

Klassifizierung von Bewegungsdaten eines Beschleunigungssensors über maschinelles Lernen

Bachelorarbeit

zur Erlangung des Grades eines Bachelor of Science im Studiengang Medizintechnik
an der Hochschule Bremerhaven zusammen mit

Fraunhofer-Institut MEVIS in Bremen

von
Tchoutou Kongueb Jovanile (33050)

26 November 2018

Erstgutachter: Prof. Dr. Richard Rascher-Friesenhausen

Zweitgutachter: Dr. Jochen Hirsch

Eigenständigkeitserklärung

Ich versichere, die von mir vorgelegte Arbeit selbständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner an anderer Prüfungsbehörde vorgelegen

Bremerhaven, den 26 November 2018

Danksagungen

Ich danke Gott für seine Liebe und seine Gnade, die er mir jeden Tag schenkt. Ich danke meinen Eltern, die an mich geglaubt haben und haben es mir ermöglicht nach Deutschland zu kommen und zu studieren.

Mein Dank geht ganz besonders an das Fraunhofer-Institut MEVIS in Bremen und deren Mitarbeiter, die mir stets hilfreich zur Seite stand.

Weiter danke ich meinen Brüdern und Schwestern, meinem Freund Steeve Nkemegni für die finanziell und moralische Unterstützung.

Speziell möchte ich hier Prof. Dr. Richard Rascher-Friesenhausen, für seine konstruktive, kollegiale Unterstützung und für die Begutachtung dieser Arbeit bedanken. Ich danke Dr. Jochen Hirsch für die Betreuung dieser Arbeit

Zusammenfassung

Die automatische Identifizierung menschlicher Bewegungen hat in den letzten Jahren immer mehr Aufmerksamkeit auf sich gezogen. Jedoch bleibt die Erkennung menschlicher Aktivitäten ein schwieriges und aktives Forschungsgebiet. Mit der Weiterentwicklung der Sensortechnologie kann die automatische Aktivitätserkennung auf unauffällige und nicht aufdringliche Weise erfolgen. In dieser Arbeit werden Methoden zur Klassifizierung menschlicher Bewegungen beschrieben. Es wurde neben alltäglichen Bewegungen wie Gehen, Laufen, Treppen hinauf- und heruntersteigen auch Rehabilitation Übungen wie Halbe Kniebeugung, Beinpendeln und Kniebeugen-strecken klassifiziert. Die Bewegungserkennung erfolgt durch maschinelles Lernen mit Klassifizierungsalgorithmen.

Zunächst wurden die benötigten Sensordaten eines Beschleunigungssensors AX3 von der Firma Axivity (Großbritannien) von zwei Testpersonen gesammelt und gespeichert. Der Sensor wurde an der Außenseite des Knies getragen. Aus den gesammelten Daten wurden vier verschiedene Merkmale extrahiert. Die Merkmalsdaten wurden als Trainingsdaten und Testdaten für die Klassifizierungsalgorithmen benötigt. Die Leistungen zweier Klassifizierungsalgorithmen, nämlich „K-Nearest Neighbor“ und „Decision Tree“ wurden verglichen. Die beste Genauigkeit von 96 % wurde mit der K-Nearest Neighbor erzielt. Bezüglich der Aktivitäten liefert die Rehabilitation Aktivitäten bei beiden Klassifizierungsalgorithmen die besten Ergebnisse von 99%.

Inhaltverzeichnis

Inhaltverzeichnis	i
Tabellenverzeichnis.....	iii
Abbildungsverzeichnis	iv
Formelverzeichnis	vii
Abkürzungsverzeichnis	viii
1 Einleitung.....	1
1.1 Allgemeine Vorgehensweise für menschliche Aktivitätserkennung.....	1
1.2 Vorherige Untersuchungen.....	1
1.3 Ziele und Aufbau der Arbeit	4
2 Grundlagen.....	5
2.1 Beschleunigungssensor	5
2.2 Informatik Grundlagen	6
2.2.1 Python	6
2.2.2 Jupyter Notebook	7
2.2.3 Bokeh	8
3 Datenakquisition	9
3.1 Axivity AX3	9
3.2 Aufnahme	10
3.3 Aktivitäten	13
4 Datenverarbeitung.....	16
4.1 Konvertierung von CWA- Dateien in CSV-Dateien	16
4.2 Daten Visualisierung	16
4.3 Vorverarbeitung der Datensätze	18
4.3.1 Teilen von Daten in Aktivitätsbereiche.....	19
4.3.2 Zusammenfügen von Daten	21

4.4	Fensterung	23
5	Merkmale	24
6	Klassifizierung	27
6.1	K-Nearest Neighbor Algorithmus	27
6.2	Decision Tree	29
6.3	Implementierung	30
7	Ergebnisse und Bewertung der Klassifizierung.....	33
7.1	Ergebnisse	33
7.2	Bewertung der Ergebnisse	34
7.2.1	K-Nearest Neighbor	34
7.2.2	Decision Tree	35
7.2.3	Bewertung der Ergebnisse in Bezug auf die Aktivitätsgruppen	36
8	Fazit und Ausblick	38
	Literaturverzeichnis.....	40
A.	Benutzerhandbuch.....	43
A.1	Show Data	43
A.2	Show Cutpoints and Split File 1	44
A.2	Show Cutpoints and Split File 2	45
A.4	Show Cutpoints and Split File 3	46
A.5	Show Box and Math operation	47
B.	Datei Teilen	49
B.1	Bessere Schnittpunkte Algorithmus der ersten Variante	49
B.2	Bessere Schnittpunkte Algorithmus der dritte Variante	51
C.	Tabellen und Bildern	55

Tabellenverzeichnis

Tabelle 1-1: Übersicht über weitere Untersuchungen.....	3
Tabelle 3-1: Empfindlichkeitsstufen dem Axivity AX3	10
Tabelle 3-2: Anzahl Datensätze Pro Probanden und Aktivitäten	15
Tabelle 7-1: Erkennungsrate von beiden Algorithmen geordnet nach Aufteilungsmodell und K-Werte bei 3s Fensterung.....	33
Tabelle C-1: Erkennungsrate von beide Algorithmen geordnet nach Aufteilungsmodell und K-Werte bei 1s Fensterung.....	55
Tabelle C-2: Erkennungsrate von beide Algorithmen geordnet nach Aufteilungsmodell und K-Werte bei 6s Fensterung.....	55

Abbildungsverzeichnis

Abbildung 1-1: Allgemeine Vorgehensweise zur Aktivitätserkennung	1
Abbildung 2-1: Beschleunigungssensor	6
Abbildung 3-1: Axivity -AX 3	10
Abbildung 3-2: Axivity-AX3 Montage.....	11
Abbildung 3-3: Beispiel von "recordSetup"	12
Abbildung 3-4: Visualisierung von den Aktivitäten halbe Kniebeugung und Kniebeugen strecken mit der OM GUI.....	12
Abbildung 3-5: Halbe Kniebeugung	13
Abbildung 3-6: Beinpendeln	14
Abbildung 3-7: Kniebeugen strecken	14
Abbildung 4-1: Beschleunigungssignal für halbe Kniebeugung	17
Abbildung 4-2: Beschleunigungssignal für Beinpendeln	17
Abbildung 4-3: Beschleunigungssignal für Kniebeugen-strecken.....	17
Abbildung 4-4: Beschleunigungssignal für Gehen	17
Abbildung 4-5: Beschleunigungssignal für Laufen	18
Abbildung 4-6: Beschleunigungssignal für Treppen heruntersteigen.....	18
Abbildung 4-7: Beschleunigungssignal für Treppen hinaufsteigen.....	18
Abbildung 4-8: Auswahlmenü von Tab“ Show Cutpoints and Split File 1“	19
Abbildung 4-9: Auswahlmenü von Tab“ Show Cutpoints and Split File 3“	20
Abbildung 4-10: Python Programm für das Zusammenfügen von Dateien.....	22
Abbildung 4-11: Auswahl Menü für das Tabs „merge CSV_File and save“	22
Abbildung 4-12: Beispiel Daten Fensterung mit 50% Überlappung	23
Abbildung 5-1: Auswahl Menü Tab „Feature calcul and save“	26
Abbildung 6-1: Beispiel Klassifikation mit K-NN, wenn $k = 6$	28
Abbildung 6-2: Funktion für Klassifizierung mit dem K-NN Algorithmus	31

Abbildungsverzeichnis

Abbildung 6-3: Funktion für Klassifizierung mit dem Dtree Algorithmus	31
Abbildung 6-4: Auswahl Menü Tab „classification“	32
Abbildung 7-1: Konfusionsmatrix bei K-NN mit $K = 3$, Modell „70 / 30“ und 3s Fensterung	35
Abbildung 7-2: Konfusionsmatrix bei Dtree mit dem Modell „70 / 30“ und 3s Fensterung	36
Abbildung A-1: Tab Show Data - Auswahlmenü	43
Abbildung A-2: Tab Show Cutpoints and Split File 1 - Auswahlmenü	45
Abbildung A-3: Tab Show Cutpoints and Split File 2 – Auswahlmenü	46
Abbildung A-4: Tab Show Cutpoints and Split File 3 – Auswahlmenu	47
Abbildung A-5: Tab Show Box and Math operation - Auswahlmenü	48
Abbildung B-1: Algorithmus für das Methode1	49
Abbildung B-2: Der erste Schritt des Algorithmus	50
Abbildung B-3: Der zweite Schritt des Algorithmus	50
Abbildung B-4: Der dritte Schritt des Algorithmus	50
Abbildung B-5: Entfernung der Schnittpunkte 2 und 4	51
Abbildung B-6: Algorithmus für die Methode 3	52
Abbildung B-7: Der erste Schritt des Algorithmus	53
Abbildung B-8: Schritt 2	53
Abbildung B-9: neue Datei 1	53
Abbildung B-10: Letzter Schritte	54
Abbildung B-11: Bessere Schnittpunkte	54
Abbildung C-1: Beispiel eine ausgefügte Aktivität Arbeitsblatt	56

Formelverzeichnis

Formel 1: Berechnung von Vektor Länge	20
Formel 2: Berechnung Mittelwert.....	24
Formel 3: Berechnung der Standardabweichung	24
Formel 4: Berechnung der Energie in Frequenzbereich	25
Formel 5: Korrelation zwischen zwei Punkte	25
Formel 6: Manhattan Formel	29
Formel 7: Berechnung der „Gini - Index“	30

Abkürzungsverzeichnis

Abkürzung	Erklärung
CWA	Proprietäres Dateiformat des Axivity AX3
CSV	Comma-separated Values (Dateiformat)
Dtree	Decision Tree
FFT	Fast Fourier Transformation
K-NN	K-Nearest Neighbor
OM-GUI	Open Movement Graphical User Interface
Reha	Rehabilitation

1 Einleitung

Jeden Tag wird in verschiedenen Bereichen geforscht, wie die Lebensqualität eines Menschen verbessert werden kann. Unter diesen Forschungsbereichen befindet sich die Klassifizierung von menschlichen Bewegungen oder Aktivitäten. Diese zielen darauf ab, verschiedene körperliche Aktivitäten von Menschen automatisch zu erkennen. Diese Forschung ist derzeit ein herausforderndes, aber aufregendes Thema, da menschliche Aktivitäten komplex sind und von Individuum zu Individuum unterschiedlich ausgeübt werden. Aber die Erkennung von menschlichen Bewegungen ist für viele Bereiche wie Sport und Gesundheit sehr attraktiv, weil dadurch personalisierte Dienste angeboten werden können (1).

Mit der Weiterentwicklung der Sensortechnologie kann die automatische Aktivitätserkennung auf unauffällige und nicht aufdringliche Weise erfolgen, da die entsprechenden Beschleunigungssensoren heutzutage klein, preiswert und weit verfügbar sind (2). Die Aktuelle Erkennung menschlicher Bewegungen kann über maschinelles Lernen Klassifizierungsalgorithmen erfolgen. In diesem Zusammenhang sollten Merkmale erfasst oder verfügbar sein.

1.1 Allgemeine Vorgehensweise für menschliche Aktivitätserkennung

Die Abbildung 1-1 stellt eine allgemeine Vorgehensweise für die Aktivitätserkennung dar. Daraus kann man erkennen, wie relevante Merkmale aus den Beschleunigungssensoren extrahiert werden. Außerdem können Aktivitätsklassen mit Hilfe dieses Verfahrens klassifiziert werden.

Einleitung

Um eine Bewegung zu erkennen, müssen die charakteristischen Eigenschaften jeder Bewegung dargestellt werden. Die Erkennung oder Mustererkennung erlaubt bei der Bewertung, sowohl auf die vorliegende Aktivität zurückzuschließen als auch Regelmäßigkeiten und Ähnlichkeiten zu identifizieren. Um die genaueren Eigenschaften herauszufinden, werden Daten von jeder einzelnen Aktivität mittels Sensoren aufgenommen (Datenerfassung). Diese werden in kleinere Abschnitte aufgeteilt und zu jedem Abschnitt werden Merkmale berechnet (Merkmalsextraktion). Diese Merkmale können statistische Werte sein, wie z.B. die Standardabweichung, den Mittelwert oder digitale Signalverarbeitung-Werte wie z.B. Fast Fourier Transformation. Anschließend werden die Eigenschaften zusammengefasst und durch Klassifizierungsmethoden (Klassifizierung) bewertet.

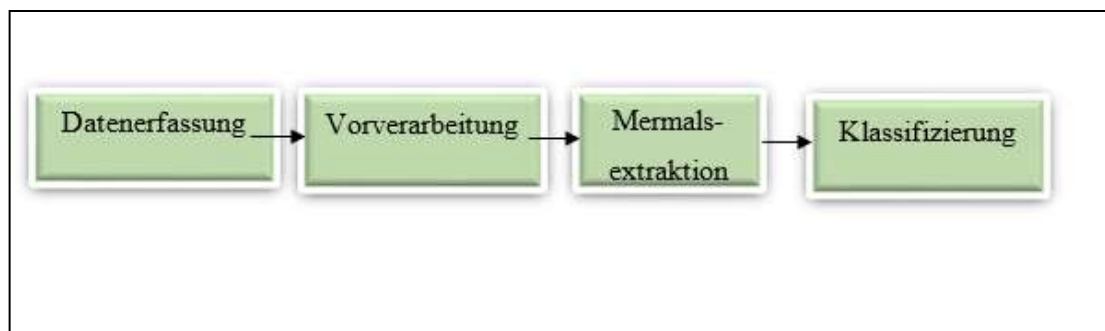


Abbildung 1-1: Allgemeine Vorgehensweise zur Aktivitätserkennung

1.2 Vorherige Untersuchungen

Zur Aktivitätserkennung mit Beschleunigungssensoren existieren bereits zahlreiche Arbeiten. Die Beobachtung von den Aktivitäten wurde auf Basis eines oder mehrerer Sensoren, die an bestimmten Körperpositionen befestigt waren, durchgeführt. Gemäß Dallas et. al. wurde ein Aktivitätserkennungssystem (3) entwickelt, das in der realen Anwendung der Patientenüberwachung eingesetzt werden kann. Es wurde ein an der Hüfte befestigter drahtloser dreiachsiger Beschleunigungssensor verwendet. Es wurde acht Aktivitäten betrachtet und zu jeder Aktivität wurden mehrere Merkmale wie die Energie, der Mittelwert, die Varianz usw. extrahiert. Die Aktivitätsklassifizierung wurde mit den Klassifikatoren „Naive Bayes“ und „k-nearest Neighbour“ (k-NN) durchgeführt und daraus entstand eine Erkennungsrate von ca. 98%.

Einleitung

Intile et. al. (4) haben Algorithmen entwickelt, um körperliche Aktivitäten durch Beschleunigungssensoren zu erkennen. Während des Experiments trugen die Probanden fünf zweiachsige Beschleunigungssensoren an dem Oberschenkel, Knöchel, Arm, Handgelenk, Hüfte. Es wurde eine Vielzahl von Aktivitäten wie Gehen, Sitzen, Stehen, Fernsehen, Laufen, Radfahren, Essen, Lesen usw. durchgeführt. Der Mittelwert, die Energie, die Frequenzbereich, Entropie und Korrelation der Beschleunigungsdaten wurden als Erkennungsmerkmale extrahiert und wurden zum Trainieren einer Gruppe von Klassifikationen benutzt. Die beste Leistung wurde mit dem Entscheidungsbaum erreicht und mit dieser Studie haben die Forscher die Aktivitäten mit einer Gesamtgenauigkeit von 84% erkannt.

Ein sogenannter mobiler Sensor wurde auch verwendet, wie es in (5) zu lesen ist. Gerald Bieber hat in seiner Arbeit ein Gesamtmodell zur Aktivitätserkennung für Patienten, die an Adipositas leiden, entwickelt. Für die Durchführung von Aktivitäten wurde ein Beschleunigungssensor von einem Mobiletelefon verwendet, das in einer Hosentasche getragen wurde. Dabei wurden die Aktivitäten: Gehen, Laufen, Fahrrad fahren, Ruhen, Auto fahren und aktiv sein klassifiziert. Es wurden als Erkennungsmerkmale Phasenlage, Richtungswechsel und Fourier Transformation herangezogen. Dazu wurde die Klassifizierung mittels Mehrheitsentscheids Klassifikation durchgeführt und die Erkennungsrate lag bei ca. 97%.

Die Tabelle 1-1 zeigt eine Übersicht weiterer vorhandener Untersuchungen, bei denen die Bewegungserkennung mit Hilfe von Beschleunigungssensoren durchgeführt worden ist. Neben den angegebenen zusätzlichen Sensoren wurde immer der Beschleunigungssensor verwendet.

