Universität Paderborn — Fakultät EIM-I
Fachgebiet Analytic Information Systems and Business Intelligence
Jun.-Prof. Dr. Artus Krohn-Grimberghe

Bachelorarbeit

Erkennung körperlicher Aktivitäten mittels Smartphone- und Smartwatch-Sensoren und Machine Learning

Christian Brüggemann

03.04.2017

Betreut von:
Jun.-Prof. Dr. Artus Krohn-Grimberghe

Bachelorarbeit
am Fachgebiet Analytic Information Systems and Business Intelligence
Jun-.Prof. Dr. Artus Krohn-Grimberghe

Institut für Informatik
Fakultät für Elektrotechnik, Informatik und Mathematik
Universität Paderborn

Vorgelegt von:
Christian Brüggemann
Matrikelnummer: 7004878
Salbeiweg 39
33100 Paderborn

am
03.04.2017

Betreut durch:
Jun.-Prof. Dr. Artus Krohn-Grimberghe
Zweitgutachter:
Prof. Dr. Eyke Hüllermeier

TODO: FG-Logo ändern

Zusammenfassung

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Abstract

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

1.1. Motivation

Smartphones haben im letzten Jahrzehnt an großer Bedeutung gewonnen. TODO: Cite Daneben existieren mittlerweile ergänzend dazu sogenannte *Smartwatches* und *Fitness-Tracker*. Smartwatches sind Armbanduhren, die in der Regel drahtlos mit einem Smartphone verbunden sind und Informationen wie beispielsweise Benachrichtigungen am Handgelenk zugänglich machen. Fitness-Tracker besitzen ähnliche Funktionen, zielen allerdings primär darauf ab, die Fitness und Gesundheit des Nutzers zu fördern, indem Statistiken wie beispielsweise die Schrittanzahl des Nutzers pro Tag gesammelt und grafisch aufbereitet werden. In beiden Geräteformen werden üblicherweise Sensoren verbaut, mit denen sich die Bewegungen des Trägers nachvollziehen lassen.

Mit einigen Fitness-Trackern des Unternehmens *Fitbit* existieren bereits kommerzielle Produkte, die über die reine Sammlung und grafische Aufbereitung von Statistiken hinausgehen: Die Funktion *SmartTrack* erkennt kontinuierliche Aktivitäten mit hoher Bewegung teilweise automatisch, ohne dass der Anwender vorher manuell einstellen muss, welcher Aktivität er in den nächsten Minuten nachgehen wird. TODO: Cite https://help.fitbit.com/articles/en_US/Help_article/1933 . Dies hat den Vorteil, dass der Nutzer sich nicht daran erinnern muss, im Fitness-Tracker die richtige Aktivität einzustellen, um kategorisierte Statistiken zu erhalten.

SmartTrack unterstützt die folgenden Aktivitäten: Gehen, Laufen, Fahrradfahren, Crosstrainer-Training und Schwimmen, sowie zwei allgemeine Kategorien "Sport" (Fußball, Basketball, Tennis, etc.) und "aerobes Training" (Zumba, Tanzen).

Es existieren weitere mögliche Anwendungsgebiete der automatisierten Aktivitätenerkennung. Für Smartphone-Betriebssysteme könnte das Wissen, dass der Anwender gerade Sport treibt, interessant sein, um eingehende Anrufe eines nicht als wichtig markierten Kontaktes zu unterdrücken. Des Weiteren könnte das Forschungsgebiet der "Transportation Mode Recognition" von solchen Methoden profitieren: Soll erkannt werden, mit welchem Verkehrsmittel sich der Nutzer gerade fortbewegt, könnte neben dem Parameter der Geschwindigkeit ebenfalls von Interesse ein, ob mithilfe der Methode die Aktivität "Fahrradfahren" erkannt wird oder nicht. So ließe sich die Fortbewegung mittels eines Mofas von der Fortbewegung mittels eines Fahrrads unterscheiden, was insbesondere für Dienste wie "Google Now" nützlich sein könnte. Diese dienen dem Nutzer als persönlicher Assistent und warnen diesen beispielsweise vor Stau auf einer häufig befahrenen Strecke. Eine solche Warnung könnte entfallen, wenn festgestellt wurde, dass der Nutzer die Strecke nicht mit einem Mofa, sondern mit einem Fahrrad bewältigt.

