Karlsruher Institut für Technologie

CAMMP

Masterarbeit

Aktivitätserkennung auf dem Smartphone

Entwicklung von Unterrichtsmaterial für computergestützte mathematische Modellierungsprojekte

Vorgelegt von	Katja Hoeffer [redacted]
Erstprüfer	Prof. Dr. Martin Frank Steinbuch Centre for Computing (SCC) Karlsruher Institut für Technologie (KIT)
Zweitprüferin	Dr. Ingrid Lenhardt Fakultät für Mathematik - Abteilung für Didaktik Karlsruher Institut für Technologie (KIT)
Koreferentin	Dr. Sarah Schönbrodt Steinbuch Centre for Computing (SCC) Karlsruher Institut für Technologie (KIT)

Karlsruhe, den 24. Oktober 2022

.

Inhaltsverzeichnis

1. Einleitung

Mobile Geräte, wie zum Beispiel Smartphones, sind mittlerweile ständige Begleiter im Alltag. Weltweit nutzen rund 3,9 Milliarden Menschen ein Smartphone[1]. Zudem ist in den letzten Jahren das Interesse an der Auswertung der Gewohnheiten und der täglichen Routinen der Menschen gestiegen. Es hat sich gezeigt, dass die Analyse des menschlichen Verhaltens von zentraler Bedeutung für das Verständnis der Bedürfnisse und Anforderungen einer Person ist. Diese Beobachtungen sind in vielen Bereichen, von der Pädagogik, Medizin und Soziologie bis hin zum Marketing, relevant (vgl. Banos et al., 2014, S. 6475). Zu den menschlichen Gewohnheiten zählen auch die täglichen Aktivitäten einer Person. Durch die enorme Weiterentwicklung der in den Smartphones eingebauten Sensoren können diese zur Erkennung der menschlichen Aktivitäten genutzt werden. In den letzten Jahren hat sich daher die Aktivitätserkennung zu einem aktiven Forschungsfeld entwickelt, wobei sich die meisten Studien auf die Erkennung von Aktivitäten wie Sitzen, Stehen, Gehen, Joggen etc. konzentrieren (vgl. Su et al., 2014, S. 235).

Das Ziel der menschlichen Aktivitätserkennung ist es, die Aktivitäten oder Situationen, in denen sich eine Nutzerin oder ein Nutzer[2] befindet, über Sensoren am Smartphone zu bestimmen und gegebenenfalls darauf zu reagieren (vgl. Banos et al., 2014, S. 6475). In den letzten Jahren wurden die meisten gängigen Smartphones mit zahlreichen Sensoren ausgestattet, unter anderem mit Beschleunigungssensoren, GPS-Sensoren, Gyroskopen, Barometern und Kompassen. Diese Sensoren stellen eine reichhaltige Datenquelle dar, um die täglichen Gewohnheiten und Routinen der Menschen aufzuzeichnen und zu analysieren (vgl. Su et al., 2014, S. 235). Im Zusammenhang mit der Aktivitätserkennung haben sich vor allem die 3-Achsen-Beschleunigungsmesser (Accelerometer) als nützlich erwiesen (vgl. Ustev et al., 2013, S. 1429). Vielfach kommen maschinelle Lernverfahren zum Einsatz, um die aufgenommenen Sensordaten den entsprechenden Aktivitäten zuordnen zu können (vgl. Banos et al., 2014, S. 6475).

Für die Nutzer können sich aus der Aktivitätserkennung mit dem Smartphone große Vorteile ergeben, die ihnen unter Umständen den Alltag bedeutend erleichtern. Neben bekannten Anwendungen wie Fitness-Tracking und der Überwachung von Gesundheitsdaten könnte das Smartphone beispielsweise auch die Schrift vergrößern, wenn der Nutzer versucht, im Gehen (Chat-)Nachrichten zu lesen (vgl. Dittrich, 2014, S. 17). Darüber hinaus kann die Aktivitätserkennung auch für Unternehmen zum Beispiel bei der Personalisierung von Werbung von Nutzen sein. Trotz der zahlreichen Vorteile und Anwendungsgebiete steht die Aktivitätserkennung auf dem Smartphone auch vor einigen Herausforderungen. Dazu zählen zum Beispiel die eingeschränkten Energie- und

[1] `https://de.statista.com/themen/581/smartphones/#dossierKeyfigures`, letzter Aufruf: 15.09.2022.

[2] Nachfolgend werden Nutzerinnen und Nutzer unter der Bezeichnung „Nutzer“ zusammengefasst. Analog wird mit den Personengruppen Schülerinnen und Schüler, Lehrerinnen und Lehrer, u. a. verfahren.

Rechenressourcen im Vergleich zu einem leistungsstarken Rechner aufgrund der begrenzten Akkulaufzeit und des beschränkten Speicherplatzes (vgl. Ustev et al., 2013, S. 1428).

Ziel dieser Abschlussarbeit ist es, Lehr- und Lernmaterial zur Aktivitätserkennung auf dem Smartphone zu entwickeln. In dem erstellten Lernmodul arbeiten Schüler ab Klasse zehn mit realen Aktivitätsdaten und entwickeln ihr eigenes mathematisches Modell zur Aktivitätserkennung. Dabei verwenden sie aktuelle Methoden aus dem Bereich der künstlichen Intelligenz. Das Material wurde so konzipiert, dass es sich sowohl für den Einsatz im Mathematikunterricht eignet als auch im Rahmen eines Projekttages bearbeitet werden kann.

Im folgenden Kapitel 2 dieser Abschlussarbeit wird zunächst der didaktische Hintergrund erläutert. Dazu zählt die Vorstellung des Projekts CAMMP sowie eine fachdidaktische Einführung in die mathematische Modellierung. Anschließend wird in Kapitel 3 der mathematische Hintergrund des Lernmoduls erläutert. In Kapitel 4 wird das didaktisch-methodische Konzept des Lernmaterials vorgestellt. Dabei werden zum einen die Ziele und die curriculare Einbindung des entwickelten Lernmoduls dargelegt sowie der Ablauf des Moduls im Rahmen eines Projekttages und im Rahmen einer Unterrichtseinheit im Fach Mathematik erläutert. Zum anderen werden die digitalen Werkzeuge und die entwickelten Materialien vorgestellt. Anschließend wird in Kapitel 5 die Durchführung des Lernmoduls im Rahmen eines Projekttages sowie einer zwei Doppelstunden umfassenden Unterrichtseinheit evaluiert. Den Abschluss der Arbeit bildet ein kurzer Ausblick auf weitere Möglichkeiten zur Verbesserung und Erweiterung des erstellten Lernmaterials.

2. Didaktischer Hintergrund

Das erstellte Lernmodul wurde im Rahmen des Schülerprogramms CAMMP entwickelt und erprobt. Daher wird CAMMP zu Beginn dieses Kapitels kurz vorgestellt. Anschließend werden die didaktischen Prinzipien, die dem Lernmaterial zugrunde liegen, erläutert. Da sich alle Projekte von CAMMP und somit auch das entwickelte Lernmodul mit der mathematischen Modellierung beschäftigen, stehen die fachdidaktischen Theorien der mathematischen Modellierung im Vordergrund.

2.1. CAMMP

Bei CAMMP handelt es sich um ein außerschulisches Angebot des Karlsruher Instituts für Technologie (KIT) und ein Schülerlabor der RWTH Aachen. Die Abkürzung CAMMP steht für **C**omputational **a**nd **M**athematical **M**odeling **P**rogram (computergestütztes mathematisches Modellierungsprogramm) (vgl. CAMMP, 2022h). Gegründet wurde CAMMP 2011 von Prof. Dr. Ahmed Ismail, Dr. Nicole Faber und Prof. Dr. Martin Frank an der RWTH Aachen. Durch den Wechsel von Prof. Dr. Martin Frank ans KIT entstand im September 2017 der zweite Standort von CAMMP in Karlsruhe (vgl. CAMMP, 2022d). In Karlsruhe ist CAMMP Teil des Projektes Simulierte Welten[3].

In verschiedenen Lernmodulen und Veranstaltungsformaten von CAMMP haben Schüler die Möglichkeit, in die Rolle von Wissenschaftlern zu schlüpfen und realen Problemen aus dem Alltag, der Industrie und der Forschung nachzugehen. Zur Lösung der Probleme setzen die Schüler mathematische Methoden ein und nutzen Computersimulationen. Im Fokus steht dabei die mathematische Modellierung. Ziel ist es, den Schülern anhand von realen Problemstellungen die Grundlagen der mathematischen Modellierung näher zu bringen (vgl. CAMMP, 2022d).

CAMMP hat sich zum übergeordneten Ziel gesetzt, sowohl Schülern als auch Lehrkräften die Bedeutung von Mathematik und Simulationswissenschaften für unsere Gesellschaft aufzuzeigen. Durch die längere und intensivere Auseinandersetzung mit einer Problemstellung erhalten Schüler Einblicke in die Berufswelt von Mathematikern, Informatikern und Ingenieuren. CAMMP fungiert somit auch als Berufs- und Studienorientierung. Neben der Förderung mathematischer Kompetenzen werden durch die Zusammenarbeit in kleinen Gruppen auch prozessbezogene Kompetenzen wie Kommunikations- und Teamfähigkeit gestärkt. Darüber hinaus stellt CAMMP einen guten Anknüpfungspunkt für das Schulfach IMP (Informatik, Mathematik und Physik) dar. In den verschiedenen Formaten von CAMMP wird das Zusammenspiel dieser drei Fächer umfassend genutzt (vgl. CAMMP, 2022d).

[3]Nähere Infos unter `https://simulierte-welten.de`, letzter Aufruf: 17.09.2022.

2.1.1. Veranstaltungsformate von CAMMP

Die meisten Angebote von CAMMP richten sich an Schüler ab der achten Klasse. Darüber hinaus bietet CAMMP auch Studierenden und Lehrkräften die Möglichkeit, sich näher mit der mathematischen Modellierung auseinanderzusetzen (vgl. CAMMP, 2022g).

Angebote für Schüler

Der CAMMP day und die CAMMP week stellen den Kern der Veranstaltungsformate von CAMMP dar. Diese werden im Folgenden kurz vorgestellt.

Der *CAMMP day* ist ein mathematischer Modellierungstag, bei dem sich Schülergruppen mit praxisorientierten Fragestellungen beschäftigen. Dabei handelt es sich meistens um Klassen oder Kurse eines allgemeinbildenden Gymnasiums. Zusätzlich werden für einzelne mathematikinteressierte Schüler auch sogenannte offene CAMMP days angeboten (vgl. CAMMP, 2022f). Während des Modellierungstages arbeiten die Schüler in kleinen Gruppen an einer vorab didaktisch-methodisch ausgearbeiteten Problemstellung. Dabei werden sie von wissenschaftlichen Mitarbeitern (vielfach Studierende des Lehramts Mathematik) unterstützt und erhalten einen Einblick in die Grundlagen der mathematischen Modellierung. Zudem bieten die Modellierungstage den Schülern einen Einblick in die Problemlösestrategien zahlreicher MINT-Berufe und Studiengänge. Durchgeführt werden die CAMMP days sowohl vor Ort an den Universitäten bzw. Schulen als auch mit Hilfe verschiedener Kommunikationstools online (vgl. Wohak et al., 2021, S. 37 ff.). Zur Zeit werden beispielsweise die folgenden Lernmodule angeboten[4]:

- Wie funktioniert Musikerkennung und was hat das mit Mathe zu tun?
- Gibt es den Klimawandel wirklich?
- Die Netflix Challenge und was das mit Mathe zu tun hat!

Bei der *CAMMP week* handelt es sich um eine Modellierungswoche, in der mathematikinteressierte Schüler der Oberstufe an Fragestellungen aus der Forschung von Firmen und Universitätsinstituten arbeiten (vgl. CAMMP, 2022e). Während der fünf Tage forschen die Schülerteams an individuellen Problemstellungen. Dabei werden sie stets von einem Wissenschaftler betreut. Im Rahmen einer Abschlussveranstaltung am Ende der Modellierungswoche präsentieren die Schülergruppen ihre Ergebnisse den Vertretern der Firmen und Universitäten (vgl. Schönbrodt et al., 2022, S. 155 ff.).

[4] Weitere Lernmodule und genauere Beschreibungen unter `https://www.cammp.online/116.php`, letzter Aufruf: 17.09.2022.

Neben dem CAMMP day und der CAMMP week können Schüler auch im Rahmen eines *Praktikums*[5] die mathematische Modellierung in vielen Bereichen der Industrie und der Forschung kennenlernen (vgl. CAMMP, 2022b).

Angebote für (angehende) Lehrkräfte

Neben den CAMMP days unterstützt CAMMP auch Lehrkräfte bei der eigenständigen Durchführung von Lernmodulen. Dazu stellt CAMMP den Lehrkräften Unterrichtsmaterialien zu realen, authentischen Problemstellungen zur Verfügung. Hierzu wurde unter anderem das *CAMMP book* veröffentlicht, in dem didaktisch aufbereitetes Unterrichtsmaterial zu ausgewählten Lernmodulen rund um die mathematische Modellierung vorgestellt wird. Weitere Angebote sind ein- und mehrtägige Lehrerfortbildungen sowie die Kooperation mit Partnerschulen (vgl. CAMMP, 2022a). Für Lehramtsstudierende bietet CAMMP Seminare sowie Abschlussarbeiten zur mathematischen Modellierung an (vgl. CAMMP, 2022c).

2.2. Mathematische Modellierung

Die mathematische Modellierung steht im Fokus des Projekts CAMMP und somit auch im Fokus des im Rahmen dieser Arbeit entwickelten Lernmoduls. Im Folgenden werden zentrale fachdidaktische Prinzipien der mathematischen Modellierung vorgestellt. Dabei wird auch auf die Bedeutung digitaler Werkzeuge im Modellierungsprozess eingegangen sowie der Bezug zum Bildungsplan des Faches Mathematik in Baden-Württemberg aufgezeigt.

2.2.1. Mathematisches Modellieren und mathematische Modelle

Die mathematische Modellierung stellt einen Teilbereich der angewandten Mathematik dar, bei dem das Lösen von Problemen aus der Realität in den Vordergrund gestellt wird (vgl. Greefrath et al., 2013, S. 11).

Im Verlauf der mathematischen Modellierung wird ein *mathematisches Modell* entwickelt, das die Realität vereinfacht darstellt. Ziel der Vereinfachung ist es, den Einsatz mathematischer Methoden zur Lösung des Problems zu ermöglichen. Bei der Modellbildung werden nur die als wesentlich erachteten und hinreichend objektivierbaren Aspekte der Realität berücksichtigt. Formal stellt ein mathematisches Modell ein Tripel (R, M, f) dar, wobei R ein gewisser Ausschnitt der Realität, M eine Teilmenge der mathematischen Welt und f eine Abbildung von der Realität R in die mathematische Welt M ist (vgl. Greefrath et al., 2013, S. 12).

[5] Nur am Standort Karlsruhe; nähere Infos unter `https://www.scc.kit.edu/forschung/12132.php`, letzter Aufruf: 17.09.2022.

Anhand dieser Charakterisierung von mathematischen Modellen wird deutlich, dass einerseits die Modelle zur Beschreibung der Realität nicht eindeutig sind. Die Vereinfachungen können auf unterschiedliche Art und Weise vorgenommen werden, denn verschiedene Aspekte der Realität können als wichtig erachtet werden. Andererseits ist die Beschreibung und Bearbeitung realer Probleme mit Hilfe der mathematischen Modellierung dadurch begrenzt, dass die komplexe Realität nicht vollständig durch ein Modell dargestellt werden kann. Eine solche detaillierte Darstellung der Realität ist meist auch nicht wünschenswert, denn es würde dem Ziel der Modellbildung widersprechen. Modelle sollen für eine überschaubare Verarbeitung der realen Daten geeignet sein (vgl. Greefrath et al., 2013, S. 13).

2.2.2. Modellierungskreisläufe

Zur idealtypischen Beschreibung des Modellierungsprozesses, der im Wesentlichen aus der Übersetzung eines realen Problems in die Mathematik, der Arbeit mit mathematischen Methoden und der Rückübersetzung der mathematischen Lösung in die Realität besteht, werden häufig Kreisläufe genutzt (vgl. Greefrath & Siller, 2018, S. 3). In der Literatur existieren viele verschiedene Modellierungskreisläufe, die sich anhand der unterschiedlichen Phasen des Modellierungsprozesses unterscheiden lassen und mit unterschiedlichen Zielsetzungen entwickelt wurden (vgl. Greefrath et al., 2013, S. 14). Im Folgenden werden drei Modellierungskreisläufe vorgestellt, die hinsichtlich des entwickelten Lernmoduls relevant sind.

Siebenschrittiger Modellierungskreislauf

Im Rahmen des DISUM-Projekts wurde der in Abbildung 1 dargestellte, siebenschrittige Modellierungskreislauf von Blum und Leiß entwickelt. Ziel dieses Kreislaufs ist es, den Umgang der Lernenden mit Modellierungsaufgaben möglichst genau beschreiben zu können (vgl. Greefrath et al., 2013, S. 17). Die einzelnen Teilschritte dieses Modellierungskreislaufs entsprechen den Teilkompetenzen des Modellierens, auf die in Abschnitt 2.2.3 genauer eingegangen wird.

Diese sehr detaillierte Darstellung der einzelnen Schritte im siebenschrittigen Modellierungskreislauf eignet sich insbesondere für Diagnosezwecke, da kognitive Aspekte des Modellierens berücksichtigt werden (vgl. Niss & Blum, 2020, S. 25). Zur Beurteilung der Modellierungsprozesse der Schüler können Lehrkräfte die einzelnen Modellierungsschritte anhand dieses Kreislaufs separat bewerten (vgl. Niss & Blum, 2020, S 99 f.).

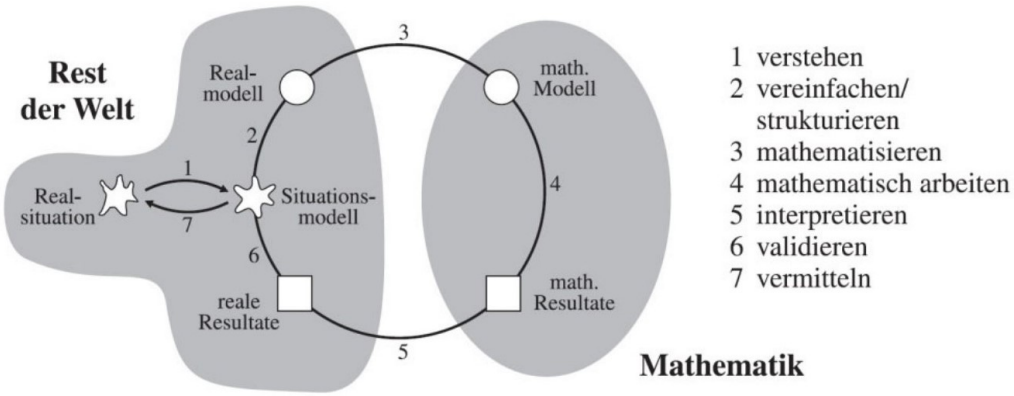


Abbildung 1: siebenschrittiger Modellierungskreislauf nach Blum und Leiß (entnommen aus Greefrath et al., 2013, S. 18)

Vierschrittiger Modellierungskreislauf

Im Hinblick darauf, auch Lernenden einen Überblick und eine Orientierung über den eigenen Modellierungsprozess zu geben und metakognitive Kompetenzen zu fördern, scheint ein reduzierter Kreislauf besser geeignet. Im Rahmen von Veranstaltungen des Projekts CAMMP wird der in Abbildung 2 dargestellte vierschrittige Modellierungskreislauf genutzt. Dieser ist an den von Blum (1985) entwickelten Modellierungskreislauf angelehnt (vgl. Frank et al., 2022, S. 2 f.). Auch bei dem erstellten Lernmodul zur Aktivitätserkennung wird dieser Kreislauf als Orientierungshilfe für Schüler eingesetzt.

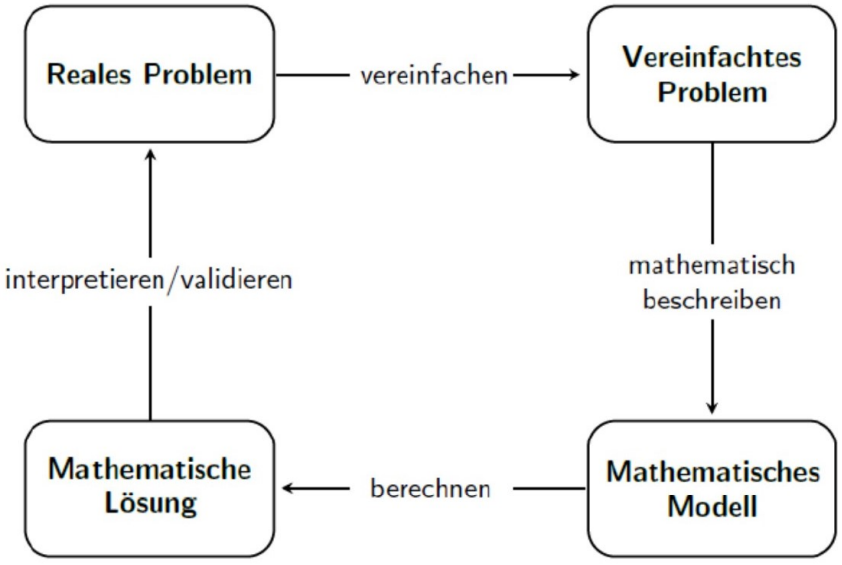


Abbildung 2: Von CAMMP genutzter Modellierungskreislauf

Ausgangspunkt dieses Kreislaufs ist wiederum das reale Problem, das im ersten Schritt vereinfacht, strukturiert und präzisiert wird. Das dadurch gewonnene vereinfachte Problem wird im zweiten Schritt mit Hilfe mathematischer Beschreibungen in ein mathematisches Modell übersetzt. Anschließend können auf diesem mathematischen Modell Berechnungen durchgeführt werden. Hierbei ist das Ziel, eine mathematische Lösung des Problems zu finden. Im vierten und letzten Schritt wird die gefundene mathematische Lösung in Bezug auf das reale Problem interpretiert und validiert. Ist die erzielte Lösung noch nicht zufriedenstellend, wird der Kreislauf (ganz oder teilweise) erneut durchlaufen. Dabei werden die zuvor getroffenen Annahmen, Vereinfachungen und Berechnungsmethoden überdacht und gegebenenfalls angepasst (vgl. Frank et al., 2022, S. 2 f.).

Computergestützte Modellierungsspirale

Eine Weiterentwicklung des vierschrittigen Modellierungskreislaufs stellt die von der Initiative Computer-Based Math eingeführte *Solution Helix of Math* dar (s. Abb. 3). Diese computergestützte Modellierungsspirale wurde zum einen um den Aspekt der Annäherung an eine immer bessere Lösung durch die Wiederholung der Modellierungsschritte erweitert. Zum anderen wurden die mathematischen Berechnungen, also der dritte Schritt des Modellierungskreislaufs, an einen Computer übergeben (vgl. Frank et al., 2018, S. 149). Auf die Einbindung digitaler Werkzeuge in den Modellierungsprozess und die damit verbundenen Vorteile wird in Abschnitt 2.2.5 näher eingegangen.

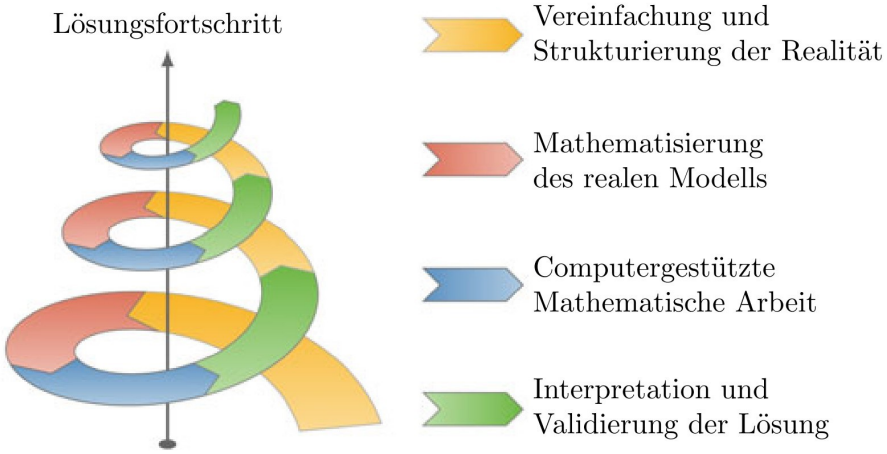


Abbildung 3: Computergestützte Modellierungsspirale (entnommen aus Frank et al., 2018, S. 140)

2.2.3. Modellierungskompetenzen

Nach Weinert (2001) wird der Kompetenzbegriff wie folgt definiert:

> *„Dabei versteht man unter Kompetenzen die bei Individuen verfügbaren oder durch sie erlernbaren kognitiven Fähigkeiten und Fertigkeiten, um bestimmte Probleme zu lösen, sowie die damit verbundenen motivationalen, volitionalen und sozialen Bereitschaften und Fähigkeiten, um die Problemlösungen in variablen Situationen erfolgreich und verantwortungsvoll nutzen zu können“* (Weinert, 2001, zitiert nach Stender, 2016, S. 36 f.).

In den Bildungsstandards des Fachs Mathematik für die allgemeine Hochschulreife zählt das *mathematische Modellieren* zu den sechs zu erwerbenden allgemeinen mathematischen Kompetenzen (vgl. Kultusministerkonferenz, 2012, S. 11). Auch im Bildungsplan Mathematik für Gymnasien des Landes Baden-Württemberg ist *Modellieren* eine der fünf prozessbezogenen Kompetenzen, die im Laufe des Bildungsprozesses bei allen Schülern entwickelt werden sollen (vgl. Ministerium für Kultus, Jugend und Sport Baden-Württemberg, 2016, S. 6). Unter der Kompetenz *Modellieren* wird dabei das Folgende verstanden:

> *„Die Schülerinnen und Schüler bearbeiten realitätsbezogene Fragestellungen, indem sie deren Struktur analysieren, sie vereinfachen und Annahmen treffen. Sie übersetzen die Situation in ein mathematisches Modell, finden im mathematischen Modell ein Ergebnis und interpretieren es in der Realsituation. Sie überprüfen das Ergebnis im Hinblick auf Stimmigkeit und Angemessenheit. Sie diskutieren die Tragweite von durch Modellierung gewonnenen Prognosen kritisch“* (Ministerium für Kultus, Jugend und Sport Baden-Württemberg, 2016, S. 13).

Die Beschreibung der prozessbezogene Kompetenz *Modellieren* umfasst damit alle sieben Teilkompetenzen der mathematischen Modellierung. Diese Teilkompetenzen stellen zudem die einzelnen Teilschritte des siebenschrittigen Modellierungskreislaufs in Abbildung 1 dar und werden in Tabelle 1 näher erläutert.

Zur Förderung der Modellierungskompetenz können zwei Ansätze verfolgt werden: der *atomistische Ansatz* und der *holistische Ansatz*.
Beim *atomistischen Ansatz* werden die einzelnen Teilkompetenzen getrennt voneinander gefördert. Dadurch ergibt sich die Möglichkeit, die Komplexität für Lehrende und Lernende zu reduzieren. Dieser Ansatz bietet sich vor allem bei Schülern mit geringer Modellierungserfahrung an. Der Einstieg in die Behandlung von Modellierungsaufgaben im Unterricht ist weniger zeitaufwändig und eine Fokussierung auf den Lernprozess ist möglich. Blomhøj und Højgaard weisen darauf hin, dass ein Problem dieses Ansatzes darin liegt, dass es für die Entwicklung der Kompetenzen im Bereich der Strukturierung einer komplexen Fragestellung nicht ausreicht, Schüler nur mit vorstrukturierten Problemen arbeiten zu lassen (vgl. Stender, 2016, S. 39).

Im Gegensatz dazu bearbeiten Schüler beim *holistischen Ansatz* Modellierungsaufgaben in ihrer ganzen Komplexität. Blomhøj und Højgaard schätzen diesen Ansatz grundsätzlich als motivierender für Schüler ein, da der Grad an Authentizität durch das vollständige Durchlaufen des Modellierungsprozesses erhöht wird. Ein Problem dieses Ansatzes ist der notwendige Zeitaufwand, weshalb im Unterricht selten Modellierungsaufgaben in ihrer ganzen Komplexität durchgeführt werden (vgl. Stender, 2016, S. 39 f.).

Tabelle 1: Teilkompetenzen der mathematischen Modellierung

Teilkompetenz	**Indikator**
Verstehen	Die Schüler entwickeln ausgehend von der *Realsituation* ein eigenes mentales Modell, das *Situationsmodell*. Zur Erstellung des mentalen Modells müssen die Schüler die Problemsituation verstehen (vgl. Greefrath & Siller, 2018, S. 6).
Vereinfachen & Strukturieren	Um aus dem Situationsmodell ein *Realmodell* entwickeln zu können, müssen die Schüler wichtige Angaben identifizieren und strukturieren. Zudem müssen vereinfachende und idealisierende Annahmen getroffen werden, um die Komplexität des Problems zu verringern (vgl. Greefrath & Siller, 2018, S. 6).
Mathematisieren	Die Schüler übersetzten durch Mathematisierungen das Realmodell in ein *mathematisches Modell*. Dazu nutzten sie Terme, Gleichungen, Diagramme und Funktionen (vgl. Greefrath & Siller, 2018, S. 6).
Mathematisch arbeiten	Die Schüler nutzen bei der Arbeit mit dem zuvor entwickelten mathematischen Modell ihre mathematischen Kenntnisse und geeignete Verfahren, um ein *mathematisches Resultat* zu erhalten. Bei dieser Teilkompetenz kommen häufig auch digitale Werkzeuge zum Einsatz (vgl. Greefrath & Siller, 2018, S. 6).
Interpretieren	Die gewonnenen Resultate werden in die Realsituation übertragen. Durch die Interpretation der Ergebnisse im Bezug zur realen Situation erhalten die Schüler ein *reales Resultat* (vgl. Greefrath & Siller, 2018, S. 6).
Validieren	Die Schüler beurteilen das reale Resultat im Situationsmodell. Darüber hinaus können in diesem Schritt verschiedene mathematische Modelle zu einer Realsituation verglichen und bewertet werden (vgl. Greefrath & Siller, 2018, S. 6).
Vermitteln	Wird das reale Resultat im Situationsmodell als angemessen erachtet, so übertragen die Schüler die Resultate auf die reale Situation und haben eine Lösung des Problems gefunden. Falls die Resultate noch nicht zufriedenstellend sind, müssen die Teilschritte zwei bis sechs unter veränderten Annahmen, Vereinfachungen und Berechnungsmethoden erneut durchlaufen werden (vgl. Greefrath & Siller, 2018, S. 6).

2.2.4. Ziele des Modellierens

Durch den Einsatz von Modellierungsaufgaben im Mathematikunterricht können nicht nur die Kompetenzen in diesem Bereich gefördert, sondern auch verschiedene Bildungsziele erreicht und die Winter'schen Grunderfahrungen adressiert werden. Nach Winter (1995, S. 37) soll der Mathematikunterricht folgende Grunderfahrungen ermöglichen und dadurch zur Allgemeinbildung beitragen:

> 1. *„Erscheinungen der Welt um uns, die uns alle angehen oder angehen sollten, aus Natur, Gesellschaft und Kultur, in einer spezifischen Art wahrzunehmen und zu verstehen,*
> 2. *mathematische Gegenstände und Sachverhalte, repräsentiert in Sprache, Symbolen, Bildern und Formeln, als geistige Schöpfungen, als eine deduktiv geordnete Welt eigener Art kennen zu lernen und zu begreifen,*
> 3. *in der Auseinandersetzung mit Aufgaben Problemlösefähigkeiten, die über die Mathematik hinaus gehen, (heuristische Fähigkeiten) zu erwerben.“*

Gerade die erste Grunderfahrung entspricht einer der Hauptcharakteristika der mathematischen Modellierung: das Verständnis einer realen und relevanten Fragestellung aus Natur, Gesellschaft und Kultur erfordert Mathematik. Beim Einsatz von Modellierungsaufgaben wird zudem die dritte Grunderfahrung berücksichtigt, während die zweite untergeordnet behandelt wird (vgl. Frank et al., 2022, S. 1).

Darüber hinaus werden bei der Beschäftigung mit Modellierungsaufgaben unterschiedliche Ziele verfolgt. Vier dieser Ziele werden im Folgenden näher erläutert:

- **Allgemeine Ziele:**
  Den Schülern soll durch den Einsatz von Modellierungsaufgaben ein ausgewogenes Bild der Mathematik als Wissenschaft, das alle Teilaspekte der Mathematik umfasst, vermittelt werden (vgl. Stender, 2016, S. 17). Darüber hinaus sollen die Schüler zu einem verantwortungsvollen Mitglied der Gesellschaft erzogen werden. Als solches sollen sie in der Lage sein, verwendete Modelle (z. B. Steuermodelle) kritisch zu hinterfragen. Auch soziale Kompetenzen sollen durch das gemeinsame Arbeiten an Modellierungsaufgaben gefördert werden (vgl. Greefrath et al., 2013, S. 20).

- **Inhaltsorientierte Ziele:**
  Im Rahmen der Bearbeitung von Modellierungsaufgaben beschäftigen sich die Schüler mit realen Problemen aus ihrer Umwelt und lernen so, sich ihre Umwelt mit Hilfe von mathematischen Methoden zu erschließen. Dazu zählt auch, dass die Schüler in der Lage sein sollen, Erscheinungen unserer Welt wahrzunehmen und zu verstehen. Die inhaltsorientierten Ziele adressieren somit die erste der drei Winter'schen Grunderfahrungen (vgl. Greefrath et al., 2013, S. 20).

- **Prozessbezogene Ziele:**
  Im Zuge der mathematischen Modellierung werden auch allgemeine mathematische Kompetenzen wie beispielsweise die *Problemlösefähigkeit*, die *Kommunikationsfähigkeit* und die *Argumentationsfähigkeit* gefördert. Zusätzlich können zentrale *heuristische Strategien* des Problemlösens erlernt und weiterentwickelt werden. Dazu zählen zum Beispiel das Arbeiten mit Analogien oder das Rückwärtsarbeiten. In den prozessbezogenen Zielen lässt sich somit die dritte Winter'sche Grunderfahrung wiederfinden (vgl. Greefrath et al., 2013, S. 20).

- **Lernpsychologische Ziele:**
  Aus lernpsychologischer Sicht soll durch die Arbeit an Modellierungsaufgaben die Motivation und das Interesse der Schüler an der Mathematik gesteigert werden. Dies soll dazu beitragen, dass die mathematischen Inhalte leichter verstanden und besser behalten werden können (vgl. Greefrath et al., 2013, S. 20).

2.2.5. Modellieren mit digitalen Werkzeugen

Bedingt durch die fortschreitende Digitalisierung an den Schulen gewinnt auch der Einsatz digitaler Werkzeuge im Mathematikunterricht immer mehr an Bedeutung. Dabei sollten jedoch die folgenden beiden Aussagen von Stender (2016, S. 43 f.) berücksichtigt werden:

> - *„Wird das entsprechende Instrument beherrscht, so ist es beim Verstehen von Mathematik hilfreich und ein mächtiges Mittel beim Lösen von (zu dem Instrument passenden) Problemen."*
> - *„Wird das Instrument (noch) nicht beherrscht, so muss es zunächst eingeführt werden und der Umgang mit der jeweiligen Software von den Schülerinnen und Schülern erlernt werden. Daher ist es dann zunächst eine zusätzliche Hürde im Lern- und Arbeitsprozess und es wird zur Einführung des Werkzeugs Unterrichtszeit in nicht unerheblichem Umfang benötigt."*

Durch den Einsatz digitaler Werkzeuge wird die Integration komplexer Anwendungen und Modellierungsaufgaben in die tägliche Unterrichtspraxis ermöglicht. Dabei rückt der eigentliche Modellierungsprozess und das tiefergehende Verständnis für dieses Verfahren in den Vordergrund, denn der Rechenaufwand kann durch die digitalen Werkzeuge erheblich reduziert werden. In Abbildung 4 wird verdeutlicht, in welchen Teilschritten des Modellierungskreislaufs digitale Werkzeuge zum Einsatz kommen können.

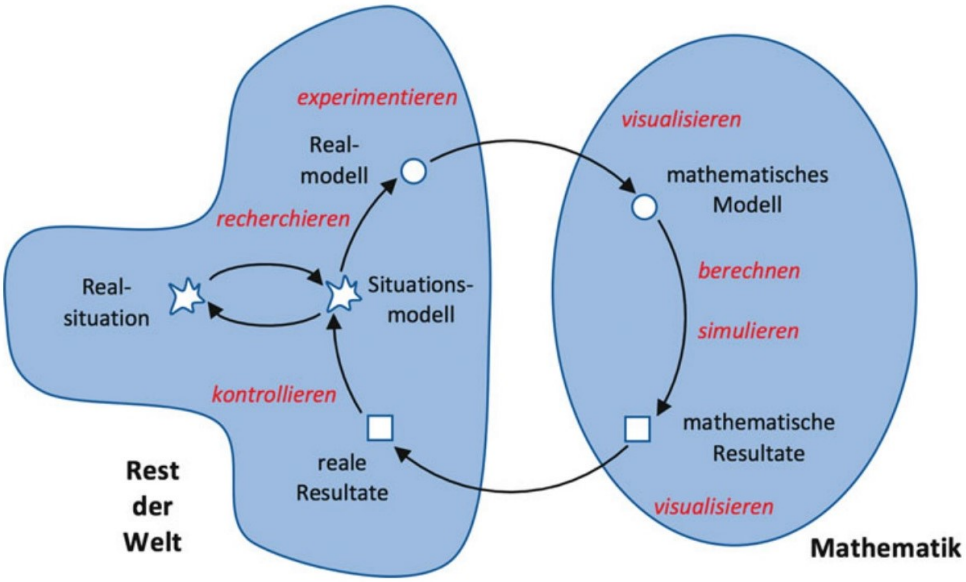


Abbildung 4: Einsatzmöglichkeiten digitaler Werkzeuge im Modellierungskreislauf (entnommen aus Greefrath & Siller, 2018, S. 12)

Beim Verständnis der Realsituation und der Entdeckung mathematischer Zusammenhänge können digitale Werkzeuge zum *Experimentieren* und *Recherchieren* genutzt werden. Zum einen können die Schüler in einer Internetrecherche Informationen über die reale Situation sammeln oder erzielte Ergebnisse validieren (vgl. Stender, 2016, S. 44). Zum anderen können die Daten der Realsituation beispielsweise durch den Einsatz einer dynamischen Geometriesoftware oder einer Tabellenkalkulation in ein geometrisches Modell überführt werden. Anschließend können die Schüler in diesem Modell Experimente durchführen, um sich die reale Situation zu veranschaulichen und Erkenntnisse über den dargestellten Sachverhalt zu gewinnen (vgl. Greefrath & Siller, 2018, S. 9). Mit Hilfe digitaler Werkzeuge können auch unterschiedliche Darstellungen der Daten sowie der Ergebnisse erzeugt werden. Digitale Werkzeuge können also auch zur *Visualisierung* genutzt werden (vgl. Greefrath & Siller, 2018, S. 10). Eines der Haupteinsatzgebiete der digitalen Werkzeuge ist das *Berechnen* von Ergebnissen mit Hilfe von Computeralgebrasystemen. Diese Systeme sind dann relevant, wenn die Schüler die gewünschten Ergebnisse nicht oder nicht in einer angemessenen Zeit ohne entsprechende Systeme erhalten können. Zu diesem Einsatzgebiet zählt auch das Finden algebraischer Darstellungen und das *Simulieren* (vgl. Kaiser et al., 2015, S. 372). Darüber hinaus können digitale Werkzeuge auch das *Kontrollieren* und *Überprüfen* der erhaltenen Lösungen unterstützen (vgl. Greefrath & Siller, 2018, S. 10).

Bei dem im Rahmen dieser Arbeit entwickelten Lernmodul sind digitale Werkzeuge von zentraler Bedeutung. Konkret arbeiten die Schüler mit der Programmiersprache Python und mit sogenannten Jupyter Notebooks. Auf diese beiden Werkzeuge wird in Kapitel 4.3.1 ausführlicher eingegangen.

2.3. Weitere didaktische Prinzipien

Im Folgenden werden weitere didaktische Prinzipien vorgestellt, die bei der Erstellung des Lernmaterials berücksichtigt wurden. Dazu zählen das EIS-Prinzip, das Prinzip der minimalen Hilfe und der Umgang mit heterogenen Lerngruppen.

2.3.1. Das EIS-Prinzip

Nach Bruner erfolgt die Denkentwicklung gleichzeitig auf drei unterschiedlichen Darstellungsebenen:

- der *enaktiven* Darstellungsebene (E),
- der *ikonischen* Darstellungsebene (I),
- und der *symbolischen* Darstellungsebene (S).

Für den Lernprozess ist insbesondere der Wechsel zwischen den einzelnen Darstellungsebenen entscheidend (vgl. Sill, 2019, S. 138).

Auf der *enaktiven Ebene* erfassen die Schüler den Lerngegenstand durch eigene oder vorgestellte Handlungen (vgl. Sill, 2019, S. 138). Beispielsweise könnte das bei der Bruchrechnung durch das Aufteilen eines Kuchens in unterschiedliche Anteile realisiert werden.
Bei der *ikonischen Ebene* geht es um die bildliche Darstellung der Sachverhalte. Die Schüler erlernen durch die Erfassung von Bildern, Graphen, Tabellen, Skizzen usw. mathematische Sachverhalte (vgl. Sill, 2019, S. 138). Im Fall der Bruchrechnung könnte das durch das Färben von Flächen passend zum gegeben Anteil oder durch die Bestimmung des Anteils gefärbter Flächen umgesetzt werden.
Auf der *symbolischen Ebene* wird der Lerngegenstand durch verbale Beschreibungen oder formale Zeichensysteme erfasst (vgl. Sill, 2019, S. 138). Für die Bruchrechnung wird also zum Beispiel die Notation $\frac{1}{4}$ und die Sprechweise „ein Viertel“ eingeführt.

Auch bei dem im Rahmen dieser Arbeit entwickelten Lernmodul wird auf die verschiedenen Darstellungsebenen eingegangen, wobei besonders die ikonische und symbolische Ebene zum Einsatz kommen.

2.3.2. Das Prinzip der minimalen Hilfe

Wie bereits in Abschnitt 2.2.3 durch den Vergleich des atomistischen und des holistischen Ansatzes zur Förderung von Modellierungskompetenzen aufgezeigt wurde, lassen sich die Kompetenzen in diesem Bereich nur durch das eigene Tun und somit das selbständige Modellieren erweitern. Angesichts des hohen Komplexitätsgrades von Modellierungsaufgaben kann von den Schülern im unterrichtlichen Rahmen jedoch keine komplett selbständige Bearbeitung erwartet werden. Für Lehrkräfte ergibt sich daraus der Konflikt, den Lernenden einerseits so weit wie möglich selbständiges Arbeiten zu

ermöglichen, andererseits aber auch genügend Hilfen zur Verfügung zu stellen, sodass bei der Modellierung ein akzeptables Ergebnis erreicht werden kann (vgl. Stender, 2016, S. 75).

Der Schweizer Didaktiker Hans Aebli formulierte bereits im Jahre 1983 mit dem *Prinzip der minimalen Hilfe* eine Handlungsempfehlung für Lehrkräfte, wie mit dem oben beschriebenen Problem umgegangen werden sollte (vgl. Stender, 2016, S. 76). Nach Aebli sollen die Lernenden so lange selbständig arbeiten, wie sie zu sinnvollen und zur Lösung des Gesamtproblems notwendigen Erkenntnissen kommen und keine zusätzlichen Hilfen erfragen. Selbst wenn die Schüler Hilfen benötigen, soll die Lehrkraft keinesfalls direkt massiv eingreifen, wie es zum Beispiel durch das Liefern von Lösungselementen der Fall wäre. Zunächst sollen allgemeine Aufforderungen zur Beobachtung oder zum Nachdenken an die gesamte Klasse gerichtet werden. Erst bei anhaltenden Schwierigkeiten übernimmt die Lehrkraft die Leitung, um die Lernenden den richtigen Erkenntnissen entgegen zu führen. Dazu gibt sie Hinweise auf bestimmte Problemgegenstände, die besonders beachtet werden sollten. Erst in letzter Konsequenz zeigt die Lehrkraft den Schülern durch eng gefasste Fragen und Aufforderungen den Weg zur richtigen Lösung (vgl. Aebli, 2006, S. 300).

Aufbauend auf dem Prinzip der minimalen Hilfe entwickelte Friedrich Zech im Jahre 1996 ein fünfstufiges Handlungsmodell für Lehrkräfte. Die einzelnen Stufen sollen schrittweise und an den notwendigen Grad der Unterstützung der Schüler angepasst eingesetzt werden. In diesem Sinne stellt das Handlungskonzept von Zech eine Realisierung des Prinzips der minimalen Hilfe dar (vgl. Stender, 2016, S. 78). Im Folgenden werden die einzelnen Stufen näher erläutert.

1. **Motivationshilfen:**
   Durch *Motivationshilfen* sollen die Schüler zur Weiterarbeit an einer Aufgabe ermutigt werden. Diese Art der Hilfe hat jedoch keinerlei Bezug zur eigentlichen Aufgabenstellung (vgl. Zech, 1998, S. 316). Ein Beispiel für eine Motivationshilfe wäre „Du wirst die Aufgabe schon schaffen!“ (Zech, 1998, S. 319).

2. **Rückmeldehilfen:**
   Bei *Rückmeldehilfen* erhalten die Schüler Feedback zu ihrer bisher geleisteten Arbeit. Diese Art von Hilfe berücksichtigt den konkreten Arbeitsstand sowie den Arbeitsfortschritt der Schüler und wirkt im Falle einer positiven Rückmeldung motivierend (vgl. Zech, 1998, S. 316). Zech (1998, S. 319) führt als Beispiele Äußerungen wie „Du bist auf dem richtigen Weg!“, „Da musst du nochmal nachrechnen!“ und „Mach weiter so!“ auf.

3. **Allgemein-strategische Hilfen:**
   Durch *allgemein-strategische Hilfen* sollen Schüler auf (fachübergreifende oder fachspezifische) Methoden zur Problemlösung aufmerksam gemacht werden. Ziel

dabei ist es, den Arbeitsprozess der Lernenden zu steuern, ohne dabei auf konkrete Fachinhalte einzugehen (vgl. Zech, 1998, S. 316 f.). Dies geschieht beispielsweise durch Aussagen wie „Mach dir doch mal eine Zeichnung!“ oder „Versuche, die gegebenen Daten in einen Zusammenhang zu bringen!“ (Zech, 1998, S. 319).

4. **Inhaltsorientiert-strategische Hilfen:**
   Die *inhaltsorientiert-strategischen Hilfen* enthalten ebenfalls noch keine Informationen über den Inhalt der jeweiligen Aufgabe (vgl. Zech, 1998, S. 317). Die Schüler sollen durch Äußerungen wie „Versuche, das Problem graphisch zu lösen!“ oder „Versuche, deine Kenntnisse bezüglich Geschwindigkeit anzuwenden!“ (Zech, 1998, S. 319) dazu angeregt werden, fachspezifische Arbeitsverfahren zu nutzen (vgl. Zech, 1998, S. 317).

5. **Inhaltliche Hilfen:**
   Inhaltliche Hilfen liefern den Lernenden konkrete Informationen über den Inhalt des aktuellen Arbeitsschrittes. Dies kann einerseits durch die Benennung des benötigten mathematischen Verfahrens (z. B. „Wie ist Geschwindigkeit definiert!“ (Zech, 1998, S. 319)) und andererseits durch die Vorgabe der nächsten Arbeitsschritte (z. B. „Versuche, aus zwei der Größen v, s, t die dritte zu berechnen!“ (Zech, 1998, S. 319)) erfolgen (vgl. Zech, 1998, S. 317).

2.3.3. Heterogenität von Lerngruppen

Unter dem Begriff *Heterogenität* wird in der Pädagogik die Unterscheidung der Lernenden anhand sogenannter Heterogenitätsmerkmale verstanden. Beispiele für solche lernrelevanten Merkmale sind das Alter und Geschlecht, das Vorwissen, die Interessen, die Lernmotivation sowie die kulturelle und soziale Herkunft (vgl. Sorgalla, 2015, S. 2).

Auch im Bildungsplan Mathematik für Gymnasien des Landes Baden-Württemberg ist das Thema Heterogenität zu finden. Darin wird gefordert, dass die Lehrkräfte den individuellen Lernvoraussetzungen der Schüler durch binnendifferenzierende Maßnahmen, Differenzierungen bei den Zugangsweisen oder individualisierende Unterrichtsformate gerecht werden. Dabei sollen sich die Individualität der Schüler entfalten sowie die personalen und sozialen Kompetenzen weiterentwickeln können. Darüber hinaus soll die Differenzierung und Individualisierung genutzt werden, um MINT-begeisterten Schülern weitere Vertiefungen anzubieten (vgl. Ministerium für Kultus, Jugend und Sport Baden-Württemberg, 2016, S. 10).

Bei der Entwicklung des Lernmaterials wurde versucht, diesen Anforderungen gerecht zu werden. Das Lernmaterial kann zum einen binnendifferenziert eingesetzt und zum anderen als Vertiefung für MINT-begeisterte Schüler genutzt werden.

3. Mathematischer und fachlicher Hintergrund

In diesem Kapitel wird der mathematische und fachliche Hintergrund der Aktivitätserkennung auf dem Smartphone vorgestellt. Zunächst werden die Begriffe künstliche Intelligenz und maschinelles Lernen erläutert. Anschließend werden die notwendigen mathematischen Grundlagen dargestellt sowie eine Einführung in die Aktivitätserkennung gegeben. Am Ende dieses Kapitels werden die einzelnen Modellierungsschritte des Lernmoduls vorgestellt.

3.1. Künstliche Intelligenz und Maschinelles Lernen

In diesem Abschnitt werden die Begriffe *künstliche Intelligenz* und *maschinelles Lernen* erläutert sowie der im Lernmodul verwendete Klassifikationsalgorithmus vorgestellt.

Vorab sei jedoch angemerkt, dass sich für beide Begriffe in der Literatur nicht *die eine Definition* finden lässt. Je nachdem in welcher Fachrichtung (Informatik, Mathematik, ...) sich mit diesen Begriffen auseinandergesetzt wird, werden andere Definitionen und Beschreibungen aufgeführt. Um dennoch Schülern sowie Lehrkräften, die sich zuvor nicht mit diesem Thema beschäftigt haben, eine Vorstellung beider Begriffe liefern zu können, werden im Folgenden die Auffassungen der Begriffe erläutert, die für das entwickelte Lernmodul relevant sind.

3.1.1. Künstliche Intelligenz

Der Begriff der *künstlichen Intelligenz (KI)* ist in den letzten Jahren immer populärer geworden. Methoden aus dem Bereich KI sind die Basis vieler Anwendungen aus unserem täglichen Leben. Copeland (2019) definierte den Begriff KI wie folgt:

> *„Künstliche Intelligenz ist die Fähigkeit eines Computers oder computergesteuerten Roboters, Aufgaben zu lösen, die normalerweise von intelligenten Wesen erledigt werden“* (zitiert nach Paaß & Hecker, 2020, S. 1).

Ein Computersystem soll also in der Lage sein, ähnlich wie ein Mensch intelligent zu handeln und eigenständig zu lernen. Diese Definition nach Copeland ist allerdings sehr vage, da der Begriff der Intelligenz nicht klar definiert ist und sich nur schwer abgrenzen lässt (vgl. Paaß & Hecker, 2020, S. 1). So stellt sich beispielsweise die Frage, ob nach Copelands Definition nicht bereits ein Taschenrechner als KI bezeichnet werden müsste – denn Rechenaufgaben werden von intelligenten Menschen gelöst. In diesem Sinne ist diese Definition mit Vorsicht zu betrachten und soll nur als eine grobe Orientierung dienen.

Das Thema KI stellt aber keinesfalls ein neues Forschungsgebiet dar. Bereits im Jahr 1950 beschäftigte sich der britische Mathematiker Alan Turing mit der Frage, ob Maschinen bzw. Computersysteme intelligent sein können. Zur Untersuchung dieser Frage

schlug er den folgenden Test vor (s. Abb. 5). Ein menschlicher Schiedsrichter kommuniziert elektronisch mit zwei Partnern (ein Partner ist ein Mensch, der andere ein Computer) und stellt ihnen beliebige Fragen. Kann der Schiedsrichter nach vielen Fragen nicht entscheiden, bei welchem der Partner es sich um den Computer handelt, so wird der Computer als intelligent angesehen. Dieser Test wird auch als Turing-Test bezeichnet (vgl. Paaß & Hecker, 2020, S. 2).

Abbildung 5: Turing Test (entnommen aus Paaß & Hecker, 2020, S. 3)

Der Begriff KI wurde allerdings erst sechs Jahre später von Prof. John Mc Carthy beim Dartmouth-Workshop vorgeschlagen (vgl. Paaß & Hecker, 2020, S. 11)[6].

3.1.2. Maschinelles Lernen

In der Literatur wird das *maschinelle Lernen* meist als Teilgebiet von KI aufgefasst. Der Begriff maschinelles Lernen wird von Mitchell (1997) folgendermaßen definiert:

> *„Machine Learning is the study of computer algorithms that improve automatically through experience“* (Mitchell, 1997).

Ausgangspunkt vieler gängiger maschineller Lernverfahren ist eine große Menge an Daten bzw. Beispielen. Ziel ist es, eine bestimmte Aufgabe (z. B. Bilderkennung) zu lösen. Zur Erfüllung dieser Aufgabe wird ein Modell entwickelt. Dieses Modell wird vorab **nicht** von einem Spezialisten durch komplexe Regelsysteme festgelegt. Stattdessen werden die vorhandenen Daten und Informationen genutzt, um die Parameter des Modells bestmöglich zu wählen. Dieser Prozess wird als *Lernen* bezeichnet (vgl. Schönbrodt, 2022, S. 156).

Eine Anwendung, bei der maschinelle Lernverfahren zum Einsatz kommen, sind Spamfilter eines E-Mail Postfachs. Anhand von Mail-Beispielen wird ein Modell entwickelt und gewisse Modellparameter bestmöglich „gelernt“ mit dem Ziel, Spam E-Mails

[6]Ein kurzes, aufschlussreiches Video zur Geschichte von KI finden interessierte Leser unter `https://www.youtube.com/watch?v=09LotPHTZtU`, letzter Aufruf: 16.09.2022.

von gewöhnlichen E-Mails (so genannte Ham E-Mails) unterscheiden zu können (vgl. Géron, 2020, S. 4).

Strategien des maschinellen Lernens

Im Bereich des maschinellen Lernens gibt es unterschiedliche Ansätze: das *überwachte Lernen*, das *unüberwachte Lernen* und das *bestärkende Lernen* (vgl. Frochte, 2021, S. 21). Das im Rahmen dieses Lernmoduls eingesetzte Verfahren lässt sich dem überwachten Lernen zuordnen, weshalb dies hier ausführlicher beschrieben wird.

Überwachtes Lernen
Bei der Methode des *überwachten Lernens* (engl. *supervised learning*) besteht der Beispieldatensatz sowohl aus den Eingangsdaten als auch aus den gewünschten Ergebnissen (so genannte *Labels*). In diesem Fall spricht man auch von einem *gelabelten* Datensatz (vgl. Frochte, 2021, S. 21). Ein Datenpunkt im Datensatz X besteht also aus dem Merkmalsvektor $\vec{x}_i$ und dem zugehörigen Label y_i. Im Vektor $\vec{x}_i$ werden die Werte aller Merkmale des i-ten Datenpunkts gespeichert. Setzt sich ein Datenpunkt aus n Merkmalen zusammen, so haben die Vektoren $\vec{x}_i$ die Dimension n (vgl. Géron, 2020, S. 42). Ein Datensatz X mit m Datenpunkten enthält dann die Merkmalsvektoren $\vec{x}_i$, $i = 1, ..., m$ und die Labels y_i, $i = 1, ..., m$:

$$X = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), ..., (\vec{x}_m, y_m)\} \subseteq \mathbb{R}^n \times Y. \tag{3.1}$$

Ziel des überwachten maschinellen Lernverfahrens ist es, eine Funktion

$$f : X \to Y \tag{3.2}$$

zu finden, die jedem Vektor $\vec{x}_i$ aus dem Datensatz X das zugehörige Label y_i aus der Menge Y zuordnet (vgl. Frochte, 2021, S. 21).

In Abbildung 6 ist das Prinzip des überwachten maschinellen Lernens dargestellt. Der gelabelte Beispieldatensatz wird zu Beginn in zwei kleinere Datensätze, die *Trainings-* und die *Testdaten* aufgeteilt. Meistens werden 70 bis 80 Prozent der Daten den Trainingsdaten und 20 bis 30 Prozent der Daten den Testdaten zugeordnet (vgl. Krause & Natterer, 2019, S. 23 f.). Mit Hilfe des Trainingsdatensatzes wird dann eine Funktion f gelernt, die die Trainingsdatenpunkte den zugehörigen Labels zuordnet. Im Anschluss an die *Trainingsphase*, wird die Funktion f in der *Testphase* auf den Testdatensatz angewandt. Da es sich bei den Testdaten ebenfalls um vorab gelabelte Datenpunkte handelt, kann überprüft werden, ob die Funktion f den Testdatenpunkten die richtigen Labels zuordnet. Dadurch kann der Lernerfolg bewertet werden.

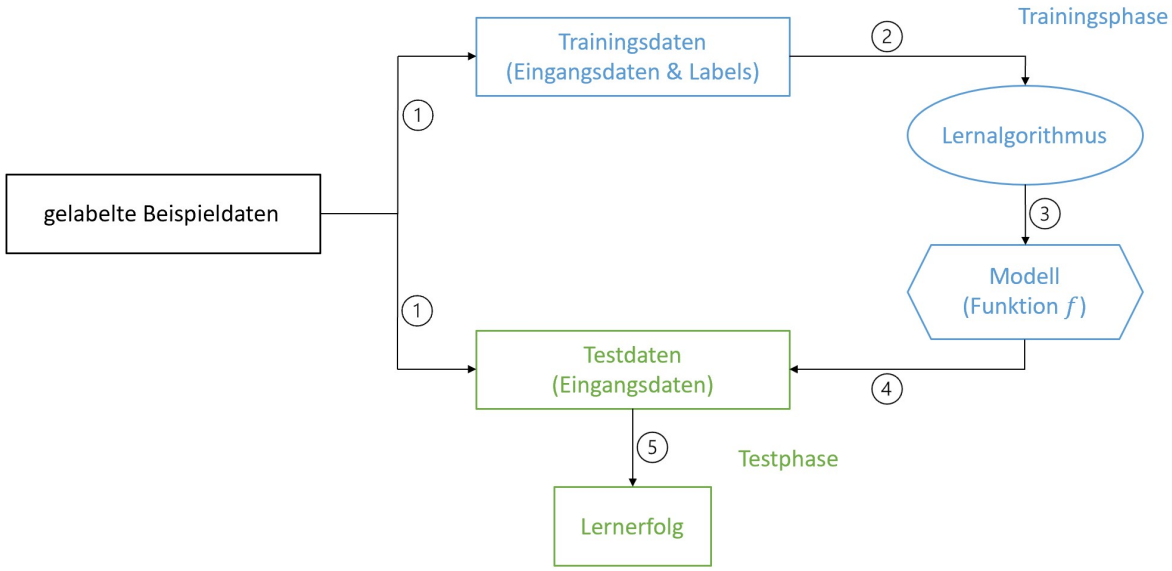


Abbildung 6: Prinzip des überwachten maschinellen Lernens

Die Problemstellungen des überwachten maschinellen Lernens werden in zwei Aufgabentypen unterteilt: die *Klassifikation* und die *Regression* (vgl. Frochte, 2021, S.22).

Bei einem *Klassifizierungsproblem* geht es darum, Datenpunkte verschiedenen Klassen zuzuordnen. Die Menge C aller Klassen ist eine abgeschlossene und diskrete Menge. Auf Basis des vorliegenden Trainingsdatensatzes X, der sowohl die Eingangsdaten $\vec{x}_i$, $i = 1, ..., m$ als auch die zugehörigen Klassen c_i, $i = 1, ..., m$ enthält, soll eine Funktion

$$f : X \to C \tag{3.3}$$

gelernt werden. Diese Funktion f soll unbekannten Datenpunkten, die nicht im Trainingsdatensatz enthalten sind, eine Klasse aus der Menge C zuordnen.
Das Beispiel des Spamfilters stellt ein solches Klassifizierungsproblem dar. Die Menge der Klassen C ist gegeben durch

$$C = \{\text{Spam}, \text{Ham}\}. \tag{3.4}$$

Anhand der Trainingsdaten wird dann eine Funktion f erlernt, die die Mail-Beispiele einer der beiden Klassen zuordnet. Nach der Trainingsphase können dann neue, unbekannte E-Mails als Spam oder nicht Spam (Ham) klassifiziert werden (s. Abb. 7).

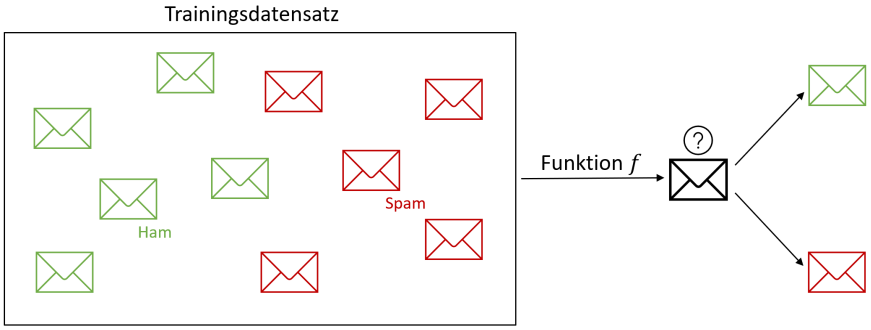


Abbildung 7: Klassifikation von E-Mails als Spam oder Ham

Eine weitere typische Problemstellung des überwachten Lernens ist die *Regression*. Der Unterschied zwischen den Klassifizierungs- und Regressionsproblemen liegt in der Menge C. Während die Werte der Menge C im Falle der Klassifikation nominal skaliert sind, stammen sie bei der Regression aus einem kontinuierlichen Bereich und sind metrisch skaliert (vgl. Frochte, 2021, S. 23).
Ein Beispiel für ein Regressionsproblem ist die Vorhersage des Preises eines Autos anhand verschiedener Merkmale (z. B. Alter, Marke, PS-Zahl, gefahrene Kilometer, etc.) (vgl. Géron, 2020, S. 10). Die vorhergesagten Preise verschiedener Autos lassen sich der Größe nach sortieren, sind also metrisch skaliert. Im Gegensatz dazu können die Klassen bei Klassifizierungsproblemen nicht geordnet werden. Eine E-Mail der Klasse Spam ist nicht „besser“ oder „schlechter“ als eine E-Mail der Klasse Ham.

Neben der Unterscheidung der Problemstellungen in Klassifikation und Regression lassen sich die Verfahren des überwachten Lernens auch anhand zweier Ansätze differenzieren: der *Eager Learner* und der *Lazy Learner* (vgl. Frochte, 2021, S. 24).

Beim *Eager Learner* wird in einer meist aufwendigen Trainingsphase ein *globales* Modell entwickelt. Auf Basis dieses Modells werden anschließend neue Eingangsdaten einem Wert aus der Menge C zugeordnet. Im Gegensatz zur Trainingsphase erfolgt die spätere Abfrage der erlernten Funktion relativ schnell (vgl. Frochte, 2021, S. 24). Dieser Ansatz ist in Abbildung 6 dargestellt.

Der *Lazy Learner* dagegen arbeitet mit einem *lokalen* Modell, das für jede einzelne Abfrage neu gebildet wird. Die Trainingsphase ist hier also weniger zeitintensiv als beim Eager Learner, die Abfrage jedoch teurer (vgl. Frochte, 2021, S. 24). Im Vergleich zum Eager Learner bietet der Lazy Learner den Vorteil, dass das lokale Modell für jede Abfrage passgenau gebildet werden kann und daher meist genauere bzw. bessere Ergebnisse liefert. Aus ökonomischen Gründen wird jedoch häufig auf Eager Learning-Ansätze zurückgegriffen. Die Eager Learner sind im Gegensatz zu den Lazy Learnern im Einsatz deutlich kostengünstiger, da die teure Modellentwicklung nur einmal in der Trainingsphasen erfolgt und die anschließenden Abfragen günstiger sind (vgl. Frochte, 2021, S. 24).

Unüberwachtes Lernen
Eine weitere Lernart des maschinellen Lernens ist das *unüberwachte Lernen* (engl. *unsupervised learning*). Im Unterschied zum überwachten Lernen besteht der Beispieldatensatz hier nur aus den Eingangsdaten, es wird also kein gelabelter Datensatz verwendet. Das Ziel des unüberwachten Lernens ist es, Strukturen und Muster in den Daten zu finden. Da die einzelnen Datenpunkte unmarkiert sind, lässt sich die Lösung des Computers nicht direkt beurteilen (vgl. Frochte, 2021, S. 25).
Ein typisches Beispiel für das unüberwachte Lernen ist die Einteilung von Kunden anhand ihres Kaufverhaltens beim Online-Shopping. Im Gegensatz zum überwachten Lernen werden die Kunden nicht einer bestimmten Klasse zugeordnet. Vielmehr geht es darum, die Gruppen anhand ihres ähnlichen Verhaltens zu bilden. Das „Label" der Gruppe spielt dabei keine Rolle (vgl. Frochte, 2021, S. 25).

Bestärkendes Lernen
Neben dem überwachten und dem unüberwachten Lernen gibt es noch eine dritte Lernart, die hier vorgestellt werden soll: das *bestärkende Lernen* (engl. *reinforcement learning*). Im Gegensatz zu den anderen beiden Lernarten ist der Ausgangspunkt hier kein gegebener Datensatz. Stattdessen sind die Algorithmen des bestärkenden Lernens fast immer agentenbasiert. Durch kontinuierliche Rückmeldungen in Form von Belohnungen und Bestrafungen soll ein Agent nach und nach eine (möglichst) optimale Strategie zur Lösung eines Problems finden (vgl. Frochte, 2021, S. 24).
Ein solcher Agent könnte zum Beispiel ein Putzroboter sein, der nach einer optimalen Strategie für die Reinigung eines Gebäudes in kürzester Zeit und mit möglichst wenig Energie sucht (vgl. Frochte, 2021, S. 24).

3.1.3. Klassifikation mit dem k-nächste-Nachbarn-Algorithmus

Das Problem der Aktivitätserkennung mit dem Smartphone stellt ein Klassifizierungsproblem dar, das im entwickelten Lernmodul mit Hilfe des *k-nächste-Nachbarn-Algorithmus (kNN-Algorithmus)* gelöst wird. Bei diesem Algorithmus handelt es sich um ein überwachtes maschinelles Lernverfahren, das nach dem Prinzip des Lazy Learners arbeitet. Die Grundidee des kNN-Algorithmus besteht darin, unbekannte Datenpunkte auf Basis der Eigenschaften (Klassen) der k nächstliegenden Datenpunkte, auch *Nachbarn* genannt, zu klassifizieren (vgl. Frochte, 2021, S. 122).

In das Koordinatensystem in Abbildung 8 sind 16 gelabelte Datenpunkte eingetragen, die durch die Ausprägungen zweier Merkmale, hier x und y, repräsentiert werden. Ziel ist es, einen unbekannten Datenpunkt, hier dargestellt durch das rote X, einer der drei Klassen zuzuordnen. Dazu werden die fünf nächstliegenden Datenpunkte betrachtet. Unter den fünf nächsten Nachbarn des unbekannten Datenpunkts befinden sich vier Datenpunkte der Klasse zwei (orangene Quadrate) und ein Datenpunkt der Klasse drei (grüne Kreise). Da die Klasse zwei unter den fünf nächsten Nachbarn am häufigsten vorkommt, wird der unbekannte Datenpunkt dieser Klasse zugeordnet.

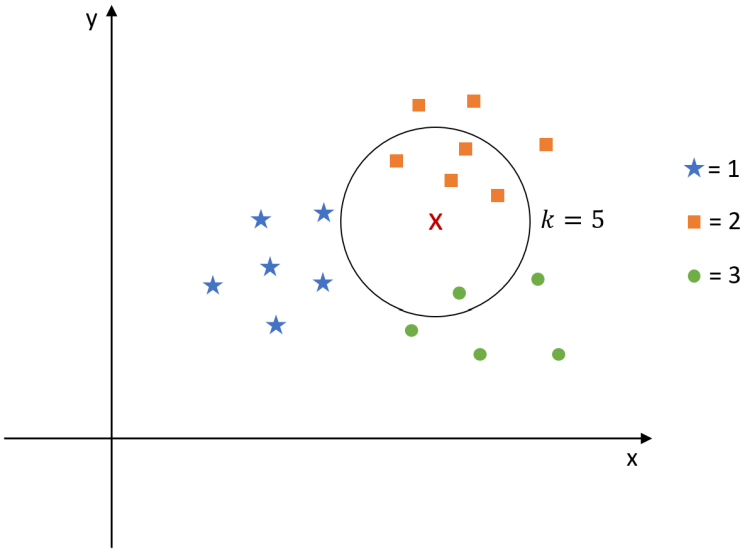


Abbildung 8: Grundidee des kNN-Algorithmus

Der kNN-Algorithmus ist ein Algorithmus, der nur auf wenigen Parametern beruht. Es werden lediglich die Anzahl der Nachbarn k und die verwendete Metrik zur Bestimmung der nächstgelegenen Nachbarn festgelegt. In dem Beispiel aus Abbildung 8 wurde die euklidische Metrik verwendet. Es ist durchaus möglich auch andere Metriken zur Bestimmung der nächstgelegenen Datenpunkte zu verwenden. Darauf wird in Abschnitt 3.2.2 ausführlicher eingegangen.

3.2. Mathematische Grundlagen

In diesem Abschnitt werden die mathematischen Grundlagen, die zur Bearbeitung des Lernmoduls benötigt werden, vorgestellt.

3.2.1. Statistische Kenngrößen

Zur Vorverarbeitung der aufgenommenen Sensordaten werden verschiedene statistische Kenngrößen verwendet. Bei statistischen Kenngrößen wird zwischen den *Maßen der zentralen Tendenz* (auch *Lagemaße* genannt) und den *Streuungsmaßen* unterschieden. Während die Lagemaße alle Messwerte einer Verteilung repräsentieren und ihre grobe Lage beschreiben, geben die Streuungsmaße Auskunft über die Variation der Messwerte (vgl. Planing, 2022). Da Streuungsmaße nicht Inhalt des Mathematikunterrichts sind, kommen im Lernmodul nur Lagemaße zum Einsatz. Im Folgenden werden die im Lernmodul eingesetzten Lagemaße vorgestellt.

Es seien $x_1, ..., x_n$ die Messwerte eines Sensors.

- **Minimum und Maximum einer Messreihe:**
  Das *Minimum* $x_{\min}$ beschreibt den kleinsten und das *Maximum* $x_{\max}$ den größten Wert unter den Messwerten $x_1, ..., x_n$. Es gilt:

$$x_{\min} := \min_{i=1,...,n} x_i \qquad \text{und} \qquad x_{\max} := \max_{i=1,...,n} x_i. \tag{3.5}$$

- **Arithmetisches Mittel einer Messreihe:**
  Das wohl bekannteste Lagemaß ist das arithmetische Mittel $\bar{x}$ von $x_1, ..., x_n$. Dieses ist definiert durch

$$\bar{x} := \frac{1}{n} \cdot (x_1 + ... + x_n) = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i. \tag{3.6}$$

- **Median einer Messreihe:**
  Beschreibt $x_{(j)}$ für $j = 1, ..., n$ den j-kleinsten Wert unter den Messwerten, so gilt insbesondere

$$x_{(1)} = x_{\min} \qquad \text{und} \qquad x_{(n)} = x_{\max}. \tag{3.7}$$

  Werden die Messwerte der Größe nach sortiert, ergibt sich eine Reihe der Messwerte

$$x_{(1)} \leq x_{(2)} \leq ... \leq x_{(n)}. \tag{3.8}$$

  Der *Median* beschreibt dann den mittleren Wert dieser geordneten Reihe an Messwerten. Es gilt

$$x_{1/2} := \begin{cases} x_{(\frac{n+1}{2})}, & \text{falls } n \text{ eine ungerade Zahl ist,} \\ \frac{1}{2} \cdot \left(x_{(\frac{n}{2})} + x_{(\frac{n}{2}+1)} \right), & \text{falls } n \text{ eine gerade Zahl ist.} \end{cases} \tag{3.9}$$

  Der Median teilt die Messwerte also in zwei gleichgroße Hälften, es sind mindestens 50 % der Messwerte kleiner oder gleich $x_{1/2}$ und mindestens 50 % größer oder gleich $x_{1/2}$. Im Gegensatz zum arithmetischen Mittel ist der Median robust gegenüber Ausreißern.

- **unteres und oberes Quartil einer Messreihe:**
  Eine Verallgemeinerung des Medians stellen die *p-Quartile* dar. Für eine Zahl p mit $0 < p < 1$ ist das p-Quartil gegeben durch

$$x_p := \begin{cases} x_{(\lfloor np+1 \rfloor)}, & \text{falls } np \notin \mathbb{N}, \\ \frac{1}{2} \cdot \left(x_{(np)} + x_{(np+1)} \right), & \text{falls } np \in \mathbb{N}. \end{cases} \tag{3.10}$$

  Besondere Fälle der p-Quartile sind das *untere Quartil* ($p = 0.25$) und das *obere Quartil* ($p = 0.75$). Sie sind die Mediane der unteren bzw. oberen Hälfte der geordneten Reihe an Messwerten (vgl. Henze, 2018, S. 27 ff.).

3.2.2. Abstandsmetriken

Beim kNN-Algorithmus werden unbekannte Datenpunkte anhand der Klassen der k nächstliegenden Datenpunkte klassifiziert. Hierzu ist es erforderlich, den Abstand zwischen zwei Datenpunkten zu bestimmen und zu definieren, was unter „nächstgelegen“ verstanden wird. Im Folgenden werden kurz die wichtigsten Inhalte der linearen Algebra wiederholt, die zur Abstandsberechnung benötigt werden.

Jeder Datenpunkt im Datensatz ist ein Vektor, in dem die Merkmale des Datenpunktes gespeichert sind. Der Abstand zwischen zwei Datenpunkten bzw. Vektoren lässt sich durch eine *Metrik* bestimmen.

Definition: Metrik auf $(\mathbb{V}, +)$ (vgl. Frochte, 2021, S. 110)
Sei $(\mathbb{V}, +)$ ein Vektorraum. Eine Abbildung $d : \mathbb{V} \times \mathbb{V} \to \mathbb{R}$ heißt *Metrik* auf $\mathbb{V}$, wenn für beliebige Elemente x, y und z aus $\mathbb{V}$ gilt:

- $d(x, x) = 0$.
- $d(x, y) = 0 \quad \Rightarrow \quad x = y$.
- $d(x, y) = d(y, x)$ (Symmetrie).
- $d(x, y) \leq d(x, z) + d(z, y)$ (Dreiecksungleichung).

Eine Metrik auf $\mathbb{V}$ kann über

$$d(x, y) = \|x - y\| \tag{3.11}$$

auch durch eine Norm auf $\mathbb{V}$ induziert werden (vgl. Frochte, 2021, S. 110).

Im entwickelten Lernmodul kommen verschiedene Metriken zur Berechnung des Abstandes zwischen zwei Datenpunkten zum Einsatz. Diese sollen im Folgenden am Beispiel des Abstandes zwischen den beiden Vektoren $\vec{x}$ und $\vec{y}$ mit den Einträgen x_i, $i = 1, ..., n$ und y_i, $i = 1, ..., n$ vorgestellt werden.

Euklidische Metrik:
Die *euklidische Metrik* wird von der euklidischen Norm

$$\|\vec{x}\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}, \tag{3.12}$$

induziert und ist definiert durch

$$d_{\text{Euklid}} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}. \tag{3.13}$$

Sie beschreibt die Länge bzw. den Betrag des Verbindungsvektors $\vec{d}$ der beiden Vektoren $\vec{x}$ und $\vec{y}$ (vgl. Chomboon et al., 2015, S. 281). Für den dreidimensionalen Fall ist dies in Abbildung 9 anhand des Abstands der beiden Punkte $P_1(x_1, y_1, z_1)$ und $P_2(x_2, y_2, z_2)$ beispielhaft dargestellt.

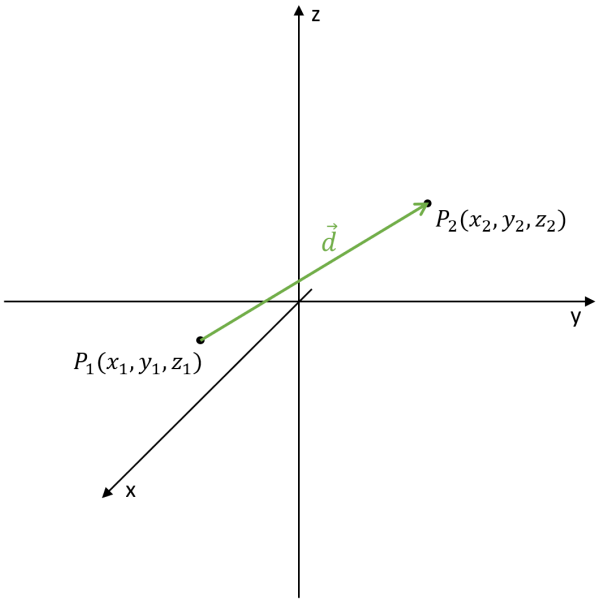


Abbildung 9: Abstand zwischen den beiden Punkten P_1 und P_2 mit der euklidischen Metrik im dreidimensionalen Fall

Manhattan-Metrik

Eine weitere Möglichkeit den Abstand zwischen zwei Vektoren zu bestimmen, ist durch die *Manhattan-Metrik* gegeben. Diese wird durch die Manhattan-Norm

$$\|\vec{x}\|_1 = \sum_{i=1}^{n} |x_i| \tag{3.14}$$

induziert und ist definiert durch (vgl. Chomboon et al., 2015, S. 281)

$$d_{\text{Manhattan}} = \sum_{i=1}^{n} |x_i - y_i|. \tag{3.15}$$

Der Name Manhattan-Metrik stammt von der schachbrettmusterartigen Anordnung der Gebäudeblöcke und dem orthogonalen Straßengitter in New Yorks Stadtteil Manhattan. Ein Taxifahrer kann die Entfernung zwischen zwei Adressen nur durch die Aneinanderreihung vertikaler und horizontaler Wegstücke überwinden (z. B. blaue, orangene und gelbe Linien in Abb. 10) (vgl. Wikipedia, 2021). Zum Vergleich ist in Abbildung 10 auch die euklidische Metrik (grüne Linie) dargestellt.

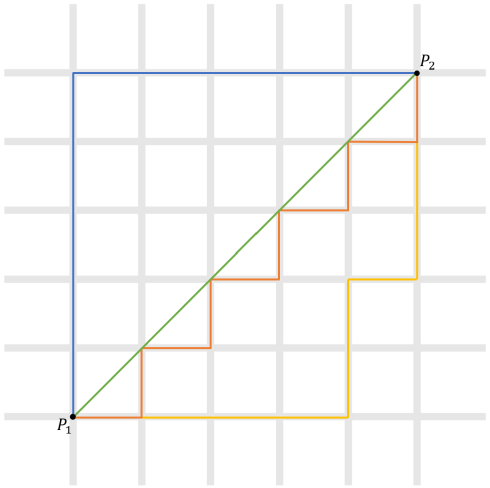


Abbildung 10: Abstand zwischen den beiden Punkten P_1 und P_2 mit der Manhattan-Metrik (blaue, orange und gelbe Linien) und der euklidischen Metrik (grüne Linie) im zweidimensionalen Fall

Kosinus-Metrik:
Der Abstand zwischen zwei Vektoren kann auch mit Hilfe des eingeschlossenen Winkels θ zwischen den beiden Vektoren definiert werden (vgl. Chomboon et al., 2015, S. 282). In Abbildung 11 ist dies beispielhaft für den dreidimensionalen Fall und den Winkel zwischen den Ortsvektoren der beiden Punkte $P_1(x_1, y_1, z_1)$ und $P_2(x_2, y_2, z_2)$ beispielhaft dargestellt.

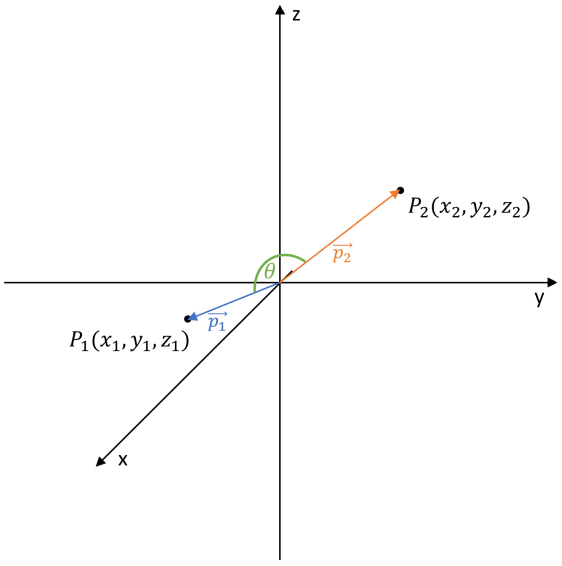


Abbildung 11: Winkel θ zwischen den Ortsvektoren der beiden Punkte P_1 und P_2 im dreidimensionalen Fall

Mit der *Kosinus-Metrik*, die über das Standardskalarprodukt und die euklidische Norm hergeleitet werden kann, ist der Abstand zwischen zwei Vektoren $\vec{x}$ und $\vec{y}$ gegeben durch (vgl. Chomboon et al., 2015, S. 282)

$$d_{\text{Kosinus}} = 1 - \cos(\theta) = 1 - \frac{\vec{x} \cdot \vec{y}}{|\vec{x}| \cdot |\vec{y}|}. \tag{3.16}$$

Tschebyschew-Metrik:
Eine weitere Metrik, die im Lernmodul zum Einsatz kommt, ist die *Tschebyschew-Metrik* bzw. *Maximumsmetrik*. Sie wird von der Maximumsnorm

$$\|\vec{x}\|_{\max} = \max_{i=1,\dots,n} |x_i|. \tag{3.17}$$

induziert und definiert den Abstand zwischen zwei Vektoren $\vec{x}$ und $\vec{y}$ wie folgt (vgl. Chomboon et al., 2015, S. 282):

$$d_{\text{Tschebyschew}} = \max_{i=1,\dots,n} |x_i - y_i|. \tag{3.18}$$

3.2.3. Qualitätsmaße zur Bewertung von Klassifikationsergebnissen

Nachdem bei einem Klassifizierungsproblem in der Trainingsphase ein Modell erlernt wurde, wird in der Testphase anhand der Testdaten die Generalisierungsfähigkeit und damit der Lernerfolg des Modells überprüft. Zur quantitativen Bewertung der Güte eines Modells werden verschiedene *Qualitätsmaße* verwendet. Diese sollen am Beispiel eines binären Klassifizierers mit den beiden Klassen „positiv“ und „negativ“ erläutert werden (vgl. Dittrich, 2014, S. 24).

Die Ergebnisse der binären Klassifikation mit m Testdatenpunkten können in Form einer *Wahrheitsmatrix* bzw. *Konfusionsmatrix* dargestellt werden (s. Tab. 2).

Tabelle 2: Wahrheitsmatrix für eine binäre Klassifikation mit m Testdatenpunkten (vgl. Dittrich, 2014, S. 24)

	positiv klassifiziert	**negativ klassifiziert**	**Summe**
tatsächlich positiv	richtig positiv (RP)	falsch negativ (FN)	$RP + FN$
tatsächlich negativ	falsch positiv (FP)	richtig negativ (RN)	$FP + RN$
Summe	$RP + FP$	$FN + RN$	m

Mit Hilfe dieser Wahrheitsmatrix können im Anschluss verschiedene Qualitätsmaße berechnet werden (vgl. Dittrich, 2014, S. 24). Im Lernmodul werden drei Qualitätsmaße zur Bewertung der Klassifikationsergebnisse verwendet: die *Genauigkeit*, die *Fehlerrate* und die *Präzision*.

Die *Genauigkeit* (engl. *accuracy*) gibt das Verhältnis der richtig klassifizierten Datenpunkte zu allen Datenpunkten an (vgl. Dittrich, 2014, S. 24). Es gilt also:

$$\text{accuracy} = \frac{\text{Anzahl der richtig klassifizierten Datenpunkte}}{\text{Anzahl aller Datenpunkte}} = \frac{RP + RN}{m}. \tag{3.19}$$

Im Gegensatz zur Genauigkeit gibt die *Fehlerrate* das Verhältnis der falsch klassifizierten Datenpunkte zu allen Datenpunkten an (vgl. Dittrich, 2014, S. 25). Es gilt:

$$\text{error rate} = \frac{\text{Anzahl der falsch klassifizierten Datenpunkte}}{\text{Anzahl aller Datenpunkte}} = \frac{FP + FN}{m}. \tag{3.20}$$

Die *Präzision* wird im Gegensatz zu den beiden vorherigen Qualitätsmaßen nicht für die gesamte Klassifikation berechnet, sondern für jede Klasse einzeln. Sie gibt das Verhältnis der richtig als Klasse z klassifizierten Datenpunkte zu allen als Klasse z klassifizierten Datenpunkten an (vgl. Dittrich, 2014, S. 24). Für die Klasse „positiv“ ergibt sich dann

$$\text{precision}_{\text{pos}} = \frac{\text{Anzahl der richtig als positiv klassifizierten Datenpunkte}}{\text{Anzahl aller als positiv klassifizierten Datenpunkte}} = \frac{RP}{RP + FP} \tag{3.21}$$

und für die Klasse „negativ“

$$\text{precision}_{\text{neg}} = \frac{\text{Anzahl der richtig als negativ klassifizierten Datenpunkte}}{\text{Anzahl aller als negativ klassifizierten Datenpunkte}} = \frac{RN}{FN + RN}. \tag{3.22}$$

Für eine gute Klassifikation sollten die Genauigkeit und die Präzision der einzelnen Klassen maximiert und die Fehlerrate minimiert werden (vgl. Dittrich, 2014, S. 25).

3.3. Grundlagen der Aktivitätserkennung

In diesem Abschnitt werden die Grundlagen der Aktivitätserkennung erläutert. Zunächst wird ein kurzer Überblick über die Geschichte der Aktivitätserkennung gegeben. Im Anschluss wird näher auf die Sensoren sowie die Aktivitäten eingegangen und der Prozess der Aktivitätserkennung erläutert. Den Abschluss bilden Herausforderungen und Probleme sowie Anwendungen und Einsatzgebiete der Aktivitätserkennung.

3.3.1. Geschichte der Aktivitätserkennung

Einer der ersten Ansätze der Aktivitätserkennung war die Bildsensorik. Es wurde versucht mit Hilfe von Kameras die Aktivitäten von Personen zu bestimmen. Später wurde der Fokus auf die Trägheitssensorik mittels bewegungsabhängigen Sensoren, die am Körper des Nutzers angebracht waren, gelegt. Ein Nachteil dieses Ansatzes war, dass sich die Nutzer oft durch die Sensoren gestört und in ihrem Alltag beeinträchtigt fühlten. Heutzutage können Mobiltelefone zur Aktivitätserkennung genutzt werden. Zu

Beginn wurden hauptsächlich GSM-Signale[7] verwendet, um einfache Fortbewegungen zu erkennen. Auch externe Sensoren, die am Körper des Nutzers positioniert und mit dem Smartphone verbunden waren, kamen erneut zum Einsatz. Durch die Weiterentwicklung und Integration zahlreicher Sensoren stellen Smartphones mittlerweile eine geeignete Plattform für die Aktivitätserkennung dar, sodass keine externen Sensoren mehr benötigt werden (vgl. Incel et al., 2013, S. 145).

3.3.2. Sensoren

Sensoren liefern die Rohdaten, die zur Aktivitätserkennung benötigt werden. Die unterschiedlichen Sensoren lassen sich in drei Kategorien einteilen:

- **Videosensoren:**
  Bei den Videosensoren handelt es sich um Kameras, die an festen Orten (z. B. am Ein- und Ausgang von Gebäuden) installiert sind. Sie liefern Informationen über das Aussehen und die Handlungen von Personen und werden vor allem bei der Überwachung sowie der Terrorismus- und Verbrechensbekämpfung eingesetzt (vgl. Su et al., 2014, S. 236).

- **Umgebungsbasierte Sensoren:**
  Umgebungsbasierte Sensoren werden verwendet, um die Interaktion eines Nutzers mit seiner Umgebung zu erkennen. Zu den umgebungsbasierten Sensoren zählen unter anderem WLAN- und Bluetooth-Sensoren. Diese werden vor allem in Innenräumen, wie zum Beispiel Bürogebäuden eingesetzt. Sie zeichnen den Aufenthalt sowie die Interaktion zwischen Nutzern und anderen mit Sensoren ausgestatteten Geräten an bestimmten Orten auf (vgl. Su et al., 2014, S. 236).

- **Tragbare Sensoren:**
  Bei den tragbaren Sensoren handelt es sich um sehr kleine mobile Sensoren, die während den täglichen Aktivitäten am Körper eines Nutzers getragen werden können. Sie sind in der Lage den Bewegungszustand eines Nutzers aufzuzeichnen. Dazu zählen zum Beispiel die Bewegungsrichtung und die Geschwindigkeit. Der Großteil dieser tragbaren Sensoren ist heutzutage in Smartphones eingebaut (vgl. Su et al., 2014, S. 236).

Die meisten Forschungsansätze im Bereich der Aktivitätserkennung fokussieren sich auf tragbare Sensoren, die in Smartphones integriert sind. Zu den wichtigsten dieser Sensoren zählen der GPS-Sensor, das Accelerometer, das Gyroskop und das Magnetometer. Mit Hilfe des *GPS-Sensors* können der Standort und die Geschwindigkeit eines Nutzers bestimmt werden (vgl. Incel et al., 2013, S. 147). Das *Accelerometer* (*Beschleunigungsmesser*) erfasst die Beschleunigungsvorgänge des Smartphones in allen drei Raumdimensionen x, y und z (s. Abb. 12). Um die Rotationsrate des Smartphones bestimmen zu können, misst das *Gyroskop* die Roll-, Nick- und Gierbewegung des

[7] Die Abkürzung GSM steht für Global System for Mobile Communications und ist ein 1990 eingeführter Mobilfunkstandard für volldigitale Mobilfunknetze (vgl. Wikipedia, 2022a).

Smartphones entlang der x-, y- und z-Achse. Im Bereich der Aktivitätserkennung wird das Gyroskop auch zur Bestimmung der Orientierung des Smartphones genutzt. Das Magnetometer arbeitet wie ein Kompass und bestimmt die Richtung des Smartphones in Bezug auf den Nord-Pol der Erde mit Hilfe von Magnetismus. Bei der Aktivitätserkennung liefert das Magnetometer Informationen über mögliche Richtungsänderungen eines Nutzers zum Beispiel beim Gehen (vgl. Su et al., 2014, S. 237).

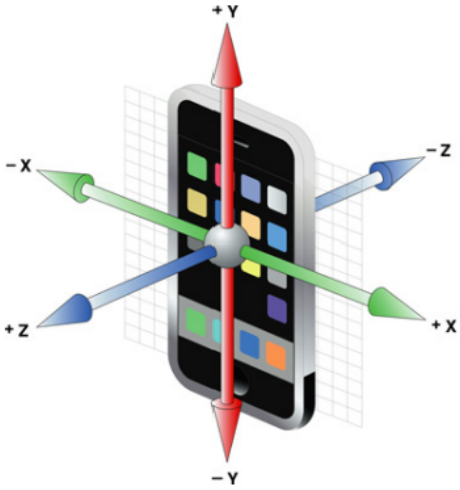


Abbildung 12: Die drei Achsen des Accelerometers (entnommen aus Dittrich, 2014, S. 26)

Für die menschliche Aktivitätserkennung hat sich hauptsächlich das Accelerometer durchgesetzt, da es den Bewegungszustand eines Nutzers aufzeichnet. Beginnt ein Nutzer während des Gehens zu Joggen, so wird sich dies unmittelbar in den Beschleunigungswerten der vertikalen Achse zeigen (vgl. Su et al., 2014, S. 237). Zusammen mit dem Accelerometer werden häufig noch das Gyroskop und das Magnetometer verwendet, um die Lage und Neigung des Smartphones zu bestimmen (vgl. Dittrich, 2014, S. 26).

Um neben den Aktivitäten auch Informationen über die Umgebung, die soziale Interaktion und den Standort eines Nutzers gewinnen zu können, werden weitere Sensoren wie zum Beispiel Mikrophone, Kameras, Mobilfunk- und WLAN-Schnittstellen sowie Näherungs- und Umgebungslichtsensoren verwendet (vgl. Incel et al., 2013, S. 146).

3.3.3. Aktivitäten

Erste Arbeiten im Bereich der Aktivitätserkennung beschäftigten sich mit einer sehr oberflächlichen Erkennung der Aktivitäten im Zusammenhang mit Standortinformationen. Dazu zählten zum Beispiel zu Hause sein oder im Büro bei der Arbeit sein.

Diese Schlussfolgerungen gaben aber keine Auskunft über die genauen Aktivitäten eines Nutzers (vgl. Incel et al., 2013, S. 148).

Durch die Weiterentwicklung der Sensoren und die Ausstattung der Smartphones mit immer leistungsstärkeren Sensoren ist es mittlerweile möglich, genauere Informationen über die Aktivitäten eines Nutzers zu gewinnen. Mit Hilfe des Accelerometers kann zum Beispiel erkannt werden, dass ein Nutzer sitzt. Über das Mikrophon kann zusätzlich festgestellt werden, dass sich der Nutzer in einer Konversation befindet. Der Bluetooth-Sensor liefert die Information, dass sich ein Arbeitskollege in der Nähe befindet. Anhand dieser Daten kann geschlussfolgert werden, dass sich der Nutzer in einem Meeting im Büro befindet. Außerdem ist es mittlerweile möglich komplexe sportliche Aktivitäten wie Radfahren, Fußballspielen, Nordic Walking, Rudern und Laufen zu unterscheiden. Darüber hinaus können sowohl Aktivitäten des alltäglichen Lebens wie Frühstücken und Einkaufen als auch gefährliche Situationen wie Stürze erkannt werden (vgl. Incel et al., 2013, S. 148).

Die meisten Studien konzentrieren sich jedoch auf fünf grundlegende Bewegungsaktivitäten, darunter Sitzen, Stehen, Gehen, Laufen und Treppen auf- und absteigen, da diese Aktivitäten von vielen Menschen häufig im Alltag ausgeführt werden (vgl. Kwapisz et al., 2010, S. 76).

3.3.4. Prozess der Aktivitätserkennung

Im Folgenden wird der Prozess der Entwicklung eines Modells zur Aktivitätserkennung erläutert.