Tabelle 1-1: Übersicht über weitere Untersuchungen

Lit.	untersuchte Bewegungen	Sensoren	Merkmale	Klassifizierungsalgorithmen
(6)	Gehen, Joggen, Treppen steigen und absteigen, Sitzen und Stehen	Dreiachsige Beschleunigungssensor von 3 verschiedenen Android Handys Anzahl pro Pers.=1	Mittelwert, Standardabweichung Mittelwert der absoluten differenz, Mittelwert resultierender Beschleunigung	Dtree, logistic regression, neural networks
(7)	Stehen, Gehen, Laufen, Treppen steigen, Staubsaugen, Zähne putzen	dreiachsigen Beschleunigungssensor	Mittelwert, Standardabweichung Energie, Korrelation	Plurality Voting 17 andere
(8)	Gehen, Gehen beim Tragen von Gegenständen, Stehen, Laufen, und 15 andere	Zweiachsige Accceloremeter Anzahl pro Pers.= 5	Mittelwert, Energie, Entropie und Korrelation	Dtree, und Naive Bayes
(9)	Gehen, laufen, Auto fahren, Fahrradfahren	Dreiachsige Beschleunigung, Orientierungssensor, Magnetisches Feld Sensor, GPS von Android	Mittelwert, Standardabweichung	Naive Bayes, Support Vector machine, Dtree
(2)	Rennen, Treppen hinunter- hinaufsteigen, Gehen, Ruhestand, Rolltreppe fahren, Aufzug fahren	Dreiachsige Beschleunigungssensor von Android	Mittelwert, Standardabweichung Fouriertransformation	Dtree, Hidden Markov Modelle

1.3 Ziele und Aufbau der Arbeit

Es gibt zahlreiche Arbeiten zum Thema Menschliche Aktivitätserkennung, aber in den meisten Arbeiten, wo die Beschleunigungssensoren verwendet wurden, wurden meist alltägliche Übungen, wie Gehen, Auto fahren untersucht. Außerdem wurden bisher fast keine Rehabilitationsübungen wie Knie strecken, Knie Beugen betrachtet. In dieser Arbeit werden neben den alltäglichen Aktivitäten, auch Rehabilitation Aktivitäten klassifiziert. Es werden zwei Klassifikationsverfahren gegenübergestellt: „K-Nearest Neighbor“ (K-NN) und „Decision Tree“ (Dtree) zur Klassifikation von Bewegungen verwendet. Da hier beide Verfahren auf demselben Datensatz getestet werden, lassen sich die Ergebnisse direkt vergleichen.

Diese Arbeit ist wie folgt gegliedert: Im Folgenden Kapitel 2 werden Grundlagen vorgestellt, die für die hier umgesetzte Bewegungserkennung relevant sind. Dabei wird auf den gewendeten Sensortyp eingegangen und die verwendeten Tools und Klassifizierungsalgorithmen werden erläutert. Das Kapitel 3 beschreibt einerseits das für diese Arbeit verwendete Gerät für die Datensammlung und die dazugehörige Software und andererseits die betrachteten Aktivitäten. Kapitel 4 geht auf die Aufbereitung der Daten ein. Hier wird auf die Visualisierung und Einteilung der gesammelten Daten eingegangen. Im Kapitel 5 werden die relevanten Aktivitätsmerkmale dargestellt. Das Kapitel 6 geht auf die verwendeten Klassifikationsverfahren und ihre Implementierung ein. Die gewonnenen Ergebnisse werden im darauffolgenden Kapitel 7 vorgestellt. Außerdem werden die beiden verwendeten Klassifikationsverfahren bewertet und verglichen. Abschließend werden im Kapitel 8 ein Fazit dieser Arbeit sowie Anmerkungen für Zukünftige Arbeiten gemacht.

2 Grundlagen

Dieser Abschnitt führt zu den wichtigen Grundlagen, die für das Verständnis dieser Arbeit notwendig sind. Es werden die technischen Grundlagen, sowie die verwendeten Signalverarbeitungsalgorithmen erläutert. Außerdem werden die verwendeten Python Frameworks und Tools vorgestellt.

2.1 Beschleunigungssensor

Ein Beschleunigungssensor auch Accelerometer genannt ist ein Sensor, welcher die Änderung des Bewegungszustandes eines Gegenstandes oder eines Körpers messen kann (9). Beschleunigung ist definiert als ein gerichteter Vektor, welcher die Geschwindigkeit in Bezug auf die zeitliche Änderungsrate angibt. Die Formel für die Beschleunigung lautet, $a = \frac{\Delta v}{\Delta t}$ wobei a die Beschleunigung, Δv die Differenz der Geschwindigkeiten und Δt die Zeitspanne repräsentieren. Die Beschleunigung wird angegeben durch die SI-Einheit m/s^2 .

Ein Beschleunigungssensor misst in der Regel in G-Kräften die Beschleunigung als ein Vielfaches der Erdbeschleunigung, welche $9,81 m/s^2$ oder auch $1g$ entspricht. Ein Beschleunigungssensor in Ruhe zeigt die Erdbeschleunigung von $1g$ an. Es gibt verschiedene Arten von Beschleunigungssensoren, welche sich in der Messtechnik und den Messfühlern unterscheiden. Die meist verbreiteten Formen sind kapazitive-, piezoelektrische-, Hall-Effekt- und Mikro-Elektro-Mechanisches System (Abb. MEMS) -Accelerometer. (10)

Im Rahmen dieser Arbeit werden MEMS-Accelerometer verwendet. Bei der MEMS-Technologie handelt es sich um in Mikrometer kleine Strukturen, welche aus Silizium hergestellt werden (siehe Abbildung 2-1). Diese Sensoren sind Feder-Masse-Systeme, wobei die Federn und Massen aus dem Silizium herausgeätzt werden. Durch Beschleunigung wird die Masse ausgelenkt und die korrespondierende Kapazitätsänderung gemessen.

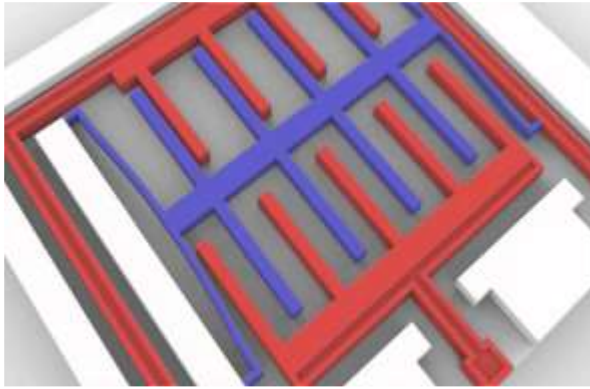


Abbildung 2-1: Beschleunigungssensor (25)

2.2 Informatik Grundlagen

2.2.1 Python

Python ist eine interpretierte, Objekt-orientierte Programmiersprache. Python wurde in dieser Arbeit benutzt, um die Codes zu schreiben. Während der Arbeit wurde die dritte Version verwendet. Python besteht aus umfangreichen Bibliotheken. Im Folgenden werden einige Hauptbibliotheken vorgestellt, die während der Arbeit benutzt wurden:

➤ **Pandas**

Um Daten aus CSV-Datei zu lesen und die zu bearbeiten wurde die Bibliothek Pandas verwendet. Pandas ist eine Bibliothek, welche eine hoch performante Datenstruktur bereitstellt. Sie bietet umfangreiche Datenstrukturen und Funktionen, die entwickelt wurden, um die Arbeit mit strukturierten Daten schnell und einfach zu machen. (12)

➤ **NumPy**

NumPy ist eine fundamentale Bibliothek für numerische Berechnungen in Python. Sie definiert ein N-dimensionales Array Objekt und stellt zugehörige Methoden bereit. (13)

➤ **Ipywidgets**

Ipywidgets, auch Ipython Widgets genannt, sind interaktive Python-Objekte, die oft als Steuerelemente wie Think-Slider, Testboxen, Buttons usw., sowohl im Kernel als auch im Frontend dargestellt werden. Diese können auch verwendet sowohl werden,

um interaktive grafische Benutzeroberflächen zu erstellen, als auch um zustandsbehaftete und zustandslose Information zwischen Python und anderen Programmiersprachen wie JavaScript zu synchronisieren. (14)

➤ **Scikit-learn/ Sklearn**

Zur Implementierung der Klassifizierung der Sensordaten wurde Scikit-learn verwendet. Scikit-learn ermöglicht eine schnelle Entwicklung aufgrund der zahlreichen bereits existierenden Funktionen, wodurch das Testen verschiedener Verfahren erleichtert wird. (13)

➤ **K-Nearest Neighbor (K-NN)**

Für die K-NN, auf Deutsch k-Nächste-Nachbarn, wurde die Funktion „*KNeighborsClassifier()*“ von Sklearn verwendet. Sie erlaubt das Anpassen von K-NN Algorithmus an das Modell und stellt die verschiedene Metrik Kriterien zur Verfügung. In Kapitel 0 wird genauer auf das verwendete Verfahren eingegangen.

➤ **Decision Tree (Dtree)**

Für die Dtree, auf Deutsch Entscheidungsbäume, wurde auch die Funktion „*DecisionTreeClassifier()*“ von Sklearn verwendet. Sie ist in der Lage Klassifikationsbäume und verschiedene Teilungskriterien zu erstellen. Im Kapitel 6.2 wird genauer auf das verwendete Verfahren eingegangen.

2.2.2 Jupyter Notebook

Jupyter ist eine Webanwendung, die unterschiedliche Funktionen miteinander verbindet. Dazu zählt eine Schnittstelle, die das Ausführen von Computercodes in mehr als 40 unterschiedlichen Programmiersprachen wie bspw. Python oder Ruby ermöglicht. Dabei stützt sich Jupyter auf unabhängige Programme, die in der Lage sind, die Sprache zu interpretieren, in der der Code geschrieben ist. In Jupyter's Terminologie werden diese Interpreter auch „Kernel“ genannt. Das ist ein Werkzeug, mit dem Multimedia-Dokumente erstellt werden können. Diese Dokumente können die Texte, mathematische Formeln, Grafiken, Bilder und sogar Animationen und Videos enthalten. (14)

2.2.3 Bokeh

Um Daten graphisch darzustellen, wurde Bokeh benutzt. Bokeh ermöglicht mit Hilfe von JavaScript-Widgets die Erstellung interaktive, zoombare Diagramme. Bokeh ist besonders dann geeignet, wenn die Visualisierung als Dashboard im Webbrowser erfolgen soll. (13)

3 Datenakquisition

Im Rahmen dieser Arbeit wurden die Daten von drei Probanden gesammelt. Während der Aufnahme wurden die Probanden aufgefordert, zufällige Sequenzen von Aktivitäten durchzuführen, die auf einem Arbeitsblatt definiert waren. Die Probanden sollten die Aktivitäten in ihrem eigenen Tempo und ihrer eigenen Reihenfolge durchführen und sollten auch die Start- und Endzeiten jeder Aktivität auf Blatt schreiben. Ein Beispiel von diesem Aktivität-Arbeitsblatt befindet sich in Anhang C, Abbildung C-1. Die Durchführung erfolgte zum einen im Institut Fraunhofer MEVIS und zum anderen zu Hause.

Für die Datenaufnahme wurde ein dreiachsiger Beschleunigungsmesser Axivity AX3 von der Firma Axivity benutzt. Die aufgenommenen Daten haben die folgenden Attribute: Zeit, Beschleunigung entlang der x-Achse, der y-Achse und der z-Achse.

3.1 Axivity AX3

Axivity AX3 ist ein Datenlogger der Firma Axivity (Großbritannien). Wie es auf der Abbildung 3-1 zu sehen ist, handelt es sich um ein kleines Gerät mit einer Größe von 32x32.5x7.6 mm. Der AX3 ist ein kombinierter Logging-Sensor zur Erfassung von Daten physikalischer Aktivitäten des Menschen. Im Zentrum des Sensors befinden sich ein dreiachsiger Mikro-elektro-mechanischer System-Beschleunigungssensor (MEMS) und ein NAND-Flash-Speicherchip, der über einen USB-fähigen Mikrocontroller verbunden ist. Ein Temperatursensor, ein Umgebungslichtsensor, eine Echtzeituhr (RTC) und eine Lithium-Polymer-Stromquelle sind ebenfalls in der hermetisch dichten Kunststoffkapselung integriert. Der Datenlogger erlaubt den Zugriff auf die Rohdaten, die auf einem NAND-Speicher gespeichert werden. Jeder Datensatz wird dabei mit einem präzisen Zeitstempel markiert und gespeichert (15).



Abbildung 3-1: Axivity -AX 3

Der Beschleunigungssensor besitzt eine einstellbare Empfindlichkeit, um eine optimale Datenaufnahme verschiedener Aktivitäten zu ermöglichen. Die wählbaren Bereiche sind $\pm 2g$, $\pm 4g$, $\pm 8g$ und $\pm 16g$. Hierbei ist g die Beschleunigung aufgrund der Schwerkraft. Die vier Empfindlichkeitsbereiche und die dazugehörigen Aktivitäten, um eine optimale Datenaufnahme zu gewährleisten befindet sich in Tabelle 3-1.

Tabelle 3-1: Empfindlichkeitsstufen dem Axivity AX3

Empfindlichkeit	Beispiel Anwendungsfall
$\pm 2g$	feine Bewegungen wie Handschrift oder Malerei
$\pm 4g$	Milde Aktivitäten wie Wandern
$\pm 8g$	Moderate Aktivitäten wie Sprinten, Springen
$\pm 16g$	schwere Aktivitäten wie Boxen

3.2 Aufnahme

Für eine erfolgreiche Datenerfassung sollte der Bewegungssensor sicher am gewünschten Körperteil angebracht sein. Jede ungewollte Vibration oder Bewegung

kann die Daten verfälschen. Außerdem müssen für alle aufgezeichneten Daten die Ausrichtungen und der Montageort des „Axivity AX3“ identisch sein, um vergleichbare Datensätze zu erhalten.

Für diese Arbeit wurden zwei Positionen ausprobiert: am Bein und am Knie. Dabei hat sich herausgestellt, dass einige Übungen schwer zu erkennen sind, wenn das Gerät am Fuß getragen wurde. Deshalb wurde das Gerät daraufhin am Knie befestigt. Das Gerät wurde während der Durchführung der Übungen mit einem Klebeband fixiert, damit es nicht rutscht. Für die Ausrichtung des Geräts wurde die Empfehlung des Herstellers befolgt und die Richtung des USB nach unten gesetzt (16). Die Abbildung 3-2 zeigt, wie das Gerät getragen wurde.

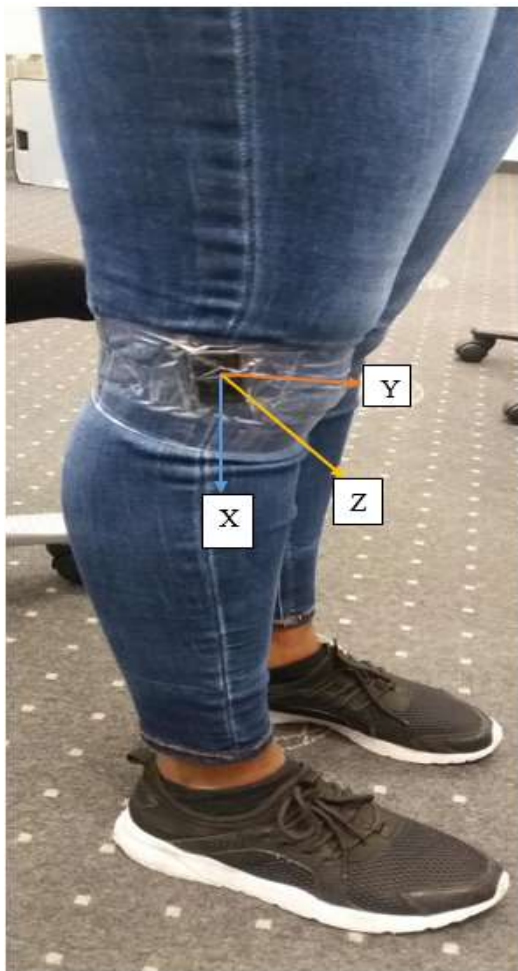


Abbildung 3-2: Axivity-AX3 Montage

Bevor die Datenerfassung beginnt, muss der „Axivity AX3“ konfiguriert werden. Dazu wird der AX3 an einen Computer angeschlossen, auf dem die Software OM GUI installiert wurde. Die OM GUI Software ist die Benutzeroberfläche zur Steuerung des AX3. Anschließend werden alle relevanten Parameter ausgewählt. Diese Parameter umfassen Abtastrate, Empfindlichkeit, Band, Sport und Ort und werden

Datenakquisition

als „**recordSetup**“ (Siehe Abbildung 3-3) gespeichert, nachdem der AX3 mit dem Computer getrennt wurde. In dieser Arbeit wurden eine Frequenz von 100 Hz und eine Empfindlichkeit von $\pm 8g$ verwendet.

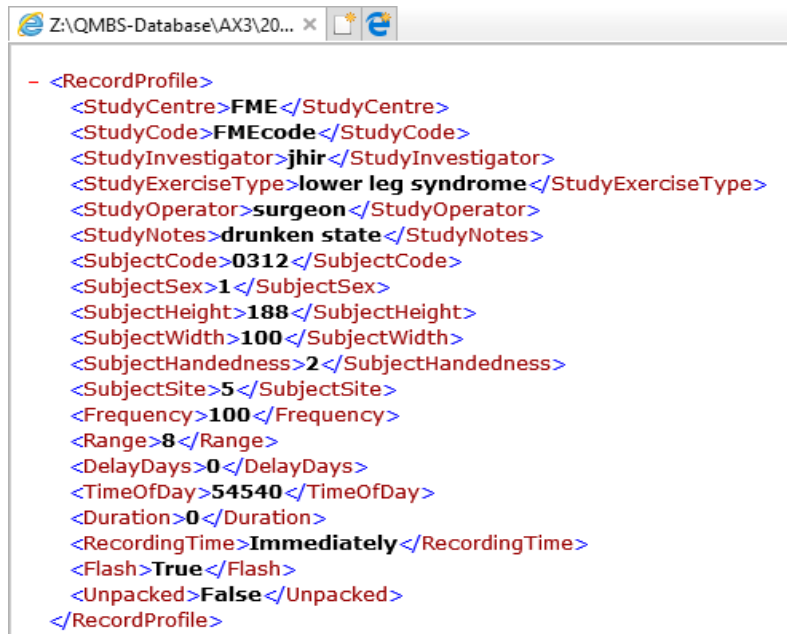


Abbildung 3-3: Beispiel von "recordSetup"

Während dieser Arbeit wurde das Aufnahmeverfahren „**Immediately**“ gewählt. Das bedeutet, dass sobald das Gerät mit dem Computer getrennt wird, startet die Aufnahme. Um die Aufnahme zu beenden, muss das Gerät wieder an den Computer angeschlossen und durch den Klick auf „**Stop**“ gestoppt werden. Die aufgenommenen Daten werden direkt aufgezeigt und mit Hilfe des Knopfes „**Download**“ werden die in einer CWA-Datei gespeichert. Ein Beispiel für die Visualisierung von Daten mit der OM GUI Software ist in **Abbildung 3-4** zu sehen. Alle während dieser Arbeit aufgenommenen Daten wurden auf dem „**gaia-server**“ von dem Fraunhofer Institut MEVIS gespeichert.