In der Literatur ist [18] hervorzuheben. Die Autoren vergleichen in ihrem Paper die Genauigkeit der Aktivitätenerkennung eines Smartphones mit der einer Smartwatch und kommen zu dem Schluss, dass die Güte der jeweiligen Erkennung insbesondere von der Aktivität selbst abhängig ist. Es liegt auf der Hand, dass nur mithilfe eines Smartphones beispielsweise eine

Unterscheidung zwischen "Zähneputzen" und "Stehen" schwer möglich ist, während analog dazu nur mithilfe einer Smartwatch die Unterscheidung zwischen "Gehen" und "Fußball schießen" ebenfalls herausfordernd ist. Naheliegend ist daher, eine Kombination beider Datenquellen einzusetzen, um die durchschnittliche Erkennungsrate zu verbessern, ohne eine Beschränkung der erkennbaren Aktivitäten einzuführen.

1.2. Ziele der Arbeit

Evaluiert werden soll ein zu entwickelndes Verfahren, das auf Basis von *maschinellem Lernen* (siehe Definition 1) und eben jenen gesammelten Daten feststellt, welcher Aktivität der Träger der Geräte in bestimmten Zeitintervallen nachgegangen ist. Hierzu wird zunächst eine Software benötigt, welche die synchrone Aufzeichnung von Sensordaten eines Fitness-Trackers oder einer Smartwatch und eines Smartphones ermöglicht.

Es ergibt sich insbesondere die Frage, inwiefern sowohl personalisierte, das heißt nutzerspezifische, als auch unpersonalisierte Modelle durch die Hinzunahme einer weiteren Datenquelle genauer werden.

Um eine Evaluation zu ermöglichen, wird ein Beispieldatensatz benötigt, der durch ein Experiment mit 10 Probanden aufgebaut wird. Orientiert ist diese Zahl an der Anzahl der Probanden in [18], an dessen Experiment 17 Probanden teilgenommen haben. Im Experiment sollen diese voneinander unabhängig mehreren definierten Aktivitäten nachgehen, während parallel dazu Sensordaten mithilfe der entwickelten Software aufgezeichnet werden. Um die Vergleichbarkeit mit den Ergebnissen aus [18] zu gewährleisten, werden die Probanden in diesem Experiment denselben Aktivitäten nachgehen.

Im Folgenden beschränken wir uns auf den uns zugänglichen Fitness-Tracker *Microsoft Band 2*, da dieser im Vergleich zur ebenfalls vorhandenen Smartwatch *Pebble Time* mehr Sensoren besitzt und letztere während der Durchführung des Experimentes nach einem erzwungenen Software-Update falsche Zeitstempel für Sensordaten lieferte.

2. Grundlagen

2.1. Erläuterung der verwendeten Sensoren

Während in der Einleitung nur generisch von "Sensordaten" die Rede war, werden diese nun konkretisiert. Zur Aufnahme werden das Android-Smartphone OnePlus 3 sowie der Fitness-Tracker Microsoft Band 2 verwendet, der mit dem Smartphone via Bluetooth verbunden ist. Beide Geräte besitzen einen Beschleunigungssensor sowie ein Gyroskop. Des Weiteren besitzt das Band unter anderem einen Sensor, der den elektrischen Widerstand der Haut des Trägers misst, sowie einen Hauttemperatursensor.

2.1.1. Beschleunigungssensor

Ein Beschleunigungssensor, auch *Accelerometer* genannt, misst die echte Beschleunigung eines Objektes im Raum je physikalischer Achse (x, y und z). Dies beinhaltet auch die Beschleunigung, die von der Erdgravitation ausgeht: Liegt das Objekt beispielsweise auf dem Boden, erfährt es auf der im rechten Winkel zum Boden stehenden Achse eine Beschleunigung von $g \approx 9.81\ m/s^2$[7][17]. Über diesen Sensor kann demnach die Orientierung des Objektes im Raum bestimmt werden. Abbildung 2.1 illustriert die Sensordaten eines unbewegten Smartphones, dessen Bildschirm parallel zum Boden gehalten wird.
Die Daten dieses Sensors werden von beiden Geräten aufgezeichnet.

