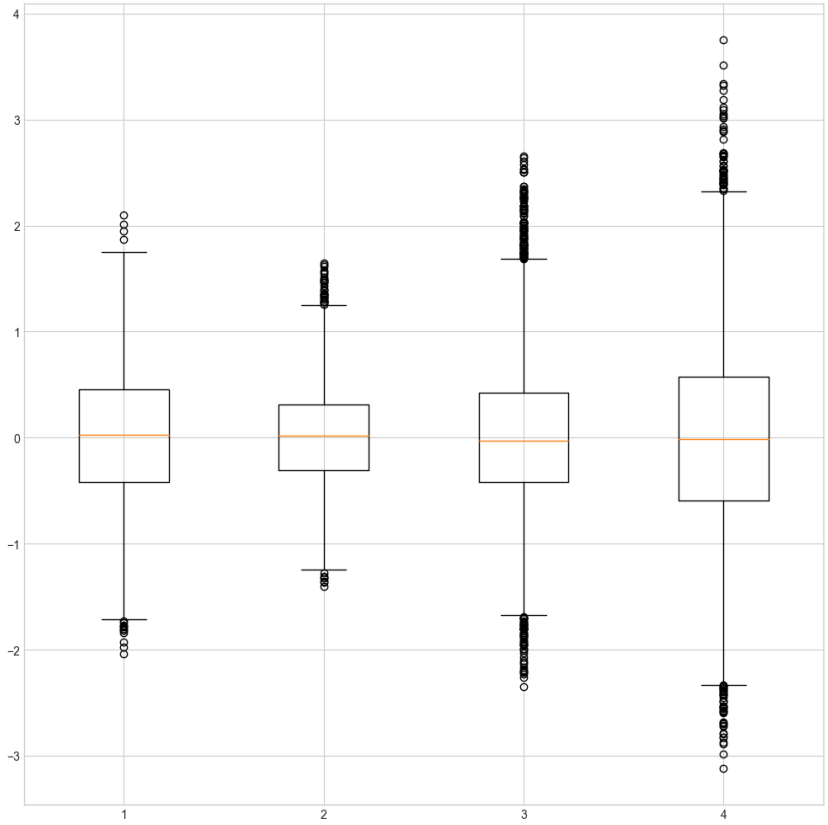

Abbildung 4.1.: Boxplot ax (1: Normalzustand, 2: Holzsockel, 3: Unwucht mit einem Gewicht, 4: Unwucht mit zwei Gewichten)

Die Diagramme in Abbildung 4.2 unterscheiden sich von den Vibrationen in x und z Richtung sowie der Gesamtbeschleunigung insofern, als es bei der Vibration in dieser Ebene keinerlei Ausreißer im Sinne des Boxplotdiagramms gibt. Ansonsten zeigen auch diese Diagramme ein ähnliches Bild wie die Diagramme der anderen Zustände. Die Werte der beiden Fehlerzustände haben eine größere Amplitude als die Werte des ”Normalzustands” und des Zustands ”Holzsockel”.

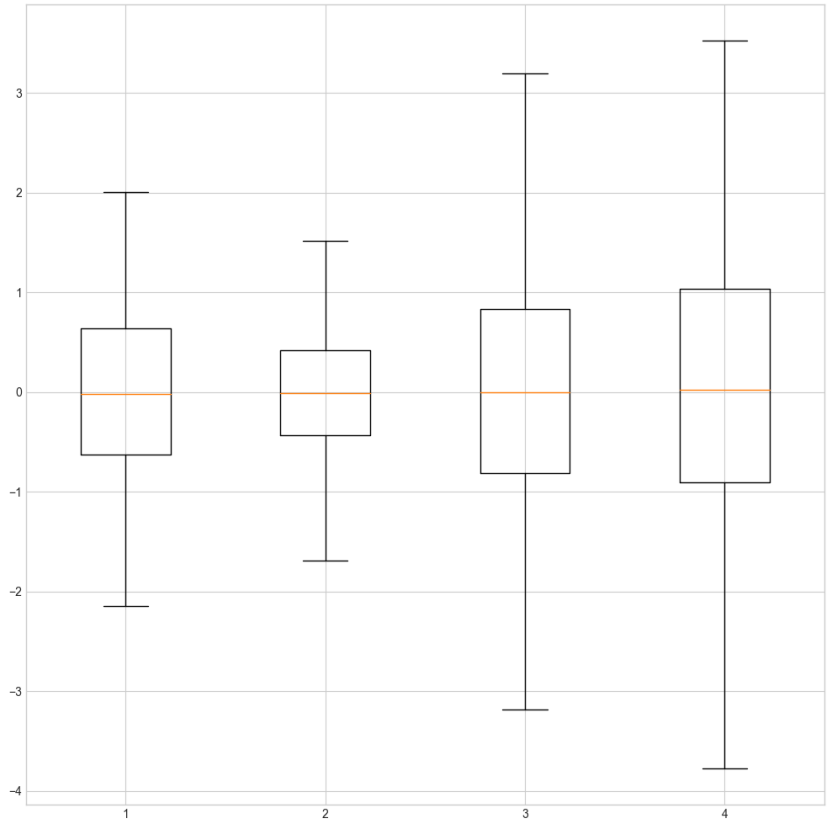

Abbildung 4.2.: Boxplot ay (1: Normalzustand, 2: Holzsockel, 3: Unwucht mit einem Gewicht, 4: Unwucht mit zwei Gewichten)

Während das Maximum des Zustands vier in den anderen Raumachsen den höchsten Wert annimmt, ist in Abbildung 4.3 zu sehen, dass die Werte des Zustands drei in der z-Ebene über denen des Zustands vier liegen. Sonst zeichnet sich in dieser Raumachse ein ähnliches Bild ab, wie bei den anderen.

Die Boxplot Diagramme der Gesamtbeschleunigung in Abbildung 4.4 zeigen, dass der Holzsockel die Vibration dämpft, während das Anbringen von Gewichten dazu führt, dass die Gesamtbeschleunigung gegenüber dem Normalzustand erhöht. Durch die erhöhte Gesamtbeschleunigung liegt der Median des Zustands drei 0,39 und des Zustan-

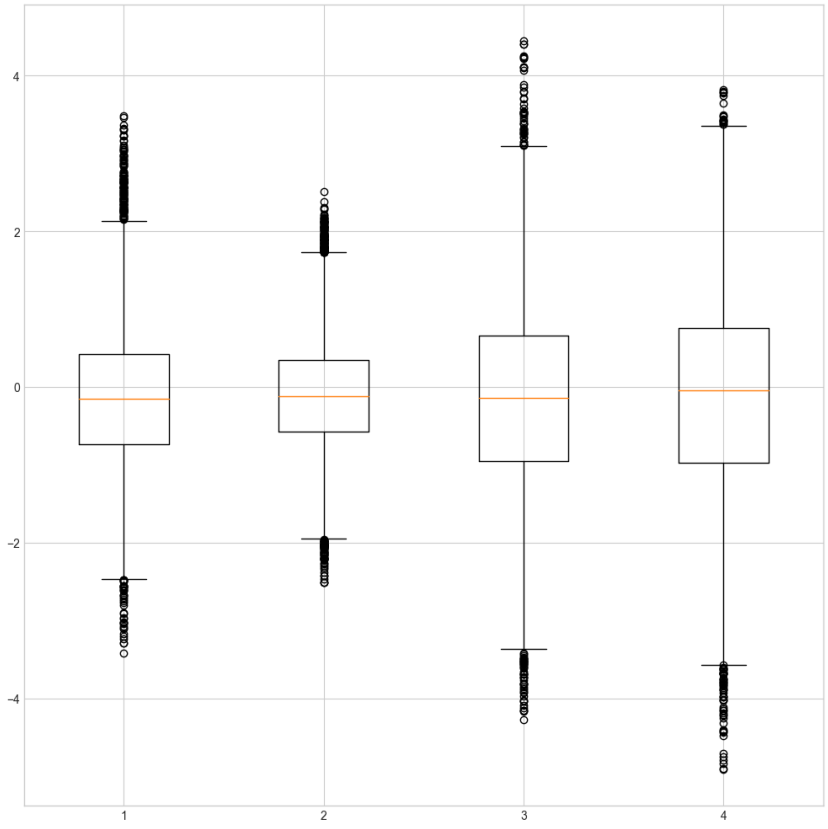

Abbildung 4.3.: Boxplot az (1: Normalzustand, 2: Holzsockel, 3: Unwucht mit einem Gewicht, 4: Unwucht mit zwei Gewichten)

des vier 0,68 über dem des Normalzustands. Durch den Holzsockel wird die Vibration im Mittel auf 0,87 abgesenkt.

Durch die Betrachtung der Boxplot Diagramme und der statistischen Kennzahlen können erste Rückschlüsse auf die Merkmale gezogen werden, die im Weiteren für die Klassifizierung der Zustände mittels maschineller Lernalgorithmen verwendet werden. Dadurch, dass sich der Median der Vibrationen in den Raumebenen nur gering voneinander unterscheidet, bieten diese Werte wahrscheinlich keine gute Grundlage, um die Daten korrekt zu klassifizieren. Bei der Betrachtung der Gesamtbeschleunigung hingegen kann der Median ein Merkmal sein, das für eine Klassifizierung geeignet ist. Ebenso unterscheiden sich die Standardabweichungen der verschiedenen Zustände, wodurch dieses Merkmal wie auch die Varianz zur Unterscheidung der Zustände mittels eines maschinellen Lernalgorithmus angewendet werden kann.

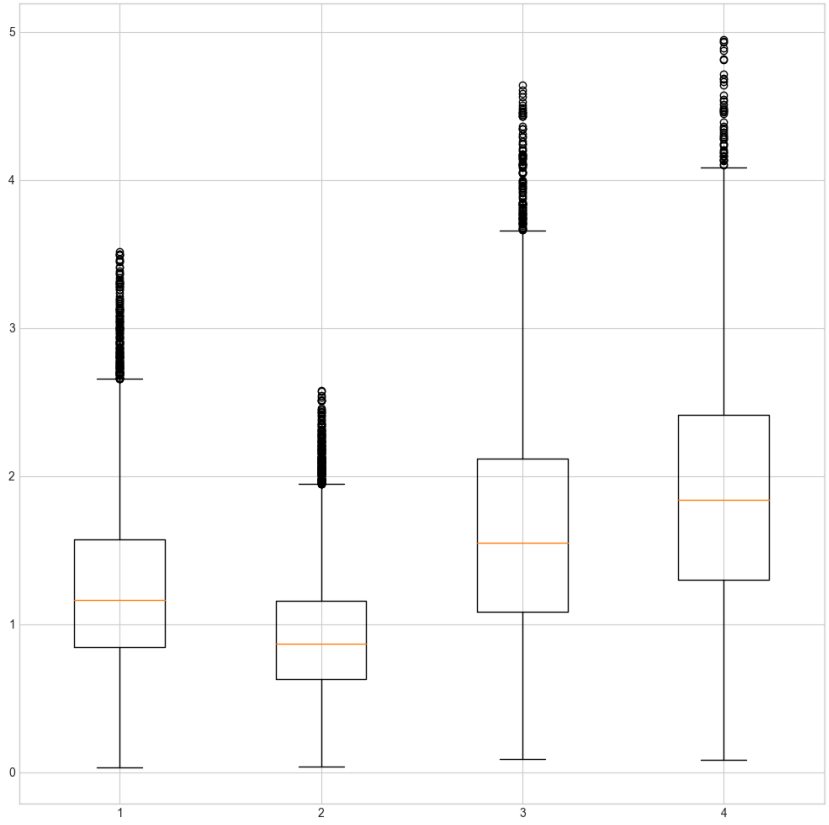

Abbildung 4.4.: Boxplot aT (1: Normalzustand, 2: Holzsockel, 3: Unwucht mit einem Gewicht, 4: Unwucht mit zwei Gewichten)

4.2. Datenvorverarbeitung

Bei der Datenvorverarbeitung werden zunächst alle Ausreißer, die außerhalb des 4-Sigma-Bereichs um den Mittelwert liegen, aus dem Datensatz entfernt. Anschließend werden, wie in Kapitel 3.2.2 beschrieben, Merkmale aus den vorliegenden Daten gebildet. Dazu wird in einem ersten Schritt die Größe der jeweiligen Datensätze angepasst, sodass diese in jeweils gleich große Blöcke, mit der gleichen Anzahl an Datenpunkten zusammengefasst und statistische Merkmale daraus abgeleitet werden können. Die angepasste Verteilung der Datenpunkte für den jeweiligen Maschinenzustand ist in Tabelle 4.6 dargestellt.

Die Daten werden in Blöcken mit jeweils 40 Datenpunkten zusammengefassten. Damit entspricht ein Block einem Zeitintervall von ungefähr 0,4 Sekunden. Die folgenden statistischen Merkmale für jeden Vibrationssensor werden aus den unterteilten Daten ermittelt:

- Mittelwert,

Tabelle 4.6.: Angepasste Verteilung der Datenpunkte im Datensatz

Maschinenzustand	**Anzahl Datenpunkte**
Normalzustand	10760
Holzsockel	6520
Unwucht mit einem Gewicht	9200
Unwucht mit zwei Gewichten	9160
Gesamt	35640

- Maximalwert,
- Minimalwert,
- Median,
- Standardabweichung,
- Wölbung,
- Schiefe,
- Varianz und
- Spannweite.

Daraus ergeben sich insgesamt 36 Merkmale mit jeweils 890 Einträgen. Da die Werte der ermittelten Merkmale nicht stark voneinander abweichen und in einem Wertebereich zwischen -4,91 und 8,74 liegen, wurden diese nicht normalisiert oder standardisiert.