Datenaufnahme

Der erste Schritt bei der Entwicklung eines Modells zur Aktivitätserkennung ist die Datenaufnahme. Vorher muss jedoch festgelegt werden, mit welchen Sensoren und zu welchen Aktivitäten Daten aufgenommen und welche Abtastrate verwendet werden soll.

Die *Abtastrate* ist eine wichtige Stellschraube der Aktivitätserkennung und gibt an, wie viele Messwerte pro Sekunde aufgenommen werden. Sie wird in der Einheit Hertz (Hz) angegeben. Eine hohe Abtastrate liefert auf der einen Seite viele Informationen, kann aber auf der anderen Seite auch mehr Rauschen in den Daten verursachen (vgl. Su et al., 2014, S. 239). In der Literatur werden verschiedene Abtastraten zwischen 20 Hz und 100 Hz verwendet (vgl. Kwapisz et al., 2010, S. 75; Ustev et al., 2013, S. 1430).

Nachdem die Aktivitäten und die Abtastrate festgelegt wurden, können die Daten mit den gewählten Sensoren aufgenommen werden. Dabei sollte darauf geachtet werden, dass die Daten von möglichst vielen unterschiedlichen Personen stammen. Außerdem

sollten für alle Aktivitäten möglichst gleich viele Daten existieren, sodass keine Klasse übertrainiert wird (vgl. Dittrich, 2014, S. 18 f.).

Datenvorverarbeitung

Auf die Datenaufnahme folgt die Datenvorverarbeitung. Im Rahmen der Datensegmentierung wird der dichte Strom an Sensordaten in kleinere Zeitfenster, sogenannte *Windows* unterteilt. Neben der Abtastrate ist auch die Größe der Windows eine wichtige Stellschraube der Aktivitätserkennung. Kleine Fenstergrößen benötigen weniger Ressourcen und ermöglichen so eine schnelle Erkennung der Aktivitäten, während größere Fenster aufgrund der Einbeziehung einer größeren Datenmenge die Bestimmung komplexer Aktivitäten gestatten. Die Fenstergröße sollte also auf den jeweiligen Anwendungsfall abgestimmt sein. In der Literatur finden sich verschiedene Fenstergrößen zwischen zwei und zehn Sekunden (vgl. Kwapisz et al., 2010, S. 75; Ustev et al., 2013, S. 1430).

Neben der Datensegmentierung zählt auch der Prozess der Merkmalsextraktion zur Datenvorverarbeitung. In dieser Phase wird die große Menge an Rohdaten auf eine kleinere Menge von möglichst aussagekräftigen Werten (sogenannte *Features*) reduziert. Diese Features werden für die vorher eingeteilten Windows berechnet und sollen die Eigenschaften des Signals bestmöglich beschreiben (vgl. Incel et al., 2013, S. 150). In den meisten Studien zur Aktivitätserkennung werden hauptsächlich zeitabhängige Features verwendet. Um die Ergebnisse der Klassifikation zu verbessern, kommen zusätzlich auch frequenzabhängige Features zum Einsatz (vgl. Dittrich, 2014, S. 27 f.). In Tabelle 3 sind verschiedene zeit- und frequenzabhängige Features, die bei der Aktivitätserkennung genutzt werden, aufgelistet. Die zeitabhängigen Features beschreiben die grundlegenden Statistiken eines Datensegments und werden meist für die Werte der einzelnen Achsen sowie den Betrag des Beschleunigungsvektors berechnet. Mit Hilfe der frequenzabhängigen Features kann die Periodizität des Signals beschrieben werden. Sie werden in der Regel auf Grundlage der Fast Fourier Transformation (FFT) berechnet (vgl. Su et al., 2014, S. 240).

Tabelle 3: Beispiele für zeit- und frequenzabhängige Features

zeitabhängige Features	**frequenzabhängige Features**
- Mittelwert	- erste zehn FFT-Koeffizienten
- Minimum und Maximum	- Energie
- Standardabweichung	- Entropie
- Varianz	- Zeit zwischen den Ausschlägen des Signals
- Korrelation	- Nulldurchgangsrate
- Median	- ...
- unteres und oberes Quartil	
- ...	

Klassifikation

Auf die Datenvorverarbeitung folgt die Entwicklung eines Klassifikationsalgorithmus. Bei der Klassifikation sollen die für die Daten der einzelnen Windows berechneten Features den unterschiedlichen Aktivitäten zugeordnet werden. Neben dem im Lernmodul verwendeten kNN-Algorithmus werden weitere überwachte maschinelle Lernverfahren zur Klassifikation von Sensordaten eingesetzt. In der Literatur finden sich:

- Naive Bayes Klassifikatoren
- Entscheidungsbäume
- Stützvektormethode
- Verdeckte Markovmodelle
- Neuronale Netze
- etc.

Dabei zeigt der kNN-Algorithmus im Vergleich zu den anderen Algorithmen vielfach die besten Ergebnisse (vgl. Mohsen et al., 2022, S. 305).

Evaluation

In der Testphase wird überprüft, ob die über das maschinelle Lernverfahren zugeordneten Aktivitäten mit den tatsächlichen Aktivitäten, die ein Nutzer ausführt bzw. ausgeführt hat, übereinstimmen. Um den Lernerfolg des Systems bewerten zu können, kommen verschiedene Qualitätsmaße (s. Abschn. 3.2.3) zum Einsatz (vgl. Incel et al., 2013, S. 151).

3.3.5. Herausforderungen und Probleme der Aktivitätserkennung

Die Aktivitätserkennung anhand der in Smartphones eingebauten Sensoren steht noch vor mehreren ungelösten Problemen und Herausforderungen, von denen einige im Folgenden vorgestellt werden.

Subjektive Empfindlichkeit

Die Genauigkeit der Aktivitätserkennung ist stark von den einzelnen Nutzern abhängig. Jeder Mensch hat seine eigenen Gewohnheiten und Verhaltensweisen und führt daher Bewegungen im Vergleich zu anderen Nutzern unterschiedlich aus (vgl. Su et al., 2014, S. 242). Um mögliche Fehlklassifikationen vorzubeugen, sollten die Trainingsdaten aus so vielen Daten wie möglich von so vielen unterschiedlichen Nutzern wie möglich zusammengesetzt sein. Dem stehen jedoch die begrenzten Speicher- und Rechenressourcen eines Smartphones gegenüber.

Standortempfindlichkeit

Die aufgenommenen Sensordaten, besonders die des Accelerometers, sind stark von der Ausrichtung und der Position der Sensoren am Körper des Nutzers abhängig. Hält eine Person das Smartphone in der Hand während sie geht und liest dabei (Chat-) Nachrichten, so unterscheiden sich die aufgenommenen Sensordaten deutlich von den Daten, die aufgenommen werden, wenn sich das Handy in der Hosentasche befindet. Zur Lösung dieses Problems können weitere Sensoren (z. B. das Gyroskop und das Magnetometer) einbezogen werden, um die Ausrichtung und Position der Sensoren vorab bestimmen zu können bzw. die Sensordaten in Erdkoordinaten umzurechnen (vgl. Su et al., 2014, S. 242 f.).

Energie- und Ressourcenbeschränkung

Der Einsatz vieler Sensoren wirkt sich zum einen auf die Akkulaufzeit und zum anderen auf den Speicherplatz aus (vgl. Dittrich, 2014, S. 31 f.). Auch die Ausführung des Klassifikationsalgorithmus kann aufgrund der geringen Rechenressourcen eines Smartphones im Vergleich zu einem leistungsstarken Rechner zu Problemen führen. Diese Herausforderungen könnten durch eine Auslagerung der Klassifikation auf einen externen Server zumindest teilweise bewältigt werden (vgl. Incel et al., 2013, S. 152 f.).

Komplexität der Aktivitäten

Werden mehrere Aktivitäten gleichzeitig ausgeführt, beispielsweise Sitzen und Frühstücken, sind diese schwer zu unterscheiden bzw. zu erkennen. Darüber hinaus stellt auch der Übergang zwischen zwei Aktivitäten ein Problem für die Aktivitätserkennung dar (vgl. Su et al., 2014, S. 243).

3.3.6. Anwendungen und Einsatzgebiete der Aktivitätserkennung

Die menschliche Aktivitätserkennung kann in vielen Bereichen von Nutzen sein. Einerseits kann sie enorme Vorteile für den Endnutzer mit sich bringen, andererseits kann sie auch für einzelne Unternehmen und die Gesellschaft nützlich sein. Im Folgenden werden verschiedene Anwendungen und Einsatzgebiete der Aktivitätserkennung aufgeführt.

Anwendungen für den Endnutzer

- **Fitness-Tracking:**
  Die Aktivitätserkennung kann im Rahmen des Fitness-Trackings die Schritte zählen, die täglich verbrauchten Kalorien angeben, die zurückgelegte Strecke messen und die Anzahl der gestiegenen Treppenstufen anzeigen. Diese Informationen können zur Überprüfung des aktuellen Fitnessstands genutzt werden und zu einer gesünderen Lebensweise motivieren (vgl. Dittrich, 2014, S. 29).

- **Gesundheitsüberwachung:**
  Durch eine kontinuierliche Überwachung der Gewohnheiten und der täglichen Aktivitäten von Patienten kann die Diagnose durch den Arzt verbessert werden. Zudem können die aufgenommenen Daten einen Ausgangspunkt für weitere medizinische Untersuchungen darstellen (vgl. Dittrich, 2014, S. 29).

- **Sturzerkennung:**
  Die Sturzerkennung anhand der Sensoren eines Smartphones ermöglicht sowohl das automatische Absetzen eines Notrufs als auch die Ortung des Handys, sodass die verletzte Person schnell gefunden und gerettet werden kann (vgl. Incel et al., 2013, S. 149).

- **Kontextbezogenes Verhalten:**
  Das Smartphone kann auf die Aktivitäten des Nutzers reagieren, indem es zum Beispiel die Schrift vergrößert, wenn der Nutzer im Gehen (Chat-)Nachrichten liest. Dadurch kann der Nutzer in seiner aktuellen Situation und Aktivität durch das Smartphone unterstützt werden (vgl. Dittrich, 2014, S. 29).

Anwendungen für Unternehmen und Dritte

- **Gezielte Werbung:**
  Durch die Aufzeichnung der Aktivitäten eines Nutzers kann die Werbung auf die täglichen Gewohnheiten des Nutzers abgestimmt werden. Dadurch wirkt sie weniger aufdringlich, denn sie ist für die Aktivitäten und Interessen des Nutzers relevant. Durch diese zielgerichtete Werbung soll eine größtmögliche Effektivität aus der Werbung gezogen werden können (vgl. Dittrich, 2014, S. 29 f.).

- **Forschungsplattformen:**
  Im Bereich der Forschung wird häufig zuerst eine ausreichende Menge an Daten benötigt. Daher sind Forscher in vielen Bereichen, vom Marketing bis zum Gesundheitswesen, an der Sammlung von Aktivitätsdaten interessiert (vgl. Dittrich, 2014, S. 30).

- **Unternehmensführung:**
  Im Bereich der Unternehmensführung kann die Aktivitätserkennung bei der Mitarbeiterverwaltung und bei der Abrechnung der Arbeitszeiten unterstützend eingesetzt werden (vgl. Lockhart et al., 2012, S. 3).

- **Versicherungen:**
  Manche Versicherungsgesellschaften bieten ihren Versicherungsnehmern einen Rabatt zum Beispiel bei der Autoversicherung an, wenn eine sichere Fahrweise durch elektronische Geräte wie Smartphones erkannt wird (vgl. Lockhart et al., 2012, S. 3).

Anwendungen für Gruppen und Crowds

- **Soziale Netzwerke:**
  Aktivitätsbasierte soziale Netzwerke können ihren Nutzern einen Freundeskreis anhand des Aufenthaltsortes sowie den ähnlichen Aktivitätsmustern vorschlagen (vgl. Lockhart et al., 2012, S. 4).

- **Crowd-Sourcing:**
  Im Rahmen des aktivitätsbezogenen Crowd-Sourcings können Gegenden identifiziert werden, in denen bestimmte Aktivitäten besonders häufig ausgeführt werden. Einem Nutzer der gerne Fahrrad fährt, können dann beliebte Fahrradstrecken vorgeschlagen werden. Darüber hinaus können diese Systeme auch erkennen, wenn in einer Region die Aktivitäten stark von den normalerweise beobachteten Aktivitäten abweichen. Dadurch können mögliche Unfälle und Katastrophen schneller erkannt werden (vgl. Lockhart et al., 2012, S. 4).

Die Aktivitätserkennung kann jedoch trotz der zahlreichen Anwendungen und Einsatzgebiete auch kritisch angesehen werden, gerade hinsichtlich des Datenschutzes und der Privatsphäre sowie der Überwachung durch einen totalitären Staat.

3.4. Umsetzung im Lernmodul

In diesem Abschnitt werden die einzelnen Modellierungsschritte, die im Lernmodul bei der Erstellung eines Modells zur Aktivitätserkennung durchlaufen werden, vorgestellt. Die Schritte orientieren sich an dem in Abschnitt 3.3.4 vorgestellten Prozess der Aktivitätserkennung.

Im Vorfeld wurden Daten zu verschiedenen Aktivitäten mit Hilfe der App phyphox[8] aufgenommen. Während der Datenaufnahme war das Handy immer gleich positioniert (in der rechten vorderen Hosentasche). Es wurden Daten des Accelerometers, des Gyroskops und des Magnetometers aufgezeichnet. Zur Vereinfachung werden im Lernmodul jedoch nur die Daten des Accelerometers verwendet. Jeder Aktivität und jeder Testperson, die Daten zu den einzelnen Aktivitäten aufgenommen hat, wurden eine natürliche Zahl zugeordnet, die sogenannte *ActivityID* bzw. *UserID*.

3.4.1. Erkunden des Datensatzes

Zunächst wird der Datensatz, der dem Lernmodul zu Grunde liegt, vorgestellt. Jeder Datenpunkt in diesem Datensatz besteht aus der UserID und der ActivityID sowie der Zeit ab Beginn der Datenaufnahme in Sekunden und den Beschleunigungswerten der drei Raumdimensionen x, y und z in $\frac{\mathrm{m}}{\mathrm{s}^2}$ (s. Abb. 13).

[8] Phyphox ist eine an der RWTH Aachen entwickelte App, die es ermöglicht mit Hilfe der Sensoren des Smartphones physikalische Experimente durchzuführen (`https://phyphox.org/de/home-de/`, letzter Aufruf: 15.09.2022).

UserID	ActivityID	Time (s)	x (m/s^2)	y (m/s^2)	z (m/s^2)
1	1	0.028140	5.480853	-4.508624	-6.869365
1	1	0.053292	5.488187	-4.504134	-6.870263
1	1	0.078443	5.494324	-4.496799	-6.872658
1	1	0.103595	5.492229	-4.508475	-6.871311
1	1	0.128747	5.472320	-4.499044	-6.856192
...	...	...	...	...	...

Abbildung 13: Screenshot eines Ausschnittes des Datensatzes aus dem digitalen Lernmaterial

Insgesamt wurden Daten zu fünf Aktivitäten von zehn Testpersonen (fünf weiblich und fünf männlich) aufgenommen. Die jüngste Testperson war 18 Jahre alt und die älteste Testperson war 52 Jahre alt. Die Aktivitäten mit der zugehörigen ActivityID sind:

- ActivityID 1: Sitzen,
- ActivityID 2: Stehen,
- ActivityID 3: Gehen,
- ActivityID 4: Laufen / Joggen,
- ActivityID 5: Treppen auf- und absteigen.

In Abbildung 14 sind die Sensordaten der einzelnen Aktivitäten für drei Sekunden graphisch dargestellt. Es ist zu erkennen, dass die Beschleunigungswerte der einzelnen Raumdimensionen bei den Aktivitäten Sitzen und Stehen nahezu auf einer Gerade liegen, während bei den anderen Aktivitäten deutliche Schwankungen zu erkennen sind. Das liegt daran, dass sich das Smartphone beim Sitzen und Stehen in Ruhe befindet und daher die Beschleunigung konstant ist. Die meisten und die stärksten Ausschläge sind bei der Aktivität Laufen zu erkennen, da hier die meisten Schritte innerhalb einer gewissen Zeitspanne gemacht werden und sich am stärksten vom Boden abgedrückt wird. Dadurch ist die Beschleunigung bei einem Schritt im Vergleich zu den Aktivitäten Gehen und Treppen auf- und absteigen am größten. Außerdem lässt sich erkennen, dass die größten Beschleunigungswerte auf der y-Achse zu finden sind. Der Grund dafür ist die Erdbeschleunigung. Diese ist zum Erdmittelpunkt hin gerichtet und wirkt daher bei allen Aktivitäten bis auf Sitzen auf die y-Achse des Accelerometers, denn das Smartphone befand sich während der Datenaufnahme senkrecht in der vorderen rechten Hosentasche.

Jede Aktivität wurde bei der Datenaufnahme von den einzelnen Testpersonen 180 Sekunden, also drei Minuten ausgeführt. Innerhalb dieser 180 Sekunden wurden ca. 7150 Datenpunkte aufgenommen. Somit wurden die Daten mit einer Abtastrate von ca.

$$\frac{7150}{180\text{s}} \approx 40\text{Hz} \tag{3.23}$$

aufgenommen.

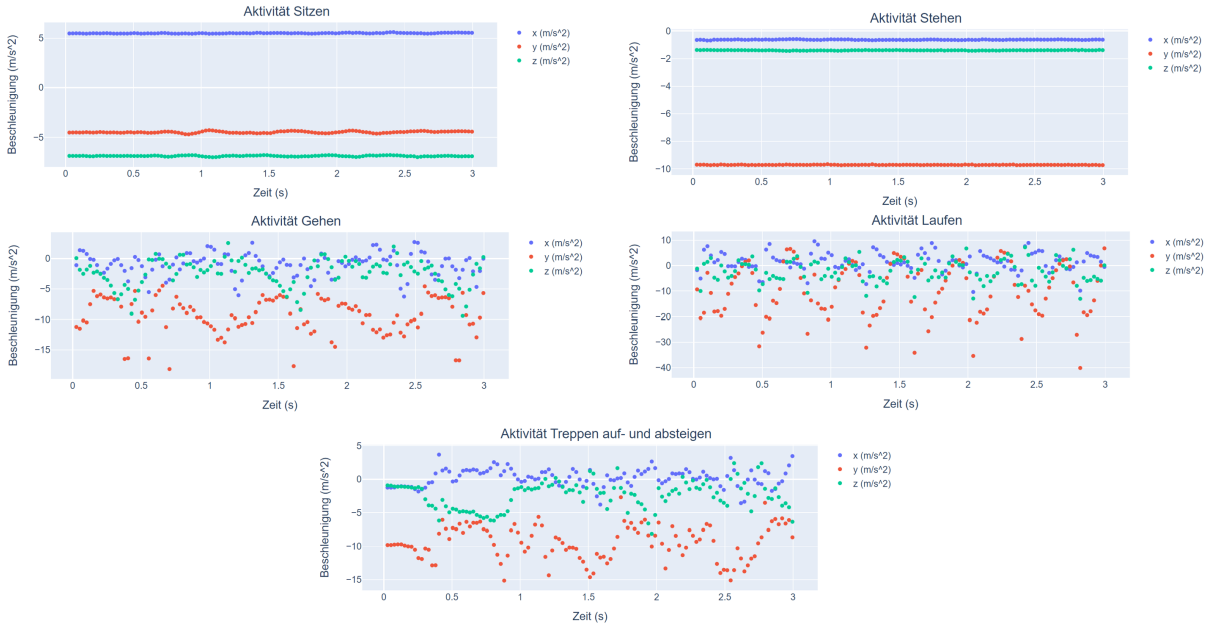


Abbildung 14: Graphische Darstellung der Sensordaten der einzelnen Aktivitäten für drei Sekunden

3.4.2. Vorverarbeitung der Daten

Zu Beginn der Vorverarbeitung der Sensordaten werden die Beschleunigungswerte der drei Achsen kombiniert, indem der Betrag des Beschleunigungsvektors

$$\vec{a} = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} \tag{3.24}$$

berechnet wird, wobei a_x, a_y und a_z die Beschleunigungswerte der einzelnen Raumdimensionen x, y und z bezeichnen. Der Betrag des Beschleunigungsvektors ist definiert durch

$$|\vec{a}| = \sqrt{a_x^2 + a_y^2 + a_z^2} \tag{3.25}$$

und wird im Lernmodul als *Vector Length* bezeichnet.

Anschließend werden die Daten in kleinere Zeitfenster einer festen Länge unterteilt. Die Größe der Windows wird auf drei Sekunden festgelegt und jedem Datenpunkt wird eine *WindowID* zugeordnet. Den Daten zu Aktivität 1 von Nutzer 1, die zwischen null und drei Sekunden liegen, wird die WindowID 1 zugeordnet, den Daten zu Aktivität 1 von Nutzer 1, die zwischen drei und sechs Sekunden liegen, die WindowID 2, usw. Insgesamt werden die aufgenommenen Sensordaten in 3000 Windows unterteilt. Im Anschluss werden die WindowID und der Betrag des Beschleunigungsvektors als zusätzliche Spalten dem Datensatz hinzugefügt (s. Abb. 15).

WindowID	UserID	ActivityID	Time (s)	x (m/s^2)	y (m/s^2)	y (m/s^2)	Vector Length (m/s^2)
1	1	1	0.028140	5.480853	-4.508624	-4.508624	8.408040
1	1	1	0.053292	5.488187	-4.504134	-4.504134	8.408010
1	1	1	0.078443	5.494324	-4.496799	-4.496799	8.404166
1	1	1	0.103595	5.492229	-4.508475	-4.508475	8.415299
1	1	1	0.128747	5.472320	-4.499044	-4.499044	8.392204

Abbildung 15: Screenshot eines Ausschnittes des um die WindowID und den Betrag des Beschleunigungsvektors ergänzten Datensatzes

Auf die Datensegmentierung folgt die Merkmalsextraktion. Die große Menge an Rohdaten wird auf eine kleinere Menge möglichst aussagekräftiger Werte reduziert. Als aussagekräftige Werte werden zunächst die statistischen Kenngrößen Minimum und Maximum sowie das arithmetische Mittel des Betrags des Beschleunigungsvektors gewählt. Diese Kenngrößen werden mit Hilfe des Computers für alle 3000 Windows berechnet und zusammen mit der WindowID, ActivityID und UserID in einem neuen Datensatz, dem *Feature-Set 1*, abgespeichert (s. Abb. 16).

WindowID	UserID	ActivityID	Mean Vector Length (m/s^2)	Max Vector Length (m/s^2)	Min Vector Length (m/s^2)
1.0	1.0	1.0	8.382189	8.593413	8.136967
2.0	1.0	1.0	8.335617	8.571866	8.052737
3.0	1.0	1.0	8.289633	8.384486	8.059064
4.0	1.0	1.0	8.262474	8.533645	7.956085
5.0	1.0	1.0	8.268336	8.325962	8.224459

Abbildung 16: Screenshot eines Ausschnittes aus dem Feature-Set 1

3.4.3. Entwicklung eines Klassifikationsalgorithmus

Jedes Window im Feature-Set 1 entspricht einem Datenpunkt in drei Dimensionen (s. Abb. 17). Die Dimensionen sind die drei berechneten Features Minimum, Maximum und arithmetisches Mittel des Betrags des Beschleunigungsvektors.

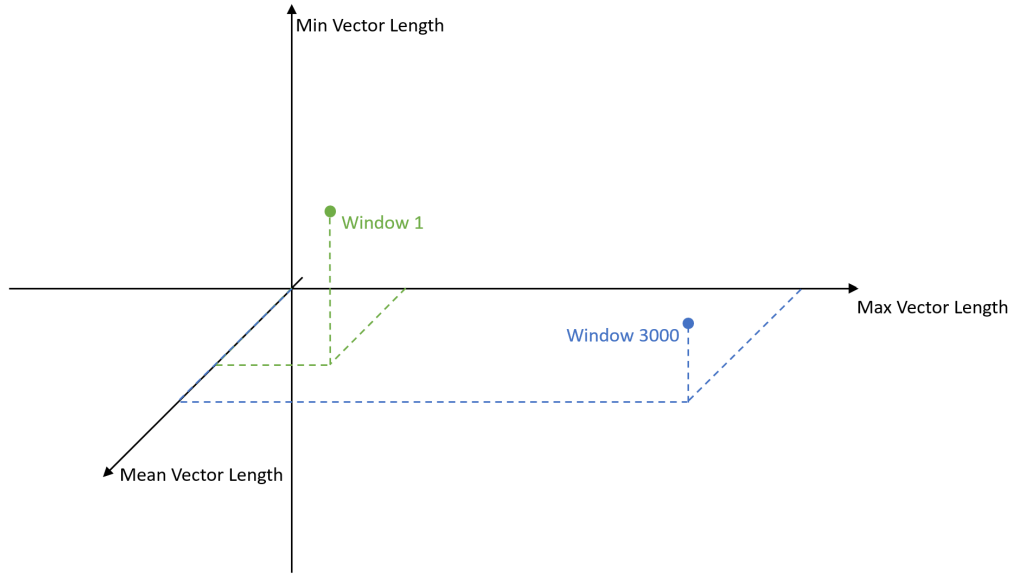


Abbildung 17: Datenpunkte der Windows 1 und 3000

In Abbildung 18 sind die Datenpunkte aller Windows der Person mit UserID 1 graphisch dargestellt.

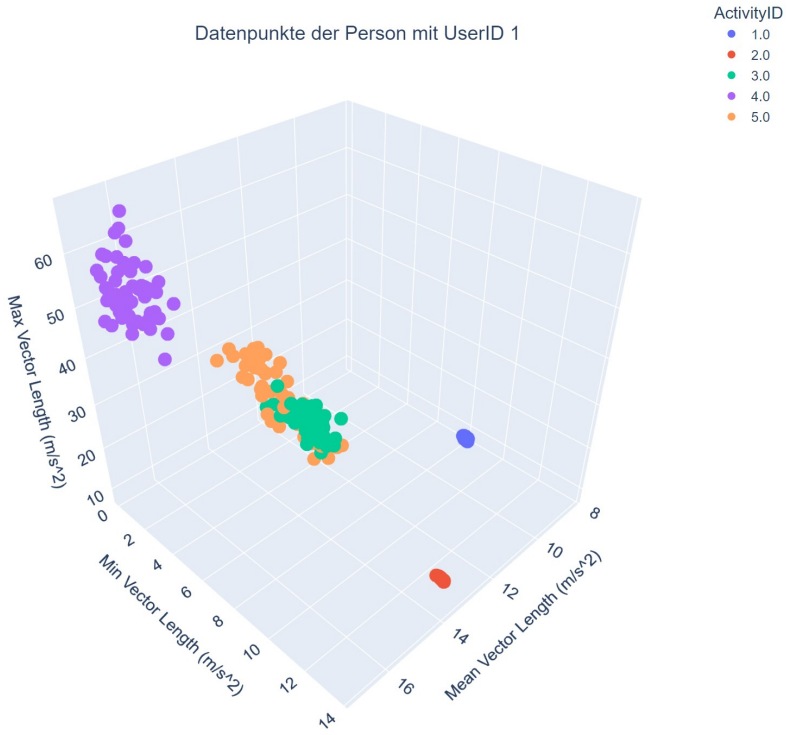


Abbildung 18: Datenpunkte aller Windows der Person mit UserID 1

In der Abbildung ist zu erkennen, dass die Datenpunkte der Aktivitäten 3 und 5 (Gehen und Treppen auf- und absteigen) sehr nahe aneinander liegen und sich die Datenwolken zum Teil überschneiden. Bei der Klassifikation von Sensordaten dieser beiden Aktivitäten mit dem kNN-Algorithmus kann es daher zu Problemen kommen, denn unter den k nächsten Nachbarn eines Datenpunktes können auch Datenpunkte der „falschen“ Aktivität liegen. Die Datenpunkte der anderen Aktivitäten liegen weit auseinander, sodass hier weniger Probleme bei der Klassifikation zu erwarten sind.

Zur Berechnung des Abstands zwischen den Datenpunkten zweier Windows wird im Lernmodul zunächst die euklidische Metrik (s. Abschn. 3.2.2) verwendet. Die k nächsten Nachbarn eines ausgewählten Test-Windows werden bestimmt, indem die Abstände des Test-Windows zu allen anderen Windows berechnet und die k kleinsten Abstände ermittelt werden. Nachdem die k nächsten Nachbarn des Test-Windows gefunden wurden, wird die Aktivität des Test-Windows über einen Mehrheitsentscheid bestimmt. Die Aktivität, die unter den k nächsten Nachbarn am häufigsten vorkommt, bestimmt die Aktivität des Test-Windows. Im sehr seltenen Falle eines Gleichstands werden hier zunächst beide Aktivitäten ausgegeben, die unter den k nächsten Nachbarn am häufigsten vorkommen. Später wird im Falle eines Gleichstands anhand der Anzahl der Datenpunkte einer Aktivität entschieden. Die Aktivität, zu der mehr Datenpunkte im Datensatz gehören, „gewinnt“ den Mehrheitsentscheid.

3.4.4. Bewertung der Ergebnisse der Klassifikation

Zur Bewertung der Ergebnisse der Klassifikation wird das Prinzip des überwachten maschinellen Lernens angewandt (s. Abschn. 3.1.2). Das Feature-Set 1 wird im Verhältnis 80:20 in Trainings- und Testdaten aufgeteilt. Der Trainingsdatensatz besteht somit aus 2400 Datenpunkten und der Testdatensatz aus 600 Datenpunkten. Für die Anzahl der betrachteten nächsten Nachbarn wird $k = 3$ gewählt.

Die Ergebnisse der Klassifikation der Testdaten werden in Form einer Wahrheitsmatrix ausgegeben (s. Tab. 4).

Tabelle 4: Wahrheitsmatrix für die Klassifikation der Testdaten mit dem Feature-Set 1 und $k = 3$

	Klassifiziert als 1 (Sitzen)	**Klassifiziert als 2 (Stehen)**	**Klassifiziert als 3 (Gehen)**	**Klassifiziert als 4 (Laufen)**	**Klassifiziert als 5 (Treppen auf- und absteigen)**
Tatsächlich 1 (Sitzen)	123	0	0	0	0
Tatsächlich 2 (Stehen)	0	115	0	0	0
Tatsächlich 3 (Gehen)	0	0	79	10	19
Tatsächlich 4 (Laufen)	0	0	14	120	1
Tatsächlich 5 (Treppen auf- und absteigen)	1	0	24	1	93

Insgesamt werden 530 Windows richtig und 70 Windows falsch klassifiziert. Bei der Aktivität 1 (Sitzen) werden mit 123 Datenpunkten die meisten Datenpunkte richtig zugeordnet, während bei Aktivität 3 (Gehen) mit 29 Datenpunkten die meisten Datenpunkte falsch klassifiziert werden. Zwischen den beiden Aktivitäten 3 (Gehen) und 5 (Treppen auf- und absteigen) liegen die meisten Überschneidungen vor. Anstelle von Gehen werden 19 Datenpunkte als Treppen auf- und absteigen klassifiziert und 24 Datenpunkte werden anstelle von Treppen auf- und absteigen als Gehen klassifiziert.

Mit Hilfe der Wahrheitsmatrix können die in Abschnitt 3.2.3 vorgestellten Qualitätsmaße berechnet werden. Es bezeichne CM_{ij} den Eintrag in Spalte j und Zeile i der Wahrheitsmatrix sowie CM_{Sum} die Summer aller Einträge in der Wahrheitsmatrix. Die Genauigkeit ist dann gegeben durch

$$\text{accuracy} = \frac{CM_{11} + CM_{22} + CM_{33} + CM_{44} + CM_{55}}{CM_{\text{Sum}}} \approx 0.88. \tag{3.26}$$

Für die Fehlerrate ergibt sich

$$\text{error rate} = 1 - \text{accuracy} \approx 0.12 \tag{3.27}$$

und die Präzision der einzelnen Aktivitäten ist gegeben durch

$$\begin{aligned}
\text{precision}_{\text{Sitzen}} &= \frac{CM_{11}}{CM_{11} + CM_{21} + CM_{31} + CM_{41} + CM_{51}} \approx 0.99, \\
\text{precision}_{\text{Stehen}} &= \frac{CM_{22}}{CM_{12} + CM_{22} + CM_{32} + CM_{42} + CM_{52}} = 1.0, \\
\text{precision}_{\text{Gehen}} &= \frac{CM_{33}}{CM_{13} + CM_{23} + CM_{33} + CM_{43} + CM_{53}} \approx 0.68, \\
\text{precision}_{\text{Laufen}} &= \frac{CM_{44}}{CM_{14} + CM_{24} + CM_{34} + CM_{44} + CM_{54}} \approx 0.92, \\
\text{precision}_{\text{Treppen}} &= \frac{CM_{55}}{CM_{15} + CM_{25} + CM_{35} + CM_{45} + CM_{55}} \approx 0.82.
\end{aligned} \tag{3.28}$$

Da der Wert der Präzision für die Aktivitäten Sitzen und Stehen nahe bei eins liegt, kann geschlussfolgert werden, dass nahezu alle Windows, die einer der beiden Aktivitäten zugeordnet werden, auch tatsächlich zu dieser Aktivität gehören. Die Präzision ist für die Aktivitäten Gehen und Treppen auf- und absteigen am kleinsten. Das liegt daran, dass die Datenpunkte der beiden Aktivitäten sehr nahe aneinander liegen und es dadurch häufiger zu falschen Klassifikationen kommt (s. Abb. 18). Für diese beiden Aktivitäten sind die Ergebnisse der Klassifikation noch nicht zufriedenstellend. Daher wird der entwickelte Klassifikationsalgorithmus im Folgenden durch zwei Veränderungen verbessert.

3.4.5. Verbesserung des Klassifikationsalgorithmus

Am entwickelten Klassifikationsalgorithmus werden zwei Veränderungen vorgenommen, um die Klassifikationsergebnisse zu verbessern. Zum einen wird das Feature-Set 1 um weitere Features ergänzt, zum anderen wird die Wahl des Parameters k optimiert.

Erweiterung des Feature-Sets 1

Das Feature-Set 1 wird um die Kenngrößen Minimum und Maximum sowie das arithmetische Mittel jeder der drei Beschleunigungsrichtungen (a_x, a_y und a_z) erweitert. Zudem werden der Median sowie das untere und das obere Quartil des Betrags des Beschleunigungsvektors ergänzt. Der neu entstandene Datensatz wird als *Feature-Set 2* bezeichnet (s. Abb. 19).

WindowID	UserID	ActivityID	Mean Vector Length (m/s^2)	Max Vector Length (m/s^2)	Min Vector Length (m/s^2)	Median Vector Length (m/s^2)	Upper Quartile Vector Length (m/s^2)	Lower Quartile Vector Length (m/s^2)
1.0	1.0	1.0	8.382189	8.593413	8.136967	8.385889	8.437079	8.323079
2.0	1.0	1.0	8.335617	8.571866	8.052737	8.338597	8.377425	8.323079
3.0	1.0	1.0	8.289633	8.384486	8.059064	8.290660	8.312062	8.323079
4.0	1.0	1.0	8.262474	8.533645	7.956085	8.275744	8.313050	8.323079
5.0	1.0	1.0	8.268336	8.325962	8.224459	8.269243	8.277988	8.323079

Mean x (m/s^2)	Max x (m/s^2)	Min x (m/s^2)	Mean y (m/s^2)	Max y (m/s^2)	Min y (m/s^2)	Mean z (m/s^2)	Max z (m/s^2)	Min z (m/s^2)
5.505176	5.615423	5.450765	-4.469158	-4.271966	-4.689299	-6.878246	-6.785539	-7.006031
5.519143	5.657186	5.373675	-4.416821	-4.172723	-4.578230	-6.900961	-6.728059	-7.023245
5.486834	5.620662	5.331613	-4.393822	-4.272116	-4.490362	-6.943257	-6.869365	-7.107370
5.509859	5.862559	5.325925	-4.353382	-4.147874	-4.539909	-6.948716	-6.761290	-7.173083
5.513971	5.550308	5.474566	-4.356670	-4.315825	-4.397256	-6.946351	-6.911877	-6.987170

Abbildung 19: Screenshot eines Ausschnittes des Feature-Sets 2

Nach der Erweiterung des Feature-Sets 1 werden die Daten erneut dem Klassifikationsalgorithmus übergeben. Für die Anzahl der nächsten Nachbarn wird wieder $k = 3$ gewählt. Tabelle 5 zeigt die Ergebnisse in Form einer Wahrheitsmatrix.

Tabelle 5: Wahrheitsmatrix für die Klassifikation der Testdaten mit dem Feature-Set 2 und $k = 3$

	Klassifiziert als 1 (Sitzen)	**Klassifiziert als 2 (Stehen)**	**Klassifiziert als 3 (Gehen)**	**Klassifiziert als 4 (Laufen)**	**Klassifiziert als 5 (Treppen auf- und absteigen)**
Tatsächlich 1 (Sitzen)	123	0	0	0	0
Tatsächlich 2 (Stehen)	0	115	0	0	0
Tatsächlich 3 (Gehen)	0	0	105	2	1
Tatsächlich 4 (Laufen)	0	0	2	133	10
Tatsächlich 5 (Treppen auf- und absteigen)	0	0	10	2	107

Die Qualitätsmaße verbessern sich durch die Erweiterung des Feature-Sets 1 zu

$$\begin{aligned} \text{accuracy} &\approx 0.97, \\ \text{error rate} &\approx 0.03, \\ \text{precision}_{\text{Sitzen}} &= 1.0, \\ \text{precision}_{\text{Stehen}} &= 1.0, \\ \text{precision}_{\text{Gehen}} &\approx 0.90, \\ \text{precision}_{\text{Laufen}} &\approx 0.97, \\ \text{precision}_{\text{Treppen}} &\approx 0.99. \end{aligned} \tag{3.29}$$

Im Mittel werden nun 97 von 100 Aktivitäten richtig klassifiziert. Bei der Aktivität 3 (Gehen) treten weiterhin die meisten Fehlklassifikationen auf.

Optimierung des Parameters k

Zur Bestimmung der optimalen Anzahl der nächsten Nachbarn k wird folgendes Optimierungsproblem gelöst:

Optimierungsproblem:
Finde die Anzahl der Nachbarn k, für die die Fehlerrate minimal wird.

Zunächst wird die Fehlerrate gegen die Anzahl der Nachbarn k aufgetragen (s. Abb. 20).

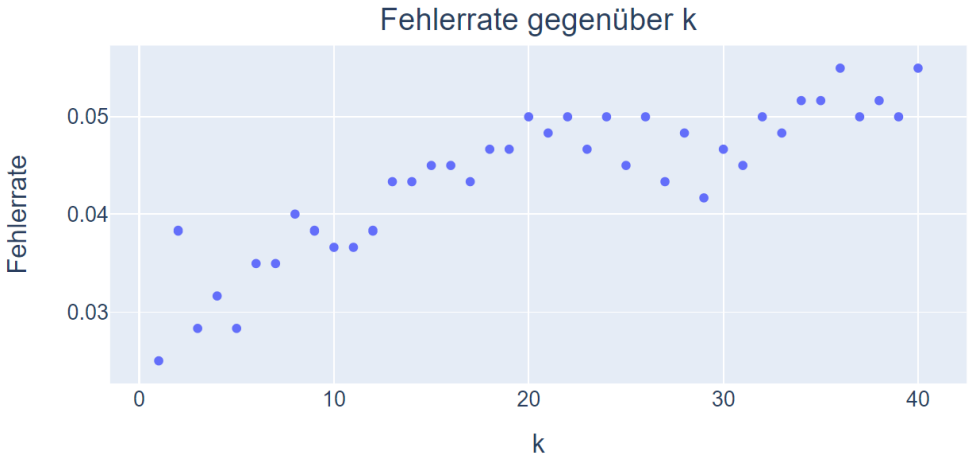


Abbildung 20: Fehlerrate gegenüber der Anzahl der Nachbarn k

Es ist zu erkennen, dass die Fehlerrate für $k = 1$ am kleinsten ist. Außerdem zeigt sich, dass für kleine k die Fehlerrate stark schwankt. Das liegt daran, dass für kleine und

gerade k die Wahrscheinlichkeit für das Auftreten eines „Unentschieden“ beim Mehrheitsentscheid sehr groß ist. Für große k nimmt diese Wahrscheinlichkeit ab, weshalb die Schwankungen kleiner werden. Falls ein Unentschieden vorliegt, wird anhand der Daten im Trainingsdatensatz entschieden. Die Klasse, zu der mehr Datenpunkte im Trainingsdatensatz gehören, „gewinnt“ den Mehrheitsentscheid. Ein ähnliches Verhalten der Fehlerrate für gerade Werte von k ist bei Mohsen et al. (2022, S. 307) zu sehen. Allerdings wird die Fehlerrate hier im Gegensatz zu den oben aufgeführten Ergebnissen für große Werte von k (z. B. $k = 20$) minimal.

Anschließend wird das optimale k auch rechnerisch ermittelt, indem mit Hilfe einer for-Schleife die Fehlerraten für alle k zwischen 1 und 40 bestimmt und in einer Liste abgespeichert werden. Die Position der kleinsten Fehlerrate in dieser Liste liefert das optimale k. Auch dieses algorithmische Vorgehen liefert $k = 1$ als optimale Wahl des Parameters.

Die Wahrheitsmatrix für die Klassifikation der Testdaten mit $k = 1$ und dem Feature-Set 2 ist in Tabelle 6 zu sehen.

Tabelle 6: Wahrheitsmatrix für die Klassifikation der Testdaten mit dem Feature-Set 2 und $k = 1$

	Klassifiziert als 1 (Sitzen)	**Klassifiziert als 2 (Stehen)**	**Klassifiziert als 3 (Gehen)**	**Klassifiziert als 4 (Laufen)**	**Klassifiziert als 5 (Treppen auf- und absteigen)**
Tatsächlich 1 (Sitzen)	123	0	0	0	0
Tatsächlich 2 (Stehen)	0	115	0	0	0
Tatsächlich 3 (Gehen)	0	0	105	2	1
Tatsächlich 4 (Laufen)	0	0	2	133	10
Tatsächlich 5 (Treppen auf- und absteigen)	0	0	10	2	107

Die Qualitätsmaße für die Klassifikation mit dem Feature-Set 2 und $k = 1$ sind gegeben durch

$$\begin{aligned} \text{accuracy} &\approx 0.98, \\ \text{error rate} &\approx 0.02, \\ \text{precision}_{\text{Sitzen}} &= 1.0, \\ \text{precision}_{\text{Stehen}} &= 1.0, \\ \text{precision}_{\text{Gehen}} &\approx 0.90, \\ \text{precision}_{\text{Laufen}} &\approx 0.98, \\ \text{precision}_{\text{Treppen}} &= 1.0. \end{aligned} \tag{3.30}$$

Im Vergleich zur ersten Klassifikation mit $k = 3$ und dem Feature-Set 1 (s. Abschn. 3.4.4) konnten die Ergebnisse der Klassifikation durch die beiden Veränderungen des

Klassifikationsalgorithmus deutlich verbessert werden. Für die aufgenommenen Sensordaten liefert das entwickelte Modell zufriedenstellende Klassifikationsergebnisse und eignet sich damit zur Aktivitätserkennung.

3.4.6. Schwierigkeiten des Klassifikationsalgorithmus

Eine große Herausforderung der Aktivitätserkennung ist die Positionierung und Ausrichtung der Sensoren am Körper des Benutzers. Um zu testen, wie gut der entwickelte Klassifikationsalgorithmus mit diesem Problem umgeht, wurden weitere Daten aufgenommen. Einmal befand sich das Handy während der Datenaufnahme in der Hand des Nutzers und das andere Mal war es in einem Rucksack verstaut. Für die neu aufgenommenen Daten wurden die Features des Feature-Sets 2 berechnet. Bei den im Folgenden dargestellten Klassifikationsergebnissen werden jeweils die neu aufgenommenen Daten als Testdaten und das Feature-Set 2 des ursprünglichen Datensatzes als Trainingsdaten verwendet. Für die Anzahl der betrachteten Nachbarn wird $k = 1$ gewählt.

Für den Fall, dass sich das Handy in der Hand des Nutzers befand, ergibt sich die in Tabelle 7 dargestellte Wahrheitsmatrix.

Tabelle 7: Wahrheitsmatrix für die Klassifikation der Testdaten, bei denen sich das Handy in der Hand des Nutzers befand, mit dem Feature-Set 2 und $k = 1$

	Klassifiziert als 1 (Sitzen)	**Klassifiziert als 2 (Stehen)**	**Klassifiziert als 3 (Gehen)**	**Klassifiziert als 4 (Laufen)**	**Klassifiziert als 5 (Treppen auf- und absteigen)**
Tatsächlich 1 (Sitzen)	60	0	0	0	0
Tatsächlich 2 (Stehen)	60	0	0	0	0
Tatsächlich 3 (Gehen)	60	0	0	0	0
Tatsächlich 4 (Laufen)	11	0	29	2	18
Tatsächlich 5 (Treppen auf- und absteigen)	60	0	10	2	107

Fast alle Testdatenpunkte werden fälschlicherweise der Aktivität Sitzen zugeordnet, weshalb die einzelnen Qualitätsmaße nun deutlich schlechter ausfallen. Diese sind gegeben durch

$$\begin{aligned} \text{accuracy} &\approx 0.20, \\ \text{error rate} &\approx 0.80, \\ \text{precision}_{\text{Sitzen}} &\approx 0.24, \\ \text{precision}_{\text{Gehen}} &= 0.0, \\ \text{precision}_{\text{Laufen}} &= 1.0, \\ \text{precision}_{\text{Treppen}} &= 0.0. \end{aligned} \tag{3.31}$$

Bei der Aktivität Stehen tritt der unbestimmte Ausdruck $\frac{0}{0}$ auf, weshalb für diese Aktivität keine Präzision berechnet werden konnte.

Für die Klassifikation der Sensordaten, bei denen das Handy im Rucksack verstaut war, ergibt sich die in Tabelle 8 dargestellte Wahrheitsmatrix.

Tabelle 8: Wahrheitsmatrix für die Klassifikation der Testdaten, bei denen sich das Handy im Rucksack des Nutzers befand, mit dem Feature-Set 2 und $k = 1$

	Klassifiziert als 1 (Sitzen)	**Klassifiziert als 2 (Stehen)**	**Klassifiziert als 3 (Gehen)**	**Klassifiziert als 4 (Laufen)**	**Klassifiziert als 5 (Treppen auf- und absteigen)**
Tatsächlich 1 (Sitzen)	60	0	0	0	0
Tatsächlich 2 (Stehen)	0	60	0	0	0
Tatsächlich 3 (Gehen)	0	0	0	0	60
Tatsächlich 4 (Laufen)	0	0	0	0	60
Tatsächlich 5 (Treppen auf- und absteigen)	0	0	26	0	34

Die Ergebnisse sind etwas besser als für die Daten, bei denen das Handy in der Hand gehalten wurde. Die einzelnen Qualitätsmaße sind gegeben durch

$$\begin{aligned} \text{accuracy} &\approx 0.51, \\ \text{error rate} &\approx 0.49, \\ \text{precision}_{\text{Sitzen}} &= 1.0, \\ \text{precision}_{\text{Stehen}} &= 1.0, \\ \text{precision}_{\text{Gehen}} &= 0.0, \\ \text{precision}_{\text{Treppen}} &\approx 0.22. \end{aligned} \tag{3.32}$$

Diesmal tritt bei der Aktivität Laufen der unbestimmte Ausdruck $\frac{0}{0}$ auf, weshalb für diese Aktivität keine Präzision berechnet werden konnte.

Der entwickelte Klassifikationsalgorithmus funktioniert also nur für die Beschleunigungsdaten sehr gut, bei denen sich das Handy in der vorderen rechten Hosentasche befand. Das liegt daran, dass die charakteristischen Ausschläge der Beschleunigungswerte einer bestimmten Achse je nach Orientierung des Smartphones nun auf einer anderen Achse zu finden sind. Befindet sich das Handy zum Beispiel beim Stehen in der Hosentasche wirkt die Erdbeschleunigung auf die y-Achse des Accelerometers. Wird das Handy nun aber in der Hand gehalten, zum Beispiel um (Chat-)Nachrichten zu lesen, wirkt die Erdbeschleunigung auf die z-Achse. Da dem Klassifikationsalgorithmus nur Daten, bei denen sich das Handy in der vorderen rechten Hosentasche befand, als Trainingsdaten zur Verfügung stehen, kann er die neuen Daten, bei denen sich das Handy in der Hand oder im Rucksack des Nutzers befand, nicht richtig zuordnen. Wird jedoch zur Diversifizierung des Trainingsdatensatzes die Hälfte der Daten,

bei denen sich das Handy in der Hand bzw. im Rucksack des Nutzers befand, den Trainingsdaten zugeordnet, verbessern sich die Ergebnisse der Klassifikation deutlich. Als Testdatensatz wird jeweils die andere Hälfte der Daten, bei denen sich das Handy nicht in der vorderen rechten Hosentasche befand, gewählt. Die Wahrheitsmatrix sowie die einzelnen Qualitätsmaße dieser Klassifizierungen sind im Anhang H.1 zu finden.

Das Problem der Positionierung und Ausrichtung der Sensoren am Körper des Nutzers könnte durch die Einbeziehung weiterer Sensoren gelöst werden. Beispielsweise lassen sich die Messwerte des Accelerometers mit Hilfe der Messwerte des Gyroskops und des Magnetometers in Erdkoordinaten umrechnen. Bei der späteren Klassifikation liegen dann alle Sensordaten in Erdkoordinaten vor, sodass das Problem der Positionierung und Ausrichtung der Sensoren am Körper des Nutzers umgangen werden kann (vgl. Ustev et al., 2013, S. 1430).

3.4.7. Auswirkungen anderer Metriken

Die Ergebnisse der Klassifikation sind von der verwendeten Metrik zur Abstandsberechnung abhängig. Daher wird im Folgenden untersucht, wie sich die Ergebnisse verändern, wenn unterschiedliche Metriken verwendet werden. Die folgenden Klassifikationen werden immer mit dem Feature-Set 1 und $k = 3$ durchgeführt, d.h. die Ergebnisse sind mit denen aus Abschnitt 3.4.4 zu vergleichen.

Zuerst wird anstelle der euklidischen Metrik die Manhattan-Metrik verwendet. Für diesen Fall sind die Ergebnisse in Form einer Wahrheitsmatrix in Tabelle 9 zu sehen.

Tabelle 9: Wahrheitsmatrix für die Klassifikation der Testdaten mit der Manhattan-Metrik, dem Feature-Set 1 und $k = 3$

	Klassifiziert als 1 (Sitzen)	**Klassifiziert als 2 (Stehen)**	**Klassifiziert als 3 (Gehen)**	**Klassifiziert als 4 (Laufen)**	**Klassifiziert als 5 (Treppen auf- und absteigen)**
Tatsächlich 1 (Sitzen)	123	0	0	0	0
Tatsächlich 2 (Stehen)	0	115	0	0	0
Tatsächlich 3 (Gehen)	0	0	77	9	22
Tatsächlich 4 (Laufen)	0	0	16	118	1
Tatsächlich 5 (Treppen auf- und absteigen)	0	0	24	1	94

Die Qualitätsmaße für die Klassifikation der Testdaten mit der Manhattan-Metrik er-

geben sich zu

$$\begin{aligned} \text{accuracy} &\approx 0.88, \\ \text{error rate} &\approx 0.12, \\ \text{precision}_{\text{Sitzen}} &= 1.0, \\ \text{precision}_{\text{Stehen}} &= 1.0, \\ \text{precision}_{\text{Gehen}} &\approx 0.66, \\ \text{precision}_{\text{Laufen}} &\approx 0.92, \\ \text{precision}_{\text{Treppen}} &\approx 0.80. \end{aligned} \tag{3.33}$$

Im Vergleich zur Klassifikation mit der euklidischen Metrik sind die Ergebnisse etwas schlechter. Das liegt daran, dass bei der Manhattan-Metrik große Abweichungen auf einer Achse weniger stark ins Gewicht fallen als bei der euklidischen Metrik. So ist beispielsweise der Abstand eines Windows, das auf einer Achse sehr weit entfernt, aber auf den anderen beiden Achsen sehr nahe beim Test-Window liegt, genauso groß wie der Abstand eines Windows, das auf allen Achsen leichte Abweichungen zeigt. Während die Daten des zweiten Windows höchst wahrscheinlich zur Aktivität des Test-Windows gehören, stammen die Daten des ersten Windows vermutlich von einer anderen Aktivität. Dennoch kann es passieren, dass das erste Window unter den k nächsten Nachbarn des Test-Windows liegt. Damit steigt die Wahrscheinlichkeit für eine Fehlklassifikation.

Als zweite Metrik wurde die Kosinus-Metrik verwendet. Die Ergebnisse sind in der folgenden Wahrheitsmatrix zu sehen (s. Tab. 10).

Tabelle 10: Wahrheitsmatrix für die Klassifikation der Testdaten mit der Kosinus-Metrik, dem Feature-Set 1 und $k = 3$

	Klassifiziert als 1 (Sitzen)	**Klassifiziert als 2 (Stehen)**	**Klassifiziert als 3 (Gehen)**	**Klassifiziert als 4 (Laufen)**	**Klassifiziert als 5 (Treppen auf- und absteigen)**
Tatsächlich 1 (Sitzen)	80	42	0	0	1
Tatsächlich 2 (Stehen)	38	77	0	0	0
Tatsächlich 3 (Gehen)	0	0	61	20	27
Tatsächlich 4 (Laufen)	0	0	26	100	9
Tatsächlich 5 (Treppen auf- und absteigen)	0	0	39	13	67

Die Qualitätsmaße für die Klassifikation der Testdaten mit der Kosinus-Metrik sind

gegeben durch

$$\begin{aligned} \text{accuracy} &\approx 0.64, \\ \text{error rate} &\approx 0.36, \\ \text{precision}_{\text{Sitzen}} &\approx 0.68, \\ \text{precision}_{\text{Stehen}} &\approx 0.65, \\ \text{precision}_{\text{Gehen}} &\approx 0.48, \\ \text{precision}_{\text{Laufen}} &\approx 0.75, \\ \text{precision}_{\text{Treppen}} &\approx 0.64. \end{aligned} \tag{3.34}$$

Im Gegensatz zur Manhattan-Metrik und der euklidischen Metrik sind die Ergebnisse der Klassifikation mit der Kosinus-Metrik deutlich schlechter. Das liegt daran, dass der Abstand zwischen zwei Datenpunkten hier über den Winkel zwischen den beiden Ortsvektoren der beiden Punkte bestimmt wird. Die Ortsvektoren von Datenpunkten unterschiedlicher Aktivitäten können in die gleiche Richtung zeigen, sich aber in ihrer Länge unterscheiden. In diesem Fall wäre der Winkel zwischen den beiden Ortsvektoren und damit der Abstand zwischen den beiden Datenpunkten klein, obwohl die beiden Punkte zu unterschiedlichen Aktivitäten gehören.

Als letzte Metrik wird die Tschebyschew-Metrik verwendet. Tabelle 11 zeigt die Ergebnisse in Form einer Wahrheitsmatrix.

Tabelle 11: Wahrheitsmatrix für die Klassifikation der Testdaten mit der Tschebyschew-Metrik, dem Feature-Set 1 und $k = 3$

	Klassifiziert als 1 (Sitzen)	**Klassifiziert als 2 (Stehen)**	**Klassifiziert als 3 (Gehen)**	**Klassifiziert als 4 (Laufen)**	**Klassifiziert als 5 (Treppen auf- und absteigen)**
Tatsächlich 1 (Sitzen)	123	0	0	0	0
Tatsächlich 2 (Stehen)	0	115	0	0	0
Tatsächlich 3 (Gehen)	0	0	76	12	20
Tatsächlich 4 (Laufen)	0	0	14	121	0
Tatsächlich 5 (Treppen auf- und absteigen)	0	0	20	2	97

Die Qualitätsmaße sind gegeben durch

$$\begin{aligned} \text{accuracy} &\approx 0.89, \\ \text{error rate} &\approx 0.11, \\ \text{precision}_{\text{Sitzen}} &= 1.0, \\ \text{precision}_{\text{Stehen}} &= 1.0, \\ \text{precision}_{\text{Gehen}} &\approx 0.69, \\ \text{precision}_{\text{Laufen}} &\approx 0.90, \\ \text{precision}_{\text{Treppen}} &\approx 0.83. \end{aligned} \tag{3.35}$$

Verglichen mit den anderen drei Metriken liefert die Tschebyschew-Metrik die besten Ergebnisse. Das liegt daran, dass immer nur die Achse, auf der der Abstand zwischen den beiden Datenpunkten am größten ist, beachtet wird. Sobald ein Window auf einer Achse deutliche Unterschiede zum Test-Window zeigt, ist der Abstand groß. Nur wenn die Werte aller Achsen im Bereich des Test-Windows liegen ist der Abstand klein. Es werden also nur Windows, bei denen sich alle Features den Features des Test-Windows ähneln, der Aktivität des Test-Windows zugeordnet.

4. Didaktisch-methodisches Konzept

Basierend auf dem in Kapitel 3 vorgestellten mathematischen und fachlichen Hintergrund wurde ein computergestütztes, mathematisches Lernmodul für Schüler ab der zehnten Klasse entwickelt und im Rahmen eines Modellierungstages (CAMMP day) sowie einer zwei Doppelstunden umfassenden Unterrichtseinheit erprobt. In diesem Kapitel wird das dem Lernmodul zugrundeliegende didaktisch-methodische Konzept vorgestellt. Dazu werden die Ziele sowie die curriculare Einbindung des Lernmoduls erläutert, der Einsatz im Rahmen eines Modellierungstages sowie einer Unterrichtseinheit im Fach Mathematik dargelegt und ein Überblick über die erstellten Materialien gegeben.

4.1. Ziele des entwickelten Lernmoduls

Mit dem Einsatz des entwickelten Lernmoduls wird das übergeordnete Ziel verfolgt, den Schülern durch die Auseinandersetzung mit einem realen Problem die Relevanz der mathematischen Modellierung für das alltägliche Leben aufzuzeigen. Die Schüler sollen durch die Bearbeitung des Lernmoduls einen Einblick in den Sinn und die praktische Nutzbarkeit der Mathematik erhalten und Problemlösefähigkeiten, die zum Teil auch über die Mathematik hinausreichen, erwerben. In diesem Sinne adressiert das entwickelte Lernmodul die erste und die dritte Winter'sche Grunderfahrung (s. Abschn. 2.2.4).

Durch die authentische und alltagsnahe Problemstellung soll das Interesse und die Motivation der Schüler für die Mathematik gefördert werden. Aus der Schule bekannte Inhalte sollen in realen Anwendungen vertieft bzw. weiter zurückliegende Inhalte wiederholt und aufgefrischt werden. Dabei soll den Lernenden auch das Zusammenspiel der Fächer Mathematik, Informatik und Physik sowie die Relevanz digitaler Werkzeuge zur Lösung mathematischer Probleme aufgezeigt werden. Hierzu erfordert das Lernmodul jedoch keinerlei Programmierkenntnisse.

Das entwickelte Lernmodul soll der Heterogenität einer Lerngruppe gerecht werden. Durch den Einsatz verschiedener Zusatzaufgaben sollen leistungsstärkere Schüler gefördert werden. Für leistungsschwächere Schüler wurden zum einen Hilfekarten konzipiert und zum anderen Infoblätter zu grundlegenden mathematischen Inhalten erstellt. Darüber hinaus wurden zwei Arbeitsblätter auf zwei unterschiedlichen Niveaustufen entwickelt, sodass auch Schüler ab der zehnten Klasse, die noch keine Grundkenntnisse im Bereich der Vektorrechnung haben, das Lernmodul bearbeiten können.

Auf der inhaltlichen Ebene soll das Lernmodul den Schülern ermöglichen, ein grundlegendes Verständnis zu den Themen KI und maschinelles Lernen aufzubauen. Die Lernenden sollen das Prinzip überwachter maschineller Lernverfahren am Beispiel des kNN-Algorithmus kennenlernen und verstehen. Im Laufe des Lernmoduls entwickeln die Schüler schrittweise ihr eigenes maschinelles Lernverfahren und orientieren sich

dabei am vierschrittigen Modellierungskreislauf (s. Abschn. 2.2.2). Durch die Bearbeitung einer realen Problemstellung und das weitestgehend eigenständige Durchlaufen des Modellierungsprozesses sollen die Lernenden ein tiefergehendes Verständnis der mathematischen Modellierung erlangen.

4.2. Curriculare Einbindung des entwickelten Lernmoduls

Das entwickelte Lernmodul baut auf den fachlichen Inhalten des Mathematikunterrichts der Klassenstufen 5 bis 10 sowie des Physikunterrichts der Stufen 9/10 auf und ist daher für Schulklassen ab Jahrgangsstufe 10 nach der Einführung der Vektorrechnung geeignet. Durch die Erstellung zusätzlicher Arbeitsblätter auf unterschiedlichen Niveaustufen und den Einbau von Infoblättern zu grundlegenden mathematischen Inhalten kann das Lernmodul auch vor der Einführung der Vektorrechnung mit Schülern der zehnten Klasse durchgeführt werden.

Besonders die Leitideen *Daten und Zufall*, *Messen* sowie *Raum und Form* der inhaltsbezogenen Kompetenzen aus dem Bildungsplan Mathematik des Landes Baden-Württemberg sind von zentraler Bedeutung im entwickelten Lernmodul. In den Klassenstufen 5/6 lernen die Schüler im Bereich der Leitidee *Daten und Zufall* die Kenngrößen Maximum, Minimum und Mittelwert (arithmetisches Mittel) kennen. In diesem Zusammenhang beschäftigen sich die Schüler auch mit unterschiedlichen Darstellungsformen von Daten (z. B. Texten, Diagrammen, bildlichen Darstellungen und Tabellen). Dabei lernen sie, Daten aus vorgegebenen Sekundärquellen zu entnehmen, diese auch bei unterschiedlichen Darstellungsformen auszuwerten, zu vergleichen und zu deuten sowie statistische Darstellungen hinsichtlich ihrer Eignung und möglicher Irreführung zu beurteilen (vgl. Ministerium für Kultus, Jugend und Sport Baden-Württemberg, 2016, S. 22). In den Klassenstufen 7/8 werden die Kompetenzen in diesem Bereich weiter ausgebaut. Die Schüler lernen zum einen die Kenngrößen unteres und oberes Quartil sowie den Median kennen. Zum anderen formulieren und bewerten die Schüler Aussagen, die auf einer Datenanalyse basieren (vgl. Ministerium für Kultus, Jugend und Sport Baden-Württemberg, 2016, S. 28). Dieses Vorwissen wird sowohl bei der Analyse der Sensordaten und den verschiedenen graphischen Darstellungen als auch bei der Berechnung der statistischen Kenngrößen des Feature-Sets 1 und Feature-Sets 2 benötigt. In den Klassen 9/10 lernen die Schüler im Bereich der Leitidee *Raum und Form* den Satz des Pythagoras kennen und nutzen diesen zur Berechnung von Streckenlängen (vgl. Ministerium für Kultus, Jugend und Sport Baden-Württemberg, 2016, S. 33). Darauf aufbauend bestimmen die Schüler im Bereich der Leitidee *Messen* den Abstand zweier Punkte, berechnen den Betrag eines Vektors und interpretieren diesen als Länge des Vektors (vgl. Ministerium für Kultus, Jugend und Sport Baden-Württemberg, 2016, S. 32). Dies wird zur Berechnung des Betrags des Beschleunigungsvektors und zur Bestimmung des euklidischen Abstands zwischen den Datenpunkten zweier Windows als Vorwissen benötigt.
Neben den inhaltsbezogenen Kompetenzen sind auch die prozessbezogenen Kompetenzen *Modellieren* und *Probleme lösen* im entwickelten Lernmodul von Bedeutung.

Im Bereich der Leitidee *Mechanik (Kinematik)* der inhaltsbezogenen Kompetenzen aus dem Bildungsplan Physik des Landes Baden-Württemberg lernen die Schüler in den Klassenstufen 9/10 die Beschleunigung kennen. Im Rahmen dessen interpretieren sie die Beschleunigung als Änderungsrate der Geschwindigkeit, die wiederum die Änderungsrate des Ortes ist. Außerdem analysieren die Schüler verschiedene Bewegungsdiagramme (z. B. s-t-Diagramme, v-t-Diagramme, a-t-Diagramme) und beschreiben unterschiedliche Bewegungen verbal und rechnerisch (vgl. Ministerium für Kultus und Jugend und Sport Baden-Württemberg, 2016, S. 23). Dieses Vorwissen zur Beschleunigung ist bei der Interpretation der verschiedenen Graphen, in denen die Beschleunigungswerte der einzelnen Achsen für die unterschiedlichen Aktivitäten gegen die Zeit aufgetragen sind, wünschenswert, jedoch nicht zwingend erforderlich.

4.3. Aufbau und Elemente des entwickelten Lernmoduls

Im Folgenden werden der Aufbau und die einzelnen Elemente des Lernmoduls vorgestellt. Dabei wird besonders auf die beiden digitalen Werkzeuge Python und Jupyter Notebooks eingegangen.

4.3.1. Python und Jupyter Notebooks als digitale Werkzeuge

Die digitalen Arbeitsblätter des Lernmoduls wurden als Jupyter Notebooks[9] erstellt. Als Programmiersprache wurde Python[10] verwendet.

Python ist eine höhere Programmiersprache, die von Guido van Rossum Anfang der 1990er Jahre am Centrum Wiskunde & Informatice in Amsterdam entwickelt wurde. Sein Ziel war es, eine Programmiersprache zu entwickeln, die sich durch Einfachheit und Übersichtlichkeit auszeichnet. Darüber hinaus soll Python einen gut lesbaren und knappen Programmierstil fördern. Dies zeigt sich zum Beispiel in der Strukturierung von Blöcken durch Einrückungen anstelle von geschweiften Klammern (vgl. Wikipedia, 2022c). In den meisten Lernmodulen von CAMMP wird die Programmiersprache *Julia* verwendet. Allerdings hat sich in den letzten Jahren gezeigt, dass ein Großteil der Schüler diese Programmiersprache nicht kennt, in der Schule aber bereits ersten Kontakt zur Programmiersprache Python hatte. Daher wurde für das im Rahmen dieser Abschlussarbeit entwickelte Lernmodul die Programmiersprache Python gewählt.

Das *Project Jupyter* ist eine Non-Profit-Organisation, die im Jahre 2014 von Fernando Pérez als eine Ausgliederung aus dem Projekt IPython gegründet wurde. Ziel dieser Organisation ist die Entwicklung von Open-Source-Software für ein interaktives Arbeiten mit verschiedenen Programmiersprachen. Durch das Project Jupyter wurden die Produkte *Jupyter Notebook*, *JupyterHub* und *JupyterLab* entwickelt. Als Jupyter Notebook wird eine web-basierte interaktive Umgebung bezeichnet, mit der Jupyter

[9] `https://jupyter.org/`, letzter Aufruf: 22.09.2022.
[10] `https://www.python.org/`, letzter Aufruf: 22.09.2022.

Notebook-Dokumente erstellt werden können. Diese Dokumente bestehen aus einer Liste von Eingabe- und Ausgabezellen, die wiederum Code, Text oder Plots beinhalten können. Die Elemente des Jupyter Notebooks werden seit 2018 in einer flexibleren Benutzeroberfläche, dem JupyterLab, angeboten. Durch den JupyterHub wird zusätzlich noch ein Multi-User-Server für Jupyter Notebooks angeboten (vgl. Wikipedia, 2022b).

Die entwickelten Jupyter Notebooks des Lernmoduls sind auf Servern des KIT gespeichert. Über einen JupyterHub können die Schüler auf die Arbeitsblätter zugreifen und diese bearbeiten. Es ist also seitens der Schüler keine Installation von Programmen oder ein Download von Dateien notwendig.

Wie in Abschnitt 2.2.5 erläutert wurde, kann der eigentliche Modellierungsprozess durch den Einsatz digitaler Werkzeuge fokussiert werden. Auch im entwickelten Lernmodul wird durch die Verwendung einer Computersoftware die Förderung der Modellierungskompetenzen und das problemorientierte Bearbeiten der Aufgaben in den Vordergrund gestellt. Darüber hinaus dient Python der Visualisierung.

Für die Bearbeitung des Lernmoduls sollen keinerlei Programmierkenntnisse vorausgesetzt werden. Es soll aber auch keine ausführliche Einführung in die Programmiersprache Python notwendig sein. Lediglich eine kurze Erläuterung der Struktur der Arbeitsblätter soll zu Beginn durchgeführt werden. Daher wurde bei der Erstellung der Lernmaterialien darauf geachtet, den Code so einfach und so übersichtlich wie möglich zu gestalten.

4.3.2. Strukturierung der Arbeitsblätter

Die digitalen Arbeitsblätter sind so strukturiert, dass den Schüler zunächst in einem kurzen Infotext der Inhalt des jeweiligen Arbeitsblattes bzw. der entsprechenden Aufgabe erläutert wird. Anschließend folgt die konkrete Aufgabenstellung. Dabei wird durch verschiedene Icons (s. Abb. 21) festgelegt, wie die jeweilige Aufgabe zu bearbeiten ist. Durch den Computer wird signalisiert, dass die Schüler eine Eingabe im Codefeld machen müssen. Der Stift steht für Notizen auf dem Antwortblatt und die Lupe für eine Internetrecherche. Durch die Sprechblase, die meist am Ende der Arbeitsblätter zu finden ist, wird die Diskussion der Ergebnisse im Plenum angekündigt.

Abbildung 21: Auf den digitalen Arbeitsblättern verwendete Icons

Die meisten Aufgaben bearbeiten die Schüler durch eine Eingabe in einem Codefeld. Die Codefelder sind als eine Art Lückentext aufgebaut (s. Abb. 22). Die Stellen, an

denen die Schüler Zahlenwerte oder Formeln eingeben müssen, sind durch den Ausdruck NaN[11] gekennzeichnet.

```
[ ]: samplingDuration = NaN;
     samplingRate = NaN;

     # Hier nichts ändern!
     checkDurationRate(samplingDuration, samplingRate)
```

Abbildung 22: Aufbau eines Codefelds am Beispiel der Aufgabe 3, Teil b des ersten Arbeitsblattes

Nachdem die Schüler den Ausdruck NaN durch ihre Lösung ersetzt und das Codefeld ausgeführt haben, wird ihnen einerseits das Ergebnis ausgegeben und andererseits erhalten sie eine Rückmeldung darüber, ob ihre Lösung richtig ist oder nicht. Der Einsatz einer solchen Überprüfefunktion hat den Vorteil, dass keine gemeinsame Sicherung nach jeder einzelnen Aufgabe notwendig ist und die Schüler die Arbeitsblätter selbständig und in ihrem eigenen Tempo bearbeiten können. Zudem stellt die Überprüfefunktion eine Rückmeldehilfe nach Zech dar (s. Abschn. 2.3.2).

4.3.3. Hilfekarten und Infoblätter

Neben der Überprüfefunktion wurden weitere Hilfe- und Unterstützungselemente in die Materialien eingebaut, auf die die Schüler je nach Bedarf zurückgreifen können. Zum einen wurden zu manchen Begriffen kurze Erläuterungen in Form von Ausklappfeldern eingebaut. Dies ist in Abbildung 23 für den Begriff der Abtastrate auf Arbeitsblatt eins beispielhaft dargestellt.

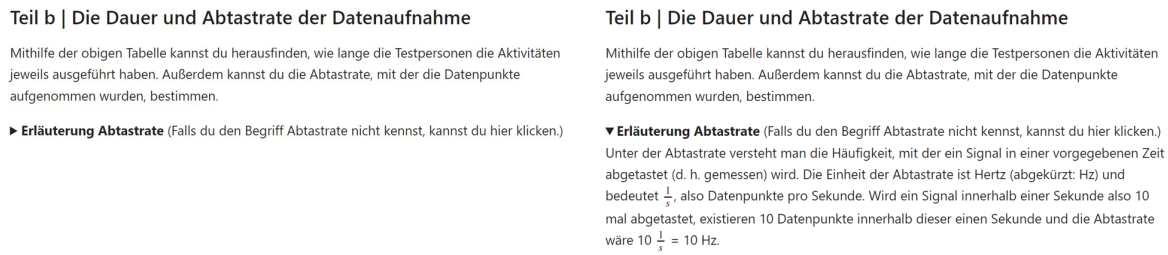


Abbildung 23: Ausklappfeld am Beispiel des Begriffs Abtastrate auf Arbeitsblatt 1 (links unausgeklappt, recht ausgeklappt)

Zum anderen wurden digitale Hilfekarten erstellt, auf die die Schüler während der Bearbeitung einer Aufgabe je nach Bedarf zurückgreifen können. Diese sind als Verlinkungen vor den Codefeldern der entsprechenden Aufgaben eingefügt (s. Abb. 24)

[11] NaN steht für „Not a Number“.

und bestehen häufig aus zwei gestuften Tipps, die sich die Schüler nacheinander anschauen können. Die meisten Tipps stellen Hilfen der Stufen 3 bis 5 des fünfstufigen Handlungskonzeptes nach Zech dar (s. Abschn. 2.3.2).

Ersetze das erste `NaN` *durch die Dauer der Datenaufnahme und das zweite* `NaN` *durch die Abtastrate, mit der die Daten aufgenommen wurden. Runde deine Lösungen auf ganze Zahlen. Führe anschließend den Code aus.*

Falls du Hilfe benötigst, kannst du dir hier einen Tipp ansehen.

Abbildung 24: Verlinkung einer Hilfekarte am Beispiel der Aufgabe 3, Teil b des ersten Arbeitsblattes

Darüber hinaus wurden Infoblätter zu den wichtigsten mathematischen Inhalten des Lernmoduls erstellt. Diese sind ebenfalls auf den Arbeitsblättern verlinkt.

Durch die verschiedenen Unterstützungsmethoden soll das Lernmodul der Heterogenität einer Lerngruppe gerecht werden. Zudem soll das eigenständige Arbeiten der Schüler gefördert werden, auch wenn Probleme und Schwierigkeiten auftreten. Um leistungsstärkeren Schülern gerecht zu werden, wurden verschiedene Zusatzmaterialien (Zusatzaufgaben und Zusatzblätter) erstellt.

4.3.4. Plenumsdiskussionen

Nach der Bearbeitung der Arbeitsblätter werden immer wieder kurze Plenumsdiskussionen durchgeführt, in denen die Ergebnisse und Erkenntnisse der einzelnen Arbeitsblätter gesichert und neue Inhalte gemeinsam erarbeitet werden. Im Rahmen dieser Diskussionen wird auch der Fortschritt im Modellierungskreislauf dargestellt. Dies soll den Schülern im Modellierungsprozess als Orientierung dienen. Darüber hinaus bieten diese Plenumsphasen Raum für Diskussionen zu den verschiedenen Inhalte des Lernmoduls oder den unterschiedlichen Lösungsansätzen der Schüler. Auf diese Weise können die Schüler im Ausgleich zu den relativ stark geführten Arbeitsblättern verstärkt eigene Ideen einbringen.

4.4. Einsatz des Lernmoduls im Rahmen eines Modellierungstages

Im Folgenden wird der Ablauf des Lernmoduls im Rahmen eines Modellierungstages bei CAMMP vorgestellt. Zusätzlich ist der Ablauf für Dozenten stichpunktartig im methodischen Konzept (s. Anh. F.2) zu finden. Auf die erstellten Materialien wird in Abschnitt 4.6 ausführlicher eingegangen.

Für die Durchführung des Lernmoduls im Rahmen eines Modellierungstages sind sechseinhalb Zeitstunden vorgesehen. Für die reine Bearbeitungszeit der Arbeitsblätter durch die Schüler und die Sicherung der Ergebnisse in kurzen Plenumsdiskussionen sind vier Stunden und 20 Minuten eingeplant. Zusätzlich ist eine 30-minütige Einführung und eine 20-minütige Nachbesprechung inklusive Evaluation vorgesehen. Außerdem sind zwei zehnminütige Pausen und eine Stunde Mittagspause geplant. Die tatsächliche Bearbeitungszeit des Lernmoduls ist jedoch stark von der Schülergruppe, deren Motivation und Vorwissen abhängig, weshalb die Zeitangaben nur der groben Orientierung dienen.

Der Ablauf des Lernmoduls ist wie folgt konzipiert:

1. **Begrüßung und Einstiegspräsentation** (30 Minuten)
   Nach einer Begrüßung und Vorstellung des Projekts CAMMP durch die Dozenten wird den Schülern der zeitliche Ablauf des Modellierungstages vorgestellt. Anschließend erhalten die Schüler einen Einblick in die mathematische Modellierung. Dazu wird ein Modellierungsvortrag[12] genutzt, in dem der Modellierungsprozess am Beispiel eines aktuellen Forschungsthemas veranschaulicht wird. Im Anschluss wird die Problemstellung im Rahmen einer Einstiegspräsentation vorgestellt. Dabei wird zunächst kurz der Prozess der menschlichen Aktivitätserkennung erläutert und auf die Themen KI und maschinelles Lernen eingegangen. In diesem Zusammenhang wird auch diskutiert, warum bei der Aktivitätserkennung maschinelle Lernverfahren zum Einsatz kommen. Außerdem werden die Ziele sowie die einzelnen Schritte des Lernmoduls präsentiert und die Problemstellung in den mathematischen Modellierungskreislauf eingeordnet. Nach der Vorstellung der Problemstellung erhalten die Schüler eine kurze Einführung in das Arbeiten mit den digitalen Arbeitsblättern.

2. **Erste Arbeitsphase und Sicherung** (1 Stunde, 20 Minuten)
   In der ersten Arbeitsphase bearbeiten die Schüler die Arbeitsblätter eins und zwei. Dabei erkunden sie den Datensatz und führen die Vorverarbeitung der Daten durch. Zur Dokumentation ihrer Ergebnisse erhalten die Schüler Antwortblätter in Papierform. Darüber hinaus stehen je nach Bedarf digitale Hilfekarten zur Verfügung, die die Schüler bei der eigenständigen Bearbeitung der Arbeitsblätter unterstützten sollen. Die Ergebnisse der Schüler werden nach der Bearbeitung des jeweiligen Arbeitsblattes im Rahmen von Plenumsdiskussionen gesichert.

3. **Pause** (10 Minuten)

4. **Zweite Arbeitsphase und Sicherung** (1 Stunde, 20 Minuten)
   Nach einer kurzen Pause lernen die Schüler auf Arbeitsblatt drei den kNN-Algorithmus kennen und implementieren diesen. Außerdem bewerten sie auf Arbeitsblatt vier die Güte des Klassifikationsalgorithmus mit Hilfe der Wahrheitsmatrix und den verschiedenen Qualitätsmaßen. Auch hier stehen den Schülern

[12] Der Modellierungsvortrag wird vom Projekt CAMMP erstellt.

Antwortblätter und Hilfekarten zur Verfügung. Im Anschluss an die einzelnen Arbeitsblätter werden die Ergebnisse im Rahmen kurzer Plenumsdiskussionen gesichert. Schüler, die die Arbeitsblätter besonders schnell bearbeiten, können die Zusatzaufgabe auf Arbeitsblatt drei und das Zusatzblatt nach Arbeitsblatt vier bearbeiten. In diesen Aufgaben beschäftigen sich die Schüler mit weiteren Abstandsmetriken und deren Auswirkungen auf die Klassifikationsergebnisse.

5. **Mittagspause** (1 Stunde)

6. **Dritte Arbeitsphase und Sicherung** (1 Stunde)
   Nach der Mittagspause bearbeiten die Schüler die Arbeitsblätter fünf und sechs. Auf diesen wird das Klassifikationsmodell durch die Erweiterung des Feature-Sets 1 und die Optimierung des Parameters k verbessert. Die Ergebnisse der Schüler werden nach der Bearbeitung der Arbeitsblätter jeweils kurz gesichert. Zur Dokumentation der Lösungen erhalten die Schüler erneut Antwortblätter. Außerdem können sie sich bei Bedarf verschiedene Hilfekarten anschauen.

7. **Pause** (10 Minuten)

8. **Vierte Arbeitsphase und Sicherung** (30 Minuten)
   In der letzten Arbeitsphase untersuchen die Schüler auf Arbeitsblatt sieben Schwierigkeiten des Klassifikationsalgorithmus. In diesem Zusammenhang werden Sensordaten, bei denen das Handy in der Hand gehalten wurde bzw. sich in einem Rucksack befand, klassifiziert. Auch hier steht den Schülern ein Antwortblatt zur Verfügung. In der anschließenden Plenumsdiskussion wird über weitere Schwierigkeiten, die nicht mit der Ausrichtung und Position des Smartphones zusammenhängen, sowie über Anwendungen und Missbrauchsgefahren der Aktivitätserkennung diskutiert.

9. **Abschlusspräsentation, Evaluation und Verabschiedung** (30 Minuten)
   Im Anschluss an die Diskussion nach Arbeitsblatt sieben werden die einzelnen Schritte des Lernmoduls mit Hilfe des Zusammenfassungsarbeitsblattes wiederholt. Anschließend füllen die Schüler einen Evaluationsbogen aus, in dem sie das Lernmodul bewerten und den Dozenten Rückmeldung geben können. Auf die Evaluation folgt eine kurze Verabschiedung, in der die Schüler Informationen zu weiteren Angeboten von CAMMP erhalten.

Der Einsatz des Lernmoduls im Rahmen eines Modellierungstages bei CAMMP wurde als Teil dieser Abschlussarbeit erprobt. Die Beobachtungen bei der Durchführung, die Ergebnisse der Evaluation und die daraus resultierenden Verbesserungen des Lernmaterials werden in Kapitel 5 beschrieben.

4.5. Einsatz des Lernmoduls im Rahmen einer Unterrichtseinheit im Fach Mathematik

Das entwickelte Lernmodul lässt sich auch im Rahmen einer Unterrichtseinheit im Fach Mathematik durchführen. Hierzu sollten 8 Schulstunden (à 45 Minuten) bzw. 4 Doppelstunden (à 90 Minuten) eingeplant werden. Der Ablauf der einzelnen Unterrichtsstunden könnte wie folgt aussehen:

- **Erste Doppelstunde:**
  In der ersten Doppelstunde erhalten die Schüler eine Einführung in die mathematische Modellierung und lernen die Problemstellung kennen. Dazu werden der Modellierungsvortrag sowie die Einstiegspräsentation genutzt. Anschließend erkunden die Schüler auf dem ersten Arbeitsblatt den Datensatz. Den Abschluss der ersten Doppelstunde bildet die Diskussion der Ergebnisse von Arbeitsblatt eins und ein kurzer Ausblick auf die nächste Doppelstunde.

- **Zweite Doppelstunde:**
  Im Rahmen der zweiten Doppelstunde bearbeiten die Schüler die Arbeitsblätter zwei und drei. Sie führen also zum einen die Datenvorverarbeitung durch und lernen zum anderen den kNN-Algorithmus kennen. Die Erkenntnisse der beiden Arbeitsblätter werden in kurzen Plenumsdiskussionen gesichert.

- **Dritte Doppelstunde:**
  Nachdem die Schüler in der zweiten Doppelstunde den kNN-Algorithmus erarbeitet haben, bewerten sie auf Arbeitsblatt vier die Güte dieses Algorithmus mit Hilfe der Wahrheitsmatrix und den verschiedenen Qualitätsmaßen. Im Anschluss an eine kurze Sicherung starten die Schüler mit der Verbesserung des Vorgehens bei der Klassifikation, indem sie auf Arbeitsblatt fünf das Feature-Set 1 erweitern. Den Abschluss der dritten Doppelstunde bildet die Ergebnissicherung des fünften Arbeitsblattes.

- **Vierte Doppelstunde:**
  In der letzten Doppelstunde schließen die Schüler auf dem sechsten Arbeitsblatt die Verbesserung des Klassifikationsalgorithmus durch die Optimierung des Parameters k ab. Außerdem untersuchen sie auf Arbeitsblatt sieben Probleme und Schwierigkeiten der Aktivitätserkennung. Zudem notieren sie Ideen zu Anwendungen und Einsatzgebieten. Den Abschluss der Unterrichtseinheit bildet eine kritische Diskussion zum Thema Aktivitätserkennung sowie eine Wiederholung und Zusammenfassung der einzelnen Schritte des Lernmoduls.

Der Einsatz des Lernmoduls im Rahmen einer Unterrichtseinheit im Fach Mathematik wurde bisher nur in einer auf zwei Doppelstunden gekürzten Version durchgeführt. Näheres zur Durchführung und den Beobachtungen ist in Kapitel 5 zu finden.

4.6. Erstellte Materialien

Im Folgenden werden die im Rahmen dieser Arbeit erstellten Materialien vorgestellt. Zu Beginn werden die für die Schüler relevanten Materialien in der Reihenfolge, in der sie bei der Durchführung des Lernmoduls zum Einsatz kommen, beschrieben. Anschließend wird auf das zusätzliche Material eingegangen, das die Dozenten bei der Durchführung des Lernmoduls unterstützen soll. Die Vorträge, die zur Vorstellung von CAMMP, zum Einstieg in die mathematische Modellierung und zur Präsentation weiterer Angebote von CAMMP genutzt werden, wurden von Mitarbeitern des Projekts CAMMP erstellt. Daher wird auf diese Präsentationen nicht weiter eingegangen. Alle erstellten Materialien sind im Anhang zu finden.

4.6.1. Einstiegspräsentation

Die Einstiegspräsentation (s. Anh. A.1 und A.2) dient der Einführung in das Thema des Lernmoduls. Die Schüler lernen im Rahmen dieser Präsentation die Problemstellung kennen, mit der sie sich während des Modellierungstages bzw. der Unterrichtseinheit beschäftigen.

Zu Beginn der Präsentation wird den Schülern die Fragestellung des Lernmoduls „Wie können Smartphones die Aktivitäten ihrer Nutzer erkennen und was hat das mit Mathe zu tun?“ vorgestellt. Danach wird der Prozess der menschlichen Aktivitätserkennung erläutert. Dabei wird bereits kurz auf die Sensordaten, die Datenvorverarbeitung und die Klassifizierung mit Hilfe von KI eingegangen.

Anschließend folgt ein Exkurs zum Thema KI und maschinelles Lernen. Zunächst sollen sich die Schüler Gedanken zu den folgenden beiden Fragestellungen machen:

- Was versteht ihr unter dem Begriff KI?
- Welche Anwendungen der KI kennt ihr?

Nach einer kurzen Diskussion zu diesen beiden Fragen wird anhand des Turing-Tests auf das Thema KI näher eingegangen. Anschließend wird das maschinelle Lernen als Teilbereich der KI erläutert. Das grundlegende Prinzip des maschinellen Lernens wird am Beispiel eines Spamfilters erklärt. Den Abschluss des Exkurses zu KI bildet die Fragestellung, wieso bei der Aktivitätserkennung ein KI-System benötigt wird. Zur Klärung dieser Frage wird die Klassifizierung der Sensordaten durch ein herkömmliches Computerprogramm der Klassifizierung mit Hilfe eines maschinellen Lernverfahrens gegenübergestellt.

Im Anschluss werden das Ziel, die Entwicklung eines eigenen Klassifikationsalgorithmus, sowie der Ablauf des Lernmoduls vorgestellt. Dieser gliedert sich in die folgenden sechs Schritte:

1. Erkunden des Datensatzes,
2. Vorverarbeitung der Daten,
3. Schrittweise Entwicklung eines eigenen Klassifikationsalgorithmus,
4. Bewertung der Ergebnisse des Klassifikationsalgorithmus mit Hilfe verschiedener Qualitätsmaße,
5. Verbesserung des entwickelten Klassifikationsalgorithmus,
6. Diskussion von Problemen sowie Anwendungsgebieten der menschlichen Aktivitätserkennung.

Außerdem wird die Problemstellung des Lernmoduls auf den zuvor im Modellierungsvortrag kennengelernten Modellierungskreislauf übertragen. Das reale Problem ist die Aktivitätserkennung mit Hilfe von Sensordaten eines Smartphones. Dieses wird durch drei Annahmen bzw. Entscheidungen vereinfacht:

- Es werden nur Daten des Beschleunigungsmessers berücksichtigt.
- Es sollen nur wenige Aktivitäten erkannt werden.
- Das Smartphone ist immer gleich positioniert und befindet sich bei der Datenaufnahme senkrecht in der rechten vorderen Hosentasche.

Diese Annahmen werden zusammen mit den Schülern besprochen. Während des Modellierungstages bzw. der Unterrichtseinheit wird der Modellierungskreislauf immer wieder aufgegriffen, um den Fortschritt im Modellierungsprozess aufzuzeigen. Er dient somit den Schülern bei der Bearbeitung des Lernmoduls als Orientierung.

Den Abschluss der Einstiegspräsentation bilden Hinweise zur Bearbeitung der Arbeitsblätter sowie eine Erläuterung zum Zugriff auf das Material. Gemeinsam mit den Schülern öffnet der Dozent das Jupyter Notebook *AB1-SuS.ipynb* und erklärt kurz die wesentliche Struktur der Arbeitsblätter mitsamt der Codefelder.

4.6.2. Arbeitsblatt 1, Antwortblatt 1 und Hilfekarte 1

Bevor die Schüler mit der Bearbeitung des ersten Arbeitsblattes (s. Anh. B.1) beginnen, erhalten sie von den Dozenten das Antwortblatt zum ersten Arbeitsblatt (s. Anh. E.1). Auf diesem können sie während der Bearbeitung des Arbeitsblattes ihre Erkenntnisse und Ergebnisse festhalten.

Zu Beginn des ersten Arbeitsblattes wird noch einmal kurz das Thema des Lernmoduls, die Aktivitätserkennung auf dem Smartphone, vorgestellt sowie die Aufnahme der Sensordaten erläutert. Dadurch haben die Schüler die konkrete Problemstellung immer vor Augen. Ziel des ersten Arbeitsblattes ist das Erkunden des Datensatzes. Dabei beantworten die Schüler die folgenden Fragen:

1. Aus welchen Daten besteht ein Datenpunkt im Datensatz?
2. Zu wie vielen Aktivitäten wurden Daten aufgenommen?
3. Welche Aktivitäten sind dies?
4. Wie viele Testpersonen haben Daten aufgenommen?
5. Wie viele Testpersonen sind weiblich und wie viele sind männlich?
6. Wie alt ist die jüngste bzw. älteste Testperson?
7. Wie lange wurden Daten zu den jeweiligen Aktivitäten von den Testpersonen aufgenommen?
8. Mit welcher Abtastrate wurden die Daten aufgenommen?

In der ersten Aufgabe werden die Aktivitäten, zu denen Sensordaten aufgenommen wurden, vorgestellt. Im ersten Teil dieser Aufgabe wird der Datensatz in Form einer Tabelle angezeigt. Diese können die Schüler nach unterschiedlichen Spalten sortieren. In Abbildung 25 ist der Datensatz beispielsweise nach der Spalte ActivityID sortiert.

	UserID	ActivityID	Time (s)	x (m/s^2)	y (m/s^2)	z (m/s^2)
0	1	1	0.028140	5.480853	-4.508624	-6.869365
1	5	1	120.105365	5.624554	0.045356	-8.159083
2	5	1	120.080207	5.603448	0.041164	-8.130642
3	5	1	120.055049	5.624254	0.026794	-8.108788
4	5	1	120.029891	5.635032	0.026645	-8.147557
...	...	...	...	...	...	...
351431	6	5	123.069965	-5.208868	-9.523196	0.483944
351432	6	5	123.044808	-6.893764	-11.925401	-0.428259
351433	6	5	123.019651	-10.597213	-16.777262	-1.619183
351434	6	5	123.296378	-1.868564	-5.554799	-0.721200
351435	10	5	179.978142	3.050956	-10.346634	-3.517685

Abbildung 25: Screenshot eines Ausschnittes des Datensatzes sortiert nach der Spalte ActivityID

Anhand dieser Tabelle können die ersten beiden Fragen der obigen Aufzählung beantwortet werden, denn es sind die Daten, aus denen ein Datenpunkt im Datensatz

besteht, und die größte ActivityID zu erkennen. Im zweiten Teil der Aufgabe können sich die Schüler die Aktivitäten zu den jeweiligen ActivityID's ausgeben lassen und somit Frage drei beantworten. Den Abschluss der ersten Aufgabe stellt die graphische Darstellung der aufgenommenen Sensordaten der einzelnen Aktivitäten dar (s. Abb. 14 in Abschn. 3.4.1). Diese Graphen interpretieren die Schüler hinsichtlich folgender Fragestellungen:

- Welche Unterschiede sind zwischen den einzelnen Aktivitäten zu erkennen?
- Bei welcher Aktivität sind am wenigsten / am meisten Ausschläge zu erkennen und warum?
- Auf welcher der drei Achsen sind die größten Ausschläge zu erkennen und warum?

Dadurch soll sichergestellt werden, dass sich die Schüler intensiv mit den Graphen beschäftigen.

In der nächsten Aufgabe geht es um die Testpersonen, die Daten zu den einzelnen Aktivitäten aufgenommen haben. Den Schülern wird eine Tabelle mit Informationen zu den einzelnen Testpersonen gegeben. Mit Hilfe dieser Tabelle lassen sich die Fragen vier bis sechs beantworten.

In der letzten Aufgabe werden die Dauer der Datenaufnahme und die Abtastrate bestimmt, sodass die letzten beiden Fragen beantwortet werden können. Dazu werden den Schülern die Daten zu Aktivität 1 von Testperson 1 ausgeben. Anhand der Zeit des ersten und letzten Datenpunktes kann die Dauer der Datenaufnahme bestimmt werden. Außerdem wird den Schülern die Anzahl der Datenpunkte zu Aktivität 1 von Testperson 1 angezeigt. Mit dieser Information und der zuvor bestimmten Dauer der Datenaufnahme können die Schüler anschließend die Abtastrate bestimmen. Für Schüler, die den Begriff der Abtastrate nicht kennen, wurde eine kurze Erläuterung geschrieben, die sich die Lernenden bei Bedarf anschauen können. Außerdem wurde die Hilfekarte eins (s. Anh. D.1) erstellt. Diese soll Schüler unterstützen, die bei der Berechnung der Abtastrate auf Probleme stoßen.

Für Schüler, die das erste Arbeitsblatt besonders schnell bearbeiten, wurde eine Zusatzaufgabe erstellt. In dieser recherchieren die Schüler weitere Sensoren, die in Smartphones eingebaut sind, und überlegen sich, welche Informationen diese Sensoren mit Blick auf die Aktivitätserkennung liefern könnten.

4.6.3. Plenumsdiskussion 1

Nach der Bearbeitung des ersten Arbeitsblattes werden die Ergebnisse im Plenum gesichert. Diese Sicherung wird durch einen Dozenten moderiert und orientiert sich an den Präsentationsfolien sowie den Notizen für die erste Plenumsdiskussion (s. Anh. A.3 und A.4).

Zum einen nennen und erläutern die Schüler ihre Antworten zu den acht Fragen, die zu Beginn des ersten Arbeitsblattes gestellt wurden. Zum anderen wird gemeinsam im Plenum die Interpretationsaufgabe zu den graphischen Darstellungen der Sensordaten der einzelnen Aktivitäten besprochen. Darüber hinaus wird der Modellierungskreislauf aufgegriffen, um gemeinsam mit den Schülern den absolvierten Modellierungsschritt, die *Vereinfachung*, zu reflektieren und kurz auf den nächsten Schritt einzugehen.