2.1.2. Gyroskop

Ein Gyroskop, normalerweise Gyrometer genannt, misst die Rotationsgeschwindigkeit je physikalischer Achse[7]. Demnach bedeutet der Messwert $(x = 0, y = 0, z = 0)$, dass das

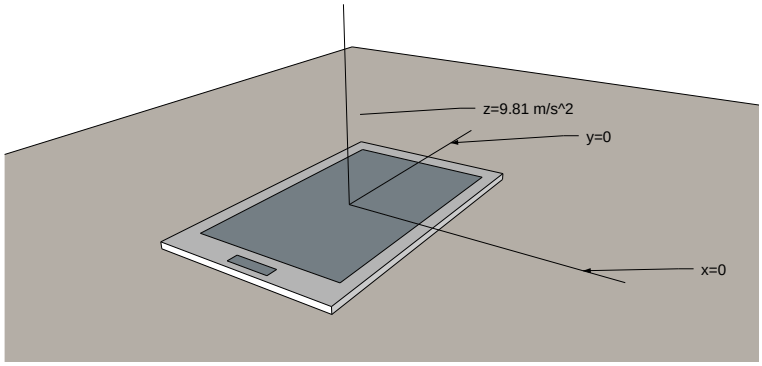

Abbildung 2.1.: Illustration des Beschleunigungssensors

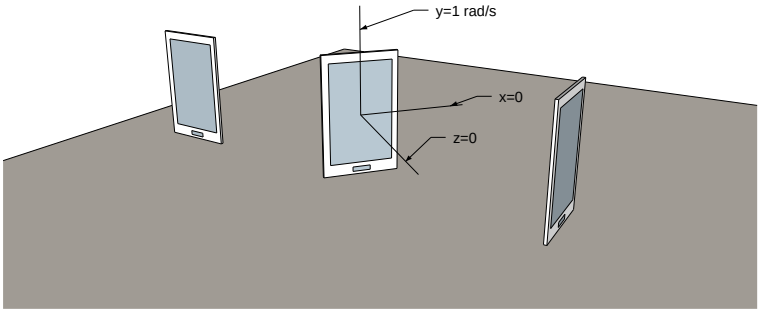

Abbildung 2.2.: Illustration des Gyroskops. Die Achsenbeschriftungen geben die Rotationsgeschwindigkeit um die jeweilige Achse an. Von links nach rechts wird dasselbe Gerät bei fortschreitender Zeit gezeigt.

Objekt relativ zur Erde nicht bewegt wird, während $(x = 0, y = 1, z = 0)$ bedeutet, dass das Objekt um die y-Achse gedreht wird, wie es beispielhaft in Abbildung 2.2 von links nach rechts gezeigt wird.
Die Daten dieses Sensors werden von beiden Geräten aufgezeichnet.

2.1.3. Hautwiderstandssensor

Je nach aktueller körperlicher Betätigung ändert sich die elektrische Leitfähigkeit der Hautoberfläche durch Schweiß. Der Hautwiderstandssensor misst den elektrischen Widerstand der Haut, der sich umgekehrt proportional zur Leitfähigkeit verhält. Dieser Sensor ist nur im Band integriert, das daher für diesen die einzige Datenquelle ist.

2.1.4. Hauttemperatursensor

Der Hauttemperatursensor misst die Temperatur der Haut des Nutzers, die an der Unterseite des Band anliegt. Leider reagierte dieser Sensor in einem Test nur langsam auf Temperaturveränderungen: Nachdem der Tracker abgelegt wurde, übermittelte dieser den Temperaturabfall erst mehrere Sekunden später. Aus diesem Grund wurde auf die Aufzeichnung dieser Daten verzichtet.

2.1.5. Laufgeschwindigkeitssensor

Der Laufgeschwindigkeitssensor ist ein virtueller Sensor, der auf Basis des elektronischen Schrittzählers im Microsoft Band 2 generiert wird. Er gibt die Laufgeschwindigkeit des Trägers in cm/s an.

2.2. Wichtige Begriffe

In den folgenden Kapiteln wird von einem grundlegenden Verständnis von Machine Learning-Klassifikation ausgegangen.

Definition 1 (ML-Klassifikation)**.** *Gegeben sei ein Datensatz $D \ni (x_1,...,x_n,y)$ mit $|D| = m$ Instanzen. Dann sind $x = (x_1,...,x_n)$ die Input-Features und y das Target. Im Falle eines Klassifikationsproblems ist y nominal, d.h. ein deskriptiver Wert zur Identifikation. Gesucht ist nun eine Hypothesenfunktion $h(x) = \hat{y}$, die zu gegebenen Features den wahrscheinlichsten Wert des Targets bestimmt[15].*

Neben Definition 1 sind für das Verständnis dieser Arbeit die folgenden Begriffe wichtig:

1. (Überwachtes) Lernen, bzw. Training. Definiert in Kapitel 2.1 [9].
2. Overfitting und Train-Test-Split. Definiert und erläutert in Kapitel 7.2 [9].
3. Kreuzvalidierung. Definiert und erläutert in Kapitel 7.10 [9].
4. Hyperparameter. Dabei handelt es sich um Parameter eines ML-Algorithmus, die dieser nicht eigenständig optimieren kann.