Für das Training sowie den anschließenden Test des erstellten maschinellen Lernmodells wird der Datensatz unterteilt. Die Unterteilung erfolgt in 80 % Trainingsdaten und 20 % Testdaten. Damit die Ergebnisse reproduzierbar sind, wurde ein Random State von 42 verwendet.

Die Korrelationsmatrix in Abbildung 4.5 zeigt, dass die abgeleiteten Merkmale teilweise eine starke lineare Korrelation bezogen auf die Klasse aufweisen. Diese starken Korrelationen können dafür sprechen, dass einige der Merkmale einen ähnlichen Einfluss auf die Ergebnisse des maschinellen Lernmodells haben können und zu keiner wesentlichen Verbesserung der Ergebnisse führen, wenn beide Merkmale eingesetzt werden. Da bei der großen Anzahl von Merkmalen zudem nicht ausgeschlossen werden kann, dass zwischen den Merkmalen und der Klasse eine nicht lineare Korrelation vorliegt, wird für die Merkmalsreduzierung entweder die Hauptkomponentenanalyse, in der Merkmale zusammengefasst werden oder die sequenzielle Vorwärtsauswahl verwendet, welche die besten Merkmale bezüglich des maschinellen Lernalgorithmus auswählt und die anderen verwirft.

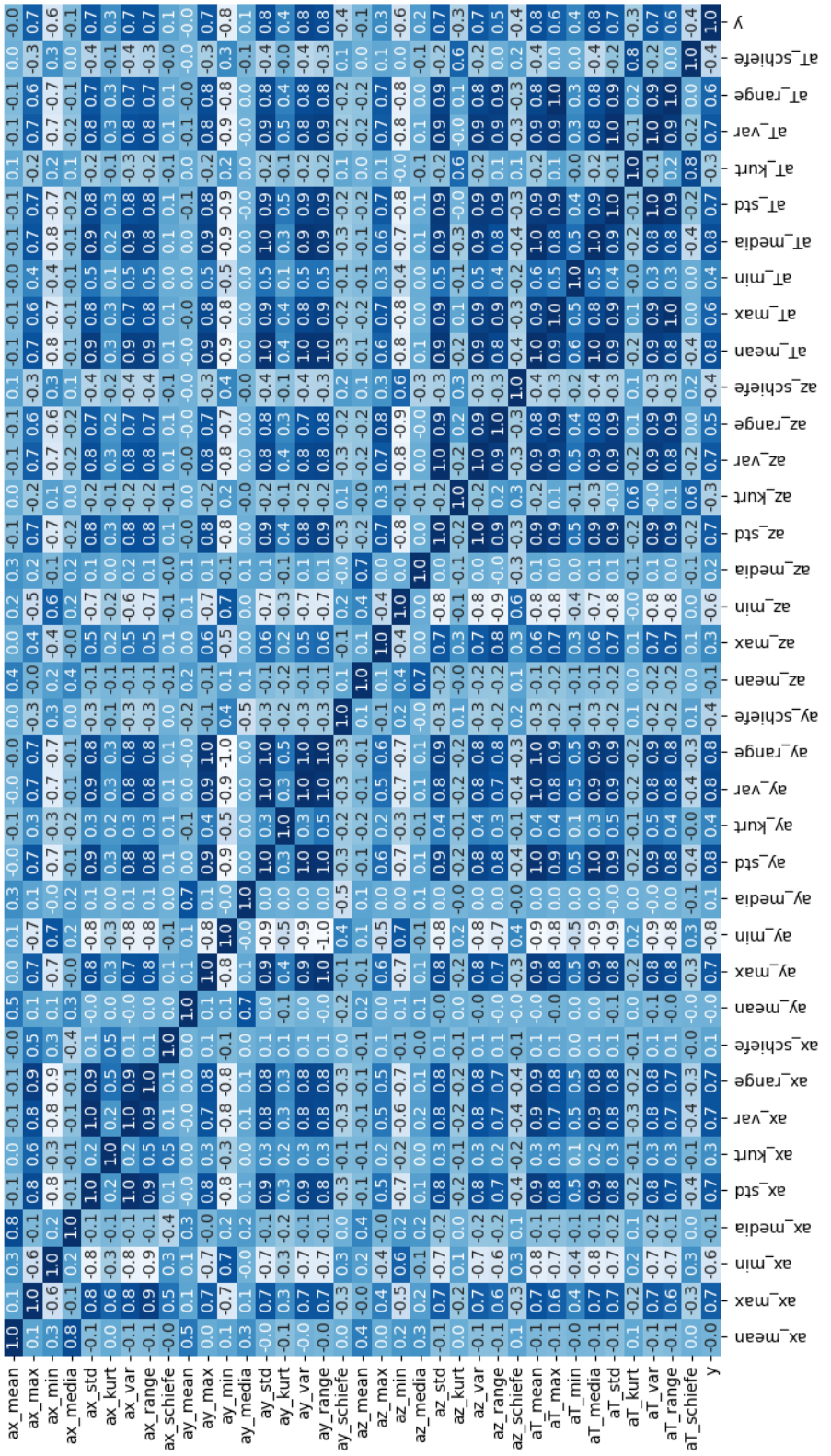

Abbildung 4.5.: Korrelationsmatrix

Zur Auswahl der Methode, welche im späteren Verlauf für die Reduzierung der Merkmale genutzt wird, werden die Klassifizierungsergebnisse des KNN und der logistischen Regression unter der Verwendung der Hauptkomponentenanalyse sowie der sequenziellen Vorwärtsauswahl miteinander verglichen. Das beste Ergebnis, das die Hauptkomponentenanalyse für den KNN erreicht, ist eine Genauigkeit der Trainingsdaten von 96,07 % und eine Genauigkeit bei der Klassifizierung der Testdaten von 94,38 %. Ebenfalls bei dem KNN ist das beste Ergebnis der sequenziellen Vorwärtsauswahl eine Genauigkeit der Trainingsdaten von 97,33 % und eine Genauigkeit der Testdaten von 97,19 %. Bei der logistischen Regression erzielt ebenfalls die sequenzielle Vorwärtsauswahl mit einem Ergebnis von 97,19 % für die Trainings- und Testdaten bessere Klassifizierungsergebnisse. Die logistische Regression erreicht mit der Datenreduzierung der Hauptkomponentenanalyse eine Genauigkeit von 96,77 % für die Trainingsdaten und 96,63 % für die Testdaten. Damit wird im Folgenden die sequenzielle Vorwärtsauswahl für die Merkmalsreduzierung verwendet.

Um die bestmögliche Genauigkeit der einzelnen Lernalgorithmen zu erreichen, werden mehrere Durchläufe zur Identifizierung der Merkmale vorgenommen, welche am besten für die Klassifikation geeignet sind. Die Merkmale werden mit einer sequenziellen Vorwärtsauswahl ermittelt. Die Funktion SequentialFeatureSelector von scikit-learn führt bei der Ermittlung der Merkmale gleichzeitig eine Kreuzvalidierung mit fünf Unterdatensätzen durch, um zu prüfen, ob das Modell mit den ausgewählten Merkmalen generalisierbar ist. Dazu werden die Datenpunkte der Trainings- und Testdatensätze bei jedem Durchlauf gleich aufgeteilt, um den Einfluss der Anzahl der Merkmale auf die Klassifikation vergleichbar zu machen.

4.3. Datenanalyse

Im Zuge der Datenanalyse werden die Parameter identifiziert, mit denen die verschiedenen maschinellen Lernalgorithmen die besten Klassifizierungsergebnisse liefern. Die Parameter, welche in diesem iterativen Verfahren angepasst werden, sind die Anzahl der Merkmale und die Merkmale, welche durch die sequenzielle Vorwärtsauswahl ermittelt werden. Zusätzlich wird die maximale Größe beim Entscheidungsbaum sowie beim Random Forest variiert, um die bestmögliche Klassifizierungsgenauigkeit zu erreichen, ohne

dass die Algorithmen über- oder unterangepasst sind. Abschließend werden die Merkmale sowie der Aufbau des neuronalen Netzes angepasst.

In Tabelle 4.7 ist die Genauigkeit des KNN für eine steigende Anzahl von Merkmalen, die während der Klassifizierung berücksichtigt werden, aufgelistet. Bei der Ermittlung der Anzahl an Merkmalen, wurde der in sk-learn hinterlegte Standardwert von $k = 5$ verwendet.

Tabelle 4.7.: Vergleich der Anzahl der Merkmale auf die Genauigkeit der Klassifikation (in Prozent) des K-Nearest Neighbors bei einem Random State von 42

Anzahl der Merkmale	Trainingsdaten	Testdaten
2	96,35	97,19
4	97,05	96,63
5	97,33	97,19
6	97,33	97,13
8	96,91	96,63
10	96,63	96,07
12	96,91	97,19
14	95,93	94,38
18	96,48	94,63
22	96,21	94,38
26	96,91	93,82
30	96,35	94,38
34	96,91	94,94

Der KNN erreicht nach den Ergebnissen in Tabelle 4.7 die besten Klassifizierungsergebnisse mit insgesamt fünf Merkmalen.

Da beim KNN neben der Anzahl der Merkmale, welche für die Klassifizierung verwendet werden, mit dem Parameter k die Anzahl der Nachbarn angepasst werden kann, welche bei der Mehrheitsentscheidung berücksichtigt werden, wurden in Tabelle 4.8 die Klassifikationsergebnisse für unterschiedliche Parameter von k dargestellt.

Die Ergebnisse in Tabelle 4.8 zeigen, dass die Klassifizierungen für die meisten Werte von k gute Ergebnisse liefern. Die besten Ergebnisse werden allerdings für die Werte drei, fünf und sieben erreicht. Zwar weisen die Ergebnisse für die Klassifizierung von $k = 7$ die geringste Differenz zwischen den Test- und Trainigsdaten auf, doch klassifiziert der KNN mit diesem Parameter einen Normalzustand als Fehler, wohingegen die Fehler in der Klassifikation von $k = 5$ ausschließlich Fehler sind, die der falschen Fehlerkategorie zugeordnet wurden. Daher wird für den Lernalgorithmus weiterhin der Wert von $k = 5$ verwendet.

Tabelle 4.8.: Einfluss unterschiedlicher Werte für den Parameter k auf die Genauigkeit der Klassifikation (in Prozent) des K-Nearest Neighbors bei einem Random State von 42

k	Trainingsdaten	Testdaten
2	97,61	95,51
3	97,75	97,19
4	97,19	95,51
5	97,33	97,19
6	97,33	97,75
7	97,19	97,19
8	96,77	98,31

Die sechs Merkmale, welche durch die sequenzielle Vorwärtsauswahl mit einem Parameter von $k = 5$ identifiziert wurden sind:

- Standardabweichung ax Sensor,
- Varianz ax Sensor,
- Standardabweichung ay Sensor,
- Mittelwert aT und
- Varianz aT.

Mit diesen Merkmalen wird eine Genauigkeit für die Klassifizierung der Trainingsdaten von 97,33 % und 97,19 % für die Klassifizierung des Testdatensatzes erreicht. Die beiden Werte liegen nah beieinander, weshalb davon ausgegangen werden kann, dass das Modell nicht überangepasst ist.

Bei dem Algorithmus der logistischen Regression werden wie zuvor beim KNN die Genauigkeiten der Klassifizierungen der Trainings- und Testdatensätze bei einer ansteigenden Anzahl von Merkmalen miteinander verglichen. Damit kann zum einem die optimale Anzahl der Merkmale ermittelt und zum anderen die Merkmale identifiziert werden, mit denen der Algorithmus die besten Ergebnisse liefert. Die Ergebnisse für die Klassifikationen sind in Tabelle 4.9 dargestellt.

Aus Tabelle 4.9 geht hervor, dass der maschinelle Lernalgorithmus der logistischen Regression die besten Ergebnisse für die insgesamt 25 Merkmale liefert:

- Mittelwert ax Sensor,
- Maximum ax Sensor,
- Median ax Sensor,

Tabelle 4.9.: Vergleich der Anzahl der Merkmale auf die Genauigkeit der Klassifikation (in Prozent) der Logistischen Regression bei einem Random State von 42

Anzahl der Merkmale	Trainingsdaten	Testdaten
2	94,52	95,51
4	95,08	96,63
6	96,21	96,63
8	96,07	96,70
10	96,49	96,63
12	96,91	96,07
14	97,19	96,07
18	97,33	96,07
22	97,05	95,51
25	97,19	97,19
26	96,35	96,63
30	96,91	94,94
34	96,49	93,26

- Standardabweichung ax Sensor,
- Kurtosis ax Sensor,
- Varianz ax Sensor,
- Schiefe ax Sensor,
- Mittelwert ay Sensor,
- Minimum ay Sensor,
- Median ay Sensor,
- Standardabweichung ay Sensor,
- Kurtosis ay Sensor,
- Varianz ay Sensor,
- Spannweite ay Sensor,
- Mittelwert az Sensor,
- Maximum az Sensor,
- Median az Sensor,
- Standardabweichung az Sensor,
- Varianz az Sensor,
- Schiefe az Sensor,
- Mittelwert Gesamtbeschleunigung,
- Minimum Gesamtbeschleunigung,
- Median Gesamtbeschleunigung,
- Kurtosis Gesamtbeschleunigung und

- Varianz Gesamtbeschleunigung

Bei den meisten Modellen liegt die Genauigkeit des Testdatensatzes über dem vom Trainingsdatensatz, was dafür sprechen könnte, dass die guten Ergebnisse zufällig zustande kommen und nicht, weil das Modell gut angepasst wurde. Bei der Verwendung von insgesamt 25 Merkmalen, liegen für den Trainings- und Testdatensatz die gleichen Ergebnisse vor.