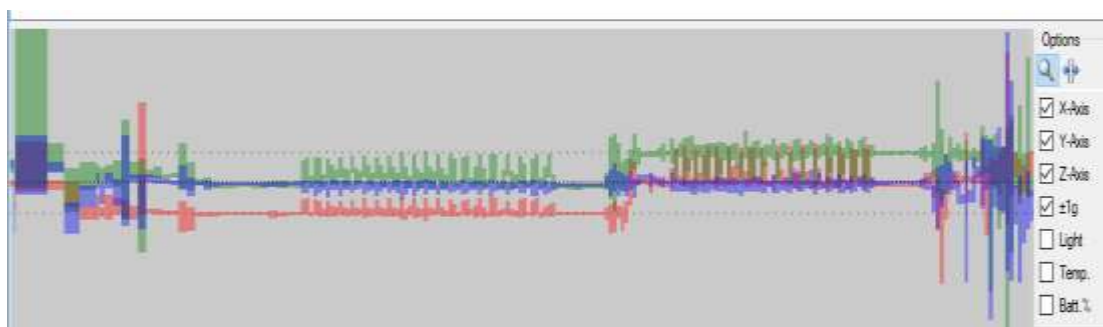


Abbildung 3-4: Visualisierung von den Aktivitäten halbe Kniebeugung und Kniebeugen strecken mit der OM GUI

3.3 Aktivitäten

Die gesammelten Aktivitäten werden aufgeteilt in alltägliche Aktivitäten, d.h. Aktivitäten, die im Alltag durchgeführt werden, wie z.B. Gehen und in die Reha-Aktivitäten, wie z.B. Kniebeugung.

Alltägliche Aktivitäten, die ausgeführt wurden, sind: Gehen, Laufen und Treppen hinauf- und heruntersteigen. Zu den Rehabilitation-Aktivitäten gehörten die halbe Kniebeugung, das Kniebeugen-strecken und das Beinpendeln. Die Durchführung der Rehabilitation-Übung wurde genau erklärt und mehrere Male wiederholt.

➤ **Halbe Kniebeugung**

In dieser Übung muss der Proband mit den Füßen schulterweit auseinander stehen. Die Hände werden dann nach vorne gestreckt. Bei Bedarf ist es erlaubt, sich an der Rückenlehne eines Stuhls oder an einer Wand festzuhalten. Danach muss die Brust hochgehalten und die Hüften langsam um etwa 10 cm gesenkt werden, als ob der Proband sich auf einen Stuhl setzen und wieder aufstehen würde.

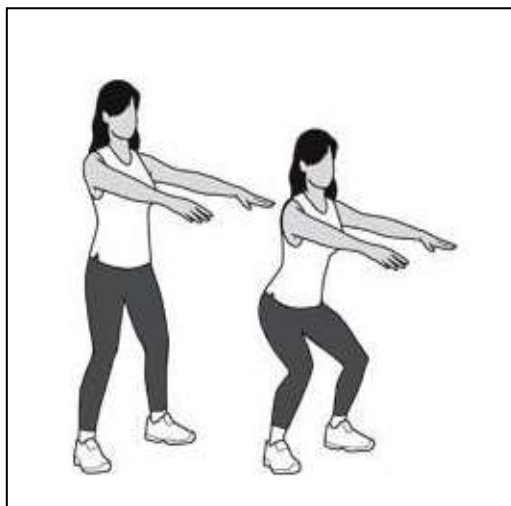


Abbildung 3-5: Halbe Kniebeugung (28)

➤ **Beinpendeln**

Für die Übung sitzt der Proband entweder auf einem Tisch oder auf einem Stuhl. Danach werden die Zehen nach oben gezogen, die Oberschenkelmuskeln gestrafft und am Ende das Bein vor und hinter bewegt.

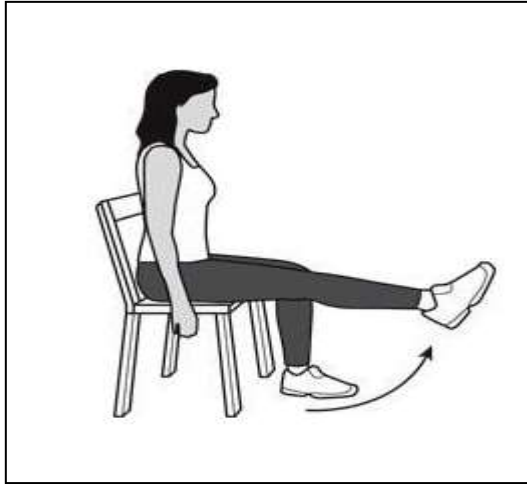


Abbildung 3-6: Beinpendeln (28)

➤ **Kniebeugen-strecken**

Während dieser Übung muss der Proband auf dem Fußboden gerade liegen, das Knie in Richtung Brust beugen, danach das Bein wieder langsam strecken und schließlich das Bein wieder auf den Boden bringen.

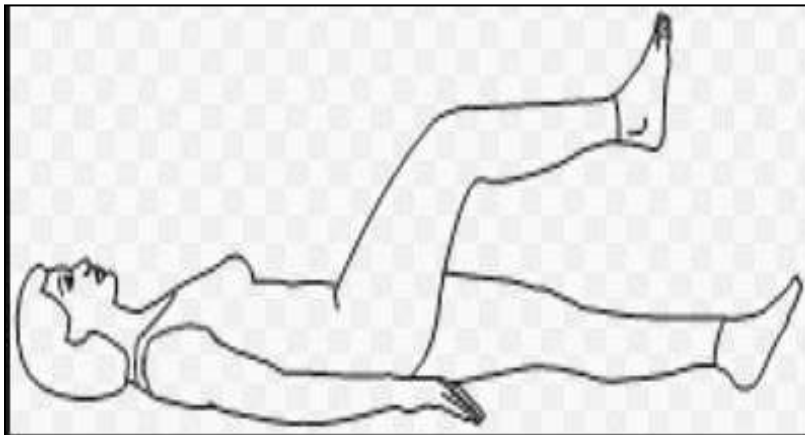


Abbildung 3-7: Kniebeugen strecken

Für die Klassifizierung wurden nur die Daten von zwei Probanden verwendet, weil bei einem der Probanden das Gerät falsch am Knie montiert wurde (die Richtung des USB zeigte nach oben). Die Tabelle 3-2 gibt Ausschlüsse über die Anzahl der gesammelten Daten pro Probanden und Aktivitäten. Insgesamt wurden 112955 Datensätzen aufgenommen.

Tabelle 3-2: Anzahl Datensätze Pro Probanden und Aktivitäten

	Anzahl von Datensätze Pro Probanden		
Aktivitäten	Proband 1	Proband 2	Summe
halbe Kniebeugung	7496	11039	18535
Beinpendeln	9827	11442	21269
Kniebeugen-strecken	5882	5147	11029
Gehen	26153	7722	33875
Laufen	8171	3712	11883
Treppen hinaufsteigen	5317	2797	8114
Treppen heruntersteigen	4968	3282	8250
Summe	67814	45141	112955

4 Datenverarbeitung

4.1 Konvertierung von CWA- Dateien in CSV-Dateien

Um die Datenverarbeitung zu vereinfachen, wurden die CWA-Dateien in CSV-Dateien umgewandelt. Die Umwandlung geschieht mithilfe der OM-GUI Software. Dabei wurden zuerst die zu konvertierenden CWA-Dateien ausgewählt und der Exporttyp durch das Tool „**Export**“ gewählt. In diesem Fall wurde „**Export Raw CSV**“ gewählt. Dann musste der Speicherort definiert und das Konvertierungszeit-Format, in diesem Fall „**Formatted (Y-M-D h:m:s,f)**“, festgelegt werden. Anschließend wurde die Konvertierung durch den Knopf „**Convert**“ gestartet. Am Ende wurde die CSV-Datei automatisch gespeichert.

4.2 Daten Visualisierung

Wie es in Kapitel 3.2 zu lesen ist, bietet die Software OM-GUI die Möglichkeit an, die Daten zu visualisieren, (siehe Abbildung 3-4). Im Rahmen dieser Arbeit wurden mehrere Notebooks programmiert. Dazu zählt ein Programm, das die Messdaten visualisiert.

Das Notebook für die Visualisierung wurde „**Visualisation_interactiveInput.v1.7**“ genannt. Eine Anleitung zur Verwendung des Notebooks befindet sich in Anhang A unter dem Kapitel „Benutzerhandbuch“. Dieses Notebook wurde in mehrere Tabs gegliedert und der Tab zum reinen Anzeigen der Messwerte besitzt den Namen „**Show Data**“. Dieser Tab besitzt mehrere Funktionen. So können Datensätze gleichzeitig mit ihren gleitenden Mittelwerten und Standardabweichungen zusammen oder separat angezeigt werden. Die Daten können entweder in absoluter Zeit oder in Sekunden geplottet werden. Die folgenden Abbildungen zeigen jeweils 20 Sekunden von den sieben verschiedenen Aktivitäten. Die blaue Farbe ist die X-Achse, die rote die Y-Achse und die grüne die Z-Achse von den Daten.

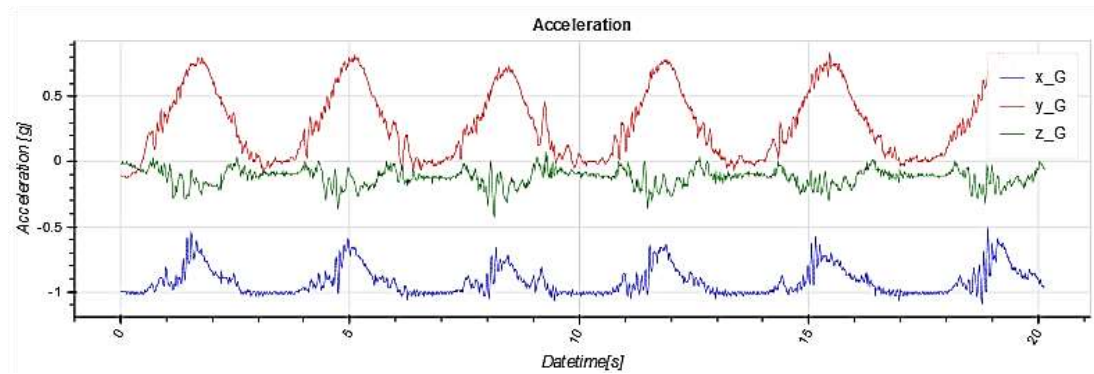


Abbildung 4-1: Beschleunigungssignal für halbe Kniebeugung

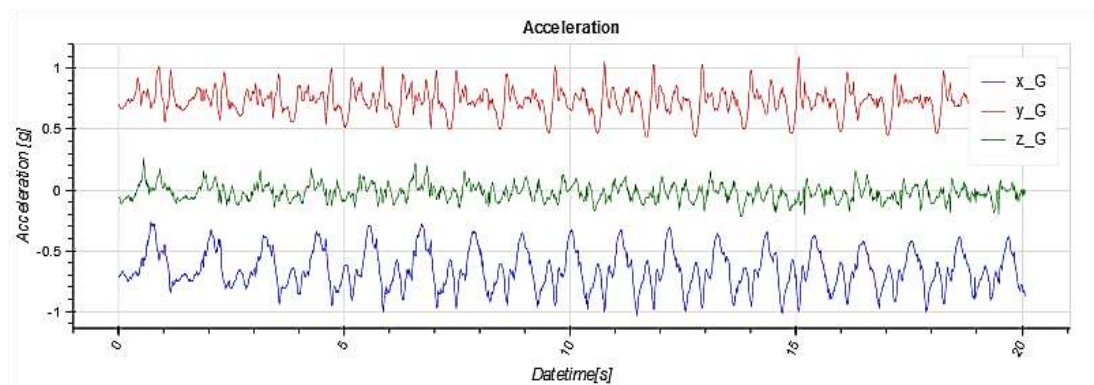


Abbildung 4-2: Beschleunigungssignal für Beinpendeln

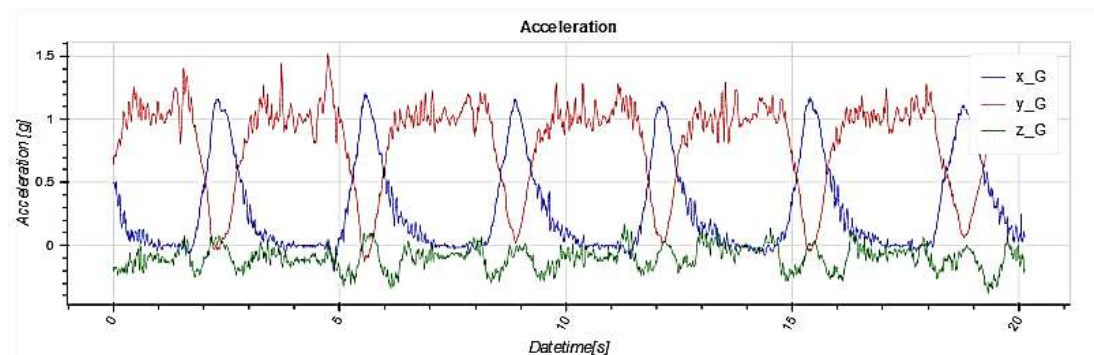


Abbildung 4-3: Beschleunigungssignal für Kniebeugen-strecken

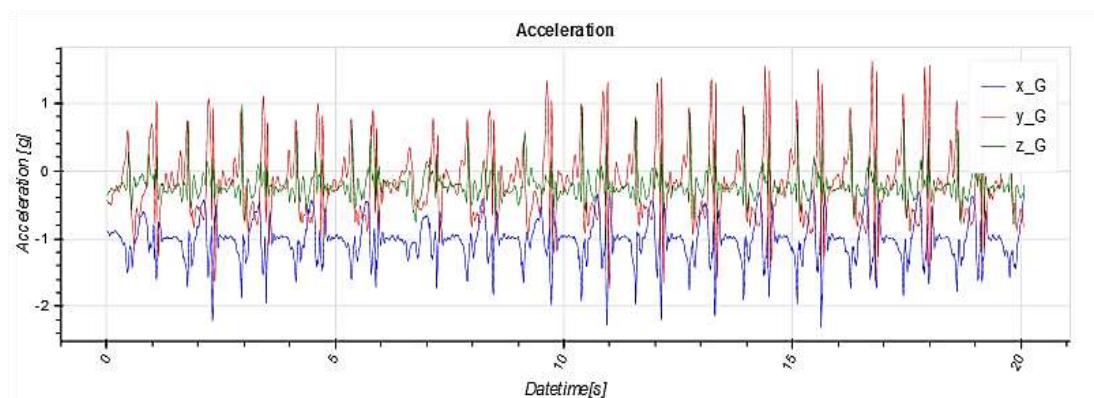


Abbildung 4-4: Beschleunigungssignal für Gehen

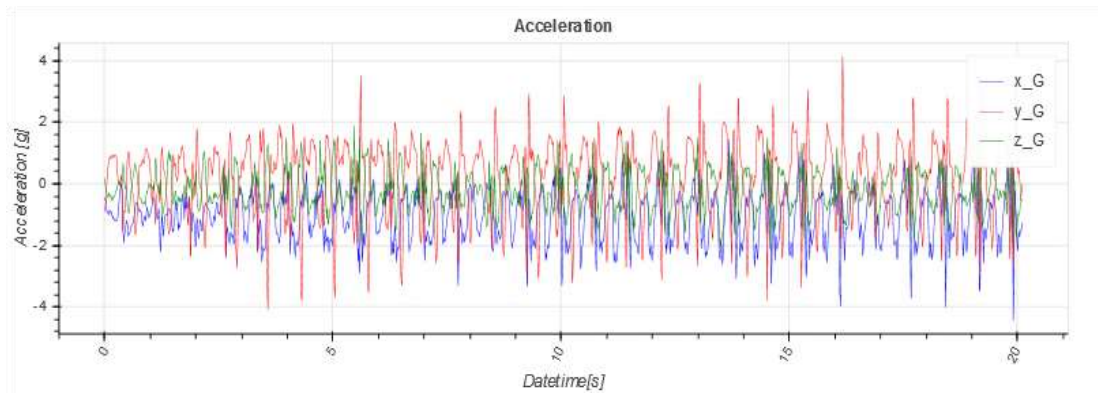


Abbildung 4-5: Beschleunigungssignal für Laufen

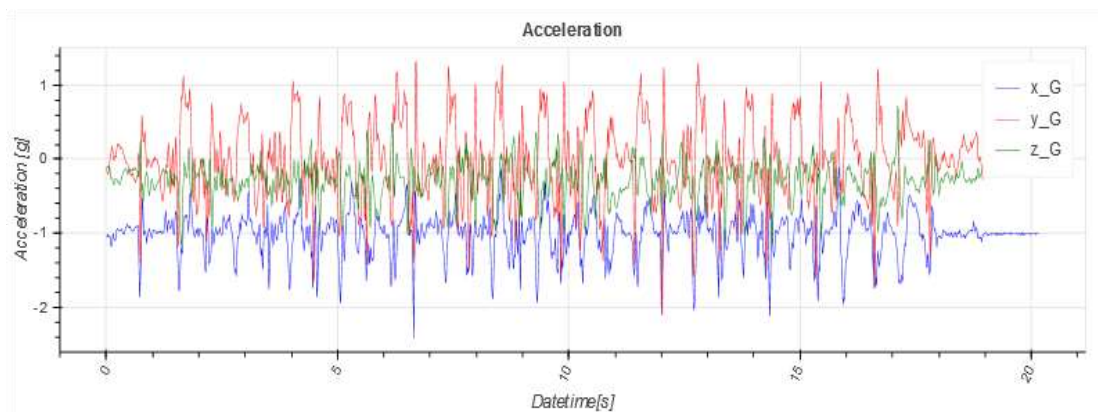


Abbildung 4-6: Beschleunigungssignal für Treppen heruntersteigen

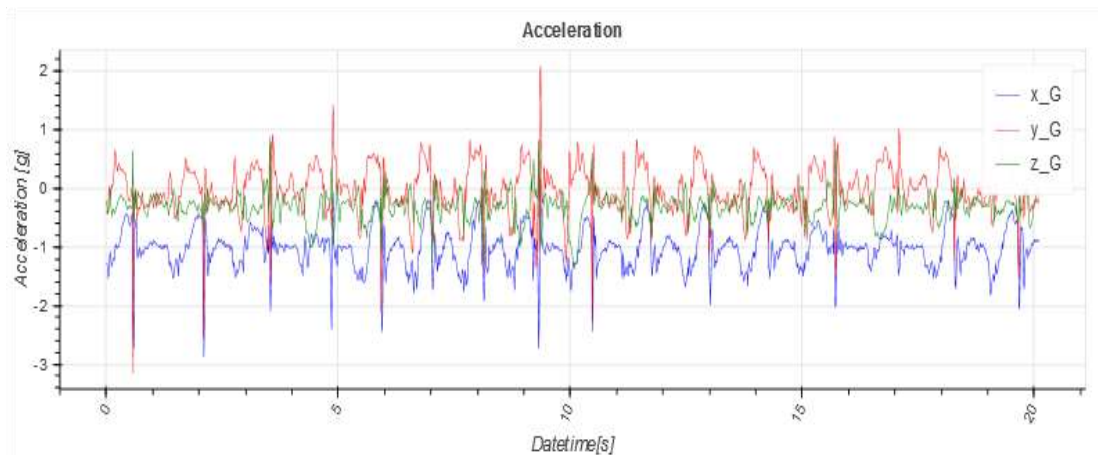


Abbildung 4-7: Beschleunigungssignal für Treppen hinaufsteigen

4.3 Vorverarbeitung der Datensätze

Da die aufgenommenen Rohdaten als zahlreiche Einzeldateien vorliegen, wurden die Daten in Aktivitätsbereiche geschnitten und später zusammengefügt, sodass für jede der Aktivitäten ein einzelner durchgängiger Datensatz vorliegt.