3. Review of the State of the Art

TODO: Review, grouped by idea not author

4. Experiment

In diesem Kapitel werden Aufbau und Durchführung des Experiments erläutert, durch das der Datensatz zusammengestellt wurde.

Insgesamt haben 10 Teilnehmer am Experiment teilgenommen und die in Abbildung 4.1 achtzehn gelisteten Aktivitäten durchgeführt. Weiss et al. nahmen jeweils zwei Minuten pro Teilnehmer und Aktivität auf[18], mussten allerdings zum Anfang und Ende jeder Aufnahme je zehn Sekunden abschneiden, um Ausreißer zu entfernen. Aus diesem Grund betrug die Dauer jeder Aufnahme in diesem Experiment drei Minuten, wodurch der gesamte Aufnahmeprozess pro Person rund zwei Stunden dauerte. Vor dem Experiment wurden Einverständniserklärungen der Teilnehmer eingeholt.

Die Aufnahme erfolgte mit dem Fitness-Tracker Microsoft Band 2 und dem Smartphone OnePlus 3, auf dem das Betriebssystem Android 6 installiert war. Der Fitness-Tracker wurde am Handgelenk des Teilnehmers befestigt, während das Smartphone in seiner Hosentasche platziert wurde, wie in [18] jeweils auf der dominanten Seite des Teilnehmers. Dies war notwendig, da insbesondere Aktivitäten wie das Dribbeln eines Basketballs mit der dominanten Hand durchgeführt werden. Während dies für Armbanduhren nicht notwendigerweise eine realistische Konfiguration ist, da diese üblicherweise auf der nicht-dominanten Seite getragen werden, besteht dieses Problem bei Fitness-Trackern nicht unbedingt. So lässt sich beispielsweise in Fitness-Trackern der Firma Fitbit einstellen, auf welcher Seite dieser getragen wird, was die Vermutung nahelegt, dass genügend Nutzer mit dem Tragen auf dieser Seite kein Problem haben [6].

Auf dem Smartphone wurde eine eigens für das Experiment entwickelte Anwendung ausgeführt, in der zunächst der Name des Teilnehmers eingegeben wurde. Sowohl auf dem Smartphone selbst als auch per drahtloser Fernsteuerung wurde die durchzuführende Aktivität eingestellt. Gestartet und gestoppt wurde die Aufnahme anschließend per Fernsteuerung, um Ausreißer am Anfang und Ende der Aufnahme zu vermeiden, obgleich trotzdem jeweils zehn Sekunden abgeschnitten wurden.

4.1. Beschreibung der Aktivitäten

Im Folgenden werden die einzelnen Aktivitäten, die von den Teilnehmern ausgeführt wurden, genauer beschrieben. Um eine Vergleichbarkeit mit [18] zu ermöglichen, handelt es sich mangels detaillierter Beschreibungen zumindest um ähnliche Aktivitäten, die dieselben Bezeichnungen haben.

4.1.1. Allgemeine Aktivitäten (nicht handorientiert)

Gehen

Der Teilnehmer bewegt sich im Schrittempo auf einem Gehweg.

Joggen

Der Teilnehmer joggt auf einem Gehweg. Das Tempo variiert je nach sportlicher Verfassung des Teilnehmers.

Treppensteigen

Der Teilnehmer bewegt sich abwechselnd eine Treppe hoch unter herunter. Steigung und Länge sind variabel.

Sitzen

Der Teilnehmer sitzt auf einem Stuhl und versucht, sich dabei nicht unruhig zu verhalten.

Stehen

Der Teilnehmer steht und versucht, sich dabei nicht unruhig zu verhalten.

Fußball schießen

Der Teilnehmer schießt einen Fußball wiederholt mit mittlerer Kraft zu einem Mitspieler und versucht, Sprinten zu vermeiden.

4.1.2. Allgemeine Aktivitäten (handorientiert)

Basketball dribbeln

Der Teilnehmer schleudert einen Basketball wiederholt Richtung Boden und versucht, dabei möglichst an einer Stelle stehen zu bleiben.