4.6.4. Arbeitsblatt 2, Antwortblatt 2 und Hilfekarten 2 und 3

Auf dem zweiten Arbeitsblatt (s. Anh. B.2 und B.3) führen die Schüler die Datenvorverarbeitung durch. Auch hier steht wieder ein Antwortblatt (s. Anh. E.2) zur Verfügung, um Erkenntnisse während der Bearbeitung festzuhalten. Im Rahmen der Datenvorverarbeitung muss der Betrag des Beschleunigungsvektors gebildet werden. Da der Betrag eines Vektors erst am Ende der Klassenstufe 10 behandelt wird, wurde das zweite Arbeitsblatt auf zwei Niveaustufen erstellt, sodass das Lernmodul auch zu Beginn der zehnten Klasse durchgeführt werden kann.

Im ersten Teil der ersten Aufgabe werden die Beschleunigungswerte der einzelnen Achsen kombiniert, indem der Betrag des Beschleunigungsvektors bestimmt wird. Schülern, die Schwierigkeiten beim Aufstellen der Formel für den Betrag haben, steht die Hilfekarte zwei (s. Anh. D.2) zur Verfügung. Diese ist in zwei gestufte Tipps unterteilt. Im zweiten Teil der Aufgabe wird der Betrag des Beschleunigungsvektors als zusätzliche Spalte dem Datensatz hinzugefügt. Hier müssen die Schüler lediglich das Codefeld ausführen und ihnen wird anschließend der erweiterte Datensatz angezeigt. Für Schüler, die zu Beginn der zehnten Klasse den Betrag eines Vektors noch nicht kennen, wurden die beiden ersten Aufgabenteile auf der Kurzversion des zweiten Arbeitsblattes zusammengefasst. Die Schüler müssen in dieser Version die Formel für den Betrag eines Vektors nicht selbst herleiten, stattdessen wird ihnen direkt der Datensatz mit der zusätzlichen Spalte für den Betrag des Beschleunigungsvektors ausgegeben. Für Schüler, die dennoch wissen möchten, wie der Betrag eines Vektors bestimmt wird, wurde das Infoblatt zu Vektoren und deren Betrag (s. Anh. C.1) erstellt. Die restlichen Aufgaben des zweiten Arbeitsblattes sind in beiden Niveaustufen identisch.

In der zweiten Aufgabe werden die Daten in kleinere Zeitfenster einer festen Länge unterteilt. Dazu überlegen sich die Schüler im ersten Teil der Aufgabe eine sinnvolle Länge für die Windows. Anschließend wird im zweiten Teil der Aufgabe der Einheitlichkeit wegen die Länge der Windows auf drei Sekunden festgelegt und jedem Datenpunkt eine WindowID zugeordnet. Diese wird analog zum Betrag des Beschleunigungsvektors als zusätzliche Spalte dem Datensatz hinzugefügt. Auch hier müssen die Schüler nur den Code ausführen und bekommen anschließend den erweiterten Datensatz ausgegeben.

In Aufgabe drei geht es um die Wahl der Features. Im ersten Teil der Aufgabe notieren die Schüler auf ihrem Antwortblatt statistische Kenngrößen, die sie bereits aus dem Mathematikunterricht kennen. Für alle Schüler, die den Begriff statistische Kenngröße

nicht kennen, wurde eine kurze Erläuterung erstellt, die sich die Schüler bei Bedarf durchlesen können. Anschließend untersuchen die Schüler erneut die graphischen Darstellungen der Sensordaten der einzelnen Aktivitäten und überlegen, mit welchen statistischen Kenngrößen sie diese Daten beschreiben können. Sie halten ihre Erkenntnisse auf dem Antwortblatt fest. Im dritten Teil der Aufgabe werden für sieben Datenpunkte aus dem Window mit der WindowID 121 die drei ausgewählten statistischen Kenngrößen Minimum, Maximum und Mittelwert des Betrags des Beschleunigungsvektors berechnet. Schülern, die bei der Berechnung der statistischen Kenngrößen auf Probleme stoßen, steht die Hilfekarte drei (s. Anh. D.3) zur Verfügung. Da jedes Window aus ca. 120 Datenpunkten besteht, wäre es sehr aufwändig die statistischen Kenngrößen für alle Datenpunkte eines Windows per Hand zu berechnen. Daher können sich die Schüler im vorletzten Teil der dritten Aufgabe die statistischen Kenngrößen für beliebige Windows ausgeben lassen. Im letzten Teil dieser Aufgabe wird das Feature-Set 1 erstellt und ausgegeben.

4.6.5. Plenumsdiskussion 2

Im Anschluss an die Bearbeitung des zweiten Arbeitsblattes werden die Ergebnisse gesichert. Dazu liegen Präsentationsfolien sowie Notizen zu den einzelnen Folien bereit (s. Anh. A.5 und A.6).

Zunächst werden kurz die Ergänzungen des Datensatzes um die Spalte des Betrags des Beschleunigungsvektors sowie um die Spalte der WindowID thematisiert. Anschließend nennen die Schüler die statistischen Kenngrößen, die sie bereits aus dem Mathematikunterricht kennen, und erläutern, welche dieser Kenngrößen sie zur Beschreibung der Daten eines Windows nutzen würden. Im Anschluss an die Sicherung der Ergebnisse wird erneut der Modellierungskreislauf aufgegriffen, um den Fortschritt im Modellierungsprozess zu reflektieren. Im Zuge dessen wird auch der nächste Schritt, die Zuordnung der Daten der einzelnen Windows zu den verschiedenen Aktivitäten, erläutert. Dazu wird eine kurze Einführung in Klassifizierungsprobleme gegeben und die Grundidee des kNN-Algorithmus anhand eines einfachen Beispiels (s. Abb. 8 in Abschn. 3.1.3) mit den Schülern erarbeitet.

4.6.6. Arbeitsblatt 3, Antwortblatt 3 und Hilfekarten 4 und 5

Ziel des dritten Arbeitsblattes (s. Anh. B.4 und B.5) ist die Implementierung des kNN-Algorithmus. Zur Dokumentation der Ergebnisse steht wieder ein Antwortblatt (s. Anh. E.3) zur Verfügung. Das dritte Arbeitsblatt wurde wie das zweite Arbeitsblatt in zwei Niveaustufen erstellt. Zur Implementierung des kNN-Algorithmus muss der euklidische Abstand zweier Punkte bzw. Vektoren bestimmt werden. Dies ist Inhalt der zehnten Klasse, sodass für Schüler, die gerade die neunte Klasse absolviert haben, eine Kurzversion des dritten Arbeitsblattes entwickelt wurde.

Zur Festigung des Verständnisses der Grundidee des kNN-Algorithmus, die zuvor in der zweiten Plenumsdiskussion erarbeitet wurde, ordnen die Schüler in der ersten Aufgabe des dritten Arbeitsblattes den roten Datenpunkt aus Abbildung 8 in Abschnitt 3.1.3 einer der drei Klassen (blaue Sterne, orangene Quadrate und grüne Kreise) zu.

In der zweiten Aufgabe wird eine Abstandsfunktion definiert, mit der der Abstand zwischen den Datenpunkten zweier Windows bestimmt werden kann. Zuvor schauen sich die Schüler im ersten Teil der zweiten Aufgabe die Datenpunkte aller Windows der Person mit UserID 1 graphisch an (s. Abb. 18 in Abschn. 3.4.3). Diese Darstellung wird hinsichtlich der Fragestellung, bei welchen Aktivitäten der kNN-Algorithmus bei der Klassifikation Schwierigkeiten haben könnte, interpretiert. Dadurch wird sichergestellt, dass sich die Schüler intensiv mit der Darstellung auseinandersetzten. Im Anschluss stellen die Schüler eine Formel für den euklidischen Abstand zweier Punkten auf, um den Abstand zwischen den Datenpunkten zweier beliebiger Windows berechnen zu können. Für Schüler, die bei dieser Aufgabe Schwierigkeiten haben, liegt Hilfekarte vier (s. Anh. D.4) bereit. Diese ist in zwei gestufte Tipps aufgeteilt. Auf der Kurzversion des dritten Arbeitsblattes müssen die Schüler die Formel für den Abstand nicht selbst herleiten, stattdessen berechnen sie mit einem vorgegebenen Code den Abstand zwischen den Datenpunkten zweier beliebiger Windows. Für interessierte Schüler wurde ein Infoblatt (s. Anh. C.2) erstellt, auf dem die Berechnung des Abstands zweier Datenpunkte erläutert wird.

Nach dem Aufstellen einer Funktion zur Abstandsberechnung werden in Aufgabe drei die k nächsten Nachbarn eines beliebigen Test-Windows bestimmt. Auch diese Aufgabe wurde auf zwei Niveaustufen erstellt. Während auf der Kurzversion die Funktion zur Bestimmung der k nächsten Nachbarn als Blackbox angegeben wird, entwickeln die Schüler auf der Langversion einen eigenen Code zur Bestimmung der k nächsten Nachbarn. Auch hier wurde für interessierte Schüler, die die Kurzversion bearbeiten, ein Infoblatt (s. Anh. C.3) erstellt, auf dem der Code zur Suche der k nächsten Nachbarn schrittweise erklärt wird. Auf der langen Version des dritten Arbeitsblattes schreiben die Schüler anhand verschiedener Funktionen, die ihnen zur Verfügung gestellt werden (s. Abb. 26), und eines vorgefertigten Codegerüsts (s. Abb. 27) einen Code, mit dem die k nächsten Nachbarn bestimmt werden können. Zur Unterstützung der Schüler, die beim eigenständigen Schreiben des Codes auf Probleme stoßen, wurde die gestufte Hilfekarte fünf (s. Anh. D.5) entwickelt. Nach erfolgreicher Eingabe des Codes erhalten die Schüler die k nächsten Nachbarn des Test-Windows als Ausgabe. Basierend auf den k nächsten Nachbarn legen die Schüler in einer abschließenden Interpretationsaufgabe die Aktivität via Mehrheitsentscheid fest.

In Aufgabe vier wird der Mehrheitsentscheid für beliebige Test-Windows computergestützt durchgeführt. Diesen müssen die Schüler nicht selbst implementieren. Stattdessen wird eine Funktion als Blackbox gegeben, die die Aktivität ausgibt, die unter den k nächsten Nachbarn am häufigsten vorkommt. Anschließend wird im zweiten Aufgabenteil überprüft, ob die Aktivität, die mit dem kNN-Algorithmus dem Test-Window

zugeordnet wurde, mit der tatsächlichen Aktivität, zu der die Daten des Test-Windows aufgenommen wurden, übereinstimmt.

- `sortDistances(distances)`:
  Mit dieser Funktion kannst du die Abstände in der Liste `distances` der Größe nach (von klein zu groß) sortieren.
- `computeDistance(i, TestWindow)`:
  Diese Funktion liefert den Abstand zwischen dem Datenpunkt mit der WindowID = i und dem Datenpunkt des Test-Windows.
- `getNeighbors(distances, k)`:
  Diese Funktion liefert dir die k ersten Einträge der Liste `distances`.
- `addDistances(i, dist, distances)`:
  Mit dieser Funktion kannst du den berechneten Abstand `dist` zur Liste `distances` aller Abstände hinzufügen.

Abbildung 26: Zur Verfügung stehende Funktionen zur Suche der k nächsten Nachbarn

```
[ ]: k = NaN;
TestWindow = NaN;

# Leere Liste, in der nach und nach die Abstände gespeichert werden sollen.
distances = []

# For-Schleife zur Berechnung der Abstände zwischen allen Windows und dem Test-Window
# und Abspeichern in der Liste distances
for i in range(1, 3000):
    dist = NaN
    NaN

# Löschen des Abstands des Test-Windows zu sich selbst
del distances[TestWindow-1]

# Sortieren der Liste nach den kürzesten Abständen zum Test-Window
NaN

# Filtern der k nächsten Nachbarn
kneighbors = NaN

# Hier nichts ändern!
checkKNeighbors(kneighbors, k, TestWindow)
```

Abbildung 27: Codegerüst für den Code zur Suche der k nächsten Nachbarn

Schnellere Schüler können an dieser Stelle des Lernmoduls erneut eine Zusatzaufgabe bearbeiten. Bei dieser recherchieren die Schüler im Internet weitere Möglichkeiten zur Abstandsberechnung. Anschließend können sie ihre Rechercheergebnisse mit den Informationen auf dem Infoblatt zu weiteren Abstandsfunktionen (s. Anh. C.4) abgleichen. Diese Zusatzaufgabe ist nur auf der langen Version des dritten Arbeitsblattes zu finden.

4.6.7. Plenumsdiskussion 3

Die Ergebnisse des dritten Arbeitsblattes werden mit Hilfe der Plenumsdiskussion drei (s. Anh. A.7 und A.8) gesichert.

Zunächst wird anhand der graphischen Darstellung der Datenpunkte aller Windows der Person mit UserID 1 diskutiert, bei welchen Aktivitäten der kNN-Algorithmus bei der Klassifikation Schwierigkeiten haben könnte. Anschließend wird, falls die lange Version des dritten Arbeitsblattes bearbeitet wurde, die Definition der Abstandsfunktion sowie das Vorgehen bei der Suche nach den k nächsten Nachbarn besprochen. Unabhängig von der bearbeiteten Niveaustufe wird der Mehrheitsentscheid am Beispiel der fünf nächsten Nachbarn des Test-Windows 200 thematisiert. Abschließend wird der Modellierungskreislauf aufgegriffen, um gemeinsam mit den Schülern den Fortschritt im Modellierungsprozess zu reflektieren. Zudem wird das Prinzip des überwachten maschinellen Lernens vorgestellt, mit Hilfe dessen auf dem nächsten Arbeitsblatt die Güte des Klassifikationsalgorithmus anhand der Wahrheitsmatrix und den verschiedenen Qualitätsmaßen bewertet werden soll.

4.6.8. Arbeitsblatt 4, Antwortblatt 4 und Hilfekarten 6,7 und 8

Auf Arbeitsblatt vier (s. Anh. B.6) sollen die Klassifikationsergebnisse mit Hilfe der Wahrheitsmatrix und den verschiedenen Qualitätsmaßen bewertet werden. Zur Dokumentation der Ergebnisse können die Schüler das Antwortblatt vier (s. Anh. E.4) nutzen.

Zur Bewertung der Güte des kNN-Algorithmus wird das Prinzip des überwachten maschinellen Lernens eingesetzt. Dieses wird zu Beginn des vierten Arbeitsblattes im Rahmen einer kurzen Einleitung erläutert, sodass sichergestellt wird, dass alle Schüler dieses Prinzip kennen und verstehen.

In der ersten Aufgabe werden die Daten des Feature-Sets 1 in Trainings- und Testdaten aufgeteilt. Dazu werden die Daten im ersten Teil der Aufgabe in Features und Aktivitäten unterteilt und jeweils in einem Array abgespeichert. Anschließend werden 80 % der Daten dem Trainingsdatensatz und 20 % der Daten dem Testdatensatz zufällig zugeteilt. In beiden Aufgabenteilen müssen die Schüler nur die Codefelder ausführen und nichts selbst programmieren, da nicht zu erwarten ist, dass die Schüler über die erforderlichen Programmierkenntnisse verfügen.

In Aufgabe zwei lernen die Schüler die Wahrheitsmatrix kennen. Dazu werden im ersten Aufgabenteil die Testdaten mit $k = 3$ und dem Feature-Set 1 klassifiziert. Die Klassifikation wird im Hintergrund durchgeführt und den Schülern werden nach dem Ausführen des Codefelds die Ergebnisse in Form einer Wahrheitsmatrix ausgegeben. Diese untersuchen die Schüler anschließend hinsichtlich der folgenden Fragen:

- Wie viele Datenpunkte wurden insgesamt richtig klassifiziert?
- Wie viele Datenpunkte wurden insgesamt falsch klassifiziert?
- Bei welcher Aktivität wurden die meisten Datenpunkte richtig klassifiziert?
- Bei welcher Aktivität wurden die meisten Datenpunkte falsch klassifiziert?
- Bei welchen beiden Aktivitäten liegen die meisten Überschneidungen vor?

Dadurch wird sichergestellt, dass sich die Schüler verständig mit der Wahrheitsmatrix und deren Einträgen auseinandersetzen.

Inhalt der dritten Aufgabe sind die drei Qualitätsmaße Genauigkeit, Fehlerrate und Präzision. Die Schüler stellen in den jeweiligen Aufgabenteilen basierend auf den Einträgen der Wahrheitsmatrix die Formeln zur Berechnung der einzelnen Qualitätsmaße auf. Bei auftretenden Schwierigkeiten können sich die Schüler die Hilfekarten sechs, sieben und acht (s. Anh. D.6, D.7 und D.8) anschauen. Zu jedem Qualitätsmaß wurde eine zweistufige Hilfekarte erstellt. Falls die Formel für ein Qualitätsmaß richtig aufgestellt wurde, wird den Schülern der Wert dieses Qualitätsmaßes ausgegeben. Den Abschluss der dritten Aufgabe bildet ein Interpretationsauftrag. Die Schüler beurteilen die Werte der einzelnen Qualitätsmaße anhand der folgenden Fragen:

- Was bedeutet es, wenn der Wert der Präzision nahe bei 1 liegt? Bei welchen Aktivitäten ist dies der Fall?
- Bei welcher Aktivität werden die Datenpunkte am häufigsten richtig zugeordnet und warum?
- Bei welchen beiden Aktivitäten werden die Datenpunkte am häufigsten falsch zugeordnet und was könnten Gründe dafür sein?
- Sind die Ergebnisse der Klassifikation im Großen und Ganzen zufriedenstellend? Falls nein, wo müssten die Ergebnisse noch verbessert werden?

Dadurch soll das Verständnis der einzelnen Qualitätsmaße gefestigt werden.

In der letzten Aufgabe notieren die Schüler Möglichkeiten, wie das bisherige Vorgehen bei der Klassifikation verändert werden könnte, um die Ergebnisse der Klassifikation zu verbessern.

4.6.9. Zusatzblatt

Um etwaige Wartezeiten zwischen dem vierten Arbeitsblatt und der vierten Plenumsdiskussion zu minimieren, wurde das Zusatzblatt (s. Anh. B.10) erstellt. Auf diesem untersuchen die Schüler, wie sich die Ergebnisse der Klassifikation verändern, wenn anstelle der euklidischen Metrik eine andere Metrik zur Abstandsberechnung genutzt

wird. Für Schüler, die auf Arbeitsblatt drei die Zusatzaufgabe nicht bearbeitet haben, wurde zu Beginn des Zusatzblattes das Infoblatt zu weiteren Abstandsfunktionen (s. Anh. C.4) verlinkt. Auf diesem werden die drei zusätzlichen Abstandsmetriken, Manhattan-Metrik, Kosinus-Metrik und Tschebyschew-Metrik, vorgestellt.

In der ersten Aufgabe des Zusatzblattes werden die Testdaten unter Verwendung der Manhattan-Metrik mit $k = 3$ und dem Feature-Set 1 klassifiziert. Die Klassifikation wird im Hintergrund durchgeführt. Den Schülern werden die Wahrheitsmatrix sowie die Werte der einzelnen Qualitätsmaße ausgegeben. Analog werden in den nächsten beiden Aufgaben die Testdaten mit der Kosinus- und der Tschebyschew-Metrik klassifiziert. Um sicherzustellen, dass sich die Schüler mit den Ergebnissen der einzelnen Klassifikationen auseinandersetzten, vergleichen sie diese hinsichtlich der folgenden Fragen:

- Welche Abstandsfunktion liefert die besten Klassifikationsergebnisse?
- Welche Abstandsfunktionen liefern bessere bzw. schlechtere Klassifikationsergebnisse im Vergleich zur euklidischen Distanz?
- Welche Abstandsfunktion liefert die schlechtesten Klassifikationsergebnisse und was könnten Gründe dafür sein?

Ihre Überlegungen halten die Schüler auf einem Antwortblatt (s. Anh. E.8) fest.

4.6.10. Plenumsdiskussion 4

Auch nach Arbeitsblatt vier werden die Erkenntnisse im Rahmen einer Plenumsdiskussion (s. Anh. A.9 und A.10) gesichert.

Zuerst wird die Aufteilung des Feature-Sets 1 in Trainings- und Testdaten thematisiert. Anschließend werden die Einträge der Wahrheitsmatrix gesichert und die Schüler erläutern ihre Antworten auf die verschiedenen Fragen zur Interpretation der Wahrheitsmatrix. Außerdem werden die Formeln und die Werte der einzelnen Qualitätsmaße besprochen. Auch hier erklären die Schüler ihre Überlegungen zu den gestellten Fragen. Im Anschluss werden verschiedene Möglichkeiten diskutiert, wie das Vorgehen verändert werden kann, um die Klassifikationsergebnisse zu verbessern. Zum Schluss wird erneut der Modellierungskreislauf aufgegriffen. Es wird gezeigt, dass dieser nun das erste Mal komplett durchlaufen wurde, und ein kurzer Ausblick auf die nächsten beiden Schritte, die Erweiterung des Feature-Sets 1 und die Optimierung des Parameters k, gegeben.

4.6.11. Arbeitsblatt 5, Antwortblatt 5 und Hilfekarte 9

Ziel des fünften Arbeitsblattes (s. Anh. B.7) ist es, die Ergebnisse der Klassifikation durch eine Erweiterung des Feature-Sets 1 zu verbessern. Auf Antwortblatt fünf (s. Anh. E.5) können die Schüler ihre Ergebnisse und Erkenntnisse festhalten.

In der ersten Aufgabe wird das Feature-Set 1 um den Mittelwert, das Maximum und das Minimum jeder der drei Beschleunigungsrichtungen ergänzt. Diese neuen Features werden für alle 3000 Windows berechnet und als zusätzliche Spalten an das Feature-Set 1 angehängt. Die Schüler müssen dies nicht selbst programmieren. Stattdessen wird ihnen das erweiterte Feature-Set 1 nach der Ausführung des Codefelds ausgegeben.

Inhalt der zweiten Aufgabe ist die Berechnung neuer Features. Dazu notieren die Schüler im ersten Aufgabenteil statistische Kenngrößen, die sie neben dem Minimum, Maximum und Mittelwert bereits aus dem Mathematikunterricht kennen. Anschließend berechnen sie im zweiten Aufgabenteil die Kenngrößen Median sowie das obere und das untere Quartil für sieben Datenpunkte aus dem Window mit der WindowID 121. Schülern, die diese Kenngrößen nicht kennen oder nicht mehr wissen, wie diese bestimmt werden, steht Hilfekarte neun (s. Anh. D.9) zur Verfügung. Diese ist in zwei gestufte Tipps unterteilt. Im nächsten Aufgabenteil können sich die Schüler die neuen statistischen Kenngrößen für beliebige Windows ausgeben lassen. Bei der im Hintergrund ablaufenden Berechnung werden alle 120 Datenpunkte eines Windows berücksichtigt. Im letzten Teil der zweiten Aufgabe wird das Feature-Set 2 erstellt, das aus dem Feature-Set 1 durch Ergänzung der neuen statistischen Kenngrößen entsteht. Hierzu müssen die Schüler lediglich das Codefeld ausführen und ihnen wird das neue Feature-Set 2 ausgegeben.

Anschließend werden in der dritten Aufgabe die Testdaten mit Feature-Set 2 und $k = 3$ klassifiziert. Die Klassifikation verläuft wieder im Hintergrund. Den Schülern werden die Wahrheitsmatrix sowie die Werte der unterschiedlichen Qualitätsmaße nach dem Ausführen des Codefelds ausgegeben. Im Anschluss vergleichen die Schüler die Ergebnisse mit denen von Arbeitsblatt vier und beantworten dabei die folgenden Fragen:

- Was fällt dir auf, wenn du die Qualitätsmaße mit den Qualitätsmaßen der Klassifikation auf Arbeitsblatt vier vergleichst?
- Bei welcher Aktivität treten noch am häufigsten falsche Klassifikationen auf?
- Wie viele von 100 Aktivitäten werden im Mittel richtig klassifiziert?

4.6.12. Plenumsdiskussion 5

Im Rahmen der fünften Plenumsdiskussion (s. Anh. A.11 und A.12) werden die Ergebnisse des fünften Arbeitsblattes gesichert.

Zunächst nennen die Schüler weitere statistische Kenngrößen, die sie bereits aus dem Mathematikunterricht kennen. Anschließend wird das Feature-Set 2, das nun aus 15 verschiedenen Features besteht, besprochen. Außerdem werden die Einträge der Wahrheitsmatrix sowie die Werte der einzelnen Qualitätsmaße für die Klassifikation mit dem Feature-Set 2 und $k = 3$ gesichert sowie die drei Fragen am Ende des fünften Arbeitsblattes geklärt. Den Abschluss bildet wieder der Modellierungskreislauf mit einem Ausblick auf den nächsten Schritt im Modellierungsprozess.

4.6.13. Arbeitsblatt 6, Antwortblatt 6 und Hilfekarte 10

Auf Arbeitsblatt sechs (s. Anh. B.8) sollen die Ergebnisse der Klassifikation durch eine Optimierung des Parameters k verbessert werden. Es wird der Wert für k gesucht, für den die Fehlerrate minimal ist. Auch hier können die Schüler ihre Lösungen auf einem Antwortblatt (s. Anh. E.6) festhalten.

Das Optimierungsproblem wird in der ersten Aufgabe zunächst graphisch gelöst. Dazu wird den Schülern ein Graph, in dem die Fehlerrate gegen die Anzahl der Nachbarn k aufgetragen ist, ausgegeben (s. Abb. 20 in Abschn. 3.4.5). Um sicherzustellen, dass sich die Schüler verständig mit dem Graphen auseinandersetzen, sollen sie die folgenden beiden Fragen beantworten:

- Für welchen Wert von k ist die Fehlerrate am kleinsten?
- Warum schwankt die Fehlerrate für kleine k stark? Wie hängen die Schwankungen mit dem Wert von k zusammen?

In der zweiten Aufgabe wird das Optimierungsproblem mit Hilfe von Python gelöst. Die Schüler schreiben einen eigenen Code, der den Wert von k ausgibt, für den die Fehlerrate minimal ist. Den Schülern werden verschiedene Funktionen sowie ein vorgefertigtes Codegerüst zur Verfügung gestellt (s. Abb. 28 und Abb. 29). Für Schüler, die beim Schreiben des Codes Probleme haben, wurde Hilfekarte zehn (s. Anh. D.10) erstellt. Diese ist in zwei gestufte Tipps unterteilt. Falls der Code der Schüler korrekt ist, wird der Wert für k ausgegeben, für den die Fehlerrate minimal ist.

Im Anschluss an die Optimierung des Parameters k werden die Testdaten erneut klassifiziert, diesmal mit $k = 1$ und dem Feature-Set 2. Auch hier wird die Klassifikation im Hintergrund ausgeführt. Den Schülern werden die Wahrheitsmatrix sowie die Werte der einzelnen Qualitätsmaße ausgegeben. Anschließend werden die Ergebnisse mit den vorherigen Klassifikationsergebnissen verglichen. Dazu beantworten die Schüler erneut die drei Fragen vom Ende des fünften Arbeitsblattes.

- `L.append(x)` :
  Mit dieser Funktion kannst du den Wert x der Liste `L` hinzufügen.
- `L.index(x)` :
  Diese Funktion liefert dir die Position des Werts x in der Liste `L` (**Achtung:** Python beginnt bei der Nummerierung der Einträge einer Liste bei 0).
- `computeErrorRate(i)` :
  Diese Funktion liefert dir die Fehlerrate für $k = i$.
- `min(L)` :
  Mit dieser Funktion kannst du dir den minimalen Wert einer Liste `L` ausgeben lassen.

Abbildung 28: Zur Verfügung stehende Funktionen zur Optimierung des Parameters k

```
[ ]: # Leere Liste, in der die Fehlerraten abgespeichert werden
errorRates = []

# For-Schleife zur Berechnung der Fehlerraten und
# Abspeichern der Fehlerraten in der Liste errorRates
for i in range(1,41):
    errorRate_i = NaN
    NaN

# Suchen des kleinsten Werts in der Liste errorRates
min_errorRates = NaN

# Position des kleinsten Werts in der Liste errorRates
pos_min = NaN

# Optimales k, für das die Fehlerrate am kleinsten ist
k = NaN

# Hier nichts ändern!
checkPosMinErrorRate(k, Min_ErrorRates, Pos_Min)
```

Abbildung 29: Codegerüst für den Code zur Optimierung des Parameters k

4.6.14. Plenumsdiskussion 6

In Plenumsdiskussion sechs (s. Anh. A.13 und A.14) werden die Ergebnisse des sechsten Arbeitsblattes gesichert.

Zunächst wird der Graph, in dem die Fehlerrate gegen die Anzahl der Nachbarn k aufgetragen ist, gemeinsam mit den Schülern hinsichtlich der beiden Fragestellungen aus der ersten Aufgabe analysiert. Im Anschluss wird das algorithmische Vorgehen bei der Lösung des Optimierungsproblems mit Hilfe von Python besprochen. Zudem werden die Ergebnisse der Klassifikation der Testdaten mit $k = 1$ und dem Feature-Set 2 mit den vorherigen Klassifikationsergebnissen verglichen und die Schüler erläutern ihre Antworten zu den drei gestellten Fragen. Den Abschluss der Plenumsdiskussion stellt wieder die Einordnung in den Modellierungskreislauf dar. Dieser wurde nun erneut komplett durchlaufen und es wurde eine zufriedenstellende Lösung des realen Problems gefunden.

4.6.15. Arbeitsblatt 7 und Antwortblatt 7

Zum Abschluss des Lernmoduls setzen sich die Schüler auf dem siebten Arbeitsblatt (s. Anh. B.9) einerseits mit Problemen und Schwierigkeiten und andererseits mit Anwendungen und Einsatzgebieten der Aktivitätserkennung auseinander. Zur Dokumentation der Überlegungen steht ein Antwortblatt zur Verfügung (s. Anh. E.7).

In der ersten Aufgabe werden Schwierigkeiten und Probleme der Aktivitätserkennung auf dem Smartphone thematisiert. Dazu werden Sensordaten klassifiziert, bei denen sich das Handy entweder in der Hand des Nutzers befand oder im Rucksack verstaut

war. Den Schülern werden die Ergebnisse der beiden Klassifikationen in Form einer Wahrheitsmatrix und den einzelnen Qualitätsmaßen ausgegeben. Anschließend notieren die Schüler Gründe, warum der entwickelte Klassifikationsalgorithmus die Sensordaten, bei denen sich das Smartphone nicht in der vorderen Hosentasche befand, falsch zuordnet. Zum Abschluss der ersten Aufgabe diskutieren die Schüler mit ihrem Sitznachbarn weitere Herausforderungen und Schwierigkeiten der Aktivitätserkennung.

In der letzten Aufgabe tauschen sich die Schüler innerhalb ihrer Gruppe über Anwendungen und Einsatzgebiete der Aktivitätserkennung aus und halten ihre Ergebnisse auf dem Antwortblatt fest. Dabei sollen sie besonders auf drei Bereiche eingehen: Einzelpersonen, Unternehmen und Gruppen.

4.6.16. Plenumsdiskussion 7, Abschlusspräsentation und Zusammenfassungsarbeitsblatt

Auf die Bearbeitung des siebten Arbeitsblattes folgt die siebte Plenumsdiskussion sowie die Abschlusspräsentation (s. Anh. A.15 und A.16). In diesen beiden Diskussionsphasen werden die Ergebnisse des letzten Arbeitsblattes gesichert und das komplette Lernmodul abschließend zusammengefasst.

Zunächst werden die Ergebnisse der Klassifikation der Sensordaten, bei denen sich das Handy entweder in der Hand oder im Rucksack des Nutzers befand, besprochen. Es werden Gründe, warum die Klassifikationsergebnisse schlecht ausfallen, sowie weitere Herausforderungen und mögliche Lösungswege diskutiert. Anschließend folgt eine Diskussion zu verschiedenen Anwendungen und Einsatzgebieten der Aktivitätserkennung. Dabei werden insbesondere auch kritische Aspekte und Missbrauchsmöglichkeiten aufgegriffen. Unter anderem werden der Datenschutz und die Privatsphäre sowie die Überwachung durch einen totalitären Staat angesprochen.

Zum Abschluss des Lernmoduls werden die einzelnen Schritte und die Ergebnisse zusammengefasst. Dazu wurde das Zusammenfassungsarbeitsblatt (s. Anh. E.9) erstellt. Gemeinsam mit den Schülern wird der Lückentext ausgefüllt und die wichtigsten Inhalte des Lernmoduls kompakt gesichert. Dadurch haben die Schüler am Ende des Modellierungstages bzw. der Unterrichtseinheit vor Augen, was sie erreicht haben.

4.6.17. Begleitmaterial für Dozenten

Neben den bereits vorgestellten Materialien stehen den Dozenten weitere Unterlagen zur Verfügung. Diese können einerseits zur Einarbeitung in das Thema des Lernmoduls genutzt werden und sollen die Dozenten andererseits bei der Durchführung unterstützen.

Im Basic Paper (s. Anh. F.1) werden alle fachlichen und mathematischen Grundlagen erläutert, die zur Bearbeitung des Lernmoduls benötigt werden. Dieses Dokument eig-

net sich insbesondere, um sich in das Thema das Lernmoduls einzulesen. Bei der Durchführung des Lernmoduls sollen die Dozenten durch das methodische Konzept (s. Anh. F.2) und die Musterlösung (s. Anh. F.3) unterstützt werden. Im methodischen Konzept werden alle benötigten Materialien aufgeführt und der Ablauf des Modellierungstages tabellarisch dargestellt. In der Musterlösung sind die Lösungen zu allen Aufgaben kompakt zusammengestellt, sodass diese während der Durchführung als Nachschlagewerk genutzt werden kann. Darüber hinaus wurden zu allen Präsentationsfolien Notizen erstellt, in denen kurz beschrieben wird, was die einzelnen Folien beinhalten und welche Informationen durch die Dozenten an die Schüler weitergegeben werden sollen.

5. Durchführung und Evaluation

Das im Rahmen dieser Arbeit erstellte Lernmodul wurde mit einer Schülergruppe im Rahmen eines offenen CAMMP days sowie mit einer zehnten Klasse im Rahmen einer zwei Doppelstunden umfassende Unterrichtseinheit erprobt. Im Folgenden werden die Beobachtungen der beiden Durchführungen, die Ergebnisse der Evaluationen und die daraus resultierenden Verbesserungen des Lernmoduls vorgestellt.

5.1. Rahmenbedingungen

Das entwickelte Lernmodul wurde am Freitag, den 09. September 2022 erstmalig mit einer Schülergruppe im Rahmen eines offenen CAMMP days erprobt. Der Modellierungstag wurde in Präsenz in Räumlichkeiten des Karlsruher Instituts für Technologie (KIT) durchgeführt und begann um 09:00 Uhr. Das Ende war auf 15:30 Uhr angesetzt, allerdings war bereits gegen 14:40 Uhr die Bearbeitung des Lernmoduls abgeschlossen und der Modellierungstag beendet. Von 12:15 Uhr bis 13:00 Uhr fand eine Mittagspause statt. Insgesamt nahmen neun Schüler, davon drei weiblich und sechs männlich, am Modellierungstag teil. Die Schüler kamen von unterschiedlichen Gymnasien aus der Umgebung von Karlsruhe. Die meisten besuchten die elfte, zwölfte oder dreizehnte Klasse. Zwei Schülerinnen waren erst in der neunten bzw. zehnten Klasse und ein Schüler hatte die Schule bereits abgeschlossen. Insgesamt ist davon auszugehen, dass die Schüler besonders motiviert und interessiert waren, da sie sich in den Sommerferien freiwillig für den CAMMP day angemeldet hatten.

Im Rahmen einer zwei Doppelstunden umfassenden Unterrichtseinheit wurde das Lernmodul mit einer zehnten Klasse des Kant-Gymnasiums Karlsruhe durchgeführt und somit mit einer als heterogen anzunehmenden Lerngruppe erprobt. Die beiden Doppelstunden fanden am Mittwoch, den 28. September 2022 von 11:30 Uhr bis 13:05 Uhr und am Freitag, den 30. September 2022 von 09:40 Uhr bis 11:10 Uhr in Räumlichkeiten des KIT statt. In der zehnten Klasse waren 23 Schüler, wobei beim zweiten Termin nur 19 Schüler anwesend waren, davon neun weiblich und zehn männlich. Aufgrund des deutlich kürzen Zeitrahmens wurden die Inhalte des Lernmoduls für diese Durchführung gekürzt und manche Themen der Arbeitsblätter in Plenumsdiskussionen ausgelagert.

5.2. Leitende Gesichtspunkte der Beobachtungen und Evaluation

Zur Bewertung und Verbesserung des entwickelten Lernmoduls wurde während der beiden Durchführungen das Arbeitsverhalten der Schüler beobachtet. Darüber hinaus füllten die Schüler am Ende der beiden Erprobungen einen Evaluationsbogen (s. Anh. G.1) aus. Dieser wurde an den von CAMMP erstellten Evaluationsbogen angelehnt und um spezifische, auf das Lernmodul zugeschnittene Fragen ergänzt. Somit wurde das Lernmodul sowohl aus Dozentensicht als auch aus Schülersicht beurteilt.

Die Beobachtungen während den beiden Durchführungen und die Fragen des Evaluationsbogens orientierten sich an den folgenden Gesichtspunkten, die an die in Abschnitt 4.1 vorgestellten Ziele des Lernmoduls angelehnt sind:

- **Gestaltung und Ablauf des Lernmoduls:**
  Inwieweit hat den Schülern die Gestaltung der Arbeitsblätter gefallen? Empfanden die Schüler die Aufgaben als abwechslungsreich? Waren die Lern- und Arbeitszeiten angemessen (nicht zu lange oder zu kurz mit ausreichend Pausen)? Erschienen die Plenumsdiskussionen den Schülern als sinnvoll? Konnten die Schüler eigenständig arbeiten und ihre eigenen Ideen einbringen?

- **Schwierigkeitsgrad des Lernmoduls:**
  Waren die Aufgaben zu einfach oder zu schwierig? Konnten die Schüler die Aufgaben selbständig bearbeiten? Inwieweit hat das mathematische Vorwissen der Schüler zur Bearbeitung der Aufgaben ausgereicht?

- **Verständlichkeit und Hilfestellungen:**
  Wurden die Inhalte in den Plenumsdiskussionen gut erläutert? Waren die Aufgabenstellungen auf den Arbeitsblättern klar und verständlich formuliert? Inwieweit waren die Grafiken und Abbildungen für das Verständnis der Inhalte hilfreich? In welchem Maße haben die Schüler Hilfestellungen durch die Dozenten und die Hilfekarten benötigt? Konnten die Hilfekarten die Schüler bei Schwierigkeiten ausreichend unterstützen?

- **Motivation und Interesse:**
  Inwieweit fanden die Schüler das Thema des Lernmoduls interessant? Empfanden die Schüler die Lern- und Arbeitsatmosphäre als angenehm? Inwieweit konnte durch das Lernmodul das Interesse der Schüler an Themen der angewandten Mathematik sowie der Naturwissenschaften und der Technik geweckt werden? In welchem Maße wurde das Interesse der Schüler an einem Studium oder einer Ausbildung im Bereich der angewandten Mathematik, der Naturwissenschaften oder der Technik gesteigert? Würden die Schüler einen ähnlichen Modellierungstag bzw. eine ähnliche Unterrichtseinheit noch einmal besuchen und dieses Lernmodul anderen weiterempfehlen?

- **Umgang mit Jupyter Notebooks und Python:**
  Inwieweit hatten die Schüler bei der Bearbeitung der Arbeitsblätter Schwierigkeiten im Umgang mit der Programmiersprache Python? War die kurze Einführung in Python und Jupyter Notebooks ausreichend? Empfanden die Schüler die automatischen Rückmeldungen, die sie nach der Eingabe ihrer Lösungen im Codefeld erhalten hatten, als hilfreich? In welchem Maße hat den Schülern das Arbeiten mit den Codefeldern Spaß gemacht? Hatten die Schüler bereits vor dem Modellierungstag bzw. der Unterrichtseinheit mit Jupyter Notebooks und Python gearbeitet?

- **Lernzuwachs in den Bereichen mathematische Modellierung und KI:**
  In welchem Maße wurde das Verständnis der mathematischen Modellierung durch das Lernmodul gesteigert? Inwieweit waren die Schüler in der Lage, die durch das mathematische Modell erhaltenen Resultate auf die Realsituation zu beziehen? In welchem Maße wurde das Verständnis der mathematischen Hintergründe von KI-Systemen verbessert? Haben die Schüler das Prinzip des überwachten maschinellen Lernens verstanden?

- **Lob, Kritik und Verbesserungsvorschläge:**
  Was hat den Schülern am Lernmodul gefallen und was könnte noch verbessert werden?

Die Struktur des Evaluationsbogens ist an die leitenden Gesichtspunkte angelehnt. Die einzelnen Aspekte werden im Fragebogen durch mehrere Aussagen repräsentiert, die die Schüler hinsichtlich ihrer Zustimmung bewerten können. Dazu stehen ihnen die fünf Optionen „Stimmt völlig“, „Stimmt eher“, „Stimmt eher nicht“, „Stimmt gar nicht“ und „nicht beurteilbar“ zur Auswahl. Außerdem gibt es offene Fragen, zu denen die Schüler ihre Antworten als Freitext formulieren können.

5.3. Beobachtungen und Ergebnisse der Evaluation der ersten Durchführung

In diesem Abschnitt werden die Beobachtungen und Ergebnisse der Evaluation der ersten Durchführung des Lernmoduls vorgestellt.

Im Großen und Ganzen verlief die erste Erprobung des Lernmoduls im Rahmen des offenen CAMMP days reibungslos. Die Schüler arbeiteten während des gesamten Modellierungstages sehr konzentriert und hatten kaum Verständnisfragen. Die Arbeitsblätter und die hinterlegten Codefelder funktionierten ohne Probleme und der Umgang der Schüler mit den Materialien war unproblematisch. Auch die Eingabe in den Codefeldern bereitete den Schülern keine Schwierigkeiten. Zudem wurden die Antwortblätter von den Lernenden umfangreich genutzt. Im Laufe des Tages brachten sich die Schüler immer reger in die Plenumsdiskussionen ein, sodass am Ende eine ausführliche Abschlussdiskussion geführt werden konnte.

Im Folgenden wird kurz auf die Beobachtungen während der Bearbeitung der einzelnen Arbeitsblätter und den Diskussionen im Plenum eingegangen:

- **Einstiegspräsentation:**
  Die Einstiegspräsentation verlief ohne Probleme und die Schüler hatten keinerlei Verständnisschwierigkeiten. Auf die Fragen, was sie unter KI verstehen und wo ihnen KI im Alltag begegnet, hatten zwei Schüler eine Antwort. Der eine fasste KI als ein Programm auf, dass sich selbst mit der Zeit verbessert, weil es quasi Erfahrungen sammelt. Der andere verband die Begriffe maschinelles Lernen und

neuronale Netze mit KI und nannte als Anwendung selbstfahrende Autos. Anhand der wenigen Antworten lies sich erkennen, dass sich nur ein kleiner Teil der Teilnehmenden bereits im Vorfeld mit KI und maschinellem Lernen auseinandergesetzt hat.

- **Arbeitsblatt 1 & Plenumsdiskussion 1:**
  Bei der Bearbeitung des ersten Arbeitsblattes zeigten sich nur selten Schwierigkeiten. Lediglich drei Schüler hatten bei der Bestimmung der Abtastrate kleinere Probleme, da sie die Definition der Abtastrate nicht direkt verstanden hatten. Ansonsten arbeiteten die Schüler sehr konzentriert. Auffällig war jedoch, dass die Schüler die Aufgaben sehr unterschiedlich schnell bearbeiteten. Während die erste Gruppe bereits nach einer Viertelstunde fertig war, brauchte die langsamste Gruppe eine halbe Stunde. Dadurch ist für die schnelleren Gruppen eine kurze Pause bis zur ersten Plenumsdiskussion entstanden. In der anschließenden Plenumsdiskussion konnten die Schüler alle Fragen beantworten und auch die Graphen zu den einzelnen Aktivitäten richtig interpretieren.

- **Arbeitsblatt 2 & Plenumsdiskussion 2:**
  Die Schüler bearbeiteten das zweite Arbeitsblatt sehr konzentriert und ohne Verständnisschwierigkeiten. Auch die Teilnehmer, denen aus der Schule noch keine Vektorrechnung bekannt war, konnten die Aufgaben mit Hilfe der Infoblätter bearbeiten. Während der Plenumsdiskussion beantworteten die Schüler die unterschiedlichen Fragen ohne Probleme. Auffällig war allerdings, dass nur wenigen Schüler der Begriff statistische Kenngröße bekannt war. Nachdem dieser Begriff erklärt wurde, konnten die Schüler viele verschiedene Kenngrößen nennen. Neben Maximum, Minimum, Durchschnitt und Median fiel beispielsweise auch der Begriff Steigungsänderung. Der Einstieg in den kNN-Algorithmus verlief ebenfalls ohne zu beobachtende Verständnisschwierigkeiten auf Seiten der Schüler. Bei der Zuordnung des roten X (s. Abb. 8) schlugen die Schüler die Klasse 2 vor, da sich das rote X im Bereich der Datenpunkte dieser Klasse befindet. Außerdem schlugen sie folgendes allgemeines Verfahren zur Klassifizierung vor: „Messen, welcher Datenpunkt am nächsten zum unbekannten Datenpunkt liegt, und anhand dessen die Klasse des unbekannten Datenpunkts bestimmen.“ Es lässt sich also festhalten, dass die Grundidee des kNN-Algorithmus im Rahmen der zweiten Plenumsdiskussion gemeinsam mit den Schülern erarbeitet werden konnte.

- **Arbeitsblatt 3 & Plenumsdiskussion 3:**
  Auch bei der Bearbeitung des dritten Arbeitsblattes waren keine Verständnisschwierigkeiten zu beobachten. Lediglich beim Codefeld zu Aufgabe drei kam es zu Problemen, was jedoch an einem Fehler im Hintergrundcode lag. Während eine Gruppe zusätzlich zum vorgefertigten Codegerüst einen eigenen Code zur Suche der k nächsten Nachbarn schrieb, beschäftigte sich eine andere Gruppe, die die Aufgaben ebenfalls besonders schnell bearbeitet hatte, mit der Zusatzaufgabe am Ende des dritten Arbeitsblattes. Im Anschluss zeigte sich anhand der

richtigen Antworten auf die Fragen in der dritten Plenumsdiskussion, dass die Schüler die Inhalte dieses Arbeitsblattes verstanden hatten.

- **Arbeitsblatt 4 & Plenumsdiskussion 4:**
  Die Bearbeitung des vierten Arbeitsblattes verlief problemlos. Lediglich bei der Berechnung der Präzision hatten fast alle Gruppen zunächst einen Denkfehler: Sie vertauschten die Bedeutung der Zeilen und Spalten in der Wahrheitsmatrix. Zwei Gruppen bearbeiteten die Aufgaben besonders schnell und beschäftigten sich zusätzlich mit dem Zusatzblatt. Auch hierbei hatten die Schüler keine Verständnisschwierigkeiten. In der anschließenden Plenumsdiskussion konnten die Teilnehmenden die Fragen zu den einzelnen Aufgaben korrekt beantworten. Bei der Diskussion zur Verbesserung des Verfahrens hatten die Schüler viele gute Ideen. Ein Schüler schlug vor, weitere statistische Kenngrößen zu berücksichtigen. Ein anderer machte den Vorschlag, die Nachbarn nach ihrer „Nähe“ zu gewichten. Darüber hinaus hatten weitere Schüler Ideen, wie die Beschleunigungsachsen unterschiedlich zu gewichten, den Wert von k zu verändern, den Datensatz zu vergrößern oder zu Beginn Daten von weiteren Sensoren aufzunehmen. Diese Beiträge lassen vermuten, dass die Schüler die Inhalte des Lernmoduls bisher sehr gut verstanden hatten.

- **Arbeitsblatt 5 & Plenumsdiskussion 5:**
  Nach der Mittagspause verlief die Bearbeitung des fünften Arbeitsblattes ohne große Schwierigkeiten. Die meisten Schüler kannten die statistischen Kenngrößen Median sowie oberes und unteres Quartil nicht, da in den Klassenstufe sieben und acht keine Boxplots unterrichtet wurden. Dennoch konnten alle Schüler anhand der Hilfekarten die Aufgaben bearbeiten. Ein Schüler stellte die Nachfrage, ob bei der Klassifikation nun alle 15 Features verwendet werden und die Datenpunkte der Windows in einem 15-dimensionalen Raum liegen. Dies wurde in der anschließenden Plenumsdiskussion kurz aufgegriffen. Die einzelnen Fragen im Rahmen der Ergebnissicherung konnten die Schüler richtig beantworten.

- **Arbeitsblatt 6 & Plenumsdiskussion 6:**
  Bei der Bearbeitung des sechsten Arbeitsblattes hatte nur eine Schülerin Probleme beim Schreiben des Codes zur Bestimmung des optimalen Werts für k. Diese Schülerin hatte zuvor keinerlei Erfahrungen im Informatikbereich gesammelt, konnte aber anhand der Hilfekarten und einer kurzen Erklärung seitens der Dozenten die Aufgabe lösen. Bei der anschließenden Diskussion konnten die Schüler den Graphen zur Bestimmung des optimalen Werts von k (s. Abb. 20) richtig interpretieren und erklären, warum für kleine Werte von k große Schwankungen entstehen. Auch das Verfahren zur rechnerischen Bestimmung des optimalen Werts von k konnten die Teilnehmenden problemlos erläutern. Insgesamt lässt sich also festhalten, dass die Schüler das Optimierungsproblem trotz ihrer geringen Vorkenntnisse in diesem Bereich anhand des Arbeitsblattes eigenständig lösen konnten.

- **Arbeitsblatt 7 & Plenumsdiskussion 7:**
  Auch bei der Bearbeitung des siebten Arbeitsblattes zeigten die Schüler keine Verständnisschwierigkeiten. Auffällig war allerdings, dass die Schüler bei der Bearbeitung dieses Arbeitsblattes deutlich unruhiger waren als bei den Vorhergehenden. Das könnte einerseits daran gelegen haben, dass sich die Schüler auf diesem Arbeitsblatt viel mit ihrem Sitznachbarn austauschen und über verschiedene Fragestellungen diskutieren sollten. Andererseits könnte es auch damit einhergegangen sein, dass es bereits Nachmittag war und die Konzentration der Schüler nachgelassen hatte. Trotz allem wurde in der anschließenden Plenumsdiskussion ausführlich über Probleme und Schwierigkeiten sowie Anwendungen und Einsatzgebiete der Aktivitätserkennung diskutiert. Die Schüler brachten viele gute Einwände und waren zudem in der Lage die Aktivitätserkennung kritisch zu reflektieren. Ein Schüler nannte in diesem Zusammenhang Krankenversicherungen, die die Tarif erhöhen, wenn Personen wenige körperliche Aktivitäten ausführen. Andere Schüler warfen die Themen Überwachung und Manipulation von Daten ein.

- **Abschlusspräsentation:**
  Bei der abschließenden Zusammenfassung der einzelnen Schritte des Lernmoduls anhand des Zusammenfassungsarbeitsblattes brachten sich die Schüler rege ein und konnten den Lückentext ohne Probleme ausfüllen. Zum Abschluss lässt sich anhand der Beobachtungen festhalten, dass die Schüler die Inhalte des Lernmoduls verstanden hatten und am Ende erklären konnten, wie die Aktivitätserkennung auf dem Smartphone funktioniert.

Im Folgenden werden außerdem die Beobachtungen und die Ergebnisse der Evaluation hinsichtlich der einzelnen Gesichtspunkte aus Abschnitt 5.2 vorgestellt. Die detaillierten Ergebnisse der Evaluation sind in Anhang G.2 zu finden.

Gestaltung und Ablauf des Lernmoduls

Die im Vorfeld angesetzten Bearbeitungszeiten erwiesen sich während der ersten Durchführung als realistisch. Alle Gruppen konnten innerhalb der geplanten Zeit die Arbeitsblätter bearbeiten. Insgesamt war der Modellierungstag 40 Minuten früher beendet als ursprünglich geplant. Dies lässt sich zum einen auf die besonders motivierte und interessierte Schülergruppe zurückführen. Zum anderen wurden die beiden zehnminütigen Pausen zwischen der ersten und zweiten sowie der dritten und vierten Arbeitsphase übersprungen. Gerade bei den ersten beiden Arbeitsblättern bearbeiteten zwei Gruppen die Aufgaben deutlich schneller als die anderen Gruppen, sodass für diese Schüler etwas Leerlauf bis zur nächsten Plenumsdiskussion entstanden ist. Dieser Unterschied zeigte sich auch in der Bewertung der Lern- und Arbeitszeiten im Evaluationsbogen. Sechs Schüler stimmten der Aussage „Die Lern- und Arbeitszeiten waren angemessen.“ völlig zu, während drei Schüler dem eher nicht zustimmten. Ein Grund für den großen Unterschied in den Bearbeitungszeiten könnten die sehr unterschiedlichen Altersstufen

der am Modellierungstag teilnehmenden Schüler gewesen sein.

Die Gestaltung der Arbeitsblätter empfanden alle Schüler als ansprechend und übersichtlich. Fünf Schüler stimmten der Aussage völlig und vier eher zu. Die Aussage „Die Aufgabenstellungen waren abwechslungsreich.“ bewerteten sieben Schüler mit „Stimmt eher“ und zwei Schüler mit „Stimmt eher nicht“. Einige Schüler hätten sich mehr Aufgaben, bei denen sie eigenständig etwas programmieren können, gewünscht. Dies zeigte sich auch durch die Antwort „Man musste eigentlich nichts selber programmieren.“ auf die Frage, was den Schülern am Lernmodul nicht gefallen hat. Dies hängt vermutlich damit zusammen, dass sich viele der teilnehmenden Schüler sehr für Mathematik und Informatik interessieren und bereits im Vorfeld selbst programmiert haben.

Die Plenumsdiskussionen nach den einzelnen Arbeitsblättern erschienen den Schülern im Großen und Ganzen als sinnvoll. Außerdem waren sich die meisten Schüler einig, dass die Diskussionsphasen eher nicht ausführlicher sein sollten. Lediglich eine Person, hätte sich die Diskussionen umfangreicher gewünscht.

Während des Modellierungstages hatten sechs Schüler das Gefühl, dass sie neue Inhalte selbständig erarbeiten konnten. Dies zeigte sich auch in der Antwort „Arbeitsblätter waren ideal, um sich die Themen selbstständig zu erarbeiten.“ auf die Frage, was den Schülern am Lernmodul besonders gut gefallen hat. Drei Schüler stimmten der Aussage, dass sie sich neue Inhalte selbständig erarbeiten konnten, eher nicht zu. Während den Diskussionsphasen hatten die meisten Schüler das Gefühl, dass sie ihre eigenen Ideen einbringen konnten. Zwei Schüler stimmten dem eher nicht zu.

Schwierigkeitsgrad des Lernmoduls

Während der Durchführung und in den Ergebnissen der Evaluation zeigte sich, dass der Schwierigkeitsgrad für die Schülergruppe mit sehr unterschiedlichem Vorwissen angemessen war. Die Schüler hatten während der Bearbeitung der Aufgaben so gut wie keine Verständnisschwierigkeiten. Auch in der Evaluation bewertete kein Teilnehmer die Aussagen, dass die Aufgaben zu einfach oder zu schwer waren, mit „stimmt völlig.“ Zudem gaben alle Schüler an, dass sie die Aufgaben immer oder zumindest in den meisten Fällen eigenständig bearbeiten konnten. Dies wurde auch während der Durchführung beobachtet. Auf die Frage, was die Schüler im Lernmodul bzw. an den Aufgaben (besonders) schwierig fanden, wurde einerseits geantwortet, dass eigentlich nichts sonderlich schwer war. Andererseits fand ein Teilnehmer die Aufgaben schwierig, bei denen ihm das Grundwissen gefehlt hat. Für einen anderen Schüler waren die Aufgaben, bei denen programmiert werden musste, schwierig, da er keine Programmiervorkenntnisse hatte.

Die Aussage „Mein bisheriges mathematisches Vorwissen hat ausgereicht, um die Aufgaben bearbeiten zu können“ bewerteten vier Schüler mit „Stimmt völlig“, vier Schüler

mit „Stimmt eher“ und ein Schüler mit „Stimmt eher nicht“. Dies hat sich zum Teil auch während der Durchführung gezeigt, denn viele Schüler kannten den Begriff statistische Kenngröße nicht bzw. nicht mehr. Auch die Größen Median sowie das obere und untere Quartil waren nicht allen Schülern bekannt. Das könnte daran liegen, dass diese Inhalte bei vielen Schülern zu weit zurückliegen oder aufgrund der Corona-Pandemie in den letzten beiden Jahren nicht unterrichtet wurden. Die fehlenden Kenntnisse konnten durch die Hilfekarten und Infoblättern ausgeglichen werden, sodass dennoch alle Schüler das Lernmodul eigenständig bearbeiten konnten.

Insgesamt ist festzuhalten, dass der Schwierigkeitsgrad des Lernmoduls für diese Schülergruppe angemessen war. Die aufgezählten Schwierigkeiten sind meistens bei den beiden Teilnehmenden aus der neunten und zehnten Klasse aufgetreten, sodass diese Probleme bei einer Durchführung mit einer Schulklasse aus der Sekundarstufe II nicht zu erwarten sind.

Verständlichkeit und Hilfestellungen

Aus Sicht der Schüler wurden die Inhalte in den Präsentationen gut erläutert. Sechs Schüler stimmten dieser Aussage im Evaluationsbogen völlig und drei eher zu. Auch die Aufgabenstellungen fanden die meisten Schüler verständlich formuliert. Acht Schüler bewerteten diese Aussage mit „Stimmt völlig“ oder „Stimmt eher“. Lediglich ein Teilnehmer stimmte der Aussage eher nicht zu. Das deckt sich auch mit den Beobachtungen während der Durchführung. Die Schüler arbeiteten eigenständig und hatten nur selten Rückfragen zu den Aufgabenstellungen.

Während den Plenumsdiskussionen erläuterten die Schüler viele Zusammenhänge anhand verschiedener Abbildungen, sodass davon auszugehen ist, dass diese für das Verständnis der Inhalte hilfreich waren. Dies stimmt mit den Ergebnissen der Evaluation überein. Die Aussage „Die Grafiken und Abbildungen haben beim Verständnis der Inhalte geholfen.“ bewerteten sechs Schüler mit „Stimmt völlig“ und drei Schüler mit „Stimmt eher“.

Aus Sicht der Dozenten nutzten die Teilnehmer, vor allem der Sekundarstufe II, nur selten die Hilfekarten. Das deckt sich mit der Bewertung in der Evaluation. Hier gab nur ein Schüler an, dass er häufig die Hilfekarten bei der Bearbeitung der Aufgaben benutzen musste. Die restlichen Schüler stimmten dem eher nicht bzw. gar nicht zu. Lediglich sechs Teilnehmer gaben eine Bewertung zur Aussage „Die Tipps/Hilfekarten waren hilfreich.“ ab. Davon bewerteten fünf die Aussage mit „Stimmt völlig“ und „Stimmt eher“. Ein Schüler stimmte der Aussage eher nicht zu. Drei Schüler beurteilten diese Aussage nicht, was damit zu erklären sein könnte, dass diese Schüler die Hilfekarten zur Bearbeitung der Aufgaben nicht benötigt und daher auch nicht angeschaut haben.

Motivation und Interesse

Aus Dozentensicht zeigten die Schüler während der Durchführung großes Interesse am Thema des Lernmoduls und den zugrundeliegenden mathematischen Inhalten. Dies war sowohl in den regen Diskussionsrunden während den Sicherungsphasen als auch in den Ergebnissen der Evaluation erkennbar. Alle Teilnehmenden stimmten der Aussage „Das Thema des Workshops fand ich interessant.“ völlig zu. Auch die Lern- und Arbeitsatmosphäre empfanden alle Schüler als angenehm. Sechs Schüler stimmten der Aussage völlig und drei eher zu.

In der Evaluation gaben die Schüler an, dass ihr Interesse an Themen der angewandten Mathematik sowie der Naturwissenschaften und der Technik durch den Modellierungstag gesteigert werden konnte. Darüber hinaus bestätigten die meisten Schüler, dass sie im Lernmodul etwas Neues gelernt haben, dass ihnen in der Schule, in einem Studium oder im Beruf weiterhelfen kann. Zudem können sich die meisten Schüler vorstellen, ein Studium oder eine Ausbildung im Bereich der angewandten Mathematik, der Naturwissenschaften oder der Technik zu beginnen. Lediglich eine Person stimmt dem eher nicht zu. Die Aussage „Durch den Workshop habe ich interessante Berufs- und Studienmöglichkeiten kennengelernt“ bewerteten die Schüler sehr unterschiedlich. Dennoch stimmte der Großteil der Aussage eher zu.

Alle Schüler bewerteten die beiden Aussagen „Ich würde so einen Workshop (zu einem anderen Thema) gerne noch einmal besuchen“ und „Ich würde diesen Workshop anderen Weiterempfehlen“ mit „Stimmt völlig“ und „Stimmt eher“. Dies verdeutlicht, dass das Lernmodul im Großen und Ganzen das Interesse der Schüler geweckt und ihnen die Arbeit mit den Lernmaterialien Spaß gemacht hat.

Umgang mit Jupyter Notebooks und Python

Während der Durchführung hat sich gezeigt, dass die Schüler kaum Probleme im Umgang mit den Jupyter Notebooks und Python hatten. Dies zeigte sich auch in der Evaluation. Sieben Schüler gaben an, dass ihnen der Umgang mit der Programmiersprache Python nicht oder nur selten schwer gefallen ist. Lediglich zwei Schüler hatten bei der Arbeit mit Python kleinere Schwierigkeiten. Die Einführung zu Beginn in Jupyter Notebooks und Python fand die Mehrheit der Teilnehmer hilfreich, der Rest beantwortete diese Frage nicht. Eine Person hätte sich zu Beginn eine etwas ausführlichere Einführung gewünscht.

Der Aussage „Die automatische Rückmeldung, die ich nach der Eingabe meiner Lösung im Code erhalten habe, fand ich hilfreich.“ bewerteten sieben Teilnehmer mit „Stimmt völlig“ und zwei mit „Stimmt eher“. Die Überprüfefunktion scheint die Schüler also bei der Bearbeitung der Aufgaben sinnvoll unterstützt zu haben. Auch die Arbeit mit den Codefeldern hat den meisten Schülern Spaß gemacht. Nur eine Person stimmte dem

eher nicht zu.

Vor dem Modellierungstag hatten bereits fünf Schüler mit Python und Jupyter Notebooks gearbeitet. Jeweils ein Teilnehmer bewertete die Aussage mit „Stimmt eher“ und „Stimmt eher nicht“. Zwei Schüler hatten zuvor noch nie Jupyter Notebooks und Python genutzt.

Insgesamt lässt sich festhalten, dass den Schülern die Arbeit mit den interaktiven Arbeitsblättern Spaß gemacht hat und sie kaum Probleme im Umgang mit den digitalen Werkzeugen hatten.

Lernzuwachs in den Bereichen mathematische Modellierung und KI

Weitere Hauptziele des Lernmoduls sind die Förderung der Modellierungskompetenz und der Aufbau eines grundlegendes Verständnisses zu den Themen KI und maschinelles Lernen.

In der Evaluation stimmten drei Teilnehmer völlig und sechs eher zu, dass sie durch das Lernmodul die mathematische Modellierung besser begriffen haben. Auch den Vortrag über die mathematische Modellierung fanden alle Schüler zumindest teilweise hilfreich. Anhand der Antworten auf die Frage, was die Teilnehmer unter mathematischer Modellierung verstehen, lässt sich ebenfalls erkennen, dass bei den Schülern am Ende des Modellierungstages ein subjektives Verständnis der mathematischen Modellierung vorhanden war. So schreibt ein Teilnehmer, dass er unter mathematischer Modellierung das Reduzieren und Vereinfachen von Alltagsproblemen, sodass die Probleme auf mathematischer Ebene beschrieben, gelöst und optimiert werden können, versteht. Ein anderer Schüler gab die Antwort: „Mit der Mathematischen Modellierung wird ein reelles Problem so vereinfacht, dass sich daraus ein mathematisches Problem ergibt, welches anschließend gelöst wird. Diese Lösung wird für die Anwendung beim echten Problem interpretiert.“

Im Bereich KI bewerteten acht Schüler die Aussage „Durch den Workshop habe ich die mathematischen Hintergründe von KI-Systemen besser begriffen.“ mit „Stimmt völlig“ oder „Stimmt eher“. Unter künstlicher Intelligenz verstand ein Schüler am Ende des Modellierungstags ein „Programm was man nicht 'fertig programmieren' muss sondern was sich eigenständig weiterentwickelt und dazulernt.“ Andere Schüler schrieben, dass künstliche Intelligenz Computerprogramme sind, die sich durch den Erhalt von Daten und Erfahrungen stetig verbessern und lernen. Anhand dieser Aussagen wird deutlich, dass die Schüler am Ende des Modellierungstages zwar eine Vorstellung von KI hatten, diese aber oft mit maschinellem Lernen gleichsetzten. Das Prinzip des überwachten maschinellen Lernens, das zur Bewertung der Güte des Klassifikationsalgorithmus eingesetzt wurde, konnten nur zwei Schüler in der Evaluation richtig erläutern. So schrieb einer der beiden: „Ein Datensatz wird in zwei Gruppen geteilt, die Testdaten und

die Trainingsdaten. Die Testdaten werden anhand der Trainingsdaten klassifiziert und anhand dieser Klassifizierung wird überprüft wie qualitativ das Programm ist.“ Die anderen Schüler erklärten anstelle des Prinzips des überwachten maschinellen Lernens das Vorgehen im Lernmodul oder generell maschinelles Lernen. Die „falschen“ Antworten könnten also auch darauf zurückzuführen sein, dass die Lernenden die Frage im Evaluationsbogen nicht richtig verstanden hatten.

Lob, Kritik und Verbesserungsvorschläge

Die Frage, ob den Schülern etwas am Lernmodul absolut nicht gefallen hat, beantwortete nur ein Teilnehmer mit „Ja“. Er hätte sich gewünscht, mehr selbst programmieren zu können.

Besonders gefallen hat einem Teilnehmer, dass er die Möglichkeit hatte, sich alles selbst zu erarbeiten, aber es trotzdem nochmal erklärt wurde. Anderen hat die Datenverarbeitung, das gute Erklären und das Kennenlernen neuer mathematischer Größen wie das Quartil besonders zugesagt.

Auf die Frage „Hättest du gerne noch etwas anderes gesehen oder erfahren?“ gaben fünf Schüler eine Antwort. Ein Teilnehmer hätte sich spannendere Aktivitäten wie Fahrrad Fahren, Schwimmen, Klettern und Surfen gewünscht. Andere Schüler hätten gerne mehr Input zur Erstellung eines guten Datensatzes, zum unbeaufsichtigten Lernen und zur Optimierung der Rechenauslastung erhalten. Darüber hinaus hätte sich ein Teilnehmer gerne noch tiefer mit dem Thema Aktivitätserkennung beschäftigt und dafür gerne einen längeren Modellierungstag besucht.

Fazit der ersten Durchführung

Insgesamt bewerteten die Schüler den Modellierungstag mit der Note 1,9 und die Betreuer mit der Note 1,1. Außerdem gaben die Schüler an, dass sie das Lernmodul „super“, „spannend“ und „lehrreich“ fanden, die Aufgabe aber gerne noch etwas geöffnet werden könnten. Zusammenfassend war die erste Durchführung sehr zufriedenstellend.

5.4. Vorgenommene Verbesserungen des Lernmoduls nach der ersten Durchführung

Basierend auf den Beobachtungen der ersten Durchführung und den Ergebnissen der Evaluation wurden einige Verbesserungen und Veränderungen am Lernmodul vorgenommen. Diese werden im Folgenden vorgestellt.

Zunächst wurde in der Einstiegspräsentation eine weitere Folie zum maschinellen Lernen ergänzt, auf der anhand des Beispiels eines Spam-Filters das Prinzip des maschinellen Lernens ausführlicher erläutert wird. Um den möglicherweise entstehenden Leerlauf

zwischen der Bearbeitung des ersten Arbeitsblattes und der darauffolgenden Plenumsdiskussion zu verringern, wurde am Ende des ersten Arbeitsblattes eine Zusatzaufgabe ergänzt. In dieser recherchieren die Schüler weitere Sensoren, die in Smartphones eingebaut sind, und überlegen, welche Informationen diese Sensoren für die Aktivitätserkennung liefern könnten. Zudem wurde auf Arbeitsblatt fünf eine kurze Erläuterung ergänzt, in der thematisiert wird, dass bei der Klassifikation nun alle 15 Features berücksichtigt werden und die Windows Datenpunkten in einem 15-dimensionalen Raum entsprechen. Darüber hinaus wurden die Antwortblätter auf die Länge der Schülerantworten angepasst sowie kleinere Fehler in Verlinkungen und Codefeldern verbessert.

5.5. Beobachtungen der zweiten Durchführung

In diesem Abschnitt werden die Beobachtungen der zweiten Durchführung des Lernmoduls vorgestellt.

Die Inhalte des Lernmoduls wurden für diese Durchführung gekürzt und zum Teil in Plenumsdiskussionen ausgelagert, da nur zwei Doppelstunden zur Verfügung standen. Die Arbeitsblätter eins und zwei wurden gemeinsam mit den Schülern im Plenum besprochen. Dazu ging der Dozent die Arbeitsblätter mit Hilfe eines Beamers schrittweise durch. Anschließend sollten die Schüler die Kurzversion von Arbeitsblatt drei sowie das vierte Arbeitsblatt eigenständig bearbeiten. Im Anschluss wurden die Inhalte der Arbeitsblätter fünf, sechs und sieben mit Hilfe der Präsentationsfolien vorgestellt sowie die Schwierigkeiten und Anwendungen der Aktivitätserkennung im Plenum diskutiert.

Da das Lernmodul im Rahmen dieser beiden Doppelstunden nicht im eigentlich geplanten Umfang durchgeführt werden konnte, werden im Folgenden nur kurz die wichtigsten Punkte der Beobachtungen erläutert und nicht näher auf die Evaluation eingegangen.

Insgesamt lässt sich festhalten, dass die meisten Schüler gerade in der ersten Doppelstunde große Schwierigkeiten hatten. Die Unterrichtseinheit wurde zu Beginn der zehnten Klasse durchgeführt, weshalb die Schüler noch keine Vektoren kannten. Aufgrund der geringen Zeit konnten diese auch nicht ausführlich eingeführt werden, sodass die Darstellung der Windows als Datenpunkte in einem dreidimensionalen Koordinatensystem nur für einzelne Schüler verständlich war. Dadurch war es für viele Schüler schwer, sich die Abstandsberechnung zwischen zwei Windows räumlich vorzustellen. Zudem war auffällig, dass den Schülern Begriffe wie Code, Algorithmus und statistische Kenngröße nicht bekannt waren, weshalb die Bearbeitung der Arbeitsblätter erschwert wurde. Die Schüler erkannten die Codefeldern nicht als solche und führten diese daher auch nicht aus. Darüber hinaus lasen nur wenige Schüler die Aufgabenstellungen und bearbeiteten die Aufgaben. Viele Schüler waren mit anderen Dingen beschäftigt und interessierten sich wenig für die Inhalte des Lernmoduls. Dies zeigte sich auch in den Plenumsdiskussionen, an denen sich nur einzelne Schüler beteiligten.

In der zweiten Doppelstunde arbeiteten deutlich mehr Schüler an den Arbeitsblättern, sodass auch die Diskussionen im Anschluss durch mehr Beiträge bereichert wurden. Auch die Inhalte verstanden die Schüler nun besser. Dennoch waren auch hier vereinzelt Schüler dabei, die sich nicht an der Bearbeitung der Aufgaben in ihrer Gruppe und den anschließenden Plenumsdiskussionen beteiligten.

Zusammenfassend lässt sich nach dieser zweiten Durchführung die Erkenntnis festhalten, dass das Lernmodul mit einer gesamten Klasse erst nach der Einführung von Vektoren durchgeführt werden sollte. Anhand einzelner, leistungsstarker Schüler lies sich aber auch feststellen, dass das Lernmodul im Rahmen von Vertiefungsmöglichkeiten für gute Schüler bereits vor der Einführung von Vektoren eingesetzt werden kann. Darüber hinaus sollte vor der Durchführung mit den Lehrkräften abgeklärt werden, inwieweit statistische Kenngrößen bekannt sind, um diese bei Bedarf vor den jeweiligen Arbeitsblättern in den Plenumsdiskussionen einzuführen. Außerdem hat die zweite Durchführung gezeigt, dass sich das Lernmodul für Kurse der Sekundarstufe II problemlos auf zwei Doppelstunden kürzen lässt und die Schüler trotzdem die wesentlichen Inhalte des Lernmoduls erarbeiten sowie einen Einblick in die mathematische Modellierung erhalten können.

5.6. Fazit

Durch die beiden Durchführungen wurde das entwickelte Lernmodul einerseits erprobt und andererseits verbessert. Besonders die erste Durchführung im Rahmen des offenen CAMMP days verlief zufriedenstellend. Leider konnte das Lernmodul nicht mit einem kompletten, heterogenen Kurs der Sekundarstufe II getestet werden. Bei der ersten Durchführung nahmen hauptsächlich mathematikinteressierte Schüler teil. Zwar waren die Teilnehmer durchaus in sehr unterschiedlichen Altersstufen, im Hinblick auf das Interesse und die Motivation der Schüler ist jedoch nicht von einer heterogenen Lerngruppe auszugehen. Bei der zweiten Durchführung mit einer zehnten Klasse konnte das Lernmodul nur in einem gekürzten Rahmen realisiert werden. Für zukünftige Erprobungen wäre es somit interessant, das Lernmodul im vollem Umfang und mit ganzen Schulklassen, also mit Lerngruppen mit unterschiedlichem Leistungsniveau und Interesse zu erproben. Dennoch lässt sich festhalten, dass das Ziel dieser Abschlussarbeit, die Erstellung von Lehr- und Lernmaterial zur Aktivitätserkennung auf dem Smartphone, erfüllt wurde.

6. Ausblick

Im letzten Kapitel dieser Abschlussarbeit werden Möglichkeiten zur Weiterentwicklung und Verbesserung des im Rahmen dieser Arbeit entwickelten Lernmoduls vorgestellt.

In dem erstellten Lernmodul arbeiten die Schüler mit Sensordaten des Accelerometers. Zur weiteren Vertiefung könnten zusätzlich Sensordaten des Gyroskops und des Magnetometers berücksichtigt werden. Dadurch wird eine Umrechnung der aufgenommenen Beschleunigungswerte in Erdkoordinaten möglich. Bei der Klassifikation der Daten sollte dann die Position und Ausrichtung des Smartphones weniger stark ins Gewicht fallen, sodass auch die Daten, bei denen das Handy in der Hand gehalten wurde bzw. sich im Rucksack des Nutzers befand, richtig klassifiziert werden können. Die passenden Messwerte des Gyroskops und Magnetometers zu den im Lernmodul verwendeten Beschleunigungswerten des Accelerometers wurden bei der Datenaufnahme direkt mit aufgezeichnet und abgespeichert, sodass für diese Erweiterung des Lernmoduls kein neuer Datensatz erstellt werden muss. Eine weitere Möglichkeit das Lernmodul im Bereich der verwendeten Sensordaten auszuweiten, besteht darin, Daten der linearen Beschleunigung, also Beschleunigungswerte bei denen die Erdbeschleunigung eliminiert wurde, zu klassifizieren. Auch diese Daten wurden bereits mit aufgenommen und abgespeichert. Darüber hinaus könnten zusätzlich Standortdaten (z. B. GPS) berücksichtigt werden. Hierzu müssten jedoch neue Sensordaten aufgenommen werden.

Bisher klassifizieren die Schüler im Lernmodul Daten, die bereits im Vorfeld aufgezeichnet wurden. Um das Interesse und die Motivation der Schüler zu erhöhen, könnten die Schüler im Rahmen des Lernmoduls auch eigene Sensordaten aufnehmen. Hierzu wird schließlich nur ein Smartphone und die kostenfreie App phyphox benötigt. Anschließend könnten die Schüler anhand der eigenständig aufgenommenen Beschleunigungswerte überprüfen, wie gut der entwickelte Klassifikationsalgorithmus auf neuen, unbekannten Daten funktioniert.

Im Moment liegt der Fokus des Lernmoduls auf dem kNN-Algorithmus. Um den Schwerpunkt mehr auf das maschinelle Lernen im Allgemeinen zu legen, wäre es zusätzlich möglich, die Sensordaten mit verschiedenen maschinellen Lernverfahren zu klassifizieren. Anschließend könnten die Vor- und Nachteile der unterschiedlichen Verfahren diskutiert und ein Algorithmus bestimmt werden, mit dem die Sensordaten am besten den verschiedenen Aktivitäten zugeordnet werden können.

Bisher werden im Lernmodul nur zeitabhängige Features bei der Datenvorverarbeitung eingesetzt. Die verwendeten Features könnten zusätzlich um frequenzabhängige Features ergänzt werden. Zur Berechnung der frequenzabhängigen Features wird jedoch eine Fourieranalyse benötigt. In diesem Zusammenhang wäre eine Anlehnung an den bereits bestehenden Workshop zum Thema Fitnesstracker denkbar. In diesem Workshop werden Aktivitätsdaten mit Hilfe der Fourieranalyse ausgewertet, um die Schritte

zu zählen sowie die Aktivitäten Gehen, Laufen, Sprinten und Stehen zu klassifizieren (vgl. Marnitz, 2017).

Darüber hinaus wäre eine Erprobung des Lernmoduls im Rahmen einer Unterrichtseinheit im Schulfach IMP sowie eine stärkere Fokussierung auf den Bereich Data Science denkbar.

Anhang

A. Präsentationen

A.1. Einstiegspräsentation

Wie können Smartphones die Aktivitäten ihrer Nutzer erkennen... und was hat das mit Mathe zu tun?

Einführung in die Problemstellung

Aktivitätserkennung auf dem Smartphone

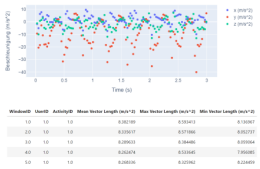
- Smartphones sind ständige Begleiter in unserem Alltag
- Weltweit nutzen ca. 3,9 Milliarden Menschen ein Smartphone
- Schnelle Entwicklung der in den Smartphones eingebauten Sensoren ist Grundlage zahlreicher Anwendungsgebiete

→ menschliche Aktivitätserkennung

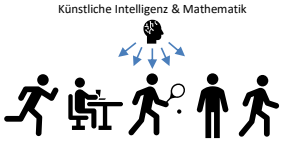
- Aber wie funktioniert das und was hat das mit Mathe zu tun?

Prozess der menschlichen Aktivitätserkennung

Sensordaten
↓
Vorverarbeitung der Daten
↓
Klassifizierung



Künstliche Intelligenz & Mathematik



Exkurs | Was ist eine künstliche Intelligenz?

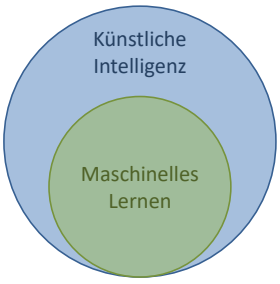
- Was versteht ihr unter dem Begriff künstliche Intelligenz?
- Welche Anwendungen der künstlichen Intelligenz kennt ihr?

Exkurs | Was ist eine künstliche Intelligenz?

Turing-Test von Alan Turing:

Exkurs | Was ist eine künstliche Intelligenz?

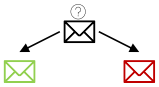
Maschinelles Lernen ist ein Teilbereich der künstlichen Intelligenz.



Exkurs | Maschinelles Lernen am Beispiel eines Spamfilters

- **Ziel:** Lösen einer speziellen Aufgabe (z.B. Spamfilter)
- **Ausgangspunkt:** große Datenmenge
- Entwicklung eines Modells anhand der Informationen in den Daten
- Umsetzung erfolgt mit Computern

- Lösung der Aufgabe mit Hilfe des entwickelten Modells



Wieso benötigen wir eine künstliche Intelligenz bei der Aktivitätserkennung?

Herkömmliches Computerprogramm

- Arbeitet nach festen Regeln
- Beispiel:
  Ordne den Sensordaten $a_x = -3.09$, $a_y = -9.58$ und $a_z = -2.66$ die Aktivität „Gehen“ zu

→ Eine solche Regel müsste für alle Sensordatenpunkte aufgestellt werden

Maschinelles Lernen

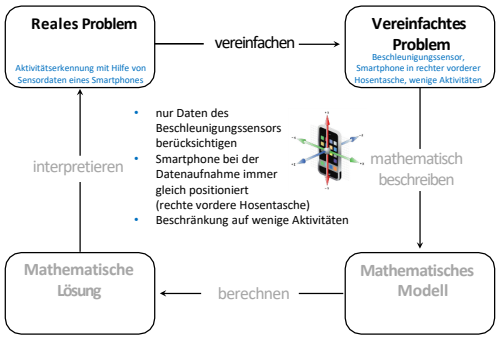
- Regeln werden aus Daten gelernt
- Beispiel:
  Erkennen von Mustern in den Sensordaten einer bestimmten Aktivität. Anhand dieser Muster (Regeln) erfolgt anschließend die Zuordnung der Sensordaten zu den einzelnen Aktivitäten.

Ziel und Ablauf des Workshops

Ziel: Entwicklung eines eigenes Klassifikationsalgorithmus!

1. Erkunden des Datensatzes
2. Vorverarbeitung der Daten
3. Schrittweise Entwicklung eines eigenen Klassifikationsalgorithmus
4. Bewertung der Ergebnisse des Klassifikationsalgorithmus mit Hilfe verschiedener Qualitätsmaße
5. Verbesserung des entwickelten Klassifikationsalgorithmus
6. Diskussion von Problemen sowie Anwendungsgebieten der menschlichen Aktivitätserkennung

Modellierungskreislauf



Jetzt seid ihr dran!

Keine Panik!

ORGANISIEREN!

- Bearbeitet die Arbeitsblätter!
- Lest die Aufgabenstellung sorgfältig!
- Teamwork!
- Helft euch gegenseitig!
- Nutzt die Tipps!
- Nutzt das Internet!
- Fragt die Betreuer!

Schritte zum Arbeitsmaterial

- Gehe auf https://workshops.cammp.online
- Auf „Zugriff auf Lernmaterial“ und „Registrieren!“ klicken
- Account erstellen: der Username muss Präfix **cammp_** enthalten (z. B. **cammp_laura1234**)
- Auf „Anmelden!“ klicken, Accountdaten eingeben und einloggen
- Erstes graues Feld der Willkommens-Datei ausführen ▶
- Warten … ⧗
- Im Dropdown Menü Workshop *Aktivitätserkennung* auswählen und das Feld ausführen ▶
- Links den Ordner 🖿 *Aktivitätserkennung* und dann den Ordner 🖿 *worksheets* öffnen
- Los geht’s mit dem ersten Arbeitsblatt!

A.2. Notizen Einstiegspräsentation

Notizen Einstiegspräsentation

Folie 1: Wie können Smartphones die Aktivitäten ihrer Nutzer erkennen… und was hat das mit Mathe zu tun?

Nachdem ihr mehr über die mathematische Modellierung erfahren habt, starten wir nun mit dem Workshop zum Thema Aktivitätserkennung auf dem Smartphone.

Folie 2: Aktivitätserkennung auf dem Smartphone

Mobile Geräte, wie zum Beispiel Smartphones, sind mittlerweile ständige Begleiter in unserem Alltag. Weltweit nutzen rund 3,9 Milliarden Menschen ein Smartphone. Zudem ist in den letzten Jahren das Interesse an der Auswertung der Gewohnheiten und täglichen Routinen der Menschen gestiegen. Dazu zählt unter anderem auch die Analyse der ausgeführten Aktivitäten. Die schnelle und enorme Weiterentwicklung der in den Smartphones eingebauten Sensoren bildet die Grundlage zahlreicher Anwendungsgebiete, wie zum Beispiel der Aktivitätserkennung. Doch wie funktioniert die Aktivitätserkennung und was hat das mit Mathe zu tun?

Dieser Frage werden wir im Laufe des Workshops nachgehen.

Folie 3: Prozess der menschlichen Aktivitätserkennung

Schauen wir uns den Prozess der menschlichen Aktivitätserkennung einmal genauer an.

Unser Ausgangspunkt sind die Sensordaten, die mit dem Handy aufgenommen werden.

klicken

Anschließend werden wir die Daten vorverarbeiten und die große Menge an Sensordaten auf eine kleinere Menge von möglichst aussagekräftigen Werten reduzieren. Wie das genau funktioniert, werdet ihr im Laufe des Workshops sehen.

klicken

Der letzte Schritt ist dann die Klassifizierung. Bei der Klassifizierung werden die Daten mit Hilfe einer künstlichen Intelligenz und Mathematik den unterschiedlichen Aktivitäten zugeordnet. Aber was ist überhaupt eine künstliche Intelligenz?

Folie 4: Exkurs | Was ist eine künstliche Intelligenz?

Fragen:
- Was versteht ihr unter dem Begriff künstliche Intelligenz?
- Welche Anwendungen der künstlichen Intelligenz kennt ihr?

Antwort:
- Antworten der Schüler:innen sammeln!

Folie 5: Exkurs | Was ist eine künstliche Intelligenz?

Der Begriff der künstlichen Intelligenz ist in den letzten Jahren immer populärer geworden. Die künstliche Intelligenz ist die Basis vieler Anwendungen aus unserem täglichen Leben. Das Thema künstliche Intelligenz stellt aber keinesfalls ein neues Forschungsgebiet dar.

Bereits im Jahr 1950 beschäftigte sich der britische Mathematiker Alan Turing mit der Frage, ob Maschinen bzw. Computersysteme intelligent sein können. Zur Untersuchung dieser Frage schlug er den folgenden Test

vor. Ein menschlicher Schiedsrichter kann mit zwei Partnern (ein Partner ist ein Mensch und der andere ein Computer) elektronisch kommunizieren und beliebige Fragen stellen. Wenn der Schiedsrichter nach vielen Fragen nicht anhand der Antworten entscheiden kann, welcher der Partner der Computer ist, so gilt der Computer als intelligent.

Der Begriff „künstliche Intelligenz“ wurde allerdings erst Jahre später von Prof. John Mc Carthy auf einer Konferenz zu diesem Themengebiet vorgeschlagen.

Neben den Chatbots sind seitdem noch zahlreiche weitere Anwendungen der künstlichen Intelligenz entstanden, wie zum Beispiel Empfehlungssysteme und Vorhersagemodelle.

Folie 6: Exkurs | Was ist eine künstliche Intelligenz?

Ein Teilbereich der künstlichen Intelligenz ist das maschinelle Lernen.

Folie 7: Exkurs | Maschinelles Lernen am Beispiel eines Spamfilters

Das Ziel beim maschinellen Lernen ist es, eine spezielle Aufgabe zu lösen (z.B. Spamfilter). Der Ausgangspunkt ist meistens eine große Menge an Beispieldaten, hier Mail-Beispiele. Zur Erfüllung der Aufgabe wird mit Hilfe der Beispieldaten ein Modell entwickelt. Diese Modellentwicklung wird mit Computern umgesetzt. Im Beispiel wird also ein Modell entwickelt, das Spam E-Mails (rot) von gewöhnlichen E-Mails (grün) unterscheiden kann.

klicken

Mit dem entwickelten Modell lässt sich anschließend die Aufgabe lösen. Das entwickelte Modell kann also entscheiden, ob es sich bei einer neuen E-Mail im Postfach um eine Spam E-Mail oder eine gewöhnliche E-Mail handelt. Wichtig dabei ist, dass die Regeln, die zur Lösung des Problems angewendet werden, nicht wie bei einem herkömmlichen Computerprogramm von einem Spezialisten fest vorgegeben werden, sondern von der künstlichen Intelligenz auf Basis des vorliegenden Datensatzes selbst entwickelt werden.

Folie 8: Wieso benötigen wir eine künstliche Intelligenz bei der Aktivitätserkennung?

Wieso benötigen wir jetzt für die Aktivitätserkennung eine künstliche Intelligenz?

Stellen wir uns einmal vor, wir wollen das Problem mit einem herkömmlichen Computerprogramm, das nach festen Regeln arbeitet, lösen. Dann müssten wir für alle möglichen Kombinationen von Sensordaten eine Regel aufstellen, welcher Aktivität diese zugeordnet werden sollen. Zum einen wäre das ziemlich aufwändig und zum anderen ist das nicht immer möglich, da manche Kombinationen von Sensordaten bei mehreren Aktivitäten auftreten können.

Verwenden wir jedoch eine künstliche Intelligenz bzw. Methoden aus dem Bereich des maschinellen Lernens, müssen wir die Regeln nicht mehr selbst definieren, sondern unser Computerprogramm lernt diese aus den Daten. In unserem Fall erkennt das Computerprogramm Muster in den Sensordaten der unterschiedlichen Aktivitäten und ordnet anschließend die Sensordaten anhand der Muster den einzelnen Aktivitäten zu. Ein Beispiel für ein Muster wäre zum Beispiel der Mittelwert der Sensordaten. Alle Daten, die ungefähr den gleichen Mittelwert haben, werden dann einer Klasse bzw. Aktivität zugeordnet.

Folie 9: Ziel und Ablauf des Workshops

Das Ziel des Workshops ist es, einen eigenen Klassifikationsalgorithmus zu entwickeln.

klicken

Dazu werdet ihr euch zu Beginn mit dem Datensatz vertraut machen.

klicken

Darauf folgt die Vorverarbeitung der Daten.

klicken

Im Anschluss werden wir schrittweise einen eigenen Klassifikationsalgorithmus entwickeln.

klicken

Nachdem wir den Algorithmus entwickelt haben, werden wir die Ergebnisse der Klassifikation anhand verschiedener Qualitätsmaße bewerten.

klicken

Anschließend werden wir versuchen unser Verfahren durch 2 Veränderungen zu verbessern.

klicken

Den Abschluss des Workshops bildet eine Diskussion von Problemen sowie Anwendungsgebieten der menschlichen Aktivitätserkennung.

Folie 10: Modellierungskreislauf

Zur Erstellung des Klassifikationsalgorithmus durchlaufen wir den Modellierungskreislauf. Unser reales Problem ist die Aktivitätserkennung mit Hilfe von Sensordaten eines Smartphones.

klicken

Dieses reale Problem vereinfachen wir, indem wir nur Sensordaten des Beschleunigungsmessers berücksichtigen. Dieser zeichnet die Beschleunigung in allen drei Raumdimensionen (x, y und z) in der Einheit $\frac{m}{s^2}$ auf. Außerdem wurde vor der Datenaufnahme die Vereinfachung getroffen, dass das Smartphone bei der Datenaufnahme immer gleich positioniert ist. Das Smartphone befand sich also während der Datenaufnahme immer in der rechten vorderen Hosentasche. Zudem werden wir uns auf einige wenige Aktivitäten beschränken.

Folie 11: Jetzt seid ihr dran!

Jetzt dürft ihr endlich aktiv werden. Öffnet Arbeitsblatt 1 und erkundet den Datensatz!

Folie 12: Schritte zum Arbeitsmaterial

Weg zum Arbeitsmaterial erklären.

A.3. Plenumsdiskussion 1

Aktivitätserkennung | Erkunden des Datensatzes

Diskussion nach Arbeitsblatt 1

Aufgabe 1 | Der Datensatz

- Aus welchen Daten besteht ein Datenpunkt im Datensatz?

UserID	ActivityID	Time (s)	x (m/s^2)	y (m/s^2)	z (m/s^2)
1	1	0.028140	5.480853	-4.508624	-6.869365
1	1	0.053292	5.488187	-4.504134	-6.870263
1	1	0.078443	5.494324	-4.496799	-6.872658
1	1	0.103595	5.492229	-4.508475	-6.871311
1	1	0.128747	5.472320	-4.499044	-6.856192
...	...	...	...	...	...

Aufgabe 1 | Die Aktivitäten

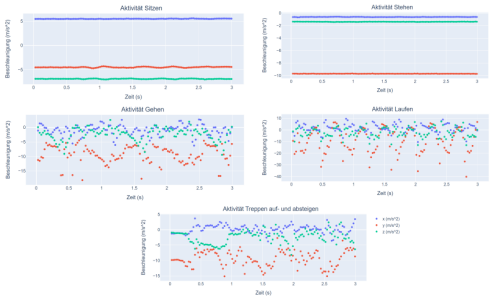
- Zu wie vielen Aktivitäten wurden Daten aufgenommen?
- Welche Aktivitäten sind dies?

ActivityID	Aktivität
1	Sitzen
2	Stehen
3	Gehen
4	Laufen / Joggen
5	Treppen auf- und absteigen

Aufgabe 1 | Die Aktivitäten

Graphische Darstellung der Sensordaten:

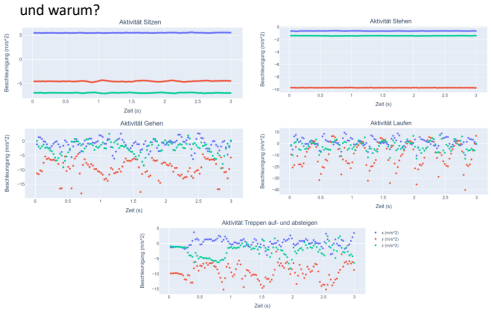
- Welche Unterschiede sind zwischen den einzelnen Aktivitäten zu erkennen?



Aufgabe 1 | Die Aktivitäten

Graphische Darstellung der Sensordaten:

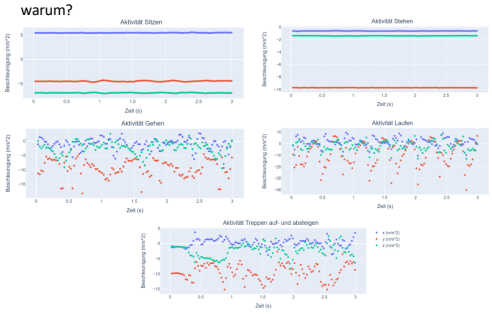
- Bei welcher Aktivität sind am wenigsten / am meisten Ausschläge zu erkennen und warum?



Aufgabe 1 | Die Aktivitäten

Graphische Darstellung der Sensordaten:

- Auf welcher der drei Achsen sind die größten Ausschläge zu erkennen und warum?



Aufgabe 2 | Die Testpersonen

- Wie viele Testpersonen haben Daten aufgenommen?

→ 10 Personen

- Wie viele Testpersonen sind weiblich und wie viele sind männlich?

→ 5 weiblich, 5 männlich

- Wie alt ist die jüngste bzw. älteste Testperson?

→ jüngste: 18 Jahre, älteste: 52 Jahre

UserID	Geschlecht	Alter
1	w	23
2	m	23
3	w	25
4	m	18
5	m	20
6	w	20
7	w	25
8	m	27
9	w	52
10	m	52

Aufgabe 3 | Die Datenaufnahme

- Wie lange wurden Daten zu den jeweiligen Aktivitäten von den Testpersonen aufgenommen?