4.3.1 Teilen von Daten in Aktivitätsbereiche

Zum Teilen der Daten in Aktivitätsbereiche wurden drei Methoden implementiert.

Alle drei Methoden befinden sich in dem Notebook *Visualisation_interactiveInput.v1.7*. Durch die Visualisierung konnte schon ermittelt werden, wie viele Schnittpunkte zum Einsatz kommen können.

➤ Methode 1

Die Ausführung dieser Methode befindet sich in dem Tab „*Show cutpoints and Split File 1*“ des Notebooks. Die Abbildung 4-8 stellt die GUI von diesem Tab dar.

Für diese Methode muss zuerst in „*Crop marks*“ die Anzahl der Schnittpunkte eingegeben werden. Dann werden die Daten nach der Anzahl der Schnittpunkte in kleine Bereiche aufgeteilt und jeder Bereich wird nochmal in kleine Blöcke geteilt. Die Anzahl der kleinen Blöcke muss in „*Window number*“ festgelegt werden. Zu jedem Block werden die Standardabweichungen von allen drei Achsen berechnet und daraus wird der Mittelwert errechnet. Anschließend wird der Block mit dem kleinsten Mittelwert gesucht. Am Ende wird der Mittelpunkt von diesem Block als Schnittpunkt für den Bereich genommen. Eine ausführliche Beschreibung dieser Tab befindet sich im Anhang A und der Algorithmus zu dieser Methode im Anhang B.

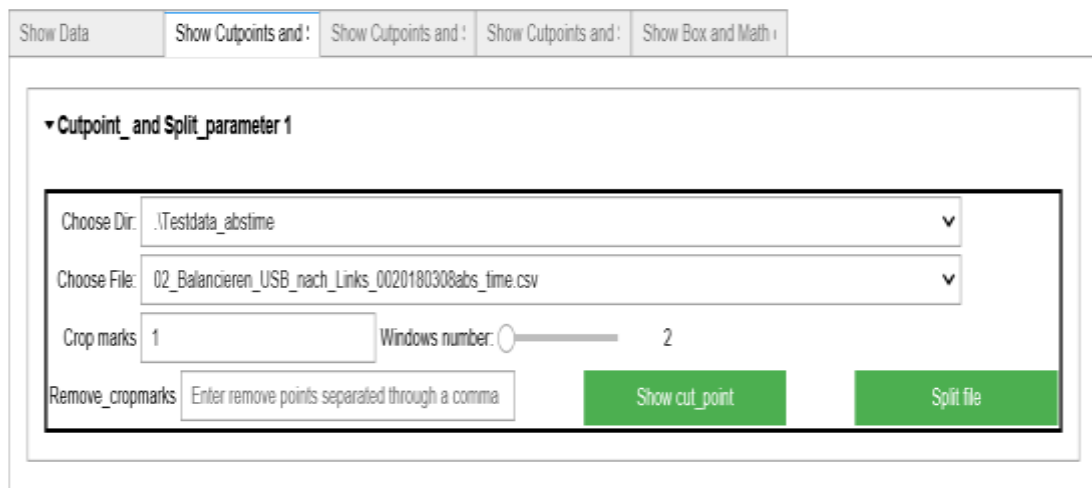


Abbildung 4-8: Auswahlmenü von Tab“ Show Cutpoints and Split File 1“

➤ Methode 2

In dieser Methode geht es darum, den gewünschten Schnittpunkt selbst einzugeben. Alle Punkte müssen in Sekunden eingegeben und durch ein Koma getrennt werden. Im Vergleich zu der ersten Methode wurde hier kein Algorithmus geschrieben, denn

die eingegebenen Punkte werden automatisch übernommen und eingesetzt. Das Programm für diese Methode ist in dem Tab „**Show Cutpoint and Split File 2**“ zu finden (siehe Anhang A).

➤ Methode 3

Die dritte Methode befindet sich in dem Tab „**Show Cutpoints and Split File 3**“.

Abbildung 4-9 zeigt, wie die GUI von diesem Tab aussieht. Hier muss auch, wie bei der ersten Methode, die Anzahl der gewünschten Schnittpunkte in das dritte Feld des Menüs eingegeben werden. Danach wird aus der Länge von den Vektoren mit den Komponenten (x_G, y_G, z_G) ein vierter Datensatz gebildet. Formel 1 stellt die Gleichung für die Berechnung dieser Länge dar.

Formel 1: Berechnung von Vektor Länge (14)

$$|P| = \sqrt{x_G^2 + y_G^2 + z_G^2}$$

Nach der Erstellung des vierten Datensatzes wird die gleitende Standardabweichung von diesen Daten berechnet. Dann wird der Punkt mit dem kleinsten Wert gesucht und in einer Liste gespeichert. Weiterhin muss eine Zahl in „**Remove Window**“ ausgewählt werden. Dann werden alle Daten, die sich ober- oder unterhalb der in „**Remove Window**“ eingegebenen Zahl von dem Punkt mit dem kleinsten Wert befinden, gelöscht, inklusive dieses Punkts. Dies wird so oft wiederholt, bis die angegebene Anzahl von Schnittpunkten erreicht ist. Zu dieser Methode wurde auch ein Algorithmus geschrieben. Die Beschreibung davon befindet sich im Anhang A.

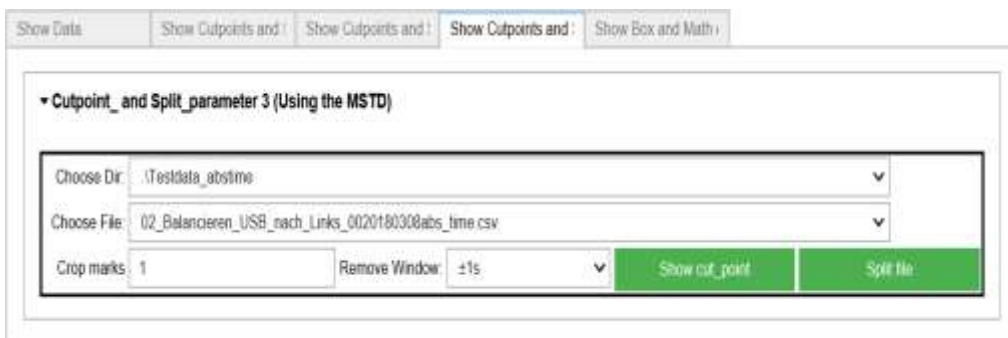


Abbildung 4-9: Auswahlmenü von Tab“ Show Cutpoints and Split File 3“

Die drei Funktionen, um die Schnittpunkte zu berechnen bzw. festzulegen, wurden in das Python-Modul mit dem Namen „**generalfunctions**“ geschrieben. Sie tragen die Namen „**cutpoint1()**“, „**cutpoint2()**“, „**cutpoint3()**“. Um die Daten an den Schnitt-

punkten zu teilen, wurde mit Hilfe von dem Modul „**Pandas**“ drei unterschiedliche Funktionen in dem oben genannten Python-Modul geschrieben. Sie wurden jeweils mit den Namen „**split_file1**“, „**split_file2**“ und „**split_file3**“ benannt.

Nachdem alle Parameter in dem Notebook eingegeben worden sind, können bei allen drei Methoden durch den Klick auf den Knopf „**Show Cut_point**“ die Schnittpunkte berechnet und angezeigt werden. Die Daten werden durch die Auswahl des Knopfes „**Split file**“ aufgeteilt und als CSV-Datei in einem automatischen erzeugten Unterverzeichnis gespeichert. Jedes Unterverzeichnis hat eine Bezeichnung, die sich aus dem Namen der geschnittenen CSV-Datei und der Schnittmethode zusammensetzt.

Beispielhaft lässt sich hier folgende Datei anführen: „**02_Balancieren_USB_nach_Links_0020180308abs_time_cut_point1**“. Die Teildateien haben jeweils als Name: „**Name der Datei + _Index**“. Ein Beispiel „**02_Balancieren_USB_nach_Links_0020180308abs_time_001**“

Da Bokerh keine großen Datenmengen visualisieren kann, gibt es die Möglichkeit, Daten vorher einzulesen, in kleine Datenmengen zu teilen und als CSV-Dateien zu speichern. Dafür wurden drei Funktionen geschrieben. Die Daten werden automatisch in einem ausgewählten Zeitabschnitt geteilt. Alle Zeitabschnittswerte wurden in Sekunden umgerechnet und werden während der Teilung mit der Frequenz der aufgenommenen Daten multipliziert. Die drei Funktionen wurden in einem Notebook mit dem Namen „**Split CSVFile.v1.3**“ zusammengefügt.

4.3.2 Zusammenfügen von Daten

Nachdem die Daten geschnitten wurden, wurden die Dateien manuell nach Aktivitäten sortiert und in dem zugeordneten Ordner gespeichert. Das heißt, für jede Aktivität gab es einen Ordner. In diesem Ordner wurden alle Daten, die zu dieser Aktivität gehören, gespeichert.

Des Weiteren wurden die Dateien so zusammengefügt, dass es am Ende für jede Aktivität nur eine Datei gab. Dafür wurde eine Funktion mit dem Namen „**Data_merge()**“ geschrieben. Wie auf der Abbildung 4-10 zu sehen ist, wurde die Methode „**Concat()**“ von Pandas benutzt.

```
def Data_merge():
    """this function merge multiple csv files"""

    dflist = [pd.read_csv(open(f, 'r'), parse_dates=[0], index_col=[0], header=None)

    for f in os.listdir(os.getcwd()) if f.endswith('.csv')]
    Neudf = pd.concat(dflist, axis=0, join='outer')
    return(Neudf)
```

Abbildung 4-10: Python Programm für das Zusammenfügen von Dateien

Die CSV-Dateien werden mit Pandas geöffnet, zusammengefügt und als Dataframe zurückgegeben. Diese Funktion wurde in einem selbst erstellten Python-Module mit dem Namen „**Generalfunction1**“ gespeichert. Um diese Methode auszuführen, wurde ein Tab mit dem Namen „**merge CSV_File and Save**“ in dem Notebook „**Merge_Feature_classification**“ erstellt. Die Abbildung 4-11 zeigt, wie das Auswahl-Menü aussieht. Hier geschieht die Auswahl der Verzeichnisse und Dateien über ein Dropdownmenü. Nach der Auswahl von einem Ordner unter „**Choose Dir**“ werden automatisch in dem „**Choose File**“ alle CSV-Dateien, die sich in diesem Ordner befinden, aufgelistet. So kann überprüft werden, ob alle Dateien, die sich in dem gewählten Ordner befinden, die richtigen sind. Durch den Klick auf den Knopf „**Merge CSV_file**“ werden die Dateien zusammengefügt und als Dataframe dargestellt. Nachdem der Namen der neuen Dateien in „**Neu_Filename**“ und des Speicherorts in „**directory_name**“ geschrieben worden sind, können durch den Klick auf dem Knopf „**Save Merge CSV_file**“ die Daten als CSV-Datei gespeichert werden.

Abbildung 4-11: Auswahl Menü für das Tabs „merge CSV_File and save“

4.4 Fensterung

Die Standard-Klassifikationsalgorithmen können nicht direkt auf Roh-Zeitserien-Beschleunigungsmesserdaten (2) angewendet werden. Deswegen wurden aus den daraus entstandenen durchgängigen Datensätzen kleine Fenstergrößen generiert. Diese Fenster wurden zur Merkmalsextraktion genutzt.

In früheren Arbeiten variieren die Fenstergrößen zwischen eine Sekunde (18) und zwölf Sekunden (19). Eine Fenstergröße von einer Sekunde erklärt sich dadurch, dass eine Sekunde ausreichen für einen Fußschritt ist (18). Die optimal Fenstergröße hängt von der Übung ab (9). In dieser Arbeit wurden die zusammenhängenden Daten in drei unterschiedlichen Fenstergrößen aufgeteilt. Die generierten Fenster sind jeweils eine Sekunde, drei Sekunden und sechs Sekunden groß. Das heißt mit einer Frequenz von 100 Hz ergeben sich bei einer Fenstergröße von einer Sekunde 100 Datensätze, bei drei Sekunden 300 Datensätze und bei sechs Sekunden 600 Datensätze pro Fenster.

Es ist meistens sinnvoll, dass sich die Fenster überlappen, denn so könnten nicht nur mehrere Zeitfenster betrachtet werden, sondern auch die Informationsverluste an den Fensterrändern reduziert werden (20). In dieser Arbeit wurde eine Fensterüberlappung von 50% betrachtet. Die Abbildung 4-12 zeigt ein Beispiel von Fensterung mit Überlappung von 50 %. Es sei das Fenster 1 mit einer Länge L . Bei einer Fensterung mit Überlappung von 50 % fängt das Fenster 2 in der Mitte von dem Fenster 1 an. Das heißt bei $\frac{L}{2}$

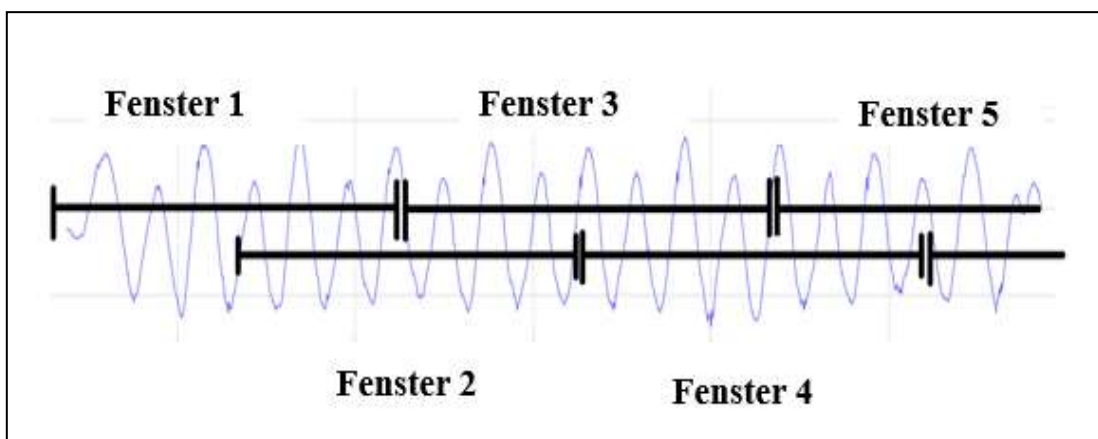


Abbildung 4-12: Beispiel Daten Fensterung mit 50% Überlappung

5 Merkmale

Die Merkmale spielen in der Mustererkennung eine wichtige Rolle. Sie repräsentieren Daten, welche die Eigenschaften eines Musters numerisch beschreiben und sie verdichten das Ursprungssignal zu einem charakteristischen Wert. In dem Bereich der Aktivitätserkennung kommen sehr viele verschiedene Merkmale ins Frage. In dieser Arbeit wurden durch eine Literaturrecherche [(21), (19), (4), (7)] häufig verwendete Merkmale aus anderen Anwendungen ermittelt und verwendet. Momentan werden die Merkmale Mittelwert, Standardabweichung, Energie und Korrelation berücksichtigt.

Sei $X = \{X_1, \dots, X_n\}$ eine Menge von Beschleunigungsdaten, mit $X_k = (x, y, z)$.

➤ Mittelwert

Der Mittelwert ermöglicht allgemeine Rückschlüsse auf die Bewegung.

Formel 2: Berechnung Mittelwert

$$\bar{X} = \frac{1}{N} * \sum_{k=0}^{n-1} X_k$$

➤ Standardabweichung

Dieses Merkmal beschreibt, wie stark die Signale bezüglich ihres Durchschnitts schwanken schwankt (5)

Formel 3: Berechnung der Standardabweichung

$$\sigma_x = \sqrt{\frac{1}{N} * \sum_{k=0}^{n-1} (X_k - \bar{X})^2}$$

➤ Energie

Dieses Merkmal beschreibt die Intensität der durchgeführten Bewegung (5). Dieses Merkmal kann im Zeitbereich durch die Quadratsumme aller Beschleunigungen geteilt durch die Fensterbreite, berechnet werden. Das kann auch in Frequenzbereich berechnet werden. Dabei ist die Energie, die Summe der quadrierten diskreten FFT-Komponentengrößen X_i des Signals. Die Summe wurde durch die Fensterlänge (N) für die Normalisierung geteilt (7).

Formel 4: Berechnung der Energie in Frequenzbereich (7)

$$\text{Energie} = \frac{\sum_{i=1}^{|w|} |x_i|^2}{|N|}$$

➤ **Korrelation**

Die Korrelation beschreibt eine Beziehung zwischen zwei oder mehreren Signalen. Die Korrelation wird zwischen jedem Achsenpaar als das Verhältnis der Kovarianz $\text{cov}(x, y)$ und des Produkts der Standardabweichungen $(\delta_x \delta_y)$ berechnet (22).

Formel 5: Korrelation zwischen zwei Punkte (23)

$$\text{corr}(x, y) = \frac{\text{cov}(x, y)}{\delta_x \delta_y}$$

Für die Berechnung dieser Merkmale wurden Funktionen mit Hilfe von „Pandas“ und „Numpy“ in dem Modul „**generalfunctions**“ geschrieben. Diese tragen jeweils die Namen „**calcul_mean ()**“, „**calcul_std ()**“, „**calcul_corr ()**“ und „**calcul_energy ()**“. Für die Ausführung der Funktionen wurde in dem Notebook „**Merge_Feature_classification**“ ein Tab mit dem Namen „**calcul Feature and Save**“ (Siehe Abbildung 5-1) erstellt. In diesem Tab wird die ausgewählte Datei in einem Zeitfenster von einigen Sekunden aufgeteilt. Das gewünschte Zeitfenster wird unter „**Choose Blocksize**“ ausgewählt und die Werte werden mit der in „**Frequency**“ eingegebenen Aufnahmefrequenz multipliziert. Für jedes dieser Zeitfenster werden die Merkmale berechnet.