Mit einem Tennisball Fangen spielen

Zwei Personen werfen sich abwechselnd einen Tennisball zu und versuchen dabei, möglichst an einer Stelle stehen zu bleiben.

Auf einer Tastatur tippen

Der Teilnehmer tippt sitzend seinen Gewohnheiten nach einen Text an einer beliebigen Computertastatur ab. Mindestens grobe Fehler sollten korrigiert werden. Um Verständnisproblemen aus dem Weg zu gehen, handelt es sich bei dem Text um ein Diktat für Siebtklässler.

Auf Papier schreiben

Der Teilnehmer schreibt sitzend seinen Gewohnheiten nach denselben Text wie mit der Tastatur mit einem Kugelschreiber auf ein Blatt Papier im Format DIN A4 ab.

Klatschen

Der Teilnehmer begleitet sitzend das Lied "Viva la Vida" von Coldplay klatschend.

Zähneputzen

Der Teilnehmer benutzt stehend eine Handzahnbürste (nicht elektrisch), um sich damit die Zähne zu putzen.

Kleidung falten

Der Teilnehmer faltet stehend der Reihe nach T-Shirts auf einem Tisch.

4.1.3. Essaktivitäten (handorientiert)

Spaghetti essen

Der Teilnehmer isst Spaghetti mit einer Soße.

Suppe essen

Der Teilnehmer isst eine Suppe seiner Wahl.

Brot essen

Der Teilnehmer isst ein belegtes Brot.

Chips essen

Der Teilnehmer isst Chips aus einer handelsüblichen Tüte.

Aus einer Tasse oder einem Glas trinken

Der Teilnehmer trinkt wiederholt aus einer Tasse oder einem Glas.

5. Methode

Dieses Kapitel erläutert die Art und Weise, mit der aus den im Experiment gewonnenen Daten Modelle trainiert wurden, welche die ausgeführte Aktivität automatisch erkennen können. Der erste Schritt ist die Aufnahme der Rohdaten, die durch eine in Abschnitt 5.1 Android-App realisiert wurde. Die gespeicherten Rohdaten sind noch nicht für einen herkömmlichen ML-Klassifikationsalgorithmus verwertbar, weshalb eine in Abschnitt 5.2 beschriebene Transformation ausgeführt wird. Als dritter und letzter Schritt erfolgt die Ausführung diverser ML-Klassifikationsalgorithmen, die in Abschnitt 5.4 erläutert werden.

5.1. Implementierung der Aufzeichnungssoftware

5.1.1. Definition der Messdaten

Definition 2. *Ein Reading besteht aus einem Unix-Zeitstempel in Millisekunden, einer Gerätequelle, einem Sensortyp und den Werten des Sensors gruppiert nach Komponente. Ein Beispiel ist* $(1000, Band, Gyroscope, \{x \to 0.0, y \to 1.0, z \to 1.0\})$.

5.1.2. Struktur der Aufzeichnungssoftware

Die Aufzeichnungssoftware wurde als Anwendung für das Android-Betriebssystem implementiert, da für diese Plattform auch ein *Software Development Kit (SDK)* für das Microsoft Band 2 existiert.

Abbildung 5.1 bietet einen Überblick über die Struktur der Aufzeichnungssoftware. Im Zentrum steht die Klasse *CollectionService*, die für die robuste Sammlung der Daten verantwortlich ist und daher als persistenter Android-*Service* implementiert ist, der im Gegensatz zu anderen Komponenten einer Android-App nicht ohne weiteres durch das System zur Schonung der Ressourcen beendet werden kann. Gesteuert wird die Aufnahme über eine grafische Benutzeroberfläche auf dem Smartphone selbst (*AndroidUI*), sowie optional auch über eine Weboberfläche, die über einen einfachen HTTP-Server zur Verfügung steht. In den Benutzeroberflächen kann der Name des Probanden, sowie die ausgeführte Aktivität angegeben werden. Des Weiteren wird hierüber die Aufnahme gestartet und gestoppt.

Der *CollectionService* delegiert die Aufnahme an den *BandDataRecorder* und den *LocalDataRecorder* weiter, die jeweils für das Microsoft Band 2 respektive die Smartphone-lokalen Sensoren verantwortlich sind. Über den *BandConnector* wird die Bluetooth-Verbindung zum Band hergestellt und aufrecht erhalten. *BandDataRecorder* und *LocalDataRecorder* zeichnen Daten in Form von *Readings* auf. Die Methode *getValuesPerComponent()* liefert beispielsweise Daten der Form $\{x \to 1.0, y \to 2.0, z \to 3.0\}$, wenn die Komponenten des jeweiligen Sensors der jeweiligen *Source* $\{x, y, z\}$ sind.