Bei dem Entscheidungsbaum wird zur Identifizierung der Parameter, welche zu einem bestmöglichen Klassifizierungsergebnis führen, nicht nur die Anzahl der Merkmale, sondern auch die maximale Tiefe des erstellten Entscheidungsbaums angepasst.

Tabelle 4.10.: Vergleich der Anzahl der Merkmale auf die Genauigkeit der Klassifikation (in Prozent) des Entscheidungsbaums bei einem Random State von 42

Tiefe	Datensatz	Anzahl der Merkmale							
		1	2	3	4	6	8	10	12
2	Training	75,70	90,17	90,17	90,17	90,17	90,17	90,17	90,17
	Test	75,84	88,20	88,20	88,20	88,20	88,20	88,20	88,20
4	Training	91,43	96,35	96,77	97,05	97,05	97,05	97,05	97,47
	Test	89,33	96,63	97,19	96,07	96,07	96,07	96,07	96,07
5	Training	91,71	96,77	98,03	98,03	98,17	98,31	98,31	98,31
	Test	88,20	95,51	97,19	96,63	96,63	97,19	97,19	96,63
6	Training	92,84	97,61	97,61	98,03	98,46	98,46	98,31	98,46
	Test	89,33	99,07	98,82	94,94	96,63	97,19	94,94	96,63
8	Training	94,38	98,88	99,58	99,72	99,72	99,16	99,30	99,30
	Test	89,33	95,51	93,82	93,82	94,94	93,82	94,94	93,82

Aus Tabelle 4.10 geht hervor, dass der Entscheidungsbaum die besten Vorhersageergebnisse mit den drei Merkmalen:

- Varianz ax Sensor,
- Varianz ay Sensor,
- Median Gesamtbeschleunigung

und einer maximalen Tiefe des Baumes von fünf erreicht. Der mit diesen Vorgaben erstellte Entscheidungsbaum ist in Abbildung 4.6 dargestellt.

Der Entscheidungsbaum wurde mittels des Gini-Index erstellt und erreicht für die Trainingsdaten eine fehlerfreie Klassifikation der Klassen ”Normalzustand” und ”Holzsockel”. Lediglich die Fehlerklassen ”Unwucht mit einem Gewicht” und ”Unwucht mit zwei Gewichten” werden nicht vollständig voneinander getrennt, da eine weitere Unterteilung der

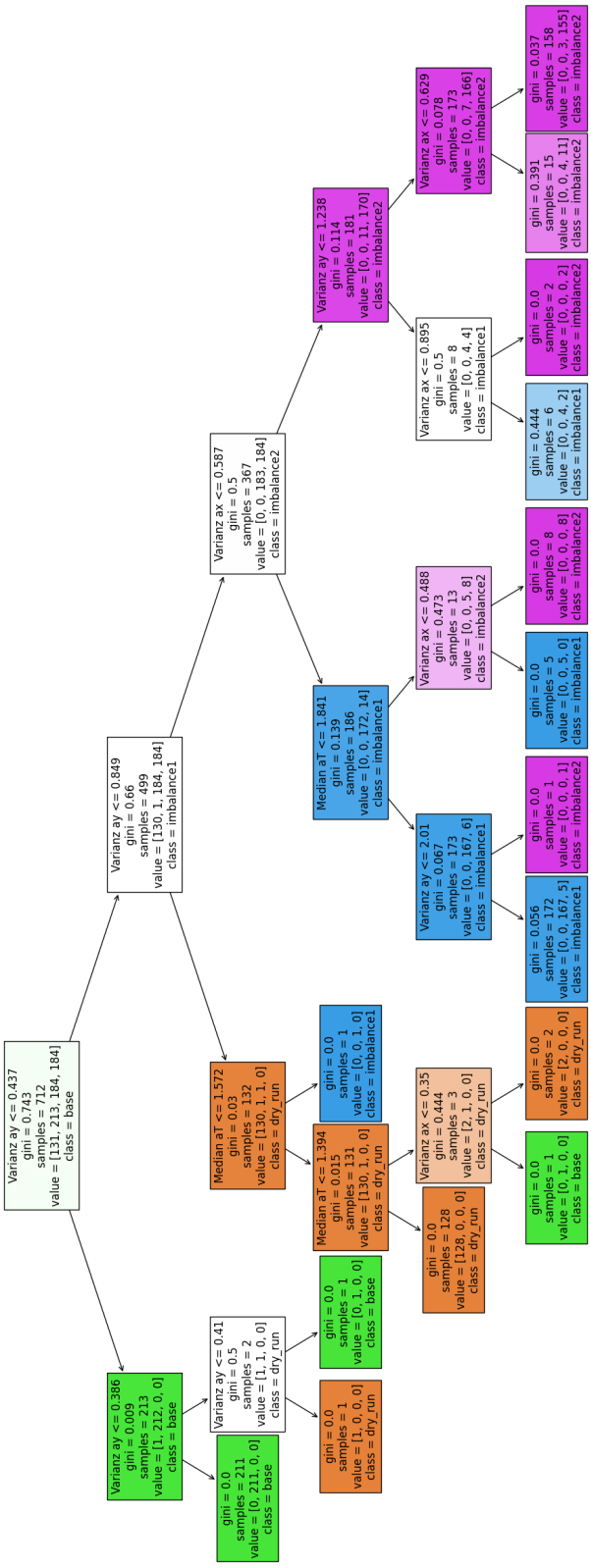

Abbildung 4.6.: Entscheidungsbaum

Merkmale zu einer Überanpassung des Entscheidungsbaums führen würde, die sich negativ auf die Klassifizierungsergebnisse der Testdaten auswirkt.

Wie zuvor beim Entscheidungsbaum, wird auch beim Random Forest die Anzahl der Merkmale sowie die maximale Tiefe der verschiedenen Bäume variiert, um die bestmöglichen Parameter für die Klassifizierung zu identifizieren. Tabelle 4.11 stellt die Ergebnisse für die unterschiedlichen Tiefen und Merkmale des Random Forests dar. Die verschiedenen Modelle des Random Forest werden ebenfalls mittels des Gini-Index erstellt. Dabei erstellt diese Ensemble Methode insgesamt 100 verschiedene Entscheidungsbäume, aus denen durch eine Mehrheitsentscheidung die Eingabewerte einer Klasse zugeordnet werden.

Tabelle 4.11.: Vergleich der Anzahl der Merkmale auf die Genauigkeit der Klassifikation (in Prozent) des Random Forest bei einem Random State von 42

Tiefe	Datensatz	Anzahl der Merkmale						
		2	4	5	6	8	10	12
2	Training	94,24	95,93	95,79	95,79	95,79	96,07	95,79
	Test	94,38	95,51	95,51	95,51	95,51	95,51	95,51
4	Training	96,63	96,77	97,05	96,91	96,91	97,47	97,19
	Test	96,63	96,07	97,75	96,63	97,75	97,75	97,75
6	Training	98,03	98,60	98,76	98,17	99,02	98,46	98,88
	Test	97,75	97,19	98,31	96,63	97,75	97,75	96,63
8	Training	99,16	99,58	100	99,58	99,58	99,86	99,58
	Test	97,75	95,51	97,75	97,19	97,75	96,63	96,63

Die besten Ergebnisse werden mit den insgesamt fünf Merkmalen:

- Kurtosis ax Sensor,
- Varianz ax Sensor,
- Schiefe ax Sensor,
- Varianz ay Sensor,
- Median Gesamtbeschleunigung

und einer maximalen Tiefe der einzelnen Bäume von sechs erreicht. Werden weitere Merkmale oder tiefere Bäume erstellt, passt sich das Modell zu sehr an die Trainingsdaten an, was dazu führt, dass das Modell überangepasst ist.

Das SVM Modell für mehrere Klassen wird mittels der Strategie ”Einer gegen alle” erstellt. Bei diesem Algorithmus wird, wie in Tabelle 4.12 dargestellt ist, die Anzahl der verwendeten Merkmale variiert, um die optimalen Merkmale für die Klassifizierung zu

identifizieren.

Tabelle 4.12.: Vergleich der Anzahl der Merkmale auf die Genauigkeit der Klassifikation (in Prozent) der SVM bei einem Random State von 42

Anzahl der Merkmale	Trainingsdaten	Testdaten
2	96,07	97,19
4	96,63	97,75
5	96,77	97,75
6	96,77	97,75
7	96,63	98,31
8	96,63	98,31
10	97,06	98,31
12	96,49	98,31
14	96,35	97,19
18	96,21	96,63
22	96,21	96,07
26	97,05	97,75
30	96,77	95,07
34	96,91	96,07

Aus den Einträgen in Tabelle 4.12 geht hervor, dass die SVM in vielen Fällen gute Klassifikationsergebnisse liefert. Allerdings liegt bei den meisten Ergebnissen die Genauigkeit der Testergebnisse über der der Trainingsergebnisse, was dafür sprechen kann, dass die Ergebnisse auf einem Zufall beruhen und nicht dadurch zustanden gekommen sind, dass der Lernalgorithmus entsprechend gut trainiert ist. Um zu prüfen, ob der Algorithmus bei anderen Trainings- und Testdaten zu ähnlichen Ergebnissen kommt, werden die jeweiligen Eingabedaten für die Anzahl der Merkmale, welche zu den besten Ergebnissen gekommen sind, erneut mit zufälligen Daten trainiert. In Tabelle 4.13 sind die Ergebnisse der Test- und Trainingsdaten für unterschiedliche Random States, bei der Verwendung von insgesamt 10 Merkmalen, aufgeführt.

Die in Tabelle 4.13 aufgelisteten Ergebnisse zeigen, dass in den meisten Fällen die Klassifizierungsgenauigkeit der Trainingsdaten besser ist, als die der Testdaten. Deshalb wird für den weiteren Verlauf davon ausgegangen, dass die Ergebnisse der Testdaten aus Tabelle 4.12 nicht zufällig entstanden sind.

Die 10 Merkmale, welche für die Bildung des SVM Modells verwendet werden, sind:

- Mittelwert ax Sensor,
- Median ax Sensor,
- Standardabweichung ax Sensor,

Tabelle 4.13.: Variation des Random States für die Trainings- und Testdaten der SVM für 10 Merkmale

Random State	Ergebnis Trainingsdaten	Ergebnis Testdaten
10	97,19	96,63
20	97,05	94,94
23	97,19	97,19
30	97,33	97,19
40	97,47	96,07
50	97,19	96,07
60	97,19	97,19
70	97,05	96,07
80	97,33	96,07
90	96,63	97,75
100	96,91	98,88

- Varianz ax Sensor,
- Median ay Sensor,
- Standardabweichung ay Sensor,
- Median az Sensor,
- Minimum Gesamtbeschleunigung,
- Median Gesamtbeschleunigung und
- Varianz Gesamtbeschleunigung.

Um zu identifizieren, welches neuronale Netz gut für die Klassifizierung der vier Zustände geeignet ist, wurde zunächst ein Netz erstellt, welches aus einer Eingabeschicht besteht, die der Größe der verwendeten Merkmale entspricht. Nach der Eingabeschicht wurde eine verdeckte Schicht mit acht Neuronen eingefügt. Die Ausgabeschicht besteht aus vier Neuronen, welche die vier Zustände widerspiegeln. Als Aktivierungsfunktion für die verdeckten Schichten wird die ReLu-Funktion und für die Optimierung durch das Gradientenabstiegsverfahren das ADAM-Verfahren mit einer Lernrate von 0,001 verwendet.

In Tabelle 4.14 sind die Ergebnisse des ersten neuronalen Netzes dargestellt.

Aus der Tabelle 4.14 ergibt sich, dass sich die Ergebnisse des Testdatensatzes ab 19 Merkmalen deutlich verschlechtern. Mit 14 Merkmalen weist das neuronale Netz die besten Ergebnisse auf und klassifiziert 97,19 % der Testdaten sowie 97,61 % der Trainingsdaten korrekt.

Um zu überprüfen, ob sich eine Verringerung oder Erhöhung der Neuronen sowie eine weitere verdeckte Schicht positiv auf das Klassifikationsergebnis auswirken, wur-

Tabelle 4.14.: Vergleich der Anzahl der Merkmale auf die Genauigkeit der Klassifikation (in Prozent) des Neuronalen Netzes mit einer verdeckten Schicht, die acht Neuronen enthält, bei einem Random State von 42

Anzahl der Merkmale	Trainingsdaten	Testdaten
2	96,07	97,19
4	96,49	98,31
5	97,05	97,19
6	96,91	96,07
8	97,47	95,51
10	97,33	98,31
12	97,47	96,63
14	97,61	97,19
16	97,47	95,51
18	97,61	95,51
19	96,91	92,13
20	97,89	92,70
22	98,31	93,82
24	98,03	92,70
30	98,17	96,07
34	98,17	93,82

den Netze erstellt, die jeweils eine verdeckte Schicht mit sechs, sieben, neun und zehn Neuronen haben. Außerdem wurde ein neuronales Netz erstellt, das neben einer verdeckten Schicht mit acht Neuronen eine verdeckte Schicht mit sechs Neuronen hat. Da in dem ersten neuronalen Netz, welches erstellt wurde, ab dem 19 Merkmal eine Überanpassung vorliegt, wurden für die weiteren Netze maximal 20 Merkmale identifiziert. Die vollständigen Ergebnisse der einzelnen neuronalen Netze sind in Anhang B enthalten.