Nachdem die Zeitfenster und die Aktivität ausgewählt wurden, können die Merkmale mit dem Klick auf den Knopf „**Show Feature**“ als Dataframe mit 13 Spalten zurückgegeben werden. In der 13. Spalte befindet sich der Name der durchgeführten Aktivität. Dieses Dataframe kann durch den Klick auf „**Save Feature**“ als CSV-Datei gespeichert werden. Die Dateien werden in dem Ordner „**Feature**“ angelegt und sie tragen die Namen „**Feature + Name der ausgewählten Aktivität**“.

Später wurden für diese Arbeit die Dateien mit den Merkmalen so zusammengefügt, dass es am Ende für die weiteren Schritte nur eine Datei mit allen Merkmalen und Aktivitäten gab.

Merkmale

The screenshot shows a software interface with three tabs: 'merge CSV_File an...', 'calcul Feature and S', and 'Classification'. The 'calcul Feature and S' tab is active. Below the tabs is a section titled '▼ Calcul Feature Parameters'. This section contains several input fields and buttons:

- 'Choose Dir:' with a dropdown menu showing './feature3s'.
- 'Choose File:' with a dropdown menu showing 'all_feature3s.csv'.
- 'choose Bolcksize' with a dropdown menu showing '0s'.
- 'frequency' with a text input field containing 'Enter frequency'.
- 'choose Activity' with a dropdown menu showing 'Halbe_Kniebeugung'.
- 'directory_name' with a text input field containing 'Enter the Name of Director'.
- A green button labeled 'show Feature'.
- A green button labeled 'Save Feature'.

There is also a label 'Widgets for length of the Data Block' next to the 'choose Bolcksize' dropdown.

Abbildung 5-1: Auswahl Menü Tab „Feature calcul and save“

6 Klassifizierung

Zur Klassifizierung von körperlichen Aktivitäten sind maschinelle Lernalgorithmen notwendig, welche die extrahierten Merkmale mit Merkmalen von bekannten Aktivitäten vergleichen und dadurch die ausgeführten Aktivitäten bestimmen. Die Erkennungsrate ist dabei von den ausgewählten Merkmalen, den Trainingsdaten sowie dem eingesetzten Klassifizierungsalgorithmus abhängig. Im Folgenden werden die eingesetzten Klassifizierungsalgorithmen kurz dargestellt.

6.1 K-Nearest Neighbor Algorithmus

K-Nearest Neighbor (Abgekürzt: K-NN) ist ein überwachter Lernalgorithmus und eine Art von Instanz-basiertem Lernen oder Lazy Learning, bei dem die Funktion nur lokal angenähert wird und alle Berechnungen bis zur Klassifizierung zurückgestellt werden (24). Der K-NN Algorithmus ist eine Methode zum Klassifizieren von Objekten, auf der Grundlage der nächsten oder ähnlichsten Trainingsbeispiele im Merkmalsraum. Dieser Algorithmus gehört zu den einfachsten aller maschinellen Lernalgorithmen. Ein Objekt wird durch einen Mehrheitsbeschluss seiner Nachbarn klassifiziert, wobei das Objekt dem Klassenlabel zugeordnet wird, welcher am häufigsten unter seinen sogenannten k-nächsten Nachbarn vorkommt. K ist dabei eine positive ganze Zahl. Wenn $K=1$ ist, dann wird das Objekt einfach der Klasse seines nächsten Nachbarn zugewiesen. Die Abbildung 6-1 zeigt ein Beispiel für eine K-NN Klassifizierung. Die Testprobe (hellblaue Stein) sollte entweder in die Klasse Laufen (rote Kreise) oder in die Klasse Gehen (blaue Dreiecke) eingeteilt werden. In diesem Fall ist $k=6$ (grüner Kreis). Es wird die Klasse Laufen zugewiesen, da es drei Dreiecke und nur zwei Kreise innerhalb des inneren Kreises gibt.

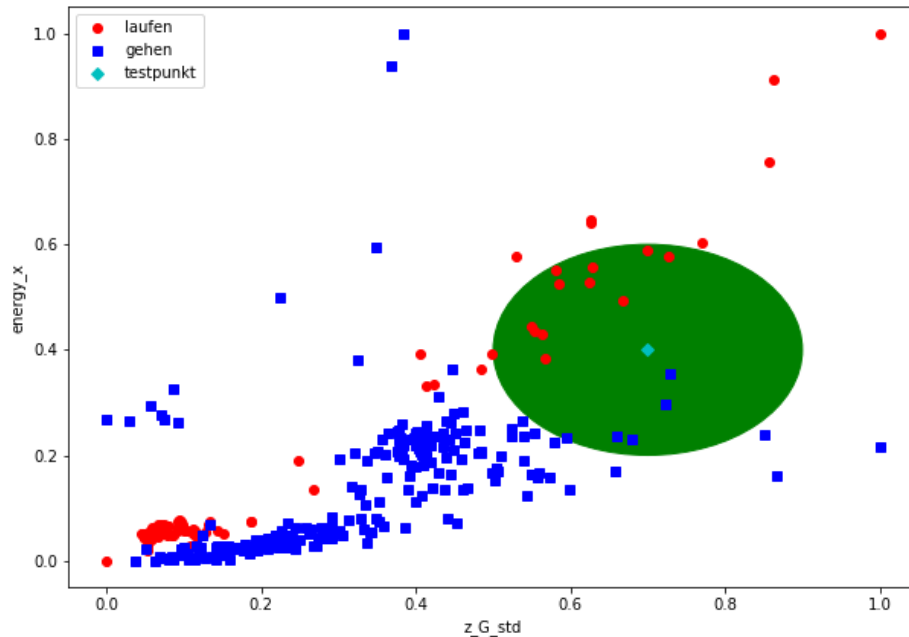


Abbildung 6-1: Beispiel Klassifikation mit K-NN, wenn $k = 6$

Die beste Wahl von k hängt von den Daten ab. Wenn k sehr klein ist, besteht die Gefahr, dass Rauschen die Klassifizierungsergebnisse verschlechtert. Wenn k jedoch zu groß ist, besteht die Gefahr, dass Punkte mit großem Abstand in die Ergebnisse mit einbezogen werden (25). Ein gutes k kann durch verschiedene heuristische Techniken ausgewählt werden. Für diese Arbeit wurde der Wert durch das Plotten der Erkennungsrate bezüglich der k -Werte ermittelt. Dadurch konnte der k -Wert mit der allerbesten Erkennungsrate detektiert und daraufhin verwendet werden.

Auch bei der Mehrheitsentscheidungsklassifizierung besteht die Gefahr, dass die Klassen mit den häufigsten Beispielen dazu neigen, die Vorhersage des neuen Vektors zu dominieren, weil diese aufgrund ihrer großen Anzahl in den nächsten Nachbarn auftauchen. Ein Weg, um dieses Problem zu überwinden, besteht darin, die Klassifizierung, unter Berücksichtigung der Entfernung von dem Testpunkt zu jedem seiner k -nächsten Nachbarn, zu gewichten. Es gibt mehrere Möglichkeiten, den Abstand zwischen zwei Punkten im mehrdimensionalen Raum zu berechnen. In diesem Fall wurde die „*Manhattan Formel*“ verwendet, um den Abstand zwischen beliebigen, mehrdimensionalen Punkten zu berechnen.

Angenommen, es gäbe zwei Punkte G_1 und G_2 wobei jeder Punkt ein n -dimensionaler Vektor ist, mit $G_1 = (X_1, X_2, \dots, X_n)$ und $G_2 = (Y_1, Y_2, \dots, Y_n)$. Die

Formel 6 zeigt wie der Abstand $D(G_1, G_2)$ mit den Manhattan Formel berechnet wird.

Formel 6: Manhattan Formel (22)

$$D(G_1, G_2) = \sum_i X_i - Y_i$$

6.2 Decision Tree

Ein Decision Tree (abgekürzt Dtree), auf Deutsch Entscheidungsbaum, ist einer der beliebtesten maschinellen Lernalgorithmen, welcher sowohl für die Klassifikations-, als auch für Regressionsprobleme verwendet wird (26). Dieser Algorithmus beschreibt einen Baum, in dem jeder Knoten ein Merkmal, jeder Zweig eine Entscheidung und jedes Blatt ein Ergebnis darstellt. Um für die Klassifizierung einen Datensatz vorherzusagen, wird an der Wurzel angefangen. Das Root-Merkmal wird mit dem Merkmal des Datensatzes verglichen. Die daraus entstandene Entscheidung führt zu weiteren Knoten, bis ein Blattknoten mit der vorhergesagten Aktivitätsklasse erreicht wird. (5)

Eine Schwierigkeit bei dem Dtree-Algorithmus ist die Positionierung der Merkmale an den Knoten, besonders wenn der Datensatz aus vielen Merkmalen besteht. Es ist ein komplizierter Schritt, zu entscheiden, welches Merkmal an der Wurzel oder auf verschiedenen Ebenen des Baumes als Knoten platziert werden soll. Durch die zufällige Auswahl eines beliebigen Knotens als Wurzel kann das Problem nicht gelöst werden. Auch wenn ein zufälliger Ansatz befolgt wird, kann es zu schlechten Ergebnissen mit geringer Genauigkeit führen.

Um dieses Auswahlproblem zu lösen, gibt es einige Lösungen. Es kann zum Beispiel ein Kriterium wie Informationsgewinn oder Gini Index (bzw. Gini Koeffizient) verwendet werden. Diese Kriterien berechnen Werte für jedes Merkmal. Die Werte werden sortiert und Merkmale werden in dem Baum angeordnet, indem die Reihenfolge eingehalten wird. Für diese Arbeit wurde als Teilungskriterium der Gini Index verwendet. Dies ist eine Metrik, um zu messen, wie oft ein zufällig ausgewähltes Element falsch identifiziert wird. Hier wird das Attribut mit dem niedrigeren Gini Index bevorzugt.

Klassifizierung

Es seien die Klassen $J = 1, \dots, k$ gegeben und P_j die relative Häufigkeit der Elemente in der Menge, die zu Klasse J gehören. Der Gini Index G_i wird mit der Formel 7 berechnet

Formel 7: Berechnung der „Gini - Index“ (2)

$$G_i = 1 - \sum_{j=1}^k P_j^2$$

6.3 Implementierung

Um die Anzahl der für die Klassifizierung verwendeten Eigenschaften zu reduzieren, wurden die im Kapitel 5 genannten Merkmale aus den gesammelten Daten extrahiert. Die Merkmale wurden jeweils für jede Achse des Datenpunkts von Jedem Zeitfenster berechnet.

Der Ablauf der reinen Klassifizierung besteht aus mehreren Schritten. Als erstes wurden die Merkmalsdaten normalisiert. Diese verhindert, dass Merkmale mit größerem Definitionsbereich nicht ungleich stärker in die Abstandsmessung gehen, als die Merkmale mit einem kleinen Definitionsbereich. Es wurde die „*Minimum-Maximum*“ Normalisierung durchgeführt. Dabei werden die Daten zwischen 0 und 1 skaliert. Das heißt der Mindestwert wird auf 0 und der Höchstwert auf 1 skaliert.

Nach der Normalisierung wurde ein Modell aus den Merkmalsdaten erstellt. Dafür wurde die Methode „*train_test_split ()*“ von „*Sklearn*“ verwendet. Diese Methode dient zum Trennen der Daten bzw. Merkmalsdaten in *Trainings-* und *Testdaten*. Es wurden in dieser Arbeit sechs *Trainingsdaten / Testdaten* Aufteilungsmodelle (90% / 10%, 80% / 20%, 70% / 30%, 60% / 40%, 50% / 50%, 40% / 60%) getestet. Die Modelle wurden für die beiden Klassifizierungsalgorithmen verwendet. Weiter wurden die Maschinellen lernen Algorithmus an die Trainingsdaten angepasst. Für die K-NN wurde die Funktion „*KNNclasi ()*“ und für den Entscheidungsbaum die Funktion „*DTreeclasi ()*“ geschrieben. Bei der Klassifizierung der K-NN Algorithmus wurde 3 verschiedene k-Werte (2, 3, 5) getestet. Die Abbildung 6-2 und Abbildung 6-3 zeigen jeweils die erstellten Funktionen für die beiden Algorithmen. Diese Funktionen liefern die Erkennungsrate und die Konfusionsmatrix zurück.

```
def KNNcalsi(Filename, K, test_size):  
    """ classification using K-NN  
  
        input  
        Filename: Name of the File  
    """  
    # get training- and testdata  
    data = set_training_test_data(Filename, test_size)  
    x_train = data[0]  
    x_test = data[1]  
    y_train = data[2]  
    y_test = data[3]  
  
    #get the best k  
    k = int (K)  
    # fit KNN algorithm on training data  
    knn = KNeighborsClassifier(n_neighbors = k, p=1)  
    knn.fit(x_train, y_train)  
  
    #clasi score  
    y_pred = knn.predict(x_test)  
    score = accuracy_score(y_pred, y_test)  
  
    # Matrix  
    Matrix = confusion_matrix(y_test, y_pred)  
  
    return score, Matrix, k
```

Abbildung 6-2: Funktion für Klassifizierung mit dem K-NN Algorithmus

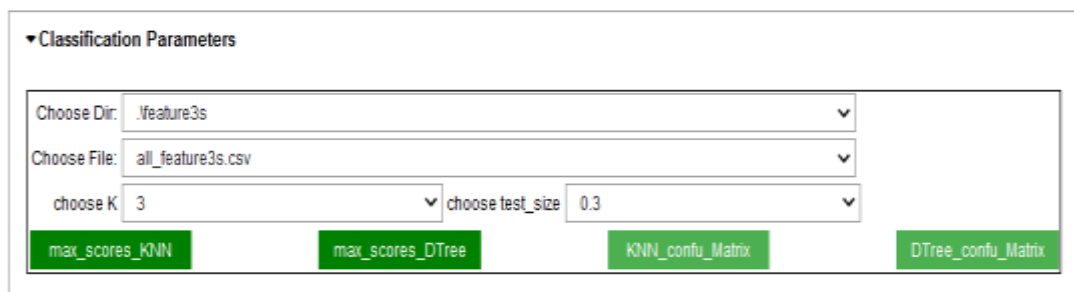
```
def DTreecalsi(Filename, test_size):  
    """ classification using Dtree  
  
        input  
        Filename: Name of the File  
    """  
    # get training- and testdata  
    data = set_training_test_data(Filename, test_size)  
    x_train = data[0]  
    x_test = data[1]  
    y_train = data[2]  
    y_test = data[3]  
  
    # fit KNN algorithm on training data  
    DTree = DecisionTreeClassifier(criterion = 'gini', random_state = 42)  
    DTree.fit(x_train, y_train)  
  
    #clasi score  
    y_pred = DTree.predict(x_test)  
    score = accuracy_score(y_pred, y_test)  
  
    # get confusion matrix  
    Matrix = confusion_matrix(y_test, y_pred)  
  
    return score, Matrix
```

Abbildung 6-3: Funktion für Klassifizierung mit dem Dtree Algorithmus

Klassifizierung

Um die Klassifizierung zu erleichtern wurde auch in das Notebook „*Merge_Feature_classification*“ ein Tab mit dem Namen „*classification*“ erstellt. Mit dieser Tab können passende CSV-Datei unter „*Choose File*“ ausgewählt werden. Unter „*choose K*“ kann der K-Wert für den K-NN Algorithmus gewählt und unter „*choose test_size*“ das gewünschte Modell.

Nach Eingabe aller Parameter können mit dem Klick auf die Knöpfe „*max_score_KNN*“ und „*max_score_DTree*“ die Erkennungsrate von jeweils dem K-NN und Entscheidungsbaum Algorithmus zurückgegeben werden. Außerdem können auch die Konfusionsmatrizen mit Hilfe von den Knöpfen „*Knn_confu_matrix*“ und „*DTree_confu_matrix*“ geplottet und gespeichert werden.



▼Classification Parameters

Choose Dir: .feature3s

Choose File: all_feature3s.csv

choose K: 3 choose test_size: 0.3

max_scores_KNN max_scores_DTree KNN_confu_Matrix DTree_confu_Matrix

Abbildung 6-4: Auswahl Menü Tab „*classification*“

7 Ergebnisse und Bewertung der Klassifizierung

In diese Kapitel werden die Klassifizierungsleitungen von den beiden Klassifizierungsalgorithmen dargestellt und bewertet.

7.1 Ergebnisse

Die Tabelle 7-1 stellt die erzielte Erkennungsrate von beiden Algorithmen bei drei Sekunden Fensterung. Diese wurden nach Modellen und K-Werte für die K-NN Algorithmus geordnet. Es wurden Merkmale auf drei verschiedene Fenstergrößen extrahiert. Alle drei Fenstergrößen haben gute Ergebnisse geliefert. Jedoch liefert die Fenstergröße von drei Sekunden die besten Ergebnisse. Zu den sechs Modellen wurden bei 70% Trainings- und 30% Testdaten gute und konstante Ergebnisse sowohl bei K-NN als auch bei Dtree erzielt. Bei der K-NN Algorithmus lieferten alle drei K-Werte gute Ergebnisse, aber $K = 3$ war ein Stück besser.

Die Ergebnisse für die zwei anderen Fenster befindet sich in Anhang C (Tabelle C-1 und Tabelle C-2).

Tabelle 7-1: Erkennungsrate von beiden Algorithmen geordnet nach Aufteilungsmodell und K-Werte bei 3s Fensterung

Prozent von Training / Testdaten	Erkennungsrate in Prozent bei 3s Fensterung			
	K-NN			Dtree
	K=2	K=3	K= 5	
90/10	96,1	97,4	97,4	90,9
80/20	96,7	96,7	97,3	92,1
70/30	96,0	96,7	96,6	94,3
60/40	95,7	97,0	97,3	93,4
50/50	95,0	96,8	95,8	93,7
40/60	94,5	96,2	95,6	92,7

7.2 Bewertung der Ergebnisse

Aufgrund der oben gegebenen Ergebnisse wurden weiter nur die drei-Sekunden-Fensterung, das Modell 70/30 und den K-Wert 3 betrachtet und bewertet. Die Ergebnisse werden in Bezug auf die Algorithmen und auf die Aktivitätsgruppen

7.2.1 K-Nearest Neighbor

Insgesamt bietet die K-NN eine sehr gute Erkennungsrate, wie bei früheren Arbeiten (7), (21). Es wurde eine Erkennungsrate von mehr als 96 % erreicht. Dabei lieferten die Rehabilitation-Aktivitäten: halbe Kniebeugung, Beinpendeln, Kniebeugenstrecken eine Erkennungsrate von mehr als 99 % zurück. Im Gegensatz dazu liefern die alltäglichen Aktivitäten Gehen, Laufen, Treppen hinaufsteigen und Treppe heruntersteigen nur 93 %.