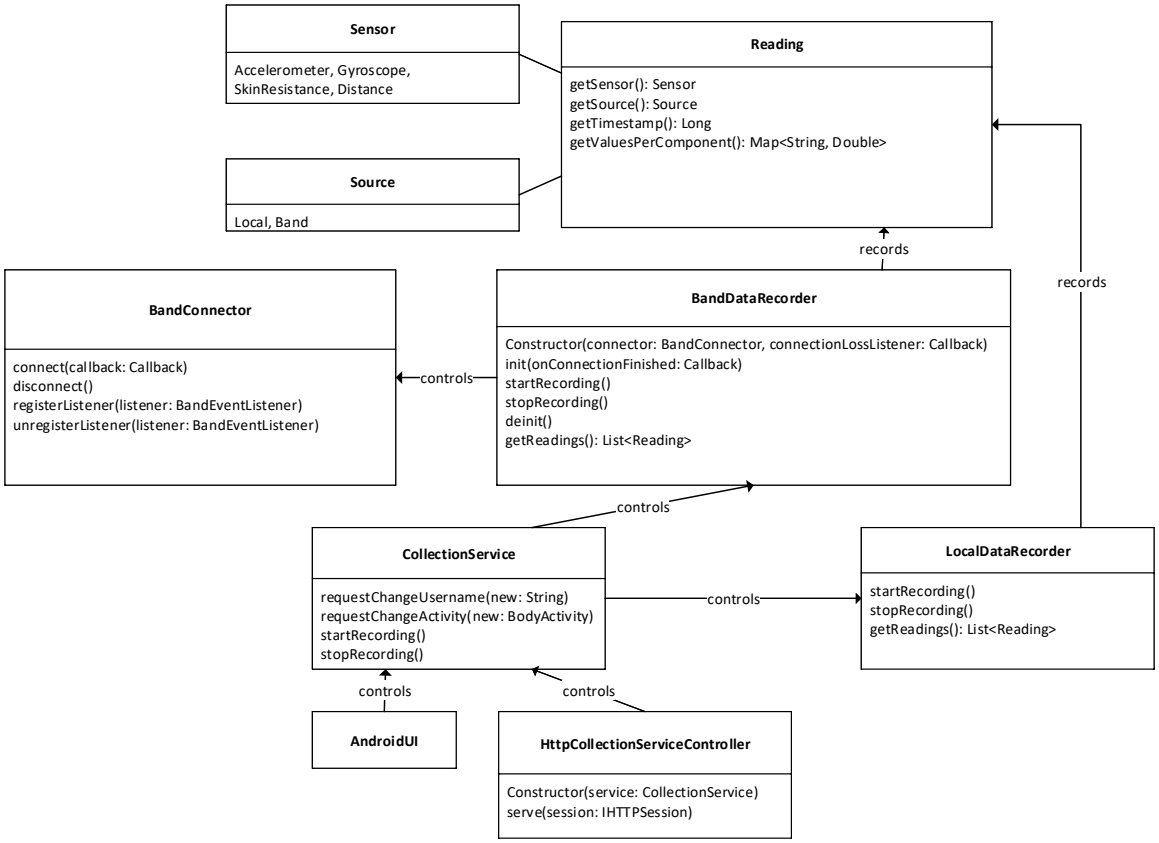

Abbildung 5.1.: Struktur der Aufzeichnungssoftware

Wird die Aufnahme gestoppt, werden die Readings mitsamt den Metadaten Name des Probanden und Aktivität serialisiert und per SFTP auf einen entfernten Speicher hochgeladen. Anschließend werden die Daten von dort abgerufen und wie im folgenden Abschnitt beschrieben weiterverarbeitet.

5.2. Transformation der Daten

Jede einzelne Aufnahme eines Nutzers und einer Aktivität besteht aus einer Reihe von Daten mit Zeitstempeln. In dieser Form kann noch kein konventioneller ML-Klassifikationsalgorithmus mit den Daten arbeiten, da diese Daten in Form einer Liste von *Instanzen* erwarten. Eine Instanz besteht aus einer Menge von *Attributen/Features*, die in Rahmen dieser Arbeit bis auf das nominale *Target*-Attribut allesamt kardinal sind. Das Target-Attribut ist dabei die durchgeführte Aktivität. Entsprechend ist eine Transformation der Daten notwendig, um diese tatsächlich nutzbar zu machen.