Tabelle 4.15 stellt die jeweils besten Klassifikationsergebnisse der verschiedenen neuronalen Netze dar.

Tabelle 4.15.: Vergleich der Ergebnisse der Klassifikation der verschiedenen neuronalen Netze bei einem Random State von 42 (in Prozent)

Anzahl der Neuronen in der verdeckten Schicht	Anzahl der Merkmale	Trainingsdaten	Testdaten
6	9	97,89	97,19
7	5	97,47	97,19
8	14	97,61	97,19
9	10	97,89	97,75
10	9	97,47	97,19
8:6	15	97,75	97,75

Wie Tabelle 4.15 zeigt, erreichen alle neuronalen Netze ähnliche Ergebnisse bei der

Klassifizierung der Zustände. Lediglich das neuronale Netz mit neun Neuronen in der verdeckten Schicht und das Netz, welches aus den verdeckten Schichten mit acht und sechs Neuronen besteht, erreichen mit einer Genauigkeit von 97,75 % etwas bessere Ergebnisse. Weil das neuronale Netz mit neun Neuronen in den Trainingsdaten bessere Ergebnisse liefert, wurde dieses Netz mit den folgenden Merkmalen:

- Standardabweichung ax Sensor,
- Varianz ax Sensor,
- Schiefe ax Sensor,
- Standardabweichung ay Sensor,
- Minimum az Sensor,
- Median az Sensor,
- Schiefe az Sensor,
- Median Gesamtbeschleunigung,
- Standardabweichung Gesamtbeschleunigung und
- Kurtosis Gesamtbeschleunigung

ausgewählt.

4.4. Ergebnisdarstellung und -interpretation

In diesem Abschnitt werden die Ergebnisse der Modelle mit den Parametern und Merkmalen, welche die besten Ergebnissen bezogen auf die Genauigkeit erreicht haben, beschrieben und miteinander verglichen. Damit die Ergebnisse der Modelle verglichen werden können, werden diese wie zuvor in Kapitel 4.3 durch einen Random State anhand der gleichen Trainings- und Testdaten trainiert und bewertet.

Im Folgenden werden die beiden Fehlerklassen ”Unwucht mit einem Gewicht” und ”Unwucht mit zwei Gewichten” als ”Fehler 1” und ”Fehler 2” bezeichnet.

Da es bei der prädiktiven Instandhaltung um das Erkennen eins Fehlers geht, bevor dieser zu einem Ausfall führt, woraufhin eine Instandsetzung geplant und in einer Produktionspause durchgeführt wird, ist es entscheidend, dass der Zustand sowie der vorliegende Fehler richtig klassifiziert werden. Sollte ein Algorithmus einen Fehlerzustand aufzeigen, der nicht vorliegt, würde das dazu führen, dass Instandhaltungsarbeiten vorbereitet

und durchgeführt werden, die nicht notwendig gewesen wären. Dadurch entstehen wie bei der präventiven Instandhalten höhere Kosten.

In der Konfusionsmatrix 4.16 des KNNs ist zu erkennen, dass dieser Algorithmus die Klassen ”Normalzustand” und ”Holzsockel” fehlerfrei klassifiziert. Lediglich die beiden Fehlerklassen ”Fehler 1” und Fehler 2” werden nicht immer korrekt klassifiziert.

Tabelle 4.16.: Konfusionsmatrix KNN

Klassifikation / Tatsächlich	**Normal-zustand**	**Holzsockel**	**Fehler 1**	**Fehler 2**	**Summe**
Normalzustand	32	0	0	0	32
Holzsockel	0	56	0	0	56
Fehler 1	0	0	43	3	46
Fehler 2	0	0	2	42	44
Summe	32	56	45	45	178

Das die beiden Klassen ”Normalzustand” und ”Holzsockel” korrekt klassifiziert wurden, spiegelt sich in den Klassifikationsergebnissen in Tabelle 4.17 wieder. Aufgrund der falschen Zuordnungen der Fehlerklassen ergibt sich für den ”Fehler 1” ein F_1-Score von 0,95 und für den ”Fehler 2” ein Score von 0,94.

Tabelle 4.17.: Klassifikationsergebnis KNN

Ergebnis / Zustand	**Relevanz**	**Sensitivität**	**F1-Score**
Normalzustand	1	1	1
Fehler 1	1	1	1
Fehler 2	0,96	0,93	0,95
Fehler 3	0,93	0,95	0,94

Die Zuordnungen der Klassifikation der logistischen Regression, welche in Tabelle 4.18 dargestellt sind, zeigen, dass der Zustand ”Holzsockel” fehlerfrei klassifiziert wurde. Allerdings wurde ein Fehler dem ”Normalzustand” zugeordnet und damit nicht erkannt. Dies führt, wie in Tabelle 4.19 dargestellt, zu einer Relevanz von 0,97 und einem F_1-Score von 0,98 für die Klasse ”Normalzustand”. Darüber hinaus wurden die Fehlerklassen nicht immer der Klasse zugeordnet, die korrekt gewesen wäre. Damit ergibt sich ein F_1-Score für die Klassen ”Fehler 1” und ”Fehler 2” von 0,95.

Wie in der Konfusionsmatrix 4.20 des Entscheidungsbaums dargestellt ist, klassifiziert dieser Algorithmus die Klassen ”Normalzustand” und ”Holzsockel” korrekt. Der Zustand ”Fehler 1” wurde aber vier mal fehlerhaft der Klasse ”Fehler 2” zugeordnet.

Tabelle 4.18.: Konfusionsmatrix Logistische Regression

Klassifikation / Tatsächlich	**Normal-zustand**	**Holzsockel**	**Fehler 1**	**Fehler 2**	**Summe**
Normalzustand	32	0	0	0	32
Fehler 1	0	56	0	0	56
Fehler 2	1	0	43	2	46
Fehler 3	0	0	2	42	44
Summe	33	56	45	44	178

Tabelle 4.19.: Klassifikationsergebnis Logistische Regression

Ergebnis / Zustand	**Relevanz**	**Sensitivität**	**F1-Score**
Normalzustand	0,97	1	0,98
Holzsockel	1	1	1
Fehler 1	0,96	0,93	0,95
Fehler 2	0,95	0,95	0,95

Tabelle 4.20.: Konfusionsmatrix Entscheidungsbaum

Klassifikation / Tatsächlich	**Normal-zustand**	**Holzsockel**	**Fehler 1**	**Fehler 2**	**Summe**
Normalzustand	32	0	0	0	32
Holzsockel	0	56	0	0	56
Fehler 1	0	0	42	4	46
Fehler 2	0	0	1	43	44
Summe	32	56	43	47	178

Aus der Konfusionsmatrix 4.20 ergeben sich die Klassifikationsergebnisse in Tabelle 4.21, die aufzeigen, dass während der F_1-Score für die Klassen ”Normalzustand” und ”Holzsockel” eins ist, für die Klasse ”Fehler 1” ein F_1-Score von 0,94 und für den ”Fehler 2” ein F_1-Score von 0,95 erreicht wird.

Tabelle 4.21.: Klassifikationsergebnis Entscheidungsbaum

Ergebnis / Zustand	**Relevanz**	**Sensitivität**	**F1-Score**
Normalzustand	1	1	1
Holzsockel	1	1	1
Fehler 1	0,98	0,91	0,94
Fehler 2	0,91	0,98	0,95

Beim Random Forest Algorithmus werden die Klassen ”Normalzustand” und ”Holzsockel”, wie in der Konfusionsmatrix 4.22 dargestellt ist, korrekt zugeordnet. Die Zuord-

nung der Fehlerklassen ist allerdings fehlerhaft, aber im Gegensatz zu den bisherigen Algorithmen wurden mit insgesamt drei falschen Zuweisungen weniger Zustände falsch zugeordnet.

Tabelle 4.22.: Konfusionsmatrix Random Forest

Klassifikation / Tatsächlich	**Normal-zustand**	**Holzsockel**	**Fehler 1**	**Fehler 2**	**Summe**
Normalzustand	32	0	0	0	32
Holzsockel	0	56	0	0	56
Fehler 1	0	0	44	2	46
Fehler 2	0	0	1	43	44
Summe	32	56	45	45	178

Durch die bessere Zuordnung der Zustände zu den Klassen ergibt sich, wie in Tabelle 4.23 zu sehen ist, ein besserer F_1-Score von 0,97 für die Klassen ”Fehler 1” und ”Fehler 2”.

Tabelle 4.23.: Klassifikationsergebnis Random Forest

Ergebnis / Zustand	**Relevanz**	**Sensitivität**	**F1-Score**
Normalzustand	1	1	1
Holzsockel	1	1	1
Fehler 1	0,98	0,96	0,97
Fehler 2	0,96	0,98	0,97

Mittels der SVM ist es, wie in Tabelle 4.24 dargestellt, möglich, die beiden Klassen ”Normalzustand” und ”Holzsockel” ohne einen Fehler zu klassifizieren. Bei den Klassen ”Fehler 1” und ”Fehler 2” kommt es allerdings zu Fehlern, durch die die Daten der Klasse des jeweils anderen Fehlers zugeordnet werden.

Tabelle 4.24.: Konfusionsmatrix SVM

Klassifikation / Tatsächlich	**Normal-zustand**	**Holzsockel**	**Fehler 1**	**Fehler 2**	**Summe**
Normalzustand	32	0	0	0	32
Holzsockel	0	56	0	0	56
Fehler 1	0	0	44	2	46
Fehler 2	0	0	1	43	44
Summe	32	56	45	45	178

Durch die falsche Zuordnung der Fehlerklassen ergibt sich, wie in Tabelle 4.25 aufgelistet ist, für beide Fehlerklassen ein F_1-Score von 0,97. Da die anderen Klassen nicht

falsch klassifiziert oder diesen Klassen fälschlicherweise Daten zugeordnet wurden, die nicht dieser Klasse entsprechen, ergibt sich für diese Klassen ein Score von 1.

Tabelle 4.25.: Klassifikationsergebnis SVM

Ergebnis / Zustand	**Relevanz**	**Sensitivität**	**F1-Score**
Normalzustand	1	1	1
Holzsockel	1	1	1
Fehler 1	0,98	0,96	0,97
Fehler 2	0,96	0,98	0,97

Die Konfusionsmatrix 4.26 des neuronalen Netzes zeigt, dass die Zustände ”Normalzustand” und ”Holzsockel” richtig erfasst werden. Allerdings kommt es auch bei diesem Modell zu fehlerhaften Zuordnungen bei den den Zuständen ”Fehler 1 und ”Fehler 2”.

Tabelle 4.26.: Konfusionsmatrix neuronales Netz

Klassifikation / Tatsächlich	**Normal-zustand**	**Holzsockel**	**Fehler 1**	**Fehler 2**	**Summe**
Normalzustand	32	0	0	0	32
Holzsockel	0	56	0	0	56
Fehler 1	0	0	44	2	46
Fehler 2	0	0	2	42	44
Summe	32	56	46	44	178

Das Modell erreicht, wie in Tabelle 4.27 dargestellt, ist für die Klassen ”Fehler 1” einen F_1-Score von 0,96 und ”Fehler 2” einen F_1-Score von 0,95. Diese Werte kommen dadurch Zustande, dass jeweils zwei Fehler fälschlicherweise der anderen Fehlerklasse zugeordnet wurden.

Tabelle 4.27.: Klassifikationsergebnis neuronales Netz

Ergebnis / Zustand	**Relevanz**	**Sensitivität**	**F1-Score**
Normalzustand	1	1	1
Holzsockel	1	1	1
Fehler 1	0,96	0,96	0,96
Fehler 2	0,95	0,95	0,95

Tabelle 4.28 fasst abschließend die besten Ergebnisse der verschiedenen maschinellen Lernalgorithmen zusammen. Dabei werden nicht nur die Klassifikationsergebnisse betrachtet, sondern auch, ob die Algorithmen den Normalzustand und den gedämpften

Zustand von den Fehlerzuständen unterscheiden kann. Die Klassifizierung eines Normalzustands als Fehlerzustand würde dazu führen, dass eine Reparatur veranlasst wird und Ausfallzeiten entstehen die nicht notwendig gewesen wären.

Tabelle 4.28.: Verglich der Klassifikationergebnisse

maschineller Lernalgorithmus	Anzahl der Merkmale	Ergebnisse Trainingsdaten	Ergebnisse Testdaten	Korrekte Trennung
KNN	5	97,33	97,19	Ja
logistische Regression	25	97,19	97,19	Nein
Entscheidungsbaum	6	98,46	97,19	Ja
Random Forest	8	98,60	98,31	Ja
SVM	10	97,06	98,31	Ja
neuronales Netz	10	97,89	97,75	Ja

Wie aus Tabelle 4.28 hervorgeht, erreichen alle maschinellen Lernalgorithmen mit Genauigkeiten von 97,19 % bis 98,31 % gute Ergebnisse bei der Klassifizierung der Zustände. In den meisten Fällen werden ebenfalls die Normalzustände korrekt von den Fehlerzuständen unterschieden. Lediglich bei der logistischen Regression werden Normal- und Fehlerzustände miteinander vermischt. Ansonsten erreichen die Algorithmen Random Forest und SVM mit 98,31 %, gefolgt von dem neuronalen Netz, welches eine Genauigkeit von 97,89 % hat, die besten Ergebnisse. Die anderen Algorithmen erreichen Genauigkeiten von 97,19 %.