→ 180 Sekunden = 3 Minuten

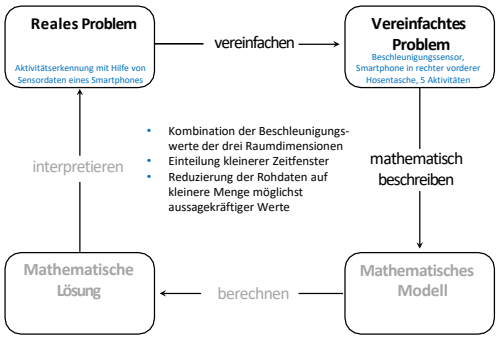
UserID	ActivityID	Time (s)	x (m/s^2)	y (m/s^2)	z (m/s^2)	UserID	ActivityID	Time (s)	x (m/s^2)	y (m/s^2)	z (m/s^2)
1	1	0.028140	5.480853	-4.508624	-6.869365	1	1	179.843615	5.377268	-4.423451	-6.996002
1	1	0.053292	5.488187	-4.504134	-6.870263	1	1	179.868768	5.409301	-4.418212	-6.991062
1	1	0.078443	5.494324	-4.496799	-6.872658	1	1	179.893921	5.422624	-4.475094	-7.008276
1	1	0.103595	5.492229	-4.508475	-6.871311	1	1	179.919075	5.384004	-4.435426	-6.995104
1	1	0.128747	5.472320	-4.499044	-6.856192	1	1	179.944228	5.397027	-4.384532	-6.978638

- Mit welcher Abtastrate wurden die Daten aufgenommen?

→ 7150 Messpunkte in 180 Sekunden

→ $\texttt{samplingRate} = \frac{7150}{180} \approx 40\,\text{Hz}$

Modellierungskreislauf



Jetzt seid ihr dran!

Öffnet **Arbeitsblatt 2** und beginnt mit der Vorverarbeitung der Daten!

- Bearbeitet die Arbeitsblätter!
- Lest die Aufgabenstellung sorgfältig!
- Teamwork!
- Helft euch gegenseitig!
- Nutzt die Tipps!
- Nutzt das Internet!
- Fragt die Betreuer!

A.4. Notizen Plenumsdiskussion 1

Notizen Zwischen- und Abschlusspräsentation

Diskussion nach Arbeitsblatt 1 | Erkunden des Datensatzes

Folie 1: Aktivitätserkennung | Erkunden des Datensatzes

Ihr habt auf dem ersten Arbeitsblatt den Datensatz, mit dem wir im Laufe des Workshops arbeiten werden, kennengelernt. Wir werden nun die Ergebnisse und Erkenntnisse dieses Arbeitsblattes gemeinsam im Plenum besprechen.

Folie 2: Aufgabe 1 | Der Datensatz

In der ersten Aufgabe habt ihr den Datensatz kennengelernt und konntet ihn nach unterschiedlichen Spalten sortieren.

Frage:
- Aus welchen Daten besteht ein Datenpunkt im Datensatz?

klicken

Antwort:
- Jeder Datenpunkt besteht aus der UserID und ActivityID sowie der Zeit und den Beschleunigungswerten in allen drei Raumdimensionen.

Folie 3: Aufgabe 1 | Die Aktivitäten

Neben dem Datensatz habt ihr in der ersten Aufgabe auch Informationen zu den unterschiedlichen Aktivitäten erhalten, zu denen Daten aufgenommen wurden.

Fragen:
- Zu wie vielen Aktivitäten wurden Daten aufgenommen?
- Welche Aktivitäten sind dies?

klicken

Antworten:
- Es wurden Daten zu 5 Aktivitäten aufgenommen.
- Die unterschiedlichen Aktivitäten sind Sitzen (1), Stehen (2), Gehen (3), Laufen / Joggen (4) und Treppen auf- und absteigen (5).

Folie 4: Aufgabe 1 | Die Aktivitäten

Außerdem habt ihr euch in Aufgabe 1 die Datenpunkte der einzelnen Aktivitäten auch graphisch angeschaut.

Frage:
- Welche Unterschiede sind zwischen den einzelnen Aktivitäten zu erkennen?

Antwort:
- Bei den Aktivitäten Sitzen und Stehen liegen die Datenpunkte der einzelnen Raumdimensionen auf einer Geraden.
- Bei den Aktivitäten Gehen, Laufen und Treppen auf- und absteigen liegen die Datenpunkte nicht auf einer Geraden, stattdessen sind deutliche Schwankungen zu erkennen.
- etc.

Folie 5: Aufgabe 1 | Die Aktivitäten

Frage:
- Bei welcher Aktivität sind am wenigsten / am meisten Ausschläge zu erkennen und warum?

Antwort:
- Die wenigsten Ausschläge sind bei den Aktivitäten Sitzen und Stehen zu erkennen. Bei diesen Aktivitäten befindet sich das Smartphone in Ruhe. Die Beschleunigung ist also konstant.
- Die meisten Ausschläge sind bei der Aktivität Laufen zu erkennen, da hier die meisten Schritte innerhalb einer gewissen Zeitspanne gemacht werden und sich am stärksten vom Boden abgedrückt wird. Daher sind die Beschleunigungswerte hier am größten und die meisten Ausschläge zu erkennen.

Folie 6: Aufgabe 1 | Die Aktivitäten

Frage:
- Auf welcher der drei Achsen sind die größten Ausschläge zu erkennen und warum?

Antwort:
- Auf der y-Achse sind die größten Ausschläge zu erkennen.
- Der Grund dafür ist die Erdbeschleunigung. Die Erdbeschleunigung wirkt bei allen Aktivitäten bis auf Sitzen auf die y-Achse des Beschleunigungsmessers, denn die Erdbeschleunigung ist zum Erdmittelpunkt hin gerichtet. Befindet sich das Handy also senkrecht in der vorderen rechten Hosentasche wirkt die Erdbeschleunigung entweder in positive oder negative y-Richtung, je nachdem wie das Handy in der Hose orientiert ist (Kamera oben bzw. Kamera unten).

Folie 7: Aufgabe 2 | Die Testpersonen

In Aufgabe 2 habt ihr Informationen über die Testpersonen, die Daten zu den einzelnen Aktivitäten aufgenommen haben, erhalten.

Frage:
- Wie viele Testpersonen haben Daten aufgenommen?

klicken

Antwort:
- Insgesamt haben 10 Personen Daten zu den verschiedenen Aktivitäten aufgenommen.

klicken

Frage:
- Wie viele Testpersonen sind weiblich und wie viele sind männlich?

klicken

Antwort:
- 5 Testpersonen sind weiblich und 5 Testpersonen sind männlich.

klicken

Frage:
- Wie alt ist die jüngste bzw. älteste Testperson?

klicken

Antwort:
- Die jüngste Testperson ist 18 Jahre alt und die älteste Testperson ist 52 Jahre alt.

Folie 8: Aufgabe 3 | Die Datenaufnahme

In der dritten Aufgabe habt ihr euch mit der Länge der Datenaufnahme und der Abtastrate beschäftigt.

Frage:
- Wie lange wurden Daten zu den jeweiligen Aktivitäten von den Testpersonen aufgenommen?

klicken

Antwort:
- Jede Aktivität wurde während der Datenaufnahme von den Testpersonen 180 Sekunden, also 3 Minuten ausgeführt.

klicken

Frage:
- Mit welcher Abtastrate wurden die Daten aufgenommen?

klicken

Antwort:
- In 180 Sekunden wurden ca. 7150 Messpunkte aufgenommen. Das entspricht einer Abtastrate von ca. 40 Hertz.

Folie 9: Modellierungskreislauf

Wir haben unser Problem der Aktivitätserkennung mit Hilfe von Sensordaten eines Smartphones nun vereinfacht. Wir betrachten nur Daten des Beschleunigungssensors, beschränken uns auf 5 Aktivitäten und die Orientierung des Smartphones bei der Datenaufnahme war immer gleich (rechte vordere Hosentasche).

klicken

In einem nächsten Schritt werden wir unsere Rohdaten noch etwas aufbereiten, um später für diese Daten einen Klassifikationsalgorithmus entwickeln zu können. Im Rahmen dieser Datenaufbereitung werden wir die Beschleunigungswerte der drei Raumdimensionen kombinieren, die Daten in kleinere Zeitfenster unterteilen und die große Menge der Daten auf eine kleinere Menge möglichst aussagekräftiger Werte reduzieren.

Folie 10: Jetzt seid ihr dran!

Jetzt seid ihr wieder dran. Öffnet Arbeitsblatt 2 und beginnt mit der Vorverarbeitung der Daten!

A.5. Plenumsdiskussion 2

Aktivitätserkennung | Vorverarbeitung der Daten

Diskussion nach Arbeitsblatt 2

Aufgabe 1 | Der Betrag des Beschleunigungsvektors

- Formel für den Betrag des Beschleunigungsvektors:

$$\texttt{vectorLength} = \sqrt{a_x^2 + a_y^2 + a_z^2}$$

- Ergänzung im Datensatz:

UserID	ActivityID	Time (s)	x (m/s^2)	y (m/s^2)	z (m/s^2)	Vector Length (m/s^2)
1	1	0.028140	5.480853	-4.508624	-6.869365	8.408040
1	1	0.053292	5.488187	-4.504134	-6.870263	8.408010
1	1	0.078443	5.494324	-4.496799	-6.872658	8.404166
1	1	0.103595	5.492229	-4.508475	-6.871311	8.415299
1	1	0.128747	5.472320	-4.499044	-6.856192	8.392204
1	1	0.153899	5.456753	-4.472849	-6.885382	8.353975
1	1	0.179050	5.481302	-4.499344	-6.910080	8.398384
1	1	0.204202	5.500462	-4.511618	-6.910679	8.424041
1	1	0.229354	5.493127	-4.491260	-6.871161	8.397457
1	1	0.254505	5.481451	-4.456682	-6.861881	8.352864

Aufgabe 2 | Einteilung der Windows

- Länge der Windows: 3 Sekunden
- Ergänzung der WindowID im Datensatz:

WindowID	UserID	ActivityID	Time (s)	x (m/s^2)	y (m/s^2)	y (m/s^2)	Vector Length (m/s^2)
1	1	1	0.028140	5.480853	-4.508624	-4.508624	8.408040
1	1	1	0.053292	5.488187	-4.504134	-4.504134	8.408010
1	1	1	0.078443	5.494324	-4.496799	-4.496799	8.404166
1	1	1	0.103595	5.492229	-4.508475	-4.508475	8.415299
1	1	1	0.128747	5.472320	-4.499044	-4.499044	8.392204

WindowID	UserID	ActivityID	Time (s)	x (m/s^2)	y (m/s^2)	y (m/s^2)	Vector Length (m/s^2)
3000	10	5	179.877510	1.800755	-5.976322	-5.976322	8.641503
3000	10	5	179.902668	2.119592	-6.945407	-6.945407	10.048384
3000	10	5	179.927826	3.062482	-9.900562	-9.900562	14.332517
3000	10	5	179.952984	2.949916	-10.929522	-10.929522	15.735657
3000	10	5	179.978142	3.050956	-10.346634	-10.346634	14.947040

Aufgabe 3 | Statistische Kenngrößen

- Welche statistischen Kenngrößen sind euch aus dem Mathematikunterricht bekannt?

→ Mittelwert, Minimum, Maximum, Modalwert, Spannweite, ...

- Welche statistischen Kenngrößen würdet ihr zur Beschreibung der Daten eines Windows nutzen?

→ Mittelwert, Minimum und Maximum des Betrags des Beschleunigungsvektors

Aufgabe 3 | Das Feature-Set 1

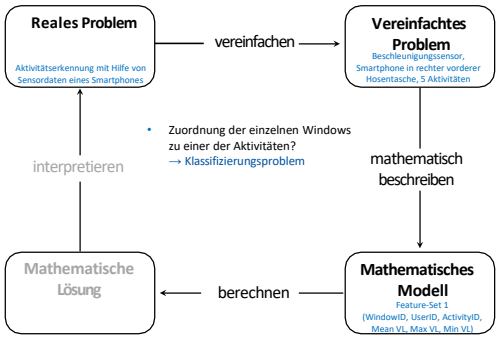
- Die ersten 5 Windows:

WindowID	UserID	ActivityID	Mean Vector Length (m/s^2)	Max Vector Length (m/s^2)	Min Vector Length (m/s^2)
1.0	1.0	1.0	8.382189	8.593413	8.136967
2.0	1.0	1.0	8.335617	8.571866	8.052737
3.0	1.0	1.0	8.289633	8.384486	8.059064
4.0	1.0	1.0	8.262474	8.533645	7.956085
5.0	1.0	1.0	8.268336	8.325962	8.224459

- Die letzten 5 Windows

WindowID	UserID	ActivityID	Mean Vector Length (m/s^2)	Max Vector Length (m/s^2)	Min Vector Length (m/s^2)
2996.0	10.0	5.0	13.059342	29.251910	4.153431
2997.0	10.0	5.0	13.351763	29.893499	5.658197
2998.0	10.0	5.0	13.004548	24.820680	3.930397
2999.0	10.0	5.0	13.723280	26.346591	4.745439
3000.0	10.0	5.0	12.273405	27.153701	4.565421

Modellierungskreislauf

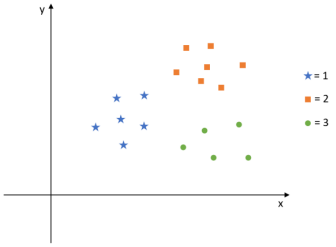


Klassifizierungsproblem

- Kennt ihr bereits aus dem Alltag Klassifizierungsprobleme?
- Wenn ja, in welchen Bereichen bzw. Anwendungen begegnen euch solche Probleme?

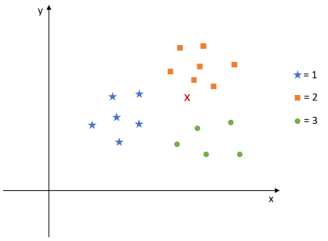
Klassifizierungsproblem

- Jeder Datenpunkt wird durch die Ausprägungen zweier Merkmale (x, y) repräsentiert
- 3 verschiedene Klassen
- **Ziel:** Zuordnung unbekannter Datenpunkte zu einer der drei Klassen



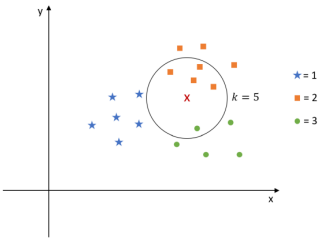
Klassifizierungsproblem

- Welcher Klasse ordnen wir das x zu und warum?
- Wie könnte ein allgemeines Vorgehen zur Klassifizierung neuer Datenpunkte aussehen?



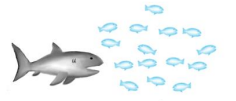
Der k-nächste-Nachbarn Algorithmus (kNN-Algorithmus)

- Grundidee: Klassifiziere unbekannte Datenpunkte anhand der Klassen der k nächst-liegenden Datenpunkte
- Klasse von x?



Jetzt seid ihr dran!

Öffnet **Arbeitsblatt 3** und beginnt mit der Entwicklung des kNN-Algorithmus!



Keine Panik!

ORGANISIEREN!

- Bearbeitet die Arbeitsblätter!
- Lest die Aufgabenstellung sorgfältig!
- Teamwork!
- Helft euch gegenseitig!
- Nutzt die Tipps!
- Nutzt das Internet!
- Fragt die Betreuer!

A.6. Notizen Plenumsdiskussion 2

Diskussion nach Arbeitsblatt 2 | Vorverarbeitung der Daten

Folie 11: Aktivitätserkennung | Vorverarbeitung der Daten

Auf dem zweiten Arbeitsblatt habt ihr die Vorverarbeitung der Daten durchgeführt. Das Vorgehen sowie die Ergebnisse und Erkenntnisse dieses Arbeitsblattes werden wir nun gemeinsam im Plenum besprechen.

Folie 12: Aufgabe 1 | Der Betrag des Beschleunigungsvektors

In der ersten Aufgabe habt ihr die Beschleunigungswerte der drei Achsen kombiniert. Dazu habt ihr euch jeden Datenpunkt in unserem Datensatz (Tabelle) als einen dreidimensionalen Vektor vorgestellt. Dieser Vektor beschreibt die Gesamtbeschleunigung des Smartphones. Um den Wert der Gesamtbeschleunigung zu bestimmen, habt ihr den Betrag dieses Vektors berechnet.

Frage:
- Wie lautet die Formel für den Betrag des Beschleunigungsvektors?

klicken

Antwort:
- $\text{vectorLength} = \sqrt{{a_x}^2 + {a_y}^2 + {a_z}^2}$

klicken

Anschließend haben wir den Betrag als zusätzliche Spalte unserem Datensatz hinzugefügt.

Bei AB2short:

In der ersten Aufgabe haben wir die Beschleunigungswerte der drei Achsen kombiniert, indem wir den sogenannten Betrag des Beschleunigungsvektors bestimmt haben. Dieser Wert beschreibt die Gesamtbeschleunigung des Smartphones.

2-mal klicken

Dieser lässt sich mit folgender Formel bestimmen.

klicken

Anschließend haben wir den Betrag als zusätzliche Spalte unserem Datensatz hinzugefügt.

Folie 13: Aufgabe 2 | Einteilung der Windows

In der zweiten Aufgabe habt ihr die große Menge an Rohdaten in kleinere Zeitfenster einer festen Länge, sogenannte **Windows**, unterteilt.

Frage:
- Welche Länge der Windows haben wir gewählt?

klicken

Antwort:
- Als Länge der Windows haben wir 3 Sekunden festgelegt. Auch andere Zeitspannen wären möglich. Der Einheitlichkeit wegen haben wir die Länge der Windows aber auf 3 Sekunden festgelegt.

klicken

Anschließend haben wir jedem Zeitfenster eine WindowID zugeordnet und diese als zusätzliche Spalte dem Datensatz hinzugefügt.

Folie 14: Aufgabe 3 | Statistische Kenngrößen

In der dritten Aufgabe habt ihr die große Menge an Rohdaten auf eine kleinere Menge möglichst aussagekräftiger Werte, sogenannte **Features** reduziert. Diese Features sind nichts anderes als statistische Kenngrößen.

klicken

Frage:
- Welche statistischen Kenngrößen sind euch aus dem Mathematikunterricht bekannt?

klicken

Antwort:
- Mittelwert, Minimum, Maximum, Modalwert, Spannweite, etc.

Anschließend habt ihr euch die Daten verschiedener Windows noch einmal graphisch angeschaut und überlegt, mit welchen statistischen Kenngrößen wir die Daten beschreiben könnten.

klicken

Frage:
- Welche statistischen Kenngrößen würdet ihr zur Beschreibung der Daten eines Windows nutzen?

klicken

Antwort:
- Die Daten eines Windows können mit sehr vielen verschiedenen statistischen Kenngrößen beschrieben werden.
- Wir haben uns aber zunächst auf den Mittelwert, das Minimum und das Maximum des Betrags des Beschleunigungsvektors beschränkt.

Folie 15: Aufgabe 3 | Das Feature-Set 1

Um für unsere Daten ein maschinelles Lernverfahren entwickeln zu können, haben wir die berechneten statistischen Kenngrößen zusammen mit der entsprechenden WindowID, ActivityID und UserID in einem neuen Datensatz abgespeichert. Diesen neuen Datensatz haben wir als **Feature-Set 1** bezeichnet.

Folie 16: Modellierungskreislauf

Wir haben unser vereinfachtes Problem nun mathematisch beschrieben, indem wir das Feature-Set 1 erstellt haben.

klicken

Als nächstes stellt sich uns die Frage, wie wir die Daten der einzelnen Windows einer der fünf Aktivitäten zuordnen können. Bei diesem Problem handelt es sich um ein sogenanntes Klassifizierungsproblem.

Folie 17: Klassifizierungsproblem

Fragen:
- Kennt ihr bereits aus dem Alltag Klassifizierungsprobleme?
- Wenn ja, in welchen Bereichen bzw. Anwendungen begegnen euch solche Probleme?

Antwort:
- Ideen der Schüler:innen sammeln!

Folie 18: Klassifizierungsproblem

Schauen wir uns das Schaubild rechts an.

klicken

Jeder Datenpunkt wird durch die Ausprägungen zweier Merkmale, hier x und y, repräsentiert.

klicken

Zudem gehört jeder Datenpunkt zu einer der drei Klassen.

klicken

Das Ziel bzw. die Aufgabe bei einem Klassifizierungsproblems ist es, neue unbekannte Datenpunkte einer der drei Klassen zuzuordnen.

Folie 19: Klassifizierungsproblem

Ein neuer unbekannter Datenpunkt könnte zum Beispiel das rote x sein.

klicken

Frage:
- Welcher Klasse würdet ihr das rote x zuordnen und warum?

Antwort:
- Verschiedene Antworten der Schüler:innen sammeln!
- Das rote x sollte der Klasse 2, also den orangenen Quadraten, zugeordnet werden, da der Abstand zu den anderen Datenpunkten dieser Klasse am kleinsten ist.

klicken

Frage:
- Wie könnte ein allgemeines Vorgehen zur Klassifizierung neuer Datenpunkte aussehen?

Antwort:
- Ideen der Schüler:innen sammeln!

Folie 20: Der k-nächste-Nachbarn-Algorithmus (kNN-Algorithmus)

Im Workshop werden wir zur Lösung des Klassifizierungsproblems den k-nächste-Nachbarn-Algorithmus nutzen.

klicken

Die Grundidee dieses Algorithmus ist es, unbekannte Datenpunkte anhand der Klassen der k nächstliegenden Datenpunkte zu klassifizieren.

klicken

Zum Beispiel könnten wir uns die fünf nächstliegenden Datenpunkte des roten x anschauen und anhand derer das rote x einer Klasse zuordnen. Die Frage, welcher Klasse das rote x zugeordnet werden sollte, werdet ihr in Aufgabe 1 des nächsten Arbeitsblattes beantworten.

Folie 21: Jetzt seid ihr dran!

Öffnet also Arbeitsblatt 3 und beginnt mit der Entwicklung des k-nächste-Nachbarn-Algorithmus!

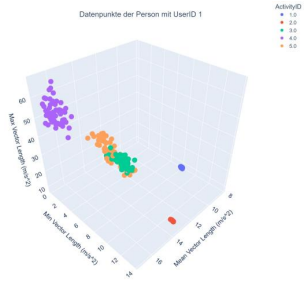
A.7. Plenumsdiskussion 3

Aktivitätserkennung | Der k-nächste-Nachbarn-Algorithmus

Diskussion nach Arbeitsblatt 3

Aufgabe 2 | Graphische Darstellung der Datenpunkte

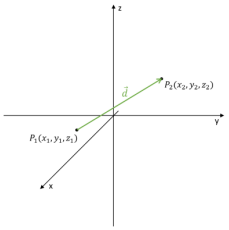
- Bei welchen Aktivitäten könnte der k-nächste-Nachbarn-Algorithmus bei der Klassifikation Schwierigkeiten haben und warum?



Aufgabe 2 | Die Abstandsfunktion

- Formel für den Abstand zwischen den beiden Punkten $P_1(x_1, y_1, z_1)$ und $P_2(x_2, y_2, z_2)$:

  `distanceP1P2` $= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$



Aufgabe 3 | Suche der k nächsten Nachbarn

- Wie sind wir bei der Suche nach den k nächsten Nachbarn vorgegangen?

Aufgabe 3 | Suche der k nächsten Nachbarn

- Wie sind wir bei der Suche nach den k nächsten Nachbarn vorgegangen?

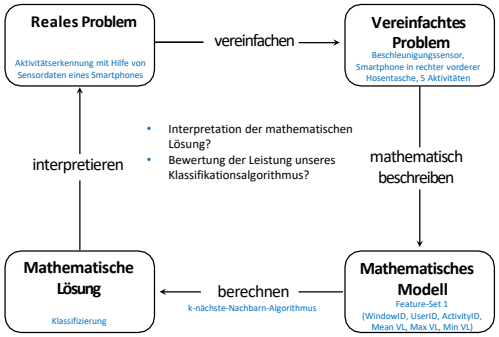
1. Leere Liste erstellen, in der nach und nach die Abstände gespeichert werden sollen.
   `distances = []`
2. For-Schleife zur Berechnung der Abstände zwischen allen Windows und dem Test-Window und Abspeichern in der Liste distances:
   ```
   for i in range(1, 3000):
       dist = computeDistances(i, TestWindow)
       addDistances(i, dist, distances)
   ```
3. Löschen des Abstands des Test-Windows zu sich selbst
   `del distances[TestWindow-1]`
4. Sortieren der Liste nach den kürzesten Abständen zum Test-Window
   `sortDistances(distances)`
5. Filtern der k nächsten Nachbarn
   `kneighbors = getNeighbors(distances, k)`

Aufgabe 4 | Die Häufigkeiten der einzelnen Klassen

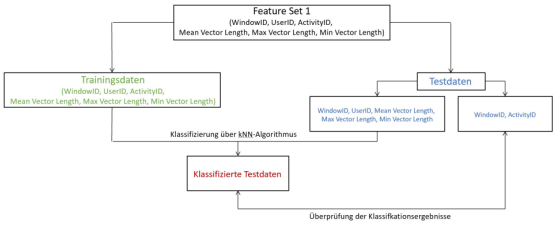
- Die fünf nächsten Nachbarn des Test-Windows 200:
  - `(2881, 0.86, 4)`
  - `(198, 0.88, 4)`
  - `(206, 1.09, 4)`
  - `(204, 1.11, 4)`
  - `(510, 1.14, 4)`
- Welcher Aktivität ordnen wir die Daten des Test-Windows 200 zu?

→ Aktivität 4 (Laufen)

Modellierungskreislauf

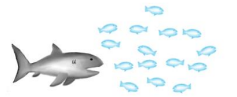


Methode des überwachten Maschinellen Lernens (engl. supervised learning)



Jetzt seid ihr dran!

Öffnet **Arbeitsblatt 4** und beginnt mit der Bewertung der Leistung des k-nächste-Nachbarn-Algorithmus!



Keine Panik!

ORGANISIEREN!

- Bearbeitet die Arbeitsblätter!
- Lest die Aufgabenstellung sorgfältig!
- Teamwork!
- Helft euch gegenseitig!
- Nutzt die Tipps!
- Nutzt das Internet!
- Fragt die Betreuer!

A.8. Notizen Plenumsdiskussion 3

Diskussion nach Arbeitsblatt 3 | Der k-nächste-Nachbarn-Algorithmus

Folie 22: Aktivitätserkennung | Der k-nächste-Nachbarn-Algorithmus

Auf dem dritten Arbeitsblatt habt ihr den k-nächste-Nachbarn-Algorithmus kennengelernt und bereits erste Datenpunkte mithilfe des Algorithmus einer Klasse bzw. einer Aktivität zugeordnet. Wir werden nun die Ergebnisse und Erkenntnisse dieses Arbeitsblattes gemeinsam im Plenum besprechen.

Folie 23: Aufgabe 2 | Graphische Darstellung der Datenpunkte

Jedes Window entspricht einem Datenpunkt in drei Dimensionen. Die Dimensionen sind die drei berechneten Features Mittelwert, Maximum und Minimum des Betrags des Beschleunigungsvektors. In Aufgabe 2 habt ihr euch die Datenpunkte der Windows graphisch angeschaut.

Frage:
- Bei welchen Aktivitäten könnte der k-nächste-Nachbarn-Algorithmus bei der Klassifikation Schwierigkeiten haben und warum?

Antwort:
- Bei den Aktivitäten 3 und 5 kann es zu Problemen kommen, da hier die Datenpunkte sehr eng aneinander liegen bzw. sich die Datenwolken zum Teil auch überschneiden. Daher können unter den k nächsten Nachbarn eines Datenpunktes auch Datenpunkte der „falschen“ Aktivität liegen.

Folie 24: Aufgabe 2 | Die Abstandsfunktion (Überspringen bei AB3short!)

Damit wir die k nächsten Nachbarn eines Datenpunktes bzw. Windows finden können, brauchen wir eine Funktion, die uns den Abstand zwischen zwei Datenpunkten angibt.

Frage:
- Wie lautet die Formel für den Abstand zwischen den beiden Punkten $P_1(x_1, y_1, z_1)$ und $P_2(x_2, y_2, z_2)$?

klicken

Antwort:
- $\text{distanceP1P2} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$

Folie 25: Aufgabe 3 | Suche der k nächsten Nachbarn (Überspringen bei AB3short!)

In Aufgabe 3 ging es um die Frage, wie wir die k nächsten Nachbarn von einem ausgewählten Datenpunkt (Window) bestimmen können.

Frage:
- Wie sind wir bei der Suche nach den k nächsten Nachbarn vorgegangen?

Antwort:
- Antworten der Schüler:innen sammeln.

Folie 26: Aufgabe 3 | Suche der k nächsten Nachbarn (Überspringen bei AB3short!)

Wir sind bei der Suche der k nächsten Nachbarn wie folgt vorgegangen:
1. Wir haben eine leere Liste erstellt, in der nach und nach die Abstände abgespeichert wurden.

2. Anschließend haben wir über eine for-Schleife die Abstände zwischen allen Windows und dem Test-Window berechnet und diese in der zuvor erstellten Liste abgespeichert.
3. Danach haben wir den Abstand des Test-Windows zu sich selbst aus der Liste gelöscht. Da Python bei der Nummerierung bei 0 beginnt, müssen wir die Zahl 1 von der WindowID des Test-Windows subtrahieren.
4. Im vorletzten Schritt haben wir die Liste nach den kürzesten bzw. kleinsten Abständen sortiert.
5. Im letzten Schritt haben wir die k ersten Einträge aus der Liste der Abstände aufgerufen. Diese sind gerade die k nächsten Nachbarn zu unserem Test-Window.

Folie 27: Aufgabe 4 | Die Häufigkeiten der einzelnen Klassen

Nachdem wir die k nächsten Nachbarn zu unserem Test-Window gefunden haben, müssen wir noch entscheiden, welcher Klasse bzw. Aktivität wir das Test-Window zuordnen.

Frage:
- Welcher Aktivität ordnen wir die Daten des Test-Windows 200 anhand der fünf nächsten Nachbarn zu?

klicken

Antwort:
- Die Daten des Test-Windows 200 ordnen wir der Aktivität 4 (Laufen) zu, da diese Aktivität unter den fünf nächsten Nachbarn am häufigsten vorkommt.

Folie 28: Modellierungskreislauf

Schauen wir nochmal auf den Modellierungskreislauf. Wir haben nun mit dem k-nächsten-Nachbarn-Algorithmus einen Algorithmus entwickelt, mit dem wir die Datenpunkte der Windows den einzelnen Aktivitäten zuordnen können. Wir können also mit Hilfe des k-nächste-Nachbarn-Algorithmus aus unserem mathematischen Modell, dem Feature-Set 1, eine mathematische Lösung, die Klassifizierung, berechnen.

klicken

Aber wie gut funktioniert unser Algorithmus und wie können wir die Ergebnisse der Klassifizierung zahlenmäßig bewerten?

Folie 29: Methode des überwachten Maschinellen Lernens (engl. supervised learning)

Zur Bewertung der Ergebnisse der Klassifizierung nutzen wir eine Methode aus dem Bereich des **überwachten Maschinellen Lernens (engl. supervised learning)**. Wir teilen unsere Daten, also das Feature-Set 1, in **Trainings-** und **Testdaten** auf. Die Trainingsdaten bestehen aus einem Großteil der Daten des Feature-Sets 1. Der Rest der Daten, die nicht den Trainingsdaten zugeordnet werden, sind unsere Testdaten. Bei den Testdaten blenden wir zunächst die ActivityID aus, d.h. wir gehen davon aus, dass wir die zugehörige Aktivität eines Datenpunktes im Testdatensatz nicht kennen. Anschließend klassifizieren wir die Testdatenpunkte mit Hilfe des k-nächste-Nachbarn-Algorithmus und den Trainingsdaten. Wir berechnen also für alle Datenpunkte im Testdatensatz die k nächsten Nachbarn aus dem Trainingsdatensatz und bestimmen über einen Mehrheitsentscheid die Aktivitäten der Datenpunkte. Nach der Klassifizierung überprüfen wir, ob unsere Zuordnungen mit den tatsächlichen Aktivitäten der einzelnen Testdatenpunkte übereinstimmen.

Folie 30: Jetzt seid ihr dran!

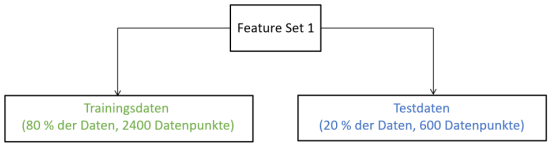
Diese Methode werdet ihr jetzt auf dem nächsten Arbeitsblatt anwenden. Öffnet also Arbeitsblatt 4!

A.9. Plenumsdiskussion 4

Aktivitätserkennung | Bewertung der Klassifikationsergebnisse

Diskussion nach Arbeitsblatt 4

Aufgabe 1 | Die Trainings- und Testdaten



Aufgabe 2 | Die Wahrheitsmatrix

	Klassifiziert als 1 (Sitzen)	Klassifiziert als 2 (Stehen)	Klassifiziert als 3 (Gehen)	Klassifiziert als 4 (Laufen)	Klassifiziert als 5 (Treppen auf- und absteigen)
Tatsächlich 1 (Sitzen)	123	0	0	0	0
Tatsächlich 2 (Stehen)	0	115	0	0	0
Tatsächlich 3 (Gehen)	0	0	79	10	19
Tatsächlich 4 (Laufen)	0	0	14	120	1
Tatsächlich 5 (Treppen auf- und absteigen)	1	0	24	1	93

Aufgabe 2 | Die Wahrheitsmatrix

- Wie viele Datenpunkte wurden insgesamt **richtig** klassifiziert?

	Klassifiziert als 1 (Sitzen)	Klassifiziert als 2 (Stehen)	Klassifiziert als 3 (Gehen)	Klassifiziert als 4 (Laufen)	Klassifiziert als 5 (Treppen auf- und absteigen)
Tatsächlich 1 (Sitzen)	123	0	0	0	0
Tatsächlich 2 (Stehen)	0	115	0	0	0
Tatsächlich 3 (Gehen)	0	0	79	10	19
Tatsächlich 4 (Laufen)	0	0	14	120	1
Tatsächlich 5 (Treppen auf- und absteigen)	1	0	24	1	93

Aufgabe 2 | Die Wahrheitsmatrix

- Wie viele Datenpunkte wurden insgesamt **falsch** klassifiziert?

	Klassifiziert als 1 (Sitzen)	Klassifiziert als 2 (Stehen)	Klassifiziert als 3 (Gehen)	Klassifiziert als 4 (Laufen)	Klassifiziert als 5 (Treppen auf- und absteigen)
Tatsächlich 1 (Sitzen)	123	0	0	0	0
Tatsächlich 2 (Stehen)	0	115	0	0	0
Tatsächlich 3 (Gehen)	0	0	79	10	19
Tatsächlich 4 (Laufen)	0	0	14	120	1
Tatsächlich 5 (Treppen auf- und absteigen)	1	0	24	1	93

Aufgabe 2 | Die Wahrheitsmatrix

- Bei welcher Aktivität wurden die meisten Datenpunkte **richtig** klassifiziert?

	Klassifiziert als 1 (Sitzen)	Klassifiziert als 2 (Stehen)	Klassifiziert als 3 (Gehen)	Klassifiziert als 4 (Laufen)	Klassifiziert als 5 (Treppen auf- und absteigen)
Tatsächlich 1 (Sitzen)	123	0	0	0	0
Tatsächlich 2 (Stehen)	0	115	0	0	0
Tatsächlich 3 (Gehen)	0	0	79	10	19
Tatsächlich 4 (Laufen)	0	0	14	120	1
Tatsächlich 5 (Treppen auf- und absteigen)	1	0	24	1	93

Aufgabe 2 | Die Wahrheitsmatrix

- Bei welcher Aktivität wurden die meisten Datenpunkte **falsch** klassifiziert?

	Klassifiziert als 1 (Sitzen)	Klassifiziert als 2 (Stehen)	Klassifiziert als 3 (Gehen)	Klassifiziert als 4 (Laufen)	Klassifiziert als 5 (Treppen auf- und absteigen)
Tatsächlich 1 (Sitzen)	123	0	0	0	0
Tatsächlich 2 (Stehen)	0	115	0	0	0
Tatsächlich 3 (Gehen)	0	0	79	10	19
Tatsächlich 4 (Laufen)	0	0	14	120	1
Tatsächlich 5 (Treppen auf- und absteigen)	1	0	24	1	93

Aufgabe 2 | Die Wahrheitsmatrix

- Bei welchen beiden Aktivitäten liegen die meisten Überschneidungen vor?

	Klassifiziert als 1 (Sitzen)	Klassifiziert als 2 (Stehen)	Klassifiziert als 3 (Gehen)	Klassifiziert als 4 (Laufen)	Klassifiziert als 5 (Treppen auf- und absteigen)
Tatsächlich 1 (Sitzen)	123	0	0	0	0
Tatsächlich 2 (Stehen)	0	115	0	0	0
Tatsächlich 3 (Gehen)	0	0	79	10	19
Tatsächlich 4 (Laufen)	0	0	14	120	1
Tatsächlich 5 (Treppen auf- und absteigen)	1	0	24	1	93

Aufgabe 3 | Die verschiedenen Qualitätsmaße

- Die Genauigkeit:

$$\texttt{accuracy} = \frac{CM_{11} + CM_{22} + CM_{33} + CM_{44} + CM_{55}}{CM_{Sum}} \approx 0.88$$

- Die Fehlerrate:

$$\texttt{errorRate} = 1 - \texttt{accuracy} \approx 0.12$$

Aufgabe 3 | Die verschiedenen Qualitätsmaße

- Die Präzision:

$$\texttt{precisionSitzen} = \frac{CM_{11}}{CM_{11} + CM_{21} + CM_{31} + CM_{41} + CM_{51}} \approx 0.99$$

$$\texttt{precisionStehen} = \frac{CM_{22}}{CM_{12} + CM_{22} + CM_{32} + CM_{42} + CM_{52}} = 1.0$$

$$\texttt{precisionGehen} = \frac{CM_{33}}{CM_{13} + CM_{23} + CM_{33} + CM_{43} + CM_{53}} \approx 0.68$$

$$\texttt{precisionLaufen} = \frac{CM_{44}}{CM_{14} + CM_{24} + CM_{34} + CM_{44} + CM_{54}} \approx 0.92$$

$$\texttt{precisionTreppen} = \frac{CM_{55}}{CM_{15} + CM_{25} + CM_{35} + CM_{45} + CM_{55}} \approx 0.82$$

Aufgabe 3 | Die verschiedenen Qualitätsmaße

- Was bedeutet es, wenn der Wert der Präzision nahe bei 1 liegt? Bei welchen Aktivitäten ist dies der Fall?

→ Nahezu alle als Aktivität z klassifizierten Datenpunkte sind richtig als Aktivität z klassifiziert

→ Aktivität *Sitzen* und Aktivität *Stehen*

- Bei welcher Aktivität werden die Datenpunkte am häufigsten richtig zugeordnet und warum?

→ Aktivität *Stehen*

Aufgabe 3 | Die verschiedenen Qualitätsmaße

- Bei welchen Aktivitäten werden die Datenpunkte am häufigsten falsch zugeordnet und was könnten Gründe dafür sein?

→ Aktivität *Gehen* und Aktivität *Treppen auf- und absteigen*

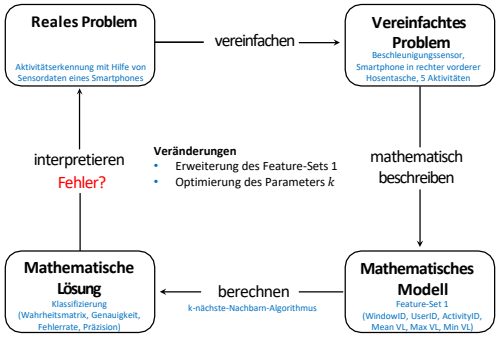
- Sind die Ergebnisse der Klassifikation im Großen und Ganzen zufriedenstellend? Falls nein, wo müssten die Ergebnisse noch verbessert werden?

→ Ergebnisse sind bei den Aktivitäten *Gehen* und *Treppen auf- und absteigen* noch nicht zufriedenstellend

Aufgabe 4 | Ideen zur Verbesserung

- Was können wir an unserem bisherigen Vorgehen verändern, um bessere Ergebnisse zu erhalten?

Modellierungskreislauf



Jetzt seid ihr dran!

Öffnet **Arbeitsblatt 5** und beginnt mit der Erweiterung des Feature-Sets 1!

- Bearbeitet die Arbeitsblätter!
- Lest die Aufgabenstellung sorgfältig!
- Teamwork!
- Helft euch gegenseitig!
- Nutzt die Tipps!
- Nutzt das Internet!
- Fragt die Betreuer!

A.10. Notizen Plenumsdiskussion 4

Diskussion nach Arbeitsblatt 4 | Bewertung der Klassifikationsergebnisse

Folie 31: Aktivitätserkennung | Bewertung der Klassifikationsergebnisse

Auf Arbeitsblatt 4 habt ihr die Leistung unseres Klassifikationsalgorithmus bewertet. Wir werden nun die Ergebnisse und Erkenntnisse gemeinsam im Plenum besprechen.

Folie 32: Aufgabe 1 | Die Trainings- und Testdaten

Der erste Schritt auf Arbeitsblatt 4 war die Aufteilung des Feature-Set 1 in Trainings- und Testdaten. Die Trainingsdaten bestanden aus 80 % der Daten, also 2400 Datenpunkten, und die Testdaten aus 20 % der Daten, also 600 Datenpunkten.

Folie 33: Aufgabe 2 | Die Wahrheitsmatrix

Die Ergebnisse der Klassifikation können wir uns in Form einer Tabelle, der sogenannten **Wahrheitsmatrix** ausgeben lassen.

klicken

Folgende Ergebnisse solltet ihr erhalten haben. Gleicht die Ergebnisse bitte mit eurer ausgefüllten Tabelle auf dem Antwortblatt ab.

Folie 34: Aufgabe 2 | Die Wahrheitsmatrix

Frage:
- Wie viele Datenpunkte wurden insgesamt **richtig** klassifiziert?

klicken

Antwort:
- Um die Anzahl der richtig klassifizierten Datenpunkte bestimmen zu können, müssen wir die grün markierten Einträge addieren.
- Insgesamt wurden also 530 Datenpunkte richtig klassifiziert.

Folie 35: Aufgabe 2 | Die Wahrheitsmatrix

Frage:
- Wie viele Datenpunkte wurden insgesamt **falsch** klassifiziert?

klicken

Antwort:
- Um die Anzahl der falsch klassifizierten Datenpunkte bestimmen zu können, müssen wir die rot markierten Einträge addieren.
- Insgesamt wurden also 70 Datenpunkte falsch klassifiziert.

Folie 36: Aufgabe 2 | Die Wahrheitsmatrix

Frage:
- Bei welcher Aktivität wurden die meisten Datenpunkte **richtig** klassifiziert?

klicken

Antwort:
- Bei Aktivität 1 (Sitzen) wurden mit 123 Datenpunkten die meisten Datenpunkte richtig klassifiziert.

Folie 37: Aufgabe 2 | Die Wahrheitsmatrix

Frage:
- Bei welcher Aktivität wurden die meisten Datenpunkte **falsch** klassifiziert?

klicken

Antwort:
- Bei Aktivität 3 (Gehen) wurden mit 29 Datenpunkten die meisten Datenpunkte falsch klassifiziert.

Folie 38: Aufgabe 2 | Die Wahrheitsmatrix

Frage:
- Bei welchen beiden Aktivitäten liegen die meisten Überschneidungen vor?

klicken

Antwort:
- Bei Aktivität 3 (Gehen) und Aktivität 5 (Treppen auf- und absteigen) liegen die meisten Überschneidungen vor.
- 19 Datenpunkte werden anstelle von Gehen als Treppen auf- und absteigen klassifiziert.
- 24 Datenpunkte werden anstelle von Treppen auf- und absteigen als Gehen klassifiziert.

Folie 39: Aufgabe 3 | Die verschiedenen Qualitätsmaße

Neben der Wahrheitsmatrix können wir die Klassifikationsergebnisse auch mit Hilfe verschiedener Qualitätsmaße bewerten. Als erstes Qualitätsmaß haben wir uns die **Genauigkeit** (engl. **accuracy**) angeschaut.

Fragen:
- Wie lautet die Formel für die Genauigkeit und was gibt sie an?
- Welchen Wert hat die Genauigkeit bei unserer Klassifikation?

klicken

Antworten:
- Die Genauigkeit gibt das Verhältnis der richtig klassifizierten Datenpunkte zu allen Datenpunkten an.
- $\text{accuracy} = \frac{CM_{11}+CM_{22}+CM_{33}+CM_{44}+CM_{55}}{CM_{Sum}} \approx 0.88$

Als zweites Qualitätsmaß haben wir uns die **Fehlerrate** (engl. **error rate**) angeschaut.

Fragen:
- Wie lautet die Formel für die Fehlerrate und was gibt sie an?

- Welchen Wert hat die Fehlerrate bei unserer Klassifikation?

klicken

Antworten:
- Die Fehlerrate gibt das Verhältnis der falsch klassifizierten Datenpunkte zu allen Datenpunkten an.
- $\text{error rate} = 1 - \text{accuracy} \approx 0.12$

Folie 40: Aufgabe 3 | Die verschiedenen Qualitätsmaße

Als drittes und letztes Qualitätsmaß haben wir uns noch die **Präzision** (engl. **precision**) angeschaut.

Fragen:
- Wie lauten die Formeln für die Präzision der einzelnen Aktivitäten und was gibt sie an?
- Welche Werte hat die Präzision bei unserer Klassifikation?

klicken

Antworten:
- Die Präzision gibt das Verhältnis der richtig als Aktivität z klassifizierten Datenpunkte zu allen als Aktivität z klassifizierten Datenpunkten an.
- $\text{precisionSitzen} = \frac{CM_{11}}{CM_{11}+CM_{21}+CM_{31}+CM_{41}+CM_{51}} \approx 0.99$
- $\text{precisionStehen} = \frac{CM_{22}}{CM_{12}+CM_{22}+CM_{32}+CM_{42}+CM_{52}} = 1.0$
- $\text{precisionGehen} = \frac{CM_{33}}{CM_{13}+CM_{23}+CM_{33}+CM_{43}+CM_{53}} \approx 0.68$
- $\text{precisionLaufen} = \frac{CM_{44}}{CM_{14}+CM_{24}+CM_{34}+CM_{44}+CM_{54}} \approx 0.92$
- $\text{precisionTreppen} = \frac{CM_{55}}{CM_{15}+CM_{25}+CM_{35}+CM_{45}+CM_{55}} \approx 0.82$

Folie 41: Aufgabe 3 | Die verschiedenen Qualitätsmaße

Wir wollen uns die Werte der Qualitätsmaße noch genauer anschauen und wichtige Informationen aus ihnen gewinnen.

Frage:
- Was bedeutet es, wenn der Wert der Präzision nahe bei 1 liegt?
- Bei welchen Aktivitäten ist dies der Fall?

klicken

Antworten:
- Wenn der Wert der Präzision nahe bei 1 liegt, sind nahezu alle als Aktivität z klassifizierten Datenpunkte auch richtig als Aktivität z klassifiziert.
- Dies ist bei den Aktivitäten 1 und 2 (Sitzen und Stehen) der Fall.

klicken

Frage:
- Bei welcher Aktivität werden die Datenpunkte am häufigsten richtig zugeordnet und warum?

klicken

Antwort:
- Bei Aktivität 2 (Stehen) werden die Datenpunkte am häufigsten richtig zugeordnet, da hier die Präzision am größten ist.

Folie 42: Aufgabe 3 | Die verschiedenen Qualitätsmaße

Fragen:
- Bei welchen Aktivitäten werden die Datenpunkte am häufigsten falsch zugeordnet und was könnten Gründe dafür sein?

klicken

Antwort:
- Bei den Aktivitäten 2 und 5 (Gehen und Treppen auf- und absteigen) werden die Datenpunkte am häufigsten falsch zugeordnet.
- Das liegt daran, dass die Datenpunkte der beiden Klassen sehr nahe aneinander liegen und sich die Datenwolken zum Teil überschneiden.
- Die charakteristischen Bewegungen von Gehen und Treppen auf- und absteigen sind außerdem sehr ähnlich. Ist beispielsweise ein Stockwerk erreicht, geht der Nutzer ein paar Schritte normal bis er wieder die nächste Treppenstufe hinauf bzw. hinabsteigt.

klicken

Fragen:
- Sind die Ergebnisse der Klassifikation im Großen und Ganzen zufriedenstellend?
- Falls nein, wo müssten die Ergebnisse noch verbessert werden?

klicken

Antwort:
- Die Ergebnisse sind bei den Aktivitäten 3 und 5 (Gehen und Treppen auf- und absteigen) noch nicht zufriedenstellend.
- Bei diesen Aktivitäten muss die Präzision noch verbessert werden.

Folie 43: Aufgabe 4 | Ideen zur Verbesserung

Wir haben nun festgestellt, dass unsere Ergebnisse noch nicht ganz zufriedenstellend sind.

Frage:
- Was können wir an unserem bisherigen Vorgehen verändern, um bessere Ergebnisse zu erhalten?

Antwort:
- Ideen der Schüler:innen sammeln!

Folie 44: Modellierungskreislauf

Wir haben nun mit der Wahrheitsmatrix und den unterschiedlichen Qualitätsmaßen eine Mathematische Lösung unseres Problems gefunden, die wir auch bewerten und interpretieren können. Leider sind unsere Ergebnisse noch nicht zu 100 % zufriedenstellend.

klicken

Daher werden wir versuchen, unsere Ergebnisse durch zwei Veränderungen zu verbessern. Zum einen werden wir unser Feature-Set 1 erweitern und zum anderen werden wir den Parameter k optimieren.

Folie 45: Jetzt seid ihr dran!

Öffnet das Arbeitsblatt 5 und beginnt mit der Erweiterung des Feature-Sets 1!

A.11. Plenumsdiskussion 5

Aktivitätserkennung | Erweiterung des Feature-Sets 1

Diskussion nach Arbeitsblatt 5

Aufgabe 2 | Weitere statistische Kenngrößen

- Welche statistischen Kenngrößen kennt ihr neben dem Mittelwert, dem Maximum und dem Minimum bereits aus dem Mathematikunterricht?

→ Median, oberes Quartil, unteres Quartil, Quartilsabstand, …

Aufgabe 2 | Das Feature-Set 2

- Die ersten 5 Windows:

WindowID	UserID	ActivityID	Mean Vector Length (m/s^2)	Max Vector Length (m/s^2)	Min Vector Length (m/s^2)	Median Vector Length (m/s^2)	Upper Quartile Vector Length (m/s^2)	Lower Quartile Vector Length (m/s^2)	Mean x (m/s^2)	Max x (m/s^2)	Min x (m/s^2)	Mean y (m/s^2)	Max y (m/s^2)	Min y (m/s^2)	Mean z (m/s^2)	Max z (m/s^2)	Min z (m/s^2)
1.0	1.0	1.0	8.382189	8.593413	8.136967	8.385889	8.437079	8.323079	5.505176	5.615423	5.450765	-4.469158	-4.271966	-4.689299	-6.878246	-6.785539	-7.006031
2.0	1.0	1.0	8.335617	8.571866	8.052737	8.338597	8.377425	8.323079	5.519143	5.657186	5.373675	-4.416821	-4.172723	-4.578230	-6.900961	-6.728059	-7.023245
3.0	1.0	1.0	8.289633	8.384486	8.059064	8.290660	8.312062	8.323079	5.486834	5.620662	5.331613	-4.393822	-4.272116	-4.490362	-6.943257	-6.869365	-7.107370
4.0	1.0	1.0	8.262474	8.533645	7.956085	8.275744	8.313050	8.323079	5.509859	5.862559	5.325925	-4.353382	-4.147874	-4.539909	-6.948716	-6.761290	-7.173083
5.0	1.0	1.0	8.268336	8.325962	8.224459	8.269243	8.277988	8.323079	5.513971	5.550308	5.474566	-4.356670	-4.315825	-4.397256	-6.946351	-6.911877	-6.987170

- Die letzten 5 Windows:

WindowID	UserID	ActivityID	Mean Vector Length (m/s^2)	Max Vector Length (m/s^2)	Min Vector Length (m/s^2)	Median Vector Length (m/s^2)	Upper Quartile Vector Length (m/s^2)	Lower Quartile Vector Length (m/s^2)	Mean x (m/s^2)	Max x (m/s^2)	Min x (m/s^2)	Mean y (m/s^2)	Max y (m/s^2)	Min y (m/s^2)	Mean z (m/s^2)	Max z (m/s^2)	Min z (m/s^2)
2996.0	10.0	5.0	13.059342	29.251910	4.153431	11.829134	17.457058	8.323079	2.393618	6.628067	-0.037123	-9.037663	-2.933300	-20.402273	-2.914359	4.172124	-8.865464
2997.0	10.0	5.0	13.351763	29.893499	5.658197	12.509836	15.154778	8.323079	3.052110	7.301067	0.417632	-9.145451	-3.231779	-20.826491	-2.046497	3.812721	-8.422236
2998.0	10.0	5.0	13.004548	24.820680	3.930397	12.639155	16.638583	8.323079	2.477606	5.964497	-0.236957	-8.984599	-2.712060	-17.457447	-2.624506	2.992876	-8.410560
2999.0	10.0	5.0	13.723280	26.346591	4.745439	13.141664	16.249636	8.323079	3.085341	6.180348	-0.250130	-9.411966	-2.899471	-18.516495	-2.318595	3.179837	-9.714199
3000.0	10.0	5.0	12.273405	27.153701	4.565421	11.657419	14.399818	8.323079	2.515001	6.569688	0.141456	-8.458025	-3.097060	-18.827698	-3.412054	1.114133	-8.295000

Aufgabe 3 | Klassifikation

	Klassifiziert als 1 (Sitzen)	Klassifiziert als 2 (Stehen)	Klassifiziert als 3 (Gehen)	Klassifiziert als 4 (Laufen)	Klassifiziert als 5 (Treppen auf- und absteigen)
Tatsächlich 1 (Sitzen)	123	0	0	0	0
Tatsächlich 2 (Stehen)	0	115	0	0	0
Tatsächlich 3 (Gehen)	0	0	105	2	1
Tatsächlich 4 (Laufen)	0	0	2	133	0
Tatsächlich 5 (Treppen auf- und absteigen)	0	0	10	2	107

Aufgabe 3 | Klassifikation

- Genauigkeit:

$$\texttt{accuracy} \approx 0.97$$

- Fehlerrate:

$$\texttt{errorRate} \approx 0.03$$

- Präzision:

$$\texttt{precisionSitzen} = 1.0$$
$$\texttt{precisionStehen} = 1.0$$
$$\texttt{precisionGehen} \approx 0.90$$
$$\texttt{precisionLaufen} \approx 0.97$$
$$\texttt{precisionTreppen} \approx 0.99$$

Aufgabe 3 | Klassifikation

- Was fällt euch auf, wenn ihr die Qualitätsmaße mit den Qualitätsmaßen der Klassifikation auf Arbeitsblatt 4 vergleicht?

→ Ergebnisse durchweg zufriedenstellender

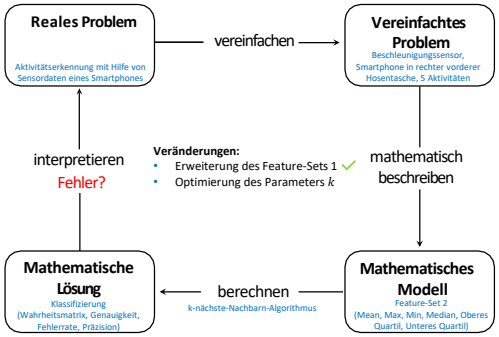
- Bei welcher Aktivität treten noch am häufigsten falsche Klassifikationen auf?

→ Aktivität *Gehen*

- Wie viele von 100 Aktivitäten werden im Mittel richtig klassifiziert?

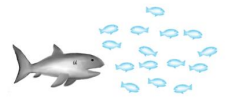
→ ca. 97 Aktivitäten

Modellierungskreislauf



Jetzt seid ihr dran!

Öffnet **Arbeitsblatt 6** und beginnt mit der Optimierung des Parameters k!



Keine Panik!

ORGANISIEREN!

- Bearbeitet die Arbeitsblätter!
- Lest die Aufgabenstellung sorgfältig!
- Teamwork!
- Helft euch gegenseitig!
- Nutzt die Tipps!
- Nutzt das Internet!
- Fragt die Betreuer!

A.12. Notizen Plenumsdiskussion 5

Diskussion nach Arbeitsblatt 5 | Erweiterung des Feature-Sets 1

Folie 46: Aktivitätserkennung | Erweiterung des Feature-Sets 1

Auf Arbeitsblatt 5 habt ihr eine Erweiterung des Feature-Sets 1 vorgenommen und geprüft, ob sich die Ergebnisse der Klassifikation verbessert haben. Wir werden nun die Erkenntnisse und Ergebnisse gemeinsam im Plenum besprechen.

Folie 47: Aufgabe 2 | Weitere statistische Kenngrößen

Wir haben unser Feature-Set 1 zum einen um den Mittelwert, das Maximum und das Minimum jeder der drei Beschleunigungsrichtungen (x, y und z) erweitert. Zum anderen haben wir aber auch neue statistische Kenngrößen mit aufgenommen.

Frage:
- Welche statistischen Kenngrößen kennt ihr neben dem Mittelwert, dem Maximum und dem Minimum bereits aus dem Mathematikunterricht?

klicken

Antwort:
- Median, oberes Quartil, unteres Quartil, Quartilsabstand, …

Folie 48: Aufgabe 2 | Das Feature-Set 2

Durch die Ergänzung der Features Median, oberes Quartil und unteres Quartil des Betrags des Beschleunigungsvektors haben wir unser Feature-Set 2 erhalten.

Folie 49: Aufgabe 3 | Klassifikation

Bei der Klassifikation mit Feature-Set 2 und $k = 3$ solltet ihr folgende Wahrheitsmatrix als Ergebnis erhalten haben. Gleicht die Ergebnisse bitte mit eurer ausgefüllten Tabelle auf dem Antwortblatt ab.

Folie 50: Aufgabe 3 | Klassifikation

Frage:
- Welche Werte haben nun die verschiedenen Qualitätsmaße?

klicken

Antwort:
- $\text{accuracy} \approx 0.97$
- $\text{error rate} \approx 0.03$
- $\text{precisionSitzen} = 1.0$
- $\text{precisionStehen} = 1.0$
- $\text{precisionGehen} \approx 0.90$
- $\text{precisionLaufen} \approx 0.97$
- $\text{precisionTreppen} \approx 0.99$

Folie 51: Aufgabe 3 | Klassifikation

Frage:
- Was fällt euch auf, wenn ihr die Qualitätsmaße mit den Qualitätsmaßen der Klassifikation auf Arbeitsblatt 4 vergleicht?

klicken

Antwort:
- Die Ergebnisse sind durchweg zufriedenstellender.

klicken

Frage:
- Bei welcher Aktivität treten noch am häufigsten falsche Klassifikationen auf?

klicken

Antwort:
- Bei Aktivität 3 (Gehen) treten noch am häufigsten falsche Klassifikationen auf (geringste Präzision).

klicken

Frage:
- Wie viele von 100 Aktivitäten werden im Mittel richtig klassifiziert?

klicken

Antwort:
- Im Mittel werden ca. 97 Aktivitäten von 100 richtig klassifiziert.

Folie 52: Modellierungskreislauf

Wir haben unsere Ergebnisse der Klassifikation durch die Erweiterung des Feature-Sets 1 verbessert. Als nächstes werden wir den Parameter k optimieren und überprüfen, ob wir auch hierdurch eine Verbesserung der Ergebnisse erzielen können.

Folie 53: Jetzt seid ihr dran!

Öffnet Arbeitsblatt 6 und beginnt mit der Optimierung des Parameters k!

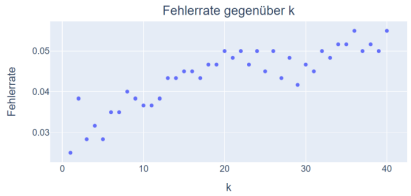
A.13. Plenumsdiskussion 6

Aktivitätserkennung | Optimierung des Parameters k

Diskussion nach Arbeitsblatt 6

Aufgabe 1 | Graphische Lösung des Optimierungsproblems

- Für welchen Wert von k ist die Fehlerrate am kleinsten?
- Warum schwankt die Fehlerrate für kleine k stark? Wie hängen die Schwankungen mit dem Wert von k zusammen?



Aufgabe 2 | Lösung des Optimierungsproblems mit Hilfe von Python

- Wie sind wir vorgegangen, um das optimale k zu bestimmen?

Aufgabe 2 | Lösung des Optimierungsproblems mit Hilfe von Python

- Wie sind wir vorgegangen, um das optimale k zu bestimmen?

1. Leere Liste erstellen, in der nach und nach die Fehlerraten gespeichert werden sollen.
   `errorRates = []`
2. For-Schleife zur Berechnung der Fehlerraten und Abspeichern der Fehlerraten in der Liste errorRates:
   ```
   for i in range(1, 41):
       errorRate_i = computeErrorRate(i)
       errorRates.append(errorRate_i)
   ```
3. Suchen des kleinsten Werts in der Liste errorRates
   `min_errorRates = min(errorRates)`
4. Bestimmen der Position des kleinsten Werts in der Liste errorRates
   `pos_min = errorRates.index(min_errorRates)`
5. Bestimmen des optimalen k, für das die Fehlerrate am kleinsten ist
   `k = pos_min + 1`

Aufgabe 3 | Klassifikation mit optimalem k

	Klassifiziert als 1 (Sitzen)	Klassifiziert als 2 (Stehen)	Klassifiziert als 3 (Gehen)	Klassifiziert als 4 (Laufen)	Klassifiziert als 5 (Treppen auf- und absteigen)
Tatsächlich 1 (Sitzen)	123	0	0	0	0
Tatsächlich 2 (Stehen)	0	115	0	0	0
Tatsächlich 3 (Gehen)	0	0	106	2	0
Tatsächlich 4 (Laufen)	0	0	2	133	0
Tatsächlich 5 (Treppen auf- und absteigen)	0	0	10	1	108

Aufgabe 3 | Klassifikation mit optimalem k

- Genauigkeit:
  $\texttt{accuracy} \approx 0.98$
- Fehlerrate:
  $\texttt{errorRate} \approx 0.02$
- Präzision:
  $\texttt{precisionSitzen} = 1.0$
  $\texttt{precisionStehen} = 1.0$
  $\texttt{precisionGehen} \approx 0.90$
  $\texttt{precisionLaufen} \approx 0.98$
  $\texttt{precisionTreppen} \approx 1.0$

Aufgabe 3 | Klassifikation mit optimalem k

- Was fällt euch auf, wenn ihr die Qualitätsmaße mit den Qualitätsmaßen der Klassifikation auf Arbeitsblatt 5 vergleicht?

→ Ergebnisse werden nochmal ein bisschen besser

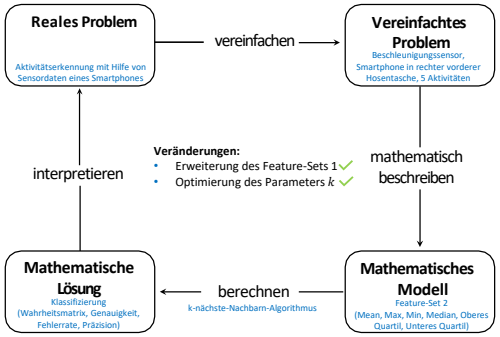
- Bei welcher Aktivität treten noch am häufigsten falsche Klassifikationen auf?

→ Aktivität *Gehen*

- Wie viele von 100 Aktivitäten werden im Mittel richtig klassifiziert?

→ ca. 98 Aktivitäten

Modellierungskreislauf



Jetzt seid ihr dran!

Öffnet **Arbeitsblatt 7** und macht euch Gedanken über Probleme und Anwendungen der Aktivitätserkennung!

- Bearbeitet die Arbeitsblätter!
- Lest die Aufgabenstellung sorgfältig!
- Teamwork!
- Helft euch gegenseitig!
- Nutzt die Tipps!
- Nutzt das Internet!
- Fragt die Betreuer!

A.14. Notizen Plenumsdiskussion 6

Diskussion nach Arbeitsblatt 6 | Optimierung des Parameters k

Folie 54: Aktivitätserkennung | Optimierung des Parameters k

Auf Arbeitsblatt 6 habt ihr die optimale Anzahl k der Nachbarn gefunden, sodass die Fehlerrate minimal wird. Wir werden nun die Ergebnisse und Erkenntnisse gemeinsam im Plenum besprechen.

Folie 55: Aufgabe 1 | Graphische Lösung des Optimierungsproblems

In der ersten Aufgabe habt ihr das Optimierungsproblem graphisch gelöst.

Fragen:
- Für welchen Wert von k ist die Fehlerrate am kleinsten?
- Warum schwankt die Fehlerrate für kleine k stark? Wie hängen die Schwankungen mit dem Wert von k zusammen?

Antworten:
- Für $k = 1$ ist die Fehlerrate am kleinsten.
- Für kleine und gerade k ist die Wahrscheinlichkeit, dass ein „Unentschieden“ beim Mehrheitsentscheid auftritt, am größten. Daher sind die Schwankungen für kleine k stärker als für große k.
- Was passiert bei einem „Unentschieden“?
  - Bei einem „Unentschieden“ wird anhand der Datenpunkte im Trainingsdatensatz entschieden.
  - Die Aktivität, zu der mehr Datenpunkte im Trainingsdatensatz gehören, gewinnt den Mehrheitsentscheid.

Folie 56: Aufgabe 2 | Lösung des Optimierungsproblems mit Hilfe von Python

Frage:
- Wie sind wir vorgegangen, um das optimale k zu bestimmen?

Antwort:
- Antworten der Schüler:innen sammeln.

Folie 57: Aufgabe 2 | Lösung des Optimierungsproblems mit Hilfe von Python

Wir sind wie folgt vorgegangen, um das optimale k zu finden:
1. Wir haben eine leere Liste erstellt, in der nach und nach die Fehlerraten abgespeichert werden.
2. Anschließend haben wir über eine for-Schleife die Fehlerrate für alle k zwischen 1 und 40 bestimmt und diese in der zuvor erstellten Liste abgespeichert.
3. Danach haben wir den kleinsten Wert in der Liste gesucht.
4. Im Anschluss haben wir die Position des kleinsten Werts in der Liste bestimmt.
5. Im letzten Schritt haben wir das optimale k bestimmt, indem wir zur Position des kleinsten Wertes 1 addiert haben. Wir müssen 1 addieren, da Python bei der Nummerierung von Listeneinträgen bei 0 beginnt.

Als optimale Anzahl der Nachbarn erhalten wir dann 1, es wird also immer nur der nächste Datenpunkt bei der Klassifikation beachtet.

Folie 58: Aufgabe 3 | Klassifikation mit optimalem k

Bei der Klassifikation mit Feature-Set 2 und $k = 1$ solltet ihr folgende Wahrheitsmatrix als Ergebnis erhalten haben. Gleicht die Ergebnisse bitte mit eurer ausgefüllten Tabelle auf dem Antwortblatt ab.

Folie 59: Aufgabe 3 | Klassifikation mit optimalem k

Frage:
- Welche Werte haben nun die verschiedenen Qualitätsmaße?

klicken

Antwort:
- $\text{accuracy} \approx 0.98$
- $\text{error rate} \approx 0.02$
- $\text{precisionSitzen} = 1.0$
- $\text{precisionStehen} = 1.0$
- $\text{precisionGehen} \approx 0.90$
- $\text{precisionLaufen} \approx 0.98$
- $\text{precisionTreppen} \approx 1.0$

Folie 60: Aufgabe 3 | Klassifikation mit optimalem k

Frage:
- Was fällt euch auf, wenn ihr die Qualitätsmaße mit den Qualitätsmaßen der Klassifikation auf Arbeitsblatt 5 vergleicht?

klicken

Antwort:
- Die Ergebnisse werden nochmal ein bissen besser.

klicken

Frage:
- Bei welcher Aktivität treten noch am häufigsten falsche Klassifikationen auf?

klicken

Antwort:
- Bei Aktivität 3 (Gehen) treten noch am häufigsten falsche Klassifikationen auf (geringste Präzision).

klicken

Frage:
- Wie viele von 100 Aktivitäten werden im Mittel richtig klassifiziert?

klicken

Antwort:
- Im Mittel werden ca. 98 Aktivitäten von 100 richtig klassifiziert.

Folie 61: Modellierungskreislauf

Wir haben unsere Ergebnisse durch die beiden Veränderungen (Erweiterung des Feature-Sets 1 und Optimierung des Parameters k) deutlich verbessern können und haben somit ein zufriedenstellendes Ergebnis für unser reales Problem, die Aktivitätserkennung mit Hilfe von Sensordaten eines Smartphones, gefunden.

Wir werden uns jetzt auf dem nächsten Arbeitsblatt noch Probleme sowie Anwendungen der Aktivitätserkennung anschauen.

Folie 62: Jetzt seid ihr dran!

Öffnet dazu Arbeitsblatt 7!

A.15. Plenumsdiskussion 7 und Abschlusspräsentation

Aktivitätserkennung | Probleme und Schwierigkeiten sowie Anwendungen und Einsatzgebiete der Aktivitätserkennung

Diskussion nach Arbeitsblatt 7

Aufgabe 1 | Herausforderungen und Schwierigkeiten

- Handy in der Hand des Nutzers:

	Klassifiziert als 1 (Sitzen)	Klassifiziert als 2 (Stehen)	Klassifiziert als 3 (Gehen)	Klassifiziert als 4 (Laufen)	Klassifiziert als 5 (Treppen auf- und absteigen)
Tatsächlich 1 (Sitzen)	59	0	0	0	0
Tatsächlich 2 (Stehen)	60	0	0	0	0
Tatsächlich 3 (Gehen)	60	0	0	0	0
Tatsächlich 4 (Laufen)	11	0	29	2	18
Tatsächlich 5 (Treppen auf- und absteigen)	60	0	0	0	0

Aufgabe 1 | Herausforderungen und Schwierigkeiten

- Handy in der Hand des Nutzers:
  - Genauigkeit:
    $accuracy \approx 0.20$
  - Fehlerrate:
    $errorRate \approx 0.80$
  - Präzision:
    $precisionSitzen \approx 0.24$
    $precisionStehen = nan$
    $precisionGehen = 0.0$
    $precisionLaufen = 1.0$
    $precisionTreppen = 0.0$

Aufgabe 1 | Herausforderungen und Schwierigkeiten

- Handy im Rucksack des Nutzers:

	Klassifiziert als 1 (Sitzen)	Klassifiziert als 2 (Stehen)	Klassifiziert als 3 (Gehen)	Klassifiziert als 4 (Laufen)	Klassifiziert als 5 (Treppen auf- und absteigen)
Tatsächlich 1 (Sitzen)	60	0	0	0	0
Tatsächlich 2 (Stehen)	0	60	0	0	0
Tatsächlich 3 (Gehen)	0	0	0	0	60
Tatsächlich 4 (Laufen)	0	0	0	0	60
Tatsächlich 5 (Treppen auf- und absteigen)	0	0	26	0	34

Aufgabe 1 | Herausforderungen und Schwierigkeiten

- Handy im Rucksack des Nutzers:
  - Genauigkeit:
    $accuracy \approx 0.51$
  - Fehlerrate:
    $errorRate \approx 0.49$
  - Präzision:
    $precisionSitzen = 1.0$
    $precisionStehen = 1.0$
    $precisionGehen = 0.0$
    $precisionLaufen = nan$
    $precisionTreppen \approx 0.22$

Aufgabe 1 | Herausforderungen und Schwierigkeiten

- Was könnten Gründe dafür sein, dass unser Algorithmus die Sensordaten, bei denen sich das Handy in der Hand bzw. im Rucksack des Nutzers befand, nicht richtig klassifiziert?

Aufgabe 1 | Herausforderungen und Schwierigkeiten

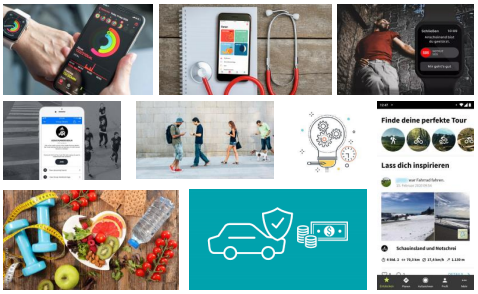
- Was sind weitere Herausforderungen und Schwierigkeiten der Aktivitätserkennung?
- Wie könnte man diese Schwierigkeiten möglicherweise umgehen?

Aufgabe 1 | Herausforderungen und Schwierigkeiten

Aufgabe 2 | Anwendungen und Einsatzgebiete

- In welchen Bereichen kann die menschliche Aktivitätserkennung von Nutzen sein?

Aufgabe 2 | Anwendungen und Einsatzgebiete



Aufgabe 2 | Anwendungen und Einsatzgebiete

- In welchen Bereichen könnte die menschliche Aktivitätserkennung eventuell auch problematisch sein bzw. missbraucht werden?

Rückblick | Die Schritte des Workshops

Ziel: Entwicklung eines eigenes Klassifikationsalgorithmus!

1. Erkunden des Datensatzes
2. Vorverarbeitung der Daten
3. Schrittweise Entwicklung eines eigenen Klassifikationsalgorithmus
4. Bewertung der Ergebnisse des Klassifikationsalgorithmus mit Hilfe verschiedener Qualitätsmaße
5. Verbesserung des entwickelten Klassifikationsalgorithmus
6. Diskussion von Problemen sowie Anwendungsgebieten der menschlichen Aktivitätserkennung

A.16. Notizen Plenumsdiskussion 7 und Abschlusspräsentation

Diskussion nach Arbeitsblatt 7 | Probleme und Schwierigkeiten sowie Anwendungen und Einsatzgebiete der Aktivitätserkennung

---

Folie 63: Aktivitätserkennung | Probleme und Schwierigkeiten sowie Anwendungen und Einsatzgebiete der Aktivitätserkennung

Auf Arbeitsblatt 7 habt ihr euch zum einen mit Problemen und Schwierigkeiten der Aktivitätserkennung auseinandergesetzt. Zum anderen habt ihr euch auch Gedanken zu Anwendungen und Einsatzgebieten der Aktivitätserkennung gemacht. Wir werden eure Ergebnisse nun gemeinsam im Plenum besprechen.

Folie 64: Aufgabe 1 | Herausforderungen und Schwierigkeiten

In der ersten Aufgabe habt ihr euch mit Herausforderungen und Schwierigkeiten der Aktivitätserkennung beschäftigt. Eine dieser Herausforderungen stellt die Position und Ausrichtung der Sensoren am Körper des Benutzers da.

klicken

Ihr habt Sensordaten klassifiziert, die aufgenommen wurden, während sich das Handy in der Hand des Nutzers befand. Bei der Klassifikation solltet ihr folgende Wahrheitsmatrix erhalten haben. Gleicht die Ergebnisse bitte mit eurer ausgefüllten Tabelle auf dem Antwortblatt ab.

Folie 65: Aufgabe 1 | Herausforderungen und Schwierigkeiten

Frage:

- Welche Werte haben nun die verschiedenen Qualitätsmaße (Handy in der Hand des Nutzers)?

klicken

Antwort:

- $\text{accuracy} \approx 0.20$
- $\text{error rate} \approx 0.80$
- $\text{precisionSitzen} \approx 0.24$
- $\text{precisionStehen} = \text{nan}$
- $\text{precisionGehen} = 0.0$
- $\text{precisionLaufen} = 1.0$
- $\text{precisionTreppen} = 0.0$

Folie 66: Aufgabe 1 | Herausforderungen und Schwierigkeiten

Anschließend habt ihr Sensordaten klassifiziert, die aufgenommen wurden, während sich das Handy im Rucksack des Nutzers befand. Bei der Klassifikation solltet ihr folgende Wahrheitsmatrix als Ergebnis erhalten haben. Gleicht die Ergebnisse bitte mit eurer ausgefüllten Tabelle auf dem Antwortblatt ab.

Folie 67: Aufgabe 1 | Herausforderungen und Schwierigkeiten

Frage:

- Welche Werte haben nun die verschiedenen Qualitätsmaße (Handy im Rucksack des Nutzers)?

klicken

Antwort:

- accuracy ≈ 0.51
- error rate ≈ 0.49
- precisionSitzen $= 1.0$
- precisionStehen $= 1.0$
- precisionGehen $= 0.0$
- precisionLaufen $=$ nan
- precisionTreppen ≈ 0.22

Folie 68: Aufgabe 1 | Herausforderungen und Schwierigkeiten

Frage:

- Was könnten Gründe dafür sein, dass unser Algorithmus die Sensordaten, bei denen sich das Handy in der Hand bzw. im Rucksack des Nutzers befand, nicht richtig klassifiziert?

Antwort:

- Antworten der SuS sammeln!
- Die charakteristischen Ausschläge der Beschleunigungswerte einer bestimmten Achse sind je nach Orientierung des Smartphones nun auf einer anderen Achse zu finden.
- z. B. bei Aktivität Stehen:
  - Befindet sich das Handy in der Hosentasche wirkt die Erdbeschleunigung auf die y-Achse.
  - Befindet sich das Handy jetzt aber in der Hand des Nutzers, während dieser z. B. eine Chat-Nachricht liest, wirkt die Erdbeschleunigung auf die z-Achse.

Folie 69: Aufgabe 1 | Herausforderungen und Schwierigkeiten

Fragen:

- Was sind weitere Herausforderungen und Schwierigkeiten der Aktivitätserkennung?
- Wie könnte man diese Schwierigkeiten möglicherweise umgehen?

Antwort:

- Antworten der SuS sammeln!

Folie 70: Aufgabe 1 | Herausforderungen und Schwierigkeiten

- Subjektive Empfindlichkeit:
  - Genauigkeit der Aktivitätserkennung ist stark von den Testpersonen abhängig.
  - Jeder Mensch hat unterschiedliche Gewohnheiten und Verhaltensweisen. Verschiedene Personen haben auch unterschiedliche Bewegungsmuster.
  - Lösungsmöglichkeit:
    So viele Daten wie möglich von so vielen unterschiedlichen Testpersonen wie möglich aufnehmen.
- Standortempfindlichkeit:
  - Sensordaten sind stark von der Ausrichtung und Position der Sensoren am Körper des Nutzers abhängig.
  - Lösungsmöglichkeit:
    Weitere Sensoren miteinbeziehen, sodass Ausrichtung und Position der Sensoren vorab bestimmt werden kann bzw. die Sensordaten in Erdkoordinaten umgerechnet werden können.
- Komplexität der Aktivitäten:
  - Werden mehrere Aktivitäten gleichzeitig ausgeführt, sind diese schwer zu unterscheiden bzw. zu erkennen.
  - Beim Übergang zwischen zwei Aktivitäten kann es auch schnell zu Fehlklassifikationen kommen.
- Energie- und Ressourcenbeschränkung:
  - Einsatz vieler Sensoren wirkt sich zum einen auf die Akkulaufzeit und zum anderen auf den Speicherplatz aus.
  - Ausführung des Klassifikationsalgorithmus auf dem Smartphone kann aufgrund der geringeren Rechenressourcen im Vergleich zu einem Laptop zu Problemen führen.

- Lösungsmöglichkeit:
  Auslagern der Klassifikation auf einen Online-Server.

Folie 71: Aufgabe 2 | Anwendungen und Einsatzgebiete

Fragen:
- In welchen Bereichen kann die menschliche Aktivitätserkennung von Nutzen sein?

Antwort:
- Antworten der SuS sammeln!

Folie 72: Aufgabe 2 | Anwendungen und Einsatzgebiete

- Anwendungen für den Endbenutzer
  - Fitness-Tracking
    - Schrittzähler, täglicher Kalorienverbrauch, zurückgelegte Strecke, gestiegene Treppenstufen, etc.
  - Gesundheitsüberwachung:
    - Kontinuierliche Überwachung der Gewohnheiten und täglichen Aktivitäten von Patienten ermöglicht eine bessere Diagnose durch den Arzt.
    - Physiotherapeuten können Therapie auf die Gewohnheiten der Patienten abstimmen.
  - Sturzerkennung:
    - Ein automatischer Notruf und die Ortung des Smartphones können unmittelbar nach dem Sturz durchgeführt werden.
  - Kontextbezogenes Verhalten:
    - Smartphone vergrößert z. B. die Schrift, wenn Nutzer beim Lesen einer Chat-Nachricht gerade geht.
  - Smartphones sind auch eine geeignete Plattform, um Nutzer zu einer gesünderen Lebensweise zu motivieren.

- Anwendungen für Unternehmen und Dritte
  - Gezielte Werbung
    - Werbung wird auf die täglichen Aktivitäten und Gewohnheiten der Nutzer abgestimmt und wirkt dadurch weniger aufdringlich, da sie für die Aktivitäten und Interessen der Nutzer relevant ist.
  - Forschungsplattformen:
    - Forscher in vielen Bereichen, vom Marketing bis zum Gesundheitswesen, werden immer daran interessiert sein, Aktivitätsdaten zu sammeln.
    - Forschungsunternehmen benötigen häufig zuerst eine ausreichende Menge an Daten. Es besteht also ein Markt für aufgenommene Rohdaten.
  - Unternehmensführung:
    - Kann bei der Mitarbeiterverwaltung und bei der Abrechnung der Arbeitszeiten helfen.
  - Versicherungen:
    - Manche Versicherungsgesellschaften bieten einen Rabatt bei der Autoversicherung an, wenn eine sichere Fahrweise erkannt wird.

- Anwendungen für Gruppen und Crowds
  - Aktivitätsbasierte soziale Netzwerke
    - Anwendung kann einen Freundeskreis vorschlagen, der ähnliche Aktivitätsmuster aufweist.
  - Aktivitätsbezogenes Crowd-Sourcing
    - Es können Gegenden identifiziert werden, in denen bestimmte Aktivitäten z. B. Radfahren häufig ausgeführt werden.
    - Einem Nutzer, der gerne Fahrrad fährt, können dann beliebte Fahrradstrecken vorgeschlagen werden.
    - System kann auch dazu genutzt werden, um zu erkennen, ob sich die Aktivitäten in einer bestimmten Region stark von den normalerweise beobachteten Aktivitäten unterscheiden. Dadurch können möglicherweise Unfälle oder Katastrophen schneller erkannt werden.

Folie 73: Aufgabe 2 | Anwendungen und Einsatzgebiete

Fragen:
- In welchen Bereichen könnte die menschliche Aktivitätserkennung eventuell auch problematisch sein bzw. missbraucht werden?

Antwort:
- Antworten der SuS sammeln!
- Mögliche problematische Bereiche:
  - Datenschutz bzw. Privatsphäre: An welche Server werden die personenbezogenen Daten weitergegeben?
  - Überwachung durch einen totalitären Staat (z. B. Social Scoring in China)

Folie 74: Rückblick | Die Schritte des Workshops (Mit ZusammenfassungsAB besprechen)

Zum Abschluss des Workshops möchten wir nochmal auf die einzelnen Schritte bei der Entwicklung des Klassifikationsalgorithmus zurückblicken:

klicken

1. Erkunden des Datensatzes
   - Datensatz besteht aus der UserID und ActivityID sowie der Zeit und den Beschleunigungswerten in allen drei Raumdimensionen.
   - Daten zu 5 Aktivitäten (Sitzen, Stehen, Gehen, Laufen, Treppen auf- und absteigen) wurden aufgenommen.
   - 10 Testpersonen haben Daten aufgenommen.
   - Jede Aktivität wurde bei der Datenaufnahme 3 Minuten ausgeführt und es wurde eine Abtastrate von 40 Hz gewählt.

klicken

2. Vorverarbeitung der Daten
   - Ergänzung der Gesamtbeschleunigung des Smartphones (Betrags des Beschleunigungsvektors) zu unserem Datensatz.
   - Einteilung der Rohdaten in kleinere Zeitfenster (Windows).
   - Berechnung erster Features (Mittelwert, Maximum und Minimum des Betrags des Beschleunigungsvektors) über die Daten eines Windows.

klicken

3. Schrittweise Entwicklung eines eigenen Klassifikationsalgorithmus
   - Definition der Abstandsfunktion zur Berechnung des Abstands zwischen den Datenpunkten zweier Windows.
   - Entwicklung einer Suchfunktion, um die k nächsten Nachbarn eines ausgewählten Windows zu bestimmen.
   - Mehrheitsentscheid unter den k nächsten Nachbarn, um die Klasse bzw. Aktivität des ausgewählten Windows zu bestimmen.

klicken

4. Bewertung der Ergebnisse des Klassifikationsalgorithmus mit Hilfe verschiedener Qualitätsmaße
   - Aufteilung der Daten in Trainings- und Testdaten.
   - Beurteilung der Ergebnisse der Klassifikation mit Hilfe der Wahrheitsmatrix.
   - Berechnung verschiedener Qualitätsmaße (Genauigkeit, Fehlerrate und Präzision) zur zahlenmäßigen Bewertung unserer Klassifikationsergebnisse.
   - Ergebnisse noch nicht 100 % zufriedenstellend.

klicken

5. Verbesserung des entwickelten Klassifikationsalgorithmus
   - Erweiterung des Feature-Sets 1 um den Mittelwert, das Maximum und das Minimum der Beschleunigungswerte aller drei Raumdimensionen.
   - Erweiterung des Feature-Sets 1 um weitere Features (Median, oberes Quartil und unteres Quartil des Betrags des Beschleunigungsvektors).
   - Optimierung der Anzahl k der nächsten Nachbarn mit dem Ergebnis, dass nur der nächste Nachbar ($k = 1$) bei der Klassifikation berücksichtigt werden sollte.
   - Durch die Veränderungen wurden die Ergebnisse der Klassifikation deutlich verbessert.

klicken

6. Diskussion von Problemen sowie Anwendungsgebieten der menschlichen Aktivitätserkennung
   - Kennenlernen eines großen Problems der Aktivitätserkennung durch die Klassifikation von Sensordaten, die aufgenommen wurden, während sich das Handy entweder in der Hand oder im Rucksack des Nutzers befand.
   - Diskussion weiterer Probleme und Schwierigkeiten der Aktivitätserkennung.
   - Diskussion verschiedener Anwendungen und Einsatzgebiete der Aktivitätserkennung.

B. Arbeitsblätter mit Lösungen

B.1. Arbeitsblatt 1

Arbeitsblatt 1: Erkunden des Datensatzes

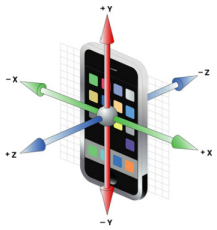
Notiere deine Ergebnisse während dem Bearbeiten des Arbeitsblattes auf diesem Antwortblatt.

Einleitung:

Mobile Geräte, wie zum Beispiel **Smartphones**, sind mittlerweile ständige Begleiter in unserem Alltag. Weltweit nutzen rund 3,9 Milliarden Menschen ein Smartphone. Die schnelle Entwicklung der in den Smartphones eingebauten **Sensoren** bietet die Grundlage für zahlreiche Anwendungsgebiete, wie zum Beispiel die **menschliche Aktivitätserkennung**.

Ziel der Aktivitätserkennung ist es, die **Aktivität** oder **Situation**, in der sich ein Nutzer befindet, über Sensoren am Smartphone zu bestimmen und gegebenenfalls darauf zu reagieren. Für den Nutzer können sich daraus große Vorteile ergeben, die ihm unter Umständen den Alltag bedeutend erleichtern. Beispielweise könnte das Smartphone die Schrift vergrößern, wenn der Nutzer versucht im Gehen Chat-Nachrichten zu lesen. Neben den Nutzern kann die Aktivitätserkennung auch für Unternehmen zum Beispiel bei der Personalisierung von Werbung von Nutzen sein.

In diesem Workshop werden wir ein Modell entwickeln, mit dem verschiedene Aktivitäten anhand von Sensordaten des Smartphones erkannt werden können. Dazu wurden im Vorfeld sogenannte **Rohdaten** aufgenommen, für die wir im Laufe des Workshops ein maschinelles Lernverfahren entwickeln werden, um die ausgeführte Aktivität zu bestimmen. Zur Aufnahme der Rohdaten wurde der Beschleunigungsmesser des Smartphones (auch **Accelerometer** genannt) genutzt, der die Beschleunigung in allen drei Raumdimensionen in der Einheit $\frac{m}{s^2}$ aufzeichnet (siehe Abbildung). Während der Datenaufnahme befand sich das Smartphone immer in der **rechten vorderen Hosentasche**.



Auf diesem Arbeitsblatt kannst du dir einen Überblick über die aufgenommenen **Rohdaten** verschaffen.

Beantworte während der Bearbeitung der einzelnen Aufgaben die folgenden Fragen auf deinem Antwortblatt:

1. Aus welchen Daten besteht ein Datenpunkt im Datensatz (= Zeile in der Tabelle)?
2. Zu wie vielen Aktivitäten wurden Daten aufgenommen?
3. Welche Aktivitäten sind dies?
4. Wie viele Testpersonen haben Daten aufgenommen?
5. Wie viele Testpersonen sind weiblich und wie viele sind männlich?
6. Wie alt ist die jüngste bzw. älteste Testperson?
7. Wie lange wurden Daten zu den jeweiligen Aktivitäten von den Testpersonen aufgenommen?
8. Mit welcher Abtastrate wurden die Daten aufgenommen?

Aufgabe 1 | Die Aktivitäten

In dieser Aufgabe lernst du sowohl den Datensatz, in dem die Rohdaten gespeichert sind, als auch die verschiedenen Aktivitäten, für die Rohdaten aufgenommen wurden, kennen. Damit wir später leichter mit den Daten arbeiten und diese besser einer bestimmten Aktivität zuordnen können, wird jeder Aktivität eine natürliche Zahl zugeordnet, die sogenannte **ActivityID**.

Teil a | Der Datensatz und die Anzahl der Aktivitäten

Führe den folgenden Code aus. Klicke dazu in das Codefeld und drücke auf den ▶-Button:

Dir wird eine Tabelle mit den ersten und letzten fünf Zeilen des Datensatzes, in dem die Rohdaten gespeichert sind, ausgegeben. Du kannst die Tabelle nach einer beliebigen Spalte sortieren. Lege fest, nach welcher Spalte (z.B. Time (s), ActivityID, ...) sortiert werden soll. Trage dazu den Namen der Spalte für die Variable `column` *ein. Du kannst außerdem festlegen, ob die Tabelle aufsteigend (* `aufsteigend = True` *) oder absteigend (* `aufsteigend = False` *) sortiert werden soll.*

⚠ Hinweis zur Eingabe im Codefeld:

- Die Gänsefüßchen `" "` müssen um die Bezeichnungen der Spalte stehen bleiben.

In [1]:
```
# Hier nichts ändern!
import sys; sys.path.append("../code"); from setupAB1 import *;

# Sortierung nach Spalte "column"
column = "ActivityID"

# aufsteigende (=True) oder absteigende (=False) Sortierung
aufsteigend = True

# Hier nichts ändern!
data.sort_values(by = [column], ascending = aufsteigend).reset_index(drop = True)
```

Out[1]:

	UserID	ActivityID	Time (s)	x (m/s^2)	y (m/s^2)	z (m/s^2)
0	1	1	0.028140	5.480853	-4.508624	-6.869365
1	5	1	120.105365	5.624554	0.045356	-8.159083
2	5	1	120.080207	5.603448	0.041164	-8.130642
3	5	1	120.055049	5.624254	0.026794	-8.108788
4	5	1	120.029891	5.635032	0.026645	-8.147557
...	...	...	...	...	...	...
351431	6	5	123.069965	-5.208868	-9.523196	0.483944
351432	6	5	123.044808	-6.893764	-11.925401	-0.428259
351433	6	5	123.019651	-10.597213	-16.777262	-1.619183
351434	6	5	123.296378	-1.868564	-5.554799	-0.721200
351435	10	5	179.978142	3.050956	-10.346634	-3.517685

351436 rows × 6 columns

Teil b | Die Art der Aktivitäten

Ersetze das `NaN` *im Code durch eine mögliche ActitvityID. Führe anschließend den Code aus. Dir wird dann die zur eingegebenen ActivityID gehörende Aktivität ausgegeben.*

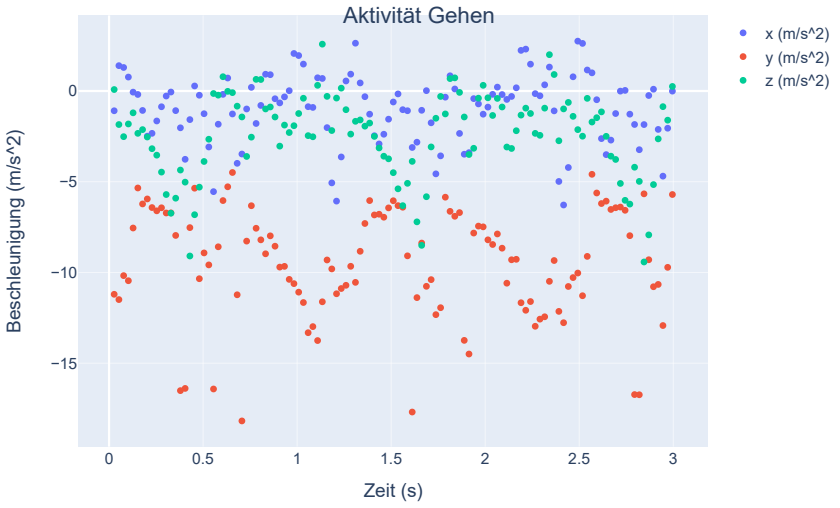
In [2]:
```
ActivityID = 3;

# Hier nichts ändern!
checkActivityID(ActivityID)
```

```
Die ActivityID 3 steht für die Aktivität Gehen.
```

Teil c | Graphische Darstellung der unterschiedlichen Aktivitäten

Ersetze das `NaN` *durch eine mögliche* `ActivityID` *und führe anschließend den Code aus. Dir wird ein Graph, in dem die Beschleunigungswerte jeder der drei Raumdimesionen gegen die Zeit aufgetragen sind, für die gewählte Aktivität angezeigt.*

In [3]:
```
ActivityID = 3;

# Hier nichts ändern!
plotActivityID(ActivityID)
```



Teil d | Interpretation der graphischen Darstellung

Schaue dir nacheinander nochmal die Graphen zu den einzelnen Aktivitäten an. Ändere dazu im obigen Codefeld die ActivityID. Beantworte dann die folgenden Fragen:

1. Welche Unterschiede sind zwischen den einzelnen Aktivitäten zu erkennen?
2. Bei welcher Aktivität sind am wenigsten / am meisten Ausschläge zu erkennen und warum?
3. Auf welcher der drei Achsen sind die größten Ausschläge zu erkennen und warum?

Notiere deine Überlegungen auf deinem Antwortblatt.

Aufgabe 2 | Die Testpersonen

In dieser Aufgabe erhälst du Informationen über die Testpersonen, die Rohdaten zu den einzelnen Aktivitäten aufgenommen haben. Auch den Testpersonen wird wie den Aktivitäten eine natürliche Zahl zugeordnet, die sogenannte **UserID**.

Führe den nachfolgenden Code aus. Dir wird eine Tabelle ausgegeben, in der das Alter und das Geschlecht der Testpersonen eingetragen ist.