Um Instanzen zu erzeugen, teilt die Transformationssoftware die Ursprungsdaten zunächst in Intervalle mit zehnsekündiger Dauer auf. Ein solches Intervall soll schließlich eine Instanz,

d.h. ein Beispiel für eine Aktivität bilden. Das erste und das letzte Intervall werden gelöscht, da in dieser Zeit die Aufnahme gestartet und gestoppt wurde, was ansonsten durch den anderen Bewegungsablauf die Ergebnisse verfälschen könnte. Aktuell umfasst das Intervall noch mehrere Readings, die aggregiert werden müssen.

5.2.1. Beschreibung der Aggregatfunktionen

Sei $A[1..N]$ eine Liste von Zahlen. Dann seien die folgenden Aggregatfunktionen in Anlehnung an Kwapisz et al.[13] wie folgt definiert:

Average

$$\text{Avg}(A) := \frac{1}{N} * \sum_{a \in A} a$$

Standard Deviation

$$\text{StdDev}(A) := \sqrt{\frac{1}{N} * \sum_{a \in A} (a - \text{Avg}(A))^2}$$

Average Absolute Difference

$$\text{AvgAbsDiff}(A) := \frac{1}{N} * \sum_{a \in A} |\text{Avg}(A) - a|$$

k-Binned Distribution

Im Gegensatz zu den obigen Funktionen liefert diese Aggregatfunktion einen k-Vektor anstatt einen Skalar. Seien $i \in \{0, ..., k-1\}, S := \frac{\max A - \min A}{k}$.

$$\text{Bin}(A, i) := \#\{a \in A | (\min A) + i * S \leq a < (\min A) + (i+1) * S\}$$

$\text{Bin}(A, i)$ ist demzufolge die Anzahl der Elemente in Korb i, wenn man A der Größe nach sortiert auf k Körbe verteilt. Der Bin ist hierbei als Multimenge zu verstehen und darf somit Elemente auch mehrfach enthalten.

Average Root of Squares

Auch diese Aggregatfunktion unterscheidet sich von den bisherigen, da sie mehrere Listen als Parameter erhält:

$$\text{Aros}(A_1, ..., A_M) = \frac{1}{M} * \sum_{i=1}^{M} \sqrt{\sum_{a \in A_i} a^2}$$

Average Time between Peaks

Diese Aggregatfunktion gibt die durchschnittliche Zeit zwischen Höhepunkten in A zurück, wofür zusätzlich eine Funktion timeForIndex : $\mathbb{N} \to \mathbb{N}$ benötigt wird. Die Wahl der Höhepunkte erfolgt durch eine Heuristik.

Algorithmus 2 beschreibt die Berechnung des Wertes. Zunächst wird mittels Algorithmus 1 bestimmt, an welchen Indizes in der Liste Werte stehen, die deutlich größer als ihre Nachfolger sind. Dadurch wird definiert, was eine Spitze in dieser spezifischen Liste ausmacht. Anschließend werden Spitzen gesucht, die maximal um den Faktor *threshold* kleiner sind als die größte Spitze, die der Algorithmus gefunden hast. Diese Grenze wird solange gesenkt, bis genügend Spitzen gefunden wurden oder die Grenze so niedrig liegt, dass es sich nicht mehr um Spitzen handelt. Als letztes berechnet der Algorithmus die durchschnittliche Zeit zwischen den Spitzen mithilfe der Funktion *timeForIndex*.

Die im Pseudocode verwendeten Konstanten erwiesen sich im praktischen Test an den gesammelten Daten als gut geeignet. Für andere Datensätze kann eine Anpassung erforderlich sein.

Algorithm 1 IndicesOfPeaks(A, t), $t \in [0,1]$

▷ Returns a list of indices i where $A[i] * t \geq A[i+1]$

```
indices ← ∅
for i ∈ {1, ..., |A| − 1} do
    if A[i] ∗ t ≥ A[i + 1] then
        indices ← indices ∪ {i}
    end if
end for
return indices
```