Die Abbildung 7-1 stellt die Konfusionsmatrix bei dem K-NN Algorithmus dar. Diese Konfusionsmatrix ermöglicht es, zu sehen, wieviel Testdaten pro Aktivität gab und wie wurden die klassifiziert. Hier ist ein Beispiel: laut der Konfusionsmatrix wurde 75 Daten von Gehen getestet und 74 wurden als tatsächlich als Gehen erkannt und einen davon als Beinpendeln.

Ergebnisse und Bewertung der Klassifizierung

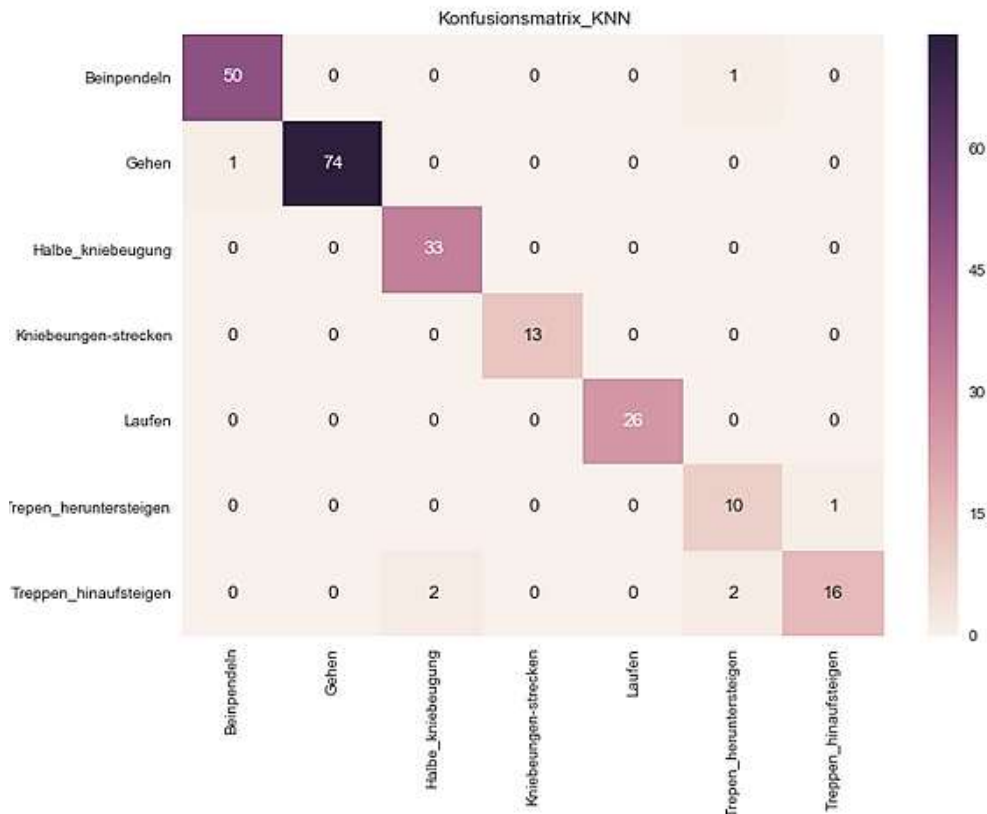


Abbildung 7-1: Konfusionsmatrix bei K-NN mit $K = 3$, Modell „70 / 30“ und 3s Fensterung

7.2.2 Decision Tree

die Rehabilitation-Aktivitäten ergattern auch bei dem Entscheidungsbaum die besten Ergebnisse. Sie haben fast 10% mehr Erkennungsrate als die alltäglichen Aktivitäten. Das heißt 100 % bei den Rehabilitation-Übungen und 90 % bei den alltäglichen Aktivitäten.

Die Abbildung 7-2 stellt die Konfusionsmatrix von dem Entscheidungsbaum. Mit Hilfe der Konfusionsmatrix wurde festgestellt, dass die Aktivität Treppen heruntersteigen meisten Treppen hinaufsteigen und umgekehrt.

Ergebnisse und Bewertung der Klassifizierung

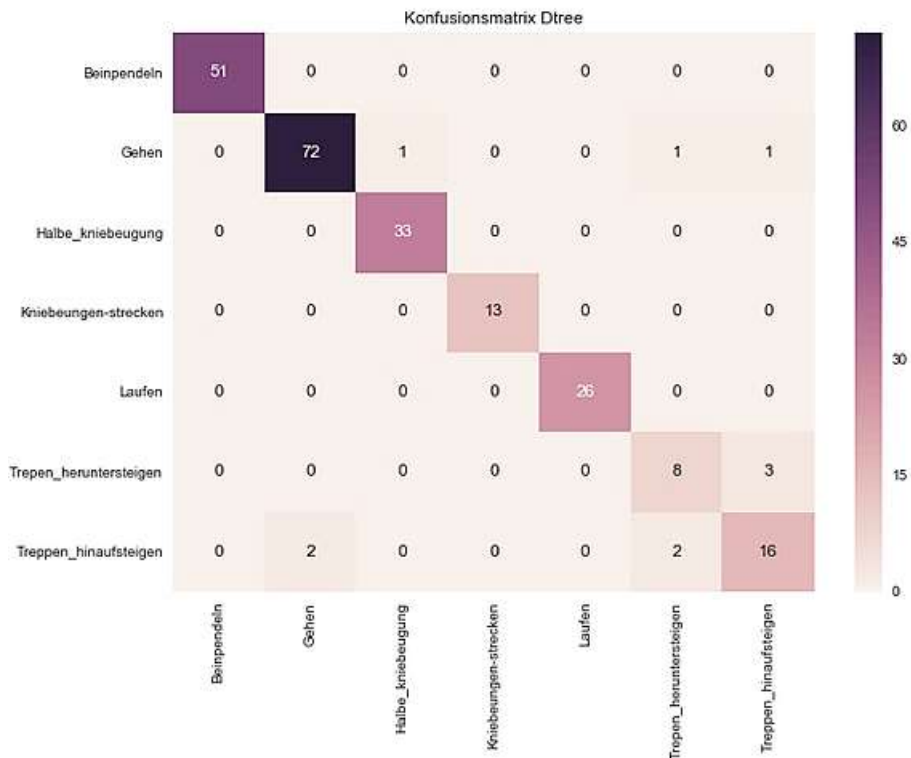


Abbildung 7-2: Konfusionsmatrix bei Dtree mit dem Modell „70 / 30“ und 3s Fensterung

7.2.3 Bewertung der Ergebnisse in Bezug auf die Aktivitätsgruppen

Die Ergebnisse der Aktivitätserkennung bei zwei Probanden zeigten für die meisten Aktivitäten eine hohe Genauigkeit. Die Gesamtgenauigkeit des Systems betrug bei beiden Klassifizierung Algorithmen etwa 95%.

Für alle Klassifikationsverfahren gilt, dass Fehlklassifikationen fast ausschließlich zwischen den einzelnen alltäglichen Bewegungsmustern (Gehen, Laufen, Treppen hinaufsteigen und Treppe heruntersteigen) Aktivitäten auftreten, d.h. Rehabilitation-Aktivitäten werden nur selten als alltäglich klassifiziert und umgekehrt. Trotz der Tatsache, dass die Erkennungsraten für die Rehabilitation-Aktivitäten durchweg besser als die der alltäglichen Aktivitäten sind, werden hier beide getrennt betrachtet:

➤ Reha Aktivitäten

Die Erkennungsraten von Kniebeugen-strecken und Halbe Kniebeugung sind bei K-NN als auch bei Dtree gleich, da sie bei den beiden Algorithmen eine Erkennung von 100% geliefert haben. Die Aktivität Beinpendeln liefert auch bei der K-NN Algorithmen 2% weniger Erkennungsrate als bei Dtree Algorithmus. Schließlich liefern sowohl die K-NN als auch die Dtree bei Rehabilitation Aktivitäten die besten Ergebnisse.

➤ **Periodische Aktivitäten**

Die Erkennungsraten für Laufen sind durchweg bei den beiden Algorithmen sehr gut und unterscheiden sich aber auch am deutlichsten von den restlichen Aktivitäten. Treppen hinaufsteigen hat unter den alltäglichen Aktivitäten die niedrigsten Erkennungsraten. Die wurde meistens als Treppen-Heruntersteigen oder Gehen und umgekehrt klassifiziert. Der Grund dafür könnte in der Tatsache liegen, dass diese Ereignisse nicht isoliert ausgeführt wurden, weil bei Treppen hinauf- und heruntersteigen immer ein Abschnitt von Gehen dazwischen gab oder weil nach den Treppen hinaufsteigen direkt runtergestiegen wurde. Es war sehr schwer diese Daten zu trennen. Nichtsdestotrotz liefert der K-NN für die alltäglichen Aktivitäten deutlich beste Erkennungsrate, etwa 3% höher im Vergleich zu dem Entscheidungsbaum.

8 Fazit und Ausblick

In dieser Arbeit wurden zwei maschinelle Lernalgorithmen zur Klassifizierung von Bewegungen anhand Beschleunigungsdaten vorgestellt. Zum einen wurde der K-NN Algorithmus verwendet und zum anderen der häufige eingesetzte Entscheidungsbaum Algorithmus benutzt. Zur Datenakquisition wurde der Beschleunigungssensor Axivity AX3 verwendet und mit dessen Hilfe wurden 112955 Daten gesammelt. Es wurde, unter Zuhilfenahme von Python-Modulen, Notebooks in Jupyter entwickelt. Mit diesen Notebooks können die Messdaten und ihr gleitender Mittelwert und Standardabweichungen angezeigt werden. Die Daten können in Aktivitätsbereiche aufgeteilt oder zusammengefügt werden. Außerdem können die Merkmale für die Klassifizierung aus den Daten gerechnet werden und die Ergebnisse in einer CSV-Datei gespeichert werden.

Für die Klassifizierung wurden drei unterschiedlichen Fenstergrößen betrachtet. Daraus ergab sich, dass die Fenstergröße von drei Sekunden die Optimale für die in diese Arbeit verwendete Daten ist. Zur Bildung des Klassifizierungsmodells wurden ebenfalls mehrere Modelle gebildet, genauer gesagt sechs Modelle und das Modell 70 % Trainingsdaten und 30% Testdaten lieferte die besten Ergebnisse.

Obwohl die Aktivitäten von verschiedenen Personen mit unterschiedlicher Geschwindigkeit und Art durchgeführt wurden, wurde festgestellt, dass das System verschiedene Aktivitäten mit hoher Genauigkeit klassifizieren kann. Die Erkennungsrate des Systems beträgt 95%. Sowohl der K-NN Algorithmus mit $K=3$ Nachbarn als auch der Entscheidungsbaum Algorithmus lieferten eine hohe Genauigkeit und vergleichbare Ergebnisse für ihre jeweiligen Merkmalssätze. Von den beiden Algorithmen erzeugt der K-NN Algorithmus Leistung bei den Alltäglichen Aktivitäten und der Entscheidungsbaum-Algorithmus Leistung bei den Rehabilitation Übungen. Insgesamt liefert die K-NN-Klassifikator die besten Ergebnisse von zwar 96 %.

Für zukünftige Arbeiten wird es interessant sein, die Ergebnisse für eine vielfältigere Probandengruppe zu analysieren. Darüber hinaus könnte das System verwendet werden, um weitere Untersuchungen zu den unterschiedlichen Aktivitäten zwischen gesunden und operierten Probanden durchzuführen. Da die Merkmalsvektoren und Klassifizierungsalgorithmen bekannt sind, wäre es für die Zukunft eine interessante Aufgabe, zu bewerten, ob eine Fensterung mit 50% Überlappung die optimale ist.

Fazit und Ausblick

Außerdem können neben dem Beschleunigungssensor auch andere Sensoren wie die Gyroskop verwendet werden. Ein Gyroskop Sensor ist ein Beschleunigungs- oder Lagesensor, der auf kleinste Beschleunigungen, Drehbewegungen oder Lageänderungen reagiert. Darüber hinaus können anderen Positionen und Richtungen des Sensor Auch betrachtet werden.

Literaturverzeichnis

1. Saisakul Chernbumroong, Anthony S. Atkins, and Hongnian Yu. *Activity classification using a single wrist-worn accelerometer*. s.l. : Reseachgate, 2014.
2. Györkös, Attila. *Erkennung von Aktivitäten mit Hilfe mobiler Geräte*. . Stuttgart : s.n., 2011.
3. Dallas, Piyush Gupta* and Tim. *Feature selection and Activity Recognition System using a single triaxial Acceloremeter* . Texas USA : IEEE, 2014.
4. Intile, Liing Bao und Stephen S. *Activity Recognition from User_Annitated Acceleration Data*. Cambrige, MA : massachusetts Institute of technology , 2004.
5. Bieber, Geral. *Methotik zur mobilen Erfassung Käperlicher Aktivitäten mittels Beschleunigungssensoren*. Rostock : s.n., 2014.
6. Jennifer R. Kwapisz, Gary M. Weiss, Samuel A. Moore. *Activity Recognition Using Cell Phone Accelerometers*. Washington, DC : SensorKDD, 2010.
7. Nishkam R., Nikhil D., Preetham M., Michael L. *Activity Recognition from Accelerometer Data*. s.l. : American association for Artifical Intelligence , 2005 .
8. Bao, Ling. *Physical Activity Recognition from Acceleration*. 2003.
9. Milker, Sven. *Bewegungserkennung mit Smartphones mittels deren Sensoren*. Koblenz-Landau : s.n., 2012.
10. Beschleunigungssendor. [Online] [Zitat vom: 10. 02 2018.] <https://www.omega.de/prodinfo/beschleunigungsmesser-vibrationsaufnehmer.html>.
11. *Practical guide to Accelerometers*. [Online] [Zitat vom: 24. 03 2018.] <http://masters.donntu.org/2009/eltf/kiseliov/library/article5.htm>.
12. McKinney, Wes. *Datenanalyse mit Python / Auswertung von Daten mit Pandas, Numpy und Ipython*. 2015.
13. AISOMA. *die 15 Wichtigsten Python Bibliotheken Datenanalyse und Machinelles Lernen* . [Online] [Zitat vom: 18. 06 2018.] <https://www.aisoma.de/die-15-wichtigsten-python-bibliotheken-fuer-datenanalyse-und-maschinelles-lernen/>.

14. Antonino Ingargiola and contributors. . Jupyter/ Ipython Notebook quick start guide. [Online] 2015. [Zitat vom: 18. 06 2018.] https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#what-is-the-jupyter-notebook.
15. Axivity AX3. [Online] [Zitat vom: 09. 11 2017.] <https://axivity.com/product/ax3>.
16. „Axivity AX3 User Manual". [Online] [Zitat vom: 11. 10 2017.] <http://axivity.com/userguides/ax3/>.
17. Papula, Lothar. *Mathematik für Ingenieure und Naturwissenschaftler Band 1 "Ein Lehr- und Arbeitsbuch für das Grundstudium"*. s.l. : pp. 45, 2009.
18. Lin.Sun1, Daqing Zhang, Bin.Li,Bin.Guo,Shijian Li. *Activity Recognition on an Accelerometer Embedded Mobile Phone with Varying Positions*. 2010.
19. Andrea Mannini, Stephen S. Intille, Mary Rosenberger, Angelo M. Sabatini1, and William. *Activity recognition using a single accelerometer placed at the wrist or ankle pp. 2193–2203*. s.l. : Med Sci Sports Exerc., 2014.
20. Akram Bayat, Marc Pomplun, Duc A. Tran. *A Study on Human Activity Recognition Using Accelerometer Data " The 11th International Conference on Mobile Systems and Pervasive Computing"*. USA : ScienceDirect, 2014.
21. Ferhat Attal, Samer Mohammed , Mariam Dedabrishvili , Faicel Chamroukhi, Latifa Oukhellou , Yacine Amirat. *Physical Human Activity Recognition Using Wearable Sensors*. s.l. : Vittorio M.N. Passaro, 2015.
22. Bao, Ling. *Physical Activity Recognition from Acceleration Data under Semi-Naturalistic Conditions*. 2003.
23. Hain, Johannes. *Lehrstuhl für Mathematik VIII – Statistik* . [Online] Uni Wuerzburg. [Zitat vom: 10. 05 2018.] https://www.uni-wuerzburg.de/fileadmin/10040800/user_upload/hain/SPSS/Abhaengigkeit.pdf.
24. uni Muenchen. Nächste-Nachbarn-Klassifikatoren. [Online] [Zitat vom: 17. 08 2018.] <http://www.dbs.informatik.uni-muenchen.de/Lehre/KDD/WS0304/Skript/kdd-3-klassifikation2.pdf>.
25. Sahak Kaghyan, Hakob Sarukhanyan. *Activity recognition using K-Nearest Neighnor Algorithm on Smartphone with tri-axial Accelerometer " Information Models and Analyses"*. Russland : s.n., 2012.

26. Gupta, Prashant. Decision Trees in Machine Learning. *Towards Data Science*. [Online] [Zitat vom: 15. 08 2018.] <https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052>.
27. Bosch Funktionsprinzip eines Drehratensensors für ESP. [Online] 09. 10 2017. <https://www.youtube.com/watch?v=KZ4Zcc74RHo>.
28. OrthoInfo. [Online] [Zitat vom: 24. 07 2018.] <https://orthoinfo.org/en/recovery/knee-conditioning-program/knee-pdf/>.
29. Niall Twomey, Tom Diethe, Xenofon Fafoutis, Atis Elsts, Ryan McConville , Peter Flach, Ian Craddock. *A Comprehensive Study of Activity Recognition Using Accelerometers*. s.l. : MPDI, 2018. pp. 26.

Anhang

A. Benutzerhandbuch

In diesem Abschnitt wird die Bedienungsanleitung zur Benutzung des Notebooks *Visualisation_interactiveInput.v1.7* ausführlich beschrieben.

Mit dem Notebook können wie der Name schon sagt, Daten visualisiert werden. Darüber hinaus können Schnittpunkte gesetzt werden, um eine Datei in Teildateien zu teilen und anschließend können auch die Dateien geschnitten werden. Außerdem können die Daten durch mathematische Methoden ausgewertet werden.

Dieses Notebook wurde in fünf Tabs gegliedert: „Show Data“, „Show Cutpoint and Split 1 File“, „Show Box and Math operation“, „Show Cutpoint and Split file 2“ und „Show Cutpoint and Split file 3“.

A.1 Show Data

In dem ersten Tab „*Show Data*“ werden Daten mit ihrer dazugehörigen „*Moving Average*“ und „*Moving Standard Deviation*“ visualisiert. Die Funktion ermöglicht die Auswahl eines Ordners und einer Datei des Ordners und verfügt auch über verschiedene Visualisierungs-Optionen.