In [4]:
```
# Hier nichts ändern!
display(HTML(user.to_html(index=False)))
```

UserID	Geschlecht	Alter
1	w	23
2	m	23
3	w	25
4	m	18
5	m	20
6	w	20
7	w	25
8	m	27
9	w	52
10	m	52

Aufgabe 3 | Die Datenaufnahme

In dieser Aufgabe erfährst du, wie lange die Testpersonen die Aktivitäten zur Aufnahmen der Daten ausgeführt haben und mit welcher **Abtastrate** die Daten aufgenommen wurden. Dazu schauen wir uns für **Testperson 1** die Daten zur **Aktivität 1** genauer an.

Teil a | Die Daten zur Aktivität 1 von Testperson 1

Führe den folgenden Code aus. Dir wird eine Tabelle mit den ersten 5 und den letzten 5 Datenpunkten zur Aktivität 1 von Testperson 1 ausgegeben.

In [5]:
```
# Hier nichts ändern!
showDataActivity1User1()
```

Die ersten 5 Zeilen der Daten zu Aktivität 1 von Testperson 1:

UserID	ActivityID	Time (s)	x (m/s^2)	y (m/s^2)	z (m/s^2)
1	1	0.028140	5.480853	-4.508624	-6.869365
1	1	0.053292	5.488187	-4.504134	-6.870263
1	1	0.078443	5.494324	-4.496799	-6.872658
1	1	0.103595	5.492229	-4.508475	-6.871311
1	1	0.128747	5.472320	-4.499044	-6.856192

Die letzten 5 Zeilen der Daten zu Aktivität 1 von Testperson 1:

UserID	ActivityID	Time (s)	x (m/s^2)	y (m/s^2)	z (m/s^2)
1	1	179.843615	5.377268	-4.423451	-6.996002
1	1	179.868768	5.409301	-4.418212	-6.991062
1	1	179.893921	5.422624	-4.475094	-7.008276
1	1	179.919075	5.384004	-4.435426	-6.995104
1	1	179.944228	5.397027	-4.384532	-6.978638

Insgesamt bestehen die Daten zu Aktivität 1 von Testperson 1 aus 7150 Zeilen und 6 Spalten.

Teil b | Die Dauer und Abtastrate der Datenaufnahme

Mithilfe der obigen Tabelle kannst du herausfinden, wie lange die Testpersonen die Aktivitäten jeweils ausgeführt haben. Außerdem kannst du die Abtastrate, mit der die Daten aufgenommen wurden, bestimmen.

▶ **Erläuterung Abtastrate** (Falls du den Begriff Abtastrate nicht kennst, kannst du hier klicken.)

Ersetze das erste `NaN` *durch die Dauer der Datenaufnahme und das zweite* `NaN` *durch die Abtastrate, mit der die Daten aufgenommen wurden. Runde deine Lösungen auf ganze Zahlen. Führe anschließend den Code aus.*

Falls du Hilfe benötigst, kannst du dir hier einen Tipp ansehen.

In [6]:
```
samplingDuration = 180;
samplingRate = 40;

# Hier nichts ändern!
checkDurationRate(samplingDuration, samplingRate)
```

✓ Deine Lösungen für die Dauer der Datenaufnahme und die Abtastrate sind korrekt.

Fazit

Auf diesem Arbeitsblatt hast du den Datensatz, mit dem wir im Laufe des Workshops arbeiten werden, kennengelernt.

Diskussion im Plenum

Wir werden nun die Erkenntnisse und Ergebnisse von Arbeitsblatt 1 gemeinsam im Plenum besprechen.

Öffne nun das nächste Arbeitsblatt.

Schon fertig?

Bearbeite die folgende Zusatzaufgabe.

Zusatzaufgabe | Weitere Sensoren

Recherchiere im Internet weitere Sensoren, die in Smartphones eingebaut sind. Überlege dir, welche Informationen diese Sensoren mit Blick auf die Aktivitätserkennung liefern können.

B.2. Arbeitsblatt 2

Arbeitsblatt 2: Vorverarbeitung der Daten

Notiere deine Ergebnisse während dem Bearbeiten des Arbeitsblattes auf diesem Antwortblatt.

Einleitung:

Für die aufgenommenen Rohdaten des Beschleunigungsmessers können wir nicht direkt ein Lernverfahren aus dem Bereich des maschinellen Lernens entwickeln. Zuvor müssen wir die Daten noch aufbereiten. Dieser Prozess wird auch als **Data Preprocessing** bezeichnet.

Zur Aufbereitung der Daten werden wir in einem ersten Schritt die Beschleunigungswerte der drei Achsen kombinieren. Anschließend werden wir die große Menge an Rohdaten auf eine kleinere Menge von möglichst aussagekräftigen Werten (sogenannte **Features**) reduzieren, die die Eigenschaften des Signals bestmöglich abbilden. Diese Features werden wir jedoch nicht über die Gesamtmenge der Rohdaten berechnen. Stattdessen werden wir die Daten zuvor in Zeitfenster einer festen Länge (auch **Windows** genannt) unterteilen. Anschließend können wir die Features über diese Windows berechnen.

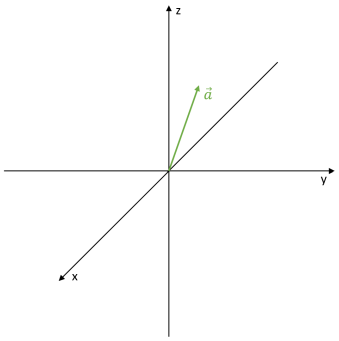
Auf diesem Arbeitsblatt lernst du den Prozess der Datenvorverarbeitung kennen.

Aufgabe 1 | Der Betrag des Beschleunigungsvektors

Jeder Datenpunkt besteht aus den Beschleunigungswerten in den drei Raumdimensionen x, y und z. Wir können uns jeden Datenpunkt deshalb auch als einen dreidimensionalen Vektor vorstellen. Dieser **Beschleunigungsvektor**

$$\vec{a} = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix}$$

beschreibt die Gesamtbeschleunigung des Smartphones.



Teil a | Berechnung des Betrags

Um den Wert der Gesamtbeschleunigung bestimmen zu können, müssen wir den Betrag des Beschleunigungsvektors berechnen. Diesen Wert bezeichnen wir als **Vector Length**.

Ersetze das `NaN` *durch eine Formel, mit der du den Betrag des Beschleunigungsvektors berechnen kannst. Führe anschließend den Code aus.*

⚠ Hinweise zur Eingabe im Codefeld:

- Du kannst die x-Komponente des Vektors $\vec{a}$ mit `a_x`, die y-Komponente mit `a_y` und die z-Komponente mit `a_z` im Code eingeben.
- Wurzeln $\sqrt{x}$ werden im Code mit `sqrt(x)` eingegeben.
- Exponenten wie z.B. x^2 werden im Code mit `x**2` eingegeben.

Falls du Hilfe benötigst, kannst du dir hier einen Tipp ansehen.

In [1]:
```
# Hier nichts ändern!
import sys; sys.path.append("../code"); from setupAB2 import *;

vectorLength = sqrt(a_x**2 + a_y**2 + a_z**2);

# Hier nichts ändern!
checkVectorLength(vectorLength)
```

```
✓ Deine Formel für den Betrag des Beschleunigungsvektors ist korrekt.
```

Teil b | Ergänzung im Datensatz

Damit wir die Gesamtbeschleunigung des Smartphones bei der späteren Auswertung der Daten nutzen können, ergänzen wir eine zusätzliche Spalte in unserem Datensatz. In dieser Spalte tragen wir den Betrag des Beschleunigungsvektors für jeden Datenpunkt ein.

Führe den Code aus. Dir werden die ersten zehn Zeilen des Datensatzes mit der zusätzlichen Spalte für den Betrag des Beschleunigungsvektors ausgegeben.

In [2]:
```
# Hier nichts ändern!
display(HTML(dataVectorLength.head(10).to_html(index=False)))
```

UserID	ActivityID	Time (s)	x (m/s^2)	y (m/s^2)	z (m/s^2)	Vector Length (m/s^2)
1	1	0.028140	5.480853	-4.508624	-6.869365	8.408040
1	1	0.053292	5.488187	-4.504134	-6.870263	8.408010
1	1	0.078443	5.494324	-4.496799	-6.872658	8.404166
1	1	0.103595	5.492229	-4.508475	-6.871311	8.415299
1	1	0.128747	5.472320	-4.499044	-6.856192	8.392204
1	1	0.153899	5.456753	-4.472849	-6.885382	8.353975
1	1	0.179050	5.481302	-4.499344	-6.910080	8.398384
1	1	0.204202	5.500462	-4.511618	-6.910679	8.424041
1	1	0.229354	5.493127	-4.491260	-6.871161	8.397457
1	1	0.254505	5.481451	-4.456682	-6.861881	8.352864

Aufgabe 2 | Einteilung der Windows

Als nächstes werden wir die große Menge an Rohdaten in kleinere Zeitfenster (**Windows**) einer festen Länge unterteilen.

Teil a | Die Länge der Windows

Bei der Wahl der Länge der Windows müssen verschiedene Faktoren berücksichtigt werden:

- Die Windows müssen mindestens so groß sein, dass mehrere charakteristische Bewegungen einer Aktivität innerhalb der Zeit eines Windows durchgeführt werden (z.B. zwei Schritte vorwärts beim Gehen).
- Die Länge des Windows sollte ein ganzzahliger Teiler von 180 sein, da wir ansonsten innerhalb eines Windows Daten zu zwei Aktivitäten vermischen würden.
- Die Windows sollten so klein wie möglich sein, da dies unsere spätere Zuordnung der Rohdaten zu einer Aktivität verbessern wird.

Ersetze das `NaN` *durch eine mögliche Länge der Windows in Sekunden und führe den Code anschließend aus.*

In [3]:
```
windowLength = 5;

# Hier nichts ändern!
checkWindowLength(windowLength)
```

✓ Deine Wahl für die Länge der Windows ist sinnvoll.

Teil b | Ergänzung im Datensatz

Der Einheitlichkeit wegen legen wir die Länge der Windows auf eine feste Zeitspanne fest (Klicke auf den Pfeil, um die gewählte Länge der Windows zu sehen).

▶ **gewählte Länge der Windows**

Um die aufgenommenen Daten den Windows optimal zuordnen zu können, nummerieren wir die Zeitfenster mit Hilfe einer sogenannten **WindowID** fortlaufend durch. Diese WindowID fügen wir als eine zusätzliche Spalte unserem Datensatz hinzu.

Führe den Code aus. Dir werden die ersten und die letzten fünf Zeilen des Datensatzes mit der zusätzlichen Spalte für die WindowID ausgegeben.

In [4]:
```
# Hier nichts ändern!
printDataWindow()
```

Die ersten 5 Zeilen des Datensatzes:

WindowID	UserID	ActivityID	Time (s)	x (m/s^2)	y (m/s^2)	y (m/s^2)	Vector Length (m/s^2)
1	1	1	0.028140	5.480853	-4.508624	-4.508624	8.408040
1	1	1	2.216343	5.487589	-4.477938	-4.477938	8.379587
1	1	1	2.191192	5.516029	-4.445905	-4.445905	8.364133
1	1	1	2.166040	5.550458	-4.380790	-4.380790	8.318066
1	1	1	2.140888	5.550907	-4.342769	-4.342769	8.278397

Die letzten 5 Zeilen des Datensatzes:

WindowID	UserID	ActivityID	Time (s)	x (m/s^2)	y (m/s^2)	y (m/s^2)	Vector Length (m/s^2)
3000	10	5	177.814545	2.127376	-6.044580	-6.044580	8.809065
3000	10	5	177.789387	2.086511	-6.294261	-6.294261	9.142700
3000	10	5	177.764229	2.523003	-6.089187	-6.089187	8.973403
3000	10	5	178.066125	0.508493	-10.040820	-10.040820	14.208965
3000	10	5	179.978142	3.050956	-10.346634	-10.346634	14.947040

Aufgabe 3 | Wahl der Features

Als nächstes werden wir verschiedene **Features** berechnen, die unsere Daten bestmöglich repräsentieren. Diese Features können wir sowohl über die Beschleunigungswerte in den drei Raumdimensionen als auch über die Gesamtbeschleunigung (Betrag des Beschleunigungsvektors) berechnen. Zunächst werden wir uns aber erstmal auf die Gesamtbeschleunigung, also den Betrag des Beschleunigungsvektors beschränken.

Die Features sind nichts anderes als statistische Kenngrößen, die die Eigenschaften des Signals bestmöglich beschreiben sollen.

Teil a | Statistische Kenngrößen

Im Mathematikunterricht hast du bereits verschiedene solcher statistischen Kenngrößen kennengelernt.

▶ **Erläuterung statistische Kenngröße** (Falls du den Begriff statistische Kenngröße nicht kennst, kannst du hier klicken.)

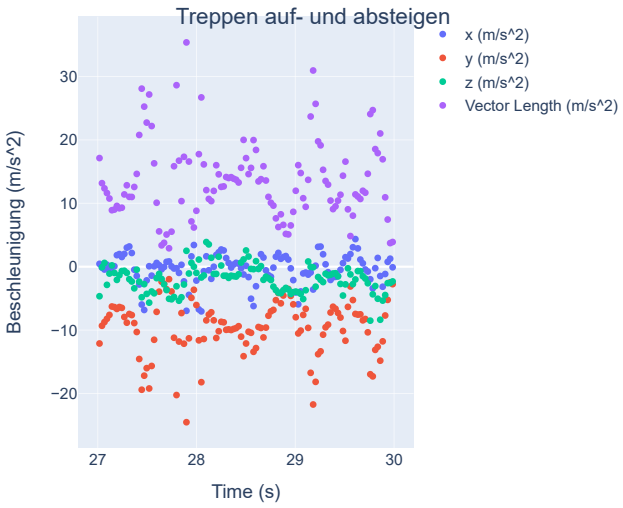
Notiere verschiedene statistische Kenngrößen (min. 3), die du bereits aus dem Mathematikunterricht kennst, auf deinem Antwortblatt.

Teil b | Graphische Darstellung

Zu Auswahl der Features, die unsere Daten bestmöglich beschreiben, werden wir uns zunächst die Daten verschiedener Windows graphisch anschauen.

Ersetze das `NaN` durch eine WindowID zwischen 1 und 300. Führe den Code aus. Dir wird ein Graph, in dem die Beschleunigungswerte der drei Achsen sowie der Betrag des Beschleunigungsvektors gegen die Zeit aufgetragen sind, für das gewählte Window ausgegeben. Notiere auf deinem Antwortblatt statistische Kenngrößen, die du zur Beschreibung der Daten eines Windows nutzen würdest.

In [5]:
```
windowID = 250;

# Hier nichts ändern!
plotWindowID(windowID)
```



Teil c | Statisitsche Kenngrößen berechnen

Bei der Beschreibung der Daten eines Windows werden wir uns zunächst sowohl auf den Betrag des Beschleunigungsvektors als auch auf drei statistische Kenngrößen beschränken (Klicke auf den Pfeil, um die ausgewählten statistischen Kenngrößen zu sehen).

▶ **ausgewählte statistische Kenngrößen**

Diese Kenngrößen werden wir nun beispielhaft für den Betrag des Beschleunigungsvektors von sieben Datenpunkten aus dem Window mit der WindowID 121 berechnen.

WindowID	UserID	ActivityID	Time(s)	x (m/s^2)	y (m/s^2)	z (m/s^2)	Vector Length (m/s^2)
121	1	3	0.0	-1.1	-11.2	0.1	11.3
121	1	3	0.5	-1.3	-8.9	-3.9	9.8
121	1	3	1.0	2.0	-1.1	-1.2	2.6
121	1	3	1.5	-6.1	-6.0	-4.5	9.7
121	1	3	2.0	-8.9	-8.2	-3.7	12.6
121	1	3	2.5	2.6	-1.1	-2.5	3.8
121	1	3	3.0	0.0	-5.7	0.3	5.7

Bestimme für die Messwerte in der Tabelle den Mittelwert, das Maximum und das Minimum des Betrags des Beschleunigungsvektors. Ersetze die `NaN` *durch deine Lösungen und führe anschließend den Code aus.*

⚠ Hinweise zur Eingabe im Codefeld:

- Gib die Lösungen entweder auf eine Nachkommastelle gerundet oder als Formel ein.

Falls du Hilfe benötigst, kannst du dir hier einen Tipp ansehen.

In [6]:
```
# Mittelwert Vector Length
meanVectorLength = 1/7 * (11.3 + 9.8 + 2.6 + 9.7 + 12.6 + 3.8 + 5.7);

# Minimum Vector Length
minVectorLength = 2.6;

# Maximum Vector Length
maxVectorLength = 12.6;

# Hier nichts ändern!
checkFeatures(meanVectorLength, minVectorLength, maxVectorLength)
```

```
✓ Dein Ergebnis für den Mittelwert ist richtig.
✓ Dein Ergebnis für das Minimum ist richtig.
✓ Dein Ergebnis für das Maximum ist richtig.
```

Teil d | Automatische Berechnung der statistischen Kenngrößen

Jedes Window enthält ca. 120 Datenpunkte. Daher wäre es ziemlich aufwändig, für jedes Window die statistischen Kenngrößen per Hand auszurechnen. Stattdessen können wir diese Aufgabe dem Computer übergeben.

Ersetze das `NaN` *durch eine beliebige WindowID zwischen 1 und 3000 und führe den Code aus. Dir werden die statistischen Kenngrößen Minimum, Maximum und Mittelwert für den Betrag des Beschleunigungsvektors und das gewählte Window ausgegeben.*

In [7]:
```
windowID = 250;

# Hier nichts ändern!
printFeatures(windowID)
```

```
WindowID: 250 , UserID: 1.0 , ActivityID: 5.0
----
Mittelwert Vector Length: 13.579089454510791
Minimum Vector Length: 2.9069814177353503
Maximum Vector Length: 35.37600029803454
```

Teil e | Abspeichern der statistischen Kenngrößen

Um für unsere Daten ein maschinelles Lernverfahren entwickeln zu können, müssen wir die oben berechneten statistischen Kenngrößen zusammen mit der entsprechenden WindowID, ActivityID und UserID in einem neuen Datensatz abspeichern. Diesen neuen Datensatz bezeichnen wir als **Feature-Set 1**. In diesem neuen Datensatz steht jede Zeile für ein Window.

Führe den Code aus. Dir werden die ersten und die letzten fünf Zeilen des neuen Datensatzes angezeigt.

In [8]:
```
# Hier nichts ändern!
printFeatureSet1()
```

```
Die ersten 5 Zeilen des neuen Datensatzes:
```

WindowID	UserID	ActivityID	Mean Vector Length (m/s^2)	Max Vector Length (m/s^2)	Min Vector Length (m/s^2)
1.0	1.0	1.0	8.382189	8.593413	8.136967
2.0	1.0	1.0	8.335617	8.571866	8.052737
3.0	1.0	1.0	8.289633	8.384486	8.059064
4.0	1.0	1.0	8.262474	8.533645	7.956085
5.0	1.0	1.0	8.268336	8.325962	8.224459

```
Die letzten 5 Zeilen des neuen Datensatzes:
```

WindowID	UserID	ActivityID	Mean Vector Length (m/s^2)	Max Vector Length (m/s^2)	Min Vector Length (m/s^2)
2996.0	10.0	5.0	13.059342	29.251910	4.153431
2997.0	10.0	5.0	13.351763	29.893499	5.658197
2998.0	10.0	5.0	13.004548	24.820680	3.930397
2999.0	10.0	5.0	13.723280	26.346591	4.745439
3000.0	10.0	5.0	12.273405	27.153701	4.565421

Fazit

Du hast die Vorverarbeitung der Daten abgeschlossen. Auf dem nächsten Arbeitsblatt werden wir uns mit der Zuordnung der Daten zu den einzelnen Aktivitäten befassen.

Diskussion im Plenum

Wir werden nun die Erkenntnisse und Ergebnisse von Arbeitsblatt 2 gemeinsam im Plenum besprechen.

Öffne nun das nächste Arbeitsblatt.

Autorin: Katja Hoeffer

B.3. Arbeitsblatt 2 short

Arbeitsblatt 2: Vorverarbeitung der Daten

Notiere deine Ergebnisse während dem Bearbeiten des Arbeitsblattes auf diesem Antwortblatt.

Einleitung:

Für die aufgenommenen Rohdaten des Beschleunigungsmessers können wir nicht direkt ein Lernverfahren aus dem Bereich des maschinellen Lernens entwickeln. Zuvor müssen wir die Daten noch aufbereiten. Dieser Prozess wird auch als **Data Preprocessing** bezeichnet.

Zur Aufbereitung der Daten werden wir in einem ersten Schritt die Beschleunigungswerte der drei Achsen kombinieren. Anschließend werden wir die große Menge an Rohdaten auf eine kleinere Menge von möglichst aussagekräftigen Werten (sogenannte **Features**) reduzieren, die die Eigenschaften des Signals bestmöglich abbilden. Diese Features werden wir jedoch nicht über die Gesamtmenge der Rohdaten berechnen. Stattdessen werden wir die Daten zuvor in Zeitfenster einer festen Länge (auch **Windows** genannt) unterteilen. Anschließend können wir die Features über diese Windows berechnen.

Auf diesem Arbeitsblatt lernst du den Prozess der Datenvorverarbeitung kennen.

Aufgabe 1 | Der Betrag des Beschleunigungsvektors

Jeder Datenpunkt besteht aus den Beschleunigungswerten in den drei Raumdimensionen x, y und z. Um die Gesamtbeschleunigung des Smartphones bestimmen zu können, müssen wir die Beschleunigungswerte der drei Raumdimensionen kombinieren. Hierzu bilden wir den sogenannten **Betrag des Beschleunigungsvektors**, den wir im folgenden als **Vector Length** bezeichnen.

Falls du mehr über Vektoren und deren Betrag wissen möchtest, kannst du dir dieses Infoblatt anschauen.

Damit wir die Gesamtbeschleunigung des Smartphones bei der späteren Auswertung der Daten nutzen können, ergänzen wir eine zusätzliche Spalte in unserem Datensatz. In dieser Spalte tragen wir den Betrag des Beschleunigungsvektors für jeden Datenpunkt ein.

Führe den Code aus. Dir werden die ersten zehn Zeilen des Datensatzes mit der zusätzlichen Spalte für den Betrag des Beschleunigungsvektors ausgegeben.

In [1]:
```
# Hier nichts ändern!
import sys; sys.path.append("../code"); from setupAB2 import *;
display(HTML(dataVectorLength.head(10).to_html(index=False)))
```

UserID	ActivityID	Time (s)	x (m/s^2)	y (m/s^2)	z (m/s^2)	Vector Length (m/s^2)
1	1	0.028140	5.480853	-4.508624	-6.869365	8.408040
1	1	0.053292	5.488187	-4.504134	-6.870263	8.408010
1	1	0.078443	5.494324	-4.496799	-6.872658	8.404166
1	1	0.103595	5.492229	-4.508475	-6.871311	8.415299
1	1	0.128747	5.472320	-4.499044	-6.856192	8.392204
1	1	0.153899	5.456753	-4.472849	-6.885382	8.353975
1	1	0.179050	5.481302	-4.499344	-6.910080	8.398384
1	1	0.204202	5.500462	-4.511618	-6.910679	8.424041
1	1	0.229354	5.493127	-4.491260	-6.871161	8.397457
1	1	0.254505	5.481451	-4.456682	-6.861881	8.352864

Aufgabe 2 | Einteilung der Windows

Als nächstes werden wir die große Menge an Rohdaten in kleinere Zeitfenster (**Windows**) einer festen Länge unterteilen.

Teil a | Die Länge der Windows

Bei der Wahl der Länge der Windows müssen verschiedene Faktoren berücksichtigt werden:

- Die Windows müssen mindestens so groß sein, dass mehrere charakteristische Bewegungen einer Aktivität innerhalb der Zeit eines Windows durchgeführt werden (z.B. zwei Schritte vorwärts beim Gehen).
- Die Länge des Windows sollte ein ganzzahliger Teiler von 180 sein, da wir ansonsten innerhalb eines Windows Daten zu zwei Aktivitäten vermischen würden.
- Die Windows sollten so klein wie möglich sein, da dies unsere spätere Zuordnung der Rohdaten zu einer Aktivität verbessern wird.

Ersetze das `NaN` *durch eine mögliche Länge der Windows in Sekunden und führe den Code anschließend aus.*

In [2]:
```
windowLength = 5;

# Hier nichts ändern!
checkWindowLength(windowLength)
```

```
✓ Deine Wahl für die Länge der Windows ist sinnvoll.
```

Teil b | Ergänzung im Datensatz

Der Einheitlichkeit wegen legen wir die Länge der Windows auf eine feste Zeitspanne fest (Klicke auf den Pfeil, um die gewählte Länge der Windows zu sehen).

▶ **gewählte Länge der Windows**

Um die aufgenommenen Daten den Windows optimal zuordnen zu können, nummerieren wir die Zeitfenster mit Hilfe einer sogenannten **WindowID** fortlaufend durch. Diese WindowID fügen wir als eine zusätzliche Spalte unserem Datensatz hinzu.

Führe den Code aus. Dir werden die ersten und die letzten fünf Zeilen des Datensatzes mit der zusätzlichen Spalte für die WindowID ausgegeben.

In [3]:
```
# Hier nichts ändern!
printDataWindow()
```

```
Die ersten 5 Zeilen des Datensatzes:
```

WindowID	UserID	ActivityID	Time (s)	x (m/s^2)	y (m/s^2)	y (m/s^2)	Vector Length (m/s^2)
1	1	1	0.028140	5.480853	-4.508624	-4.508624	8.408040
1	1	1	2.216343	5.487589	-4.477938	-4.477938	8.379587
1	1	1	2.191192	5.516029	-4.445905	-4.445905	8.364133
1	1	1	2.166040	5.550458	-4.380790	-4.380790	8.318066
1	1	1	2.140888	5.550907	-4.342769	-4.342769	8.278397

```
Die letzten 5 Zeilen des Datensatzes:
```

WindowID	UserID	ActivityID	Time (s)	x (m/s^2)	y (m/s^2)	y (m/s^2)	Vector Length (m/s^2)
3000	10	5	177.814545	2.127376	-6.044580	-6.044580	8.809065
3000	10	5	177.789387	2.086511	-6.294261	-6.294261	9.142700
3000	10	5	177.764229	2.523003	-6.089187	-6.089187	8.973403
3000	10	5	178.066125	0.508493	-10.040820	-10.040820	14.208965
3000	10	5	179.978142	3.050956	-10.346634	-10.346634	14.947040

Aufgabe 3 | Wahl der Features

Als nächstes werden wir verschiedene **Features** berechnen, die unsere Daten bestmöglich repräsentieren. Diese Features können wir sowohl über die Beschleunigungswerte in den drei Raumdimensionen als auch über die Gesamtbeschleunigung (Betrag des Beschleunigungsvektors) berechnen. Zunächst werden wir uns aber auf die Gesamtbeschleunigung, also den Betrag des Beschleunigungsvektors beschränken.

Die Features sind nichts anderes als statistische Kenngrößen, die die Eigenschaften des Signals bestmöglich beschreiben sollen.

Teil a | Statistische Kenngrößen

Im Mathematikunterricht hast du bereits verschiedene solcher statistischen Kenngrößen kennengelernt.

▶ **Erläuterung statistische Kenngröße** (Falls du den Begriff statistische Kenngröße nicht kennst, kannst du hier klicken.)

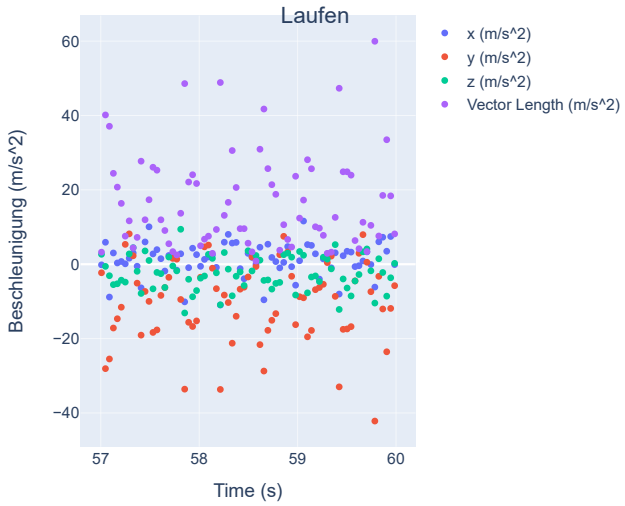
Notiere verschiedene statistische Kenngrößen (min. 3), die du bereits aus dem Mathematikunterricht kennst, auf deinem Antwortblatt.

Teil b | Graphische Darstellung

Zu Auswahl der Features, die unsere Daten bestmöglich beschreiben, werden wir uns zunächst die Daten verschiedener Windows graphisch anschauen.

Ersetze das `NaN` durch eine WindowID zwischen 1 und 300. Führe den Code aus. Dir wird ein Graph, in dem die Beschleunigungswerte der drei Achsen sowie der Betrag des Beschleunigungsvektors gegen die Zeit aufgetragen sind, für das gewählte Window ausgegeben. Notiere auf deinem Antwortblatt statistische Kenngrößen, die du zur Beschreibung der Daten eines Windows nutzen würdest.

In [4]:
```
windowID = 200;

# Hier nichts ändern!
plotWindowID(windowID)
```



Teil c | Statisitsche Kenngrößen berechnen

Bei der Beschreibung der Daten eines Windows werden wir uns zunächst sowohl auf den Betrag des Beschleunigungsvektors als auch auf drei statistische Kenngrößen beschränken (Klicke auf den Pfeil, um die ausgewählten statistischen Kenngrößen zu sehen).

▶ **ausgewählte statistische Kenngrößen**

Diese Kenngrößen werden wir nun beispielhaft für den Betrag des Beschleunigungsvektors von sieben Datenpunkten aus dem Window mit der WindowID 121 berechnen.

WindowID	UserID	ActivityID	Time(s)	x (m/s^2)	y (m/s^2)	z (m/s^2)	Vector Length (m/s^2)
121	1	3	0.0	-1.1	-11.2	0.1	11.3
121	1	3	0.5	-1.3	-8.9	-3.9	9.8
121	1	3	1.0	2.0	-1.1	-1.2	2.6
121	1	3	1.5	-6.1	-6.0	-4.5	9.7
121	1	3	2.0	-8.9	-8.2	-3.7	12.6
121	1	3	2.5	2.6	-1.1	-2.5	3.8
121	1	3	3.0	0.0	-5.7	0.3	5.7

Bestimme für die Messwerte in der Tabelle den Mittelwert, das Maximum und das Minimum des Betrags des Beschleunigungsvektors. Ersetze die `NaN` *durch deine Lösungen und führe anschließend den Code aus.*

⚠ Hinweise zur Eingabe im Codefeld:

- Gib die Lösungen entweder auf eine Nachkommastelle gerundet oder als Formel ein.

Falls du Hilfe benötigst, kannst du dir hier einen Tipp ansehen.

In [5]:
```
# Mittelwert Vector Length
```

```
meanVectorLength = 1/7 * (11.3 + 9.8 + 2.6 + 9.7 + 12.6 + 3.8 + 5.7);

# Minimum Vector Length
minVectorLength = 2.6;

# Maximum Vector Length
maxVectorLength = 12.6;

# Hier nichts ändern!
checkFeatures(meanVectorLength, minVectorLength, maxVectorLength)
```

```
✓ Dein Ergebnis für den Mittelwert ist richtig.
✓ Dein Ergebnis für das Minimum ist richtig.
✓ Dein Ergebnis für das Maximum ist richtig.
```

Teil d | Automatische Berechnung der statistischen Kenngrößen

Jedes Window enthält ca. 120 Datenpunkte. Daher wäre es ziemlich aufwändig, für jedes Window die statistischen Kenngrößen per Hand auszurechnen. Stattdessen können wir diese Aufgabe dem Computer übergeben.

Ersetze das `NaN` durch eine beliebige WindowID zwischen 1 und 3000 und führe den Code aus. Dir werden die statistischen Kenngrößen Minimum, Maximum und Mittelwert für den Betrag des Beschleunigungsvektors und das gewählte Window ausgegeben.

In [6]:
```
windowID = 200;

# Hier nichts ändern!
printFeatures(windowID)
```

```
WindowID: 200 , UserID: 1.0 , ActivityID: 4.0
----
Mittelwert Vector Length: 16.72846252238446
Minimum Vector Length: 0.8522106217195654
Maximum Vector Length: 59.96000952691293
```

Teil e | Abspeichern der statistischen Kenngrößen

Um für unsere Daten ein maschinelles Lernverfahren entwickeln zu können, müssen wir die oben berechneten statistischen Kenngrößen zusammen mit der entsprechenden WindowID, ActivityID und UserID in einem neuen Datensatz abspeichern. Diesen neuen Datensatz bezeichnen wir als **Feature-Set 1**. In diesem neuen Datensatz steht jede Zeile für ein Window.

Führe den Code aus. Dir werden die ersten und die letzten fünf Zeilen des neuen Datensatzes angezeigt.

In [7]:
```
# Hier nichts ändern!
printFeatureSet1()
```

```
Die ersten 5 Zeilen des neuen Datensatzes:
```

WindowID	UserID	ActivityID	Mean Vector Length (m/s^2)	Max Vector Length (m/s^2)	Min Vector Length (m/s^2)
1.0	1.0	1.0	8.382189	8.593413	8.136967
2.0	1.0	1.0	8.335617	8.571866	8.052737
3.0	1.0	1.0	8.289633	8.384486	8.059064
4.0	1.0	1.0	8.262474	8.533645	7.956085
5.0	1.0	1.0	8.268336	8.325962	8.224459

```
Die letzten 5 Zeilen des neuen Datensatzes:
```

WindowID	UserID	ActivityID	Mean Vector Length (m/s^2)	Max Vector Length (m/s^2)	Min Vector Length (m/s^2)
2996.0	10.0	5.0	13.059342	29.251910	4.153431
2997.0	10.0	5.0	13.351763	29.893499	5.658197
2998.0	10.0	5.0	13.004548	24.820680	3.930397
2999.0	10.0	5.0	13.723280	26.346591	4.745439
3000.0	10.0	5.0	12.273405	27.153701	4.565421

Fazit

Du hast die Vorverarbeitung der Daten abgeschlossen. Auf dem nächsten Arbeitsblatt werden wir uns mit der Zuordnung der Daten zu den einzelnen Aktivitäten befassen.

Diskussion im Plenum

Wir werden nun die Erkenntnisse und Ergebnisse von Arbeitsblatt 2 gemeinsam im Plenum besprechen.

Öffne nun das nächste Arbeitsblatt.

Autorin: Katja Hoeffer

B.4. Arbeitsblatt 3

Arbeitsblatt 3: Der k-nächste-Nachbarn-Algorithmus

Notiere deine Ergebnisse während dem Bearbeiten des Arbeitsblattes auf diesem Antwortblatt.

Einleitung:

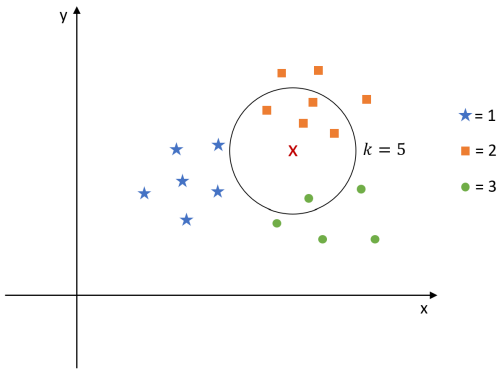
Wir werden nun die Datenpunkte, die in unserem **Feature-Set 1** abgespeichert sind, den verschiedenen Aktivitäten (Sitzen, Laufen, ...) zuordnen. Diesen Vorgang bezeichnet man als **Klassifizierung**.

Im Bereich des maschinellen Lernens werden verschiedene Algorithmen eingesetzt, um Klassifizierungsprobleme zu lösen. Wir werden unser Problem mit Hilfe des **k-nächste-Nachbarn-Algorithmus (kNN-Algorithmus)** lösen.

Auf diesem Arbeitsblatt wirst du den k-nächste-Nachbarn-Algorithmus kennenlernen und implementieren.

Aufgabe 1 | Die Grundidee des k-nächste-Nachbarn-Algorithmus

Die Grundidee des k-nächste-Nachbarn-Algortihmus ist es, unbekannte Datenpunkte anhand der Klassen der k nächstliegenden Datenpunkte (auch **Nachbarn** genannt) zu klassifizieren.



Ersetze das `NaN` *durch die Klasse (1, 2 oder 3), der du das rote x zuordnen würdest. Führe anschließend den Code aus.*

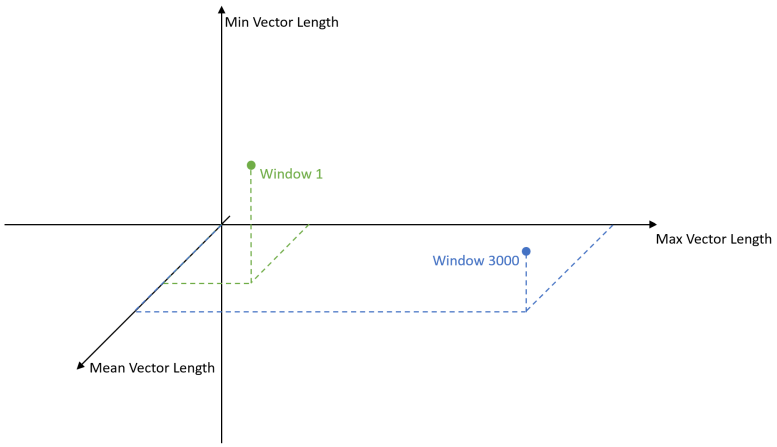
In [1]:
```
# Hier nichts ändern!
import sys; sys.path.append("../code"); from setupAB3 import *;

redX = 2;

# Hier nichts ändern!
checkRedX(redX)
```

```
✓ Du hast den Datenpunkt richtig zugeordnet.
Das rote x muss der Klasse 2, also den orangenen Quadraten zugeordnet werden.
```

Aufgabe 2 | Die Abstandsfunktion

Wir werden jetzt die Grundidee des k-nächste-Nachbarn-Algorithmus auf unser Problem, die **Aktivitätserkennung**, anwenden.

Jedes Window im Feature-Set 1 entspricht einem Datenpunkt in drei Dimensionen. Die Dimensionen sind die drei berechneten Features Mittelwert, Maximum und Minimum des Betrags des Beschleunigungsvektors.

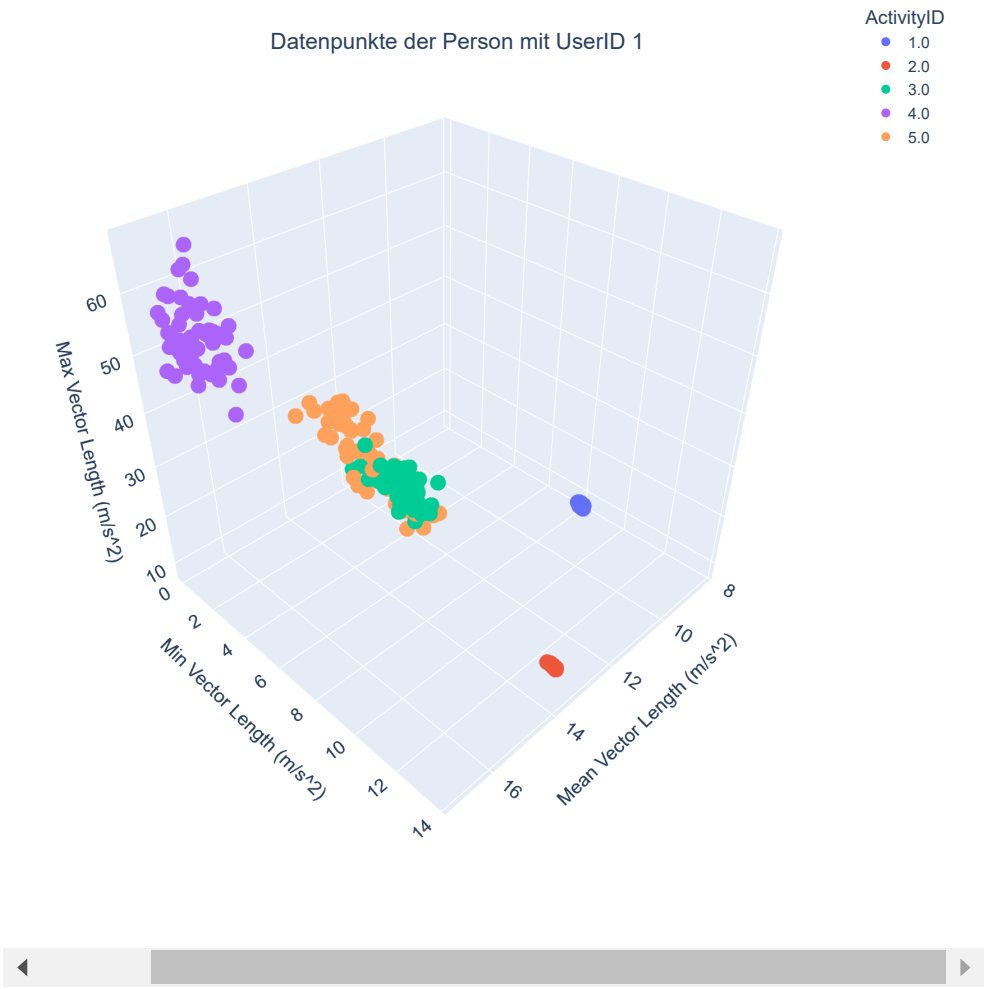


Teil a | Graphische Darstellung der Datenpunkte

Zunächst werden wir uns die Datenpunkte der einzelnen Windows der Person mit UserID 1 graphisch anschauen.

Führe den Code aus. Dir wird eine Graphik mit den Datenpunkte der Person mit UserID 1 angezeigt.

In [2]:
```
# Hier nichts ändern!
showDataUser1()
```

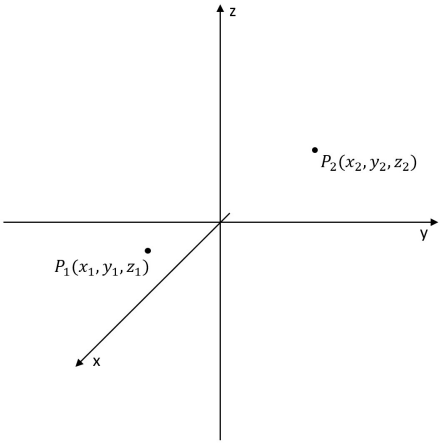


Teil b | Interpretation der graphischen Darstellung

Schaue dir die Graphik aus Teil a nochmal genauer an. Bei welchen Aktivitäten könnte der k-nächste-Nachbarn-Algorithmus bei der Klassifikation Schwierigkeiten haben und warum? Notiere deine Überlegungen auf deinem Antwortblatt.

Teil c | Definition der Abstandsfunktion

Um die k nächsten Nachbarn zu einem unbekannten Datenpunkt zu finden, müssen wir eine Funktion definieren, mit der wir den **Abstand** zwischen zwei Datenpunkten bestimmen können.



Ersetze das `NaN` *durch eine Formel, mit der du den Abstand zwischen den beiden Datenpunkten* `P1(x1, y1, z1)` *und* `P2(x2, y2, z2)` *berechnen kannst. Führe anschließend den Code aus.*

⚠ Hinweise zur Eingabe im Codefeld:

- Wurzeln $\sqrt{x}$ werden im Code mit `sqrt(x)` eingegeben.
- Exponenten wie z.B. x^2 werden im Code mit `x**2` eingeben.

Falls du Hilfe benötigst, kannst du dir hier einen Tipp ansehen.

In [3]:
```
distanceP1P2 = sqrt((x2-x1)**2 + (y2-y1)**2 + (z2-z1)**2);

# Hier nichts ändern!
checkDistanceP1P2(distanceP1P2)
```

```
✓ Deine Formel für die Berechnung des Abstands zwischen den beiden Datenpunkten ist korrekt.
```

Teil d | Berechnung des Abstands zwischen zwei Windows

Mit Hilfe unserer Abstandsfunktion können wir den Abstand zwischen den Datenpunkten zweier Windows berechnen.

Ersetze die `NaN's` *durch die WindowID's zweier beliebiger Windows. Führe anschließend den Code aus. Dir wird der Abstand zwischen den beiden Datenpunkten der von dir gewählten Windows ausgegeben.*

In [4]:
```
window1 = 10;
window2 = 170;

# Hier nichts ändern!
printDistance(window1, window2)
```

```
Der Abstand zwischen dem Window mit der WindowID 10 und dem Window mit der WindowID 170 beträgt 22.767956426809
004 LE.
```

Aufgabe 3 | Suche der k nächsten Nachbarn

Nachdem wir nun eine Funktion aufgestellt haben, mit der wir den Abstand zwischen zwei Datenpunkten bestimmen können, werden wir uns mit der Frage beschäftigen, wie wir die k nächsten Nachbarn von einem ausgewählten Window bestimmen können.

Teil a | Aufstellen einer Suchfunktion

Ersetze das erste `NaN` *durch die Anzahl der Nachbarn, die du bestimmen möchtest, und das zweite* `NaN` *durch die WindowID des Test-Windows, für das du die Nachbarn bestimmen möchtest. Erstelle anschließend mithilfe des vorgefertigten Gerüsts einen Code zur Berechnung der k nächsten Nachbarn. Du kannst dabei die folgenden Funktionen nutzen:*

- `sortDistances(distances)` :
  Mit dieser Funktion kannst du die Abstände in der Liste `distances` der Größe nach (von klein zu groß) sortieren.
- `computeDistance(i, TestWindow)` :
  Diese Funktion liefert den Abstand zwischen dem Window mit der WindowID i und dem Test-Window.
- `getNeighbors(distances, k)` :
  Diese Funktion liefert dir die k ersten Einträge der Liste `distances` .
- `addDistances(i, dist, distances)` :
  Mit dieser Funktion kannst du den berechneten Abstand `dist` zur Liste `distances` aller Abstände hinzufügen.

Führe anschließend den Code aus. Wenn du alles richtig gemacht hast, werden dir die k nächsten Nachbarn, deren Abstand und deren ActivityID ausgegeben.

Falls du Hilfe benötigst, kannst du dir hier einen Tipp ansehen.

In [5]:
```
k = 5;
TestWindow = 170;

# leere Liste, in der nach und nach die Abstände gespeichert werden sollen.
distances = []

# For-Schleife zur Berechnung der Abstände zwischen allen Windows und dem Test-Window
# und Abspeichern in der Liste distances
for i in range(1, 3000):
    dist = computeDistance(i, TestWindow)
    addDistances(i, dist, distances)

# Löschen des Abstands des Test-Windows zu sich selbst
del distances[TestWindow-1]

# Sortieren der Liste nach den kürzesten Abständen zum Test-Window
sortDistances(distances)

# Filtern der k nächsten Nachbarn
kneighbors = getNeighbors(distances, k)

# Hier nichts ändern!
checkKNeighbors(kneighbors, k, TestWindow)
```

```
✓ Deine Lösung ist richtig.
Die 5 nächsten Nachbarn des Test-Windows mit der WindowID 170 sind (WindowID, Abstand, ActivityID):
(132, 0.250151512805162, 3.0)
(155, 0.46736543466901054, 3.0)
(136, 0.5926689314414559, 3.0)
(139, 0.7042155790552647, 3.0)
(156, 0.7539688458461625, 3.0)
```

Teil b | Interpretation der Ergebnisse

Schaue dir nochmal die k nächsten Nachbarn des Test-Windows aus Teil a an. Welche Aktivität würdest du dem von dir gewählten Test-Window zuordnen und warum? Notiere deine Überlegungen auf deinem Antwortblatt.

Aufgabe 4 | Häufigkeiten der einzelnen Klassen

Wir können nun die k nächsten Nachbarn zu einem Window bestimmen. Um diesem Window eine Aktivität zuordnen zu können, müssen wir noch einen **Mehrheitsentscheid** unter den k nächsten Nachbarn durchführen. Wir bestimmen also die

Aktivität, die unter den k nächsten Nachbarn am häufigsten vorkommt. Die am häufigsten vorkommende Aktivität ordnen wir dann unserem Window zu.

Teil a | Mehrheitsentscheid

Ersetze das erste `NaN` *durch die Anzahl der Nachbarn, die du miteinbeziehen möchtest, und das zweite* `NaN` *durch die WindowID des Test-Windows, für das du die Aktivität über einen Mehrheitsentscheid bestimmen möchtest.*

In [6]:
```
k = 5;
TestWindow = 170;

# Hier nichts ändern!
PrintClassVote(k, TestWindow)
```

```
Unter den 5 nächsten Nachbarn des Test-Windows mit der WindowID 170 kommt die Aktivität mit der AktivityID 3.0
am häufigsten vor.
```

Teil b | Überprüfung des Ergebnisses

Zum Abschluss können wir noch überprüfen, ob die Aktivität, die wir dem Test-Window über den k-nächste-Nachbarn-Algorithmus zugeordnet haben, auch der Aktivität entspricht, zu der die Daten aufgenommen wurden.

Ersetze das `NaN` *durch die WindowID des Test-Windows, für das du zuvor den Mehrheitsentscheid in Teil a durchgeführt hast. Führe anschließend den Code aus. Dir wird die tatsächliche ActivityID des gewählten Test-Windows ausgegeben.*

Stimmen die beiden Aktivitäten von Teil a und Teil b überein? Notiere deine Ergebnisse auf deinem Antwortblatt.

In [7]:
```
TestWindow = 170;

# Hier nichts ändern!
PrintActivityID(TestWindow)
```

```
Das Test-Window mit der WindowID 170 enthält Daten zur Aktivität mit der ActivityID 3.0
```

Fazit

Du hast auf diesem Arbeitsblatt den k-nächste-Nachbarn-Algorithmus kennengelernt und bereits erste Windows mithilfe dieses Algorithmus einer Aktivität zugeordnet. Auf dem nächsten Arbeitsblatt werden wir uns anschauen, wie wir die Leistung unseres Algorithmus bewerten können.

Diskussion im Plenum

Wir werden nun die Erkenntnisse und Ergebnisse von Arbeitsblatt 3 gemeinsam im Plenum besprechen.

Öffne nun das nächste Arbeitsblatt.

Schon fertig?

Bearbeite die folgende Zusatzaufgabe.

Zusatzaufgabe | Weitere Abstandsfunktionen

Kennst du noch weitere Funktionen, mit denen der Abstand zweier Datenpunkte bestimmt werden kann? Falls nein, recherchiere im Internet nach weiteren Abstandsfunktionen! (Abstandsfunktionen werden in der Fachliteratur oft auch als Abstandsmetriken bezeichnet.)

Du kannst deine Überlegungen mit den Informationen auf diesem Infoblatt vergleichen.

Autorin: Katja Hoeffer

B.5. Arbeitsblatt 3 short

Arbeitsblatt 3: Der k-nächste-Nachbarn-Algorithmus

Notiere deine Ergebnisse während dem Bearbeiten des Arbeitsblattes auf diesem Antwortblatt.

Einleitung:

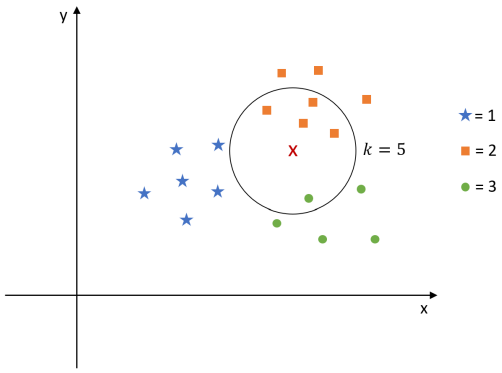
Wir werden nun die Datenpunkte, die in unserem **Feature-Set 1** abgespeichert sind, den verschiedenen Aktivitäten (Sitzen, Laufen, ...) zuordnen. Diesen Vorgang bezeichnet man als **Klassifizierung**.

Im Bereich des maschinellen Lernens werden verschiedene Algorithmen eingesetzt, um Klassifizierungsprobleme zu lösen. Wir werden unser Problem mit Hilfe des **k-nächste-Nachbarn-Algorithmus (kNN-Algorithmus)** lösen.

Auf diesem Arbeitsblatt wirst du den k-nächste-Nachbarn-Algorithmus kennenlernen und implementieren.

Aufgabe 1 | Die Grundidee des k-nächste-Nachbarn-Algorithmus

Die Grundidee des k-nächste-Nachbarn-Algortihmus ist es, unbekannte Datenpunkte anhand der Klassen der k nächstliegenden Datenpunkte (auch **Nachbarn** genannt) zu klassifizieren.



Ersetze das `NaN` *durch die Klasse (1, 2 oder 3), der du das rote x zuordnen würdest. Führe anschließend den Code aus.*

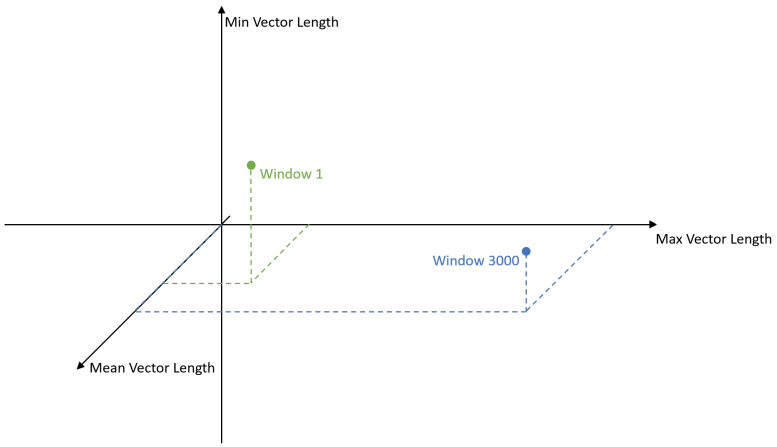
In [1]:
```
# Hier nichts ändern!
import sys; sys.path.append("../code"); from setupAB3short import *;

redX = 2;

# Hier nichts ändern!
checkRedX(redX)
```

```
✓ Du hast den Datenpunkt richtig zugeordnet.
Das rote x muss der Klasse 2, also den orangenen Quadraten zugeordnet werden.
```

Aufgabe 2 | Die Abstandsfunktion

Wir werden jetzt die Grundidee des k-nächste-Nachbarn-Algorithmus auf unser Problem, die **Aktivitätserkennung**, anwenden.

Jedes Window im Feature-Set 1 entspricht einem Datenpunkt in drei Dimensionen. Die Dimensionen sind die drei berechneten Features Mittelwert, Maximum und Minimum des Betrags des Beschleunigungsvektors.

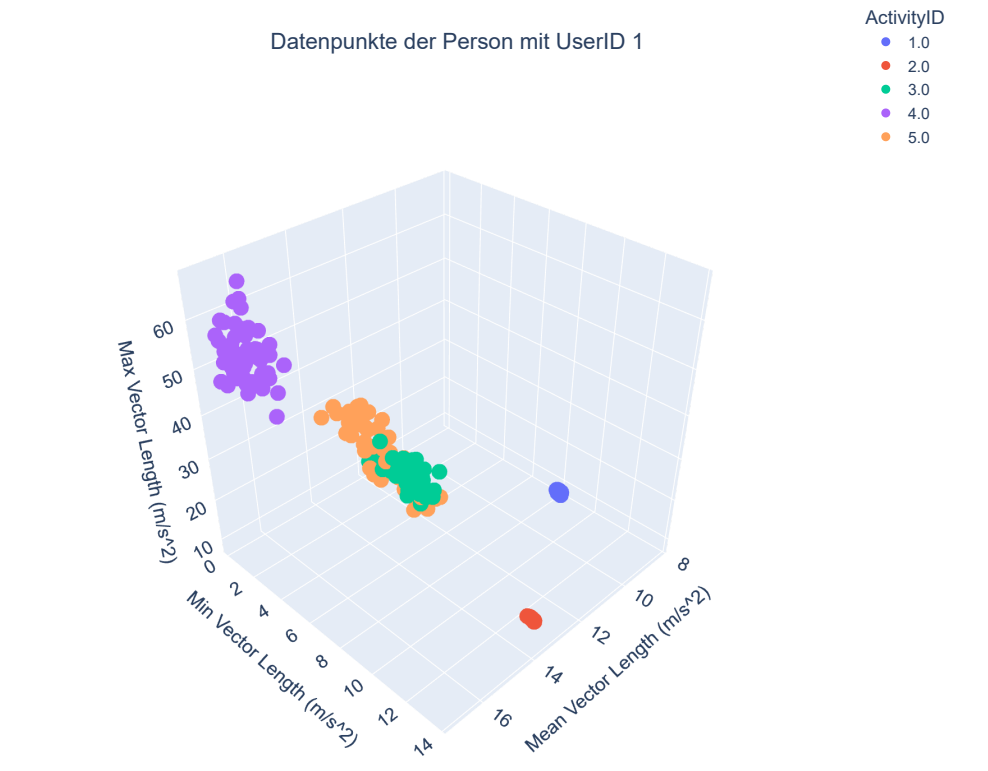


Teil a | Graphische Darstellung der Datenpunkte

Zunächst werden wir uns die Datenpunkte der einzelnen Windows der Person mit UserID 1 graphisch anschauen.

Führe den Code aus. Dir wird eine Graphik mit den Datenpunkte der Person mit UserID 1 angezeigt.

In [2]:
```
# Hier nichts ändern!
showDataUser1()
```



Teil b | Interpretation der graphischen Darstellung

Schaue dir die Graphik aus Teil a nochmal genauer an. Bei welchen Aktivitäten könnte der k-nächste-Nachbarn-Algorithmus bei der Klassifikation Schwierigkeiten haben und warum? Notiere deine Überlegungen auf deinem Antwortblatt.

Teil c | Berechnung des Abstands zwischen zwei Windows

Mit Hilfe einer **Abstandsfunktion** können wir den Abstand zwischen den Datenpunkten zweier Windows berechnen.

Falls du wissen möchtest, wie diese Abstandsfunktion aussieht und wie sie hergeleitet werden kann, kannst du dir dieses Infoblatt anschauen.

Ersetze die `NaN's` *durch die WindowID's zweier beliebiger Windows. Führe anschließend den Code aus. Dir wird der Abstand zwischen den beiden Datenpunkten der von dir gewählten Windows ausgegeben.*

In [3]:
```
window1 = 20;
window2 = 250;

# Hier nichts ändern!
printDistance(window1, window2)
```

```
Der Abstand zwischen dem Window mit der WindowID 20 und dem Window mit der WindowID 250 beträgt 27.975021282390
88 LE.
```

Aufgabe 3 | Suche der k nächsten Nachbarn

Nachdem wir nun eine Funktion aufgestellt haben, mit der wir den Abstand zwischen zwei Datenpunkten bestimmen können, werden wir uns mit der Frage beschäftigen, wie wir die k nächsten Nachbarn von einem ausgewählten Window bestimmen können.

Teil a | Die k nächsten Nachbarn

Ersetze das erste `NaN` *durch die Anzahl der nächsten Nachbarn, die du finden möchtest. Ersetze das zweite* `NaN` *durch die WindowID des Test-Windows, für das du die Nachbarn bestimmen möchtest. Führe den Code aus. Dir werden die k nächsten Nachbarn des von dir gewählten Test-Windows ausgegeben.*

Falls du wissen möchtest, wie die k nächsten Nachbarn bestimmt werden, kannst du dir dieses Infoblatt anschauen.

In [4]:
```
k = 4;
TestWindow = 250;

# Hier nichts ändern!
printKNeighbors(k, TestWindow)
```

```
Die 4 nächsten Nachbarn des Test-Windows mit der WindowID 250 sind (WindowID, Abstand, ActivityID):
(251, 0.0, 5.0)
(767, 0.26186897701406286, 3.0)
(1797, 0.2751978796892001, 5.0)
(2396, 0.3493211908282323, 5.0)
```

Teil b | Interpretation der Ergebnisse

Schaue dir nochmal die k nächsten Nachbarn des Test-Windows aus Teil a an. Welche Aktivität würdest du dem von dir gewählten Test-Window zuordnen und warum? Notiere deine Überlegungen auf deinem Antwortblatt.

Aufgabe 4 | Häufigkeiten der einzelnen Klassen

Wir können nun die k nächsten Nachbarn zu einem Window bestimmen. Um diesem Window eine Aktivität zuordnen zu können, müssen wir noch einen **Mehrheitsentscheid** unter den k nächsten Nachbarn durchführen. Wir bestimmen also die Aktivität, die unter den k nächsten Nachbarn am häufigsten vorkommt. Die am häufigsten vorkommende Aktivität ordnen wir dann unserem Window zu.

Teil a | Mehrheitsentscheid

Ersetze das erste `NaN` *durch die Anzahl der Nachbarn, die du miteinbeziehen möchtest, und das zweite* `NaN` *durch die WindowID des Test-Windows, für das du die Aktivität über einen Mehrheitsentscheid bestimmen möchtest.*

In [5]:
```
k = 4;
TestWindow = 250;

# Hier nichts ändern!
PrintClassVote(k, TestWindow)
```

```
Unter den 4 nächsten Nachbarn des Test-Windows mit der WindowID 250 kommt die Aktivität mit der ActivityID 5.0
am häufigsten vor.
```

Teil b | Überprüfung des Ergebnisses

Zum Abschluss können wir noch überprüfen, ob die Aktivität, die wir dem Test-Window über den k-nächste-Nachbarn-Algorithmus zugeordnet haben, auch der Aktivität entspricht, zu der die Daten aufgenommen wurden.

Ersetze das `NaN` *durch die WindowID des Test-Windows, für das du zuvor den Mehrheitsentscheid in Teil a durchgeführt hast. Führe anschließend den Code aus. Dir wird die tatsächliche ActivityID des gewählten Test-Windows ausgegeben.*

Stimmen die beiden Aktivitäten von Teil a und Teil b überein? Notiere deine Ergbnisse auf deinem Antwortblatt.

In [6]:
```
TestWindow = 250;

# Hier nichts ändern!
PrintActivityID(TestWindow)
```

```
Das Test-Window mit der WindowID 250 enthält Daten zur Aktivität mit der ActivityID 5.0
```

Fazit

Du hast auf diesem Arbeitsblatt den k-nächste-Nachbarn-Algorithmus kennengelernt und bereits erste Windows mithilfe dieses Algorithmus einer Aktivität zugeordnet. Auf dem nächsten Arbeitsblatt werden wir uns anschauen, wie wir die Leistung unseres Algorithmus bewerten können.

Diskussion im Plenum

Wir werden nun die Erkenntnisse und Ergebnisse von Arbeitsblatt 3 gemeinsam im Plenum besprechen.

Öffne nun das nächste Arbeitsblatt.

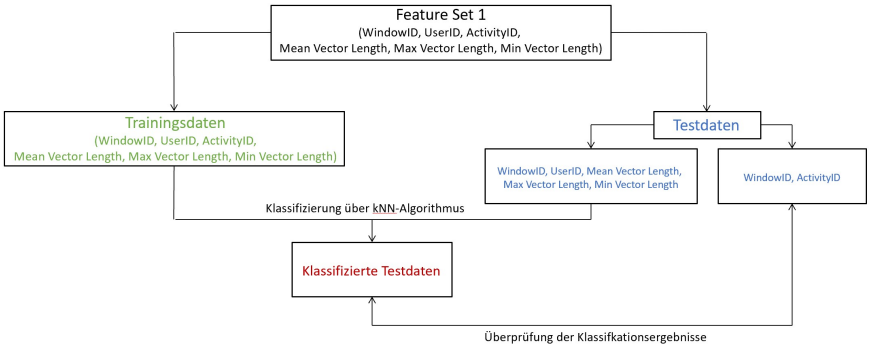
B.6. Arbeitsblatt 4

Arbeitsblatt 4: Bewertung der Klassifikationsergebnisse

Notiere deine Ergebnisse während dem Bearbeiten des Arbeitsblattes auf diesem Antwortblatt.

Einleitung:

Nachdem wir auf Arbeitsblatt 3 den k-nächste-Nachbarn-Algorithmus kennengelernt und implementiert haben, werden wir auf diesem Arbeitsblatt **bewerten**, wie gut unser Algorithmus funktioniert. Dazu nutzen wir eine Strategie aus dem Bereich des **überwachten Maschinellen Lernens (engl. supervised learning)** und teilen in einem ersten Schritt unsere Daten in **Trainings-** und **Testdaten** auf.



In unserem Fall bestehen die Trainingsdaten aus einem Großteil der Daten des Features-Sets 1. Der Rest der Daten, die wir nicht den Trainingsdaten zuordnen, sind unsere Testdaten. Zunächst blenden wir die tatsächlichen Aktivitäten der jeweiligen Datenpunkte im Testdatensatz aus. Wir gehen also davon aus, dass wir die zugehörige Aktivität eines Datenpunktes im Testdatensatz nicht kennen. Anschließend klassifizieren wir die Testdatenpunkte mit Hilfe des k-nächste-Nachbarn-Algorithmus. Das heißt wir berechnen für alle Datenpunkte im Testdatensatz die k nächsten Nachbarn aus dem Trainingsdatensatz und bestimmen über einen Mehrheitsentscheid die Aktivitäten der Testdatenpunkte. Im Anschluss überprüfen wir, ob unsere Klassifizierung mit den tatsächlichen Aktivitäten der einzelnen Testdatenpunkte übereinstimmt. Mithilfe verschiedener **Qualitätsmaße** können wir dann die Güte des Klassifikationsalgorithmus beurteilen.

Auf diesem Arbeitsblatt lernst du verschiedene Möglichkeiten zur Bewertung der Güte eines Klassifikationsalgorithmus kennen.

Aufgabe 1 | Die Trainings- und Testdaten

Als erstes werden wir das Feature-Set 1 in Trainings- und Testdaten unterteilen. Da wir diesmal die Klassifizierung komplett dem Computer übergeben möchten, müssen wir unsere Daten zuvor noch in **Features** (Mittelwert, Maximum und Minimum des Betrags des Beschleunigungsvektors) und **Klassen** (Aktivitäten) aufteilen.

Teil a | Unterteilung in Features und Klassen

Um später die Daten dem Klassifikationsalgorithmus übergegeben zu können, müssen wir zunächst die Daten in sogenannten **Arrays** abspeichern. In einem Array werden unsere Daten in Form einer Liste gespeichert. Wir benötigen einen Array, der alle Features beinhaltet, und einen zweiten Array, in dem die Aktivitäten, abgespeichert sind.

Führe den Code aus. Dir wird sowohl der Array, in dem alle Features gespeichert sind, als auch der Array, in dem die Aktivitäten abgespeichert sind, ausgegeben.

In [1]:
```
# Hier nichts ändern!
```

```
import sys; sys.path.append("../code"); from setupAB4 import *;
printFeatureArray()
printClassArray()
```

```
Feature-Array:
 [[ 8.38218932  8.59341286  8.13696711]
 [ 8.33561716  8.57186579  8.05273673]
 [ 8.28963309  8.38448555  8.05906408]
 ...
 [13.00454819 24.82067965  3.9303966 ]
 [13.72327962 26.34659082  4.74543854]
 [12.2734054  27.15370122  4.56542116]]
Activity-Array:
 [1. 1. 1. ... 5. 5. 5.]
```

Teil b | Unterteilung in Trainings- und Testdaten

Nachdem wir die Daten in Features und Aktivitäten unterteilt haben, können wir auch die Trennung in Trainings- und Testdaten vornehmen. Dazu müssen wir festlegen, wie viel Prozent unserer Daten dem Testdatensatz zugeordnet werden sollen. In der Praxis werden meistens 70 % bis 90 % der Daten dem Trainingsdatensatz zugeordnet. Wir legen die Größe des Testdatensatzes auf 20 % fest. Die Auswahl der Daten, die dem Testdatensatz zugeordnet werden, erfolgt zufällig.

Führe den Code aus.

In [2]:
```
# Hier nichts ändern!
printTrainTestSplit(XSet1_train, XSet1_test)
```

```
Die Daten wurden in Trainings- und Testdaten unterteilt.
Der Trainingsdatensatz umfasst 2400.0 Datenpunkte und der Testdatensazt beinhaltet 600.0 Datenpunkte.
```

Aufgabe 2 | Die Wahrheitsmatrix

Wir können nun die Daten unserem Klassifikationsalgorithmus, dem k-nächste-Nachbarn-Algorithmus, übergeben. Der Algorithmus ordnet jedem Datenpunkt des Testdatensatzes mithilfe der Trainingsdaten eine Aktivität zu. Anschließend wird überprüft, ob die über den Klassifikationsalgorithmus bestimmten Aktivitäten mit den tatsächlichen Aktivitäten übereinstimmen. Die Ergebnisse können wir uns anschließend in Form einer Tabelle ausgeben lassen. Die Tabelle ist wie folgt aufgebaut:

	Klassifiziert als 1 (Sitzen)	Klassifiziert als 2 (Stehen)	Klassifiziert als 3 (Gehen)	Klassifiziert als 4 (Laufen)	Klassifiziert als 5 (Treppen auf- und absteigen)
Tatsächlich 1 (Sitzen)					
Tatsächlich 2 (Stehen)					
Tatsächlich 3 (Gehen)					
Tatsächlich 4 (Laufen)					
Tatsächlich 5 (Treppen auf- und absteigen)					

Diese Tabelle bezeichnet man auch als **Wahrheitsmatrix** oder **Konfusionsmatrix**. Sie gibt die Anzahl der Datenpunkte einer Aktivität an, die richtig bzw. falsch klassifiziert wurden. Falls Datenpunkte falsch klassifiziert wurden, können wir mit Hilfe der Wahrheitsmatrix auch erkennen, welcher anderen Aktivität die Datenpunkte fälschlicherweise zugeordnet wurden.

Teil a | Die Wahrheitsmatrix zur Klassifikation mit Feature-Set 1

Führe den Code aus. Dir wird die Wahrheitsmatrix für die Klassifikation mit $k = 3$ und dem Feature-Set 1 ausgegeben.

In [3]:
```
# Hier nichts ändern!
showKonfusionMatrix(ConfusionMatrix)
```

```
Die Wahrheitsmatrix zur Klassifikation mit k =3 und Feature-Set 1:
 [[123   0   0   0   0]
 [  0 115   0   0   0]
 [  0   0  79  10  19]
 [  0   0  14 120   1]
 [  1   0  24   1  93]]
```

Teil b | Interpretation der Wahrheitsmatrix

Ergänze die Werte der Wahrheitsmatrix (Teil a) in der Tabelle auf deinem Antwortblatt. Notiere dir auch die Größe des Testdatensatzes (Aufgabe 1 | Teil b) und das gewählte k. Beantworte anschließend die folgenden Fragen ebenfalls auf deinem Antwortblatt:

1. Wie viele Datenpunkte wurden insgesamt richtig klassifiziert?
2. Wie viele Datenpunkte wurden insgesamt falsch klassifiziert?
3. Bei welcher Aktivität wurden die meisten Datenpunkte richtig klassifiziert?
4. Bei welcher Aktivität wurden die meisten Datenpunkte falsch klassifiziert?
5. Bei welchen beiden Aktivitäten liegen die meisten Überschneidungen vor?

Aufgabe 3 | Verschiedene Qualitätsmaße

Wir werden unsere Ergebnisse noch mit Hilfe verschiedener **Qualitätsmaße** bewerten.

Teil a | Die Genauigkeit

Als erstes wollen wir uns die **Genauigkeit** (engl. **accuracy**) anschauen. Sie gibt das Verhältnis der richtig klassifizierten Datenpunkte zu allen Datenpunkten an:

$$\text{accuracy} = \frac{\text{Anzahl der richtig klassifizierten Datenpunkte}}{\text{Anzahl aller Datenpunkte}}$$

Ersetze das `NaN` durch eine Formel, mit der du die Genauigkeit der Klassifikation mit Hilfe der oben ausgegebenen Wahrheitsmatrix berechnen kannst. Führe anschließend den Code aus.

Hinweise zur Eingabe im Codefeld:

- Du kannst auf die einzelnen Einträge der Wahrheitsmatrix mit `CM_ij` zugreifen. Dabei bezeichnet i die Zeilennummer und j die Spaltennummer. `CM_23` bezeichnet dann den dritten Eintrag in der zweiten Zeile.
- Du kannst auf die Summe aller Einträge in der Wahrheitsmatrix mit `CM_Sum` zugreifen.

Falls du Hilfe benötigst, kannst du dir hier einen Tipp ansehen.

In [4]:
```
accuracy = (CM_11 + CM_22 + CM_33 + CM_44 + CM_55) / CM_Sum;

# Hier nichts ändern!
checkAccuracy(Accuracy, accuracy)
```

```
✓ Deine Formel für die Genauigkeit ist richtig. Die Genauigkeit beträgt 0.8833333333333333
```

Teil b | Die Fehlerrate

Als nächstes schauen wir uns die **Fehlerrate** (engl. **error rate**) an. Sie gibt das Verhältnis der falsch klassifizierten Datenpunkte zu allen Datenpunkten an:

$$\text{error rate} = \frac{\text{Anzahl der falsch klassifizierten Datenpunkte}}{\text{Anzahl aller Datenpunkte}}$$

Ersetze das `NaN` *durch eine Formel, mit der du die Fehlerrate der Klassifikation mit Hilfe der oben ausgegebenen Wahrheitsmatrix berechnen kannst. Führe anschließend den Code aus.*

Falls du Hilfe benötigst, kannst du dir hier einen Tipp ansehen.

In [5]:
```
errorRate = 1 - accuracy;

# Hier nichts ändern!
checkErrorRate(ErrorRate, errorRate)
```

```
✓ Deine Formel für die Fehlerrate ist richtig. Die Fehlerrate beträgt 0.1166666666666667
```

Teil c | Die Präzision

Als letztes Qualitätsmaß werden wir uns noch die **Präzision** (engl. **precision**) anschauen. Sie gibt das Verhältnis der richtig als Aktivität z klassifizierten Datenpunkte zu allen als Aktivität z klassifizierten Datenpunkten an:

$$\text{precision}_z = \frac{\text{Anzahl der richtig als Aktivität } z \text{ klassifizierten Datenpunkte}}{\text{Anzahl aller als Aktivität } z \text{ klassifizierten Datenpunkte}}$$

Ersetze die `NaN's` *durch Formeln, mit denen du die Präzision der einzelnen Aktivitäten mit Hilfe der oben ausgegebenen Wahrheitsmatrix berechnen kannst. Führe anschließend den Code aus.*

Falls du Hilfe benötigst, kannst du dir hier einen Tipp ansehen.

In [6]:
```
precisionSitzen = CM_11 / (CM_11 + CM_21 + CM_31 + CM_41 + CM_51);
precisionStehen = CM_22 / (CM_12 + CM_22 + CM_32 + CM_42 + CM_52);
precisionGehen = CM_33 / (CM_13 + CM_23 + CM_33 + CM_43 + CM_53);
precisionLaufen = CM_44 / (CM_14 + CM_24 + CM_34 + CM_44 + CM_54);
precisionTreppen = CM_55 / (CM_15 + CM_25 + CM_35 + CM_45 + CM_55);

# Hier nichts ändern!
checkPrecision(precisionSitzen, precisionStehen, precisionGehen, precisionLaufen, precisionTreppen, Pre1, Pre2,
```

```
✓ Deine Formel für die Präzision der Aktivität Sitzen ist richtig.
Die Präzision der Aktivität Sitzen beträgt: 0.9919354838709677
✓ Deine Formel für die Präzision der Aktivität Stehen ist richtig.
Die Präzision der Aktivität Stehen beträgt: 1.0
✓ Deine Formel für die Präzision der Aktivität Gehen ist richtig.
Die Präzision der Aktivität Gehen beträgt: 0.6752136752136753
✓ Deine Formel für die Präzision der Aktivität Laufen ist richtig.
Die Präzision der Aktivität Laufen beträgt: 0.916030534351145
✓ Deine Formel für die Präzision der Aktivität Treppen auf- und absteigen ist richtig.
Die Präzision der Aktivität Treppen auf- und absteigen beträgt: 0.8230088495575221
```

Teil d | Interpretation der Qualitätsmaße

Notiere dir die Werte der verschiedenen Qualitätsmaße auf deinem Antwortblatt. Beantworte im Anschluss die folgenden Fragen ebenfalls auf deinem Antwortblatt:

1. Was bedeutet es, wenn der Wert der Präzision nahe bei 1 liegt? Bei welchen Aktivitäten ist dies der Fall?
2. Bei welcher Aktivität werden die Datenpunkte am häufigsten richtig zugeordnet und warum?
3. Bei welchen beiden Aktivitäten werden die Datenpunkte am häufigsten falsch zugeordnet und was könnten Gründe dafür sein?
4. Sind die Ergebnisse der Klassifikation im Großen und Ganzen zufriedenstellend? Falls nein, wo müssten die Ergebnisse noch verbessert werden?

Aufgabe 4 | Ideen zur Verbesserung

Die Ergebnisse unserer Klassifikation sind leider noch nicht ganz zufriedenstellend. Daher müssen wir unser Vorgehen nochmal überdenken.

Sammle Ideen, was wir an unserem bisherigen Vorgehen verändern könnten, um bessere Ergebnisse zu erhalten. Notiere deine Überlegungen auf deinem Antwortblatt.

Fazit

Du hast nun die Güte der Klassifikation mit $k = 3$ und dem Feature-Set 1 bewertet. Auf den nächsten beiden Arbeitsblättern werden wir versuchen die Ergebnisse unserer Klassfikation durch Veränderungen an unserem bisherigen Vorgehen zu verbessern.

Diskussion im Plenum

Wir werden nun die Erkenntnisse und Ergebnisse von Arbeitsblatt 4 gemeinsam im Plenum besprechen.

Öffne nun das nächste Arbeitsblatt.

Schon fertig?

Untersuche auf diesem Zusatzblatt, wie sich die Ergebnisse der Klassifikation verändern, wenn andere Abstandsfunktionen genutzt werden.

Autorin: Katja Hoeffer

B.7. Arbeitsblatt 5

Arbeitsblatt 5: Erweiterung des Feature-Sets 1

Notiere deine Ergebnisse während dem Bearbeiten des Arbeitsblattes auf diesem Antwortblatt.

Einleitung:

Auf Arbeitsblatt 4 haben wir die Klassifikation mit $k = 3$ und dem Feature-Set 1 bewertet und festgestellt, dass die Aktivitäten 3 (Gehen) und 5 (Treppen auf- und absteigen) noch nicht zufriedenstellend erkannt werden. Daher werden wir auf diesem Arbeitsblatt versuchen, die Klassifikation durch eine Erweiterung des Feature-Sets 1 zu verbessern.

Auf diesem Arbeitsblatt wirst du das Feature-Set 1 um weitere aussagekräftige Features erweitern.

Aufgabe 1 | Die erste Erweiterung

Als erste Erweiterung ergänzen wir unser Feature-Set 1 um den Mittelwert, das Maximum und das Minimum jeder der drei Beschleunigungsrichtungen x, y und z. Die neuen Features fügen wir als zusätzliche Spalten unserem Feature-Set 1 hinzu.

Führe den Code aus. Dir werden die ersten und die letzten fünf Zeilen des erweiterten Feature-Sets 1 ausgegeben.

In [1]:
```
# Hier nichts ändern!
import sys; sys.path.append("../code"); from setupAB5 import *;
printFeatureSet2_1()
```

```
Die ersten 5 Zeilen des erweiterten Feature-Sets 1:
```

WindowID	UserID	ActivityID	Mean Vector Length (m/s^2)	Max Vector Length (m/s^2)	Min Vector Length (m/s^2)	Mean x (m/s^2)	Max x (m/s^2)	Min x (m/s^2)	Mean y (m/s^2)	Max y (m/s^2)	Min y (m/s^2)	Mean (m/s^2
1.0	1.0	1.0	8.382189	8.593413	8.136967	5.505176	5.615423	5.450765	-4.469158	-4.271966	-4.689299	-6.87824
2.0	1.0	1.0	8.335617	8.571866	8.052737	5.519143	5.657186	5.373675	-4.416821	-4.172723	-4.578230	-6.90096
3.0	1.0	1.0	8.289633	8.384486	8.059064	5.486834	5.620662	5.331613	-4.393822	-4.272116	-4.490362	-6.94325
4.0	1.0	1.0	8.262474	8.533645	7.956085	5.509859	5.862559	5.325925	-4.353382	-4.147874	-4.539909	-6.94871
5.0	1.0	1.0	8.268336	8.325962	8.224459	5.513971	5.550308	5.474566	-4.356670	-4.315825	-4.397256	-6.94635

```
Die letzten 5 Zeilen des erweiterten Feature-Sets 1:
```

WindowID	UserID	ActivityID	Mean Vector Length (m/s^2)	Max Vector Length (m/s^2)	Min Vector Length (m/s^2)	Mean x (m/s^2)	Max x (m/s^2)	Min x (m/s^2)	Mean y (m/s^2)	Max y (m/s^2)	Min y (m/s^2)	M (m
2996.0	10.0	5.0	13.059342	29.251910	4.153431	2.393618	6.628067	-0.037123	-9.037663	-2.933300	-20.402273	-2.9
2997.0	10.0	5.0	13.351763	29.893499	5.658197	3.052110	7.301067	0.417632	-9.145451	-3.231779	-20.826491	-2.0
2998.0	10.0	5.0	13.004548	24.820680	3.930397	2.477606	5.964497	-0.236957	-8.984599	-2.712060	-17.457447	-2.6
2999.0	10.0	5.0	13.723280	26.346591	4.745439	3.085341	6.180348	-0.250130	-9.411966	-2.899471	-18.516495	-2.3
3000.0	10.0	5.0	12.273405	27.153701	4.565421	2.515001	6.569688	0.141456	-8.458025	-3.097060	-18.827698	-3.4

Aufgabe 2 | Die zweite Erweiterung

Als zweite Erweiterung möchten wir unserem erweiterten Feature-Set 1 noch drei neue statistische Kenngrößen hinzufügen.

Teil a | Weitere statistische Kenngrößen

Notiere auf deinem Antwortblatt min. drei statistische Kenngrößen, die du neben dem Mittelwert, dem Maximum und dem Minimum bereits aus dem Mathematikunterricht kennst.

Teil b | Die neuen statistischen Kenngrößen

Bei den zusätzlichen Features beschränken wir uns wieder auf den Betrag des Beschleunigungsvektors. Klicke auf den Pfeil um die neuen statistischen Kenngrößen zu sehen.

▶ **neue statistische Kenngrößen**

Diese Kenngrößen werden wir nun beispielhaft für den Betrag des Beschleunigungsvektors von sieben Datenpunkten aus dem Window mit der WindowID 121 berechnen.

WindowID	UserID	ActivityID	Time(s)	x (m/s^2)	y (m/s^2)	z (m/s^2)	Vector Length (m/s^2)
121	1	3	0.0	-1.1	-11.2	0.1	11.3
121	1	3	0.5	-1.3	-8.9	-3.9	9.8
121	1	3	1.0	2.0	-1.1	-1.2	2.6
121	1	3	1.5	-6.1	-6.0	-4.5	9.7
121	1	3	2.0	-8.9	-8.2	-3.7	12.6
121	1	3	2.5	2.6	-1.1	-2.5	3.8
121	1	3	3.0	0.0	-5.7	0.3	5.7

Bestimme für die Werte in der Tabelle den Median, das obere Quartil und das untere Quartil für den Betrag des Beschleunigungsvektors. Ersetze die `NaN` *durch Formeln bzw. Lösungen. Führe anschließend den Code aus.*

Falls du Hilfe benötigst, kannst du dir hier einen Tipp ansehen.

In [2]:
```
# Median Vector Length
medianVectorLength = 9.7;

# oberes Quartil Vector Length
upperQuartileVectorLength = 11.3;

# unteres Quartil Vector Length
lowerQuartileVectorLength = 3.8;

# Hier nichts ändern!
checkNewFeatures(medianVectorLength, upperQuartileVectorLength, lowerQuartileVectorLength)
```

```
✓ Deine Lösung für den Median ist richtig.
✓ Deine Lösung für das obere Quartil ist richtig.
✓ Deine Lösung für das untere Quartil ist richtig.
```

Teil c | Automatische Berechnung der neuen statistischen Kenngrößen

Da jedes Window ca. 120 Datenpunkte enthält, wäre es ziemlich aufwändig, für jedes Window die neuen statistischen Kenngrößen per Hand auszurechnen. Stattdessen können wir diese Aufgabe dem Computer übergeben.

Ersetze das `NaN` *durch eine beliebige WindowID zwischen 1 und 3000 und führe den Code aus. Dir werden die statistischen Kenngrößen Median, oberes Quartil und unteres Quartil für den Betrag des Beschleunigungsvektors des gewählten Windows ausgegeben.*

In [3]:
```
WindowID = 260;
```

```
# Hier nichts ändern!
printNewFeatures(WindowID)
```

```
WindowID: 260 , UserID: 1.0 , ActivityID: 5.0
----
Median Vector Length: 12.185910045879641
oberes Quartil Vector Length: 16.90316901894427
unteres Quartil Vector Length: 9.262339900920665
```

Teil d | Abspeichern der neuen statistischen Kenngrößen

Die neuen Features Median, oberes Quartil und unteres Quartil des Beschleunigungsvektors fügen wir unserem erweiterten Feature-Set 1 als zusätzliche Spalten hinzu und erhalten unser neues Feature-Set, das **Feature-Set 2**.

Führe den Code aus. Dir werden die ersten und die letzten fünf Zeilen des Feature-Sets 2 ausgegeben.

In [4]:
```
# Hier nichts ändern!
printFeatureSet2()
```

```
Die ersten 5 Zeilen des Feature-Sets 2:
```

WindowID	UserID	ActivityID	Mean Vector Length (m/s^2)	Max Vector Length (m/s^2)	Min Vector Length (m/s^2)	Median Vector Length (m/s^2)	Upper Quartile Vector Length (m/s^2)	Lower Quartile Vector Length (m/s^2)	Mean x (m/s^2)	Max x (m/s^2)	Min x (m/s^2)	Mean y (m/s^2)
1.0	1.0	1.0	8.382189	8.593413	8.136967	8.385889	8.437079	8.323079	5.505176	5.615423	5.450765	-4.469158
2.0	1.0	1.0	8.335617	8.571866	8.052737	8.338597	8.377425	8.323079	5.519143	5.657186	5.373675	-4.416821
3.0	1.0	1.0	8.289633	8.384486	8.059064	8.290660	8.312062	8.323079	5.486834	5.620662	5.331613	-4.393822
4.0	1.0	1.0	8.262474	8.533645	7.956085	8.275744	8.313050	8.323079	5.509859	5.862559	5.325925	-4.353382
5.0	1.0	1.0	8.268336	8.325962	8.224459	8.269243	8.277988	8.323079	5.513971	5.550308	5.474566	-4.356670

```
Die letzten 5 Zeilen des Feature-Sets 2:
```

WindowID	UserID	ActivityID	Mean Vector Length (m/s^2)	Max Vector Length (m/s^2)	Min Vector Length (m/s^2)	Median Vector Length (m/s^2)	Upper Quartile Vector Length (m/s^2)	Lower Quartile Vector Length (m/s^2)	Mean x (m/s^2)	Max x (m/s^2)	Min x (m/s^2)	Me (m/s
2996.0	10.0	5.0	13.059342	29.251910	4.153431	11.829134	17.457058	8.323079	2.393618	6.628067	-0.037123	-9.037
2997.0	10.0	5.0	13.351763	29.893499	5.658197	12.509836	15.154778	8.323079	3.052110	7.301067	0.417632	-9.145
2998.0	10.0	5.0	13.004548	24.820680	3.930397	12.639155	16.638583	8.323079	2.477606	5.964497	-0.236957	-8.984
2999.0	10.0	5.0	13.723280	26.346591	4.745439	13.141664	16.249636	8.323079	3.085341	6.180348	-0.250130	-9.417
3000.0	10.0	5.0	12.273405	27.153701	4.565421	11.657419	14.399818	8.323079	2.515001	6.569688	0.141456	-8.458

Aufgabe 3 | Klassifikation mit Feature-Set 2

Nach der Erweiterung des Feature-Sets 1 wollen wir überprüfen, ob die Ergänzungen der neuen Features unsere Klassifikation verbessern. Bei der Klassifikation werden nun alle 15 Features berücksichtigt. Unsere Windows entsprechen also Datenpunkten in einem 15-dimensionalen Raum.

Teil a | Ergebnisse der Klassifikation mit Feature-Set 2

Führe den Code aus. Dir werden die Wahrheitsmatrix sowie die verschiedenen Qualitätsmaße für die Klassifikation mit $k = 3$ und dem Feature-Set 2 ausgegeben.

In [5]:
```
# Hier nichts ändern!
```

```
printClassification(ConfusionMatrix, Accuracy, ErrorRate, Pre_1, Pre_2, Pre_3, Pre_4, Pre_5)
```

```
-----
Wahrheitsmatrix:
-----
 [[123   0   0   0   0]
[  0 115   0   0   0]
[  0   0 105   2   1]
[  0   0   2 133   0]
[  0   0  10   2 107]]

-----
Genauigkeit:
-----
accuracy = 0.9716666666666667

-----
Fehlerrate:
-----
error rate = 0.02833333333333332

-----
Präzision:
-----
precisionSitzen = 1.0
precisionStehen = 1.0
precisionGehen = 0.8974358974358975
precisionLaufen = 0.9708029197080292
precisionTreppen = 0.9907407407407407
```

Teil b | Interpretation der Ergebnisse

Notiere die Werte der Wahrheitsmatrix sowie die Werte der einzelnen Qualitätsmaße auf deinem Antwortblatt und beantworte anschließend die folgenden Fragen ebenfalls auf deinem Antwortblatt:

1. Was fällt dir auf, wenn du die Qualitätsmaße mit den Qualitätsmaßen der Klassifikation auf Arbeitsblatt 4 vergleichst?
2. Bei welcher Aktivität treten noch am häufigsten falsche Klassifikationen auf?
3. Wie viele von 100 Aktivitäten werden im Mittel richtig klassifiziert?

Fazit

Du hast auf diesem Arbeitsblatt dem Feature-Set 1 weitere aussagekräftige Features hinzugefügt und damit die Klassifikation verbessert. Auf dem nächsten Arbeitsblatt werden wir uns mit der Frage beschäftigen, ob wir die Klassifikation auch durch eine Veränderung des Parameters k, also der Anzahl der Nachbarn, die bei der Klassifikation berücksichtigt werden, verbessern können.

Diskussion im Plenum

Wir werden nun die Erkenntnisse und Ergebnisse von Arbeitsblatt 5 gemeinsam im Plenum besprechen.

Öffne nun das nächste Arbeitsblatt.

Autorin: Katja Hoeffer

B.8. Arbeitsblatt 6

Arbeitsblatt 6: Optimierung des Parameters k

Notiere deine Ergebnisse während dem Bearbeiten des Arbeitsblattes auf diesem Antwortblatt.

Einleitung:

Nachdem wir auf Arbeitsblatt 5 die Klassifikation durch die Erweiterung des Feature-Sets 1 verbessert haben, werden wir uns auf diesem Arbeitsblatt der Frage widmen, ob wir die Klassifikationsergebnisse auch durch die Wahl des Parameters k beeinflussen können.

In der Mathematik stellt diese Fragestellung ein sogenanntes **Optimierungsproblem** dar. Gesucht werden in der Optimierung Variablen, die eine bestimmte Zielgröße bzw. Zielfunktion maximieren oder minimieren. In unserem Fall ist die Variable der Paramter k und als Zielgröße wählen wir die Fehlerrate. Wir könnten anstelle der Fehlerrate auch die anderen Qualitätsmaße wählen, beschränken uns der Einfachheit halber aber auf die Fehlerrate. Es ergibt sich dann das folgende Optimierungsproblem:

Optimierungsproblem:
Finde die Anzahl der Nachbarn k, für die die Fehlerrate $\mathrm{error\ rate}$ minimal wird.

Auf diesem Arbeitsblatt wirst du das optimale k bestimmen, für das die Fehlerrate minimal wird.

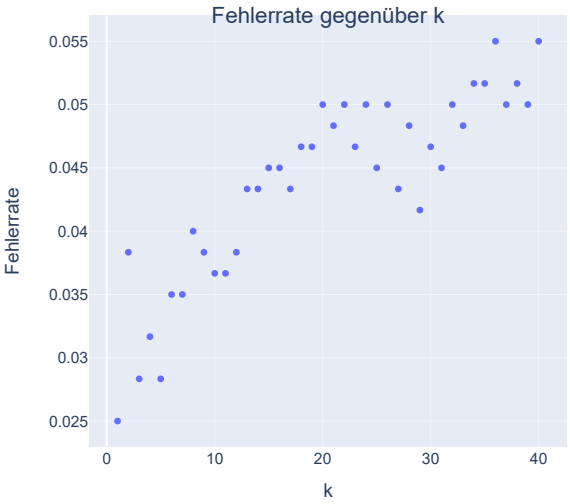
Aufgabe 1 | Graphische Lösung des Optimierungsproblems

Zunächst werden wir das optimale k graphisch bestimmen. Dazu schauen wir uns einen Graphen an, in dem die Fehlerrate $\mathrm{error\ rate}$ gegen die Anzahl der Nachbarn k aufgetragen ist.

Teil a | Der Graph

Führe den Code aus. Dir wird ein Graph, in dem die Fehlerrate $\mathrm{error\ rate}$ *gegen die Anzahl der Nachbarn* k *aufgetragen ist, ausgegeben.*

In [1]:
```
# Hier nichts ändern!
import sys; sys.path.append("../code"); from setupAB6 import *;
plotErrorRate(ErrorRates)
```



Teil b | Interpretation des Graphen

Beantworte die folgenden Fragen auf deinem Antwortblatt:

1. Für welchen Wert von k ist die Fehlerrate am kleinsten?
2. Warum schwankt die Fehlerrate für kleine k stark? Wie hängen die Schwankungen mit dem Wert von k zusammen?

Aufgabe 2 | Lösung des Optimierungsproblems mit Hilfe von Python

Als nächstes werden wir die Fehlerrate rechnerisch bzw. mit der Hilfe von Python bestimmen. Dazu werden wir die Fehlerrate für alle k zwischen 1 und 40 bestimmen und anschließend das k suchen, für das die Fehlerrate am kleinsten ist.

Erstelle mithilfe des vorgefertigten Gerüsts einen Code zur Bestimmung des optimalen k, für das die Fehlerrate am kleinsten ist. Du kannst dabei die folgenden Funktionen nutzen:

- `L.append(x)` :
  Mit dieser Funktion kannst du den Wert x der Liste `L` hinzufügen.
- `L.index(x)` :
  Diese Funktion liefert dir die Position des Werts x in der Liste `L` (**Achtung:** Python beginnt bei der Nummerierung der Einträge einer Liste bei 0).
- `computeErrorRate(i)` :
  Diese Funktion liefert dir die Fehlerrate für $k = i$.
- `min(L)` :
  Mit dieser Funktion kannst du dir den minimalen Wert einer Liste `L` ausgeben lassen.

Führe anschließend den Code aus. Wenn du alles richtig gemacht hast, werden dir das optimale k und die zugehörige Fehlerrate ausgegeben.

Falls du Hilfe benötigst, kannst du dir hier einen Tipp ansehen.