5. Diskussion

Mit den in Kapitel 4.4 verwendeten Methoden zur Datenvorbereitung und -analyse wurden die Zustände der Standbohrmaschine bei den meisten maschinellen Lernalgorithmen richtig klassifiziert. Besonders die Modelle, welche mit den Algorithmen Random Forest oder SVM erstellt wurden, lieferten mit einer Klassifizierungsgenauigkeit von 98,31 % gute Ergebnisse. Diesen Algorithmen gelang es zudem, die beiden Zustände, in denen kein Defekt vorliegt, immer richtig zu klassifizieren. Lediglich bei den beiden Fehlerzuständen kam es zu falschen Klassifizierungen. Somit kann davon ausgegangen werden, dass immer dann, wenn ein Fehlerzustand klassifiziert wird, auch ein Fehler vorliegt und keine Wartung an einer Maschine vorgenommen wird, bei der kein Fehler vorliegt. Auch das neuronale Netz erreicht ebenfalls gute Ergebnisse bei der Klassifizierung der Zustände und klassifiziert 97,75 % der Zustände richtig. Darüber hinaus erfolgt auch eine korrekte Trennung der Normal- und Fehlerzustände. Neben den zuvor genannten Algorithmen erreichen die Modelle der logistische Regression, des KNNs sowie des Entscheidungsbaums mit einer Klassifizierungsgenauigkeit von 97,19 % gute Ergebnisse. Doch während der Entscheidungsbaum und KNN die Zustände, in denen kein Defekt vorliegt, von den Fehlerzuständen sauber trennt, gelingt dies der logistischen Regression nicht. Dadurch wurde ein Fehlerzustand fälschlicherweise als Normalzustand klassifiziert.

Mittels der meisten in Kapitel 4.4 verwendeten Methoden ist es möglich, die vorliegenden Fehlerzustände mit einer hohen Genauigkeit zu klassifizieren. Es kommt zwar zwischen den Zuständen ”Unwucht mit einem Gewicht” und ” Unwucht mit zwei Gewichten” zu einigen fehlerhaften Klassifikationen, dennoch wird in den meisten Fällen kein Fehler angezeigt, wenn gar kein Fehler vorliegt.

Diese Arbeit behandelt konkret die Klassifizierung von Fehlerzuständen anhand eines vorliegenden Beispieldatensatzes. Da jedes Klassifizierungsproblem andere Herausforderungen darstellt, welche mit anderen Methoden gelöst werden müssen, lässt sich der in dieser Arbeit erstellt Code nicht ohne Anpassungen auf andere Probleme übertragen. Außerdem wurde durch die Betrachtung des vorliegenden Problems klar, dass im Rahmen der Datenvorbereitung und -analyse nicht alle im Grundlagenteil dargestellten Methoden angewendet werden konnten.

Aufgrund der geringen Datenmenge wurden die Daten in Blöcke mit jeweils 40 Da-

ten zusammengefasst, anhand derer die Merkmale für die spätere Klassifikation gebildet wurden. Damit bilden die Daten einen Zeitraum von $0,4$ Sekunden ab. Werden die Algorithmen zur Klassifizierung von Zuständen in einem System verwendet, welches kontinuierlich Daten liefert, könnten vielleicht bessere Ergebnisse erzielt werden. Diese Verbesserung der Ergebnisse könnte zum einem durch eine größere Menge an Trainingsdaten oder zum anderen durch die Zusammenfassung von Daten in größere Blöcke erreicht werden. Da es sich bei den aus den Datenblöcken gewonnenen Merkmale um Kennzahlen handelt, die den jeweiligen Block beschreiben, werden diese aussagekräftiger, je größer ein Block ist.

6. Fazit

In dieser Masterarbeit wurden die Vibrationsdaten einer Standbohrmaschine analysiert. Diese Daten waren in die vier Zustände:

- Normalzustand,
- Befestigung am einem Sockel,
- Unwucht mit einem Gewicht,
- Unwucht mit zwei Gewichten

unterteilt. In einem ersten Schritt wurden die vorliegenden Daten in einer explorativen Datenanalyse untersucht. Dabei wurden Kennzahlen wie der Mittelwert, Standardabweichung sowie das Minimum und Maximum betrachtet. Des Weiteren wurden die Datensätze in Boxplotdiagrammen visualisiert. Aus der ersten Analyse hat sich ergeben, dass sich die Datensätze bei der Standardabweichung sowie dem Median und Mittelwert bei den Daten der Gesamtschwingung unterscheiden. Im nächsten Schritt wurden die Daten bereinigt und Ausreißer, welche außerhalb des vier Sigma Bereichs, liegen aus den Datensätzen entfernt. Anschließend wurden die Datensätze so gekürzt, dass sich diese in Blöcke mit je 40 Datenpunkte unterteilen ließen, ohne dass ein Datenblock zwei verschiedenen Zustände enthält. Aus den Datenblöcken wurden dann die Merkmale:

- Mittelwert,
- Maximalwert,
- Minimalwert,
- Media,
- Standardabweichung,
- Wölbung,
- Schiefe,
- Varianz und
- Spannweite

für die Erstellung der maschinellen Lernalgorithmen gebildet. Da insgesamt 36 Merkmale erstellt wurden und einige von diesen Merkmalen eine starke Korrelation mit der Zielgröße aufweisen, wurde davon ausgegangen, dass nicht alle Merkmale einen großen Beitrag für die spätere Klassifikation haben und die Merkmale somit reduziert werden

können, ohne das Ergebnis zu beeinträchtigen. Zur Merkmalsreduktion wurden die zwei Methoden Hauptkomponentenanalyse und sequenzielle Vorwärtsauswahl in Betracht gezogen. Da die sequenzielle Vorwärtsauswahl im Vergleich zur Hauptkomponentenanalyse bei dem KNN und der logistischen Regression bessere Ergebnisse lieferte, wurden die Merkmale für alle Algorithmen mit dieser Methode ermittelt. Abschließend wurden Merkmale für die Algorithmen:

- KNN,
- logistische Regression,
- Entscheidungsbaum,
- Random Forest,
- SVM und
- neuronales Netz

ausgewählt und Modelle zur Klassifizierung der Zustände erstellt. Anschließend wurden die Parameter für die einzelnen Algorithmen variiert um die bestmöglichen Ergebnisse für die Klassifizierung zu erhalten. Abschließend wurden die Ergebnisse des Testdatensatzes mittels Konfusionsmatrizen dargestellt und interpretiert.

Diese Masterarbeit hat gezeigt, dass es mittels maschineller Lernalgorithmen möglich ist, Fehlerzustände mit einer hohen Genauigkeit zu erkennen. Dabei erreichten die Algorithmen Genauigkeiten von bis zu 98,31 % bei den Testdaten. Am besten hat die Ensemble Methode Random Forest abgeschnitten, welche mit acht Merkmalen ein Modell erstellen kann, das die Zustände der Maschine korrekt zuweist. Aber auch die anderen Methoden erzielten bei den Testdaten Genauigkeiten von 97,19 % und das neuronale Netz weist eine Genauigkeit von 97,75 % auf.

Weiterführend könnten die Erkenntnisse dieser Arbeit bei einem konkreten Anwendungsfall eingesetzt werden, um zu prüfen, wie sich die Klassifizierung des Zustands in der Praxis verhält und ob damit zuverlässig Fehlerzustände erkannt und behoben werden können.

Literaturverzeichnis

[1] Bitkom, “Investition in industrie 4.0 in deutschland in den jahren 2013 bis 2020 (in milliarden euro),” 25.07.2014. [Online]. Available: https://de.statista.com/statistik/daten/studie/372846/umfrage/investition-in-industrie-40-in-deutschland/

[2] W. Bauer, S. Schlund, D. Marrenbach, and O. Ganschar, “Industrie 4.0 – volkswirtschaftliches potenzial für deutschland,” 2014. [Online]. Available: https://www.bitkom.org/sites/default/files/file/import/Studie-Industrie-40.pdf

[3] P. Gölzer, “Big data in industrie 4.0: Eine strukturierte aufarbeitung von anforderungen, anwendungsfällen und deren umsetzung,” Dissertation, Friedrich-Alexander-Universität, Erlangen-Nürnberg, 2016. [Online]. Available: https://d-nb.info/1123806233/34

[4] H. Fleischmann, P. Gölzer, J. Franke, and M. Amberg, “Kommunikation und datenaustausch in industrie 4.0*/communication and data exchange in industry 4.0 - requirements and capabilities of propagated communication protocols,” *wt Werkstattstechnik online*, vol. 105, no. 03, pp. 84–89, 2015.

[5] B. Bertsche and M. Dazer, *Zuverlässigkeit im Fahrzeug- und Maschinenbau: Ermittlung von Bauteil- und System-Zuverlässigkeiten*, 4th ed. Berlin: Springer Vieweg, 2022. [Online]. Available: http://www.springer.com/

[6] M. Bruhn and K. Hadwich, Eds., *Smart Services: Band 1: Konzepte – Methoden – Prozesse*, ser. Springer eBook Collection. Wiesbaden: Springer Fachmedien Wiesbaden and Imprint Springer Gabler, 2022.

[7] J. Kolerus and E. Becker, *Condition Monitoring und Instandhaltungsmanagement*. Tübingen: expert, 2022.

[8] J. D. Kelleher, B. MacNamee, and A. D’Arcy, *Fundamentals of machine learning for predictive data analytics: Algorithms, worked examples, and case studies*. Cambridge, Massachusetts and London, England: The MIT Press, 2015.

[9] S. Matzka, *Künstliche Intelligenz in den Ingenieurwissenschaften: Maschinelles Lernen verstehen und bewerten*, ser. Lehrbuch. Wiesbaden and Heidelberg: Springer Vieweg, 2021.

[10] S. Raschka, “Machine learning: Lecture notes,” 2018. [Online]. Available: https://sebastianraschka.com/pdf/lecture-notes/stat479fs18/02_knn_notes.pdf

[11] E. Alpaydın, *Maschinelles Lernen*, 3rd ed., ser. De Gruyter Studium. Berlin and Boston: De Gruyter Oldenbourg, 2022. [Online]. Available: https://www.degruyter.com/isbn/9783110740141

[12] A. Géron, *Praxiseinstieg Machine Learning mit Scikit-Learn, Keras und TensorFlow: Konzepte, Tools und Techniken für intelligente Systeme*, 2nd ed. Heidelberg: O’Reilly, 2020. [Online]. Available: https://www.assets.dpunkt.de/leseproben/13339/2_Entscheidungsb%C3%A4ume.pdf

[13] T. Schulz, Ed., *Industrie 4.0: Potenziale erkennen und umsetzen*, 2nd ed. Würzburg: Vogel Communications Group, 2021. [Online]. Available: https://www.content-select.com/index.php?id=bib_view&ean=9783834362544

[14] J. HEDDERICH, *ANGEWANDTE STATISTIK: Methodensammlung mit rincludes digital download*. [S.l.]: SPRINGER, 2020.

[15] C. Manzei, L. Schleupner, and R. Heinze, Eds., *Industrie 4.0 im internationalen Kontext: Kernkonzepte, Ergebnisse, Trends*, 2nd ed. Berlin and Offenbach and Berlin and Wien and Zürich: VDE Verlag GmbH and Beuth, 2017.

[16] J. Schreiner, “Was ist predictive maintenance? definition, anwendung und beispiele,” *Industrie of Things*, 2020. [Online]. Available: https://www.industry-of-things.de/was-ist-predictive-maintenance-definition-anwendung-und-beispiele-a-693842/

[17] Staufen, “Smart factory - nutzung von predictive-maintenance-anwendungen in deutschland 2019,” Deutscher Industrie 4.0 Index 2019, 2019. [Online]. Available: https://de.statista.com/statistik/daten/studie/1078451/umfrage/nutzung-von-predictive-maintenance-anwendungen-in-deutschland/

[18] *Fachlexikon MES & Industrie 4.0: Mehr als 870 Akronyme, Bezeichnungen und Schlüsselwörter aus der Begriffwelt Manufacturing Execution Systems (MES) und Industrie 4.0*, 5th ed. Berlin and Offenbach: VDE Verlag, 2022.

[19] W. Becker, P. Ulrich, O. Schmid, and C. Feichtinger, *Industrielle Digitalisierung: Entwicklungen und Strategien für mittelständische Unternehmen*, ser. Management und Controlling im Mittelstand. Wiesbaden and Heidelberg: Springer Gabler, 2020.

[20] F. Chaari, *Smart Monitoring of Rotating Machinery for Industry 4. 0*, ser. Applied Condition Monitoring Ser. Cham: Springer International Publishing AG, 2022, vol. v.19. [Online]. Available: https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=6710511

[21] *Technische Zuverlässigkeit: Problematik, math. Grundlagen, Untersuchungsmethoden, Anwendungen*, 3rd ed. Berlin and Heidelberg: SPRINGER, 1986.

[22] M. Schenk, Ed., *Instandhaltung technischer Systeme: Methoden und Werkzeuge zur Gewährleistung eines sicheren und wirtschaftlichen Anlagenbetriebs*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. [Online]. Available: http://nbn-resolving.org/urn:nbn:de:bsz:31-epflicht-1619742

[23] G. A. Susto, A. Schirru, S. Pampuri, S. McLoone, and A. Beghi, "Machine learning for predictive maintenance: A multiple classifier approach," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 3, pp. 812–820, 2015.

[24] D. Mourtzis, J. Angelopoulos, and N. Panopoulos, "Intelligent predictive maintenance and remote monitoring framework for industrial equipment based on mixed reality," *Frontiers in Mechanical Engineering*, vol. 6, 2020.

[25] J. Wang, L. Zhang, L. Duan, and R. X. Gao, "A new paradigm of cloud-based predictive maintenance for intelligent manufacturing," *Journal of Intelligent Manufacturing*, vol. 28, no. 5, pp. 1125–1137, 2017.