Abbildung A-1: Tab Show Data - Auswahlmenü

Die folgenden Optionen sind verfügbar:

1. Mit dem Widget „*Choose Dir*“ werden alle Ordner, die in dem vordefinierten Hauptordner beinhaltet sind, aufgelistet. Dennoch wird nur ein Ordner ausgewählt.

A. Benutzerhandbuch

2. Mit dem Widget „**Choose File**“ werden alle Dateien, die in dem ausgewählten Verzeichnis umfasst sind, aufgelistet. Dabei kann nur eine Datei ausgewählt werden.
3. und 4. Mit dem Widget „**Frequency**“ und „**Time**“ wird die **Frequenz**, mit der die Daten aufgenommen wurden, und die **Dauer** der Aufnahme angezeigt. Sie werden automatisch nach der Auswahl der Datei angezeigt.
5. Mit dem Widget „**Time type**“ kann ausgewählt werden, mit welchem Zeittyp gearbeitet werden soll. Bei „**Origin_time**“ wird der Zeittyp in der Spalte „**Time**“ der Datei nicht geändert. D.h., wenn es sich um eine Datei handelt, die ursprünglich in absoluter Zeit oder in Sekunden geht, bleibt die Zeit bestehen. Bei „**Convert_to_seconds**“ wird der Zeittyp der Spalte „**Time**“ in Sekunden umgewandelt, d.h. von absoluter Zeit zu Sekunden. Wenn die Datei jedoch schon in Sekunden ist, wird keine Änderung vorkommen.
6. Mit dem Button „**Show Data**“ wird die Datei geplottet bzw. visualisiert.
7. Mit dem Widget „**Choose Window**“ wird das Fenster für der gleitende Mittelwert und der Standardabweichung gewählt. Alle Werte wurden in Sekunden umgewandelt und werden bei der Berechnung mit der Frequenz multipliziert.
8. Mit dem Button „**Update Plots**“ werden zwei Graphen generiert. Bei dem ersten werden die Datensätze und ihre „Moving Average“ gleichzeitig geplottet und bei dem zweiten wird nur die „Moving Standard Deviation“ angezeigt. Darüber hinaus werden auch die statistischen Werte (Mittelwert, Median, Standardabweichung) der Daten ausgegeben.
9. Mit dem Button „**Show AVG**“ wird nur der gleitende Mittelwert/ Moving Average gezeigt.
10. Mit dem Button „**Show STD**“ wird nur die gleitende Standardabweichung bzw. die Moving Standard Deviation geplottet.

A.2 Show Cutpoints and Split File 1

Diese Funktion ermöglicht, wie bei Show Data auch, eine Auswahl eines Ordners und einer Datei des Ordners. Außerdem können auch Schnittpunkte eingesetzt und angezeigt werden und die Datei kann auch in Teildateien aufgeteilt werden.

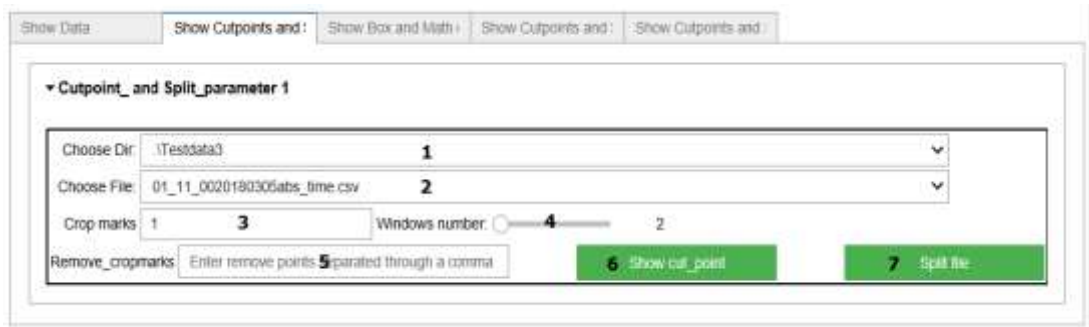


Abbildung A-2: Tab Show Cutpoints and Split File 1 - Auswahlmenü

1. Siehe Abschnitt „*Show Data*“

2. Siehe Abschnitt „*Show Data*“

3. Mit dem Widget „*Crop marks*“ wird die Anzahl der gewünschten Schnittpunkte eingegeben, die später im Graph eingesetzt werden sollen. Die Position dieser Marken hängt von der „*Windows Number*“ ab.

4. Mit dem Widget „*Window number*“ kann gewählt werden, in wie viele Fenster jeder Bereich eingeteilt werden soll, um die Standardabweichung zu berechnen.

Anmerkung: Je größer die Anzahl den Fenstern (*Window_number*), desto genauer werden die Stellen mit der kleineren Standardabweichung betrachtet und somit als Schnittmarken gezeigt.

5. Mit dem Widget „*Remove_croptmarks*“ wird eingegeben, welche Schnittmarken entfernt werden sollen. Die Zahlen müssen mit einem Komma getrennt werden.

6. Mit dem Button „*Show cut_point*“ werden die Datensätze und die Schnittpunkte angezeigt.

7. Mit dem Button „*Split file*“ wird die Teilung der Datei in die Schnittmarke durchgeführt.

A.2 Show Cutpoints and Split File 2

In diesem Tab werden die Schnittmarken auf manueller Weise eingesetzt und angezeigt. Zudem kann die Datei auch in die eingegebenen Schnittmarken geteilt werden.

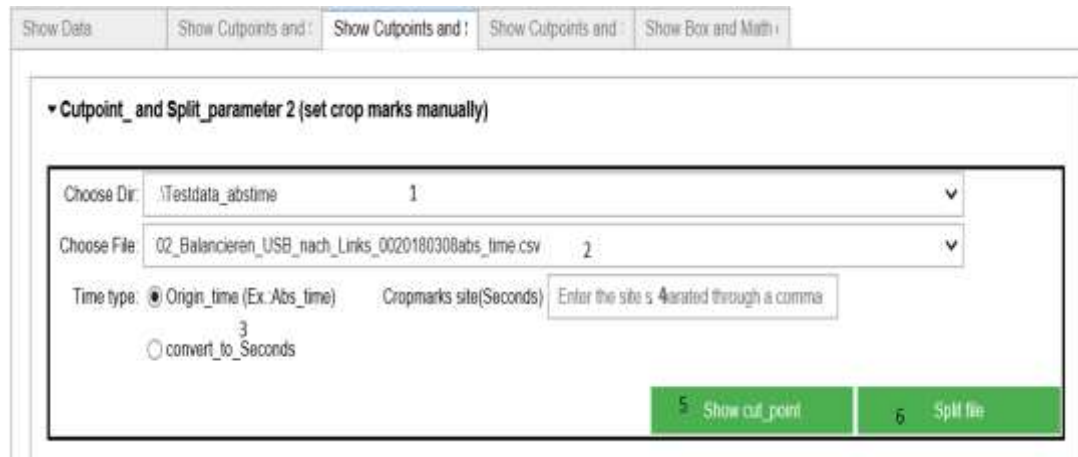


Abbildung A-3: Tab Show Cutpoints and Split File 2 – Auswahlmenü

1. Siehe Abschnitt „*Show Data*“
2. Siehe Abschnitt „*Show Data*“
3. Siehe Abschnitt „*Show Data*“
4. Mit dem Widget „*Cropmarks site (Seconds)*“ werden die Stellen der Schnittpunkte eingegeben. Diese werden in Sekunden eingegeben. Deswegen wäre es vor der Visualisierung sinnvoll, der Zeittyp in Sekunden umzuwandeln. Dabei muss ein Komma zwischen den Zahlen eingesetzt werden.
5. Mit dem Button „*Show cut_point*“ werden die Datensätze und die Schnittpunkte angezeigt.

Nachdem die Schnittmarken eingegeben wurden, wird der Graph beim Betätigen des Buttons in Bereiche gelistet.

6. Mit dem Button „*Split file*“ wird die Teilung der Datei in die Schnittmarken durchgeführt.

A.4 Show Cutpoints and Split File 3

In diesem Tab werden die Schnittmarken durch die Moving Standard Deviation der Datei bestimmt. Zudem kann die Datei auch in die eingegebenen Schnittmarken geteilt werden.

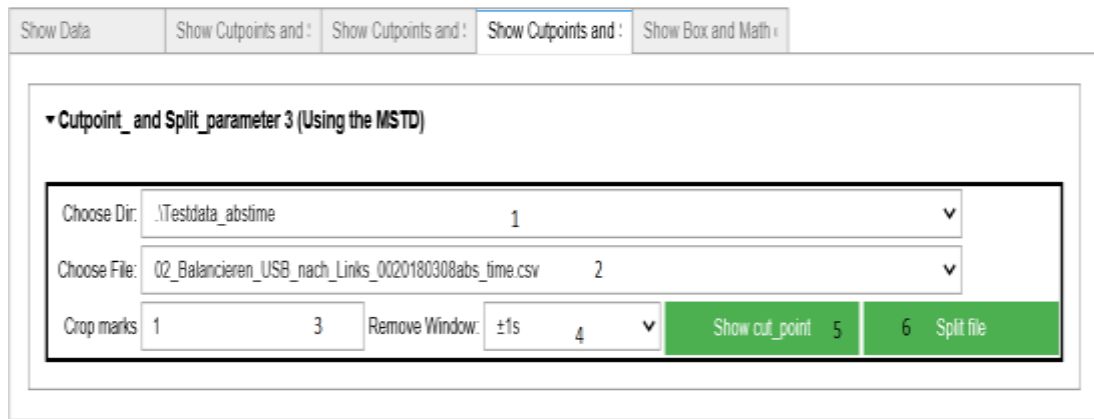


Abbildung A-4: Tab Show Cutpoints and Split File 3 – Auswahlmenu

1. Siehe Abschnitt „*Show Data*“
2. Siehe Abschnitt „*Show Data*“
3. Mit dem Widget „*Crop marks*“ wird die Anzahl der gewünschten Schnittpunkte eingegeben, die später im Graph eingesetzt werden sollen. Die Position dieser Marken hängt von der „*Tolerance Window*“ ab.
4. Mit dem Widget „*Remove Window*“ wird das Intervall festgelegt, welches nach jedem Bestimmen des Minimums in der Datei abgezogen wird, und zwar so lange, bis alle Schnittpunkte bestimmt wurden.
5. Mit dem Button „*Show cut_point*“ werden die Datensätze und die Schnittpunkte angezeigt. Nachdem die Schnittmarken eingesetzt wurden, wird der Graph durch das Betätigen des Buttons in Bereiche gelistet.
6. Mit dem Button „*Split file*“ wird die Datei in Schnittpunkte geteilt.

A.5 Show Box and Math operation

In diesem Tab können sowohl kleine Bereiche bzw. jede darin beinhaltende Aktivität separat geplottet werden, als auch einige mathematische Operationen bezüglich dieser Aktivitäten durchgeführt werden.

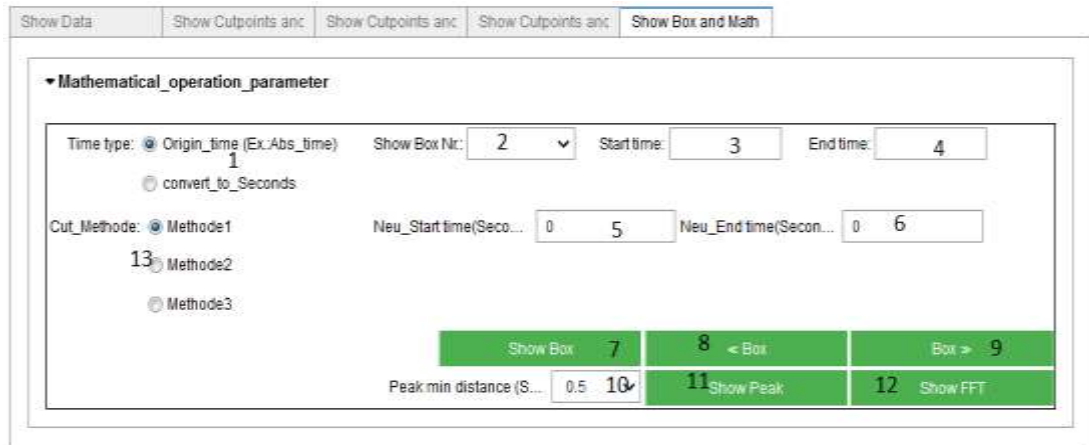


Abbildung A-5: Tab Show Box and Math operation - Auswahlmenü

1. Siehe Abschnitt „*Show Data*“
2. Mit dem Widget „*Show Box Nr.*“ werden alle aufgelisteten Bereiche im Graph durchnummeriert. Diese kommen von dem Tab von Show „*Cutpoints and Split File 1*“, in dem die Anzahl der Schnittpunkte eingegeben wurde.
3. und 4. Mit den Widgets „*Start time*“ und „*End time*“ werden die Anfangszeit und die Endzeit der ausgewählten Box angezeigt.
5. und 6. Mit den Widgets „*Neu_Start time (Seconds)*“ und „*Neu_End time (Seconds)*“ können die Anfangszeit und/oder die Endzeit der Box erneut definiert werden.
7. Mit dem Button „*Show Box*“ wird die gewählte Box geplottet.
8. Mit dem Button „*<<Box*“ wird die vorherige Box angezeigt.
9. Mit dem Button „*Box>>*“ wird die nächste Box dargestellt.
10. Mit dem Widget „*Peak min distance (sec)*“ wird der minimale Abstand zwischen den Spitzen des Graphen definiert.
11. Mit dem Button „*Show Peak*“ wird die Peak Detektion in der Box durchgeführt und geplottet.
12. Mit dem Button „*Show FFT*“ wird die Fast Fourier Transformation ausgeführt.
13. Mit dem Widget „*Cut Methode*“ wird die Schnittmethode ausgewählt

B. Datei Teilen

B.1 Bessere Schnittpunkte Algorithmus der ersten Variante

In folgende Abbildung wird ein Algorithmus zur Berechnung der besseren Schnittpunkte innerhalb einer Datei dargestellt.

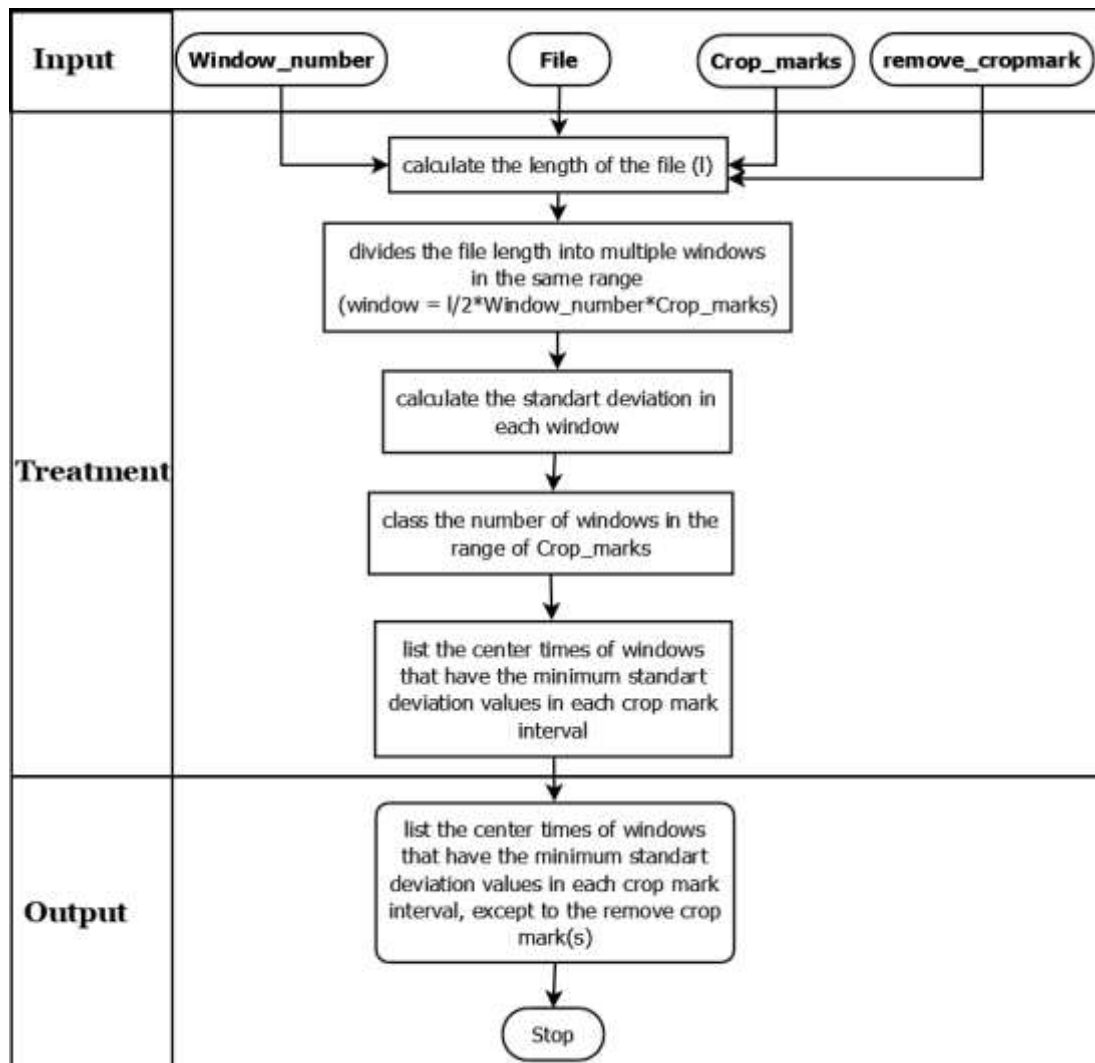


Abbildung B-1: Algorithmus für das Methode1

In folgenden werden ausführlich die Schritte des Algorithmus beschrieben:

- Als erste werden die unterschiedlichen Parameter eingegeben

Als Beispiel werden *5* als *Crop_marks*, *4* als *Window_number* und nichts als *remove_croptmark* genommen.

- Zweitens wird die Länge der Datei durch die Anzahl von Schnittpunkten dividiert (siehe Abbildung B-2), und die als Anfang Schnittpunkte betrachtet werden.