In [2]:
```
# leere Liste, in der die Fehlerraten abgespeichert werden
errorRates = []

# For-Schleife zur Berechnung der Fehlerraten und
# Abspeichern der Fehlerraten in der Liste errorRates
for i in range(1,41):
    errorRate_i = computeErrorRate(i)
    errorRates.append(errorRate_i)

# Suchen des kleinsten Werts in der Liste errorRates
min_errorRates = min(errorRates)

# Position des kleinsten Werts in der Liste errorRates
pos_min = errorRates.index(min_errorRates)

# Optimales k, für das die Fehlerrate am kleinsten ist
k = pos_min + 1

# Hier nichts ändern!
checkPosMinErrorRate(k, Min_ErrorRates, Pos_Min)
```

```
✓ Deine Lösung für das optimale k ist korrekt.
Für k = 1 ist die Fehlerrate am kleinsten und beträgt 0.025000000000000022
```

Aufgabe 3 | Klassifikation mit optimalem k

Abschließend werden wir uns noch die Ergebnisse der Klassifikation mit optimalem k anschauen.

Teil a | Ergebnisse der Klassifikation mit optimalem k

Ersetze das `NaN` *durch das zuvor bestimmte optimale k. Führe anschließend den Code aus. Dir werden die Wahrheitsmatrix sowie die verschiedenen Qualitätsmaße für die Klassifikation mit dem optimalen k und dem Feature-Set 2 ausgegeben.*

In [3]:
```
k = 1;

# Hier nichts ändern!
printClassification(k, XSet2_train, XSet2_test, ySet2_train, ySet2_test)
```

```
-----
Wahrheitsmatrix:
-----
 [[123   0   0   0   0]
[  0 115   0   0   0]
[  0   0 106   2   0]
[  0   0   2 133   0]
[  0   0  10   1 108]]

-----
Genauigkeit:
-----
accuracy = 0.975

-----
Fehlerrate:
-----
error rate = 0.025000000000000022

-----
Präzision:
-----
precisionSitzen = 1.0
precisionStehen = 1.0
precisionGehen = 0.8983050847457628
precisionLaufen = 0.9779411764705882
precisionTreppen = 1.0
```

Teil b | Interpretation der Ergebnisse

Notiere die Werte der Wahrheitsmatrix sowie die Werte der einzelnen Qualitätsmaße auf deinem Antwortblatt und beantworte anschließend die folgenden Fragen ebenfalls auf deinem Antwortblatt:

1. Was fällt dir auf, wenn du die Qualitätsmaße mit den Qualitätsmaßen der Klassifikation auf Arbeitsblatt 5 vergleichst?
2. Bei welcher Aktivität treten noch am häufigsten falsche Klassifikationen auf?
3. Wie viele von 100 Aktivitäten werden im Mittel richtig klassifiziert?

Fazit

Du hast auf diesem Arbeitsblatt die optimale Anzahl k der Nachbarn gefunden, sodass die Fehlerrate minimal wird. Auf dem nächsten Arbeitsblatt werden wir uns mit Herausforderungen und Schwierigkeiten sowie mit Anwendungen und Einsatzgebieten der Aktivitätserkennung beschäftigen.

Diskussion im Plenum

Wir werden nun die Erkenntnisse und Ergebnisse von Arbeitsblatt 6 gemeinsam im Plenum besprechen.

Öffne nun das nächste Arbeitsblatt.

Autorin: Katja Hoeffer

B.9. Arbeitsblatt 7

Arbeitsblatt 7: Probleme und Schwierigkeiten sowie Anwendungen und Einsatzgebiete der Aktivitätserkennung

Notiere deine Ergebnisse während dem Bearbeiten des Arbeitsblattes auf diesem Antwortblatt.

Einleitung:

Nachdem wir auf Arbeitsblatt 6 unseren Parameter k, also die Anzahl der nächsten Nachbarn, optimiert haben, werden wir uns auf diesem Arbeitsblatt mit Herausforderungen und Problemen sowie Anwendungen und Einsatzgebieten der Aktivitätserkennung beschäftigen.

Du wirst auf diesem Arbeitsblatt verschiedene Probleme und Herausforderungen der Aktivitätserkennung kennenlernen. Außerdem wirst du dir überlegen, in welchen Bereichen die Aktivitätserkennung zur Anwendungen kommt bzw. kommen kann.

Aufgabe 1 | Herausforderungen und Schwierigkeiten

Eine große Herausforderung der Aktivitätserkennung ist die Position und die Ausrichtung der Sensoren am Körper des Benutzers. Hält eine Person beispielsweise das Smartphone in der Hand, während er oder sie geht, unterscheiden sich die Beschleunigungsdaten deutlich von den Daten, die gemessen werden, wenn sich das Smartphone in einer der Hosentaschen befindet.

Bei der Aufnahme der Beschleunigungsdaten, für die wir auf den vorhergehenden Arbeitsblättern den Klassifikationsalgorithmus entwickelt haben, wurde das Smartphone immer in der **rechten vorderen Hosentasche** getragen. Wir werden nun überprüfen, ob unser Algorithmus auch für Beschleunigungsdaten, bei denen das Smartphone in der **Hand** gehalten wird bzw. sich in einem **Rucksack** befindet, gute Ergebnisse liefert. Hierzu führte die Testperson mit der UserID 1 die fünf verschiedenen Aktivitäten erneut jeweils drei Minuten lang aus. Diesmal befand sich das Handy aber nicht in der rechten vorderen Hosentasche sondern entweder in der Hand der Testperson oder in einem Rucksack. Die Beschleunigungsdaten sind in zwei zusätzlichen Datensätzen gespeichert. In den beiden Datensätzen wurden bereits der Betrag des Beschleunigungsvektors und die WindowID ergänzt.

Führe den Code aus. Dir werden jeweils die ersten 10 Zeilen der beiden neuen Datensätze ausgegeben.

In [1]:
```
# Hier nichts ändern!
import sys; sys.path.append("../code"); from setupAB7 import *;
showDataPos2Pos3()
```

```
Datensatz 1: Handy in der Hand des Nutzers
```

WindowID	UserID	ActivityID	Time (s)	x (m/s^2)	y (m/s^2)	z (m/s^2)	Vector Length (m/s^2)
3001	1	1	0.026879	0.117206	3.490741	9.158106	4.938044
3001	1	1	0.052034	0.188608	3.371289	9.104966	4.771452
3001	1	1	0.077188	0.292342	3.736530	9.089398	5.292332
3001	1	1	0.102343	0.383503	3.485053	9.113798	4.943507
3001	1	1	0.127497	0.294438	3.497028	9.092542	4.954301
3001	1	1	0.152652	0.260159	3.533253	9.086554	5.003542
3001	1	1	0.177807	0.267793	3.561693	9.075028	5.044109
3001	1	1	0.202961	0.216749	3.584745	9.036708	5.074227
3001	1	1	0.228116	0.149240	3.579057	9.029373	5.063751
3001	1	1	0.253271	0.112117	3.536396	9.038504	5.002476

```
-----
Datensatz 2: Handy im Rucksack des Nutzers
```

WindowID	UserID	ActivityID	Time (s)	x (m/s^2)	y (m/s^2)	z (m/s^2)	Vector Length (m/s^2)
3301	1	1	0.025215	5.189109	8.409363	1.253344	12.975423
3301	1	1	0.050370	5.208269	8.402477	1.247655	12.974178
3301	1	1	0.075526	5.195695	8.375084	1.179697	12.933650
3301	1	1	0.100682	5.195995	8.325537	1.177152	12.869635
3301	1	1	0.125838	5.156178	8.298892	1.150208	12.819103
3301	1	1	0.150993	5.083579	8.265063	1.177451	12.746187
3301	1	1	0.176149	5.059479	8.227641	1.156046	12.688044
3301	1	1	0.201305	5.077441	8.189021	1.116977	12.645178
3301	1	1	0.226461	5.118755	8.222252	1.205443	12.704822
3301	1	1	0.251616	5.091812	8.296946	1.220712	12.790824

Teil a | Vorverarbeitung der Daten

Um die Daten unserem Klassifikationsalgorithmus übergeben zu können, müssen wir für alle neuen Windows die Features Mittelwert, Maximum und Minimum für den Betrag des Beschleunigungsvektors und für alle drei Beschleunigungsrichtungen sowie den Median, das obere Quartil und das untere Quartil für den Betrag des Beschleunigungsvektors berechnen.

*Führe den Code aus. Dir werden sowohl von **Feature-Set 3** (Handy in der Hand des Nutzers) als auch von **Feature-Set 4** (Handy im Rucksack des Nutzers) die ersten und die letzten fünf Zeilen ausgegeben.*

In [2]:
```
# Hier nichts ändern!
printFeatureSet34()
```

```
Die ersten 5 Zeilen von Feature-Set 3:
```

WindowID	UserID	ActivityID	Mean Vector Length (m/s^2)	Max Vector Length (m/s^2)	Min Vector Length (m/s^2)	Median Vector Length (m/s^2)	Upper Quartile Vector Length (m/s^2)	Lower Quartile Vector Length (m/s^2)	Mean x (m/s^2)	Max x (m/s^2)	Min x (m/s^2)	Mean y (m/s^2)
3001.0	1.0	1.0	4.107287	5.292332	3.272231	4.128047	4.565631	3.630687	0.372549	0.719703	-0.059875	2.888639
3002.0	1.0	1.0	3.702255	4.596759	3.341909	3.543309	4.002226	3.470199	0.600990	0.971629	0.143252	2.581237
3003.0	1.0	1.0	2.932727	4.025974	2.218523	2.834014	3.046188	2.747502	0.573562	1.636397	-0.345482	2.010502
3004.0	1.0	1.0	3.236970	6.226262	2.565808	2.804432	3.206456	2.663097	0.707223	1.011447	-0.801882	2.222254
3005.0	1.0	1.0	7.578160	8.402050	5.339168	7.733280	7.896016	7.426527	-0.055077	0.548310	-1.029260	5.351489

```
Die letzten 5 Zeilen von Feature-Set 3:
```

WindowID	UserID	ActivityID	Mean Vector Length (m/s^2)	Max Vector Length (m/s^2)	Min Vector Length (m/s^2)	Median Vector Length (m/s^2)	Upper Quartile Vector Length (m/s^2)	Lower Quartile Vector Length (m/s^2)	Mean x (m/s^2)	Max x (m/s^2)	Min x (m/s^2)	Mean y (m/s^2)
3296.0	1.0	5.0	3.087043	5.847857	0.526257	3.025217	3.767794	2.329807	-0.341767	1.750759	-2.249971	2.023244
3297.0	1.0	5.0	2.910915	5.728526	0.828247	2.844917	3.319082	2.129652	-0.532639	1.894311	-3.064727	1.782826
3298.0	1.0	5.0	3.191214	7.097749	0.580375	2.890476	4.099980	2.047682	0.577961	2.901416	-1.678310	1.611822
3299.0	1.0	5.0	3.490647	6.673166	1.323900	3.346830	4.189879	2.687987	0.173558	2.886597	-2.265838	2.313009
3300.0	1.0	5.0	4.242730	7.082317	2.185333	4.079779	5.140122	3.408714	-0.373454	2.860851	-4.015849	2.839803

```
-----
Die ersten 5 Zeilen von Feature-Set 4:
```

WindowID	UserID	ActivityID	Mean Vector Length (m/s^2)	Max Vector Length (m/s^2)	Min Vector Length (m/s^2)	Median Vector Length (m/s^2)	Upper Quartile Vector Length (m/s^2)	Lower Quartile Vector Length (m/s^2)	Mean x (m/s^2)	Max x (m/s^2)	Min x (m/s^2)	Me (m/
3301.0	1.0	1.0	12.778882	12.975423	12.627018	12.772729	12.798779	12.747973	5.017903	5.208269	4.892276	8.31
3302.0	1.0	1.0	12.773496	13.341566	12.373738	12.766839	12.820382	12.717930	5.042310	5.420229	4.720434	8.29
3303.0	1.0	1.0	12.856205	13.742474	12.266275	12.837354	13.004385	12.725604	4.996926	5.747298	4.611760	8.37
3304.0	1.0	1.0	12.753645	12.931990	12.658920	12.748830	12.778040	12.723784	5.017454	5.149142	4.884942	8.29
3305.0	1.0	1.0	12.734785	12.803953	12.679659	12.732742	12.749986	12.717710	5.043704	5.146149	4.926705	8.26

Die letzten 5 Zeilen von Feature-Set 4:

WindowID	UserID	ActivityID	Mean Vector Length (m/s^2)	Max Vector Length (m/s^2)	Min Vector Length (m/s^2)	Median Vector Length (m/s^2)	Upper Quartile Vector Length (m/s^2)	Lower Quartile Vector Length (m/s^2)	Mean x (m/s^2)	Max x (m/s^2)	Min x (m/s^2)	(
3596.0	1.0	5.0	12.789349	23.084369	6.813902	12.133552	14.846948	9.969921	-1.273233	1.330433	-6.337820	-8
3597.0	1.0	5.0	13.656171	26.329317	6.492135	12.475431	15.641763	9.440863	-1.139693	2.592459	-7.279812	-9
3598.0	1.0	5.0	13.882503	26.235959	6.391718	11.864930	18.851976	9.732337	-1.419146	2.497556	-8.127948	-9
3599.0	1.0	5.0	14.347804	26.713494	6.503882	13.360264	19.397427	9.919602	-1.143806	2.227668	-6.642736	-10
3600.0	1.0	5.0	13.764465	28.624541	6.614979	13.221754	15.187854	11.083078	-0.958447	2.933001	-4.782405	-9

Teil b | Handy befindet sich in der Hand des Nutzers

Als nächstes werden wir überprüfen, wie gut unser Algorithmus für die Beschleunigungsdaten, bei denen das Smartphone in der Hand gehalten wurde, funktioniert. Dazu wählen wir das Feature-Set 3 als Testdatensatz und das Feature-Set 2 als Trainingsdatensatz und bewerten mit Hilfe der verschiedenen Qualitätsmaße die Klassifikationsergebnisse. Für die Anzahl der Nachbarn wählen wir $k = 1$.

Führe den Code aus. Dir werden die Wahrheitsmatrix sowie die verschiedenen Qualitätsmaße für die Klassifikation ausgegeben. Notiere die Ergebnisse auf deinem Antwortblatt.

In [3]:
```
# Hier nichts ändern!
printClassificationHand()
```

```
-----
Wahrheitsmatrix:
-----
 [[60  0  0  0  0]
[60  0  0  0  0]
[60  0  0  0  0]
[11  0 29  2 18]
[60  0  0  0  0]]

-----
Genauigkeit:
-----
accuracy = 0.20666666666666667

-----
Fehlerrate:
-----
error rate = 0.7933333333333333

-----
Präzision:
-----
precisionSitzen = 0.23904382470119523
precisionStehen = nan
precisionGehen = 0.0
precisionLaufen = 1.0
precisionTreppen = 0.0
```

Teil c | Handy befindet sich im Rucksack des Nutzers

Analog gehen wir auch für die Beschleunigungsdaten vor, bei denen sich das Handy im Rucksack des Nutzers befand, und bewerten die Ergebnisse der Klassifikation wieder mit Hilfe der verschiedenen Qualitätsmaße.

Führe den Code aus. Dir werden die Wahrheitsmatrix sowie die verschiedenen Qualitätsmaße für die Klassifikation ausgegeben. Notiere die Ergebnisse auf deinem Antwortblatt.

In [4]:
```
# Hier nichts ändern!
printClassificationBackpack()
```

```
-----
Wahrheitsmatrix:
-----
 [[60  0  0  0  0]
[ 0 60  0  0  0]
[ 0  0  0  0 60]
[ 0  0  0  0 60]
[ 0  0 26  0 34]]

-----
Genauigkeit:
-----
accuracy = 0.5133333333333333

-----
Fehlerrate:
-----
error rate = 0.4866666666666667

-----
Präzision:
-----
precisionSitzen = 1.0
precisionStehen = 1.0
precisionGehen = 0.0
precisionLaufen = nan
precisionTreppen = 0.22077922077922077
```

Teil d | Interpretation der Ergebnisse

Überlege dir, was Gründe dafür sein könnten, dass unser Algorithmus die Sensordaten, bei denen sich das Handy in der Hand bzw. im Rucksack des Nutzers befand, nicht richtig klassifiziert. Notiere deine Überlegungen auf deinem Antwortblatt.

Teil e | Weitere Herausforderungen und Schwierigkeiten

Neben dem Problem der Position und Ausrichtung der Sensoren am Körper des Nutzers gibt es noch weitere Herausforderungen und Schwierigkeiten bei der Aktivitätserkennung mit Hilfe der in Smartphones eingebauten Sensoren.

Überlege dir, was weitere Herausforderungen und Schwierigkeiten der Aktivitätserkennung sind und wie man diese möglicherweise umgehen könnte. Tausche dich mit deinem Sitznachbar aus und notiert eure Überlegungen auf diesem Antwortblatt.

Aufgabe 2 | Anwendungen und Einsatzgebiete

Trotz der bestehenden Herausforderungen und Schwierigkeiten hat die menschliche Aktivitätserkennung zahlreiche Anwendungen und Einsatzgebiete.

Überlege dir, in welchen Bereichen die menschliche Aktivitätserkennung von Nutzen ist. Gehe dabei auch auf folgende Personengruppen ein:

- Einzelpersonen / Endbenutzer
- Unternehmen
- Gruppen

Wie können die einzelnen Personengruppen von der menschlichen Aktivitätserkennung profitieren bzw. was für Anwendungen / Apps können für die einzelnen Personengruppen nützlich sein? Tausche dich mit deinem Sitznachbar aus und notiert eure Überlegungen auf diesem Antwortblatt.

Fazit

Du hast auf diesem Arbeitsblatt sowohl Schwierigkeiten und Herausforderungen als auch Anwendungen und Einsatzgebiete der Aktivitätserkennung kennengelernt.

Diskussion im Plenum

Wir werden nun die Erkenntnisse und Ergebnisse von Arbeitsblatt 7 gemeinsam im Plenum diskutieren!

Autorin: Katja Hoeffer

B.10. Zusatzblatt

Zusatzblatt: Klassifikation mit unterschiedlichen Abstandsfunktionen

Falls du auf Arbeitsblatt 3 die Zusatzaufgabe zu weiteren Abstandsfunktionen **nicht** gemacht hast, schaue dir zuerst folgendes Infoblatt an.

Auf diesem Zusatzblatt wirst du untersuchen, wie sich die Ergebnisse der Klassifikation verändern, wenn andere Abstandsfunktionen zur Berechnung des Abstandes zwischen den Datenpunkten zweier Windows verwendet werden. Als Datensatz verwenden wir unser Feature-Set 1, das wir in Trainings- und Testdaten aufgeteilt haben. Die Anzahl der Nachbarn legen wir auf $k = 3$ fest.

Aufgabe 1 | Klassifikation mit der Manhattan-Distanz

Als ertes werden wir untersuchen, wie sich die Ergebnisse der Klassifikation verändern, wenn wir anstelle der euklidischen Distanz die **Manhattan-Distanz** zur Abstandsberechnung verwenden. Mit der Manhattan-Distanz ist der Abstand zwischen den beiden Punkten $P_1(x_1, y_1, z_1)$ und $P_2(x_2, y_2, z_2)$ gegeben durch

$$d_{\text{Manhattan}} = |x_2 - x_1| + |y_2 - y_1| + |z_2 - z_1|.$$

Führe den Code aus. Dir werden die Wahrheitsmatrix sowie die verschiedenen Qualitätsmaße für die Klassifikation mit der Manhattan-Distanz ausgegeben.

In [1]:
```
# Hier nichts ändern!
import sys; sys.path.append("../code"); from setupZusatzAB import *;
printClassificationManhattan(XSet1_train, XSet1_test, ySet1_train, ySet1_test)
```

```
-----
Wahrheitsmatrix:
-----
 [[123   0   0   0   0]
[  0 115   0   0   0]
[  0   0  77   9  22]
[  0   0  16 118   1]
[  0   0  24   1  94]]

-----
Genauigkeit:
-----
accuracy = 0.8783333333333333

-----
Fehlerrate:
-----
error rate = 0.1216666666666667

-----
Präzision:
-----
precisionSitzen = 1.0
precisionStehen = 1.0
precisionGehen = 0.6581196581196581
precisionLaufen = 0.921875
precisionTreppen = 0.8034188034188035
```

Aufgabe 2 | Klassifikation mit der Kosinus-Distanz

Als nächstes führen wir die Klassifikation mit der **Kosinus-Distanz** durch und untersuchen, wie sich die Ergebnisse der Klassifikation verändern. Die Kosinus-Distanz definiert den Abstand zwischen den beiden Punkten $P_1(x_1, y_1, z_1)$ und $P_2(x_2, y_2, z_2)$ durch

$$d_{\text{Kosinus}} = 1 - \cos(\theta) = 1 - \frac{\vec{p_1} \cdot \vec{p_2}}{|\vec{p_1}| \cdot |\vec{p_2}|} = 1 - \frac{x_1 \cdot x_2 + y_1 \cdot y_2 + z_1 \cdot z_2}{\sqrt{x_1^2 + y_1^2 + z_1^2} \cdot \sqrt{x_2^2 + y_2^2 + z_2^2}}.$$

Führe den Code aus. Dir werden die Wahrheitsmatrix sowie die verschiedenen Qualitätsmaße für die Klassifikation mit der Kosinus-Distanz ausgegeben.

In [2]:
```
# Hier nichts ändern!
printClassificationCosine(XSet1_train, XSet1_test, ySet1_train, ySet1_test)
```

```
 -----
 Wahrheitsmatrix:
 -----
  [[ 80  42   0   0   1]
 [ 38  77   0   0   0]
 [  0   0  61  20  27]
 [  0   0  26 100   9]
 [  0   0  39  13  67]]

 -----
 Genauigkeit:
 -----
 accuracy = 0.6416666666666667

 -----
 Fehlerrate:
 -----
 error rate = 0.3583333333333333

 -----
 Präzision:
 -----
 precisionSitzen = 0.6779661016949152
 precisionStehen = 0.6470588235294118
 precisionGehen = 0.48412698412698413
 precisionLaufen = 0.7518796992481203
 precisionTreppen = 0.6442307692307693
```

Aufgabe 3 | Klassifikation mit der Tschebyschew-Distanz

Als letztes untersuchen wir noch, wie sich die Ergebnisse der Klassifikation verändern, wenn wir die **Tschebyschew-Distanz** zur Abstandsberechnung verwenden. Der Abstand zwischen den beiden Punkten $P_1(x_1, y_1, z_1)$ und $P_2(x_2, y_2, z_2)$ ist mit der Tschebyschew-Distanz gegeben durch

$$d_{\text{Tschebyschew}} = \max(|x_2 - x_1|, |y_2 - y_1|, |z_2 - z_1|).$$

Führe den Code aus. Dir werden die Wahrheitsmatrix sowie die verschiedenen Qualitätsmaße für die Klassifikation mit der Tschebyschew-Distanz ausgegeben.

In [3]:
```
# Hier nichts ändern!
printClassificationChebyshev(XSet1_train, XSet1_test, ySet1_train, ySet1_test)
```

```
-----
Wahrheitsmatrix:
-----
 [[123   0   0   0   0]
[  0 115   0   0   0]
[  0   0  76  12  20]
[  0   0  14 121   0]
[  0   0  20   2  97]]

-----
Genauigkeit:
-----
accuracy = 0.8866666666666667

-----
Fehlerrate:
-----
error rate = 0.11333333333333329

-----
Präzision:
-----
precisionSitzen = 1.0
precisionStehen = 1.0
precisionGehen = 0.6909090909090909
precisionLaufen = 0.8962962962962963
precisionTreppen = 0.8290598290598291
```

Aufgabe 4 | Interpretation der Ergebnisse

Beantworte die folgenden Fragen auf diesem Antwortblatt:

1. Welche Abstandsfunktion liefert die besten Klassifikationsergebnisse?
2. Welche Abstandsfunktionen liefern bessere bzw. schlechtere Klassifikationsergebnisse im Vergleich zur euklidischen Distanz?
3. Welche Abstandsfunktion liefert die schlechtesten Klassifikationsergebnisse und was könnten Gründe dafür sein?

Autorin: Katja Hoeffer

C. Infoblätter

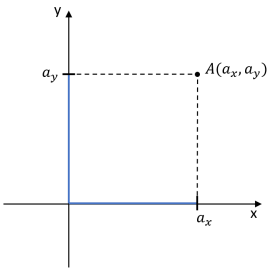
C.1. Infoblatt zu Vektoren und deren Betrag

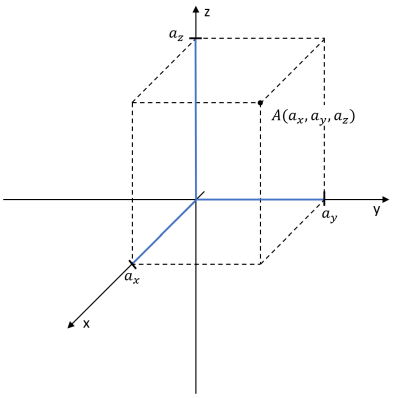
Infoblatt: Vektoren und deren Betrag

Auf diesem Infoblatt lernst du Vektoren kennen und erfährst, wie man den Betrag eines Vektors bestimmen kann.

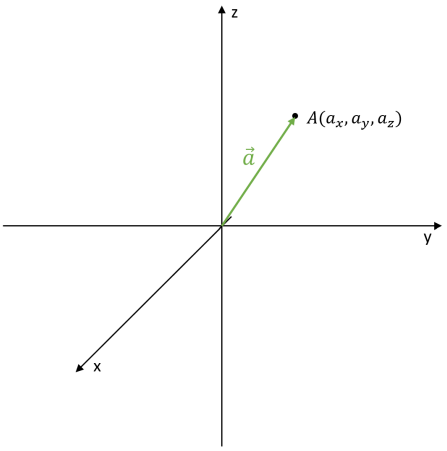
1. Vektoren

Jeder Datenpunkt in unserem Datensatz besteht aus den Beschleunigungswerten in den drei Raumdimensionen x, y und z. Wir können also jeden Datenpunkt in ein dreidimensionales Koordinatensystem mit den Achsen x, y und z eintragen. Dies erfolgt analog zur Eintragung eines Punktes in ein zweidimensionales Koordinatensystem.





Der zu dem Punkt $A(a_x, a_y, a_z)$ gehörende Vektor $\vec{a}$ ist der grüne Pfeil, der den Ursprung mit dem Punkt A verbindet.



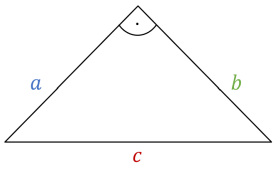
Die **Richtung** des grünen Pfeils, gibt uns die Richtung der Gesamtbeschleunigung des Smartphones an. Durch die Richtung des Pfeils können wir also bestimmen in welche Richtung das Smartphone beschleunigt wird.

Die **Länge** des grünen Pfeils beschreibt den Wert der Gesamtbeschleunigung des Smartphones. Je länger der Pfeil, umso größer ist die Gesamtbeschleunigung des Smartphones.

2. Der Betrag eines Vektors

Um den Wert der Gesamtbeschleunigung bei der späteren Auswertung der Daten nutzen zu können, müssen wir die **Länge des Vektors** (grüner Pfeil) bestimmen. Die Länge eines Vektors wird auch als **Betrag** des Vektors bezeichnet.

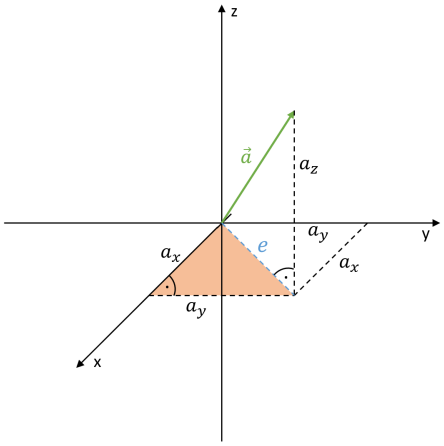
Zur Berechnung des Betrags eines Vektors benötigen wir den **Satz des Pythagoras**:
In einem rechtwinkligen Dreieck ist die Summe der quadrierten Katheten (a und b) gleich dem Quadrat der Hypotenuse (c).



$$a^2 + b^2 = c^2$$

Mit Hilfe des Satz des Pythagoras können wir dann den Betrag des Vektors $\vec{a}$ bestimmen:

Schritt 1:
Wir berechnen zuerst die Länge der blau gestrichelten Linie e, indem wir das orangene Dreieck betrachten.

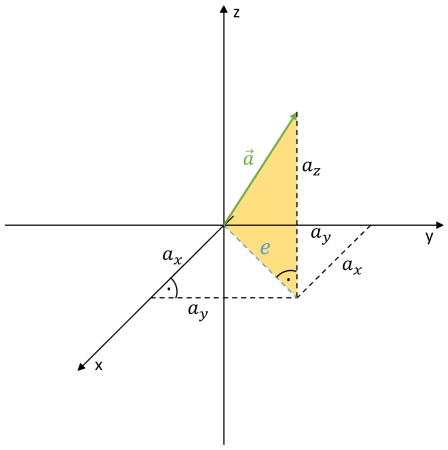


Für die Länge der blau gestrichelten Linie e gilt:

$$a_x^2 + a_y^2 = e^2 \qquad \Rightarrow \qquad e = \sqrt{a_x^2 + a_y^2}.$$

Schritt 2:

Nachdem wir die Länge der blau gestrichelten Linie e bestimmt haben, können wir mit Hilfe des Satz des Pythagoras auch die Länge des grünen Pfeils $|\vec{a}| = a$ berechnen. Dazu betrachten wir das gelbe Dreieck.



Für die Länge des grünen Pfeils gilt:

$$a^2 = e^2 + a_z^2 = \left(\sqrt{a_x^2 + a_y^2}\right)^2 + a_z^2 = a_x^2 + a_y^2 + a_z^2 \qquad \Rightarrow \qquad a = \sqrt{a_x^2 + a_y^2 + a_z^2}.$$

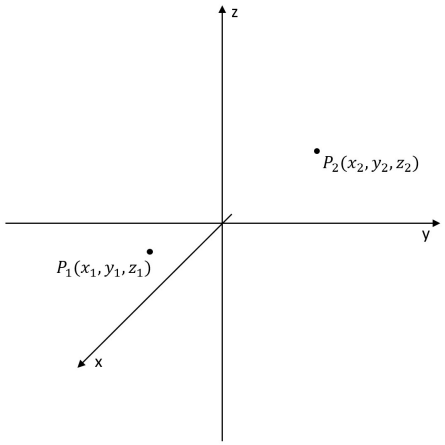
Somit gilt für den **Betrag** des Vektors $\vec{a}$:

$$|\vec{a}| = a = \sqrt{a_x^2 + a_y^2 + a_z^2}.$$

C.2. Infoblatt zur Abstandsfunktion

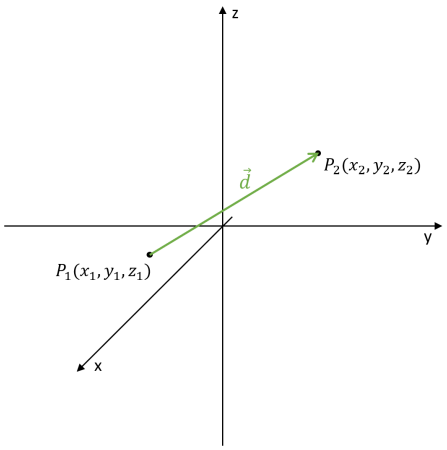
Infoblatt: Die Abstandsfunktion

Auf diesem Arbeitsblatt lernst du die Abstandsfunktion kennen, mit der wir den Abstand zwischen den beiden Punkten P_1 und P_2 bestimmen können.



Schritt 1:

Wir bilden als erstes den **Vektor** $\vec{d}$, der die beiden Punkte $P_1(x_1, y_1, z_1)$ und $P_2(x_2, y_2, z_2)$ verbindet.



Für den Vektor $\vec{d}$ gilt:

$$\vec{d} = \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{pmatrix}$$

Schritt 2:

Als nächstes bestimmen wir die **Länge** des Vektors $\vec{d}$. Dazu müssen wir den **Betrag** des Vektors $\vec{d}$ bestimmen. Es gilt:

$$|\vec{d}| = \sqrt{(x_2 - x_1)^2 + (y_2 + y_1)^2 + (z_2 + z_1)^2}.$$

Der Betrag des Vektors $\vec{d}$ liefert uns den **Abstand** zwischen den Punkten P_1 und P_2.

Hier findest du eine ausführlichere Erklärung zur Berechnung des Betrags eines Vektors.

C.3. Infoblatt zur Suche der k nächsten Nachbarn

Infoblatt: Die Bestimmung der k nächsten Nachbarn

Auf diesem Infoblatt lernst du eine Möglichkeit kennen, wie die k nächsten Nachbarn von einem ausgewählten Window bestimmt werden können.

Schritt 1:

Als erstes müssen wir die Anzahl der Nachbarn k und das Window festlegen, zu dem wir die k nächsten Nachbarn bestimmen möchten.

Führe den Code aus. Die Anzahl der Nachbarn wird auf $k = 3$ festgelegt. Als Window, zu dem die k nächsten Nachbarn bestimmt werden sollen, wird das Window mit der WindowID 200 gewählt.

In [1]:
```
# Hier nichts ändern!
import sys; sys.path.append("../code"); from setupAB3 import *;

# Schritt 1:
k = 3
TestWindow = 200
```

Schritt 2:

Als nächstes erstellen wir eine leere Liste `distances` in der nach und nach die Abstände aller Windows zum ausgewählten Test-Window mit der WindowID 200 abgespeichert werden.

Führe den Code aus.

In [2]:
```
# Schritt 2:
distances = []
```

Schritt 3:

Mit Hilfe einer for-Schleife berechnen wir die Abstände der Windows zum Test-Window und speichern sie anschließend in der Liste `distances` ab. Dazu nutzen wir die Funktionen `computeDistance(i, TestWindow)` und `addDistances(i, dist, distances)`.

Führe den Code aus.

In [3]:
```
# Schritt 3:
for i in range(1, 3000):
    dist = computeDistance(i, TestWindow) # Berechnet den Abstand zwischen dem Window mit der
                                          # WindowID i und dem Test-Window
    addDistances(i, dist, distances)      # Ergänzt den zuvor berechneten Abstand in der Liste
                                          # distances
```

Schritt 4:

Als nächstes Löschen wir mit Hilfe des Befehls `del distances[TestWindow-1]` den Abstand des Test-Windows zu sich selbst aus der Liste `distances`.

Führe den Code aus.

In [4]:
```
# Schritt 4:
del distances[TestWindow-1]
```

Schritt 5:

Anschließend sortieren wir die Einträge in der Liste `distances` der Größe nach. Dazu verwenden wir die Funktion `sortDistances(distances)`.

Führe den Code aus.

In [5]:
```
# Schritt 5:
sortDistances(distances)
```

Schritt 6:

Im letzten Schritt lassen wir uns die ersten k Einträge der Liste `distances` ausgeben. Dazu verwenden wir die Funktion `getNeighbors(distances, k)`. Die ersten k Einträge entsprechen den k nächsten Nachbarn zu unserem Test-Window.

Führe den Code aus.

In [6]:
```
# Schritt 6:
kneighbors = getNeighbors(distances, k)
print('Die', k, 'nächsten Nachbarn zu Test-Window', TestWindow, 'sind (WindowID, Abstand, ActivityID):')
for e in kneighbors:
    print(e)
```

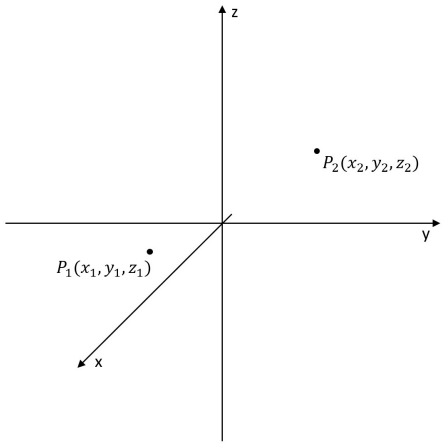
```
Die 3 nächsten Nachbarn zu Test-Window 200 sind (WindowID, Abstand, ActivityID):
(2881, 0.8596639251221669, 4.0)
(198, 0.8815728615244326, 4.0)
(206, 1.0913305848832615, 4.0)
```

Autorin: Katja Hoeffer

C.4. Infoblatt zu weiteren Abstandsfunktionen

Infoblatt: Weitere Abstandsfunktionen

Auf diesem Infoblatt lernst du verschiedene Funktionen kennen, mit denen wir den **Abstand** zwischen zwei Datenpunkten bestimmen können. Wir betrachten dazu wieder die beiden Punkte $P_1(x_1, y_1, z_1)$ und $P_2(x_2, y_2, z_2)$.

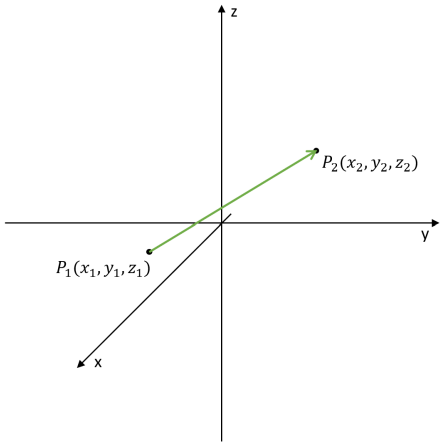


1. Euklidische Distanz

Eine Möglichkeit, den Abstand zwischen den beiden Punkten P_1 und P_2 zu bestimmen, haben wir bereits auf Arbeitsblatt 3 hergeleitet. Den Abstand d_{Euklid} haben wir definiert als

$$d_{\text{Euklid}} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}.$$

Dieser Abstand wird auch **euklidische Distanz** genannt und ist der Betrag des Verbindungsvektors der beiden Punkte P_1 und P_2 (grün eingezeichnet).



Ersetze die beiden `NaN's` *durch die WindowID's zweier beliebiger Windows. Führe anschließend den Code aus. Dir wird die euklidische Distanz zwischen den beiden Datenpunkten der von dir gewählten Windows ausgegeben.*

In [1]:
```
# Hier nichts ändern!
import sys; sys.path.append("../code"); from setupInfoAB_Abstand import *;

window1 = 12
window2 = 130

# Hier nichts ändern!
printEuclideanDistance(window1, window2)
```

```
Der Abstand zwischen dem Window mit der WindowID 12 und dem Window mit der WindowID 130 beträgt 20.919697406956
96 LE.
```

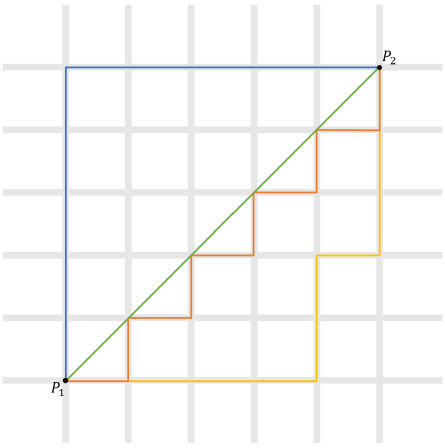
---

2. Manhattan-Distanz

Eine weitere Möglichkeit den Abstand zwischen den beiden Punkten P_1 und P_2 zu bestimmen, ist die **Manhattan-Distanz**. Hier ist der Abstand $d_{\text{Manhattan}}$ definiert als die Summe der absoluten Differenzen der Einzelkoordinaten:

$$d_{\text{Manhattan}} = |x_2 - x_1| + |y_2 - y_1| + |z_2 - z_1|.$$

Der Name stammt von der schachbrettmusterartigen Anordung der Gebäudeblöcke und dem orthogonalen Straßengitter in New Yorks Stadtteil Manhattan. Ein Taxifahrer kann die Entfernung zwischen zwei Adressen nur durch die Aneinanderreihung vertikaler und horizontaler Wegstücke überwinden (z.B. blaue, orange und gelbe Linie in der Abbildung). In der Abbildung ist zum Vergleich auch die euklidische Distanz dargestellt (grüne Linie).



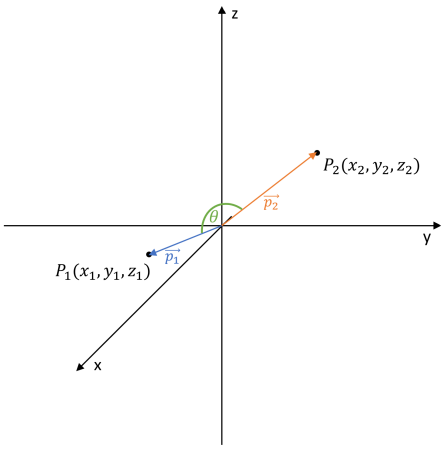
Ersetze die beiden `NaN's` *durch die WindowID's zweier beliebiger Windows. Führe anschließend den Code aus. Dir wird die Manhattan-Distanz zwischen den beiden Datenpunkten der von dir gewählten Windows ausgegeben.*

In [2]:
```
window1 = 12
window2 = 130

# Hier nichts ändern!
printManhattanDistance(window1, window2)
```

```
Der Abstand zwischen dem Window mit der WindowID 12 und dem Window mit der WindowID 130 beträgt 27.382971075267
68 LE.
```

3. Kosinus-Distanz

Der Abstand zwischen den beiden Punkten P_1 und P_2 kann auch mit Hilfe des eingeschlossenen Winkels θ zwischen den beiden Ortsvektoren $\vec{p_1}$ und $\vec{p_2}$ definiert werden.



In diesem Fall ist der Abstand d_{Kosinus} definiert als 1 minus dem Kosinus des eingeschlossenen Winkels θ:

$$d_{\text{Kosinus}} = 1 - \cos(\theta) = 1 - \frac{\vec{p_1} \cdot \vec{p_2}}{|\vec{p_1}| \cdot |\vec{p_2}|} = 1 - \frac{x_1 \cdot x_2 + y_1 \cdot y_2 + z_1 \cdot z_2}{\sqrt{x_1^2 + y_1^2 + z_1^2} \cdot \sqrt{x_2^2 + y_2^2 + z_2^2}}.$$

Diser Abstand wird als **Kosinus-Distanz** bezeichnet.

Ersetze die beiden `NaN's` durch die WindowID's zweier beliebiger Windows. Führe anschließend den Code aus. Dir wird die Kosinus-Distanz zwischen den beiden Datenpunkten der von dir gewählten Windows ausgegeben.

In [3]:
```
window1 = 12
window2 = 130

# Hier nichts ändern!
printCosineDistance(window1, window2)
```

```
Der Abstand zwischen dem Window mit der WindowID 12 und dem Window mit der WindowID 130 beträgt 0.1254323045847
7206 LE.
```

4. Tschebyschew-Distanz

Als letzte Möglichkeit, den Abstand zwischen den beiden Punkten P_1 und P_2 zu bestimmen, schauen wir uns die **Tschebyschew-Distanz** an. Sie ist definiert als das Maximum der Differenzen der Einzelkoordinaten:

$$d_{\text{Tschebyschew}} = \max(|x_2 - x_1|, |y_2 - y_1|, |z_2 - z_1|).$$

Ersetze die beiden `NaN's` durch die WindowID's zweier beliebiger Windows. Führe anschließend den Code aus. Dir wird die Tschebyschew-Distanz zwischen den beiden Datenpunkten der von dir gewählten Windows ausgegeben.

In [4]:
```
window1 = 12
window2 = 130
```

```
# Hier nichts ändern!
printChebyshevDistance(window1, window2)
```

```
Der Abstand zwischen dem Window mit der WindowID 12 und dem Window mit der WindowID 130 beträgt 20.149640238362
686 LE.
```

Autorin: Katja Hoeffer

D. Hilfekarten

D.1. Hilfekarte 1

Tipp | Dauer und Abtastrate der Datenaufnahme

Dauer der Datenaufnahme:

1. Finde in der Tabelle bei Teil a den größten Wert in der Spalte `'Time (s)'`.
2. Runde diesen Wert auf eine natürliche Zahl.

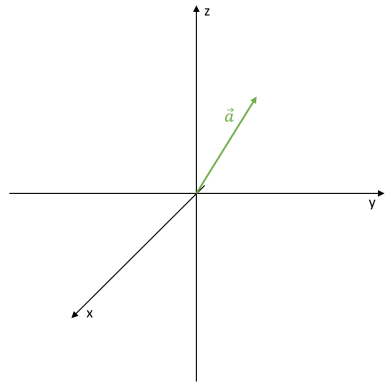
Abtastrate der Datenaufnahme:

1. Bestimme zuerst die Dauer der Datenaufnahme und die Anzahl der Datenpunkte (= Anzahl der Zeilen) zu Aktivität 1 von Testperson 1.
2. Schau dir nochmal die Erklärung der Abtastrate an.
3. Die Abtastrate gibt die Anzahl der Datenpunkte pro Zeitintervall (z.B. Dauer der Datenaufnahme) an. Überlege dir, wie du die Anzahl der Datenpunkte mit der Dauer der Datenaufnahme verrechnen musst.
4. Runde dein Ergebnis auf eine natürliche Zahl.

```
In [ ]: # Hier ist Platz für Nebenrechnungen :)
```

Tipp 1 | Berechnung des Betrags

Unter dem **Betrag eines Vektors** $\vec{a}$ versteht man die Länge des zu $\vec{a}$ gehörenden Pfeils.



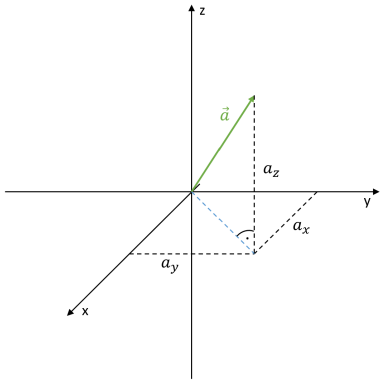
Falls du noch einen weiteren Tipp brauchst, kannst du dir hier eine weitere Hilfekarte anschauen.

Tipp 2 | Berechnung des Betrags

Unter dem **Betrag eines Vektors** $\vec{a}$ versteht man die Länge des zu $\vec{a}$ gehörenden Pfeils.

1. Berechne die Länge der blau gestrichelten Linie. Nutze hierzu den Satz des Pythagoras.
2. Berechne anschließend die Länge des grünen Pfeils. Verwende auch hier wieder den Satz des Pythagoras.

▸ **Satz des Pythagoras** (Zur Wiederholung des Satz des Pythagoras kannst du hier klicken.)



Tipp | Statistische Kenngrößen berechnen

Beispielmesswerte: 3 5 7 2 9

Mittelwert:
Der Mittelwert ist definiert als die Summe der gegebenen Werte geteilt durch die Anzahl der Werte.

Im Beispiel: $\frac{1}{5} \cdot (3+5+7+2+9) = \frac{1}{5} \cdot 26 = 5,2$

Maximum:
Das Maximum ist der größte Wert der gegebenen Werte.

Im Beispiel: 9

Minimum:
Das Minimum ist der kleinste Wert der gegebenen Werte.

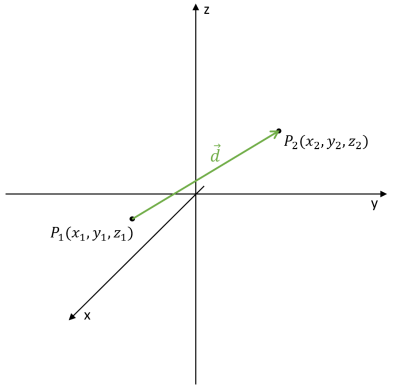
Im Beispiel: 2

Tipp 2 | Definition der Abstandsfunktion

1. Für den Vektor $\vec{d}$, der die beiden Punkte $P_1(x_1, y_1, z_1)$ und $P_2(x_2, y_2, z_2)$ verbindet, gilt:

$$\vec{d} = \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{pmatrix}$$

2. Berechne den Betrag des Vektors $\vec{d}$.

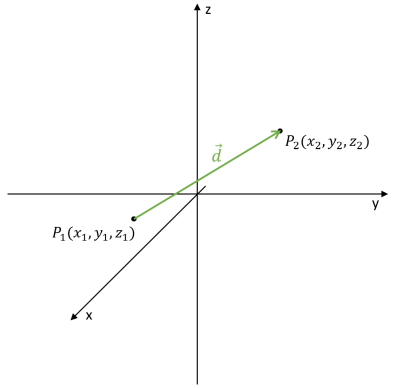


Falls du nicht mehr weißt, wie man den Betrag eines Vektors bestimmt, kannst du dir Aufgabe 1 auf Arbeitsblatt 2 oder die Hilfekarte zur Berechnung des Betrags nochmal anschauen.

Tipp 1 | Definition der Abstandsfunktion

1. Bilde als erstes den Vektor $\vec{d}$, der die beiden Punkte $P_1(x_1, y_1, z_1)$ und $P_2(x_2, y_2, z_2)$ verbindet.

2. Berechne anschließend den Betrag des Vektors $\vec{d}$.



Falls du noch einen weiteren Tipp brauchst, kannst du dir hier eine weitere Hilfekarte anschauen.

Tipp 1 | Aufstellen einer Suchfunktion

Überlege dir ein Vorgehen, wie die k nächsten Nachbarn des Test-Windows bestimmt werden können.

Hinweise:

- Wir müssen den Abstand aller Windows zum Test-Window berechnen. Hierzu kannst du eine `for`-Schleife sowie die Funktion `computeDistance(i, TestWindow)` nutzen.
- Zum Abspeichern aller Abstände bietet sich eine Liste an. Nutze dazu die Liste `distances` und die Funktion `addDistances(i, dist, distances)`.
- Zum Sortieren der Liste und zum Ausgeben der ersten k Einträge der Liste stehen dir die Funktionen `sortDistances(distances)` und `getNeighbors(distances, k)` zur Verfügung.

Falls du noch einen weiteren Tipp brauchst, kannst du dir hier eine weitere Hilfekarte anschauen.

Tipp 2 | Aufstellen einer Suchfunktion

Vorgehen zur Bestimmung der k nächsten Nachbarn des Test-Windows:

1. Erstelle eine Liste `distances` in der die Abstände aller Windows zum Test-Window gespeichert werden. Nutze dazu die `for`-Schleife sowie die beiden Funktionen `computeDistance(i, TestWindow)` und `addDistances(i, dist, distances)`.
2. Als nächstes wird mit Hilfe des Befehls `del distances[TestWindow-1]` der Abstand des Test-Windows zu sich selbst aus der Liste `distances` gelöscht.
3. Anschließend müssen die Einträge in der Liste `distances` der Größe nach sortiert werden. Dazu kannst du die Funktion `sortDistances(distances)` nutzen.
4. Im letzten Schritt müssen wir uns die ersten k Einträge unserer Liste ausgeben lassen. Diese entsprechen den k nächsten Nachbarn zu unserem Test-Window. Verwende hierzu die Funktion `getNeighbors(distances, k)`.

Tipp 2 | Die Genauigkeit

Die Genauigkeit ist definiert als das Verhältnis der richtig klassifizierten Datenpunkte zu allen Datenpunkten:

$$\text{accuracy} = \frac{\text{Anzahl der richtig klassifizierten Datenpunkte}}{\text{Anzahl aller Datenpunkte}}$$

Die jeweilige Anzahl der richtig klassifizierten Datenpunkte **einer Aktivität** ist der Diagonaleintrag der Wahrheitsmatrix (grün markiert). Mit Hilfe der Diagonaleinträge kannst du die Anzahl **aller** richtig klassifizierten Datenpunkte bestimmen.

	Klassifiziert als 1 (Sitzen)	Klassifiziert als 2 (Stehen)	Klassifiziert als 3 (Gehen)	Klassifiziert als 4 (Laufen)	Klassifiziert als 5 (Treppen auf- und absteigen)
Tatsächlich 1 (Sitzen)	CM_{11}	CM_{12}	CM_{13}	CM_{14}	CM_{15}
Tatsächlich 2 (Stehen)	CM_{21}	CM_{22}	CM_{23}	CM_{24}	CM_{25}
Tatsächlich 3 (Gehen)	CM_{31}	CM_{32}	CM_{33}	CM_{34}	CM_{35}
Tatsächlich 4 (Laufen)	CM_{41}	CM_{42}	CM_{43}	CM_{44}	CM_{45}
Tatsächlich 5 (Treppen auf- und absteigen)	CM_{51}	CM_{52}	CM_{54}	CM_{54}	CM_{55}

Tipp 1 | Die Genauigkeit

Die Genauigkeit ist definiert als das Verhältnis der richtig klassifizierten Datenpunkte zu allen Datenpunkten:

$$\text{accuracy} = \frac{\text{Anzahl der richtig klassifizierten Datenpunkte}}{\text{Anzahl aller Datenpunkte}}$$

Überlege dir, an welcher Stelle in der Wahrheitsmatrix die Anzahl der richtig klassifizierten Datenpunkte einer Aktivität zu finden ist.

	Klassifiziert als 1 (Sitzen)	Klassifiziert als 2 (Stehen)	Klassifiziert als 3 (Gehen)	Klassifiziert als 4 (Laufen)	Klassifiziert als 5 (Treppen auf- und absteigen)
Tatsächlich 1 (Sitzen)	CM_{11}	CM_{12}	CM_{13}	CM_{14}	CM_{15}
Tatsächlich 2 (Stehen)	CM_{21}	CM_{22}	CM_{23}	CM_{24}	CM_{25}
Tatsächlich 3 (Gehen)	CM_{31}	CM_{32}	CM_{33}	CM_{34}	CM_{35}
Tatsächlich 4 (Laufen)	CM_{41}	CM_{42}	CM_{43}	CM_{44}	CM_{45}
Tatsächlich 5 (Treppen auf- und absteigen)	CM_{51}	CM_{52}	CM_{54}	CM_{54}	CM_{55}

Falls du noch einen weiteren Tipp brauchst, kannst du dir hier eine weitere Hilfekarte anschauen.

Tipp 1 | Die Fehlerrate

Die Fehlerrate ist definiert als das Verhältnis der falsch klassifizierten Datenpunkte zu allen Datenpunkten:

$$\text{error rate} = \frac{\text{Anzahl der falsch klassifizierten Datenpunkte}}{\text{Anzahl aller Datenpunkte}}$$

Überlege dir, an welchen Stellen in der Wahrheitsmatrix die Anzahlen der falsch klassifizierten Datenpunkte einer Aktivität zu finden sind.

	Klassifiziert als 1 (Sitzen)	Klassifiziert als 2 (Stehen)	Klassifiziert als 3 (Gehen)	Klassifiziert als 4 (Laufen)	Klassifiziert als 5 (Treppen auf- und absteigen)
Tatsächlich 1 (Sitzen)	CM_{11}	CM_{12}	CM_{13}	CM_{14}	CM_{15}
Tatsächlich 2 (Stehen)	CM_{21}	CM_{22}	CM_{23}	CM_{24}	CM_{25}
Tatsächlich 3 (Gehen)	CM_{31}	CM_{32}	CM_{33}	CM_{34}	CM_{35}
Tatsächlich 4 (Laufen)	CM_{41}	CM_{42}	CM_{43}	CM_{44}	CM_{45}
Tatsächlich 5 (Treppen auf- und absteigen)	CM_{51}	CM_{52}	CM_{54}	CM_{54}	CM_{55}

Falls du noch einen weiteren Tipp brauchst, kannst du dir hier eine weitere Hilfekarte anschauen.

Tipp 2 | Die Fehlerrate

Die Fehlerrate ist definiert als das Verhältnis der falsch klassifizierten Datenpunkte zu allen Datenpunkten:

$$\text{error rate} = \frac{\text{Anzahl der falsch klassifizierten Datenpunkte}}{\text{Anzahl aller Datenpunkte}}$$

Die jeweiligen Anzahlen der falsch klassifizierten Datenpunkte **einer Aktivität** z sind alle Einträge der Spalte "Klassifiziert als z" außer dem Diagonaleintrag (rot markiert). Mit Hilfe dieser Einträge kannst du die Anzahl **aller** falsch klassifizierten Datenpunkte bestimmen.

	Klassifiziert als 1 (Sitzen)	Klassifiziert als 2 (Stehen)	Klassifiziert als 3 (Gehen)	Klassifiziert als 4 (Laufen)	Klassifiziert als 5 (Treppen auf- und absteigen)
Tatsächlich 1 (Sitzen)	CM_{11}	CM_{12}	CM_{13}	CM_{14}	CM_{15}
Tatsächlich 2 (Stehen)	CM_{21}	CM_{22}	CM_{23}	CM_{24}	CM_{25}
Tatsächlich 3 (Gehen)	CM_{31}	CM_{32}	CM_{33}	CM_{34}	CM_{35}
Tatsächlich 4 (Laufen)	CM_{41}	CM_{42}	CM_{43}	CM_{44}	CM_{45}
Tatsächlich 5 (Treppen auf- und absteigen)	CM_{51}	CM_{52}	CM_{54}	CM_{54}	CM_{55}

Tipp 1 | Die Präzision

Die Präzision einer Aktivität z ist definiert als das Verhältnis der richtig als Aktivität z klassifizierten Datenpunkte zu allen als Aktivität z klassifizierten Datenpunkten:

$$\text{precision}_z = \frac{\text{Anzahl der richtig als Aktivität } z \text{ klassifizierten Datenpunkte}}{\text{Anzahl aller als Aktivität } z \text{ klassifizierten Datenpunkte}}$$

Überlege dir, an welchen Stellen in der Wahrheitsmatrix die Anzahl der richtig als Aktivität z klassifizierten Datenpunkte sowie die Anzahlen aller als Aktivität z klassifizierten Datenpunkte zu finden sind.

	Klassifiziert als 1 (Sitzen)	Klassifiziert als 2 (Stehen)	Klassifiziert als 3 (Gehen)	Klassifiziert als 4 (Laufen)	Klassifiziert als 5 (Treppen auf- und absteigen)
Tatsächlich 1 (Sitzen)	CM_{11}	CM_{12}	CM_{13}	CM_{14}	CM_{15}
Tatsächlich 2 (Stehen)	CM_{21}	CM_{22}	CM_{23}	CM_{24}	CM_{25}
Tatsächlich 3 (Gehen)	CM_{31}	CM_{32}	CM_{33}	CM_{34}	CM_{35}
Tatsächlich 4 (Laufen)	CM_{41}	CM_{42}	CM_{43}	CM_{44}	CM_{45}
Tatsächlich 5 (Treppen auf- und absteigen)	CM_{51}	CM_{52}	CM_{54}	CM_{54}	CM_{55}

Falls du noch einen weiteren Tipp brauchst, kannst du dir hier eine weitere Hilfekarte anschauen.

Tipp 2 | Die Präzision

Die Präzision einer Aktivität z ist definiert als das Verhältnis der richtig als Aktivität z klassifizierten Datenpunkte zu allen als Aktivität z klassifizierten Datenpunkten:

$$\text{precision}_z = \frac{\text{Anzahl der richtig als Aktivität } z \text{ klassifizierten Datenpunkte}}{\text{Anzahl aller als Aktivität } z \text{ klassifizierten Datenpunkte}}$$

Die Anzahl der richtig als Aktivität z klassifizierten Datenpunkte ist der Diagonaleintrag der Aktivität z (für Aktivität 3 grün markiert). Die Anzahlen aller als Aktivität z klassifizierten Datenpunkte sind die Einträge der Spalte "Klassifiziert als z" (für Aktivität 3 blau markiert). Mit Hilfe dieser Einträge kannst du die Präzision für die Aktivität z berechnen.

	Klassifiziert als 1 (Sitzen)	Klassifiziert als 2 (Stehen)	Klassifiziert als 3 (Gehen)	Klassifiziert als 4 (Laufen)	Klassifiziert als 5 (Treppen auf- und absteigen)
Tatsächlich 1 (Sitzen)	CM_{11}	CM_{12}	CM_{13}	CM_{14}	CM_{15}
Tatsächlich 2 (Stehen)	CM_{21}	CM_{22}	CM_{23}	CM_{24}	CM_{25}
Tatsächlich 3 (Gehen)	CM_{31}	CM_{32}	CM_{33}	CM_{34}	CM_{35}
Tatsächlich 4 (Laufen)	CM_{41}	CM_{42}	CM_{43}	CM_{44}	CM_{45}
Tatsächlich 5 (Treppen auf- und absteigen)	CM_{51}	CM_{52}	CM_{54}	CM_{54}	CM_{55}

Tipp 1 | Die neuen statisitischen Kenngrößen

Median:
Der Median ist der mittlere Wert einer nach Größe geordneten Liste an Messwerten. Falls die Anzahl der Messwerte gerade ist, wird der Mittelwert der beiden mittleren Messwerte berechnet.

oberes Quartil:
Das obere Quartil ist der Median der oberen Hälfte der Messwerte.

unteres Quartil:
Das untere Quartil ist der Median der unteren Hälfte der Messwerte.

Falls du noch einen weiteren Tipp brauchst, kannst du dir hier eine weitere Hilfekarte anschauen.

Tipp 2 | Die neuen statisitischen Kenngrößen

Beispielmesswerte: 3 5 7 2 9

Median:
Der Median ist der mittlere Wert einer nach Größe geordneten Liste an Messwerten. Falls die Anzahl der Messwerte gerade ist, wird der Mittelwert der beiden mittleren Messwerte berechnet.

Im Beispiel:

- geordnete Liste der Messwerte: 2 3 5 7 9
- Median: 5

oberes Quartil:
Das obere Quartil ist der Median der oberen Hälfte der Messwerte.

Im Beispiel:

- obere Hälfte der Messwerte: 7 9
- oberes Quartil: $\frac{1}{2} \cdot (7 + 9) = 8$

unteres Quartil:
Das untere Quartil ist der Median der unteren Hälfte der Messwerte.

Im Beispiel:

- untere Hälfte der Messwerte: 2 3
- unteres Quartil: $\frac{1}{2} \cdot (2 + 3) = 2,5$

Tipp 1 | Lösung des Optimierungsproblems mit Hilfe von Python

Überlege dir ein Vorgehen, wie das optimale k, für das die Fehlerrate minimal wird, bestimmt werden kann.

Hinweise:

- Wir müssen die Fehlerrate für alle k zwischen 1 und 40 bestimmen. Hierzu kannst du eine `for`-Schleife sowie die Funktion `computeErrorRate(i)` nutzen.
- Zum Abspeichern aller Fehlerraten bietet sich eine Liste an. Nutze dazu die Liste `errorRates` und die Funktion `errorRates.append(x)`.
- Zur Bestimmung des minimalen Werts der Liste und dessen Position können die Funktionen `min(errorRates)` und `errorRates.index(x)` genutzt werden.

Falls du noch einen weiteren Tipp brauchst, kannst du dir hier eine weitere Hilfekarte anschauen.

Tipp 2 | Lösung des Optimierungsproblems mit Hilfe von Python

Vorgehen zur Bestimmung des optimalen k, für das die Fehlerrate minimal wird:

1. Erstelle eine Liste `errorRates`, in der die Fehlerraten für alle k zwischen 1 und 40 gespeichert sind. Nutze dazu die `for`-Schleife sowie die beiden Funktionen `computeErrorRate(i)` und `errorRates.append(x)`.
2. Bestimme als nächstes den kleinsten Wert in der Liste `errorRates`. Dazu kannst du die Funktion `min(errorRates)` nutzen.
3. Anschließend muss die Position des kleinsten Werts in der Liste `errorRates` gefunden werden. Verwende hierzu die Funktion `errorRates.index(x)`.
4. Im letzten Schritt muss das optimale k bestimmt werden. Das optimale k entspricht der Position des kleinsten Werts in der Liste `errorRates`. **Achtung:** Python beginnt bei der Nummerierung der Einträge einer Liste bei null. Du kannst also nicht einfach die Position des kleinsten Werts als das optimale k festlegen.

E. Antwortblätter

E.1. Antwortblatt 1

Antwortblatt zu Arbeitsblatt 1

Aufgabe 1 | Die Aktivitäten

Teil a | Der Datensatz und die Anzahl der Aktivitäten

- Aus welchen Daten besteht ein Datenpunkt im Datensatz (= Zeile in der Tabelle)?
- Zu wie vielen Aktivitäten wurden Daten aufgenommen?

Teil b | Die Art der Aktivitäten

- Zu welchen Aktivitäten wurden Daten aufgenommen?

Teil d | Interpretation der graphischen Darstellung

- Welche Unterschiede sind zwischen den einzelnen Aktivitäten zu erkennen?
- Bei welcher Aktivität sind am wenigsten / am meisten Ausschläge zu erkennen und warum?
- Auf welcher der drei Achsen sind die größten Ausschläge zu erkennen und warum?

Aufgabe 2 | Die Testpersonen

- Wie viele Testpersonen haben Daten aufgenommen?
- Wie viele Testpersonen sind weiblich und wie viele sind männlich?
- Wie alt ist die jüngste bzw. älteste Testperson?

_____

_____

_____

_____

Aufgabe 3 | Die Datenaufnahme

Teil b | Die Dauer und Abtastrate der Datenaufnahme

- Wie lange wurden Daten zu den jeweiligen Aktivitäten von den Testpersonen aufgenommen?
- Mit welcher Abtastrate wurden die Daten aufgenommen?

_____

_____

E.2. Antwortblatt 2

Antwortblatt zu Arbeitsblatt 2

Aufgabe 3 | Wahl der Features

Teil a | Statistische Kenngrößen

- Notiere verschiedene statistische Kenngrößen (min. 3), die du bereits aus dem Mathematikunterricht kennst.

_____

_____

_____

_____

Teil b | Graphische Darstellung

- Notiere statistische Kenngrößen, die du zur Beschreibung der Daten eines Windows nutzen würdest.

_____

_____

_____

_____

E.3. Antwortblatt 3

Antwortblatt zu Arbeitsblatt 3

Aufgabe 2 | Die Abstandsfunktion

Teil b | Interpretation der graphischen Darstellung

- Bei welchen Aktivitäten könnte der k-nächste-Nachbarn-Algorithmus bei der Klassifikation Schwierigkeiten haben und warum?

_____

_____

_____

_____

Aufgabe 3 | Suche der k nächsten Nachbarn

Teil b | Interpretation der Ergebnisse

- Welche Aktivität würdest du dem von dir gewählten Test-Window zuordnen und warum?

_____

_____

_____

_____

Aufgabe 4 | Häufigkeiten der einzelnen Klassen

Teil b | Überprüfung des Ergebnisses

- Stimmen die beiden Aktivitäten von Teil a und Teil b überein?

_____

_____

E.4. Antwortblatt 4

Antwortblatt zu Arbeitsblatt 4

Aufgabe 2 | Die Wahrheitsmatrix

Teil b | Interpretation der Wahrheitsmatrix

Größe Testdatensatz: _____ Anzahl der Nachbarn k: _____

	Klassifiziert als 1 (Sitzen)	Klassifiziert als 2 (Stehen)	Klassifiziert als 3 (Gehen)	Klassifiziert als 4 (Laufen)	Klassifiziert als 5 (Treppen auf- und absteigen)
Tatsächlich 1 (Sitzen)					
Tatsächlich 2 (Stehen)					
Tatsächlich 3 (Gehen)					
Tatsächlich 4 (Laufen)					
Tatsächlich 5 (Treppen auf- und absteigen)					

- Wie viele Datenpunkte wurden insgesamt richtig klassifiziert?

_____

- Wie vielen Datenpunkte wurden insgesamt falsch klassifiziert?

_____

- Bei welcher Aktivität wurden die meisten Datenpunkte richtig klassifiziert?

_____

- Bei welcher Aktivität wurden die meisten Datenpunkte falsch klassifiziert?

_____

- Bei welchen beiden Aktivitäten liegen die meisten Überschneidungen vor?

_____

_____

_____

_____

Aufgabe 3 | Verschiedene Qualitätsmaße

Teil d | Interpretation der Qualitätsmaße

Genauigkeit: _____ Fehlerrate: _____

Präzision Sitzen: _____ Präzision Laufen: _____

Präzision Stehen: _____ Präzision Treppen auf- und absteigen: _____

Präzision Gehen: _____

- Was bedeutet es, wenn der Wert der Präzision nahe bei 1 liegt? Bei welchen Aktivitäten ist dies der Fall?

_____

_____

_____

- Bei welcher Aktivität werden die Datenpunkte am häufigsten richtig zugeordnet und warum?

_____

- Bei welchen beiden Aktivitäten werden die Datenpunkte am häufigsten falsch zugeordnet und was könnten Gründe dafür sein?

_____

_____

_____

_____

- Sind die Ergebnisse der Klassifikation im Großen und Ganzen zufriedenstellend? Falls nein, wo müssten die Ergebnisse noch verbessert werden?

_____

_____

Aufgabe 4 | Ideen zur Verbesserung

- Was könnten wir an unserem bisherigen Vorgehen verändern, um bessere Ergebnisse zu erhalten?

_____

_____

_____

E.5. Antwortblatt 5

Antwortblatt zu Arbeitsblatt 5

Aufgabe 2 | Die zweite Erweiterung

Teil a | Weitere statistische Kenngrößen

- Notiere min. drei statistische Kenngrößen, die du neben dem Mittelwert, dem Maximum und dem Minimum bereits aus dem Mathematikunterricht kennst.

_____

_____

_____

Aufgabe 3 | Klassifikation mit Feature-Set 2

Teil b | Interpretation der Ergebnisse

	Klassifiziert als 1 (Sitzen)	Klassifiziert als 2 (Stehen)	Klassifiziert als 3 (Gehen)	Klassifiziert als 4 (Laufen)	Klassifiziert als 5 (Treppen auf- und absteigen)
Tatsächlich 1 (Sitzen)					
Tatsächlich 2 (Stehen)					
Tatsächlich 3 (Gehen)					
Tatsächlich 4 (Laufen)					
Tatsächlich 5 (Treppen auf- und absteigen)					

Genauigkeit: _____ Fehlerrate: _____

Präzision Sitzen: _____ Präzision Laufen: _____

Präzision Stehen: _____ Präzision Treppen auf- und absteigen: _____

Präzision Gehen: _____

- Was fällt dir auf, wenn du die Qualitätsmaße mit den Qualitätsmaßen der Klassifikation auf Arbeitsblatt 4 vergleichst?

_____

_____

_____

- Bei welcher Aktivität treten noch am häufigsten falsche Klassifikationen auf?

_____

- Wie viele von 100 Aktivitäten werden im Mittel richtig klassifiziert?

_____

_____

E.6. Antwortblatt 6

Antwortblatt zu Arbeitsblatt 6

Aufgabe 1 | Graphische Lösung des Optimierungsproblems

Teil b | Interpretation des Graphen

- Für welchen Wert von k ist die Fehlerrate am kleinsten?

_____

- Warum schwankt die Fehlerrate für kleine k stark? Wie hängen die Schwankungen mit dem Wert von k zusammen?

_____

_____

_____

_____

Aufgabe 3 | Klassifikation mit optimalem k

Teil b | Interpretation der Ergebnisse

	Klassifiziert als 1 (Sitzen)	Klassifiziert als 2 (Stehen)	Klassifiziert als 3 (Gehen)	Klassifiziert als 4 (Laufen)	Klassifiziert als 5 (Treppen auf- und absteigen)
Tatsächlich 1 (Sitzen)					
Tatsächlich 2 (Stehen)					
Tatsächlich 3 (Gehen)					
Tatsächlich 4 (Laufen)					
Tatsächlich 5 (Treppen auf- und absteigen)					

Genauigkeit: _____ Fehlerrate: _____

Präzision Sitzen: _____ Präzision Laufen: _____

Präzision Stehen: _____ Präzision Treppen auf- und absteigen: _____

Präzision Gehen: _____

- Was fällt dir auf, wenn du die Qualitätsmaße mit den Qualitätsmaßen der Klassifikation auf Arbeitsblatt 5 vergleichst?

_____

_____

_____

- Bei welcher Aktivität treten noch am häufigsten falsche Klassifikationen auf?

_____

- Wie viele von 100 Aktivitäten werden im Mittel richtig klassifiziert?

_____

_____

E.7. Antwortblatt 7

Antwortblatt zu Arbeitsblatt 7

Aufgabe 1 | Herausforderungen und Schwierigkeiten

Teil b | Handy befindet sich in der Hand des Nutzers

	Klassifiziert als 1 (Sitzen)	Klassifiziert als 2 (Stehen)	Klassifiziert als 3 (Gehen)	Klassifiziert als 4 (Laufen)	Klassifiziert als 5 (Treppen auf- und absteigen)
Tatsächlich 1 (Sitzen)					
Tatsächlich 2 (Stehen)					
Tatsächlich 3 (Gehen)					
Tatsächlich 4 (Laufen)					
Tatsächlich 5 (Treppen auf- und absteigen)					

Genauigkeit: _____ Fehlerrate: _____

Präzision Sitzen: _____ Präzision Laufen: _____

Präzision Stehen: _____ Präzision Treppen auf- und absteigen: _____

Präzision Gehen: _____

Teil c | Handy befindet sich im Rucksack des Nutzers

	Klassifiziert als 1 (Sitzen)	Klassifiziert als 2 (Stehen)	Klassifiziert als 3 (Gehen)	Klassifiziert als 4 (Laufen)	Klassifiziert als 5 (Treppen auf- und absteigen)
Tatsächlich 1 (Sitzen)					
Tatsächlich 2 (Stehen)					
Tatsächlich 3 (Gehen)					
Tatsächlich 4 (Laufen)					
Tatsächlich 5 (Treppen auf- und absteigen)					

Genauigkeit: _____ Fehlerrate: _____

Präzision Sitzen: _____ Präzision Laufen: _____

Präzision Stehen: _____ Präzision Treppen auf- und absteigen: _____

Präzision Gehen: _____

Teil d | Interpretation der Ergebnisse

- Was könnten Gründe dafür sein, dass der Algorithmus die Sensordaten, bei denen sich das Handy in der Hand bzw. im Rucksack des Nutzers befand, nicht richtig klassifiziert?

_____

_____

_____

_____

_____

Teil e | Weitere Herausforderungen und Schwierigkeiten

- Was sind weitere Herausforderungen und Schwierigkeiten der Aktivitätserkennung und wie könnte man diese möglicherweise umgehen?

_____

_____

_____

_____

_____

_____

_____

_____

_____

Aufgabe 2 | Anwendungen und Einsatzgebiete

- In welchen Bereichen ist die menschliche Aktivitätserkennung von Nutzen?
- Gehe dabei auch auf folgende Personengruppen ein:
  - Einzelpersonen / Endbenutzer
  - Unternehmen
  - Gruppen
- Wie können die einzelnen Personengruppen von der menschlichen Aktivitätserkennung profitieren bzw. was für Anwendungen / Apps können für die einzelnen Personengruppen nützlich sein?

_____

_____

_____

_____

_____

_____

_____

E.8. Antwortblatt Zusatzblatt

Antwortblatt zum Zusatzblatt

Aufgabe 4 | Interpretation der Ergebnisse

- Welche Abstandsfunktion liefert die besten Klassifikationsergebnisse?

- Welche Abstandsfunktionen liefern bessere bzw. schlechtere Klassifikationsergebnisse im Vergleich zur euklidischen Distanz?

- Welche Abstandsfunktion liefert die schlechtesten Klassifikationsergebnisse und was könnten Gründe dafür sein?

E.9. Zusammenfassungsarbeitsblatt

Rückblick auf die Schritte des Workshops

Schritt 1 | Erkunden des Datensatzes

- Es wurden Daten zu ____ Aktivitäten aufgenommen.
- Insgesamt haben ____ Testpersonen Daten aufgenommen.
- Jede Aktivität wurde bei der Datenaufnahme ____ Minuten ausgeführt und es wurde eine Abtastrate von ____ Hz gewählt.

Schritt 2 | Vorverarbeitung der Daten

- Um die Gesamtbeschleunigung des Smartphones zu berücksichtigen, wurde der _____ _____ zum Datensatz hinzugefügt.
- Die Rohdaten wurden in kleinere Zeitfenster (Windows) mit der Länge ____ Sekunden unterteilt.
- Daten des Feature-Sets 1: _____ _____

Schritt 3 | Schrittweise Entwicklung eines eigenen Klassifikationsalgorithmus

- Es wurde eine _____ zur Berechnung des Abstands zwischen zwei Datenpunkten (Windows) definiert.
- Mit Hilfe einer Suchfunktion werden die _____ eines Windows bestimmt.
- Um die _____ eines unbekannten Windows zu bestimmen, wird ein _____ _____ unter den k nächsten Nachbarn des unbekannten Windows durchgeführt.

Schritt 4 | Bewertung der Ergebnisse der Klassifikation

- Zu Bewertung der Ergebnisse der Klassifikation wird eine Methode aus dem Bereich des _____ _____ angewendet und die Daten in _____ und _____ unterteilt.
- Die Ergebnisse der Klassifikation werden mit Hilfe der _____ und den verschiedenen Qualitätsmaßen (_____, _____, _____) bewertet.

Schritt 5 | Verbesserung des entwickelten Klassifikationsalgorithmus

- Die Ergebnisse der Klassifikation konnten durch eine _____ des Feature-Sets 1 und die _____ des Parameters k verbessert werden.
- Optimale Anzahl der Nachbarn: ____
- Daten des Feature-Sets 2: _____ _____

F. Begleitmaterial Dozenten

F.1. Basic Paper

Wie funktioniert eigentlich...
die Aktivitätserkennung auf dem Smartphone?

Lektion 1 Einleitung

Mobile Geräte, wie zum Beispiel Smartphones, sind mittlerweile ständige Begleiter im Alltag. Weltweit nutzen rund 3,9 Milliarden Menschen ein Smartphone[1]. Zudem ist in den letzten Jahren das Interesse an der Auswertung der Gewohnheiten und der täglichen Routinen der Menschen gestiegen. Es hat sich gezeigt, dass die Analyse des menschlichen Verhaltens von zentraler Bedeutung für das Verständnis der Bedürfnisse und Anforderungen einer Person ist. Diese Beobachtungen sind in vielen Bereichen, von der Pädagogik, Medizin und Soziologie bis hin zum Marketing, relevant (vgl. [4], S. 6475). Zu den menschlichen Gewohnheiten zählen auch die täglichen Aktivitäten einer Person. Durch die enorme Weiterentwicklung der in den Smartphones eingebauten Sensoren können diese zur Erkennung der menschlichen Aktivitäten genutzt werden. In den letzten Jahren hat sich daher die Aktivitätserkennung zu einem aktiven Forschungsfeld entwickelt, wobei sich die meisten Studien auf die Erkennung von Aktivitäten wie Sitzen, Stehen, Gehen, Joggen etc. konzentrieren (vgl. [19], S. 235).

Das Ziel der menschlichen Aktivitätserkennung ist es, die Aktivitäten oder Situationen, in denen sich eine Nutzerin oder ein Nutzer[2] befindet, über Sensoren am Smartphone zu bestimmen und gegebenenfalls darauf zu reagieren (vgl. [4], S. 6475). In den letzten Jahren wurden die meisten gängigen Smartphones mit zahlreichen Sensoren ausgestattet, unter anderem mit Beschleunigungssensoren, GPS-Sensoren, Gyroskopen, Barometern und Kompassen. Diese Sensoren stellen eine reichhaltige Datenquelle dar, um die täglichen Gewohnheiten und Routinen der Menschen aufzuzeichnen und zu analysieren (vgl. [19], S. 235). Im Zusammenhang mit der Aktivitätserkennung haben sich vor allem die 3-Achsen-Beschleunigungsmesser (Accelerometer) als nützlich erwiesen (vgl. [20], S. 1429). Vielfach kommen maschinelle Lernverfahren zum Einsatz, um die aufgenommenen Sensordaten den entsprechenden Aktivitäten zuordnen zu können (vgl. [4], S. 6475).

Für die Nutzer können sich aus der Aktivitätserkennung mit dem Smartphone große Vorteile ergeben, die ihnen unter Umständen den Alltag bedeutend erleichtern. Neben bekannten Anwendungen wie Fitness-Tracking und der Überwachung von Gesundheitsdaten könnte das Smartphone beispielsweise auch die Schrift vergrößern, wenn der Nutzer versucht, im Gehen (Chat-)Nachrichten zu lesen (vgl. [6], S. 17). Darüber hinaus kann die Aktivitätserkennung auch für Unternehmen zum Beispiel bei der Personalisierung von Werbung von Nutzen sein. Trotz der zahlreichen Vorteile und Anwendungsgebiete steht die Aktivitätserkennung auf dem Smartphone auch vor einigen Herausforderungen. Dazu zählen zum Beispiel die eingeschränkten Energie- und Rechenressourcen im Vergleich zu einem leistungsstarken Rechner aufgrund der begrenzten Akkulaufzeit und des beschränkten Speicherplatzes (vgl. [20], S. 1428).

In dem erstellten Lernmodul arbeiten Schüler ab Klasse zehn mit realen Aktivitätsdaten und entwickeln ihr eigenes mathematisches Modell zur Aktivitätserkennung. Dabei verwenden sie aktuelle Methoden aus dem Bereich der künstlichen Intelligenz. Das Material wurde so konzipiert, dass es sich sowohl für den Einsatz im Mathematikunterricht eignet als auch im Rahmen eines Projekttages bearbeitet werden kann.

Das vorliegende Paper liefert eine Einführung in die mathematischen und fachlichen Grundlagen der menschlichen Aktivitätserkennung. Zunächst werden die Begriffe künstliche Intelligenz und maschinelles Lernen erläutert. Anschließend werden die notwendigen mathematischen Grundlagen dargestellt sowie eine Einführung in die Aktivitätserkennung gegeben. Zum Abschluss werden die einzelnen Modellierungsschritte des Lernmoduls vorgestellt.

[1] https://de.statista.com/themen/581/smartphones/#dossierKeyfigures, letzter Aufruf: 15.09.2022.

[2] Nachfolgend werden Nutzerinnen und Nutzer unter der Bezeichnung „Nutzer“ zusammengefasst. Analog wird mit den Personengruppen Schülerinnen und Schüler, Lehrerinnen und Lehrer, u. a. verfahren.

Lektion 2 Künstliche Intelligenz und Maschinelles Lernen

In diesem Abschnitt werden die Begriffe *künstliche Intelligenz* und *maschinelles Lernen* erläutert sowie der im Lernmodul verwendete Klassifikationsalgorithmus vorgestellt.

Vorab sei jedoch angemerkt, dass sich für beide Begriffe in der Literatur nicht *die eine Definition* finden lässt. Je nachdem in welcher Fachrichtung (Informatik, Mathematik, ...) sich mit diesen Begriffen auseinandergesetzt wird, werden andere Definitionen und Beschreibungen aufgeführt. Um dennoch Schülern sowie Lehrkräften, die sich zuvor nicht mit diesem Thema beschäftigt haben, eine Vorstellung beider Begriffe liefern zu können, wird im Folgenden die Auffassung der Begriffe erläutert, die für das entwickelte Lernmodul relevant ist.

Lektion 2.1 Künstliche Intelligenz

Der Begriff der *künstlichen Intelligenz (KI)* ist in den letzten Jahren immer populärer geworden. Methoden aus dem Bereich KI sind die Basis vieler Anwendungen aus unserem täglichen Leben. Copeland (2019) definierte den Begriff KI wie folgt:

> *"Künstliche Intelligenz ist die Fähigkeit eines Computers oder computergesteuerten Roboters, Aufgaben zu lösen, die normalerweise von intelligenten Wesen erledigt werden"* (zitiert nach [16], S. 1).

Ein Computersystem soll also in der Lage sein, ähnlich wie ein Mensch intelligent zu handeln und eigenständig zu lernen. Diese Definition nach Copeland ist allerdings sehr vage, da der Begriff der Intelligenz nicht klar definiert ist und sich nur schwer abgrenzen lässt (vgl. [16], S. 1). So stellt sich beispielsweise die Frage, ob nach Copelands Definition nicht bereits ein Taschenrechner als KI bezeichnet werden müsste – denn Rechenaufgaben werden von intelligenten Menschen gelöst. In diesem Sinne ist diese Definition mit Vorsicht zu betrachten und soll nur als eine grobe Orientierung dienen.

Das Thema KI stellt aber keinesfalls ein neues Forschungsgebiet dar. Bereits im Jahr 1950 beschäftigte sich der britische Mathematiker Alan Turing mit der Frage, ob Maschinen bzw. Computersysteme intelligent sein können. Zur Untersuchung dieser Frage schlug er den folgenden Test vor (s. Abb. 1). Ein menschlicher Schiedsrichter kommuniziert elektronisch mit zwei Partnern (ein Partner ist ein Mensch, der andere ein Computer) und stellt ihnen beliebige Fragen. Kann der Schiedsrichter nach vielen Fragen nicht entscheiden, bei welchem der Partner es sich um den Computer handelt, so wird der Computer als intelligent angesehen. Dieser Test wird auch als Turing-Test bezeichnet (vgl. [16], S. 2).

Abbildung 1: Turing Test (entnommen aus [16], S. 3)

Der Begriff KI wurde allerdings erst sechs Jahre später von Prof. John Mc Carthy beim Dartmouth-Workshop vorgeschlagen (vgl. [16], S. 11). Ein kurzes, aufschlussreiches Video zur Geschichte von KI finden interessierte Leser unter [1].

Lektion 2.2 Maschinelles Lernen

In der Literatur wird das *maschinelle Lernen* meist als Teilgebiet von KI aufgefasst. Der Begriff maschinelles Lernen wird von Mitchell (1997) folgendermaßen definiert:

> *"Machine Learning is the study of computer algorithms that improve automatically through experience"* ([14]).

Ausgangspunkt vieler gängiger maschineller Lernverfahren ist eine große Menge an Daten bzw. Beispielen. Ziel ist es, eine bestimmte Aufgabe (z. B. Bilderkennung) zu lösen. Zur Erfüllung dieser Aufgabe wird ein Modell entwickelt. Dieses Modell wird vorab **nicht** von einem Spezialisten durch komplexe Regelsysteme festgelegt. Stattdessen werden die vorhandenen Daten und Informationen genutzt, um die Parameter des Modells bestmöglich zu wählen. Dieser Prozess wird als *Lernen* bezeichnet (vgl. [18], S. 156).

Eine Anwendung, bei der maschinelle Lernverfahren zum Einsatz kommen, sind Spamfilter eines E-Mail Postfachs. Anhand von Mail-Beispielen wird ein Modell entwickelt und gewisse Modellparameter bestmöglich „gelernt" mit dem Ziel, Spam E-Mails von gewöhnlichen E-Mails (so genannte Ham E-Mails) unterscheiden zu können (vgl. [8], S. 4).

Strategien des maschinellen Lernens

Im Bereich des maschinellen Lernens gibt es unterschiedliche Ansätze: das *überwachte Lernen*, das *unüberwachte Lernen* und das *bestärkende Lernen* (vgl. [7], S. 21). Das im Rahmen dieses Lernmoduls eingesetzte Verfahren lässt sich dem überwachten Lernen zuordnen, weshalb dies hier ausführlicher beschrieben wird.

Überwachtes Lernen

Bei der Methode des *überwachten Lernens* (engl. *supervised learning*) besteht der Beispieldatensatz sowohl aus den Eingangsdaten als auch aus den gewünschten Ergebnissen (so genannte *Labels*). In diesem Fall spricht man auch von einem *gelabelten* Datensatz (vgl. [7], S. 21). Ein Datenpunkt im Datensatz X besteht also aus dem Merkmalsvektor $\vec{x}_i$ und dem zugehörigen Label y_i. Im Vektor $\vec{x}_i$ werden die Werte aller Merkmale des i-ten Datenpunkts gespeichert. Setzt sich ein Datenpunkt aus n Merkmalen zusammen, so haben die Vektoren $\vec{x}_i$ die Dimension n (vgl. [8], S. 42). Ein Datensatz X mit m Datenpunkten enthält dann die Merkmalsvektoren $\vec{x}_i$, $i = 1, ..., m$ und die Labels y_i, $i = 1, ..., m$:

$$X = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), ..., (\vec{x}_m, y_m)\} \subseteq \mathbb{R}^n \times Y. \tag{1}$$

Ziel des überwachten maschinellen Lernverfahrens ist es, eine Funktion

$$f : X \to Y \tag{2}$$

zu finden, die jedem Vektor $\vec{x}_i$ aus dem Datensatz X das zugehörige Label y_i aus der Menge Y zuordnet (vgl. [7], S. 21).

In Abbildung 2 ist das Prinzip des überwachten maschinellen Lernens dargestellt. Der gelabelte Beispieldatensatz wird zu Beginn in zwei kleinere Datensätze, die *Trainings-* und die *Testdaten* aufgeteilt. Meistens werden 70 bis 80 Prozent der Daten den Trainingsdaten und 20 bis 30 Prozent der Daten den Testdaten zugeordnet (vgl. [11], S. 23 f.). Mit Hilfe des Trainingsdatensatzes wird dann eine Funktion f gelernt, die die Trainingsdatenpunkte den zugehörigen Labels zuordnet. Im Anschluss an die *Trainingsphase*, wird die Funktion f in der *Testphase* auf den Testdatensatz angewandt. Da es sich bei den Testdaten ebenfalls um vorab gelabelte Datenpunkte handelt, kann überprüft werden, ob die Funktion f den Testdatenpunkten die richtigen Labels zuordnet. Dadurch kann der Lernerfolg bewertet werden.

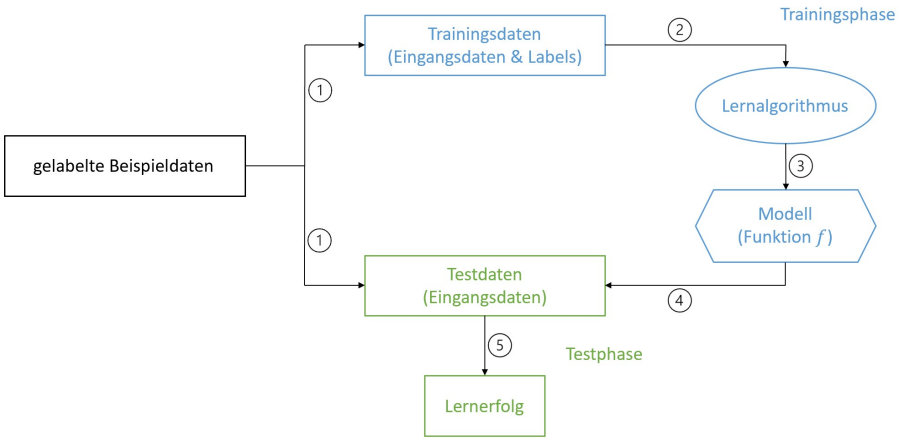


Abbildung 2: Prinzip des überwachten maschinellen Lernens

Die Problemstellungen des überwachten maschinellen Lernens werden in zwei Aufgabentypen unterteilt: die *Klassifikation* und die *Regression* (vgl. [7], S. 22).

Bei einem *Klassifizierungsproblem* geht es darum, Datenpunkte verschiedenen Klassen zuzuordnen. Die Menge C aller Klassen ist eine abgeschlossene und diskrete Menge. Auf Basis des vorliegenden Trainingsdatensatzes X, der sowohl die Eingangsdaten $\vec{x}_i$, $i = 1, ..., m$ als auch die zugehörigen Klassen c_i, $i = 1, ..., m$ enthält, soll eine Funktion

$$f : X \to C \tag{3}$$

gelernt werden. Diese Funktion f soll neuen Datenpunkten, die nicht im Trainingsdatensatz enthalten sind, eine Klasse aus der Menge C zuordnen.
Das Beispiel des Spamfilters stellt ein solches Klassifizierungsproblem dar. Die Menge der Klassen C ist gegeben durch

$$C = \{\text{Spam}, \text{Ham}\}. \tag{4}$$

Anhand der Trainingsdaten wird dann eine Funktion f erlernt, die die Mail-Beispiele einer der beiden Klassen zuordnet. Nach der Trainingsphase können dann neue, unbekannte E-Mails als Spam oder nicht Spam (Ham) klassifiziert werden (s. Abb. 3).

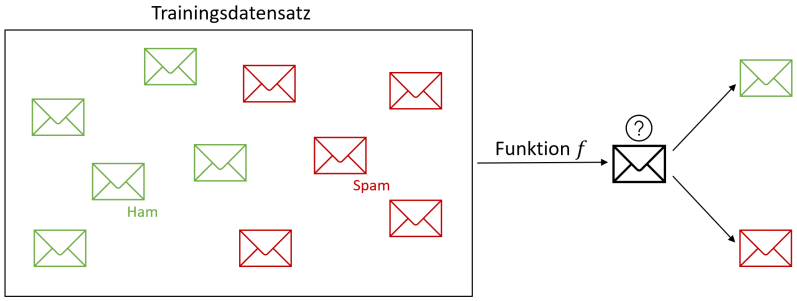


Abbildung 3: Klassifikation von E-Mails als Spam oder Ham

Eine weitere typische Problemstellung des überwachten Lernens ist die *Regression*. Der Unterschied zwischen den Klassifizierungs- und Regressionsproblemen liegt in der Menge C. Während die Werte der Menge C im Falle der Klassifikation nominal skaliert sind, stammen sie bei der Regression aus einem kontinuierlichen Bereich und sind metrisch skaliert (vgl. [7], S. 23).
Ein Beispiel für ein Regressionsproblem ist die Vorhersage des Preises eines Autos anhand verschiedener Merkmale (z. B. Alter, Marke, PS-Zahl, gefahrene Kilometer, etc) (vgl. [8], S.10). Die vorhergesagten Preise verschiedener Autos lassen sich der Größe nach sortieren, sind also metrisch skaliert. Im Gegensatz dazu können die Klassen bei Klassifikationsproblemen nicht geordnet werden. Eine E-Mail der Klasse Spam ist nicht „besser“ oder „schlechter“ als eine E-Mail der Klasse Ham.

Neben der Unterscheidung der Problemstellungen in Klassifikation und Regression lassen sich die Verfahren des überwachten Lernens auch anhand zweier Ansätze differenzieren: der *Eager Learner* und der *Lazy Learner* (vgl. [7], S. 24).

Beim *Eager Learner* wird in einer meist aufwendigen Trainingsphase ein *globales* Modell entwickelt. Auf Basis dieses Modells werden anschließend neue Eingangsdaten einem Wert aus der Menge C zugeordnet. Im Gegensatz zur Trainingsphase erfolgt die spätere Abfrage der erlernten Funktion relativ schnell (vgl. [7], S. 24). Dieser Ansatz ist in Abbildung 2 dargestellt.

Der *Lazy Learner* dagegen arbeitet mit einem *lokalen* Modell, das für jede einzelne Abfrage neu gebildet wird. Die Trainingsphase ist hier also weniger zeitintensiv als beim Eager Learner, die Abfrage jedoch teurer (vgl. [7], S. 24). Im Vergleich zum Eager Learner bietet der Lazy Learner den Vorteil, dass das lokale Modell für jede Abfrage passgenau gebildet werden kann und daher meist genauere bzw. bessere Ergebnisse liefert. Aus ökonomischen Gründen wird jedoch häufig auf Eager Learning-Ansätze zurückgegriffen. Die Eager Learner sind im Gegensatz zu den Lazy Learnern im Einsatz deutlich kostengünstiger, da die teure Modellentwicklung nur einmal in der Trainingsphasen erfolgt und die anschließenden Abfragen günstiger sind (vgl. [7], S. 24).

Unüberwachtes Lernen
Eine weitere Lernart des maschinellen Lernens ist das *unüberwachte Lernen* (engl. *unsupervised learning*). Im Unterschied zum überwachten Lernen besteht der Beispieldatensatz hier nur aus den Eingangsdaten, es wird also kein gelabelter Datensatz verwendet. Das Ziel des unüberwachten Lernens ist es, Strukturen und Muster in den Daten zu finden. Da die einzelnen Datenpunkte unmarkiert sind, lässt sich die Lösung des Computers nicht direkt beurteilen (vgl. [7], S. 25).