[26] M. von der Hude, *Predictive analytics und data mining: Eine Einführung mit R*, ser. Lehrbuch. Wiesbaden and Heidelberg: Springer Vieweg, 2020.

[27] L. Fahrmeir, T. Kneib, S. Lang, and B. Marx, *Regression: Models, methods and applications*. Berlin and Heidelberg: SPRINGER, 2013.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," 2011. [Online]. Available: https://dl.acm.org/doi/10.5555/1953048.2078195

[29] J. Frochte, *Maschinelles Lernen: Grundlagen und Algorithmen in Python*, ser. Hanser eLibrary. München: Hanser, 2018.

[30] W. Zhang, D. Yang, and H. Wang, "Data-driven methods for predictive maintenance

of industrial equipment: A survey," *IEEE Systems Journal*, vol. 13, no. 3, pp. 2213–2227, 2019.

[31] A. Khlaief, K. Nguyen, K. Medjaher, A. Picot, P. Maussion, D. Tobon, B. Chauchat, and R. Cheron, "Feature engineering for ball bearing combined-fault detection and diagnostic," in *2019 IEEE 12th International Symposium on Diagnostics for Electrical Machines, Power Electronics and Drives (SDEMPED)*. IEEE, 2019, pp. 384–390.

[32] L. Papula, *Mathematische Formelsammlung: Für Ingenieure und Naturwissenschaftler ; mit ... zahlreichen Rechenbeispielen und einer ausführlichen Integraltafel*, 12th ed., ser. Mathematik für Ingenieure und Naturwissenschaftler / Lothar Papula. Wiesbaden: Springer Vieweg, 2017.

[33] O. Kramer, *Machine learning for evolution strategies*, ser. Studies in Big Data. Cham: SPRINGER, 2016, vol. 20.

[34] J. Papa, *PyTorch kompakt: Syntax, Design Patterns und Codebeispiele für Deep-Learning-Modelle*, ser. Animals. Heidelberg: O'Reilly, 2021. [Online]. Available: http://nbn-resolving.org/urn:nbn:de:bsz:31-epflicht-1959417

[35] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017. [Online]. Available: https://openreview.net/forum?id=BJJsrmfCZ

[36] N. Amruthnath and T. Gupta, "A research study on unsupervised machine learning algorithms for early fault detection in predictive maintenance," in *2018 5th International Conference on Industrial Engineering and Applications (ICIEA)*. IEEE, 2018, pp. 355–361.