B. Datei Teilen

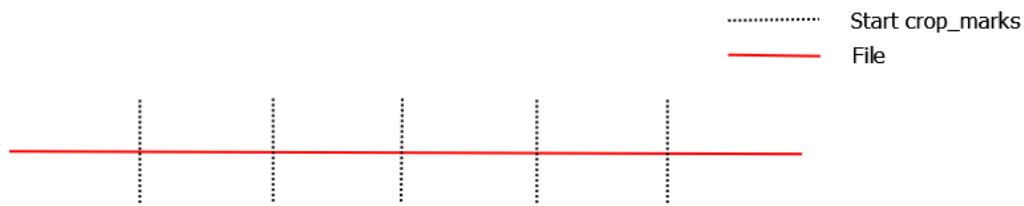


Abbildung B-2: Der erste Schritt des Algorithmus

- Drittens werden jene Blöcke noch mal durch die Anzahl der Fernstern dividiert.

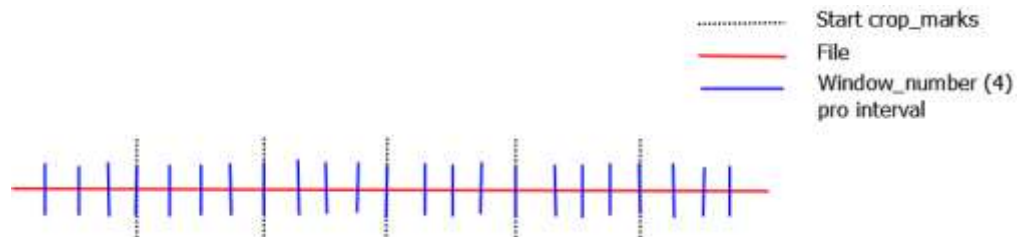


Abbildung B-3: Der zweite Schritt des Algorithmus

- Viertens wird die Standardabweichung in jedem kleinen neuen Block berechnet und in jedem zweiten größten Block werden die Bereiche, in denen die Standardabweichung am kleinsten ist, ausgewählt/markiert.

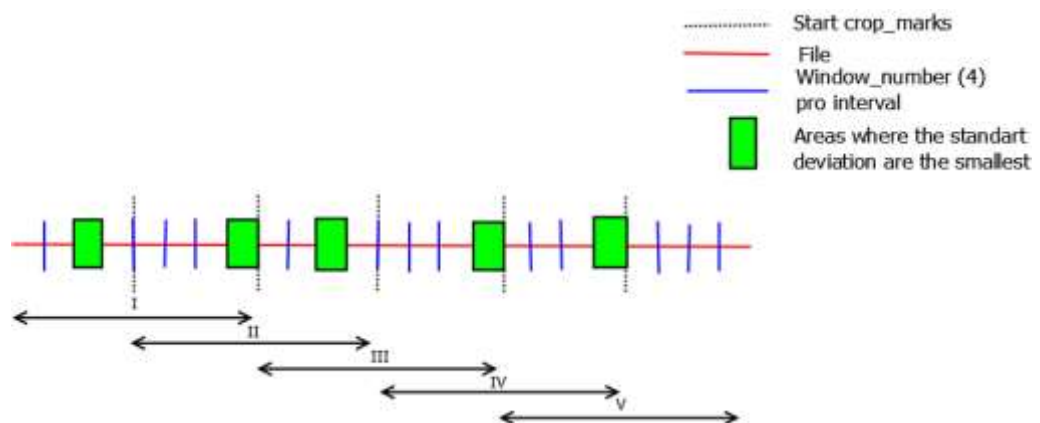


Abbildung B-4: Der dritte Schritt des Algorithmus

Je mehr die Anzahl der Fernstern (Window_number) größer wird, desto genauer wird die Stellen mit der kleineren Standardabweichung betrachtet.

- Schließlich wird die Mitte der markierten Bereiche als End Schnittpunkte dargestellt. (siehe Abbildung)

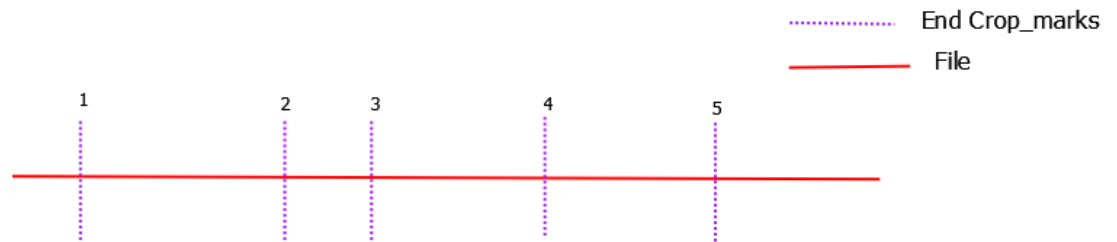


Abbildung B-4: Der vierte Schritt des Algorithmus

Damit das Programm fehlerlos/die richtigen Stellen ergeben kann, müssen einige Maße berücksichtigt werden, und zwar:

- Sollten die aufgenommenen Aktivitäten fast gleich lang sein.
- Sollte zwischen die Aktivitäten eine Ruhe-Bereich liegen lassen, damit das Programm als bessere Schnittpunkte erkennen kann.

Sonst gibt es die Möglichkeit mehrere Punkte (crop_marks) einzugeben, und nach der Visualisierung, die unnötige Schnittpunkte mit dem *remove_cropmark* Parameter zu entfernen. Also wenn wir die Schnittpunkte 2 und 4 entfernen wollen, wird nur 2,4 in *remove_cropmark* Parameter eingegeben. Danach ändert sich der Graph wie wir auf der folgenden Abbildung sehen können.

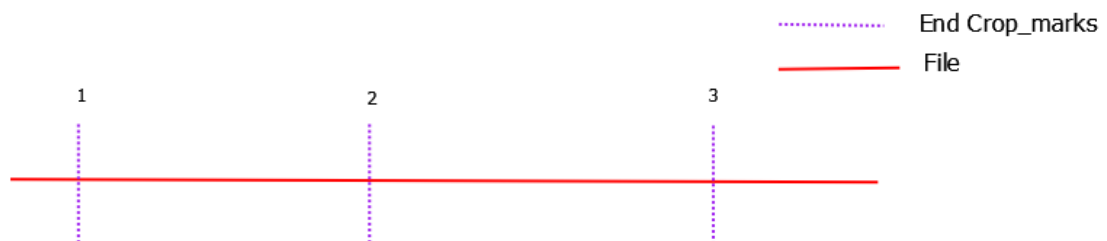


Abbildung B-5: Entfernung der Schnittpunkte 2 und 4

B.2 Bessere Schnittpunkte Algorithmus der dritte Variante

In der folgenden Abbildung wird der Algorithmus zur Berechnung der besseren Schnittpunkte für die dritte Variante dargestellt.

B. Datei Teilen

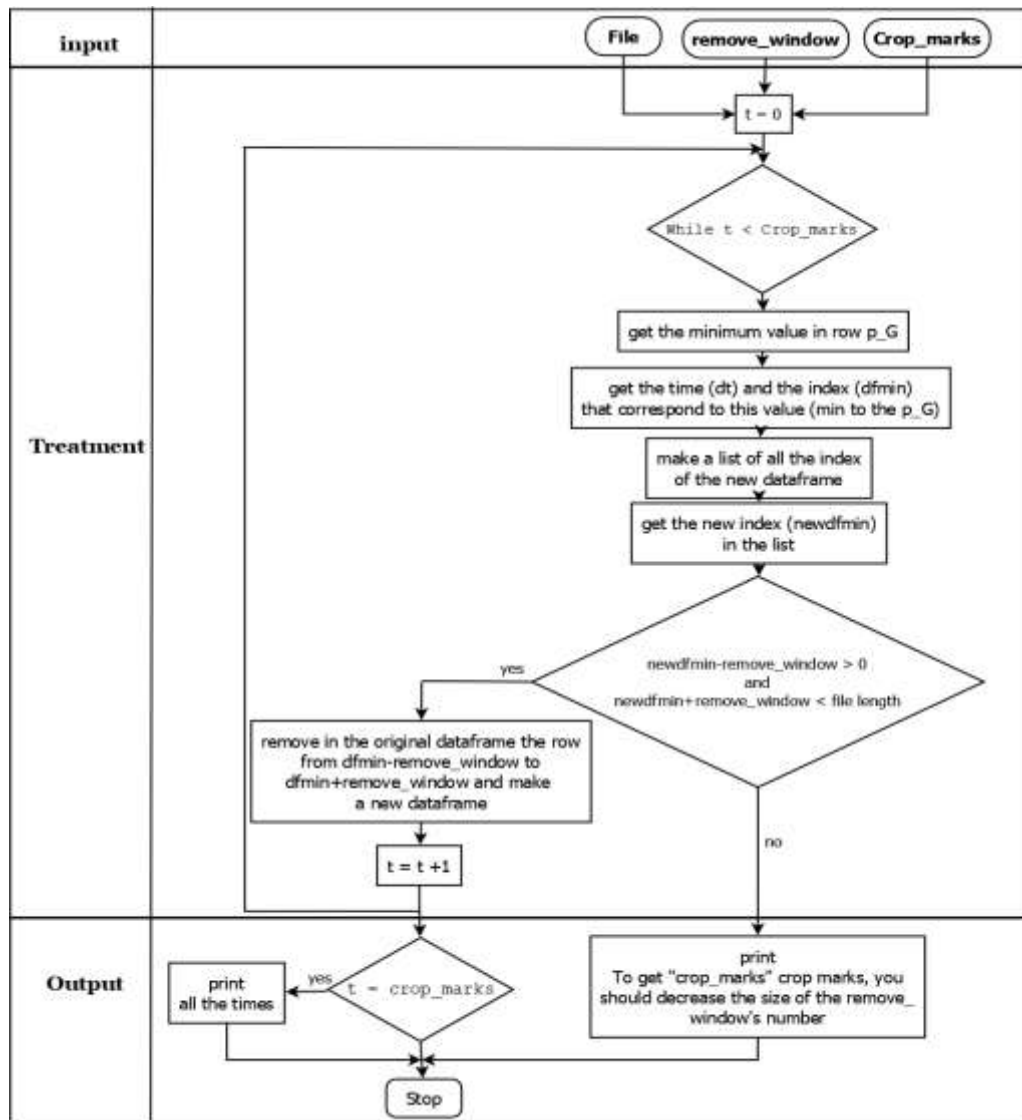


Abbildung B-6: Algorithmus für die Methode 3

In diesem Abschnitt wird genauer an der Funktionsweise des Algorithmus eingegangen.

- Als erste werden die unterschiedlichen Parameter eingetragen siehe Abbildung 17. Als Beispiel nehmen wir eine Datei mit 60 Sekunden und geben **drei** als *Crop_marks* und **±10s** als *remove_window* ein.
- Das Programm wird der minimale Wert der *p_G*-Achse bestimmen und als erste Schnittpunkte für die gesamte Datei anzeigen.

B. Datei Teilen

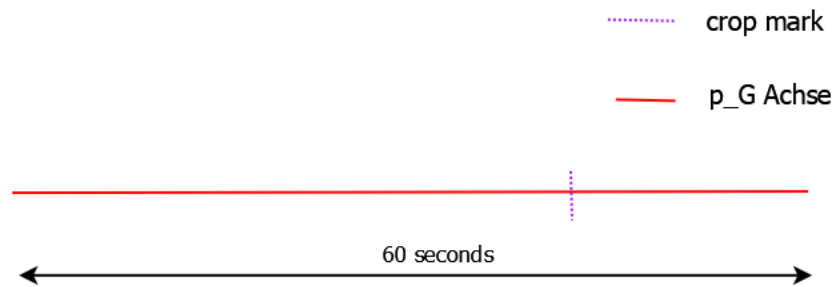


Abbildung B-7: Der erste Schritt des Algorithmus

- Dann das Programm wird in der Stelle des minimalen Wertes eine Sperrfläche von $\pm 10s$ (als *gesamte Länge 20 Sekunden*) Abbildung 18 aus der Datei subtrahieren. Daraus entsteht eine neue Datei mit einer Länge von 40 Sekunden.

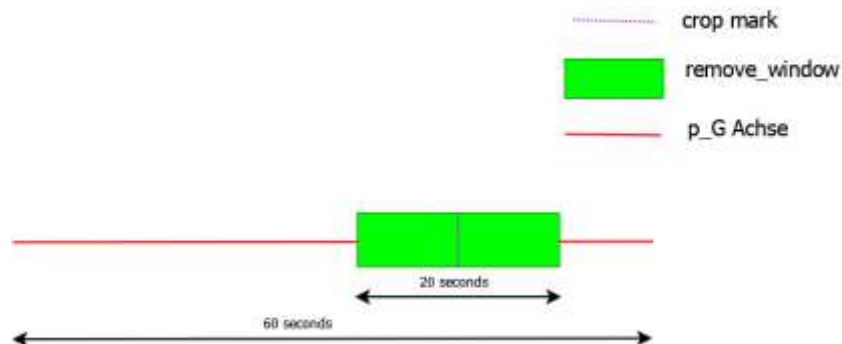


Abbildung B-8: Schritt 2

Die neue Datei wird so gebaut:

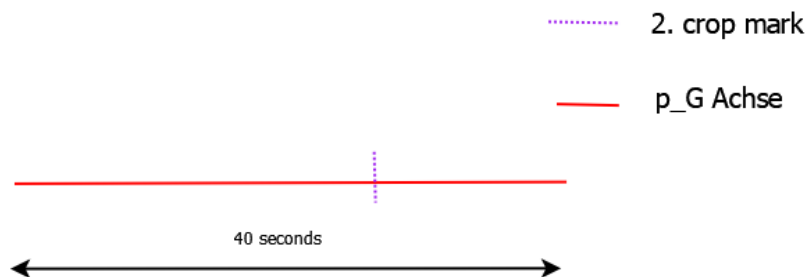


Abbildung B-9: neue Datei 1

- Dann wird wieder aus der neuen Datei, 10 Sekunden vor und nach dem neuen Schnittpunkt abgezogen. Es wird eine neue Datei mit einer Länge von 20 Sekunden erhalten und aus dieser wird die 3. Schnittpunkte berechnet.

B. Datei Teilen

Das wird solange wiederholt, bis alle Schnittpunkte bestimmt worden sind.

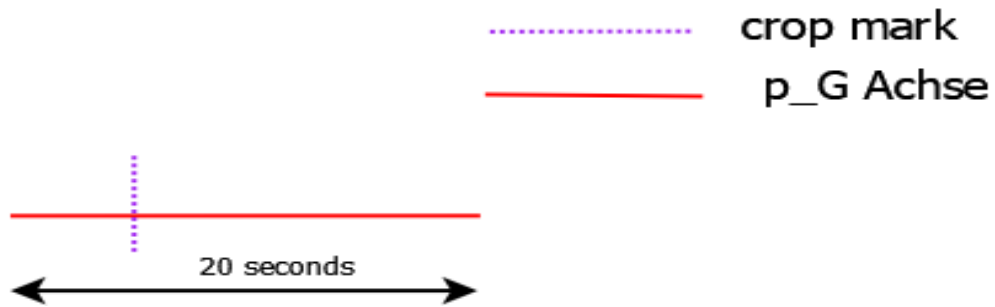


Abbildung B-10: Letzter Schritte

- Schließlich werden die drei Schnittpunkte auf der originalen Datei geplottet.

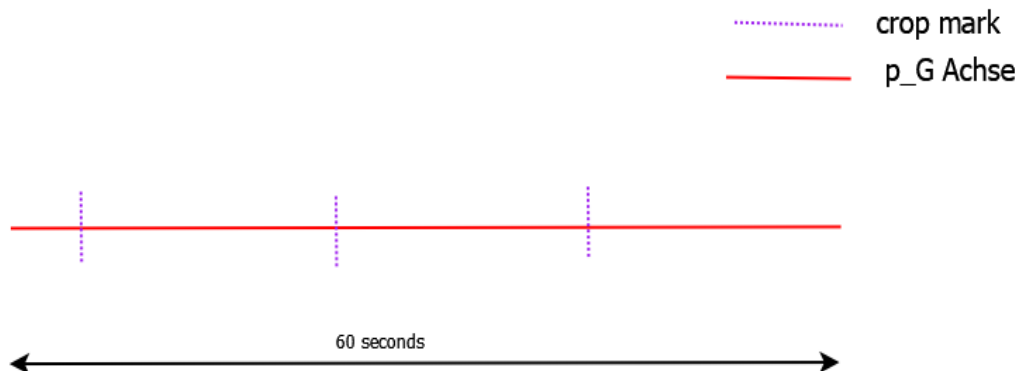


Abbildung B-11: Bessere Schnittpunkte

Damit das Programm fehlerlos funktionieren kann.

Damit das Programm fehlerlos/die richtigen Stellen ergeben kann, müssen einige Maße berücksichtigen werden, und zwar:

- *Sollten die aufgenommenen Aktivitäten fast gleich lang sein.*
- *Sollte zwischen die Aktivitäten eine ruhe Bereich liegen lassen, damit das Programm als bessere Schnittpunkte erkennen kann*

C. Tabellen und Bildern

Tabelle C-1: Erkennungsrate von beide Algorithmen geordnet nach Aufteilungsmodell und K-Werte bei 1s Fensterung

	Erkennungsrate in Prozent bei 1 s Fensterung			
Prozent von Training / Testdaten	K-NN			Dtree
	K=2	K=3	K= 5	
90/10	91,5	92,0	92,8	92,0
80/20	90,8	91,7	91,7	92,4
70/30	90,9	91,2	91,2	91,8
60/40	90,7	91,4	91,1	90,7
50/50	90,4	91,3	90,1	87,7
40/60	90,3	90,3	90,6	88,7

Tabelle C-2: Erkennungsrate von beide Algorithmen geordnet nach Aufteilungsmodell und K-Werte bei 6s Fensterung

	Erkennungsrate in Prozent bei 6s Fensterung			
Prozent von Training / Testdaten	K-NN			Dtree
	K=2	K=3	K= 5	
90/10	91,8	89,1	91,8	91,8
80/20	91,7	90,4	90,4	86,3
70/30	88,1	89,1	86,3	88,1
60/40	91,1	90,4	88,3	88,3
50/50	91,8	92,3	89,1	87,4
40/60	89,4	91,3	90,4	89,1

Aktivitäten Arbeitsblatt

Proband Name: Proband 1

Geschlecht:

Alter:

Datum: 21.08.18

Aktivitäten	Startzeit(HH:MM:SS)	Endzeit(HH:MM:SS)	Notizen
halbe Kniebeugung	13:06	13:07	Knie USB nach oben
Beinpendeln	13:08	13:09	— / —
Kniebeugen-strecken	13:09 49	13:14 37	— / —
Gehen	13:12	13:14	— / —
Laufen	13:15	13:18	— / —
Treppen hinaufsteigen	13:15	13:16	— / —
Treppen heruntersteigen	13:16 27	13:17	— / —

Abbildung C-1: Beispiel eine ausgefügte Aktivität Arbeitsblatt