Ein typisches Beispiel für das unüberwachte Lernen ist die Einteilung von Kunden anhand ihres Kaufverhaltens beim Online-Shopping. Im Gegensatz zum überwachten Lernen werden die Kunden nicht einer bestimmten Klasse zugeordnet. Vielmehr geht es darum, die Gruppen anhand ihres ähnlichen Verhaltens zu bilden. Das „Label“ der Gruppe spielt dabei keine Rolle (vgl. [7], S. 25).

Bestärkendes Lernen
Neben dem überwachten und dem unüberwachten Lernen gibt es noch eine dritte Lernart, die hier vorgestellt werden soll: das *bestärkende Lernen* (engl. *reinforcement learning*). Im Gegensatz zu den anderen beiden Lernarten ist der Ausgangspunkt hier kein gegebener Datensatz. Stattdessen sind die Algorithmen des bestärkenden Lernens fast immer agentenbasiert. Durch kontinuierliche Rückmeldungen in Form von Belohnungen und Bestrafungen soll der Agent nach und nach eine (möglichst) optimale Strategie zur Lösung eines Problems finden (vgl. [7], S. 24).

Ein solcher Agent könnte zum Beispiel ein Putzroboter sein, der nach einer optimalen Strategie für die Reinigung eines Gebäudes in kürzester Zeit und mit möglichst wenig Energie sucht (vgl. [7], S. 24).

Lektion 2.3 Klassifikation mit dem k nächste Nachbarn Algorithmus

Das Problem der Aktivitätserkennung mit dem Smartphone stellt ein Klassifizierungsproblem dar, das im entwickelten Lernmodul mit Hilfe des k-*nächste-Nachbarn-Algorithmus (kNN-Algorithmus)* gelöst wird. Bei diesem Algorithmus handelt es sich um ein überwachtes maschinelles Lernverfahren, das nach dem Prinzip des Lazy Learners arbeitet. Die Grundidee des kNN-Algorithmus besteht

darin, unbekannte Datenpunkte auf Basis der Eigenschaften (Klassen) der k nächstliegenden Datenpunkte, auch *Nachbarn* genannt, zu klassifizieren (vgl. [7], S. 122).

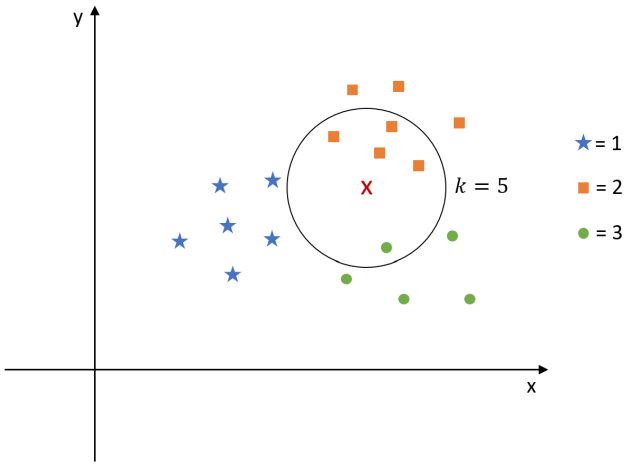


Abbildung 4: Grundidee des kNN-Algorithmus

In das Koordinatensystem in Abbildung 4 sind 16 gelabelte Datenpunkte eingetragen, die durch die Ausprägungen zweier Merkmale, hier x und y, repräsentiert werden. Ziel ist es, einen unbekannten Datenpunkt, hier dargestellt durch das rote X, einer der drei Klassen zuzuordnen. Dazu werden die fünf nächstliegenden Datenpunkte betrachtet. Unter den fünf nächsten Nachbarn des unbekannten Datenpunkts befinden sich vier Datenpunkte der Klasse zwei (orangene Quadrate) und ein Datenpunkt der Klasse drei (grüne Kreise). Da die Klasse zwei unter den fünf nächsten Nachbarn am häufigsten vorkommt, wird der unbekannte Datenpunkt dieser Klasse zugeordnet.

Der kNN-Algorithmus ist ein Algorithmus, der nur auf wenigen Parametern beruht. Es werden lediglich die Anzahl der Nachbarn k und die verwendete Metrik zur Bestimmung der nächstgelegenen Nachbarn festgelegt. In dem Beispiel aus Abbildung 4 wurde die euklidische Metrik verwendet. Es ist durchaus möglich auch andere Metriken zur Bestimmung der nächstgelegenen Datenpunkte zu verwenden. Darauf wird in Abschnitt 3.2 ausführlicher eingegangen.

Lektion 3 Mathematische Grundlagen

In diesem Abschnitt werden die mathematischen Grundlagen, die zur Bearbeitung des Lernmoduls benötigt werden, vorgestellt.

Lektion 3.1 Statistische Kenngrößen

Zur Vorverarbeitung der aufgenommenen Sensordaten werden verschiedene statistische Kenngrößen verwendet. Bei statistischen Kenngrößen wird zwischen den *Maßen der zentralen Tendenz* (auch *Lagemaße* genannt) und den *Streuungsmaßen* unterschieden. Während die Lagemaße alle Messwerte einer Verteilung repräsentieren und ihre grobe Lage beschreiben, geben die Streuungsmaße Auskunft über die Variation der Messwerte (vgl. [17]). Da Streuungsmaße nicht Inhalt des Mathematikunterrichts sind, kommen im Lernmodul nur Lagemaße zum Einsatz. Im Folgenden werden die im Lernmodul eingesetzten Lagemaße vorgestellt.

Es seien $x_1, ..., x_n$ die Messwerte eines Sensors.

- **Minimum und Maximum einer Messreihe:**
  Das *Minimum* x_{min} beschreibt den kleinsten und das *Maximum* x_{max} den größten Wert unter den Messwerten $x_1, ..., x_n$. Es gilt:

$$x_{\text{min}} := \min_{i=1,...,n} x_i \qquad \text{und} \qquad x_{\text{max}} := \max_{i=1,...,n} x_i. \tag{5}$$

- **arithmetisches Mittel einer Messreihe:**
  Das wohl bekannteste Lagemaß ist das arithmetische Mittel $\bar{x}$ von $x_1, ..., x_n$. Dieses ist definiert durch

$$\bar{x} := \frac{1}{n} \cdot (x_1 + ... + x_n) = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i. \tag{6}$$

- **Median einer Messreihe:**
  Beschreibt $x_{(j)}$ für $j = 1, ..., n$ den j-kleinsten Wert unter den Messwerten, so gilt insbesondere

$$x_{(1)} = x_{\text{min}} \qquad \text{und} \qquad x_{(n)} = x_{\text{max}}. \tag{7}$$

  Werden die Messwerte der Größe nach sortiert, ergibt sich eine Reihe der Messwerte

$$x_{(1)} \leq x_{(2)} \leq ... \leq x_{(n)}. \tag{8}$$

  Der *Median* beschreibt dann den mittleren Wert dieser geordneten Reihe an Messwerten. Es gilt

$$x_{1/2} := \begin{cases} x_{(\frac{n+1}{2})}, & \text{falls } n \text{ eine ungerade Zahl ist,} \\ \frac{1}{2} \cdot \left(x_{(\frac{n}{2})} + x_{(\frac{n}{2}+1)} \right), & \text{falls } n \text{ eine gerade Zahl ist.} \end{cases} \tag{9}$$

  Der Median teilt die Messwerte also in zwei gleichgroße Hälften, es sind mindestens 50 % der Messwerte kleiner oder gleich $x_{1/2}$ und mindestens 50 % größer oder gleich $x_{1/2}$. Im Gegensatz zum arithmetischen Mittel ist der Median robust gegenüber Ausreißern.

- **unteres und oberes Quartil einer Messreihe:**
  Eine Verallgemeinerung des Medians stellen die p-*Quartile* dar. Für eine Zahl p mit $0 < p < 1$ ist das p-Quartil gegeben durch

$$x_p := \begin{cases} x_{(\lfloor np+1 \rfloor)}, & \text{falls } np \notin \mathbb{N}, \\ \frac{1}{2} \cdot \left(x_{(np)} + x_{(np+1)} \right), & \text{falls } np \in \mathbb{N}. \end{cases} \tag{10}$$

  Besondere Fälle der p-Quartile sind das *untere Quartil* ($p = 0.25$) und das *obere Quartil* ($p = 0.75$). Sie sind die Mediane der unteren bzw. oberen Hälfte der geordneten Reihe an Messwerten (vgl. [9], S.27 ff.).

Lektion 3.2 Abstandsmetriken

Beim kNN-Algorithmus werden unbekannte Datenpunkte anhand der Klassen der k nächstliegenden Datenpunkte klassifiziert. Hierzu ist es erforderlich, den Abstand zwischen zwei Datenpunkten zu bestimmen und zu definieren, was unter „nächstgelegen“ verstanden wird. Im Folgenden werden kurz die wichtigsten Inhalte der linearen Algebra wiederholt, die zur Abstandsberechnung benötigt werden.

Jeder Datenpunkt im Datensatz ist ein Vektor, in dem die Merkmale des Datenpunktes gespeichert sind. Der Abstand zwischen zwei Datenpunkten bzw. Vektoren lässt sich durch eine *Metrik* bestimmen.

Definition: Metrik auf $(\mathbb{V}, +)$ (vgl. [7], S. 110)
Sei $(\mathbb{V}, +)$ ein Vektorraum. Eine Abbildung $d : \mathbb{V} \times \mathbb{V} \rightarrow \mathbb{R}$ heißt *Metrik* auf $\mathbb{V}$, wenn für beliebige Elemente x, y und z aus $\mathbb{V}$ gilt:

- $d(x, x) = 0$.
- $d(x, y) = 0 \quad \Rightarrow \quad x = y$.
- $d(x, y) = d(y, x)$ (Symmetrie).
- $d(x, y) \leq d(x, z) + d(z, y)$ (Dreiecksungleichung).

Eine Metrik auf $\mathbb{V}$ kann über

$$d(x, y) = \|x - y\| \tag{11}$$

auch durch eine Norm auf $\mathbb{V}$ induziert werden (vgl. [7], S. 110).

Im entwickelten Lernmodul kommen verschiedene Metriken zur Berechnung des Abstandes zwischen zwei Datenpunkten zum Einsatz. Diese sollen im Folgenden am Beispiel des Abstandes zwischen den beiden Vektoren $\vec{x}$ und $\vec{y}$ mit den Einträgen x_i, $i = 1, ..., n$ und y_i, $i = 1, ..., n$ vorgestellt werden.

Euklidische Metrik:
Die *euklidische Metrik* wird von der euklidischen Norm

$$\|\vec{x}\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}, \tag{12}$$

induziert und ist definiert durch

$$d_{\text{Euklid}} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}. \tag{13}$$

Sie beschreibt die Länge bzw. den Betrag des Verbindungsvektors $\vec{d}$ der beiden Vektoren $\vec{x}$ und $\vec{y}$ (vgl. [5], S. 281). Für den dreidimensionalen Fall ist dies in Abbildung 5 anhand des Abstands der beiden Punkte $P_1(x_1, y_1, z_1)$ und $P_2(x_2, y_2, z_2)$ beispielhaft dargestellt.

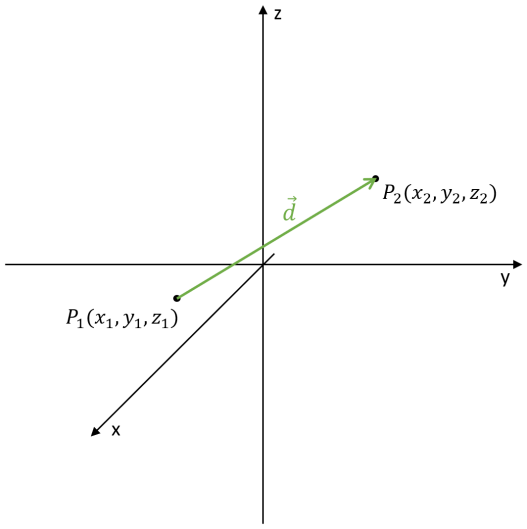


Abbildung 5: Abstand zwischen den beiden Punkten P_1 und P_2 mit der euklidischen Metrik im dreidimensionalen Fall

Manhattan-Metrik

Eine weitere Möglichkeit den Abstand zwischen zwei Vektoren zu bestimmen, ist durch die *Manhattan-Metrik* gegeben. Diese wird durch die Manhattan-Norm

$$\|\vec{x}\|_1 = \sum_{i=1}^{n} |x_i| \tag{14}$$

induziert und ist definiert durch (vgl. [5], S. 281)

$$d_{\text{Manhattan}} = \sum_{i=1}^{n} |x_i - y_i|. \tag{15}$$

Der Name Manhattan-Metrik stammt von der schachbrettmusterartigen Anordnung der Gebäudeblöcke und dem orthogonalen Straßengitter in New Yorks Stadtteil Manhattan. Ein Taxifahrer kann die Entfernung zwischen zwei Adressen nur durch die Aneinanderreihung vertikaler und horizontaler Wegstücke überwinden (z. B. blaue, orangene und gelbe Linien in Abb. 6) (vgl. [2]). Zum Vergleich ist in Abbildung 6 auch die euklidische Metrik (grüne Linie) dargestellt.

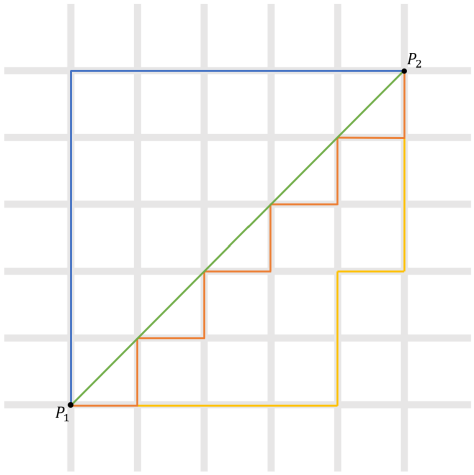


Abbildung 6: Abstand zwischen den beiden Punkten P_1 und P_2 mit der Manhattan-Metrik (blaue, orange und gelbe Linien) und der euklidischen Metrik (grüne Linie) im zweidimensionalen Fall

Kosinus-Metrik:
Der Abstand zwischen zwei Vektoren kann auch mit Hilfe des eingeschlossenen Winkels θ zwischen den beiden Vektoren definiert werden (vgl. [5], S. 282). In Abbildung 7 ist dies beispielhaft für den dreidimensionalen Fall und den Winkel zwischen den Ortsvektoren der beiden Punkte $P_1(x_1, y_1, z_1)$ und $P_2(x_2, y_2, z_2)$ beispielhaft dargestellt.

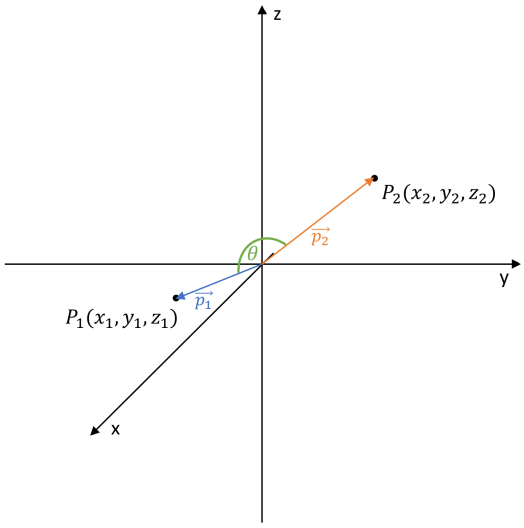


Abbildung 7: Winkel θ zwischen den Ortsvektoren der beiden Punkten P_1 und P_2 im dreidimensionalen Fall

Mit der *Kosinus-Metrik*, die über das Standardskalarprodukt und die euklidische Norm hergeleitet werden kann, ist der Abstand zwischen den zwei Vektoren $\vec{x}$ und $\vec{y}$ gegeben durch (vgl. [5], S. 282)

$$d_{\text{Kosinus}} = 1 - \cos(\theta) = 1 - \frac{\vec{x} \cdot \vec{y}}{|\vec{x}| \cdot |\vec{y}|}. \tag{16}$$

Tschebyschew-Metrik:
Eine weitere Metrik, die im Lernmodul zum Einsatz kommt, ist die *Tschebyschew-Metrik* bzw. *Maximumsmetrik*. Sie wird von der Maximumsnorm

$$\|\vec{x}\|_{\text{max}} = \max_{i=1,\dots,n} |x_i|. \tag{17}$$

induziert und definiert den Abstand zwischen zwei Vektoren $\vec{x}$ und $\vec{y}$ wie folgt (vgl. [5], S. 282):

$$d_{\text{Tschebyschew}} = \max_{i=1,\dots,n} |x_i - y_i|. \tag{18}$$

Lektion 3.3 Qualitätsmaße zur Bewertung von Klassifikationsergebnissen

Nachdem bei einem Klassifizierungsproblem in der Trainingsphase ein Modell erlernt wurde, wird in der Testphase anhand der Testdaten die Generalisierungsfähigkeit und damit der Lernerfolg des Modells überprüft. Zur quantitativen Bewertung der Güte eines Modells werden verschiedene *Qualitätsmaße* verwendet. Diese sollen am Beispiel eines binären Klassifizierers mit den beiden Klassen „positiv“ und „negativ“ erläutert werden (vgl. [6], S. 24).

Die Ergebnisse der binären Klassifikation mit m Testdatenpunkten können in Form einer *Wahrheitsmatrix* bzw. *Konfusionsmatrix* dargestellt werden (s. Tab. 1).

Tabelle 1: Wahrheitsmatrix für eine binäre Klassifikation mit m Testdatenpunkten (vgl. [6], S. 24)

	positiv klassifiziert	**negativ klassifiziert**	**Summe**
tatsächlich positiv	richtig positiv (RP)	falsch negativ (FN)	$RP + FN$
tatsächlich negativ	falsch positiv (FP)	richtig negativ (RN)	$FP + RN$
Summe	$RP + FP$	$FN + RN$	m

Mit Hilfe dieser Wahrheitsmatrix können im Anschluss verschiedene Qualitätsmaße berechnet werden (vgl. [6], S. 24). Im Lernmodul werden drei Qualitätsmaße zur Bewertung der Klassifikationsergebnisse verwendet: die *Genauigkeit*, die *Fehlerrate* und die *Präzision*.

Die *Genauigkeit* (engl. *accuracy*) gibt das Verhältnis der richtig klassifizierten Datenpunkte zu allen Datenpunkten an (vgl. [6], S. 24). Es gilt also:

$$\text{accuracy} = \frac{\text{Anzahl der richtig klassifizierten Datenpunkte}}{\text{Anzahl aller Datenpunkte}} = \frac{RP + RN}{m}. \tag{19}$$

Im Gegensatz zur Genauigkeit gibt die *Fehlerrate* das Verhältnis der falsch klassifizierten Datenpunkte zu allen Datenpunkten an (vgl. [6], S. 25). Es gilt:

$$\text{error rate} = \frac{\text{Anzahl der falsch klassifizierten Datenpunkte}}{\text{Anzahl aller Datenpunkte}} = \frac{FP + FN}{m}. \tag{20}$$

Die *Präzision* wird im Gegensatz zu den beiden vorherigen Qualitätsmaßen nicht für die gesamte Klassifikation berechnet, sondern für jede Klasse einzeln. Sie gibt das Verhältnis der richtig als Klasse z klassifizierten Datenpunkte zu allen als Klasse z klassifizierten Datenpunkten an (vgl. [6], S. 24). Für die Klasse „positiv“ ergibt sich dann

$$\text{precision}_{\text{pos}} = \frac{\text{Anzahl der richtig als positiv klassifizierten Datenpunkte}}{\text{Anzahl aller als positiv klassifizierten Datenpunkte}} = \frac{RP}{RP + FP} \tag{21}$$

und für die Klasse „negativ“

$$\text{precision}_{\text{neg}} = \frac{\text{Anzahl der richtig als negativ klassifizierten Datenpunkte}}{\text{Anzahl aller als negativ klassifizierten Datenpunkte}} = \frac{RN}{FN + RN}. \tag{22}$$

Für eine gute Klassifikation sollten die Genauigkeit und die Präzision der einzelnen Klassen maximiert und die Fehlerrate minimiert werden (vgl. [6], S. 25).

Lektion 4 Grundlagen der Aktivitätserkennung

In diesem Abschnitt werden die Grundlagen der Aktivitätserkennung erläutert. Zunächst wird ein kurzer Überblick über die Geschichte der Aktivitätserkennung gegeben. Im Anschluss wird näher auf die Sensoren sowie die Aktivitäten eingegangen und der Prozess der Aktivitätserkennung erläutert. Den Abschluss bilden Herausforderungen und Probleme sowie Anwendungen und Einsatzgebiete der Aktivitätserkennung.

Lektion 4.1 Geschichte der Aktivitätserkennung

Einer der ersten Ansätze der Aktivitätserkennung war die Bildsensorik. Es wurde versucht mit Hilfe von Kameras die Aktivitäten von Personen zu bestimmen. Später wurde der Fokus auf die Trägheitssensorik mittels bewegungsabhängigen Sensoren, die am Körper des Nutzers angebracht waren, gelegt. Ein Nachteil dieses Ansatzes war, dass sich die Nutzer oft durch die Sensoren gestört und in ihrem Alltag beeinträchtigt fühlten. Heutzutage können Mobiltelefone zur Aktivitätserkennung genutzt werden. Zu Beginn wurden hauptsächlich GSM-Signale[3] verwendet, um einfache Fortbewegungen zu erkennen. Auch externe Sensoren, die am Körper des Nutzers positioniert und mit dem Smartphone verbunden waren, kamen erneut zum Einsatz. Durch die Weiterentwicklung und Integration zahlreicher Sensoren stellen Smartphones mittlerweile eine geeignete Plattform für die Aktivitätserkennung dar, sodass keine externen Sensoren mehr benötigt werden (vgl. [10], S. 145).

Lektion 4.2 Sensoren

Sensoren liefern die Rohdaten, die zur Aktivitätserkennung benötigt werden. Die unterschiedlichen Sensoren lassen sich in drei Kategorien einteilen:

- **Videosensoren:**
  Bei den Videosensoren handelt es sich um Kameras, die an festen Orten (z. B. am Ein- und Ausgang von Gebäuden) installiert sind. Sie liefern Informationen über das Aussehen und die Handlungen von Personen und werden vor allem bei der Überwachung sowie der Terrorismus- und Verbrechensbekämpfung eingesetzt (vgl. [19], S. 236).
- **Umgebungsbasierte Sensoren:**
  Umgebungsbasierte Sensoren werden verwendet, um die Interaktion eines Nutzers mit seiner Umgebung zu erkennen. Zu den umgebungsbasierten Sensoren zählen unter anderem WLAN- und Bluetooth-Sensoren. Diese werden vor allem in Innenräumen, wie zum Beispiel Bürogebäuden, eingesetzt. Sie zeichnen den Aufenthalt sowie die Interaktion zwischen Nutzern und anderen mit Sensoren ausgestatteten Geräten an bestimmten Orten auf (vgl. [19], S. 236).
- **Tragbare Sensoren:**
  Bei den tragbaren Sensoren handelt es sich um sehr kleine mobile Sensoren, die während den täglichen Aktivitäten am Körper eines Nutzers getragen werden können. Sie sind in der Lage den Bewegungszustand eines Nutzers aufzuzeichnen. Dazu zählen zum Beispiel die Bewegungsrichtung und die Geschwindigkeit. Der Großteil dieser tragbaren Sensoren ist heutzutage in Smartphones eingebaut (vgl. [19], S. 236).

Die meisten Forschungsansätze im Bereich der Aktivitätserkennung fokussieren sich auf tragbare Sensoren, die in Smartphones integriert sind. Zu den wichtigsten dieser Sensoren zählen der GPS-Sensor, das Accelerometer, das Gyroskop und das Magnetometer. Mit Hilfe des *GPS-Sensors* können der Standort und die Geschwindigkeit eines Nutzers bestimmt werden (vgl. [10], S. 147). Das *Accelerometer* (*Beschleunigungsmesser*) erfasst die Beschleunigungsvorgänge des Smartphones in allen drei Raumdimensionen (s. Abb. 8). Um die Rotationsrate des Smartphones bestimmen zu können, misst das *Gyroskop* die Roll-, Nick- und Gierbewegung des Smartphones entlang der x-, y- und z-Achse. Im Bereich der Aktivitätserkennung wird das Gyroskop auch zur Bestimmung der Orientierung des Smartphones genutzt. Das Magnetometer arbeitet wie ein Kompass und bestimmt die Richtung des Smartphones in Bezug auf den Nord-Pol der Erde mit Hilfe von Magnetismus. Bei der

[3] Die Abkürzung GSM steht für Global System for Mobile Communications und ist ein 1990 eingeführter Mobilfunkstandard für volldigitale Mobilfunknetze (vgl. [3]).

Aktivitätserkennung liefert das Magnetometer Informationen über mögliche Richtungsänderungen eines Nutzers zum Beispiel beim Gehen (vgl. [19], S. 237 f.).

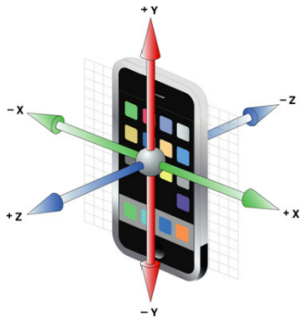


Abbildung 8: Die drei Achsen des Accelerometers (entnommen aus [6], S. 26)

Für die menschliche Aktivitätserkennung hat sich hauptsächlich das Accelerometer durchgesetzt, da es den Bewegungszustand eines Nutzers aufzeichnet. Beginnt ein Nutzer während des Gehens zu Joggen, so wird sich dies unmittelbar in den Beschleunigungswerten der vertikalen Achse zeigen (vgl. [19], S. 237). Zusammen mit dem Accelerometer werden häufig noch das Gyroskop und das Magnetometer verwendet, um die Lage und Neigung des Smartphones zu bestimmen (vgl. [6], S.26).

Um neben den Aktivitäten auch Informationen über die Umgebung, die soziale Interaktion und den Standort eines Nutzers gewinnen zu können, werden weitere Sensoren wie zum Beispiel Mikrophone, Kameras, Mobilfunk- und WLAN-Schnittstellen sowie Näherungs- und Umgebungslichtsensoren verwendet (vgl. [10], S. 146).

Lektion 4.3 Aktivitäten

Erste Arbeiten im Bereich der Aktivitätserkennung beschäftigten sich mit einer sehr oberflächlichen Erkennung der Aktivitäten im Zusammenhang mit Standortinformationen. Dazu zählten zum Beispiel zu Hause sein oder im Büro bei der Arbeit sein. Diese Schlussfolgerungen gaben aber keine Auskunft über die genauen Aktivitäten eines Nutzers (vgl. [10], S. 148).

Durch die Weiterentwicklung der Sensoren und die Ausstattung der Smartphones mit immer leistungsstärkeren Sensoren ist es mittlerweile möglich, genauere Informationen über die Aktivitäten eines Nutzers zu gewinnen. Mit Hilfe des Accelerometers kann zum Beispiel erkannt werden, dass ein Nutzer sitzt. Über das Mikrophon kann zusätzlich festgestellt werden, dass sich der Nutzer in einer Konversation befindet. Der Bluetooth-Sensor liefert die Information, dass sich ein Arbeitskollege in der Nähe befindet. Anhand dieser Daten kann geschlussfolgert werden, dass sich der Nutzer in einem Meeting im Büro befindet. Außerdem ist es mittlerweile möglich komplexe sportliche Aktivitäten wie Radfahren, Fußballspielen, Nordic Walking, Rudern und Laufen zu unterscheiden. Darüber hinaus können sowohl Aktivitäten des alltäglichen Lebens wie Frühstücken und Einkaufen als auch gefährliche Situationen wie Stürze erkannt werden (vgl. [10], S. 148).

Die meisten Studien konzentrieren sich jedoch auf fünf grundlegende Bewegungsaktivitäten, darunter Sitzen, Stehen, Gehen, Laufen und Treppen auf- und absteigen, da diese Aktivitäten von vielen Menschen häufig im Alltag ausgeführt werden (vgl. [12], S. 76).

Lektion 4.4 Prozess der Aktivitätserkennung

Im Folgenden wird der Prozess der Entwicklung eines Modells zur Aktivitätserkennung erläutert.

Datenaufnahme

Der erste Schritt bei der Entwicklung eines Modells zur Aktivitätserkennung ist die Datenaufnahme. Vorher muss jedoch festgelegt werden, mit welchen Sensoren und zu welchen Aktivitäten Daten aufgenommen und welche Abtastrate verwendet werden soll.

Die *Abtastrate* ist eine wichtige Stellschraube der Aktivitätserkennung und gibt an, wie viele Messwerte pro Sekunde aufgenommen werden. Sie wird in der Einheit Hertz (Hz) angegeben. Eine hohe Abtastrate liefert auf der einen Seite viele Informationen, kann aber auf der anderen Seite auch mehr Rauschen in den Daten verursachen (vgl. [19], S. 239). In der Literatur werden verschiedene Abtastraten zwischen 20 Hz und 100 Hz verwendet (vgl. [12], S. 75; [20], S. 1430).

Nachdem die Aktivitäten und die Abtastrate festgelegt wurden, können die Daten mit den gewählten Sensoren aufgenommen werden. Dabei sollte darauf geachtet werden, dass die Daten von möglichst vielen unterschiedlichen Personen stammen. Außerdem sollten für alle Aktivitäten möglichst gleich viele Daten existieren, sodass keine Klasse übertrainiert wird (vgl. [6], S. 18 f.).

Datenvorverarbeitung

Auf die Datenaufnahme folgt die Datenvorverarbeitung. Im Rahmen der Datensegmentierung wird der dichte Strom an Sensordaten in kleinere Zeitfenster, sogenannte *Windows* unterteilt. Neben der Abtastrate ist auch die Größe der Windows eine wichtige Stellschraube der Aktivitätserkennung. Kleine Fenstergrößen benötigen weniger Ressourcen und ermöglichen so eine schnelle Erkennung der Aktivitäten, während größere Fenster aufgrund der Einbeziehung einer größeren Datenmenge die Bestimmung komplexer Aktivitäten gestatten. Die Fenstergröße sollte also auf den jeweiligen Anwendungsfall abgestimmt sein. In der Literatur finden sich verschiedene Fenstergrößen zwischen zwei und zehn Sekunden (vgl. [20], S. 1430; [12], S. 75).

Neben der Datensegmentierung zählt auch der Prozess der Merkmalsextraktion zur Datenvorverarbeitung. In dieser Phase wird die große Menge an Rohdaten auf eine kleinere Menge von möglichst aussagekräftigen Werten (sogenannte *Features*) reduziert. Diese Features werden für die vorher eingeteilten Windows berechnet und sollen die Eigenschaften des Signals bestmöglich beschreiben (vgl. [10], S. 150). In den meisten Studien zur Aktivitätserkennung werden hauptsächlich zeitabhängige Features verwendet. Um die Ergebnisse der Klassifikation zu verbessern, kommen zusätzlich auch frequenzabhängige Features zum Einsatz (vgl. [6], S. 27 f.). In Tabelle 2 sind verschiedene zeit- und frequenzabhängige Features, die bei der Aktivitätserkennung genutzt werden, aufgelistet. Die zeitabhängigen Features beschreiben die grundlegenden Statistiken eines Datensegments und werden meist für die Werte der einzelnen Achsen sowie den Betrag des Beschleunigungsvektors berechnet. Mit Hilfe der frequenzabhängigen Features kann die Periodizität des Signals beschrieben werden. Sie werden in der Regel auf Grundlage der Fast Fourier Transformation (FFT) berechnet (vgl. [19], S. 240).

Tabelle 2: Beispiele für zeit- und frequenzabhängige Features

zeitabhängige Features	frequenzabhängige Features
- Mittelwert	- erste zehn FFT-Koeffizienten
- Minimum und Maximum	- Energie
- Standardabweichung	- Entropie
- Varianz	- Zeit zwischen den Ausschlägen des Signals
- Korrelation	- Nulldurchgangsrate
- Median	- ...
- unteres und oberes Quartil	
- ...	

Klassifikation

Auf die Datenvorverarbeitung folgt die Entwicklung eines Klassifikationsalgorithmus. Bei der Klassifikation sollen die für die Daten der einzelnen Windows berechneten Features den unterschiedlichen Aktivitäten zugeordnet werden. Neben dem im Lernmodul verwendeten kNN-Algorithmus werden weitere überwachte maschinelle Lernverfahren zur Klassifikation von Sensordaten eingesetzt. In der Literatur finden sich:

- Naive Bayes Klassifikatoren
- Entscheidungsbäume
- Stützvektormethode
- Verdeckte Markovmodelle
- Neuronale Netze
- etc.

Dabei hat der kNN-Algorithmus im Vergleich zu den anderen Algorithmen vielfach die besten Ergebnisse gezeigt (vgl. [15], S. 305).

Evaluation

In der Testphase wird überprüft, ob die über das maschinelle Lernverfahren zugeordneten Aktivitäten mit den tatsächlichen Aktivitäten, die ein Nutzer ausführt bzw. ausgeführt hat, übereinstimmen. Um den Lernerfolg des Systems bewerten zu können, kommen verschiedene Qualitätsmaße (s. Abschn. 3.3) zum Einsatz (vgl. [10], S. 151).

Lektion 4.5 Herausforderungen und Probleme der Aktivitätserkennung

Die Aktivitätserkennung anhand der in Smartphones eingebauten Sensoren steht noch vor mehreren ungelösten Problemen und Herausforderungen, von denen im Folgenden einige vorgestellt werden.

Subjektive Empfindlichkeit

Die Genauigkeit der Aktivitätserkennung ist stark von den einzelnen Nutzern abhängig. Jeder Mensch hat seine eigenen Gewohnheiten und Verhaltensweisen und führt daher Bewegungen im Vergleich zu anderen Nutzern unterschiedlich aus (vgl. [19], S. 242). Um mögliche Fehlklassifikationen vorzubeugen, sollten die Trainingsdaten aus so vielen Daten wie möglich von so vielen unterschiedlichen Nutzern wie möglich zusammengesetzt sein. Dem stehen jedoch die begrenzten Speicher- und Rechenressourcen eines Smartphones gegenüber.

Standortempfindlichkeit

Die aufgenommenen Sensordaten, besonders die des Accelerometers, sind stark von der Ausrichtung und der Position der Sensoren am Körper des Nutzers abhängig. Hält eine Person das Smartphone in der Hand während sie geht und liest dabei (Chat-)Nachrichten, so unterscheiden sich die aufgenommenen Sensordaten deutlich von den Daten, die aufgenommen werden, wenn sich das Handy in der Hosentasche befindet. Zur Lösung dieses Problems können weitere Sensoren (z. B. das Gyroskop und das Magnetometer) einbezogen werden, um die Ausrichtung und Position der Sensoren vorab bestimmen zu können bzw. die Sensordaten in Erdkoordinaten umzurechnen (vgl. [19], S. 242 f.).

Energie- und Ressourcenbeschränkung

Der Einsatz vieler Sensoren wirkt sich zum einen auf die Akkulaufzeit und zum anderen auf den Speicherplatz aus (vgl. [6], S. 31 f.). Auch die Ausführung des Klassifikationsalgorithmus kann aufgrund der geringen Rechenressourcen eines Smartphones im Vergleich zu einem leistungsstarken Rechner zu Problemen führen. Diese Herausforderungen könnten durch eine Auslagerung der Klassifikation auf einen externen Server zumindest teilweise bewältigt werden (vgl. [10], S. 152 f.).

Komplexität der Aktivitäten

Werden mehrere Aktivitäten gleichzeitig ausgeführt, beispielsweise Sitzen und Frühstücken, sind diese schwer zu unterscheiden bzw. zu erkennen. Darüber hinaus stellt auch der Übergang zwischen zwei Aktivitäten ein Problem für die Aktivitätserkennung dar (vgl. [19], S. 243).

Lektion 4.6 Anwendungen und Einsatzgebiete der Aktivitätserkennung

Die menschliche Aktivitätserkennung kann in vielen Bereichen von Nutzen sein. Einerseits kann sie enorme Vorteile für den Endnutzer mit sich bringen, andererseits kann sie auch für einzelne Unternehmen und die Gesellschaft nützlich sein. Im Folgenden werden verschiedene Anwendungen und Einsatzgebiete der Aktivitätserkennung aufgeführt.

Anwendungen für den Endnutzer

- **Fitness-Tracking:**
  Die Aktivitätserkennung kann im Rahmen des Fitness-Trackings die Schritte zählen, die täglich verbrauchten Kalorien angeben, die zurückgelegte Strecke messen und die Anzahl der gestiegenen Treppenstufen anzeigen. Diese Informationen können zur Überprüfung des aktuellen Fitnessstands genutzt werden und zu einer gesünderen Lebensweise motivieren (vgl. [6], S. 29).
- **Gesundheitsüberwachung:**
  Durch eine kontinuierliche Überwachung der Gewohnheiten und der täglichen Aktivitäten von Patienten kann die Diagnose durch den Arzt verbessert werden. Zudem können die aufgenommenen Daten einen Ausgangspunkt für weitere medizinische Untersuchungen darstellen (vgl. [6], S. 29).
- **Sturzerkennung:**
  Die Sturzerkennung anhand der Sensoren eines Smartphones ermöglicht sowohl das automatische Absetzen eines Notrufs als auch die Ortung des Handys, sodass die verletzte Person schnell gefunden und gerettet werden kann (vgl. [10], S. 149).
- **Kontextbezogenes Verhalten:**
  Das Smartphone kann auf die Aktivitäten des Nutzers reagieren, indem es zum Beispiel die Schrift vergrößert, wenn der Nutzer im Gehen (Chat-)Nachrichten liest. Dadurch kann der Nutzer in seiner aktuellen Situation und Aktivität durch das Smartphone unterstützt werden (vgl. [6], S. 29).

Anwendungen für Unternehmen und Dritte

- **Gezielte Werbung:**
  Durch die Aufzeichnung der Aktivitäten eines Nutzers kann die Werbung auf die Gewohnheiten des Nutzers abgestimmt werden. Dadurch wirkt sie weniger aufdringlich, denn sie ist für die Aktivitäten und Interessen des Nutzers relevant. Durch diese zielgerichtete Werbung soll eine größtmögliche Effektivität aus der Werbung gezogen werden können (vgl. [6], S. 29 f.).
- **Forschungsplattformen:**
  Im Bereich der Forschung wird häufig zuerst eine ausreichende Menge an Daten benötigt. Daher sind Forscher in vielen Bereichen, vom Marketing bis zum Gesundheitswesen, an der Sammlung von Aktivitätsdaten interessiert (vgl. [6], S. 30).
- **Unternehmensführung:**
  Im Bereich der Unternehmensführung kann die Aktivitätserkennung bei der Mitarbeiterverwaltung und bei der Abrechnung der Arbeitszeiten unterstützend eingesetzt werden (vgl. [13], S. 3).
- **Versicherungen:**
  Manche Versicherungsgesellschaften bieten ihren Versicherungsnehmern einen Rabatt zum Beispiel bei der Autoversicherung an, wenn eine sichere Fahrweise durch elektronische Geräte wie Smartphones erkannt wird (vgl. [13], S. 3).

Anwendungen für Gruppen und Crowds

- **Soziale Netzwerke:**
  Aktivitätsbasierte soziale Netzwerke können ihren Nutzern einen Freundeskreis anhand des Aufenthaltsortes sowie den ähnlichen Aktivitätsmustern vorschlagen (vgl. [13], S. 4).
- **Crowd-Sourcing:**
  Im Rahmen des aktivitätsbezogenen Crowd-Sourcings können Gegenden identifiziert werden, in denen bestimmte Aktivitäten besonders häufig ausgeführt werden. Einem Nutzer der gerne Fahrrad fährt, können dann beliebte Fahrradstrecken vorgeschlagen werden. Darüber hinaus können diese Systeme auch erkennen, wenn in einer Region die Aktivitäten stark von den normalerweise beobachteten Aktivitäten abweichen. Dadurch können mögliche Unfälle und Katastrophen schneller erkannt werden (vgl. [13], S. 4).

Die Aktivitätserkennung kann jedoch trotz der zahlreichen Anwendungen und Einsatzgebiete auch kritisch angesehen werden, gerade hinsichtlich des Datenschutzes und der Privatsphäre sowie der Überwachung durch einen totalitären Staat.

Lektion 5 Umsetzung im Lernmodul

In diesem Abschnitt werden die einzelnen Modellierungsschritte, die im Lernmodul bei der Erstellung eines Modells zur Aktivitätserkennung durchlaufen werden, vorgestellt. Die Schritte orientieren sich an dem in Abschnitt 4.4 vorgestellten Prozess der Aktivitätserkennung.

Im Vorfeld wurden Daten zu verschiedenen Aktivitäten mit Hilfe der App phyphox[4] aufgenommen. Während der Datenaufnahme war das Handy immer gleich positioniert (in der rechten vorderen Hosentasche). Es wurden Daten des Accelerometers, des Gyroskops und des Magnetometers aufgezeichnet. Zur Vereinfachung werden im Lernmodul jedoch nur die Daten des Accelerometers verwendet. Jeder Aktivität und jeder Testperson, die Daten zu den einzelnen Aktivitäten aufgenommen hat, wurden eine natürliche Zahl zugeordnet, die sogenannte *ActivityID* bzw. *UserID*.

Lektion 5.1 Erkunden des Datensatzes (*Arbeitsblatt 1*)

Zunächst wird der Datensatz, der dem Lernmodul zu Grunde liegt, vorgestellt. Jeder Datenpunkt in diesem Datensatz besteht aus der UserID und der ActivityID sowie der Zeit ab Beginn der Datenaufnahme in Sekunden und den Beschleunigungswerten der drei Raumdimensionen x, y und z in $\frac{\text{m}}{\text{s}^2}$ (s. Abb. 9).

UserID	ActivityID	Time (s)	x (m/s^2)	y (m/s^2)	z (m/s^2)
1	1	0.028140	5.480853	-4.508624	-6.869365
1	1	0.053292	5.488187	-4.504134	-6.870263
1	1	0.078443	5.494324	-4.496799	-6.872658
1	1	0.103595	5.492229	-4.508475	-6.871311
1	1	0.128747	5.472320	-4.499044	-6.856192
...	...	...	...	...	...

Abbildung 9: Screenshot eines Ausschnittes des Datensatzes aus dem digitalen Lernmaterial

Insgesamt wurden Daten zu fünf Aktivitäten von zehn Testpersonen (fünf weiblich und fünf männlich) aufgenommen. Die jüngste Testperson war 18 Jahre alt und die älteste Testperson war 52 Jahre alt. Die Aktivitäten mit der zugehörigen ActivityID sind:

- ActivityID 1: Sitzen,
- ActivityID 2: Stehen,
- ActivityID 3: Gehen,
- ActivityID 4: Laufen / Joggen,
- ActivityID 5: Treppen auf- und absteigen.

In Abbildung 10 sind die Sensordaten der einzelnen Aktivitäten für drei Sekunden graphisch dargestellt. Es ist zu erkennen, dass die Beschleunigungswerte der einzelnen Raumdimensionen bei den Aktivitäten Sitzen und Stehen nahezu auf einer Gerade liegen, während bei den anderen Aktivitäten deutliche Schwankungen zu erkennen sind. Das liegt daran, dass sich das Smartphone beim Sitzen und Stehen in Ruhe befindet und daher die Beschleunigung konstant ist. Die meisten und die stärksten Ausschläge sind bei der Aktivität Laufen zu erkennen, da hier die meisten Schritte innerhalb einer gewissen Zeitspanne gemacht werden und sich am stärksten vom Boden abgedrückt wird. Dadurch ist die Beschleunigung bei einem Schritt im Vergleich zu den Aktivitäten Gehen und Treppen auf- und absteigen am größten. Außerdem lässt sich erkennen, dass die größten Beschleunigungswerte auf

[4] Phyphox ist eine an der RWTH Aachen entwickelte App, die es ermöglicht mit Hilfe der Sensoren des Smartphones physikalische Experimente durchzuführen (`https://phyphox.org/de/home-de/`, letzter Aufruf: 15.09.2022).

der y-Achse zu finden sind. Der Grund dafür ist die Erdbeschleunigung. Diese ist zum Erdmittelpunkt hin gerichtet und wirkt daher bei allen Aktivitäten bis auf Sitzen auf die y-Achse des Accelerometers, denn das Smartphone befand sich während der Datenaufnahme senkrecht in der vorderen rechten Hosentasche.

Jede Aktivität wurde bei der Datenaufnahme von den einzelnen Testpersonen 180 Sekunden, also drei Minuten ausgeführt. Innerhalb dieser 180 Sekunden wurden ca. 7150 Datenpunkte aufgenommen. Somit wurden die Daten mit einer Abtastrate von ca.

$$\frac{7150}{180\text{s}} \approx 40\text{Hz} \tag{23}$$

aufgenommen.

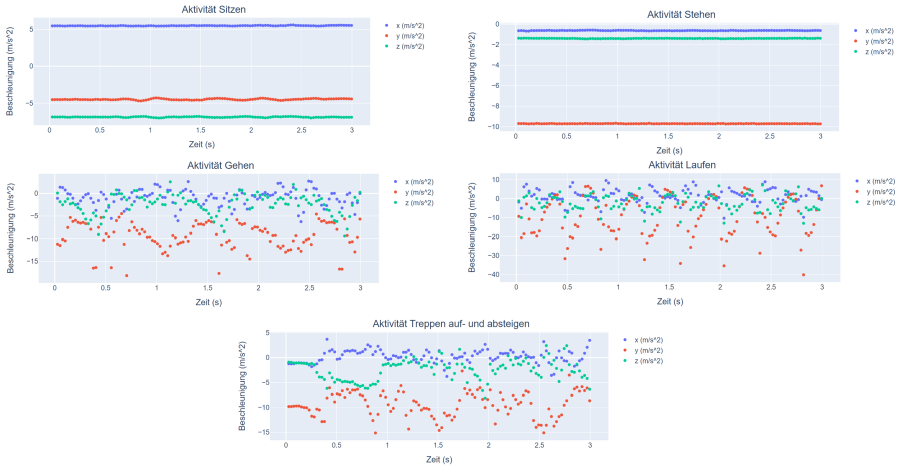


Abbildung 10: Graphische Darstellung der Sensordaten der einzelnen Aktivitäten für drei Sekunden

Lektion 5.2 Vorverarbeitung der Daten (*Arbeitsblatt 2*)

Zu Beginn der Vorverarbeitung der Sensordaten werden die Beschleunigungswerte der drei Achsen kombiniert, indem der Betrag des Beschleunigungsvektors

$$\vec{a} = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} \tag{24}$$

berechnet wird, wobei a_x, a_y und a_z die Beschleunigungswerte der einzelnen Raumdimensionen x, y und z bezeichnen. Der Betrag des Beschleunigungsvektors ist definiert durch

$$|\vec{a}| = \sqrt{a_x^2 + a_y^2 + a_z^2} \tag{25}$$

und wird im Lernmodul als *Vector Length* bezeichnet.

Anschließend werden die Daten in kleinere Zeitfenster einer festen Länge unterteilt. Die Größe der Windows wird auf drei Sekunden festgelegt und jedem Datenpunkt wird eine *WindowID* zugeordnet. Den Daten zu Aktivität 1 von Nutzer 1, die zwischen null und drei Sekunden liegen, wird die WindowID 1 zugeordnet, den Daten zu Aktivität 1 von Nutzer 1, die zwischen drei und sechs Sekunden

liegen, die WindowID 2, usw. Insgesamt werden die aufgenommenen Sensordaten in 3000 Windows unterteilt. Im Anschluss werden die WindowID und der Betrag des Beschleunigungsvektors als zusätzliche Spalten dem Datensatz hinzugefügt (s. Abb. 11).

WindowID	UserID	ActivityID	Time (s)	x (m/s^2)	y (m/s^2)	y (m/s^2)	Vector Length (m/s^2)
1	1	1	0.028140	5.480853	-4.508624	-4.508624	8.408040
1	1	1	0.053292	5.488187	-4.504134	-4.504134	8.408010
1	1	1	0.078443	5.494324	-4.496799	-4.496799	8.404166
1	1	1	0.103595	5.492229	-4.508475	-4.508475	8.415299
1	1	1	0.128747	5.472320	-4.499044	-4.499044	8.392204

Abbildung 11: Screenshot eines Ausschnittes des um die WindowID und den Betrags des Beschleunigungsvektors ergänzten Datensatzes

Auf die Datensegmentierung folgt die Merkmalsextraktion. Die große Menge an Rohdaten wird auf eine kleinere Menge möglichst aussagekräftiger Werte reduziert. Als aussagekräftige Werte werden zunächst die statistischen Kenngrößen Minimum und Maximum sowie das arithmetische Mittel des Betrags des Beschleunigungsvektors gewählt. Diese Kenngrößen werden mit Hilfe des Computers für alle 3000 Windows berechnet und zusammen mit der WindowID, ActivityID und UserID in einem neuen Datensatz, dem *Feature-Set 1*, abgespeichert (s. Abb. 12).

WindowID	UserID	ActivityID	Mean Vector Length (m/s^2)	Max Vector Length (m/s^2)	Min Vector Length (m/s^2)
1.0	1.0	1.0	8.382189	8.593413	8.136967
2.0	1.0	1.0	8.335617	8.571866	8.052737
3.0	1.0	1.0	8.289633	8.384486	8.059064
4.0	1.0	1.0	8.262474	8.533645	7.956085
5.0	1.0	1.0	8.268336	8.325962	8.224459

Abbildung 12: Screenshot eines Ausschnittes aus dem Feature-Set 1

Lektion 5.3 Entwicklung eines Klassifikationsalgorithmus (*Arbeitsblatt 3*)

Jedes Window im Feature-Set 1 entspricht einem Datenpunkt in drei Dimensionen (s. Abb. 13). Die Dimensionen sind die drei berechneten Features Minimum, Maximum und arithmetisches Mittel des Betrags des Beschleunigungsvektors.

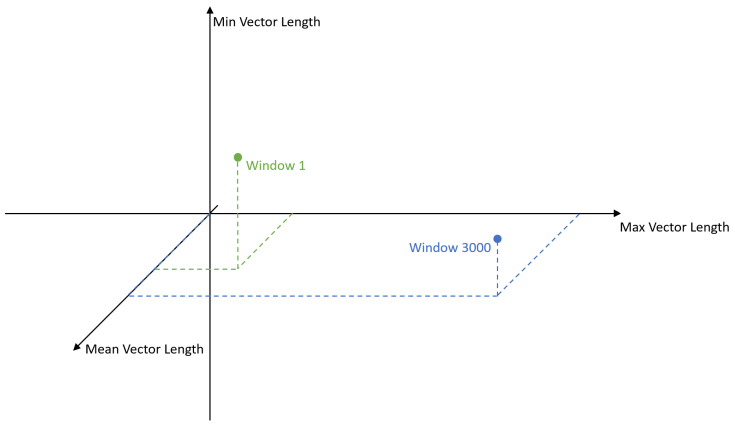


Abbildung 13: Datenpunkte der Windows 1 und 3000

In Abbildung 14 sind die Datenpunkte aller Windows der Person mit UserID 1 graphisch dargestellt.

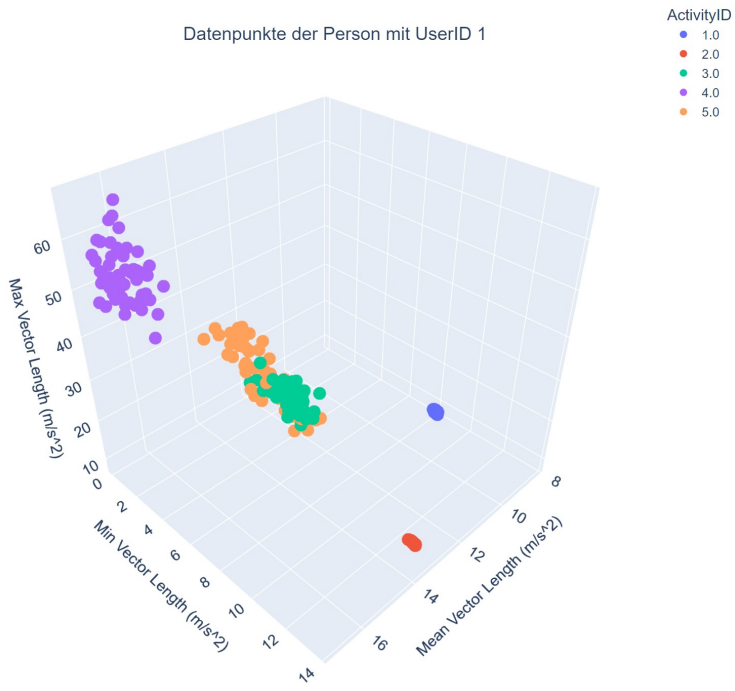


Abbildung 14: Datenpunkte aller Windows der Person mit UserID 1

In der Abbildung ist zu erkennen, dass die Datenpunkte der Aktivitäten 3 und 5 (Gehen und Treppen auf- und absteigen) sehr nahe aneinander liegen und sich die Datenwolken zum Teil überschneiden.

Bei der Klassifikation von Sensordaten dieser beiden Aktivitäten mit dem kNN-Algorithmus kann es daher zu Problemen kommen, denn unter den k nächsten Nachbarn eines Datenpunktes können auch Datenpunkte der „falschen“ Aktivität liegen. Die Datenpunkte der anderen Aktivitäten liegen weit auseinander, sodass hier weniger Probleme bei der Klassifikation zu erwarten sind.

Zur Berechnung des Abstands zwischen den Datenpunkten zweier Windows wird im Lernmodul zunächst die euklidische Metrik verwendet. Die k nächsten Nachbarn eines ausgewählten Test-Windows werden bestimmt, indem die Abstände des Test-Windows zu allen anderen Windows berechnet und die k kleinsten Abstände ermittelt werden. Nachdem die k nächsten Nachbarn des Test-Windows gefunden wurden, wird die Aktivität des Test-Windows über einen Mehrheitsentscheid bestimmt. Die Aktivität, die unter den k nächsten Nachbarn am häufigsten vorkommt, bestimmt die Aktivität des Test-Windows. Im sehr seltenen Falle eines Gleichstands werden hier zunächst beide Aktivitäten ausgegeben, die unter den k nächsten Nachbarn am häufigsten vorkommen. Später wird im Falle eines Gleichstands anhand der Anzahl der Datenpunkte einer Aktivität entschieden. Die Aktivität, zu der mehr Datenpunkte im Datensatz gehören, „gewinnt“ den Mehrheitsentscheid.

Lektion 5.4 Bewertung der Ergebnisse der Klassifikation (*Arbeitsblatt 4*)

Zur Bewertung der Ergebnisse der Klassifikation wird das Prinzip des überwachten maschinellen Lernens angewandt (s. Abschn. 2.2). Das Feature-Set 1 wird im Verhältnis 80:20 in Trainings- und Testdaten aufgeteilt. Der Trainingsdatensatz besteht somit aus 2400 Datenpunkten und der Testdatensatz aus 600 Datenpunkten. Für die Anzahl der betrachteten nächsten Nachbarn wird $k = 3$ gewählt.

Die Ergebnisse der Klassifikation der Testdaten werden in Form einer Wahrheitsmatrix ausgegeben (s. Tab. 3).

Tabelle 3: Wahrheitsmatrix für die Klassifikation der Testdaten mit dem Feature-Set 1 und $k = 3$

	Klassifiziert als 1 (Sitzen)	**Klassifiziert als 2 (Stehen)**	**Klassifiziert als 3 (Gehen)**	**Klassifiziert als 4 (Laufen)**	**Klassifiziert als 5 (Treppen auf- und absteigen)**
Tatsächlich 1 (Sitzen)	123	0	0	0	0
Tatsächlich 2 (Stehen)	0	115	0	0	0
Tatsächlich 3 (Gehen)	0	0	79	10	19
Tatsächlich 4 (Laufen)	0	0	14	120	1
Tatsächlich 5 (Treppen auf- und absteigen)	1	0	24	1	93

Insgesamt werden 530 Windows richtig und 70 Windows falsch klassifiziert. Bei der Aktivität 1 (Sitzen) werden mit 123 Datenpunkten die meisten Datenpunkte richtig zugeordnet, während bei Aktivität 3 (Gehen) mit 29 Datenpunkten die meisten Datenpunkte falsch klassifiziert werden. Zwischen den beiden Aktivitäten 3 (Gehen) und 5 (Treppen auf- und absteigen) liegen die meisten Überschneidungen vor. Anstelle von Gehen werden 19 Datenpunkte als Treppen auf- und absteigen klassifiziert und 24 Datenpunkte werden anstelle von Treppen auf- und absteigen als Gehen klassifiziert.

Mit Hilfe der Wahrheitsmatrix können die in Abschnitt 3.3 vorgestellten Qualitätsmaße berechnet werden. Es bezeichne CM_{ij} den Eintrag in Spalte j und Zeile i der Wahrheitsmatrix sowie CM_{Sum} die Summer aller Einträge in der Wahrheitsmatrix. Die Genauigkeit ist dann gegeben durch

$$\text{accuracy} = \frac{CM_{11} + CM_{22} + CM_{33} + CM_{44} + CM_{55}}{CM_{\text{Sum}}} \approx 0.88. \tag{26}$$

Für die Fehlerrate ergibt sich

$$\text{error rate} = 1 - \text{accuracy} \approx 0.12, \tag{27}$$

und die Präzision der einzelnen Aktivitäten ist gegeben durch

$$\begin{aligned}
\text{precision}_{\text{Sitzen}} &= \frac{CM_{11}}{CM_{11}+CM_{21}+CM_{31}+CM_{41}+CM_{51}} \approx 0.99,\\
\text{precision}_{\text{Stehen}} &= \frac{CM_{22}}{CM_{12}+CM_{22}+CM_{32}+CM_{42}+CM_{52}} = 1.0,\\
\text{precision}_{\text{Gehen}} &= \frac{CM_{33}}{CM_{13}+CM_{23}+CM_{33}+CM_{43}+CM_{53}} \approx 0.68,\\
\text{precision}_{\text{Laufen}} &= \frac{CM_{44}}{CM_{14}+CM_{24}+CM_{34}+CM_{44}+CM_{54}} \approx 0.92,\\
\text{precision}_{\text{Treppen}} &= \frac{CM_{55}}{CM_{15}+CM_{25}+CM_{35}+CM_{45}+CM_{55}} \approx 0.82.
\end{aligned} \tag{28}$$

Da der Wert der Präzision für die Aktivitäten Sitzen und Stehen nahe bei eins liegt, kann geschlussfolgert werden, dass nahezu alle Windows, die einer der beiden Aktivitäten zugeordnet werden, auch tatsächlich zu dieser Aktivität gehören. Die Präzision ist für die Aktivitäten Gehen und Treppen auf- und absteigen am kleinsten. Das liegt daran, dass die Datenpunkte der beiden Aktivitäten sehr nahe aneinander liegen und es dadurch häufiger zu falschen Klassifikationen kommt (s. Abb. 14). Für diese beiden Aktivitäten sind die Ergebnisse der Klassifikation noch nicht zufriedenstellend. Daher wird der entwickelte Klassifikationsalgorithmus im Folgenden durch zwei Veränderungen verbessert.

Lektion 5.5 Verbesserung des Klassifikationsalgorithmus (*Arbeitsblatt 5 & 6*)

Am entwickelten Klassifikationsalgorithmus werden zwei Veränderungen vorgenommen, um die Klassifikationsergebnisse zu verbessern. Zum einen wird das Feature-Set 1 um weitere Features ergänzt, zum anderen wird die Wahl des Parameters k optimiert.

Erweiterung des Feature-Sets 1 (*Arbeitsblatt 5*)

Das Feature-Set 1 wird um die Kenngrößen Minimum und Maximum sowie das arithmetische Mittel jeder der drei Beschleunigungsrichtungen (a_x, a_y und a_z) erweitert. Zudem werden der Median sowie das untere und das obere Quartil des Betrags des Beschleunigungsvektors ergänzt. Der neu entstandene Datensatz wird als *Feature-Set 2* bezeichnet (s. Abb. 15).

WindowID	UserID	ActivityID	Mean Vector Length (m/s^2)	Max Vector Length (m/s^2)	Min Vector Length (m/s^2)	Median Vector Length (m/s^2)	Upper Quartile Vector Length (m/s^2)	Lower Quartile Vector Length (m/s^2)
1.0	1.0	1.0	8.382189	8.593413	8.136967	8.385889	8.437079	8.323079
2.0	1.0	1.0	8.335617	8.571866	8.052737	8.338597	8.377425	8.323079
3.0	1.0	1.0	8.289633	8.384486	8.059064	8.290660	8.312062	8.323079
4.0	1.0	1.0	8.262474	8.533645	7.956085	8.275744	8.313050	8.323079
5.0	1.0	1.0	8.268336	8.325962	8.224459	8.269243	8.277988	8.323079

Mean x (m/s^2)	Max x (m/s^2)	Min x (m/s^2)	Mean y (m/s^2)	Max y (m/s^2)	Min y (m/s^2)	Mean z (m/s^2)	Max z (m/s^2)	Min z (m/s^2)
5.505176	5.615423	5.450765	-4.469158	-4.271966	-4.689299	-6.878246	-6.785539	-7.006031
5.519143	5.657186	5.373675	-4.416821	-4.172723	-4.578230	-6.900961	-6.728059	-7.023245
5.486834	5.620662	5.331613	-4.393822	-4.272116	-4.490362	-6.943257	-6.869365	-7.107370
5.509859	5.862559	5.325925	-4.353382	-4.147874	-4.539909	-6.948716	-6.761290	-7.173083
5.513971	5.550308	5.474566	-4.356670	-4.315825	-4.397256	-6.946351	-6.911877	-6.987170

Abbildung 15: Ausschnitt des Feature-Sets 2

Nach der Erweiterung des Feature-Sets 1 werden die Daten erneut dem Klassifikationsalgorithmus übergeben. Für die Anzahl der nächsten Nachbarn wird wieder $k = 3$ gewählt. Tabelle 4 zeigt die Ergebnisse in Form einer Wahrheitsmatrix.

Tabelle 4: Wahrheitsmatrix für die Klassifikation der Testdaten mit dem Feature-Set 2 und $k = 3$

	Klassifiziert als 1 (Sitzen)	**Klassifiziert als 2 (Stehen)**	**Klassifiziert als 3 (Gehen)**	**Klassifiziert als 4 (Laufen)**	**Klassifiziert als 5 (Treppen auf- und absteigen)**
Tatsächlich 1 (Sitzen)	123	0	0	0	0
Tatsächlich 2 (Stehen)	0	115	0	0	0
Tatsächlich 3 (Gehen)	0	0	105	2	1
Tatsächlich 4 (Laufen)	0	0	2	133	10
Tatsächlich 5 (Treppen auf- und absteigen)	0	0	10	2	107

Die Qualitätsmaße verbessern sich durch die Erweiterung des Feature-Sets 1 zu

$$\begin{aligned} \text{accuracy} &\approx 0.97, \\ \text{error rate} &\approx 0.03, \\ \text{precision}_{\text{Sitzen}} &= 1.0, \\ \text{precision}_{\text{Stehen}} &= 1.0, \\ \text{precision}_{\text{Gehen}} &\approx 0.90, \\ \text{precision}_{\text{Laufen}} &\approx 0.97, \\ \text{precision}_{\text{Treppen}} &\approx 0.99. \end{aligned} \tag{29}$$

Im Mittel werden nun 97 von 100 Aktivitäten richtig klassifiziert. Bei der Aktivität 3 (Gehen) treten weiterhin die meisten Fehlklassifikationen auf.

Optimierung des Parameters k (*Arbeitsblatt 6*)

Zur Bestimmung der optimalen Anzahl der nächsten Nachbarn k wird folgendes Optimierungsproblem gelöst:

Optimierungsproblem:
Finde die Anzahl der Nachbarn k, für die die Fehlerrate minimal wird.

Zunächst wird die Fehlerrate gegen die Anzahl der Nachbarn k aufgetragen (s. Abb. 16).

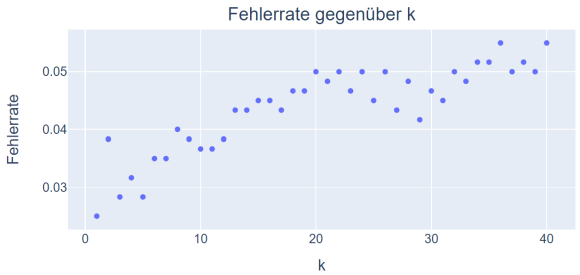


Abbildung 16: Fehlerrate gegenüber der Anzahl der Nachbarn k

Es ist zu erkennen, dass die Fehlerrate für $k = 1$ am kleinsten ist. Außerdem zeigt sich, dass für kleine k die Fehlerrate stark schwankt. Das liegt daran, dass für kleine und gerade k die Wahrscheinlichkeit für das Auftreten eines „Unentschieden“ beim Mehrheitsentscheid sehr groß ist. Für große k nimmt diese Wahrscheinlichkeit ab, weshalb die Schwankungen kleiner werden. Falls ein Unentschieden vorliegt, wird anhand der Daten im Trainingsdatensatz entschieden. Die Klasse, zu der mehr Datenpunkte im Trainingsdatensatz gehören, „gewinnt“ den Mehrheitsentscheid.

Anschließend wird das optimale k auch rechnerisch ermittelt, indem mit Hilfe einer for-Schleife die Fehlerraten für alle k zwischen 1 und 40 bestimmt und in einer Liste abgespeichert werden. Die Position der kleinsten Fehlerrate in dieser Liste liefert das optimale k. Auch dieses algorithmische Vorgehen liefert $k = 1$ als optimale Wahl des Parameters.

Die Wahrheitsmatrix für die Klassifikation der Testdaten mit $k = 1$ und dem Feature-Set 2 ist in Tabelle 5 zu sehen.

Tabelle 5: Wahrheitsmatrix für die Klassifikation der Testdaten mit dem Feature-Set 2 und $k = 1$

	Klassifiziert als 1 (Sitzen)	**Klassifiziert als 2 (Stehen)**	**Klassifiziert als 3 (Gehen)**	**Klassifiziert als 4 (Laufen)**	**Klassifiziert als 5 (Treppen auf- und absteigen)**
Tatsächlich 1 (Sitzen)	123	0	0	0	0
Tatsächlich 2 (Stehen)	0	115	0	0	0
Tatsächlich 3 (Gehen)	0	0	105	2	1
Tatsächlich 4 (Laufen)	0	0	2	133	10
Tatsächlich 5 (Treppen auf- und absteigen)	0	0	10	2	107

Die Qualitätsmaße für die Klassifikation mit dem Feature-Set 2 und $k = 1$ sind gegeben durch

$$\begin{aligned} \text{accuracy} &\approx 0.98, \\ \text{error rate} &\approx 0.02, \\ \text{precision}_{\text{Sitzen}} &= 1.0, \\ \text{precision}_{\text{Stehen}} &= 1.0, \\ \text{precision}_{\text{Gehen}} &\approx 0.90, \\ \text{precision}_{\text{Laufen}} &\approx 0.98, \\ \text{precision}_{\text{Treppen}} &= 1.0. \end{aligned} \tag{30}$$

Im Vergleich zur ersten Klassifikation mit $k = 3$ und dem Feature-Set 1 (s. Abschn. 5.4) konnten die Ergebnisse der Klassifikation durch die beiden Veränderungen des Klassifikationsalgorithmus deutlich verbessert werden. Für die aufgenommenen Sensordaten liefert das entwickelte Modell zufriedenstellende Klassifikationsergebnisse und eignet sich damit zur Aktivitätserkennung.

Lektion 5.6 Schwierigkeiten des Klassifikationsalgorithmus (*Arbeitsblatt 7*)

Eine große Herausforderung der Aktivitätserkennung ist die Positionierung und Ausrichtung der Sensoren am Körper des Benutzers. Um zu testen, wie gut der entwickelte Klassifikationsalgorithmus mit diesem Problem umgeht, wurden weitere Daten aufgenommen. Einmal befand sich das Handy während der Datenaufnahme in der Hand des Nutzers und das andere Mal war es in einem Rucksack verstaut. Für die neu aufgenommenen Daten wurden die Features des Feature-Sets 2 berechnet. Bei den im Folgenden dargestellten Klassifikationsergebnissen werden jeweils die neu aufgenommenen Daten als Testdaten und das Feature-Set 2 des ursprünglichen Datensatzes als Trainingsdaten verwendet. Für die Anzahl der betrachteten Nachbarn wird $k = 1$ gewählt.

Für den Fall, dass sich das Handy in der Hand des Nutzers befand, ergibt sich die in Tabelle 6 dargestellte Wahrheitsmatrix.

Tabelle 6: Wahrheitsmatrix für die Klassifikation der Testdaten, bei denen sich das Handy in der Hand des Nutzers befand, mit dem Feature-Set 2 und $k = 1$

	Klassifiziert als 1 (Sitzen)	**Klassifiziert als 2 (Stehen)**	**Klassifiziert als 3 (Gehen)**	**Klassifiziert als 4 (Laufen)**	**Klassifiziert als 5 (Treppen auf- und absteigen)**
Tatsächlich 1 (Sitzen)	60	0	0	0	0
Tatsächlich 2 (Stehen)	60	0	0	0	0
Tatsächlich 3 (Gehen)	60	0	0	0	0
Tatsächlich 4 (Laufen)	11	0	29	2	18
Tatsächlich 5 (Treppen auf- und absteigen)	60	0	10	2	107

Fast alle Testdatenpunkte werden fälschlicherweise der Aktivität Sitzen zugeordnet, weshalb die einzelnen Qualitätsmaße nun deutlich schlechter ausfallen. Diese sind gegeben durch

$$\begin{aligned} \text{accuracy} &\approx 0.20, \\ \text{error rate} &\approx 0.80, \\ \text{precision}_{\text{Sitzen}} &\approx 0.24, \\ \text{precision}_{\text{Gehen}} &= 0.0, \\ \text{precision}_{\text{Laufen}} &= 1.0, \\ \text{precision}_{\text{Treppen}} &= 0.0. \end{aligned} \tag{31}$$

Bei der Aktivität Stehen tritt der unbestimmte Ausdruck $\frac{0}{0}$ auf, weshalb für diese Aktivität keine Präzision berechnet werden konnte.

Für die Klassifikation der Sensordaten, bei denen das Handy im Rucksack verstaut war, ergibt sich die in Tabelle 7 dargestellte Wahrheitsmatrix.

Tabelle 7: Wahrheitsmatrix für die Klassifikation der Testdaten, bei denen sich das Handy im Rucksack des Nutzers befand, mit dem Feature-Set 2 und $k = 1$

	Klassifiziert als 1 (Sitzen)	**Klassifiziert als 2 (Stehen)**	**Klassifiziert als 3 (Gehen)**	**Klassifiziert als 4 (Laufen)**	**Klassifiziert als 5 (Treppen auf- und absteigen)**
Tatsächlich 1 (Sitzen)	60	0	0	0	0
Tatsächlich 2 (Stehen)	0	60	0	0	0
Tatsächlich 3 (Gehen)	0	0	0	0	60
Tatsächlich 4 (Laufen)	0	0	0	0	60
Tatsächlich 5 (Treppen auf- und absteigen)	0	0	26	0	34

Die Ergebnisse sind etwas besser als für die Daten, bei denen das Handy in der Hand gehalten

wurde. Die einzelnen Qualitätsmaße sind gegeben durch

$$\begin{aligned} \text{accuracy} &\approx 0.51, \\ \text{error rate} &\approx 0.49, \\ \text{precision}_{\text{Sitzen}} &= 1.0, \\ \text{precision}_{\text{Stehen}} &= 1.0, \\ \text{precision}_{\text{Gehen}} &= 0.0, \\ \text{precision}_{\text{Treppen}} &\approx 0.22. \end{aligned} \tag{32}$$

Diesmal tritt bei der Aktivität Laufen der unbestimmte Ausdruck $\frac{0}{0}$ auf, weshalb für diese Aktivität keine Präzision berechnet werden konnte.

Der entwickelte Klassifikationsalgorithmus funktioniert also nur für die Beschleunigungsdaten sehr gut, bei denen sich das Handy in der vorderen rechten Hosentasche befand. Das liegt daran, dass die charakteristischen Ausschläge der Beschleunigungswerte einer bestimmten Achse je nach Orientierung des Smartphones nun auf einer anderen Achse zu finden sind. Befindet sich das Handy zum Beispiel beim Stehen in der Hosentasche wirkt die Erdbeschleunigung auf die y-Achse des Accelerometers. Wird das Handy nun aber in der Hand gehalten, zum Beispiel um (Chat-)Nachrichten zu lesen, wirkt die Erdbeschleunigung auf die z-Achse. Da dem Klassifikationsalgorithmus nur Daten, bei denen sich das Handy in der vorderen rechten Hosentasche befand, als Trainingsdaten zur Verfügung stehen, kann er die neuen Daten, bei denen sich das Handy in der Hand oder im Rucksack des Nutzers befand, nicht richtig zuordnen.

Das Problem der Positionierung und Ausrichtung der Sensoren am Körper des Nutzers könnte durch die Einbeziehung weiterer Sensoren gelöst werden. Beispielsweise lassen sich die Messwerte des Accelerometers mit Hilfe der Messwerte des Gyroskops und des Magnetometers in Erdkoordinaten umrechnen. Bei der späteren Klassifikation liegen dann alle Sensordaten in Erdkoordinaten vor, sodass das Problem der Positionierung und Ausrichtung der Sensoren am Körper des Nutzers umgangen werden kann (vgl. [20], S. 1430).

Lektion 5.7 Auswirkungen anderer Abstandsmetriken (*Zusatzblatt*)

Die Ergebnisse der Klassifikation sind von der verwendeten Metrik zur Abstandsberechnung abhängig. Daher wird im Folgenden untersucht, wie sich die Ergebnisse verändern, wenn unterschiedliche Metriken verwendet werden. Die folgenden Klassifikationen werden immer mit dem Feature-Set 1 und $k = 3$ durchgeführt, d.h. die Ergebnisse sind mit denen aus Abschnitt 5.4 zu vergleichen.

Zuerst wird anstelle der euklidischen Metrik die Manhattan-Metrik verwendet. Für diesen Fall sind die Ergebnisse in Form einer Wahrheitsmatrix in Tabelle 8 zu sehen.

Tabelle 8: Wahrheitsmatrix für die Klassifikation der Testdaten mit der Manhattan-Metrik, dem Feature-Set 1 und $k = 3$

	Klassifiziert als 1 (Sitzen)	**Klassifiziert als 2 (Stehen)**	**Klassifiziert als 3 (Gehen)**	**Klassifiziert als 4 (Laufen)**	**Klassifiziert als 5 (Treppen auf- und absteigen)**
Tatsächlich 1 (Sitzen)	123	0	0	0	0
Tatsächlich 2 (Stehen)	0	115	0	0	0
Tatsächlich 3 (Gehen)	0	0	77	9	22
Tatsächlich 4 (Laufen)	0	0	16	118	1
Tatsächlich 5 (Treppen auf- und absteigen)	0	0	24	1	94

Die Qualitätsmaße für die Klassifikation mit der Manhattan-Metrik ergeben sich zu

$$\begin{aligned} \text{accuracy} &\approx 0.88, \\ \text{error rate} &\approx 0.12, \\ \text{precision}_{\text{Sitzen}} &= 1.0, \\ \text{precision}_{\text{Stehen}} &= 1.0, \\ \text{precision}_{\text{Gehen}} &\approx 0.66, \\ \text{precision}_{\text{Laufen}} &\approx 0.92, \\ \text{precision}_{\text{Treppen}} &\approx 0.80. \end{aligned} \tag{33}$$

Im Vergleich zur Klassifikation mit der euklidischen Metrik sind die Ergebnisse etwas schlechter. Das liegt daran, dass bei der Manhattan-Metrik große Abweichungen auf einer Achse weniger stark ins Gewicht fallen als bei der euklidischen Metrik. So ist beispielsweise der Abstand eines Windows, das auf einer Achse sehr weit entfernt, aber auf den anderen beiden Achsen sehr nahe beim Test-Window liegt, genauso groß wie der Abstand eines Windows, das auf allen Achsen leichte Abweichungen zeigt. Während die Daten des zweiten Windows höchst wahrscheinlich zur Aktivität des Test-Windows gehören, stammen die Daten des ersten Windows vermutlich von einer anderen Aktivität. Dennoch kann es passieren, dass das erste Window unter den k nächsten Nachbarn des Test-Windows liegt. Damit steigt die Wahrscheinlichkeit für eine Fehlklassifikation.

Als zweite Metrik wurde die Kosinus-Metrik verwendet. Die Ergebnisse sind in der folgenden Wahrheitsmatrix zu sehen (s. Tab. 9).

Tabelle 9: Wahrheitsmatrix für die Klassifikation der Testdaten mit der Kosinus-Metrik, dem Feature-Set 1 und $k = 3$

	Klassifiziert als 1 (Sitzen)	**Klassifiziert als 2 (Stehen)**	**Klassifiziert als 3 (Gehen)**	**Klassifiziert als 4 (Laufen)**	**Klassifiziert als 5 (Treppen auf- und absteigen)**
Tatsächlich 1 (Sitzen)	80	42	0	0	1
Tatsächlich 2 (Stehen)	38	77	0	0	0
Tatsächlich 3 (Gehen)	0	0	61	20	27
Tatsächlich 4 (Laufen)	0	0	26	100	9
Tatsächlich 5 (Treppen auf- und absteigen)	0	0	39	13	67

Die Qualitätsmaße für die Klassifikation mit der Kosinus-Metrik sind gegeben durch

$$\begin{aligned} \text{accuracy} &\approx 0.64, \\ \text{error rate} &\approx 0.36, \\ \text{precision}_{\text{Sitzen}} &\approx 0.68, \\ \text{precision}_{\text{Stehen}} &\approx 0.65, \\ \text{precision}_{\text{Gehen}} &\approx 0.48, \\ \text{precision}_{\text{Laufen}} &\approx 0.75, \\ \text{precision}_{\text{Treppen}} &\approx 0.64. \end{aligned} \tag{34}$$

Im Gegensatz zur Manhattan-Metrik und der euklidischen Metrik sind die Ergebnisse der Klassifikation mit der Kosinus-Metrik deutlich schlechter. Das liegt daran, dass der Abstand zwischen zwei Datenpunkten hier über den Winkel zwischen den beiden Ortsvektoren der beiden Punkte bestimmt wird. Die Ortsvektoren von Datenpunkten unterschiedlicher Aktivitäten können in die gleiche Richtung zeigen, sich aber in ihrer Länge unterscheiden. In diesem Fall wäre der Winkel zwischen den Ortsvektoren und damit der Abstand zwischen den beiden Datenpunkten klein, obwohl die beiden Punkte zu unterschiedlichen Aktivitäten gehören.

Als letzte Metrik wird die Tschebyschew-Metrik verwendet. Tabelle 10 zeigt die Ergebnisse in Form einer Wahrheitsmatrix.

Tabelle 10: Wahrheitsmatrix für die Klassifikation der Testdaten mit der Tschebyschew-Metrik, dem Feature-Set 1 und $k = 3$

	Klassifiziert als 1 (Sitzen)	**Klassifiziert als 2 (Stehen)**	**Klassifiziert als 3 (Gehen)**	**Klassifiziert als 4 (Laufen)**	**Klassifiziert als 5 (Treppen auf- und absteigen)**
Tatsächlich 1 (Sitzen)	123	0	0	0	0
Tatsächlich 2 (Stehen)	0	115	0	0	0
Tatsächlich 3 (Gehen)	0	0	76	12	20
Tatsächlich 4 (Laufen)	0	0	14	121	0
Tatsächlich 5 (Treppen auf- und absteigen)	0	0	20	2	97

Die Qualitätsmaße sind gegeben durch

$$\begin{aligned} \text{accuracy} &\approx 0.89, \\ \text{error rate} &\approx 0.11, \\ \text{precision}_{\text{Sitzen}} &= 1.0, \\ \text{precision}_{\text{Stehen}} &= 1.0, \\ \text{precision}_{\text{Gehen}} &\approx 0.69, \\ \text{precision}_{\text{Laufen}} &\approx 0.90, \\ \text{precision}_{\text{Treppen}} &\approx 0.83. \end{aligned} \tag{35}$$

Verglichen mit den anderen drei Metriken liefert die Tschebyschew-Metrik die besten Ergebnisse. Das liegt daran, dass immer nur die Achse, auf der der Abstand zwischen den beiden Datenpunkten zweier Windows am größten ist, beachtet wird. Sobald ein Window auf einer Achse deutliche Unterschiede zum Test-Window zeigt, ist der Abstand groß. Nur wenn die Werte aller Achsen im Bereich des Test-Windows liegen ist der Abstand klein. Es werden also nur Windows, bei denen sich alle Features den Features des Test-Windows ähneln, der Aktivität des Test-Windows zugeordnet.

Lektion Literatur

[1] https://www.youtube.com/watch?v=09LotPHTZtU, letzter Aufruf am 22.09.2022

[2] https://de.wikipedia.org/wiki/Manhattan-Metrik, letzter Aufruf am 22.09.2022

[3] https://de.wikipedia.org/wiki/Global_System_for_Mobile_Communications, letzter Aufruf am 13.10.2022

[4] Banos, O., Galvez, J.-M., Damas, M., Pomares, H. & Rojas, I. (2014). Window Size Impact in Human Activity Recognition. Sensors, 14, 6474-6499.

[5] Chomboon, K., Chujai, P., Teerarassamee, P., Kerdprasop, K. & Kerdprasop, N. (2015). An Empirical Study of Distance Metrics for k-Nearest Neighbor Algorithm. Proceedings of the 3rd International Conference on Industrial Application Engineering 2015, 280-285.

[6] Dittrich, F. (2014). Überblick menschlicher Aktivitätserkennung auf mobilen Geräten. Möglichkeiten, Grenzen und Herausforderungen. In M. A. Neumann, A. Bachmann, Y. Ding & T. Riedel (Hrsg.), Ubiquitäre Systeme (Seminar) und Mobile Computing (Proseminar) SS 2014. Mobile und Verteilte Systeme Ubiquitous Computing Teil XI. Karlsruhe: Karlsruher Institut für Technologie (KIT), Fakultät für Informatik, Lehrstuhl für Pervasive Computing Systems (PCS) und TECO.

[7] Frochte, J. (2021). Maschinelles Lernen: Grundlagen und Algorithmen in Python. München: Carl Hanser Verlag.

[8] Géron, A. (2020). Praxiseinstieg Machine Learning mit Scikit-Learn, Keras und TensorFlow: Konzepte, Tools und Techniken für intelligente System. Heidelberg: O'Reilly.

[9] Henze, N. (2018). Stochastik für Einsteiger. Eine Einführung in die faszinierende Welt des Zufalls. Wiesbaden: Springer Spektrum.

[10] Incel, O. D., Kose, M. & Ersoy, C. (2013). A Review and Taxonomy of Activity Recognition on Mobile Phones. BioNanoSci, 3 , 145-171.

[11] Krause, M. & Natterer, E. (2019). Maschinelles Lernen. Wie sich Computer an Probleme anpassen. In K. Kersting, C. Lampert & C. Rothkopf (Hrsg.), Wie Maschinen lernen. Künstliche Intelligenz verständlich erklärt (S. 21-27). Wiesbaden: Springer.

[12] Kwapisz, J. R., Weiss, G. M. & Moore, S. A. (2010). Activity Recognition using Cell Phone Accelerometers. SIGKDD Explorations, 12 , 74-82.

[13] Lockhart, J. W., Pulickal, T. & Weiss, G. M. (2012). Applications of Mobile Activity Recognition. UbiComp' 12.

[14] Mitchell, T. M. (1997). Machine Learning. New York: McGraw-Hill. (http://www.cs.cmu.edu/~tom/mlbook.html. letzter Aufruf: 13.10.2022)

[15] Mohsen, S., Elkaseer, A. & Scholz, S. G. (2022). Human Activity Recognition Using K-Nearest Neighbor Machine Learning Algorithm. In S. G. Scholz, R. J. Howlett & R. Setchi (Hrsg.), Sustainable Design and Manufacturing (Bd. 262, S. 304-313). Springer Singapore.

[16] Paaß, G. & Hecker, D. (2020). Künstliche Intelligenz. Was steckt hinter der Technologie der Zukunft. Wiesbaden: SpringerVieweg.

[17] Planing, P. (2022). Statistik Grundlagen. https://statistikgrundlagen.de/ebook/ (letzter Aufruf: 16.09.2022).

[18] Schönbrodt, S. (2022). Optimierungsprobleme in der mathematischen Modellierung. Grundlegende Aspekte und Chancen aus Sicht der Mathematikdidaktik - herausgestellt an aktuellen Problemen aus der Forschung zu künstlicher Intelligenz und erneuerbaren Energien. Dissertation, Karlsruher Institut für Technologie (KIT).

[19] Su, X., Tong, H. & Ji, P. (2014). Activity Recognition with Smartphone Sensors. Tsinghua Science and Technology, 19 (3), 235-249.

[20] Ustev, Y. E., Incel, O. D. & Ersoy, C. (2013). User, Device and Orientation Independent Human Activity Recognition on Mobile Phones: Challanges and a Proposal. UbiComp'13 , 1427-1435.

F.2. Methodisches Konzept

Methodisches Konzept für die Dozenten

Material:

Antwortblätter (ausdrucken):
- Antwortblatt1.pdf
- Antwortblatt2.pdf
- Antwortblatt3.pdf
- Antwortblatt4.pdf
- Antwortblatt5.pdf
- Antwortblatt6.pdf
- Antwortblatt7.pdf
- AntwortblattZusatz.pdf
- ZusammenfassungsAB.pdf

Jupyter Notebooks:
- AB1-SuS.ipynb
- AB2-SuS.ipynb
- AB2short-SuS.ipynb
- AB3-SuS.ipynb
- AB3short-SuS.ipynb
- AB4-SuS.ipynb
- AB5-SuS.ipynb
- AB6-SuS.ipynb
- AB7-SuS.ipynb
- ZusatzAB-SuS.ipynb

Notwendige Ordner auf Server:
- code
- data
- figs
- help
- printables
- worksheets

Präsentationen:
- EinstiegsPresentation.pptx
- ZwischenAbschlussPresentation.pptx
- opening presentation.pdf
- closing presentation.pdf

Begleitmaterial:
- Notizen EinstiegsPresentation.pdf
- Notizen ZwischenAbschlussPresentation.pdf
- MusterloesungDozenten.pdf
- BasisPaper.pdf

Ablauf:

	Inhalt	Hinweise
08:00 Uhr	Vorbereitungen	• Allgemeine Vorbereitungen (Laptops vorbereiten, …) • Präsentationen öffnen • AB1-SuS.ipynb öffnen
08:30 Uhr	Begrüßung / Einführung	• Eröffnungs- und Modellierungsvortrag halten (opening presentation.pdf) • Problemeinführungsvortrag halten (EinstiegsPresentation.pptx) • Kurze Einführung in Python und Jupyter Notebooks geben mit Hilfe des AB1-SuS.ipynb
1. Erkunden des Datensatzes		
09:00 Uhr	Arbeitsphase 1	• Antwortblatt 1 austeilen (Antwortblatt1.pdf) • Arbeitsblatt 1 (AB1-SuS.ipynb) bearbeiten lassen • Für schnelle SuS: Zusatzaufgabe am Ende von Arbeitsblatt 1 bearbeiten lassen
09:30 Uhr	Plenumsdiskussion 1	• Folien zu Arbeitsblatt 1 des Zwischenvortrags durchgehen und anhand dessen die Ergebnisse des ersten Arbeitsblattes besprechen (ZwischenAbschlussPresentation.pptx)

2. Vorverarbeitung der Daten		
09:40 Uhr	Arbeitsphase 2	• Antwortblatt 2 austeilen (**Antwortblatt2.pdf**) • Arbeitsblatt 2 (**AB2-SuS.ipynb**) bearbeiten lassen • Falls noch **keine** Vektorrechnung eingeführt wurde, **Kurzversion** von Arbeitsblatt 2 (**AB2short-SuS.ipynb**) bearbeiten lassen
10:10 Uhr	Plenumsdiskussion 2	• Folien zu Arbeitsblatt 2 des Zwischenvortrags durchgehen und anhand dessen die Ergebnisse des zweiten Arbeitsblattes besprechen (**ZwischenAbschlussPresentation.pptx**)
10:20 Uhr	Pause	
3. Der k-nächste-Nachbarn-Algorithmus		
10:30 Uhr	Arbeitsphase 3	• Antwortblatt 3 austeilen (**Antwortblatt3.pdf**) • Arbeitsblatt 3 (**AB3-SuS.ipynb**) bearbeiten lassen • Falls noch **keine** Vektorrechnung eingeführt wurde, **Kurzversion** von Arbeitsblatt 3 (**AB3short-SuS.ipynb**) bearbeiten lassen • Für schnelle SuS: Zusatzaufgabe am Ende von Arbeitsblatt 3 bearbeiten lassen (nur bei der Langversion von Arbeitsblatt 3)
11:00 Uhr	Plenumsdiskussion 3	• Folien zu Arbeitsblatt 3 des Zwischenvortrags durchgehen und anhand dessen die Ergebnisse des dritten Arbeitsblattes besprechen (**ZwischenAbschlussPresentation.pptx**)
4. Bewertung der Klassifikationsergebnisse		
11:10 Uhr	Arbeitsphase 4	• Antwortblatt 4 austeilen (**Antwortblatt4.pdf**) • Arbeitsblatt 4 (**AB4-SuS.ipynb**) bearbeiten lassen • Für schnelle SuS: Zusatzblatt (**ZusatzAB-SuS.ipynb**) bearbeiten lassen (nur für leistungsstarke SuS bzw. SuS, die die Vektorrechnung bereits kennen)
11:40 Uhr	Plenumsdiskussion 4	• Folien zu Arbeitsblatt 4 des Zwischenvortrags durchgehen und anhand dessen die Ergebnisse des vierten Arbeitsblattes besprechen (**ZwischenAbschlussPresentation.pptx**)
11:50 Uhr	Mittagspause	
5. Erweiterung des Feature-Sets 1		
12:50 Uhr	Arbeitsphase 5	• Antwortblatt 5 austeilen (**Antwortblatt5.pdf**) • Arbeitsblatt 5 (**AB5-SuS.ipynb**) bearbeiten lassen

13:15 Uhr	Plenumsdiskussion 5	• Folien zu Arbeitsblatt 5 des Zwischenvortrags durchgehen und anhand dessen die Ergebnisse des fünften Arbeitsblattes besprechen (ZwischenAbschlussPresentation.pptx)
6. Optimierung des Parameters k		
13:20 Uhr	Arbeitsphase 6	• Antwortblatt 6 austeilen (Antwortblatt6.pdf) • Arbeitsblatt 6 (AB6-SuS.ipynb) bearbeiten lassen
13:45 Uhr	Plenumsdiskussion 6	• Folien zu Arbeitsblatt 6 des Zwischenvortrags durchgehen und anhand dessen die Ergebnisse des sechsten Arbeitsblattes besprechen (ZwischenAbschlussPresentation.pptx)
13:50 Uhr	Pause	
7. Probleme und Schwierigkeiten sowie Anwendungen und Einsatzgebiete der Aktivitätserkennung		
14:00 Uhr	Arbeitsphase 7	• Antwortblatt 7 austeilen (Antwortblatt7.pdf) • Arbeitsblatt 7 (AB7-SuS.ipynb) bearbeiten lassen
14:20 Uhr	Plenumsdiskussion 7	• Folien zu Arbeitsblatt 7 des Zwischenvortrags durchgehen und anhand dessen die Ergebnisse des siebten Arbeitsblattes besprechen (ZwischenAbschlussPresentation.pptx) • Die einzelnen Schritte des Workshops nochmal mit Hilfe der Folie 74 und des Zusammenfassungsarbeitsblattes (ZusammenfassungsAB.pdf) besprechen
14:40 Uhr	Tagesabschluss und Evaluation	• Abschlussvortrag halten (closing presentation.pdf) • Evaluation ausfüllen lassen

F.3. Musterlösung

Musterlösung für Dozenten

Lösungen zu Arbeitsblatt 1 | Erkunden des Datensatzes

Aufgabe 1 | Die Aktivitäten

Teil a | Der Datensatz und die Anzahl der Aktivitäten

- Aus welchen Daten besteht ein Datenpunkt im Datensatz (= Zeile in der Tabelle)?
  → *Jeder Datenpunkt besteht aus der UserID und ActivityID sowie der Zeit und den Beschleunigungswerten in allen drei Raumdimensionen x, y und z.*

- Zu wie vielen Aktivitäten wurden Daten aufgenommen?
  → *Wird der Datensatz nach der Spalte „ActivityID" sortiert, kann die kleinste und größte ActivityID (1 und 5) abgelesen werden.*
  → *Es wurden Daten zu 5 Aktivitäten aufgenommen.*

Teil b | Die Art der Aktivitäten

- Zu welchen Aktivitäten wurden Daten aufgenommen?
  → *Das NaN muss nacheinander durch eine ActivityID zwischen 1 und 5 ersetzt werden.*
  → *ActivityID 1: Sitzen*
  → *ActivityID 2: Stehen*
  → *ActivityID 3: Gehen*
  → *ActivityID 4: Laufen*
  → *ActivityID 5: Treppen auf- und absteigen*

Teil c | Graphische Darstellung der unterschiedlichen Aktivitäten

- Es wird ein Graph, in dem die Beschleunigungswerte jeder der drei Raumdimensionen gegen die Zeit aufgetragen sind, für die gewählte Aktivität angezeigt.
  → *Das NaN muss durch eine ActivityID zwischen 1 und 5 ersetzt werden.*

Teil d | Interpretation der graphischen Darstellung

- Welche Unterschiede sind zwischen den einzelnen Aktivitäten zu erkennen?
  → *Bei den Aktivitäten Sitzen und Stehen liegen die Datenpunkte der einzelnen Raumdimensionen auf einer Geraden.*
  → *Bei den Aktivitäten Gehen, Laufen und Treppen auf- und absteigen liegen die Datenpunkte nicht auf einer Geraden, stattdessen sind deutliche Schwankungen zu erkennen.*

- Bei welcher Aktivität sind am wenigsten / am meisten Ausschläge zu erkennen und warum?
  → *Die wenigsten Ausschläge sind bei den Aktivitäten Sitzen und Stehen zu erkennen. Bei diesen Aktivitäten befindet sich das Smartphone in Ruhe. Die Beschleunigung ist also konstant.*
  → *Die meisten Ausschläge sind bei der Aktivität Laufen zu erkennen, da hier die meisten Schritte innerhalb einer gewissen Zeitspanne gemacht werden und sich am stärksten vom Boden abgedrückt wird. Daher sind die Beschleunigungswerte am größten und die meisten Ausschläge innerhalb einer gewissen Zeitspanne zu erkennen.*

- Auf welcher der drei Achsen sind die größten Ausschläge zu erkennen und warum?
  → *Auf der y-Achse sind die größten Ausschläge zu erkennen.*
  → *Der Grund dafür ist die Erdbeschleunigung. Die Erdbeschleunigung wirkt bei allen Aktivitäten bis auf Sitzen auf die y-Achse des Beschleunigungsmessers, denn die Erdbeschleunigung ist zum Erdmittelpunkt hin gerichtet. Befindet sich das Handy also senkrecht in der vorderen Hosentasche wirkt die Erdbeschleunigung entweder in positive oder negative y-Richtung, je nachdem wie das Handy in der Hose orientiert ist (Kamera oben oder unten).*

Aufgabe 2 | Die Testpersonen

- Wie viele Testpersonen haben Daten aufgenommen?
  → *Insgesamt haben 10 Personen Daten zu den unterschiedlichen Aktivitäten aufgenommen.*

- Wie viele Testpersonen sind weiblich und wie viele sind männlich?
  → *5 Testpersonen sind weiblich und 5 Testpersonen sind männlich.*

- Wie alt ist die jüngste bzw. älteste Testperson?
  → *Die jüngste Testperson ist 18 Jahre alt und die älteste Testperson ist 52 Jahre alt.*

Aufgabe 3 | Die Datenaufnahme

Teil a | Die Daten zur Aktivität 1 von Nutzer 1

- Es wird eine Tabelle angezeigt, in der die ersten fünf und die letzten fünf Datenpunkte zur Aktivität 1 von Testperson 1 eingetragen sind.