A. Anhang: Python Skript

```
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SequentialFeatureSelector
import pandas as pd
import numpy
import numpy as np
import csv
import os
import glob
import matplotlib.pyplot as plt
from sklearn import preprocessing
from enum import Enum
import copy
import matplotlib as mpl
import seaborn as sns
import numpy as np
from scipy.stats import norm, kurtosis, iqr
from sklearn.metrics import classification_report,
confusion_matrix, accuracy_score
from sklearn.decomposition import PCA

# Einbinden und Anpassen des Datensatzes
os.chdir("./Daten_mit_Label/")
all_filenames = [i for i in glob.glob('*.{}'.format(csv))]

# Alle Dateien in der Liste kombinieren
combined_csv = pd.concat([pd.read_csv(f) for f in all_filenames ])
#export to csv
combined_csv.to_csv( "combined_csv.csv", index=False,
encoding='utf-8-sig')
df = combined_csv
```

```
df.head()

# Normalzustand
data_path = ('./Daten_mit_Label/Ohne_Ausreisser_4s/
dry␣run.csv')
with open(data_path) as data:
    base = numpy.genfromtxt((line.replace(',', '.')
    for line in data),
    skip_header=1, delimiter=";")
print (base.shape)
df=pd.DataFrame(base)
df.to_csv( "./Daten_mit_Label/Ohne_Ausreisser_4s/
dry␣run.csv",
index=False, encoding='utf-8-sig')

dfh = pd.read_csv("./Daten_mit_Label/Ohne_Ausreisser_4s/
dry␣run.csv")
dfh.drop(columns=['y'])
print(dfh.describe())

# Holzsockel
data_path = ('./Daten_mit_Label/Ohne_Ausreisser_4s/
base.csv')
with open(data_path) as data:
    base = numpy.genfromtxt((line.replace(',', '.')
    for line in data),
    skip_header=1, delimiter=";")
print (base.shape)
df=pd.DataFrame(base)
df.to_csv( "./Daten_mit_Label/Ohne_Ausreisser_4s/
base.csv",
index=False, encoding='utf-8-sig')
```

```
dfn = pd.read_csv("./Daten_mit_Label/Ohne_Ausreisser_4s/
base.csv")
dfn.drop(columns=['y'])
print(dfn[['ax', 'ay', 'az', 'aT']].describe())


# Unwucht mit einem Gewicht
data_path = ('./Daten_mit_Label/Ohne_Ausreisser_4s/
imbalance␣1.csv')
with open(data_path) as data:
    base = numpy.genfromtxt((line.replace(',', '.')
    for line in data),
    skip_header=1, delimiter=";")
print (base.shape)
df=pd.DataFrame(base)
df.to_csv( "./Daten_mit_Label/Ohne_Ausreisser_4s/
imbalance␣1.csv",
index=False, encoding='utf-8-sig')

dfu1 = pd.read_csv("./Daten_mit_Label/Ohne_Ausreisser_4s/
imbalance␣1.csv")
print(dfu1[['ax', 'ay', 'az', 'aT']].describe())

# Unwucht mit zwei Gewichten
data_path = ('./Daten_mit_Label/Ohne_Ausreisser_4s/
imbalance␣2.csv')
with open(data_path) as data:
    base = numpy.genfromtxt((line.replace(',', '.')
    for line in data),
    skip_header=1, delimiter=";")
print (base.shape)
df=pd.DataFrame(base)
df.to_csv( "./Daten_mit_Label/Ohne_Ausreisser_4s/
```

```
imbalance␣2.csv",
index=False, encoding='utf-8-sig')

dfu2 = pd.read_csv("./Daten_mit_Label/Ohne_Ausreisser_4s/
imbalance␣2.csv")
dfu2.drop(columns=['y'])
print(dfu2[['ax', 'ay', 'az', 'aT']].describe())

# Boxplot
dfax = [dfh['ax'], dfn['ax'], dfu1['ax'], dfu2['ax']]
dfay = [dfh['ay'], dfn['ay'], dfu1['ay'], dfu2['ay']]
dfaz = [dfh['az'], dfn['az'], dfu1['az'], dfu2['az']]
dfaT = [dfh['aT'], dfn['aT'], dfu1['aT'], dfu2['aT']]
plt.rcParams["figure.figsize"] = [10.00, 10]
plt.boxplot(dfax)
plt.show()
plt.boxplot(dfay)
plt.show()
plt.boxplot(dfaz)
plt.show()
plt.boxplot(dfaT)
plt.show()

# Anpassen des Datensatzes
os.chdir("./")
all_filenames = [i for i in glob.glob('*./{}'.format(csv))]
print(all_filenames)

# Alle Dateien in der Liste kombinieren
combined = pd.concat([pd.read_csv(f) for f in all_filenames ])
combined.to_csv( "./Daten_mit_Label/ohne_Ausreiser/
combined.csv",
index=False, encoding='utf-8-sig')
```

```
df = combined
df.head()

# Merkmale aus den Daten erstellen
data_path = ('Daten_mit_Label\Ohne_Ausreisser_4s\combined.CSV')
df=pd.read_csv(data_path, sep=',')
print(df.head(20))

ax_mean = []
ax_max= []
ax_min= []
ax_media= []
ax_std= []
ax_kurt = []
ax_kurt_faktor = []
ax_var = []
ax_range = []
ax_schiefe = []
ax_schiefe_faktor = []
ax_quad_mean = []

ay_mean = []
ay_max= []
ay_min= []
ay_media= []
ay_std= []
ay_kurt = []
ay_kurt_faktor = []
ay_var = []
ay_range = []
ay_schiefe = []
ay_schiefe_faktor = []
ay_quad_mean = []
```

```
az_mean = []
az_max= []
az_min= []
az_media= []
az_std= []
az_kurt = []
az_kurt_faktor = []
az_var = []
az_range = []
az_schiefe = []
az_schiefe_faktor = []
az_quad_mean = []

aT_mean = []
aT_max= []
aT_min= []
aT_media= []
aT_std= []
aT_kurt = []
aT_kurt_faktor = []
aT_var = []
aT_range = []
aT_schiefe = []
aT_schiefe_faktor = []
aT_quad_mean = []
y_label = []

DATA_GROUP = 40

range_list = round(len(df)/DATA_GROUP)
print('Range List:', range_list)
for i in range(DATA_GROUP, len(df), DATA_GROUP):
```

```
for item in range(0,6):
 if item == 1:
   ax_mean.append(df.iloc[i-DATA_GROUP:i,item].mean())
   ax_max.append(df.iloc[i-DATA_GROUP:i,item].max())
   ax_min.append(df.iloc[i-DATA_GROUP:i,item].min())
   ax_media.append(df.iloc[i-DATA_GROUP:i,item].median())
   ax_std.append(df.iloc[i-DATA_GROUP:i,item].std())
   ax_kurt.append(kurtosis(df.iloc[i-DATA_GROUP:i,item]))
   ax_var.append(df.iloc[i-DATA_GROUP:i,item].var())
   ax_range.append(iqr(df.iloc[i-DATA_GROUP:i,item], rng=(0, 100)))
   ax_schiefe.append(df.iloc[i-DATA_GROUP:i,item].skew())


  if item == 2:
   ay_mean.append(df.iloc[i-DATA_GROUP:i,item].mean())
   ay_max.append(df.iloc[i-DATA_GROUP:i,item].max())
   ay_min.append(df.iloc[i-DATA_GROUP:i,item].min())
   ay_media.append(df.iloc[i-DATA_GROUP:i,item].median())
   ay_std.append(df.iloc[i-DATA_GROUP:i,item].std())
   ay_kurt.append(kurtosis(df.iloc[i-DATA_GROUP:i,item]))
   ay_var.append(df.iloc[i-DATA_GROUP:i,item].var())
   ay_range.append(iqr(df.iloc[i-DATA_GROUP:i,item],
   rng=(0, 100)))
   ay_schiefe.append(df.iloc[i-DATA_GROUP:i,item].skew())

  if item == 3:
   az_mean.append(df.iloc[i-DATA_GROUP:i,item].mean())
   az_max.append(df.iloc[i-DATA_GROUP:i,item].max())
   az_min.append(df.iloc[i-DATA_GROUP:i,item].min())
   az_media.append(df.iloc[i-DATA_GROUP:i,item].median())
   az_std.append(df.iloc[i-DATA_GROUP:i,item].std())
   az_kurt.append(kurtosis(df.iloc[i-DATA_GROUP:i,item]))
   az_var.append(df.iloc[i-DATA_GROUP:i,item].var())
```

```
    az_range.append(iqr(df.iloc[i-DATA_GROUP:i,item],
    rng=(0, 100)))
    az_schiefe.append(df.iloc[i-DATA_GROUP:i,item].skew())

   if item == 4:
    aT_mean.append(df.iloc[i-DATA_GROUP:i,item].mean())
    aT_max.append(df.iloc[i-DATA_GROUP:i,item].max())
    aT_min.append(df.iloc[i-DATA_GROUP:i,item].min())
    aT_media.append(df.iloc[i-DATA_GROUP:i,item].median())
    aT_std.append(df.iloc[i-DATA_GROUP:i,item].std())
    aT_kurt.append(kurtosis(df.iloc[i-DATA_GROUP:i,item]))
    aT_var.append(df.iloc[i-DATA_GROUP:i,item].var())
    aT_range.append(iqr(df.iloc[i-DATA_GROUP:i,item], rng=(0, 100)))
    aT_schiefe.append(df.iloc[i-DATA_GROUP:i,item].skew())


   if item == 5:
    y_value = df.iloc[i-DATA_GROUP:i,item].mean()
    y_value = round(y_value)
    y_label.append(y_value)

df = pd.DataFrame(list(zip(ax_mean, ax_max, ax_min,
ax_media, ax_std,
ax_kurt, ax_var, ax_range, ax_schiefe, ay_mean,
ay_max, ay_min,
ay_media, ay_std, ay_kurt, ay_var, ay_range,
ay_schiefe, az_mean,
az_max, az_min, az_media, az_std, az_kurt,
az_var, az_range,
az_schiefe, aT_mean, aT_max, aT_min, aT_media,
aT_std, aT_kurt,
aT_var, aT_range, aT_schiefe, y_label)),
columns= ['ax_mean', 'ax_max', 'ax_min',
```

```
'ax_media', 'ax_std', 'ax_kurt',  'ax_var',
'ax_range', 'ax_schiefe', 'ay_mean',
'ay_max', 'ay_min', 'ay_media', 'ay_std',
'ay_kurt', 'ay_var', 'ay_range',
'ay_schiefe', 'az_mean', 'az_max', 'az_min',
'az_media', 'az_std', 'az_kurt', 'az_var',
'az_range', 'az_schiefe', 'aT_mean', 'aT_max',
'aT_min', 'aT_media', 'aT_std', 'aT_kurt',
'aT_var', 'aT_range', 'aT_schiefe', 'y'])
print(df)
df.to_csv(r'./Daten_mit_Label/data.csv', index=False)

df = pd.read_csv('./Daten_mit_Label/data.csv', sep=",")

# Korrelationsmatrix
plt.figure(figsize = (15,8))
sns.heatmap(df.corr(),annot=True, cbar=False, fmt='.1f',
cmap="Blues")

X = df[['ax_mean', #1
        'ax_max', #2
        'ax_min', #3
        'ax_media', #4
        'ax_std', #5
        'ax_kurt',  #6
        'ax_var', #7
        'ax_range', #8
        'ax_schiefe', #9
        'ay_mean', #10
        'ay_max', #11
        'ay_min', #12
        'ay_media', #13
        'ay_std', #14
```

```
        'ay_kurt', #15
        'ay_var', #16
        'ay_range', #17
        'ay_schiefe', #18
        'az_mean', #19
        'az_max', #20
        'az_min', #21
        'az_media', #22
        'az_std', #23
        'az_kurt', #24
        'az_var', #25
        'az_range', #26
        'az_schiefe', #27
        'aT_mean', #28
        'aT_max', #29
        'aT_min', #30
        'aT_media', #31
        'aT_std', #32
        'aT_kurt', #33
        'aT_var', #34
        'aT_range',#35
        'aT_schiefe' #36
        ]].values
y = df[['y']]
print(X, '\n')

# converting the y data
lable_maping = {
    0: 'dry_run',
    1: 'base',
    2: 'imbalance1',
    3: 'imbalance2'
}
```

```
y = np.array(y).reshape(-1)
print(y)

print("\n", X)

# K-Nearest Neighbors
# Hauptkomponentenanalyse
from sklearn.neighbors import KNeighborsClassifier
for i in range (1,36,1):
    print(i)
    knn_clf=KNeighborsClassifier()
    pca = PCA(n_components=i)
    fit = pca.fit(X)

    #Summarize components

    print("Explained␣Variance:␣", fit.explained_variance_ratio_)

    # print(fit.components_)
    # sfs.fit(X,y)
    # sfs.get_support()
    Xknn = fit.transform(X)
    # Label = sfs.get_support()
    # print(Label)
    Xknn

    X_train, X_test, y_train, y_test = train_test_split(Xknn,
      y,
      test_size=0.2,
      shuffle=True,
      random_state=42)

    knn_clf=KNeighborsClassifier()
```

```
    knn_clf.fit(X_train,y_train)
    y_pred=knn_clf.predict(X_test)
    print('Genauigkeit␣Trainingsdaten:␣',
    knn_clf.score(X_train, y_train)* 100, '%')
    print('Genauigkeit␣Testdaten:␣', accuracy_score
    (y_test, y_pred)* 100, '%')
    print('-----------------------------------------------')
    print('Konfusionsmatrix:␣\n', confusion_matrix(
    y_test, y_pred))
    print('-----------------------------------------------')
    print('Classification␣Report:␣\n', classification_report(
    y_test, y_pred))
    print('-----------------------------------------------')

# Sequenzielle Vorwaertsauswahl
for features in range(1,36,1):
    knn_clf=KNeighborsClassifier(n_neighbors=5)
    sfs = SequentialFeatureSelector(knn_clf,
                      n_features_to_select=features,
                      cv=5)
    sfs.fit(X,y)
    sfs.get_support()
    Xknn = sfs.transform(X)
    Label = sfs.get_support()
    print(Label)
    Xknn
    X_train, X_test, y_train, y_test = train_test_split(Xknn,
        y,
        test_size=0.2,
        shuffle=True,
        random_state=Random_seed)
    knn_clf=KNeighborsClassifier(n_neighbors=5)
    knn_clf=knn_clf.fit(X_train,y_train)
```

```
    y_pred=knn_clf.predict(X_test)
    print("Anzahl␣der␣Merkmale:",features)
    print('Genauigkeit␣Trainingsdaten:␣', knn_clf.score(
    X_train, y_train) * 100, '%')
    print('Genauigkeit␣Testdaten:␣', accuracy_score(
    y_test, y_pred) * 100, '%')
    print('Konfusionsmatrix:␣\n', confusion_matrix(
    y_test, y_pred))
    print('Classification␣Report:␣\n', classification_report(
    y_test, y_pred))

# Logistische Regression
# Hauptkomponentenanalyse
from sklearn.linear_model import LogisticRegression
for i in range (1,36,1):
    print(i)
    clf_lr=LogisticRegression(max_iter=1000)
    pca = PCA(n_components=i)
    fit = pca.fit(X)

    #Summarize components

    print("Explained␣Variance:␣",
    fit.explained_variance_ratio_)

    # print(fit.components_)
    # sfs.fit(X,y)
    # sfs.get_support()
    Xlr = fit.transform(X)
    # Label = sfs.get_support()
    # print(Label)
    Xlr
```

```
    X_train, X_test, y_train, y_test = train_test_split(Xlr,
        y,
        test_size=0.2,
        shuffle=True,
        random_state=42)

    clf_lr = LogisticRegression(max_iter=1000)
    clf_lr = clf_lr.fit(X_train,y_train)
    y_pred = clf_lr.predict(X_test)
    print('Genauigkeit␣Trainingsdaten:␣', clf_lr.score(
    X_train, y_train)
    * 100, '%')
    print('Genauigkeit␣Testdaten:␣', accuracy_score(
    y_test, y_pred)
    * 100, '%')
    print('Konfusionsmatrix:␣\n', confusion_matrix(
    y_test, y_pred))
    print('Classification␣Report:␣\n', classification_report(
    y_test, y_pred))

# Sequenzielle Vorwaertsauswahl
from sklearn.linear_model import LogisticRegression

for features in range(1,36,1):
    clf_lr=LogisticRegression(max_iter=1000)
    sfs = SequentialFeatureSelector(clf_lr,
                          n_features_to_select=features,
                           cv=5)
    sfs.fit(X,y)
    sfs.get_support()
    Xlr = sfs.transform(X)
    Label = sfs.get_support()
    print(Label)
```

```
    Xlr
    X_train, X_test, y_train, y_test = train_test_split(Xlr,
         y,
         test_size=0.2,
         shuffle=True,
         random_state=Random_seed)
    clf_lr = LogisticRegression(max_iter=1000)
    clf_lr = clf_lr.fit(X_train,y_train)
    y_pred = clf_lr.predict(X_test)
    print("Anzahl␣der␣Merkmale:",features)
    print('Genauigkeit␣Trainingsdaten:␣', clf_lr.score(
    X_train, y_train)
    * 100, '%')
    print('Genauigkeit␣Testdaten:␣', accuracy_score(
    y_test, y_pred)
    * 100, '%')
    print('Konfusionsmatrix:␣\n', confusion_matrix(
    y_test, y_pred))
    print('Classification␣Report:␣\n', classification_report(
    y_test, y_pred))

# Entscheidungsbaum
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
for features in range(1,36,1):
    for tiefe in range (2,11,1):
        clf_dt = DecisionTreeClassifier(max_depth=tiefe)
        sfs = SequentialFeatureSelector(clf_dt,
                              n_features_to_select=features,
                               cv=5)
        sfs.fit(X,y)
        sfs.get_support()
        Xdt = sfs.transform(X)
```

```
        Label = sfs.get_support()
        print(Label)
        Xdt
        X_train, X_test, y_train, y_test = train_test_split(Xdt,
            y,
            test_size=0.2,
            shuffle=True,
           random_state=Random_seed)
        clf_dt = clf_dt.fit(X_train,y_train)
        y_pred = clf_dt.predict(X_test)
        print("Anzahl␣der␣Merkmale:",features)
        print("Tiefe:␣",tiefe)
        print('Genauigkeit␣Trainingsdaten:␣', clf_dt.score(
        X_train, y_train)
        * 100, '%')
        print('Genauigkeit␣Testdaten:␣', accuracy_score(
        y_test, y_pred)
        * 100, '%')
        print('Konfusionsmatrix:␣\n', confusion_matrix(
        y_test, y_pred))
        print('Classification␣Report:␣\n', classification_report(
        y_test, y_pred))

# Erstellen des Entscheidungsbaums
# Zuweisen der Lable
feature_lable=['Varianz␣ax',
               'Varianz␣ay',
               'Median␣aT']

lable_mapping = [
    'dry_run',
    'base',
    'imbalance1',
```

```
    'imbalance2']

# Diagramm des Entscheidungsbaums
fig = plt.figure(figsize=(25,20))
_ = tree.plot_tree(clf_dt,
                   feature_names=feature_lable,
                   class_names=lable_mapping,
                   filled=True,
                   fontsize=9)

# Random Forest
from sklearn.ensemble import RandomForestClassifier
for features in range(1,36,1):
    for tiefe in range (2,11,1):
        clf_rf = RandomForestClassifier(max_depth=tiefe)
        sfs = SequentialFeatureSelector(clf_rf,
                                        n_features_to_select=
                                        features,
                                        cv=5)
        sfs.fit(X,y)
        sfs.get_support()
        Xrf = sfs.transform(X)
        Label = sfs.get_support()
        print(Label)
        Xrf
        X_train, X_test, y_train, y_test = train_test_split(
                                    Xrf,
                  y,
                  test_size=0.2,
                  shuffle=True,
                  random_state=Random_seed)
        clf_rf = RandomForestClassifier(max_depth=tiefe)
        clf_rf = clf_rf.fit(X_train,y_train)
```

```
        y_pred = clf_rf.predict(X_test)
        print("Anzahl␣der␣Merkmale:",features)
        print("Tiefe:␣",tiefe)
        print('Genauigkeit␣Trainingsdaten:␣', clf_rf.score(
        X_train, y_train)
        * 100, '%')
        print('Genauigkeit␣Testdaten:␣', accuracy_score(
        y_test,y_pred)
        * 100, '%')
        print('Konfusionsmatrix:␣\n', confusion_matrix(
        y_test, y_pred))
        print('Classification␣Report:␣\n', classification_report(
        y_test, y_pred))

# Support Vector Machine
from sklearn import svm
for features in range(1,36,1):
    clf_svm = svm.SVC()
    sfs = SequentialFeatureSelector(clf_svm,
                                    n_features_to_select=features,
                                    cv=5)
    sfs.fit(X,y)
    sfs.get_support()
    Xsvm = sfs.transform(X)
    Label = sfs.get_support()
    print(Label)
    Xsvm
    X_train, X_test, y_train, y_test = train_test_split(Xsvm,
         y,
         test_size=0.2,
         shuffle=True,
         random_state=Random_seed)
    clf_svm = clf_svm.fit(X_train, y_train)
```

```
    y_pred = clf_svm.predict(X_test)
    print("Anzahl␣der␣Merkmale:",features)
    print('Genauigkeit␣Trainingsdaten:␣', clf_svm.score(
    X_train, y_train)
    * 100, '%')
    print('Genauigkeit␣Testdaten:␣', accuracy_score(
    y_test, y_pred)
    * 100, '%')
    print('Konfusionsmatrix:␣\n', confusion_matrix(
    y_test, y_pred))
    print('Classification␣Report:␣\n', classification_report(
    y_test, y_pred))

# Neuronales Netz
from sklearn.neural_network import MLPClassifier
for i in range(1,36,1):
    clf_nn = MLPClassifier(max_iter=10000, hidden_layer_sizes =
                           (8,))
    sfs = SequentialFeatureSelector(clf_nn,
                  n_features_to_select=i,
                  cv=5)
    sfs.fit(X,y)
    sfs.get_support()
    Xnn = sfs.transform(X)
    Label = sfs.get_support()
    print(Label)
    Xnn
    X_train, X_test, y_train, y_test = train_test_split(Xnn,
                y,
                test_size=0.2,
                shuffle=True,
                random_state=Random_seed)
    clf_nn = clf_nn.fit(X_train, y_train)
```

```
y_pred = clf_nn.predict(X_test)
print('Genauigkeit␣Trainingsdaten:␣', clf_nn.score(
X_train, y_train)
* 100, '%')
print('Genauigkeit␣Testdaten:␣', accuracy_score(
y_test, y_pred)
* 100, '%')
print('Konfusionsmatrix:␣\n', confusion_matrix(
y_test, y_pred))
print('Classification␣Report:␣\n', classification_report(
y_test, y_pred))
```

B. Anhang: Klassifikationsergebnisse

Tabelle B.1.: KNN Klassifikationsergebnis sequenzielle Vorwärtsauswahl (in Prozent)

Anzahl der Merkmale	Trainingsdaten	Testdaten
1	91,43	89,33
2	96,35	97,19
3	97,19	97,19
4	97,05	96,63
5	97,33	97,19
6	97,33	97,19
7	97,05	96,07
8	96,91	96,63
9	96,49	96,63
10	96,63	96,07
11	96,63	97,75
12	96,91	97,19
13	96,07	96,07
14	95,93	94,38
15	96,63	93,82
16	96,77	94,94
17	97,33	96,63
18	96,49	96,63
19	96,49	96,63
20	96,49	96,07
21	96,35	95,51
22	96,21	94,38
23	96,77	94,94
24	96,77	94,94
25	96,07	93,82
26	96,91	93,82
27	97,33	92,7
28	96,77	94,94
29	96,35	94,94
30	96,35	94,38
31	96,77	96,07
32	97,19	94,94
33	97,05	94,94
34	96,91	94,94
35	96,77	94,38

Tabelle B.2.: KNN Klassifikationsergebnis der Hauptkomponentenanalyse (in Prozent)

Anzahl der Komponenten	Ergebnis Trainigsdaten	Ergebnis Testdaten
1	83,57	79,78
2	89,19	85,39
3	89,19	89,89
4	91,15	91,01
5	93,40	92,13
6	96,07	92,70
7	96,51	92,13
8	96,07	94,38
9	95,79	93,82
10	96,07	92,70
11	96,63	92,70
12	96,63	93,82
13	96,21	93,82
14	96,49	94,38
15	96,77	94,38
16	96,35	93,82
17	96,63	93,82
18	96,63	94,38
19	96,49	93,26
20	96,49	93,26
21	96,49	93,26
22	96,49	92,70
23	96,21	93,26
24	96,35	93,82
25	96,35	93,26
26	96,35	93,26
27	96,35	93,26
28	96,36	93,26
29	96,35	93,26
30	96,35	93,26
31	96,35	93,26
32	96,35	93,26
33	96,35	93,26
34	96,35	93,26
35	96,35	93,26

Tabelle B.3.: Klassifikationsergebnis sequenzielle Vorwärtsauswahl Logistische Regression (in Prozent)

Anzahl der Merkmale	Trainingsdaten	Testdaten
1	89,61	91,01
2	94,52	95,51
3	95,22	95,51
4	95,08	96,63
5	95,79	96,63
6	96,21	96,63
7	96,21	96,63
8	96,07	96,63
9	96,63	96,63
10	96,49	96,63
11	96,49	96,63
12	96,91	96,07
13	97,19	96,07
14	97,19	96,07
15	97,19	96,07
16	97,19	96,07
17	97,05	96,07
18	97,33	96,07
19	97,19	96,07
20	96,77	96,07
21	96,91	96,07
22	97,05	95,01
23	96,91	97,19
24	97,61	96,63
25	97,19	97,19
26	97,05	97,75
27	97,19	97,75
28	97,05	97,19
29	97,19	96,63
30	96,77	96,07
31	96,77	95,51
32	97,05	94,94
33	96,91	96,07
34	96,91	96,07
35	96,91	96,07

Tabelle B.4.: Klassifikationsergebnis Hauptkomponentenanaylse Logistische Regression (in Prozent)

Anzahl der Komponenten	Ergebnis Trainigsdaten	Ergebnis Testdaten
1	81,46	83,71
2	86,24	89,33
3	88,34	91,01
4	90,17	90,45
5	90,73	90,45
6	93,96	92,69
7	94,24	92,70
8	95,22	94,51
9	95,37	95,51
10	96,07	96,07
11	95,79	95,51
12	95,65	94,94
13	95,65	94,94
14	95,93	94,94
15	96,07	95,51
16	96,07	95,51
17	96,21	94,94
18	96,21	96,07
19	96,21	96,07
20	96,50	96,07
21	96,49	96,07
22	96,77	96,63
23	96,63	96,07
24	96,63	96,07
25	96,77	96,07
26	96,91	96,07
27	96,91	96,07
28	96,91	96,07
29	96,91	96,07
30	96,91	96,07
31	96,91	96,07
32	96,91	96,07
33	96,91	96,07
34	96,91	96,07
35	96,91	96,07

Tabelle B.5.: Klassifikationsergebnis SVM (in Prozent)

Anzahl der Merkmale	Trainingsdaten	Testdaten
1	90,45	91,01
2	96,07	97,19
3	96,91	97,75
4	96,63	97,75
5	96,77	97,75
6	96,77	97,75
7	96,63	98,31
8	96,63	98,31
9	96,77	98,31
10	97,05	98,31
11	96,49	98,31
12	96,49	98,31
13	96,35	97,19
14	96,35	97,19
15	96,21	97,19
16	96,21	97,19
17	96,21	97,19
18	96,21	96,63
19	96,21	97,75
20	96,21	96,07
21	96,35	96,07
22	96,21	96,07
23	96,91	96,07
24	97,05	95,51
25	96,77	96,63
26	96,21	96,63
27	96,91	94,94
28	96,91	96,07
29	96,63	97,19
30	96,35	95,51
31	95,93	96,63
32	96,35	97,19
33	96,07	96,07
34	96,07	95,51
35	95,65	96,07

Tabelle B.6.: Klassifikationsergebnis Entscheidungsbaum (in Prozent)

Tiefe	Datensatz	Anzahl der Merkmale													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	Training	75,70	90,17	90,17	90,17	90,17	90,17	90,17	90,17	90,17	90,17	90,17	90,17	90,17	90,17
	Test	75,84	88,20	88,20	88,20	88,20	88,20	88,20	88,20	88,20	88,20	88,20	88,20	88,20	88,20
3	Training	91,29	96,21	96,21	96,21	96,21	96,21	96,21	96,21	96,21	96,21	96,21	96,21	96,21	96,21
	Test	89,33	95,51	95,51	95,51	95,51	95,51	95,51	95,51	95,51	95,51	95,51	95,51	95,51	95,51
4	Training	91,43	96,35	96,77	97,05	97,05	97,05	97,05	97,05	97,05	97,05	97,05	97,47	97,05	97,05
	Test	89,33	96,63	97,19	96,07	96,07	96,07	96,07	96,07	96,07	96,07	96,07	96,07	96,07	96,07
5	Training	91,71	96,77	98,03	98,03	98,17	98,17	98,31	98,31	98,31	98,31	98,31	98,31	98,46	98,31
	Test	88,20	95,51	97,19	96,63	97,19	96,63	96,63	97,19	97,13	97,19	96,63	96,63	96,07	97,19
6	Training	92,84	97,61	97,61	98,03	98,31	98,46	98,46	98,46	98,46	98,31	98,46	98,46	98,17	98,46
	Test	89,33	99,07	98,82	94,94	96,63	96,63	96,07	97,19	96,63	94,94	96,07	96,63	96,07	96,60
7	Training	93,12	98,46	98,46	99,02	98,74	98,74	98,88	98,74	98,88	98,6	98,88	98,74	98,74	99,02
	Test	88,76	96,63	96,63	93,26	96,93	93,82	95,91	96,63	94,38	96,07	95,51	94,38	96,07	94,94
8	Training	94,38	98,88	99,58	99,72	99,72	99,72	99,44	99,16	99,30	99,30	99,40	99,3	99,02	99,02
	Test	89,33	95,51	93,82	93,82	92,13	94,94	92,70	93,82	93,82	94,94	93,82	93,82	95,51	92,70
9	Training	94,66	99,30	100	99,72	99,72	99,72	99,86	99,58	99,72	99,72	99,72	99,72	99,86	99,86
	Test	89,89	96,07	92,70	95,51	93,82	92,70	95,51	97,19	93,82	96,07	92,70	94,38	95,51	93,82
10	Training	95,22	99,72	100	99,86	100	100	100	100	99,86	100	99,86	99,86	100	99,86
	Test	90,45	96,07	93,82	93,26	93,26	93,82	92,70	95,51	96,63	95,51	93,82	96,63	96,07	93,82

Tabelle B.7.: Klassifikationsergebnis Random Forest (in Prozent)

Tiefe	Datensatz	Anzahl der Merkmale													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	Training	73,88	94,24	95,51	95,93	95,79	95,79	95,79	95,79	95,09	96,07	95,79	95,79	96,07	96,07
	Test	74,16	94,38	94,38	95,51	95,51	95,51	96,07	95,51	94,94	95,51	95,51	95,51	95,51	95,51
3	Training	91,15	96,21	96,21	96,21	96,49	96,49	95,93	95,93	96,07	96,77	96,35	95,79	97,05	96,77
	Test	89,89	96,07	96,07	96,63	96,63	96,63	96,07	94,94	95,51	96,07	97,19	96,07	96,63	98,91
4	Training	91,15	96,63	96,77	96,91	97,05	96,91	98,03	96,91	97,47	97,47	96,77	97,19	97,33	97,19
	Test	89,89	96,63	96,07	96,07	97,75	96,63	97,75	97,75	97,75	97,75	97,75	97,75	97,75	97,19
5	Training	92,13	97,05	97,33	97,89	98,03	97,89	97,05	97,75	97,89	97,75	98,03	97,61	98,17	98,03
	Test	89,89	97,75	97,75	96,63	97,75	79,19	97,19	98,31	97,19	97,75	98,31	97,75	96,63	96,63
6	Training	92,56	98,03	98,03	98,60	98,76	98,17	98,74	99,02	98,60	98,46	99,16	98,88	99,02	99,16
	Test	89,89	97,75	97,75	97,19	98,31	96,63	96,63	97,75	97,75	97,75	97,19	96,63	97,75	97,19
7	Training	93,68	98,46	99,02	99,16	99,44	98,88	99,16	99,44	99,30	99,16	99,02	99,16	99,16	99,58
	Test	90,45	97,19	96,63	96,63	97,19	96,63	98,31	96,63	98,31	97,19	96,07	97,75	97,19	97,75
8	Training	94,52	99,16	99,16	99,58	100	99,58	99,72	99,58	99,72	99,86	99,58	99,58	99,86	99,72
	Test	90,45	97,75	96,63	95,51	97,75	97,19	97,19	97,75	97,19	96,63	97,75	96,63	97,75	96,63
9	Training	94,66	99,72	99,72	100	99,86	100	100	100	100	100	100	100	100	100
	Test	91,01	96,63	96,63	97,75	97,75	96,07	98,31	96,07	96,63	97,19	97,75	97,19	96,63	97,75
10	Training	95,79	99,86	100	100	100	100	100	100	100	100	100	100	100	100
	Test	89,33	96,63	96,63	96,63	97,75	96,63	97,19	97,19	98,31	91,19	96,63	96,07	97,75	97,75

Tabelle B.8.: Klassifikationsergebnis Neuronales Netz mit 6 Neuronen in der verdeckten Schicht (in Prozent)

Anzahl der Merkmale	Trainingsdaten	Testdaten
1	90,73	91,57
2	96,21	97,19
3	96,50	98,31
4	96,77	98,31
5	97,19	97,19
6	97,05	97,75
7	97,19	97,19
8	96,91	98,88
9	97,89	97,19
10	97,19	96,07
11	96,91	95,51
12	97,19	96,63
13	97,33	96,63
14	96,91	94,38
15	97,61	95,51
16	98,46	97,19
17	97,19	96,63
18	97,75	93,82
19	98,31	96,07
20	97,75	96,07

Tabelle B.9.: Klassifikationsergebnis Neuronales Netz mit 7 Neuronen in der Verdeckten Schicht (in Prozent)

Anzahl der Merkmale	Trainingsdaten	Testdaten
1	90,45	91,57
2	96,35	97,19
3	96,77	97,75
4	96,49	97,19
5	97,47	97,19
6	96,63	96,06
7	97,19	97,19
8	96,49	98,31
9	97,19	97,75
10	96,63	97,75
11	97,33	96,63
12	97,19	95,51
13	97,75	96,63
14	97,05	97,19
15	98,17	96,07
16	97,47	96,07
17	97,75	96,63
18	98,31	93,26
19	98,03	97,19
20	98,03	97,19

Tabelle B.10.: Klassifikationsergebnis Neuronales Netz mit 8 Neuronen in der verdeckten Schicht (in Prozent)

Anzahl der Merkmale	Trainingsdaten	Testdaten
1	90,17	91,57
2	96,07	97,19
3	96,63	98,31
4	96,49	98,31
5	97,05	97,19
6	96,91	96,07
7	96,91	97,19
8	97,47	95,51
9	97,19	97,75
10	97,33	98,31
11	97,48	96,63
12	97,47	96,63
13	97,75	96,07
14	97,61	97,19
15	97,61	97,19
16	97,47	95,51
17	97,89	96,63
18	97,61	95,51
19	96,91	92,13
20	97,89	92,70
21	98,31	93,26
22	98,31	93,82
23	98,60	96,07
24	98,03	92,70
25	97,90	94,38
26	98,74	96,07
27	97,47	94,38
28	98,03	94,38
29	97,19	94,38
30	98,17	96,07
31	97,90	95,51
32	97,89	94,38
33	97,75	94,94
34	98,17	93,82
35	97,61	93,82

Tabelle B.11.: Klassifikationsergebnis Neuronales Netz mit 9 Neuronen in der Verdeckten Schicht (in Prozent)

Anzahl der Merkmale	Trainingsdaten	Testdaten
1	90,45	91,57
2	95,93	97,75
3	96,77	98,31
4	96,07	97,19
5	97,05	97,19
6	96,91	97,19
7	97,19	94,94
8	97,19	97,75
9	97,05	97,19
10	97,89	97,75
11	96,91	95,51
12	97,61	97,19
13	97,75	96,07
14	97,33	94,94
15	97,33	96,07
16	97,47	94,94
17	97,75	96,63
18	97,61	94,38
19	97,75	95,51
20	97,75	97,19

Tabelle B.12.: Klassifikationsergebnis Neuronales Netz mit 10 Neuronen in der Verdeckten Schicht (in Prozent)

Anzahl der Merkmale	Trainingsdaten	Testdaten
1	90,31	91,57
2	96,63	97,19
3	96,63	97,75
4	97,19	97,75
5	96,63	98,31
6	97,05	97,19
7	97,47	96,63
8	97,33	97,75
9	97,47	97,19
10	97,47	97,75
11	97,75	97,19
12	98,03	96,07
13	97,05	97,75
14	97,47	96,07
15	97,90	94,94
16	96,77	95,51
17	97,61	95,51
18	97,19	94,38
19	98,03	94,94
20	97,75	95,51

Tabelle B.13.: Klassifikationsergebnis Neuronales Netz mit 8 Neuronen in der ersten verdeckten Schicht und 6 Neuronen in der zweiten verdeckten Schicht (in Prozent)

Anzahl der Merkmale	Trainingsdaten	Testdaten
1	90,03	90,45
2	93,54	94,94
3	95,51	96,63
4	96,63	98,31
5	97,19	96,63
6	96,63	96,63
7	94,10	96,07
8	96,91	97,19
9	96,77	97,75
10	97,19	95,51
11	96,91	96,63
12	97,90	96,63
13	97,61	96,63
14	98,46	97,19
15	97,75	97,75