Teil b | Die Dauer und Abtastrate der Datenaufnahme

- Wie lange wurden Daten zu den jeweiligen Aktivitäten von den Testpersonen aufgenommen?
  → *Aus der Tabelle in Teil a lässt sich die Zeit des ersten und des letzten Datenpunkts der Daten zu Aktivität 1 von Testperson 1 ablesen. Die Datenaufnahme startet bei 0 s und endet bei 180 s und dauert somit 180 Sekunden, also 3 Minuten.*

- Mit welcher Abtastrate wurden die Daten aufgenommen?
  → *Aus der Tabelle in Teil a lässt sich ablesen, dass innerhalb der 180 Sekunden 7150 Datenpunkte aufgenommen wurden. Damit ergibt sich die Abtastrate zu*

$$\text{samplingRate} = \frac{7150}{180} \approx 40\ \text{Hz}$$

Lösungen zu Arbeitsblatt 2 | Vorverarbeitung der Daten

Aufgabe 1 | Der Betrag des Beschleunigungsvektors

Teil a | Berechnung des Betrags (fällt bei AB2short weg)

- Formel für den Betrag des Beschleunigungsvektors:
  → *Mit dem Satz des Pythagoras ergibt sich folgende Formel für den Betrag des Beschleunigungsvektors:*

$$\text{vectorLength} = \sqrt{{a_x}^2 + {a_y}^2 + {a_z}^2}.$$

Teil b | Ergänzung im Datensatz

- Es wird eine Tabelle ausgegeben, in der die ersten zehn Datenpunkte des Datensatzes mit der zusätzlichen Spalte für den Betrag des Beschleunigungsvektors eingetragen sind.

Aufgabe 2 | Einteilung der Windows

Teil a | Die Länge der Windows

- Mögliche Längen der Windows:
  → *Die Länge der Windows sollte ein ganzzahliger Teiler von 180 sein, sollte so klein wie möglich sein, sollte aber auch mindestens so groß sein, dass mehrere charakteristische Bewegungen einer Aktivität innerhalb dieser Zeitspanne durchgeführt werden.*
  → *Mögliche Längen der Windows, die diese Anforderungen erfüllen, sind dann: 3, 4, 5 und 6 Sekunden.*

Teil b | Ergänzung im Datensatz

- Es wird eine Tabelle angezeigt, in der die ersten fünf und die letzten fünf Datenpunkte des Datensatzes mit der zusätzlichen Spalte für die WindowID eingetragen sind.

Aufgabe 3 | Wahl der Features

Teil a | Statistische Kenngrößen

- Folgende statistischen Kenngrößen sollten aus dem Mathematikunterricht bereits bekannt sein:
  → *Mittelwert*
  → *Minimum*
  → *Maximum*
  → *Modalwert*
  → *Spannweite*
  → *Median*
  → *Oberes Quartil*
  → *Unteres Quartil*
  → *Quartilsabstand*
  → *etc.*

Teil b | Graphische Darstellung

- Es wird ein Graph, in dem die Beschleunigungswerte jeder der drei Achsen sowie der Betrag des Beschleunigungsvektors gegen die Zeit aufgetragen sind, für die gewählte WindowID angezeigt.
  → *Das NaN muss durch eine WindowID zwischen 1 und 300 ersetzt werden.*

- Statistische Kenngrößen, die zur Beschreibung der Daten eines Windows genutzt werden können:
  → *Mittelwert,*
  → *Maximum,*
  → *Minimum,*
  → *etc.*

Teil c | Statistische Kenngrößen berechnen

- Berechnung der statistischen Kenngrößen für den Betrag des Beschleunigungsvektors von sieben Datenpunkte aus dem Window mit der WindowID 121:
  → *Mittelwert des Betrags des Beschleunigungsvektors:*

$$\text{meanVectorLength} = \frac{1}{7} \cdot (11.3 + 9.8 + 2.6 + 9.7 + 12.6 + 3.8 + 5.7) \approx 7.93$$

  → *Minimum des Betrags des Beschleunigungsvektors:*

$$\text{minVectorLength} = 2.6$$

  → *Maximum des Betrags des Beschleunigungsvektors:*

$$\text{maxVectorLength} = 12.6$$

Teil d | Automatische Berechnung der statistischen Kenngrößen

- Für eine gewählte WindowID werden die statistischen Kenngrößen Mittelwert, Minimum und Maximum des Betrags des Beschleunigungsvektors berechnet und ausgegeben.
  → *Das NaN muss durch eine WindowID zwischen 1 und 3000 ersetzt werden.*

Teil e | Abspeichern der statistischen Kenngrößen

- Es wird eine Tabelle angezeigt, in der die ersten fünf und die letzten fünf Datenpunkte des neuen Datensatzes (Feature-Set 1) eingetragen sind.

Lösungen zu Arbeitsblatt 3 | Der k-nächste-Nachbarn-Algorithmus

Aufgabe 1 | Die Grundidee des k-nächste-Nachbarn-Algorithmus

- Klasse, der das rote x zugeordnet werden sollte:
  → *Unter den 5 nächsten Nachbarn des roten x befinden sich vier Datenpunkte der Klasse zwei (orangene Quadrate) und ein Datenpunkt der Klasse 3 (grüne Kreise). Somit wird das rote x der Klasse zwei zugeordnet, da diese Klasse unter den fünf nächsten Nachbarn des roten x am häufigsten vorkommt.*

Aufgabe 2 | Die Abstandsfunktion

Teil a | Graphische Darstellung der Datenpunkte

- Es wird eine Graphik ausgegeben, in der die Datenpunkte des Feature-Sets 1 für den Nutzer mit UserID 1 eingetragen sind.

Teil b | Interpretation der graphischen Darstellung

- Bei welchen Aktivitäten könnte der k-nächste-Nachbarn-Algorithmus bei der Klassifikation Schwierigkeiten haben und warum?
  → *Bei den Aktivitäten 3 und 5 kann es zu Problemen kommen, da hier die Datenpunkte sehr eng aneinander liegen bzw. sich die Datenwolken zum Teil auch überschneiden.*
  → *Daher können unter den k nächsten Nachbarn eines Datenpunktes auch Datenpunkte der „falschen" Aktivität liegen.*

Teil c | Definition der Abstandsfunktion (fällt bei AB3short weg)

- Formel für den Abstand zwischen den beiden Datenpunkten $P_1(x_1, y_1, z_1)$ und $P_2(x_2, y_2, z_2)$:
  → *Der Verbindungsvektor zwischen den beiden Punkten P_1 und P_2 ist gegeben durch*

$$\vec{d} = \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{pmatrix}.$$

  → *Mit dem Satz des Pythagoras ergibt sich folgende Formel für den Abstand zwischen den beiden Datenpunkten P_1 und P_2:*

$$\text{distanceP1P2} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}.$$

Teil d | Berechnung des Abstands zwischen zwei Windows

- Für zwei gewählte Windows wird der Abstand zwischen den Datenpunkten der beiden Windows berechnet und ausgegeben.
  → *Die beiden NaN's müssen durch zwei WindowIDs zwischen 1 und 3000 ersetzt werden.*

Aufgabe 3 | Suche der k nächsten Nachbarn

Teil a | Aufstellen einer Suchfunktion (fällt bei AB3short weg)

- Code, mit dem die k nächsten Nachbarn von einem ausgewählten Datenpunkt bestimmt werden können:

```
k = 5;
TestWindow = 170;

# Leere Liste, in der nach und nach die Abstände gespeichert werden sollen.
distances = []

# For-Schleife zur Berechnung der Abstände zwischen allen Windows und dem Test-Window
# und Abspeichern in der Liste distances
for i in range(1, 3000):
    dist = computeDistance(i, TestWindow)
    addDistances(i, dist, distances)

# Löschen des Abstands des Test-Windows zu sich selbst
del distances[TestWindow-1]

# Sortieren der Liste nach den kürzesten Abständen zum Test-Window
sortDistances(distances)

# Filtern der k nächsten Nachbarn
kneighbors = getNeighbors(distances, k)

# Hier nichts ändern!
checkKNeighbors(kneighbors, k, TestWindow)
```

  → *Die fünf nächsten Nachbarn zu Test-Window 170 sind (WindowID, Abstand, ActivityID):*

$$(132, 0.25, 3.0)$$
$$(155, 0.47, 3.0)$$
$$(136, 0.59, 3.0)$$
$$(139, 0.70, 3.0)$$
$$(156, 0.75, 3.0)$$

Teil b | Interpretation der Ergebnisse

- Welche Aktivität würdest du dem von dir gewählten Test-Window zuordnen und warum?
  → *Da die Aktivität 3 (Gehen) unter den fünf nächsten Nachbarn am häufigsten vorkommt, sollte dem Test-Window 170 die Aktivität 3 zugeordnet werden.*

Aufgabe 4 | Häufigkeiten der einzelnen Klassen

Teil a | Mehrheitsentscheid

- Für das gewählte Test-Window wird die Aktivität, die unter den k nächsten Nachbarn des Test-Windows am häufigsten vorkommt, bestimmt und ausgegeben.
  → *Das erste NaN muss durch die Anzahl der nächsten Nachbarn und das zweite NaN durch die WindowID des Test-Windows ersetzt werden.*
  → *Unter den fünf nächsten Nachbarn des Datenpunktes, der zu WindowID 170 gehört, kommt die Aktivität mit der ActivityID 3 (Gehen) am häufigsten vor.*

Teil b | Überprüfung des Ergebnisses

- Für das gewählte Test-Window wird die tatsächliche Aktivität ausgegeben, zu der die Daten aufgenommen wurden.
  → *Das NaN muss durch die WindowID des gewählten Test-Windows ersetzt werden.*
  → *Das Window mit der WindowID 170 enthält Daten zur Aktivität mit der ActivityID 3 (Gehen).*

- Stimmen die beiden Aktivitäten von Teil a und Teil b überein?
  → *Ja, die Aktivitäten von Teil a und Teil b stimmen für das Window mit der WindowID 170 überein.*

Zusatzaufgabe | Weitere Abstandsfunktionen

- Weitere Abstandsfunktionen, mit denen der Abstand zwischen den beiden Punkten $P_1(x_1, y_1, z_1)$ und $P_2(x_2, y_2, z_2)$ bestimmt werden kann:

  → *Euklidische Distanz:*

  $$d_{\text{Euklid}} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}.$$

  → *Manhattan-Distanz:*

  $$d_{Manhattan} = |x_2 - x_1| + |y_2 - y_1| + |z_2 - z_1|.$$

  → *Kosinus-Distanz:*

  $$d_{Kosinus} = 1 - \cos(\theta) = 1 - \frac{\overrightarrow{p_1} \cdot \overrightarrow{p_2}}{|\overrightarrow{p_1}| \cdot |\overrightarrow{p_2}|} = 1 - \frac{x_1 \cdot x_2 + y_1 \cdot y_2 + z_1 \cdot z_2}{\sqrt{{x_1}^2 + {y_1}^2 + {z_1}^2} \cdot \sqrt{{x_2}^2 + {y_2}^2 + {z_2}^2}}.$$

  → *Tschebyschew-Distanz:*

  $$d_{Tschebyschew} = \max(|x_2 - x_1|, |y_2 - y_1|, |z_2 - z_1|).$$

  → *etc.*

Lösungen zu Arbeitsblatt 4 | Bewertung der Klassifikationsergebnisse

Aufgabe 1 | Die Trainings- und Testdaten

Teil a | Unterteilung in Features und Aktivitäten

- Es wird ein Array, in dem alle Feature-Werte abgespeichert sind, und ein Array, in dem alle Aktivitäten abgespeichert sind, ausgegeben.

Teil b | Unterteilung in Trainings- und Testdaten

- Es werden die Daten des Feature-Sets 1 in Trainings- und Testdaten aufgeteilt und die Größe der beiden Datensätze ausgegeben.
  → *Der Trainingsdatensatz umfasst 2400 Datenpunkte und der Testdatensatz beinhaltet 600 Datenpunkte.*

Aufgabe 2 | Die Wahrheitsmatrix

Teil a | Die Wahrheitsmatrix zur Klassifikation mit Feature-Set 1

- Es wird die Wahrheitsmatrix für die Klassifikation der Testdaten mit $k = 3$ und dem Feature-Set 1 ausgegeben:

	Klassifiziert als 1 (Sitzen)	Klassifiziert als 2 (Stehen)	Klassifiziert als 3 (Gehen)	Klassifiziert als 4 (Laufen)	Klassifiziert als 5 (Treppen auf- und absteigen)
Tatsächlich 1 (Sitzen)	123	0	0	0	0
Tatsächlich 2 (Stehen)	0	115	0	0	0
Tatsächlich 3 (Gehen)	0	0	79	10	19
Tatsächlich 4 (Laufen)	0	0	14	120	1
Tatsächlich 5 (Treppen auf- und absteigen)	1	0	24	1	93

Teil b | Interpretation der Wahrheitsmatrix

- Wie viele Datenpunkte wurden insgesamt richtig klassifiziert?
  → *Um die Anzahl der richtig klassifizierten Datenpunkte bestimmen zu können, müssen wir die Diagonaleinträge addieren.*
  → *Insgesamt wurden also 530 Datenpunkte richtig klassifiziert.*

- Wie viele Datenpunkte wurden insgesamt falsch klassifiziert?
  → *Um die Anzahl der falsch klassifizierten Datenpunkte bestimmen zu können, müssen wir alle Einträge bis auf die Diagonaleinträge addieren.*
  → *Insgesamt wurden also 70 Datenpunkte falsch klassifiziert.*

- Bei welcher Aktivität wurden die meisten Datenpunkte richtig klassifiziert?
  → *Bei Aktivität 1 (Sitzen) wurden mit 123 Datenpunkte die meisten Datenpunkte richtig klassifiziert.*

- Bei welcher Aktivität wurden die meisten Datenpunkte falsch klassifiziert?
  → *Bei Aktivität 3 (Gehen) wurden mit 29 Datenpunkte die meisten Datenpunkte falsch klassifiziert.*

- Bei welchen beiden Aktivitäten liegen die meisten Überschneidungen vor?
  → *Bei Aktivität 3 (Gehen) und Aktivität 5 (Treppen auf- und absteigen) liegen die meisten Überschneidungen vor.*
  → *19 Datenpunkte werden anstelle von Gehen als Treppen auf- und absteigen klassifiziert.*
  → *24 Datenpunkte werden anstelle von Treppen auf- und absteigen als Gehen klassifiziert.*

Aufgabe 3 | Verschiedene Qualitätsmaße

Teil a | Die Genauigkeit

- Formel zur Berechnung der Genauigkeit:
  → *Die Genauigkeit gibt das Verhältnis der richtig klassifizierten Datenpunkte zu allen Datenpunkten an. Die Formel für die Genauigkeit lautet also:*

$$\text{accuracy} = \frac{CM_{11} + CM_{22} + CM_{33} + CM_{44} + CM_{55}}{CM_{Sum}}$$

  → *Für die Klassifikation mit $k = 3$ und dem Feature-Set 1 liegt der Wert der Genauigkeit bei* 0.88.

Teil b | Die Fehlerrate

- Formel zur Berechnung der Fehlerrate:
  → *Die Fehlerrate gibt das Verhältnis der falsch klassifizierten Datenpunkte zu allen Datenpunkten an. Die Formel für die Fehlerrate lautet also:*

$$\text{error rate} = 1 - \text{accuracy} = \frac{CM_{Sum} - CM_{11} - CM_{22} - CM_{33} - CM_{44} - CM_{55}}{CM_{Sum}}$$

  → *Für die Klassifikation mit $k = 3$ und dem Feature-Set 1 liegt der Wert der Fehlerrate bei* 0.12.

Teil c | Die Präzision

- Formel zur Berechnung der Präzision:
  → *Die Präzision gibt das Verhältnis der richtig als Aktivität z klassifizierten Datenpunkte zu allen als Aktivität z klassifizierten Datenpunkten an. Die Formeln für die Präzision der einzelnen Aktivitäten lauten also:*

$$\text{precisionSitzen} = \frac{CM_{11}}{CM_{11} + CM_{21} + CM_{31} + CM_{41} + CM_{51}},$$

$$\text{precisionStehen} = \frac{CM_{22}}{CM_{12} + CM_{22} + CM_{32} + CM_{42} + CM_{52}},$$

$$\text{precisionGehen} = \frac{CM_{33}}{CM_{13} + CM_{23} + CM_{33} + CM_{43} + CM_{53}},$$

$$\text{precisionLaufen} = \frac{CM_{44}}{CM_{14} + CM_{24} + CM_{34} + CM_{44} + CM_{54}},$$

$$\text{precisionTreppen} = \frac{CM_{55}}{CM_{15} + CM_{25} + CM_{35} + CM_{45} + CM_{55}}.$$

  → *Für die Klassifikation mit $k = 3$ und dem Feature-Set 1 ergeben sich die folgenden Werte für die Präzision der einzelnen Aktivitäten:*

$$\begin{gathered}\text{precisionSitzen} \approx 0.99,\\ \text{precisionStehen} = 1.0,\\ \text{precisionGehen} \approx 0.68,\\ \text{precisionLaufen} \approx 0.92,\\ \text{precisionTreppen} \approx 0.82.\end{gathered}$$

Teil d | Interpretation der Qualitätsmaße

- Was bedeutet es, wenn der Wert der Präzision nahe bei 1 liegt? Bei welchen Aktivitäten ist dies der Fall?
  → *Wenn der Wert der Präzision nahe bei 1 liegt, sind nahezu alle als Aktivität z klassifizierten Datenpunkte auch richtig als Aktivität z klassifziert.*
  → *Dies ist bei den Aktivitäten 1 und 2 (Sitzen und Stehen) der Fall.*

- Bei welcher Aktivität werden die Datenpunkte am häufigsten richtig zugeordnet und warum?
  → *Bei Aktivität 2 (Stehen) werden die Datenpunkte am häufigsten richtig zugeordnet, da hier die Präzision am größten ist.*

- Bei welchen beiden Aktivitäten werden die Datenpunkte am häufigsten falsch zugeordnet und was könnten die Gründe dafür sein?
  → *Bei den Aktivitäten 2 und 5 (Gehen und Treppen auf- und absteigen) werden die Datenpunkte am häufigsten falsch zugeordnet.*
  → *Das liegt daran, dass die Datenpunkte der beiden Klassen sehr nahe aneinander liegen und sich die Datenwolken zum Teil überschneiden.*
  → *Die charakteristischen Bewegungen von Gehen und Treppen auf- und absteigen sind außerdem sehr ähnlich. Ist beispielsweise ein Stockwerk erreicht, geht der Nutzer ein paar Schritte normal bis er wieder die nächste Treppenstufe hinauf bzw. hinabsteigt.*

- Sind die Ergebnisse der Klassifikation im Großen und Ganzen zufriedenstellend? Falls nein, wo müssten die Ergebnisse noch verbessert werden?
  → *Die Ergebnisse sind bei den Aktivitäten 3 und 5 (Gehen und Treppen auf- und absteigen) noch nicht zufriedenstellend.*
  → *Bei diesen Aktivitäten muss die Präzision noch verbessert werden.*

Aufgabe 4 | Ideen zur Verbesserung

- Mögliche Verbesserungsideen:
  → *Erweiterung des Feature-Sets 1*
  → *Optimierung der Wahl des Parameters k*

Lösungen zu Arbeitsblatt 5 | Erweiterung des Feature-Sets 1

Aufgabe 1 | Die erste Erweiterung

- Es wird eine Tabelle angezeigt, in der die ersten fünf und die letzten fünf Datenpunkte des erweiterten Feature-Sets 1 eingetragen sind.
- Das erweiterte Feature-Set 1 enthält nun neben den Features von Arbeitsblatt 2 noch den Mittelwert, das Maximum und das Minimum jeder der drei Beschleunigungsrichtungen x, y und z.

Aufgabe 2 | Die zweite Erweiterung

Teil a | Weitere statistische Kenngrößen

- Folgende statistischen Kenngrößen sollten neben dem Mittelwert, dem Maximum und dem Minimum bereits aus dem Mathematikunterricht bekannt sein:
  → *Modalwert*
  → *Spannweite*
  → *Median*
  → *Oberes Quartil*
  → *Unteres Quartil*
  → *Quartilsabstand*
  → …

Teil b | Die neuen statistischen Kenngrößen

- Berechnung der neuen statistischen Kenngrößen für den Betrag des Beschleunigungsvektors von sieben Datenpunkte aus dem Window mit der WindowID 121:
  → *Zunächst müssen die Messwerte der Größe nach (von klein nach groß) geordnet werden:*

$$2.6 \quad 3.8 \quad 5.7 \quad 9.7 \quad 9.8 \quad 11.3 \quad 12.6$$

  → *Median des Betrags des Beschleunigungsvektors:*
  Der Median ist der mittlere Wert der nach Größe geordneten Liste an Messwerten, somit gilt:

$$\text{medianVectorLength} = 9.7$$

  → *Oberes Quartil des Betrags des Beschleunigungsvektors:*
  Das obere Quartil ist der Median der oberen Hälfte der Messwerte, somit gilt:

$$\text{upperQuartileVectorLength} = 11.3$$

  → *Unteres Quartil des Betrags des Beschleunigungsvektors:*
  Das untere Quartil ist der Median der unteren Hälfte der Messwerte, somit gilt:

$$\text{lowerQuartileVectorLength} = 3.8$$

Teil c | Automatische Berechnung der neuen statistischen Kenngrößen

- Für eine gewählte WindowID werden die statistischen Kenngrößen Median, oberes Quartil und unteres Quartil des Betrags des Beschleunigungsvektors berechnet und ausgegeben.
  → *Das NaN muss durch eine WindowID zwischen 1 und 3000 ersetzt werden.*

Teil d | Abspeichern der neuen statistischen Kenngrößen

- Es wird eine Tabelle angezeigt, in der die ersten fünf und die letzten fünf Datenpunkte des neuen Feature-Sets 2 eingetragen sind.

Aufgabe 3 | Klassifikation mit Feature-Set 2

Teil a | Ergebnisse der Klassifikation mit Feature-Set 2

- Es werden die Wahrheitsmatrix sowie die verschiedenen Qualitätsmaße für die Klassifikation der Testdaten mit $k = 3$ und dem Feature-Set 2 ausgegeben:

	Klassifiziert als 1 (Sitzen)	Klassifiziert als 2 (Stehen)	Klassifiziert als 3 (Gehen)	Klassifiziert als 4 (Laufen)	Klassifiziert als 5 (Treppen auf- und absteigen)
Tatsächlich 1 (Sitzen)	123	0	0	0	0
Tatsächlich 2 (Stehen)	0	115	0	0	0
Tatsächlich 3 (Gehen)	0	0	105	2	1
Tatsächlich 4 (Laufen)	0	0	2	133	0
Tatsächlich 5 (Treppen auf- und absteigen)	0	0	10	2	107

  → *Genauigkeit der gesamten Klassifikation:*

$$\text{accuracy} \approx 0.97$$

  → *Fehlerrate der gesamten Klassifikation:*

$$\text{error rate} \approx 0.03$$

  → *Präzision der einzelnen Aktivitäten:*

$$\begin{aligned} \text{precisionSitzen} &= 1.0, \\ \text{precisionStehen} &= 1.0, \\ \text{precisionGehen} &\approx 0.90, \\ \text{precisionLaufen} &\approx 0.97, \\ \text{precisionTreppen} &\approx 0.99. \end{aligned}$$

Teil b | Interpretation der Ergebnisse

- Was fällt dir auf, wenn du die Qualitätsmaße mit den Qualitätsmaßen der Klassifikation auf Arbeitsblatt 4 vergleichst?
  → *Die Ergebnisse sind durchweg zufriedenstellender.*

- Bei welcher Aktivität treten noch am häufigsten falsche Klassifikationen auf?
  → *Bei Aktivität 3 (Gehen) treten noch am häufigsten falsche Klassifikationen auf (geringste Präzision).*

- Wie viele von 100 Aktivitäten werden im Mittel richtig klassifiziert?
  → *Im Mittel werden ca. 97 Aktivitäten von 100 richtig klassifiziert.*

Lösungen zu Arbeitsblatt 6 | Optimierung des Parameters k

Aufgabe 1 | Graphische Lösung des Optimierungsproblems

Teil a | Der Graph

- Es wird ein Graph, in dem die Fehlerrate error rate gegen die Anzahl der Nachbarn k aufgetragen ist, ausgegeben.

Teil b | Interpretation des Graphen

- Für welchen Wert von k ist die Fehlerrate am kleinsten?
  → *Für $k = 1$ ist die Fehlerrate am kleinsten.*

- Warum schwankt die Fehlerrate für kleine k stark? Wie hängen die Schwankungen mit dem Wert von k zusammen?
  → *Für kleine und gerade k ist die Wahrscheinlichkeit, dass ein „Unentschieden" beim Mehrheits-entscheid entsteht, am größten. Daher sind die Schwankungen für kleine k stärker als für große k.*
  → *Was passiert bei einem „Unentschieden"?*
  Bei einem „Unentschieden" wird anhand der Daten im Trainingsdatensatz entschieden. Die Aktivität, zu der mehr Datenpunkte im Trainingsdatensatz gehören, gewinnt den Mehrheitsentscheid.

Aufgabe 2 | Lösung des Optimierungsproblems mit Hilfe von Python

- Code, mit dem die Anzahl k der Nachbarn, für die die Fehlerrate minimal ist, bestimmt werden kann:

```
# leere Liste, in der die Fehlerraten abgespeichert werden
errorRates = []

# For-Schleife zur Berechnung der Fehlerraten und
# Abspeichern der Fehlerraten in der Liste errorRates
for i in range(1,41):
    errorRate_i = computeErrorRate(i)
    errorRates.append(errorRate_i)

# Suchen des kleinsten Werts in der Liste errorRates
min_errorRates = min(errorRates)

# Position des kleinsten Werts in der Liste errorRates
pos_min = errorRates.index(min_errorRates)

# Optimales k, für das die Fehlerrate am kleinsten ist
k = pos_min + 1

# Hier nichts ändern!
checkPosMinErrorRate(k, Min_ErrorRates, Pos_Min)
```

→ *Für $k = 1$ ist die Fehlerrate am kleinsten und liegt bei 0.025.*

Aufgabe 3 | Klassifikation mit optimalem k

Teil a | Ergebnisse der Klassifikation mit optimalem k

- Es werden die Wahrheitsmatrix sowie die verschiedenen Qualitätsmaße für die Klassifikation der Testdaten mit $k = 1$ und dem Feature-Set 2 ausgegeben:
  → *Das NaN muss durch das zuvor bestimmte optimale k, also 1 ersetzt werden.*

	Klassifiziert als 1 (Sitzen)	Klassifiziert als 2 (Stehen)	Klassifiziert als 3 (Gehen)	Klassifiziert als 4 (Laufen)	Klassifiziert als 5 (Treppen auf- und absteigen)
Tatsächlich 1 (Sitzen)	123	0	0	0	0
Tatsächlich 2 (Stehen)	0	115	0	0	0
Tatsächlich 3 (Gehen)	0	0	106	2	0
Tatsächlich 4 (Laufen)	0	0	2	133	0
Tatsächlich 5 (Treppen auf- und absteigen)	0	0	10	1	108

→ *Genauigkeit der gesamten Klassifikation:*

$$\text{accuracy} \approx 0.98$$

→ *Fehlerrate der gesamten Klassifikation:*

$$\text{error rate} \approx 0.02$$

→ *Präzision der einzelnen Aktivitäten:*

$$\begin{aligned} \text{precisionSitzen} &= 1.0, \\ \text{precisionStehen} &= 1.0, \\ \text{precisionGehen} &\approx 0.90, \\ \text{precisionLaufen} &\approx 0.98, \\ \text{precisionTreppen} &= 1.0. \end{aligned}$$

Teil b | Interpretation der Ergebnisse

- Was fällt dir auf, wenn du die Qualitätsmaße mit den Qualitätsmaßen der Klassifikation auf Arbeitsblatt 5 vergleichst?
  → *Die Ergebnisse werden nochmal ein bisschen besser.*

- Bei welcher Aktivität treten noch am häufigsten falsche Klassifikationen auf?
  → *Bei Aktivität 3 (Gehen) treten weiterhin noch am häufigsten falsche Klassifikationen auf (geringste Präzision).*

- Wie viele von 100 Aktivitäten werden im Mittel richtig klassifiziert?
  → *Im Mittel werden ca. 98 Aktivitäten von 100 richtig klassifiziert.*

Lösungen zu Arbeitsblatt 7 | Probleme und Schwierigkeiten sowie Anwendungen und Einsatzgebiete der Aktivitätserkennung

Aufgabe 1 | Herausforderungen und Schwierigkeiten

- Es werden zwei Tabellen angezeigt, in denen jeweils die ersten zehn Datenpunkte der beiden neuen Datensätze (Handy in der Hand des Nutzers und Handy im Rucksack des Nutzers) eingetragen sind.

Teil a | Vorverarbeitung der Daten

- Es werden vier Tabellen angezeigt, in denen jeweils die ersten fünf und die letzten fünf Datenpunkte des Feature-Sets 3 (Handy in der Hand des Nutzers) und Feature-Sets 4 (Handy im Rucksack des Nutzers) eingetragen sind.

Teil b | Handy befindet sich in der Hand des Nutzers

- Es werden die Wahrheitsmatrix sowie die verschiedenen Qualitätsmaße für die Klassifikation mit $k = 1$, dem Feature-Sets 2 (Trainingsdaten) und dem Feature-Set 3 (Testdaten) ausgegeben:

	Klassifiziert als 1 (Sitzen)	Klassifiziert als 2 (Stehen)	Klassifiziert als 3 (Gehen)	Klassifiziert als 4 (Laufen)	Klassifiziert als 5 (Treppen auf- und absteigen)
Tatsächlich 1 (Sitzen)	60	0	0	0	0
Tatsächlich 2 (Stehen)	60	0	0	0	0
Tatsächlich 3 (Gehen)	60	0	0	0	0
Tatsächlich 4 (Laufen)	11	0	29	2	18
Tatsächlich 5 (Treppen auf- und absteigen)	60	0	0	0	0

→ *Genauigkeit der gesamten Klassifikation:*

$$\text{accuracy} \approx 0.20$$

→ *Fehlerrate der gesamten Klassifikation:*

$$\text{error rate} \approx 0.80$$

→ *Präzision der einzelnen Aktivitäten:*

$$\begin{aligned} &\text{precisionSitzen} \approx 0.24, \\ &\text{precisionStehen} = \text{nan}, \\ &\text{precisionGehen} = 0.0, \\ &\text{precisionLaufen} = 1.0, \\ &\text{precisionTreppen} = 0.0. \end{aligned}$$

Teil c | Handy befindet sich im Rucksack des Nutzers

- Es werden die Wahrheitsmatrix sowie die verschiedenen Qualitätsmaße für die Klassifikation mit $k = 1$, dem Feature-Sets 2 (Trainingsdaten) und dem Feature-Set 4 (Testdaten) ausgegeben:

	Klassifiziert als 1 (Sitzen)	Klassifiziert als 2 (Stehen)	Klassifiziert als 3 (Gehen)	Klassifiziert als 4 (Laufen)	Klassifiziert als 5 (Treppen auf- und absteigen)
Tatsächlich 1 (Sitzen)	60	0	0	0	0
Tatsächlich 2 (Stehen)	0	60	0	0	0
Tatsächlich 3 (Gehen)	0	0	0	0	60
Tatsächlich 4 (Laufen)	0	0	0	0	60
Tatsächlich 5 (Treppen auf- und absteigen)	0	0	26	0	34

  → *Genauigkeit der gesamten Klassifikation:*

$$\text{accuracy} \approx 0.51$$

  → *Fehlerrate der gesamten Klassifikation:*

$$\text{error rate} \approx 0.49$$

  → *Präzision der einzelnen Aktivitäten:*

$$\begin{aligned} \text{precisionSitzen} &= 1.0, \\ \text{precisionStehen} &= 1.0, \\ \text{precisionGehen} &= 0.0, \\ \text{precisionLaufen} &= \text{nan}, \\ \text{precisionTreppen} &\approx 0.22. \end{aligned}$$

Teil d | Interpretation der Ergebnisse

- Mögliche Gründe, warum der Algorithmus die Sensordaten, bei denen sich das Handy in der Hand bzw. im Rucksack des Nutzers befand, nicht richtig klassifiziert:
  → *Die charakteristischen Ausschläge der Beschleunigungswerte einer bestimmten Achse sind je nach Orientierung des Smartphones nun auf einer anderen Achse zu finden.*
  → *Beispiel: Aktivität Stehen*
  Befindet sich das Handy in der Hosentasche wirkt die Erdbeschleunigung auf die y-Achse.
  Befindet sich das Handy jetzt aber in der Hand des Nutzers, während dieser z. B. eine Chat-Nachricht liest, wirkt die Erdbeschleunigung auf die z-Achse.

Teil e | Weitere Herausforderungen und Schwierigkeiten

- Weitere Herausforderungen und Schwierigkeiten der Aktivitätserkennung sowie mögliche Lösungsansätze:
  → *Subjektive Empfindlichkeit:*
  Die Genauigkeit der Aktivitätserkennung ist stark von den Testpersonen abhängig. Jeder Mensch hat unterschiedliche Gewohnheiten und Verhaltensweisen. Daher haben verschiedene Personen auch unterschiedliche Bewegungsmuster. Dieses Problem kann minimiert werden, indem so viele Daten wie möglich von so vielen unterschiedlichen Testpersonen wie möglich aufgenommen werden.
  → *Standortempfindlichkeit:*
  Sensordaten sind stark von der Ausrichtung und Position der Sensoren am Körper des Nutzers abhängig. Dies kann umgangen werden, indem weitere Sensoren miteinbezogen werden, sodass die Ausrichtung und Position der Sensoren vorab bestimmt werden kann bzw. die Sensordaten in Erdkoordinaten umgerechnet werden können.

→ *Komplexität der Aktivitäten:*
Werden mehrere Aktivitäten gleichzeitig ausgeführt, sind diese schwer zu unterscheiden bzw. zu erkennen. Auch beim Übergang zwischen zwei Aktivitäten kann es schnell zu Fehlklassifikationen kommen.
→ *Energie- und Ressourcenbeschränkung:*
Der Einsatz vieler Sensoren wirkt sich zum einen auf die Akkulaufzeit und zum anderen auf den Speicherplatz aus. Die Ausführung des Klassifikationsalgorithmus auf dem Smartphone kann also aufgrund der geringeren Rechenressourcen im Vergleich zu einem Laptop zu Problemen führen. Eine Lösungsmöglichkeit könnte hier die Auslagerung der Klassifikation auf einen Online-Server darstellen.
→ *Telefone werden nicht bei jeder Aktivität mitgeführt.*
→ *Privatsphäre und Datenschutz*

Aufgabe 2 | Anwendungen und Einsatzgebiete

- Bereiche, in denen die menschliche Aktivitätserkennung von Nutzen sein kann:

Anwendungen für den Endbenutzer
→ *Fitness-Tracking:*
Schrittzähler, täglicher Kalorienverbrauch, zurückgelegte Strecke, gestiegene Treppenstufen, etc.
→ *Gesundheitsüberwachung:*
Kontinuierliche Überwachung der Gewohnheiten und täglichen Aktivitäten von Patienten ermöglicht eine bessere Diagnose durch den Arzt. Physiotherapeuten können die Therapie auf die Gewohnheiten der Patienten abstimmen.
→ *Sturzerkennung:*
Ein automatischer Notruf und die Ortung des Smartphones können unmittelbar nach dem Sturz durchgeführt werden.
→ *Kontextbezogenes Verhalten:*
Smartphone vergrößert z. B. die Schrift, wenn der Nutzer beim Lesen einer Chat-Nachricht gerade geht.
→ *Smartphones sind auch eine geeignete Plattform, um Nutzer zu einer gesünderen Lebensweise zu motivieren.*

Anwendungen für Unternehmen und Dritte
→ *Gezielte Werbung:*
Werbung wird auf die täglichen Aktivitäten und Gewohnheiten der Nutzer abgestimmt und wirkt dadurch weniger aufdringlich, da sie für die Aktivitäten und Interessen der Nutzer relevant ist.
→ *Forschungsplattformen:*
Forscher in vielen Bereichen, vom Marketing bis zum Gesundheitswesen, werden immer daran interessiert sein, Aktivitätsdaten zu sammeln. Da Forschungsunternehmen häufig zuerst eine ausreichende Menge an Daten benötigen, besteht also ein Markt für aufgenommene Rohdaten.
→ *Unternehmensführung:*
Aktivitätserkennung kann bei der Mitarbeiterverwaltung und bei der Abrechnung der Arbeitszeiten helfen.
→ *Versicherungen:*
Manche Versicherungsgesellschaften bieten einen Rabatt bei der Autoversicherung an, wenn eine sichere Fahrweise erkannt wird.

Anwendungen für Gruppen und Crowds
→ *Aktivitätsbasierte soziale Netzwerke:*
Anwendung kann einen Freundeskreis vorschlagen, der ähnliche Aktivitätsmuster aufweist.
→ *Aktivitätsbezogenes Crowd-Sourcing:*
Es können Gegenden identifiziert werden, in denen bestimmte Aktivitäten z. B. Radfahren häufig ausgeführt werden. Einem Nutzer, der gerne Fahrrad fährt, können dann beliebte Fahrradstrecken vorgeschlagen werden. Außerdem kann das System dazu genutzt werden, um zu erkennen, ob sich die Aktivitäten in einer bestimmten Region stark von den normalerweise beobachteten Aktivitäten unterscheiden. Dadurch können möglicherweise Unfälle oder Katastrophen schneller erkannt werden.

Lösungen zum Zusatzblatt | Klassifikation mit unterschiedlichen Abstandsfunktionen

Aufgabe 1 | Klassifikation mit der Manhattan-Distanz

- Es werden die Wahrheitsmatrix sowie die verschiedenen Qualitätsmaße für die Klassifikation der Testdaten mit $k = 3$, dem Feature-Set 1 und der Manhattan-Distanz ausgegeben:

	Klassifiziert als 1 (Sitzen)	Klassifiziert als 2 (Stehen)	Klassifiziert als 3 (Gehen)	Klassifiziert als 4 (Laufen)	Klassifiziert als 5 (Treppen auf- und absteigen)
Tatsächlich 1 (Sitzen)	123	0	0	0	0
Tatsächlich 2 (Stehen)	0	115	0	0	0
Tatsächlich 3 (Gehen)	0	0	77	9	22
Tatsächlich 4 (Laufen)	0	0	16	118	1
Tatsächlich 5 (Treppen auf- und absteigen)	0	0	24	1	94

→ *Genauigkeit der gesamten Klassifikation:*

$$\text{accuracy} \approx 0.88$$

→ *Fehlerrate der gesamten Klassifikation:*

$$\text{error rate} \approx 0.12$$

→ *Präzision der einzelnen Aktivitäten:*

$$\begin{aligned}\text{precisionSitzen} &= 1.0,\\ \text{precisionStehen} &= 1.0,\\ \text{precisionGehen} &\approx 0.66,\\ \text{precisionLaufen} &\approx 0.92,\\ \text{precisionTreppen} &\approx 0.80.\end{aligned}$$

Aufgabe 2 | Klassifikation mit der Kosinus-Distanz

- Es werden die Wahrheitsmatrix sowie die verschiedenen Qualitätsmaße für die Klassifikation der Testdaten mit $k = 3$, dem Feature-Set 1 und der Kosinus-Distanz ausgegeben:

	Klassifiziert als 1 (Sitzen)	Klassifiziert als 2 (Stehen)	Klassifiziert als 3 (Gehen)	Klassifiziert als 4 (Laufen)	Klassifiziert als 5 (Treppen auf- und absteigen)
Tatsächlich 1 (Sitzen)	80	42	0	0	1
Tatsächlich 2 (Stehen)	38	77	0	0	0
Tatsächlich 3 (Gehen)	0	0	61	20	27
Tatsächlich 4 (Laufen)	0	0	26	100	9
Tatsächlich 5 (Treppen auf- und absteigen)	0	0	39	13	67

→ *Genauigkeit der gesamten Klassifikation:*

$$\text{accuracy} \approx 0.64$$

→ *Fehlerrate der gesamten Klassifikation:*

$$\text{error rate} \approx 0.36$$

→ *Präzision der einzelnen Aktivitäten:*

$$\begin{aligned}\text{precisionSitzen} &\approx 0.68,\\ \text{precisionStehen} &\approx 0.65,\\ \text{precisionGehen} &\approx 0.48,\\ \text{precisionLaufen} &\approx 0.75,\\ \text{precisionTreppen} &\approx 0.64.\end{aligned}$$

Aufgabe 3 | Klassifikation mit der Tschebyschew-Distanz

- Es werden die Wahrheitsmatrix sowie die verschiedenen Qualitätsmaße für die Klassifikation der Testdaten mit $k = 3$, dem Feature-Set 1 und der Tschebyschew-Distanz ausgegeben:

	Klassifiziert als 1 (Stehen)	Klassifiziert als 2 (Stehen)	Klassifiziert als 3 (Gehen)	Klassifiziert als 4 (Laufen)	Klassifiziert als 5 (Treppen auf- und absteigen)
Tatsächlich 1 (Sitzen)	123	0	0	0	0
Tatsächlich 2 (Stehen)	0	115	0	0	0
Tatsächlich 3 (Gehen)	0	0	76	12	20
Tatsächlich 4 (Laufen)	0	0	14	121	0
Tatsächlich 5 (Treppen auf- und absteigen)	0	0	20	2	97

→ *Genauigkeit der gesamten Klassifikation:*

$$\text{accuracy} \approx 0.89$$

→ *Fehlerrate der gesamten Klassifikation:*

$$\text{error rate} \approx 0.11$$

→ *Präzision der einzelnen Aktivitäten:*

$$\begin{aligned}\text{precisionSitzen} &= 1.0,\\ \text{precisionStehen} &= 1.0,\\ \text{precisionGehen} &\approx 0.69,\\ \text{precisionLaufen} &\approx 0.90,\\ \text{precisionTreppen} &\approx 0.83.\end{aligned}$$

Aufgabe 4 | Interpretation der Ergebnisse

- Welche Abstandsfunktion liefert die besten Klassifikationsergebnisse?
  → *Die Tschebyschew-Distanz liefert die besten Klassifikationsergebnisse.*

- Welche Abstandsfunktionen liefern bessere bzw. schlechtere Klassifikationsergebnisse im Vergleich zur euklidischen Distanz?
  → *Die Tschebyschew-Distanz liefert minimal bessere Klassifikationsergebnisse als die euklidische Distanz.*
  → *Die Manhattan-Distanz liefert ähnlich gute Klassifikationsergebnisse wie die euklidische Distanz.*
  → *Die Kosinus-Distanz liefert deutlich schlechtere Klassifikationsergebnisse als die euklidische Distanz.*

- Welche Abstandsfunktion liefert die schlechtesten Klassifikationsergebnisse und was könnten Gründe dafür sein?
  → *Die Kosinus-Distanz liefert die schlechtesten Klassifikationsergebnisse.*
  → *Das liegt daran, dass der Abstand zwischen zwei Datenpunkten hier über den Winkel zwischen den Ortsvektoren der beiden Punkte bestimmt wird. Die Ortsvektoren von Datenpunkten unterschiedlicher Aktivitäten können allerdings auch in die gleiche Richtung zeigen, sich aber in ihrer Länge unterscheiden. In diesem Fall wäre der Winkel zwischen den beiden Ortsvektoren und damit der Abstand zwischen den beiden Datenpunkten klein, obwohl die beiden Punkte zu unterschiedlichen Aktivitäten gehören.*

G. Fragebogen und Ergebnisse der Evaluation

G.1. Fragebogen

Fragebogen zum CAMMP Workshop

Liebe Schülerinnen und Schüler,
wir, das CAMMP Team, möchten Lerneinheiten für Dich vorbereiten und anbieten, die Dein Interesse wecken und die Du gerne durchführst. Wir sind bestrebt, die verschiedenen Lerneinheiten und den Ablauf des Besuchs ständig weiterzuentwickeln. Du kannst uns dabei unterstützen, indem Du den Fragebogen wahrheitsgetreu ausfüllst. Die Durchführung des Fragebogens dauert ca. 10 Minuten.

Ansprechpartnerin für diese Befragung:
Name: Katja Hoeffer
E-Mail: cammp@scc.kit.edu

Einwilligungserklärung

Hiermit willigst Du ausdrücklich ein, dass das Karlsruher Institut für Technologie und die Rheinisch-Westfälische Technische Hochschule Aachen zum Zwecke des Forschungsprojektes folgende personenbezogene Daten von Dir erheben, speichern und nutzen dürfen: Geschlecht, Schulart, Jahrgangsstufe, Alter und gegebenenfalls Wahl der Leistungskurse.

Es wird zugesichert, dass sämtliche Daten vertraulich behandelt und nur in anonymisierter Form veröffentlicht werden. Dabei werden keine Rückschlüsse auf die jeweilige Schule, einzelne Lehrkräfte oder einzelne Schüler/innen möglich sein.

Mir ist bewusst, dass die Einwilligung freiwillig ist und ohne Nachteile verweigert oder jederzeit auch abgebrochen oder ohne Angaben von Gründen widerrufen werden kann. Ich weiß, dass im Falle eines Widerrufs die Rechtmäßigkeit der aufgrund der Einwilligung bis zum Widerruf erfolgten Verarbeitung nicht berührt wird.

Hiermit bestätige ich, diese Informationen gelesen zu haben und dass ich mich freiwillig zur Teilnahme an dieser Studie entschlossen habe. Ich habe verstanden, dass ich mich für einen Widerruf einfach an die genannte Ansprechpartnerin wenden kann.

◯ Bestätigen

Bitte lies jede Frage genau durch und beantworte sie wahrheitsgemäß! Gib die Antwort an, die Dir am passendsten erscheint. Kreuze die Antwort an oder schreibe Deine Antwort in das freie Feld. Vielen Dank!

1. Angaben zur eigenen Person und Schule

Geschlecht:

- ◯ männlich
- ◯ weiblich
- ◯ divers

Alter in Jahren:

[Bitte auswählen]

Welche Jahrgangsstufe besuchst Du?

- ◯ 8
- ◯ 9
- ◯ 10
- ◯ 11
- ◯ 12
- ◯ 13
- ◯ andere:

Welche Schule besuchst Du?

- ◯ Förderschule / Sonderschule
- ◯ Gemeinschaftsschule
- ◯ Gesamtschule
- ◯ Gymnasium
- ◯ Hauptschule
- ◯ Realschule
- ◯ Sekundarschule
- ◯ Werkrealschule

Falls Du bereits in einer Klasse der Kursstufe bzw. Oberstufe bist, welche Leistungskurse besuchst Du?

- ☐ Mathematik
- ☐ Physik
- ☐ Informatik
- ☐ andere:

An welchem Workshop von CAMMP hast Du teilgenommen?

- ◯ Abkühlprozess von Metallen
- ◯ Animationsfilme
- ◯ Aktivitätserkennung
- ◯ Computertomographie
- ◯ Datenkomprimierung (Mittelstufe)
- ◯ Datenkomprimierung (Oberstufe)
- ◯ Evolution
- ◯ Fitnesstracker
- ◯ Google
- ◯ GPS
- ◯ Klimarekorde
- ◯ Klimawandel
- ◯ Machine Learning
- ◯ Netflix
- ◯ Shazam
- ◯ Solarkraftwerk (Mittelstufe)
- ◯ Solarkraftwerk (Oberstufe)
- ◯ Soziale Netzwerke
- ◯ Wortvorschläge
- ◯ andere: _____

2. Bewertung des Workshops

2.1. Gestaltung des Workshops

Im Folgenden findest du einige Aussagen über deine Teilnahme am CAMMP Workshop. Bitte gib auf einer Skala von „Stimmt völlig“ bis „Stimmt gar nicht“ jeweils an, wie sehr die Aussage zutrifft.

	Stimmt völlig	Stimmt eher	Stimmt eher nicht	Stimmt gar nicht	nicht beurteilbar
Die Arbeitsblätter waren ansprechend und übersichtlich gestaltet.	○	○	○	○	○
Die Aufgabenstellungen waren abwechslungsreich.	○	○	○	○	○
Die Lern- und Arbeitszeiten waren angemessen (nicht zu lang oder zu kurz mit ausreichenden Pausen).	○	○	○	○	○
Die Diskussionsphasen im Plenum nach jedem Arbeitsblatt waren sinnvoll.	○	○	○	○	○
Die Diskussionsphasen im Plenum sollten ausführlicher sein.	○	○	○	○	○
Im Workshop hatte ich das Gefühl, dass ich neue Inhalte selbstständig erarbeiten konnte.	○	○	○	○	○
Ich konnte meine eigenen Ideen in den Workshop einbringen (z.B. in den Diskussionen).	○	○	○	○	○

2.2. Schwierigkeitsgrad des Workshops

Im Folgenden findest du einige Aussagen über deine Teilnahme am CAMMP Workshop. Bitte gib auf einer Skala von „Stimmt völlig“ bis „Stimmt gar nicht“ jeweils an, wie sehr die Aussage zutrifft.

	Stimmt völlig	Stimmt eher	Stimmt eher nicht	Stimmt gar nicht	nicht beurteilbar
Ich konnte die Aufgaben eigenständig bearbeiten.	○	○	○	○	○
Die Aufgaben waren zu einfach.	○	○	○	○	○
Die Aufgaben waren zu schwierig.	○	○	○	○	○
Mein bisheriges mathematisches Vorwissen hat ausgereicht, um die Aufgaben bearbeiten zu können.	○	○	○	○	○

Das fand ich im Workshop bzw. an den Aufgaben (besonders) schwierig:

2.3. Verständlichkeit und Hilfestellung

Im Folgenden findest du einige Aussagen über deine Teilnahme am CAMMP Workshop. Bitte gib auf einer Skala von „Stimmt völlig“ bis „Stimmt gar nicht“ jeweils an, wie sehr die Aussage zutrifft.

	Stimmt völlig	Stimmt eher	Stimmt eher nicht	Stimmt gar nicht	nicht beurteilbar
Die Inhalte wurden in den Präsentationen gut erklärt.	○	○	○	○	○
Die Aufgabenstellungen waren klar und verständlich formuliert.	○	○	○	○	○
Die Grafiken und Abbildungen haben beim Verständnis der Inhalte geholfen.	○	○	○	○	○
Die Tipps / Hilfekarten waren hilfreich.	○	○	○	○	○
Bei der Bearbeitung der Aufgaben musste ich häufig die Tipps / Hilfekarten benutzen.	○	○	○	○	○

2.4. Modellierung und KI sowie Umgang mit Jupyter Notebooks und Python

Im Folgenden findest du einige Aussagen über deine Teilnahme am CAMMP Workshop. Bitte gib auf einer Skala von „Stimmt völlig“ bis „Stimmt gar nicht“ jeweils an, wie sehr die Aussage zutrifft.

	Stimmt völlig	Stimmt eher	Stimmt eher nicht	Stimmt gar nicht	nicht beurteilbar
Durch den Workshop habe ich mathematisches Modellieren besser begriffen.	○	○	○	○	○
Der Vortrag über mathematische Modellierung war hilfreich.	○	○	○	○	○
Durch den Workshop habe ich die mathematischen Hintergründe von KI-Systemen besser begriffen.	○	○	○	○	○
Die Einführung in Jupyter Notebooks und Python war hilfreich.	○	○	○	○	○
Der Umgang mit der Programmiersprache Python fiel mir schwer.	○	○	○	○	○
Ich hätte mir zu Beginn des Workshops eine umfangreichere Einführung in das Arbeiten mit Python gewünscht.	○	○	○	○	○
Die automatischen Rückmeldungen, die ich nach der Eingabe meiner Lösungen im Code erhalten habe, fand ich hilfreich.	○	○	○	○	○
Das Arbeiten mit den Codefeldern hat mir Spaß gemacht und den Workshop bereichert.	○	○	○	○	○
Ich habe bereits vor dem Workshop (z.B. in der Schule) mit Jupyter Notebooks und Python gearbeitet.	○	○	○	○	○

2.5. Motivation

Im Folgenden findest du einige Aussagen über deine Teilnahme am CAMMP Workshop. Bitte gib auf einer Skala von „Stimmt völlig“ bis „Stimmt gar nicht“ jeweils an, wie sehr die Aussage zutrifft.

	Stimmt völlig	Stimmt eher	Stimmt eher nicht	Stimmt gar nicht	nicht beurteilbar
Das Thema des Workshops fand ich interessant.	○	○	○	○	○
Die Lern- und Arbeitsatmosphäre war angenehm.	○	○	○	○	○
Ich habe in diesem Workshop ein besseres Verständnis erhalten, welche Rolle die Mathematik für KI-Systeme hat.	○	○	○	○	○
Das Projekt hat mein Interesse an Themen der angewandten Mathematik sowie der Naturwissenschaften und Technik gesteigert.	○	○	○	○	○
Ich habe in diesem Workshop viel Neues gelernt, was mir für die Schule, für ein Studium und / oder Beruf weiterhelfen kann.	○	○	○	○	○
Durch den Workshop habe ich interessante Berufs- und Studienmöglichkeiten kennengelernt.	○	○	○	○	○
Ich kann mir vorstellen ein Studium oder eine Ausbildung im Bereich der angewandten Mathematik, der Naturwissenschaften oder der Technik zu beginnen.	○	○	○	○	○
Ich würde so einen Workshop (zu einem anderen Thema) gerne noch einmal besuchen.	○	○	○	○	○
Ich würde diesen Workshop anderen weiterempfehlen.	○	○	○	○	○

2.6. Lob, Kritik und Verbesserungsvorschläge

Hat Dir etwas an dem Workshop absolut nicht gefallen?

○ Nein.

○ Ja und zwar: _____

Hat Dir etwas an diesem Workshop besonders gut gefallen?

○ Nein.

○ Ja und zwar: _____

Hättest Du gerne noch etwas anderes gesehen oder erfahren?

○ Nein.

○ Ja und zwar: _____

3. Lernzuwachs

Was hast Du für Dich persönlich durch die Teilnahme am Workshop gelernt?

Beschreibe in 2 bis 3 Sätzen, was Du unter Mathematischer Modellierung verstehst.

Beschreibe in 2 bis 3 Sätzen, was Du unter Künstlicher Intelligenz verstehst.

Wir haben im Workshop zur Beurteilung unseres Klassifikationsalgorithmus eine Methode aus dem Bereich des überwachten Maschinellen Lernens verwendet. Beschreibe kurz den Ablauf dieser Methode.

4. Abschließende Bewertung

Ich gebe dem CAMMP Workshop die Schulnote

- ◯ 1 (sehr gut)
- ◯ 2 (gut)
- ◯ 3 (befriedigend)
- ◯ 4 (ausreichend)
- ◯ 5 (mangelhaft)
- ◯ 6 (ungenügend)

Ich gebe den Betreuer:innen die Schulnote

- ◯ 1 (sehr gut)
- ◯ 2 (gut)
- ◯ 3 (befriedigend)
- ◯ 4 (ausreichend)
- ◯ 5 (mangelhaft)
- ◯ 6 (ungenügend)

Abschließender persönlicher Kommentar (Lob, Kritik, Verbesserungsvorschläge, ...)

Vielen Dank für Ihre Teilnahme!

Wir möchten uns ganz herzlich für Ihre Mithilfe bedanken.

Ihre Antworten wurden gespeichert, Sie können das Browser-Fenster nun schließen.

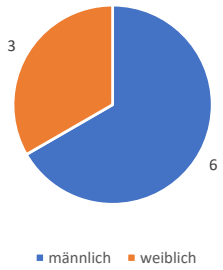
Dr. Sarah Schönbrodt, Karlsruher Institut für Technologie – 2021

G.2. Ergebnisse der Evaluation der ersten Durchführung

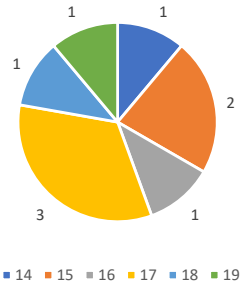
Auswertung der Evaluation
Offener CAMMP day Aktivitätserkennung 09.09.2022

1. Angaben zur eigenen Person und Schule

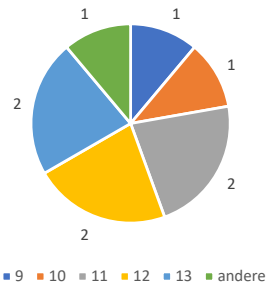
Geschlecht:



Alter in Jahren:



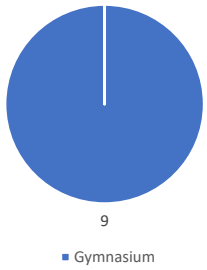
Welche Jahrgangsstufe besuchst Du?



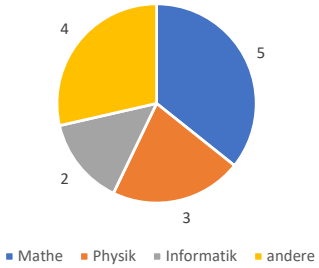
andere:

- Schule abgeschlossen

Welche Schule besuchst Du?



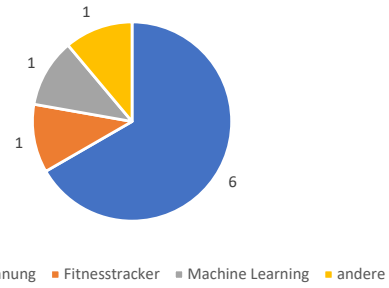
Falls Du bereits in einer Klasse der Kursstufe bzw. Oberstufe bist, welche Leistungskurse besuchst Du?



andere:

- Französisch, English
- Chemie, Wirtschaft
- Wirtschaft
- Englisch

An welchem Workshop von CAMMP hast Du teilgenommen?



andere:

- CAMMP week 2022

2. Bewertung des Workshops

2.1. Gestaltung des Workshops

Im Folgenden findest du einige Aussagen über deine Teilnahme am CAMMP Workshop. Bitte gib auf einer Skala von „Stimmt völlig“ bis „Stimmt gar nicht“ jeweils an, wie sehr die Aussage zutrifft.

	Stimmt völlig	Stimmt eher	Stimmt eher nicht	Stimmt gar nicht	Nicht beurteilbar	Nicht beantwortet
Die Arbeitsblätter waren ansprechend und übersichtlich gestaltet.	5 (55,56 %)	4 (44,44 %)				
Die Aufgabenstellungen waren abwechslungsreich.		7 (77,78 %)	2 (22,22 %)			
Die Lern- und Arbeitszeiten waren angemessen (nicht zu lang oder zu kurz mit ausreichenden Pausen).	6 (66,67 %)		3 (33,33 %)			
Die Diskussionsphasen im Plenum nach jedem Arbeitsblatt waren sinnvoll.	4 (44,44 %)	5 (55,56 %)				
Die Diskussionsphasen im Plenum sollten ausführlicher sein.	1 (11,11 %)		6 (66,67 %)	1 (11,11 %)		1 (11,11 %)
Im Workshop hatte ich das Gefühl, dass ich neue Inhalte selbstständig erarbeiten konnte.	6 (66,67 %)	3 (33,33 %)				
Ich konnte meine eigenen Ideen in den Workshop einbringen (z.B. in Diskussionen).	6 (66,67 %)	1 (11,11 %)	2 (22,22 %)			

2.2. Schwierigkeitsgrad des Workshops

Im Folgenden findest du einige Aussagen über deine Teilnahme am CAMMP Workshop. Bitte gib auf einer Skala von „Stimmt völlig“ bis „Stimmt gar nicht“ jeweils an, wie sehr die Aussage zutrifft.

	Stimmt völlig	Stimmt eher	Stimmt eher nicht	Stimmt gar nicht	Nicht beurteilbar	Nicht beantwortet
Ich konnte die Aufgaben eigenständig bearbeiten.	7 (77,78 %)	2 (22,22 %)				
Die Aufgaben waren zu einfach.		5 (55,56 %)	3 (33,33 %)			1 (11,11 %)
Die Aufgaben waren zu schwierig.			4 (44,44 %)	5 (55,56 %)		
Mein bisheriges mathematisches Vorwissen hat ausgereicht, um die Aufgaben bearbeiten zu können.	4 (44,44 %)	4 (44,44 %)	1 (11,11 %)			

Das fand ich im Workshop bzw. an den Aufgaben (besonders) schwierig:

- Die Aufgaben bei der man programmieren musste da ich keine Vorkenntnisse hatte
- Nichts
- Die Aufgaben bei der ich das Grundwissen noch nicht hatte
- -
- -
- -
- Nichts
- Nichts
- Es war eigentlich nichts sonderlih schwer.

2.3. Verständlichkeit und Hilfestellung

Im Folgenden findest du einige Aussagen über deine Teilnahme am CAMMP Workshop. Bitte gib auf einer Skala von „Stimmt völlig“ bis „Stimmt gar nicht“ jeweils an, wie sehr die Aussage zutrifft.

	Stimmt völlig	Stimmt eher	Stimmt eher nicht	Stimmt gar nicht	Nicht beurteilbar	Nicht beantwortet
Die Inhalte wurden in den Präsentationen gut erklärt.	6 (66,67 %)	3 (33,33 %)				
Die Aufgabenstellungen waren klar und verständlich formuliert.	4 (44,44 %)	4 (44,44 %)	1 (11,11 %)			
Die Grafiken und Abbildungen haben beim Verständnis der Inhalte geholfen.	6 (66,67 %)	3 (33,33 %)				
Die Tipps / Hilfekarten waren hilfreich.	2 (22,22 %)	3 (33,33 %)	1 (11,11 %)			3 (33,33 %)
Bei der Bearbeitung der Aufgaben musste ich häufig die Tipps / Hilfekarten benutzen.	1 (11,11 %)		3 (33,33 %)	5 (55,56 %)		

2.4. Modellierung und KI sowie Umgang mit Jupyter Notebooks und Python

Im Folgenden findest du einige Aussagen über deine Teilnahme am CAMMP Workshop. Bitte gib auf einer Skala von „Stimmt völlig“ bis „Stimmt gar nicht“ jeweils an, wie sehr die Aussage zutrifft.

	Stimmt völlig	Stimmt eher	Stimmt eher nicht	Stimmt gar nicht	Nicht beurteilbar	Nicht beantwortet
Durch den Workshop habe ich mathematisches Modellieren besser begriffen.	3 (33,33 %)	6 (66,67 %)				
Der Vortrag über mathematische Modellierung war hilfreich.	5 (55,56 %)	4 (44,44 %)				
Durch den Workshop habe ich die mathematischen Hintergründe von KI-Systemen besser begriffen.	4 (44,44 %)	4 (44,44 %)				1 (11,11 %)
Die Einführung in Jupyter Notebooks und Python war hilfreich.	2 (22,22 %)	3 (33,33 %)				4 (44,44 %)
Der Umgang mit der Programmiersprache Python fiel mir schwer.		2 (22,22 %)	2 (22,22 %)	5 (55,56 %)		
Ich hätte mir zu Beginn des Workshops eine umfangreichere Einführung in das Arbeiten mit Python gewünscht.		1 (11,11 %)	1 (11,11 %)	7 (77,78 %)		
Die automatischen Rückmeldungen, die ich nach der Eingabe meiner Lösungen im Code erhalten habe, fand ich hilfreich.	7 (77,78 %)	2 (22,22 %)				
Das Arbeiten mit den Codefeldern hat mir Spaß gemacht und den Workshop bereichert.	6 (66,67 %)	2 (22,22 %)	1 (11,11 %)			
Ich habe bereits vor dem Workshop (z.B. in der Schule) mit Jupyter Notebooks und Python gearbeitet.	5 (55,56 %)	1 (11,11 %)	1 (11,11 %)	2 (22,22 %)		

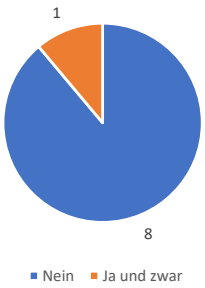
2.5. Motivation

Im Folgenden findest du einige Aussagen über deine Teilnahme am CAMMP Workshop. Bitte gib auf einer Skala von „Stimmt völlig“ bis „Stimmt gar nicht“ jeweils an, wie sehr die Aussage zutrifft.

	Stimmt völlig	Stimmt eher	Stimmt eher nicht	Stimmt gar nicht	Nicht beurteilbar	Nicht beantwortet
Das Thema des Workshops fand ich interessant.	9 (100,00 %)					
Die Lern- und Arbeitsatmosphäre war angenehm.	6 (66,67 %)	3 (33,33 %)				
Ich habe in diesem Workshop ein besseres Verständnis erhalten, welche Rolle die Mathematik für KI-Systeme hat.	6 (66,67 %)	3 (33,33 %)				
Das Projekt hat mein Interesse an Themen der angewandten Mathematik sowie der Naturwissenschaften und Technik gesteigert.	6 (66,67 %)	3 (33,33 %)				
Ich habe in diesem Workshop viel Neues gelernt, was mir für die Schule, für ein Studium und / oder Beruf weiterhelfen kann.	5 (55,56 %)	3 (33,33 %)	1 (11,11 %)			
Durch den Workshop habe ich interessante Berufs- und Studienmöglichkeiten kennengelernt.	2 (22,22 %)	4 (44,44 %)	2 (22,22 %)	1 (11,11 %)		
Ich kann mir vorstellen ein Studium oder eine Ausbildung im Bereich der angewandten Mathematik, der Naturwissenschaften oder der Technik zu beginnen.	5 (55,56 %)	3 (33,33 %)	1 (11,11 %)			
Ich würde so einen Workshop (zu einem anderen Thema) gerne noch einmal besuchen.	6 (66,67 %)	3 (33,33 %)				
Ich würde diesen Workshop anderen weiterempfehlen.	7 (77,78 %)	2 (22,22 %)				

2.6. Lob, Kritik und Verbesserungsvorschläge

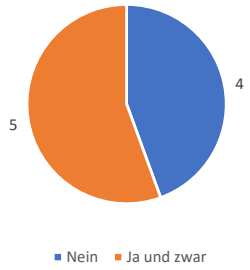
Hat Dir etwas an dem Workshop absolut nicht gefallen?



Ja und zwar:

- Man musst eigentlich nichts selber programmieren.

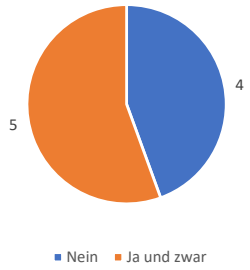
Hat Dir etwas an diesem Workshop besonders gut gefallen?



Ja und zwar:

- Man hatte die Möglichkeit sich alles selber zu erarbeiten aber es wurde trotzdem nochmal erklärt
- Mathematisches Mittel (Quartil) kenne ich jetzt.
- Gutes Erklären auch bei keinen so hohen Grundwissen
- Die Arbeitsblätter waren ideal, um sich die Themen selbstständig zu erarbeiten.
- Die Datenverarbeitung

Hättest du gerne noch etwas anderes gesehen oder erfahren?



Ja und zwar

- Die Aktivitäten hätten spannender sein können, z.B. Fahrrad fahren, schwimmen, klettern, Surfen etc.
- Wie man einen guten Datensatz erstellt.
- unbeaufsichtigtes maschinelles lernen
- Optimierung der Rechenauslastung
- Eine etwas tiefere einarbeitung in das Thema wäre schön gewesen dafür hätte der workshop auch länger dauern können

3. Lernzuwachs

Was hast Du für Dich persönlich durch die Teilnahme am Workshop gelernt?

- Ich habe besser verstanden was eig hinter dem ganzen steckt
- Neues aus der Mathe
- Wie wichtig Programmieren ist
- Einige Zusammenhänge zwischen Mathematik und Informatik
- Besseren Einblick in KI, Datenverarbeitung
- erste einblicke in math. modelle

- KNN
- Ich habe einen tieferen einblick in die Verwendung von Compizerprogrammen zur Mathematischen modellierung bekommen
- Das bestimmen der Zugehörigkeit zu einer Klasse durch den k nächsten Nachbarn.

Beschreibe in 2 bis 3 Sätzen, was Du unter Mathematischer Modellierung verstehst.

- Man stellt etwas als Modell da und versucht dieses Problem im Modell zu lösen und kontrolliert dann ob es in Wirklichkeit so stimmt und justiert dann nach
- Man Probleme vereinfacht, bearbeitet, löst und überprüft.
- Das reduzieren und vereinfachen von Alltagsproblemen, sodass sie auf mathematischer Ebene beachrieben, felöst und optimiert werden können.
- Unter Mathrmatischer Modellierung verstehe ich ein alltägliches Phanomen soweit wie möglich zu vereinfachen, es jefoch dennoch richtig beschrieben können
- Aus einer Menge an Daten eine Aussage treffen.
- Mit der Mathematischen Modellierung wird ein reelles Problem so vereinfacht, dass sich daraus ein mathematisches Problem ergibt, welches anschließend gelöst wird. Diese Lösung wird für die Anwendung beim echten Problem interpretiert
- Das abbilden und lösen eines problems mit Mathematik
- Man nimmt ein reales Problem, vereinfacht es auf die wichtigsten Aspekte und vormoliert dieses mittels Mathematik. Anschließend prüft man wie gut es funktiniert und märzt solange Fehler aus bis man zufrieden ist.f

Beschreibe in 2 bis 3 Sätzen, was Du unter Künstlicher Intelligenz verstehst.

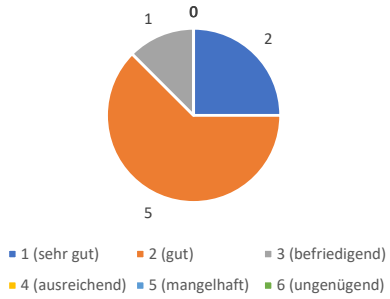
- Ein Programm was man nicht „fertig programmieren“ muss sondern was sich eigenständig weiterentwickelt und dazulernt
- Ein Programm, welches Mithilfe von Algorithmen auf eine Antwort kommt.
- Ein Computerprogramm, dass durch den Erhalt von Daten und Erfahrungen sich stetig verbessert.
- Ein „schlauer“ Computer, der sich mithilfe von daten alleine verbessert und lernt
- Selbstlernende Computerprogramme denen man keine regeln vorgibt.
- Ein Programm, dass nicht mit festen Regeln funktioniert
- Ein selbstlernendes Programm
- “

Wir haben im Workshop zur Beurteilung unseres Klassifikationsalgorithmus eine Methode aus dem Bereich des überwachten Maschinellen Lernens verwendet. Beschreibe kurz den Ablauf dieser Methode.

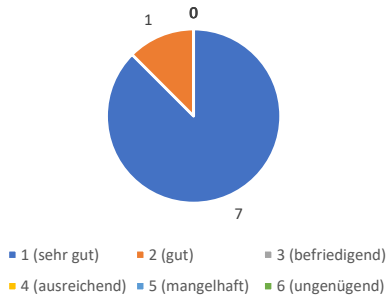
- Werte werden in einen Array eingespeist, dann sortiert und schließlich mit k usw. ausgewertet.
- Daten von Handysensoren mit sogenannten Features auf mathematischer Ebene beachrieben und diese mittels dem Computer verscheidenen Aktivitäten zugeordnet.
- Mithilfe von daten verbessert sich der computer von selbst und lernt selbst dazu
- Richtigigkeit bewerten durch anteil richtiger/falscher zuordnungen
- Ein Datensatz wird in zwei Gruppen geteilt, die Testdaten und die Trainibgsdaten. Die Testdaten werden anhand der Trainingsdaten klassifiziert und anhand dieser Klassifizierung wird überprüft wie qualitativ das Programm ist
- Man überprüft an einen richtigen Datensatz ob die neuen Daten verwendbar aind
- Man teilt einen Datensatz in 2 Teile einien mit dem man sien Methode Treniert und einen mit dem man sie überprüft.

4. Abschließende Bewertung

Ich gebe dem CAMMP Workshop die Schulnote



Ich gebe den Betreuer:innen die Schulnote



Abschließender persönlicher Kommentar (Lob, Kritik, Verbesserungsvorschläge, …)

- Es ist super, dass ihr euch so engagiert.
- Es war super.
- -
- spannend und lehrreich, aufgaben gerne etwas offener

H. Jupyter Notebooks

H.1. Weitere Klassifizierungen

Weitere Klassifizierungen der Sensordaten

In [1]:
```
# Daten und Packages laden
import sys; sys.path.append("../code"); from setupAB7 import *;
```

---

1. Handy befindet sich in der Hand des Nutzers

Zuerst werden die Sensordaten erneut klassifiziert, bei denen sich das Handy in der Hand des Nutzers befand. Für diese Daten werden die Features des Feature-Set 2 berechnet.

1.1. Aufteilung der Sensordaten in Test- und Trainingsdaten

Diesmal werden die Sensordaten, bei denen sich das Handy in der Hand des Nutzers befand, nicht komplett als Testdaten genutzt. Stattdessen wird die Hälfte der Daten dem Trainingsdatensatz zugeordnet. Somit wird der Trainingsdatensatz diversifiziert.

Im Datensatz mit den Daten, bei denen sich das Handy in der Hand des Nutzers befand, existieren zu jeder Aktivität 60 Windows. Die Hälfte der Windows der jeweiligen Aktivitäten wird den Trainingsdaten und die andere Hälfte den Testdaten zugeordnet. Zusätzlich besteht der Trainingsdatensatz aus dem Feature-Set 2, das die Windows zu den Sensordaten enthält, bei denen sich das Handy in der rechten vorderen Hosentasche befand.

In [2]:
```
# Aufteilung der Windows zu den einzelnen Aktivitäten in Trainings- und Test-Windows

## Aktivität Sitzen
Set3Train_Sitzen = Set3[Set3['WindowID'] < 3031]
Set3Test_Sitzen = Set3[(Set3['WindowID'] > 3030) & (Set3['WindowID'] < 3061)]

## Aktivität Stehen
Set3Train_Stehen = Set3[(Set3['WindowID'] > 3060) & (Set3['WindowID'] < 3091)]
Set3Test_Stehen = Set3[(Set3['WindowID'] > 3090) & (Set3['WindowID'] < 3121)]

## Aktivität Gehen
Set3Train_Gehen = Set3[(Set3['WindowID'] > 3120) & (Set3['WindowID'] < 3151)]
Set3Test_Gehen = Set3[(Set3['WindowID'] > 3150) & (Set3['WindowID'] < 3181)]

## Aktivität Laufen
Set3Train_Laufen = Set3[(Set3['WindowID'] > 3180) & (Set3['WindowID'] < 3211)]
Set3Test_Laufen = Set3[(Set3['WindowID'] > 3210) & (Set3['WindowID'] < 3241)]

## Aktivität Treppen auf- und absteigen
Set3Train_Treppen = Set3[(Set3['WindowID'] > 3240) & (Set3['WindowID'] < 3271)]
Set3Test_Treppen = Set3[Set3['WindowID'] > 3270]


# Zusammenfügen der einzelnen Datensätze zu Trainings- und Testdatensatz

## Trainingsdatensatz
TrainHand = pd.concat([Set2, Set3Train_Sitzen, Set3Train_Stehen, Set3Train_Gehen, Set3Train_Laufen, Set3Train_T

## Testdatensatz
TestHand = pd.concat([Set3Test_Sitzen, Set3Test_Stehen, Set3Test_Gehen, Set3Test_Laufen, Set3Test_Treppen], axi
```

1.2. Klassifizierung

Die Testdaten, bei denen sich das Handy in der Hand des Nutzers befand, werden nun klassifiziert. Für die Anzahl der nächsten Nachbarn wird $k = 1$ gewählt.

In [3]:
```
# Trainingsdaten: Feature- and Class-Array
XHand_train = TrainHand.iloc[:,3:17].values
yHand_train = TrainHand.iloc[:,2].values

# Testdaten: Feature- and Class-Array
XHand_test = TestHand.iloc[:,3:17].values
yHand_test = TestHand.iloc[:,2].values

# Klassifizierung mit k = 1
```

```
knnHand = KNeighborsClassifier(n_neighbors = 1)
knnHand.fit(XHand_train, yHand_train)
yHand_pred = knnHand.predict(XHand_test)

# Wahrheitsmatrix
CMHand = confusion_matrix(yHand_test, yHand_pred)

# Genauigkeit
AccuracyHand = accuracy_score(yHand_test, yHand_pred)

# Fehlerrate
ErrorRateHand = 1 - AccuracyHand

# Präzision
Pre1Hand = CMHand[0][0] / (CMHand[0][0] + CMHand[1][0] + CMHand[2][0] + CMHand[3][0] + CMHand[4][0])
Pre2Hand = CMHand[1][1] / (CMHand[0][1] + CMHand[1][1] + CMHand[2][1] + CMHand[3][1] + CMHand[4][1])
Pre3Hand = CMHand[2][2] / (CMHand[0][2] + CMHand[1][2] + CMHand[2][2] + CMHand[3][2] + CMHand[4][2])
Pre4Hand = CMHand[3][3] / (CMHand[0][3] + CMHand[1][3] + CMHand[2][3] + CMHand[3][3] + CMHand[4][3])
Pre5Hand = CMHand[4][4] / (CMHand[0][4] + CMHand[1][4] + CMHand[2][4] + CMHand[3][4] + CMHand[4][4])

# Ausgabe
print(' ----- \n Wahrheitsmatrix: \n ----- \n ', CMHand, '\n \n ----- \n Genauigkeit: \n ----- \n accuracy =',
```

```
-----
Wahrheitsmatrix:
-----
 [[13 17  0  0  0]
[12 18  0  0  0]
[ 0  0 25  0  5]
[ 0  0  0 30  0]
[ 0  0 14  0 16]]

-----
Genauigkeit:
-----
accuracy = 0.68

-----
Fehlerrate:
-----
error rate = 0.31999999999999995

-----
Präzision:
-----
precisionSitzen = 0.52
precisionStehen = 0.5142857142857142
precisionGehen = 0.6410256410256411
precisionLaufen = 1.0
precisionTreppen = 0.7619047619047619
```

---

2. Handy befindet sich im Rucksack des Nutzers

Anschließend werden die Sensordaten erneut klassifiziert, bei denen sich das Handy im Rucksack des Nutzers befand. Auch für diese Daten werden die Features des Feature-Set 2 berechnet.

2.1. Aufteilung der Sensordaten in Test- und Trainingsdaten

Anaolg zu Teil 1 wird wieder die Hälfte der Windows der jeweiligen Aktivitäten den Trainingsdaten und die andere Hälfte den Testdaten zugeordnet. Zusätzlich besteht der Trainingsdatensatz aus dem Feature-Set 2.

In [4]:
```
# Aufteilung der Windows zu den einzelnen Aktivitäten in Trainings- und Test-Windows

## Aktivität Sitzen
Set4Train_Sitzen = Set4[Set4['WindowID'] < 3331]
Set4Test_Sitzen = Set4[(Set4['WindowID'] > 3330) & (Set4['WindowID'] < 3361)]

## Aktivität Stehen
Set4Train_Stehen = Set4[(Set4['WindowID'] > 3360) & (Set4['WindowID'] < 3391)]
Set4Test_Stehen = Set4[(Set4['WindowID'] > 3390) & (Set4['WindowID'] < 3421)]

## Aktivität Gehen
Set4Train_Gehen = Set4[(Set4['WindowID'] > 3420) & (Set4['WindowID'] < 3451)]
Set4Test_Gehen = Set4[(Set4['WindowID'] > 3450) & (Set4['WindowID'] < 3481)]
```

```
## Aktivität Laufen
Set4Train_Laufen = Set4[(Set4['WindowID'] > 3480) & (Set4['WindowID'] < 3511)]
Set4Test_Laufen = Set4[(Set4['WindowID'] > 3510) & (Set4['WindowID'] < 3541)]

## Aktivität Treppen auf- und absteigen
Set4Train_Treppen = Set4[(Set4['WindowID'] > 3540) & (Set4['WindowID'] < 3571)]
Set4Test_Treppen = Set4[Set4['WindowID'] > 3570]

# Zusammenfügen der einzelnen Datensätze zu Trainings- und Testdatensatz

## Trainingsdatensatz
TrainBackpack = pd.concat([Set2, Set4Train_Sitzen, Set4Train_Stehen, Set4Train_Gehen, Set4Train_Laufen, Set4Tra

## Testdatensatz
TestBackpack = pd.concat([Set4Test_Sitzen, Set4Test_Stehen, Set4Test_Gehen, Set4Test_Laufen, Set4Test_Treppen],
```

2.2. Klassifizierung

Die Testdaten, bei denen sich das Handy im Rucksack des Nutzers befand, werden nun klassifiziert. Für die Anzahl der nächsten Nachbarn wird $k = 1$ gewählt.

In [5]:
```
# Trainingsdaten: Feature- and Class-Array
XBackpack_train = TrainBackpack.iloc[:,3:17].values
yBackpack_train = TrainBackpack.iloc[:,2].values

# Testdaten: Feature- and Class-Array
XBackpack_test = TestBackpack.iloc[:,3:17].values
yBackpack_test = TestBackpack.iloc[:,2].values

# Klassifizierung mit k = 1
knnBackpack = KNeighborsClassifier(n_neighbors = 1)
knnBackpack.fit(XBackpack_train, yBackpack_train)
yBackpack_pred = knnBackpack.predict(XBackpack_test)

# Wahrheitsmatrix
CMBackpack = confusion_matrix(yBackpack_test, yBackpack_pred)

# Genauigkeit
AccuracyBackpack = accuracy_score(yHand_test, yHand_pred)

# Fehlerrate
ErrorRateBackpack = 1 - AccuracyBackpack

# Präzision
Pre1Backpack = CMBackpack[0][0] / (CMBackpack[0][0] + CMBackpack[1][0] + CMBackpack[2][0] + CMBackpack[3][0] +
Pre2Backpack = CMBackpack[1][1] / (CMBackpack[0][1] + CMBackpack[1][1] + CMBackpack[2][1] + CMBackpack[3][1] +
Pre3Backpack = CMBackpack[2][2] / (CMBackpack[0][2] + CMBackpack[1][2] + CMBackpack[2][2] + CMBackpack[3][2] +
Pre4Backpack = CMBackpack[3][3] / (CMBackpack[0][3] + CMBackpack[1][3] + CMBackpack[2][3] + CMBackpack[3][3] +
Pre5Backpack = CMBackpack[4][4] / (CMBackpack[0][4] + CMBackpack[1][4] + CMBackpack[2][4] + CMBackpack[3][4] +

# Ausgabe
print(' ----- \n Wahrheitsmatrix: \n ----- \n ', CMBackpack, '\n \n ----- \n Genauigkeit: \n ----- \n accuracy
```

```
-----
Wahrheitsmatrix:
-----
 [[30  0  0  0  0]
[ 0 30  0  0  0]
[ 0  0 30  0  0]
[ 0  0  0 30  0]
[ 0  0  2  0 28]]

-----
Genauigkeit:
-----
accuracy = 0.68

-----
Fehlerrate:
-----
error rate = 0.31999999999999995

-----
Präzision:
-----
precisionSitzen = 1.0
precisionStehen = 1.0
precisionGehen = 0.9375
precisionLaufen = 1.0
precisionTreppen = 1.0
```

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur

Aebli, H. (2006). *Zwölf Grundformen des Lehrens: eine allgemeine Didaktik auf psychologischer Grundlage. Medien und Inhalte didaktischer Kommunikation, der Lernzyklus.* Stuttgart: Klett-Cotta.

Banos, O., Galvez, J.-M., Damas, M., Pomares, H. & Rojas, I. (2014). Window Size Impact in Human Activity Recognition. *Sensors*, *14*, 6474-6499.

CAMMP. (2022a). *Angebote für Lehrkräfte.* `https://www.cammp.online/202.php`. (letzter Aufruf: 17.09.2022)

CAMMP. (2022b). *Angebote für Schüler:innen.* `https://www.cammp.online/198.php`. (letzter Aufruf: 17.09.2022)

CAMMP. (2022c). *Angebote für Studierende.* `https://www.cammp.online/200.php`. (letzter Aufruf: 17.09.2022)

CAMMP. (2022d). *Über CAMMP.* `https://www.cammp.online/85.php`. (letzter Aufruf: 17.09.2022)

CAMMP. (2022e). *CAMMP week - Modellierungswoche.* `https://www.cammp.online/21.php`. (letzter Aufruf: 17.09.2022)

CAMMP. (2022f). *(Online) CAMMP day - mathematischer Modellierungstag.* `https://www.cammp.online/116.php`. (letzter Aufruf: 17.09.2022)

CAMMP. (2022g). *Unsere Angebote.* `https://www.cammp.online/106.php`. (letzter Aufruf: 17.09.2022)

CAMMP. (2022h). *Willkommen bei CAMMP!* `https://www.cammp.online/index.php`. (letzter Aufruf: 17.09.2022)

Chomboon, K., Chujai, P., Teerarassamee, P., Kerdprasop, K. & Kerdprasop, N. (2015). An Empirical Study of Distance Metrics for k-Nearest Neighbor Algorithm. *Proceedings of the 3rd International Conference on Industrial Application Engineering 2015*, 280-285.

Dittrich, F. (2014). Überblick menschlicher Aktivitätserkennung auf mobilen Geräten. Möglichkeiten, Grenzen und Herausforderungen. In M. A. Neumann, A. Bachmann, Y. Ding & T. Riedel (Hrsg.), *Ubiquitäre Systeme (Seminar) und Mobile Computing (Proseminar) SS 2014. Mobile und Verteilte Systeme Ubiquitous Computing Teil XI.* Karlsruhe: Karlsruher Institut für Technologie (KIT), Fakultät für Informatik, Lehrstuhl für Pervasive Computing Systems (PCS) und TECO.

Frank, M., Richter, P., Roeckerath, C. & Schönbrodt, S. (2018). Wie funktioniert eigentlich GPS? – ein computergestützter Modellierungsworkshop. In G. Greefrath

& H.-S. Siller (Hrsg.), *Digitale Werkzeuge, Simulationen und mathematisches Modellieren - Didaktische Hintergründe und Erfahrungen aus der Praxis* (S. 137-164). Wiesbaden: Springer Spektrum.

Frank, M., Roeckerath, C. & Schönbrodt, S. (2022). Einführung. In M. Frank & C. Roeckerath (Hrsg.), *Neue Materialien für einen realitätsbezogenen Mathematikunterricht 9. Realitätsbezüge im Mathematikunterricht* (S. 1-6). Berlin, Heidelberg: Springer Spektrum.

Frochte, J. (2021). *Maschinelles Lernen. Grundlagen und Algorithmen in Python.* München: Carl Hanser Verlag.

Greefrath, G., Kaiser, G., Blum, W. & Borromeo Ferri, R. (2013). Mathematisches Modellieren. Eine Einführung in theoretische und didaktische Hintergründe. In W. Blum, R. Borromeo Ferri, G. Greefrath & G. Kaiser (Hrsg.), *Mathematisches Modellieren für Schule und Hochschule. Theoretische und didaktische Hintergründe* (S. 11-37). Wiesbaden: Springer Spektrum.

Greefrath, G. & Siller, H.-S. (2018). Digitale Werkzeuge, Simulationen und mathematisches Modellieren. In G. Greefrath & H.-S. Siller (Hrsg.), *Digitale Werkzeuge, Simulationen und mathematisches Modellieren. Didaktische Hintergründe und Erfahrungen aus der Praxis* (S. 3-22). Wiesbaden: Springer Spektrum.

Géron, A. (2020). *Praxiseinstieg Machine Learning mit Scikit-Learn, Keras und TensorFlow: Konzepte, Tools und Techniken für intelligente Systeme.* Heidelberg: O'Reilly.

Henze, N. (2018). *Stochastik für Einsteiger. Eine Einführung in die faszinierende Welt des Zufalls.* Wiesbaden: Springer Spektrum.

Incel, O. D., Kose, M. & Ersoy, C. (2013). A Review and Taxonomy of Activity Recognition on Mobile Phones. *BioNanoSci*, *3*, 145-171.

Kaiser, G., Blum, W., Borromeo Ferri, R. & Greefrath, G. (2015). Anwendungen und Modellieren. In R. Bruder, L. Hefendehl-Hebeker, B. Schmidt-Thieme & H.-G. Weigand (Hrsg.), *Handbuch der Mathematikdidaktik* (S. 357-384). Heidelberg: Springer Spektrum.

Krause, M. & Natterer, E. (2019). Maschinelles Lernen. Wie sich Computer an Probleme anpassen. In K. Kersting, C. Lampert & C. Rothkopf (Hrsg.), *Wie Maschinen lernen. Künstliche Intelligenz verständlich erklärt* (S. 21-27). Wiesbaden: Springer.

Kultusministerkonferenz. (2012). *Bildungsstandards im Fach Mathematik für die Allgemeine Hochschulreife.* `https://www.kmk.org/fileadmin/Dateien/veroeffentlichungen_beschluesse/2012/2012_10_18-Bildungsstandards-Mathe-Abi.pdf`. (letzter Aufruf: 18.09.2022)

Kwapisz, J. R., Weiss, G. M. & Moore, S. A. (2010). Activity Recognition using Cell Phone Accelerometers. *SIGKDD Explorations*, *12*, 74-82.

Lockhart, J. W., Pulickal, T. & Weiss, G. M. (2012). Applications of Mobile Activity Recognition. *UbiComp' 12*.

Marnitz, M. (2017). *Wie funktionieren eigentlich Fitnesstracker und was hat das mit Mathe zu tun? Ein Lernmodul im Rahmen eines mathematischen Modellierungstages.* (Masterarbeit, RWTH Aachen)

Ministerium für Kultus, Jugend und Sport Baden-Württemberg. (2016). *Bildungsplan des Gymnasiums: Mathematik.* `http://www.bildungsplaene-bw.de/site/bildungsplan/get/documents/lsbw/export-pdf/depot-pdf/ALLG/BP2016BW_ALLG_GYM_M.pdf`. Stuttgart. (letzter Aufruf: 18.09.2022)

Ministerium für Kultus und Jugend und Sport Baden-Württemberg. (2016). *Bildungsplan des Gymnasiums: Physik.* `http://www.bildungsplaene-bw.de/site/bildungsplan/get/documents/lsbw/export-pdf/depot-pdf/ALLG/BP2016BW_ALLG_GYM_PH.pdf`. Stuttgart. (letzter Aufruf: 18.09.2022)

Mitchell, T. M. (1997). *Machine Learning.* New York: McGraw-Hill. (`http://www.cs.cmu.edu/~tom/mlbook.html`. letzter Aufruf: 13.10.2022)

Mohsen, S., Elkaseer, A. & Scholz, S. G. (2022). Human Activity Recognition Using K-Nearest Neighbor Machine Learning Algorithm. In S. G. Scholz, R. J. Howlett & R. Setchi (Hrsg.), *Sustainable Design and Manufacturing. KES-SDM 2021. Smart Innovation, Systems and Technologies* (Bd. 262, S. 304-313). Springer Singapore.

Niss, M. & Blum, W. (2020). *The Learning and Teaching of Mathematical Modelling.* London, New York: Routledge.

Paaß, G. & Hecker, D. (2020). *Künstliche Intelligenz. Was steckt hinter der Technologie der Zukunft?* Wiesbaden: SpringerVieweg.

Planing, P. (2022). *Satistik Grundlagen.* `https://statistikgrundlagen.de/ebook/`. (letzter Aufruf: 16.09.2022)

Schönbrodt, S. (2022). *Optimierungsprobleme in der mathematischen Modellierung. Grundlegende Aspekte und Chancen aus Sicht der Mathematikdidaktik - herausgestellt an aktuellen Problemen aus der Forschung zu künstlicher Intelligenz und erneuerbaren Energien.* (Dissertation, Karlsruher Institut für Technologie (KIT))

Schönbrodt, S., Wohak, K. & Frank, M. (2022). Digital Tools to Enable Collaborative Mathematical Modeling Online. *Modelling in Science Education and Learning*, *15* (1), 151-174.

Sill, H.-D. (2019). *Grundkurs Mathematikdidaktik.* Paderborn: Ferdinand Schöningh.

Sorgalla, M. (2015). *Heterogene Lerngruppen. Der DIE-Wissensbaustein für die Praxis.* https://www.die-bonn.de/wb/2015-heterogenitaet-01.pdf. (letzter Aufruf: 20.09.2022)

Stender, P. (2016). *Wirkungsvolle Lehererinterventionsformen bei komplexen Modellierungsaufgaben.* Wiesbaden: Springer Spektrum.

Su, X., Tong, H. & Ji, P. (2014). Activity Recognition with Smartphone Sensors. *Tsinghua Science and Technology*, *19* (3), 235-249.

Ustev, Y. E., Incel, O. D. & Ersoy, C. (2013). User, Device and Orientation Independent Human Activity Recognition on Mobile Phones: Challenges and a Proposal. *UbiComp'13*, 1427-1435.

Wikipedia. (2021). *Manhattan-Metrik.* https://de.wikipedia.org/wiki/Manhattan-Metrik. (letzter Aufruf: 22.09.2022)

Wikipedia. (2022a). *Global System for Mobile Communications.* https://de.wikipedia.org/wiki/Global_System_for_Mobile_Communications. (letzter Aufruf: 13.10.2022)

Wikipedia. (2022b). *Project Jupyter.* https://de.wikipedia.org/wiki/Project_Jupyter. (letzter Aufruf: 22.09.2022)

Wikipedia. (2022c). *Python (Programmiersprache).* https://de.wikipedia.org/wiki/Python_(Programmiersprache). (letzter Aufruf: 22.09.20202)

Winter, H. (1995). Mathematikunterricht und Allgemeinbildung. *Mitteilungen der Gesellschaft für Didaktik der Mathematik*, *21* (61), 37-46. https://ojs.didaktik-der-mathematik.de/index.php?journal=mgdm&page=article&op=view&path%5B%5D=69&path%5B%5D=80. (letzter Aufruf: 20.09.2022)

Wohak, K., Sube, M., Schönbrodt, S., Roeckerath, C. & Frank, M. (2021). Authentische und relevante Modellierung mit Schülerinnen und Schülern an nur einem Tag?! In M. Bracke, M. Ludwig & K. Vorhölter (Hrsg.), *Neue Materialien für einen realitätsbezogenen Mathematikunterricht 8. Realitätsbezüge im Mathematikunterricht* (S. 37-50). Wiesbaden: Springer Spektrum.

Zech, F. (1998). *Grundkurs Mathematikdidaktik. Theoretische und praktische Anleitungen für das Lehren und Lernen von Mathematik.* Weinheim, Basel: Beltz Verlag.

Erklärung

Ich versichere wahrheitsgemäß, die Arbeit selbständig verfasst, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus der Arbeit anderer unverändert oder mit Abänderungen entnommen wurde, sowie die Satzung des KIT zur Sicherung guter wissenschaftlicher Praxis in der jeweils gültigen Fassung beachtet zu haben.

Karlsruhe, den 24. Oktober 2022