5.2.2. Anwendung der Aggregatfunktionen

Die Transformationssoftware reduziert nun mithilfe der Aggregatfunktionen die Readings eines jeden Intervalls I auf eine konstante Anzahl skalarer Werte, welche die Features der Instanz darstellen. Im Folgenden sei I ein Array der Struktur $I[G][S][K] \in \mathbb{R}^{\alpha}$, das die Messungen gruppiert nach Gerät (G), Sensor (S) und Komponente (K) beinhaltet. Des Weiteren sei F_I die Menge der Features des Intervalls I. Die Folgenden Unterabschnitte zeigen den Aufbau von F_I mittels der Aggregatfunktionen:

Pro Kombination von Gerät G und Sensor S

Seien P die einzelnen Komponenten des Sensors, beispielsweise $P = \{x, y, z\}$.

$$\begin{aligned} \text{aros} &\leftarrow \text{Aros}(I[G][S][P[1]], ..., I[G][S][P[|P|]]) \\ F_I &\leftarrow F_I \cup \{((G,S), \text{aros})\} \end{aligned}$$

Im Kontext betrachtet liefert *Aros* am Beispiel des Beschleunigungssensors gewissermaßen die durchschnittliche Beschleunigung, die alle Achsen zusammen erfahren haben.

Algorithm 2 AverageTimeBetweenPeaks(A, timeForIndex, minPeaks)

▷ *First, heuristically define what a peak is*
indicesOfPeaks ← IndicesOfPeaks($A, t = 0.8$)
if indicesOfPeaks $= \emptyset$ **then**
    return Nil
end if
highestPeakIdx ← Index of highest peak in indicesOfPeaks

▷ *Lower the threshold until we have found enough peaks or the threshold is too low*
otherPeakIndices ← $\emptyset$
threshold ← 0.8
repeat
    otherPeakIndices ← Indices of A where $A[i] \geq A[\text{highestPeakIdx}] *$ threshold
    threshold ← threshold $* 0.9$
until $|\text{otherPeaksIndices}| \geq \text{minPeaks} \vee \text{threshold} < 0.3$

▷ *Check whether we have found enough peaks*
if $|\text{otherPeakIndices}| < \text{minPeaks}$ **then**
    return Nil
end if

▷ *Calculate the average time between the peaks*
timesBetweenPeaks ← $\emptyset$
for $i \in \{2, ..., |\text{otherPeakIndices}|\}$ **do**
    time ← timeForIndex(otherPeakIndices[i]) - timeForIndex(otherPeakIndices[$i-1$])
    timesBetweenPeaks ← timesBetweenPeaks $\cup \{\text{time}\}$
end for
return AVG(timesBetweenPeaks)

Pro Kombination von Gerät G, Sensor S und Komponente K

Sei *timeForIdx* hier eine Funktion, die den Zeitstempel des jeweiligen Datums im Intervall zurückgibt. Sei außerdem $k := 10$, sodass 10 Körbe verwendet werden, was sich experimentell als geeignet herausstellte.

$$
\begin{aligned}
F_I &\leftarrow F_I \cup \{((G,S,K), \text{Avg}(I[G][S][K]))\} \\
F_I &\leftarrow F_I \cup \{((G,S,K), \text{StdDev}(I[G][S][K]))\} \\
F_I &\leftarrow F_I \cup \{((G,S,K), \text{AvgAbsDiff}(I[G][S][K]))\} \\
F_I &\leftarrow F_I \cup \{((G,S,K), \text{AverageTimeBetweenPeaks}(I[G][S][K], \text{timeForIdx}, 3))\} \\
F_I &\leftarrow F_I \cup \{((G,S,K), \text{Bin}(I[G][S][K], i))\} \forall i \in \{1, ..., k\}
\end{aligned}
$$

5.2.3. Beispiel

Wir betrachten für die Aktivität *Jogging* den Beispieldatensatz

$$
\begin{aligned}
D = \{&(0, \text{Band}, \text{Gyroscope}, x \to 1.0, y \to 0.0), \\
&(0, \text{Band}, \text{Accelerometer}, x \to 1.0, y \to 1.0), \\
&(1000, \text{Band}, \text{Gyroscope}, x \to 2.0, y \to 2.0), \\
&(1000, \text{Band}, \text{Accelerometer}, x \to 0.0, y \to 1.0), \\
&(2000, \text{Band}, \text{Gyroscope}, x \to 3.0, y \to 0.0), \\
&(2000, \text{Band}, \text{Accelerometer}, x \to 1.0, y \to 3.0)\}
\end{aligned}
$$

Zur Vereinfachung wird von 2D-Sensoren ausgegangen, sodass keine z-Komponente vorhanden ist. Zur werden als Aggregatfunktionen der Durchschnitt, die *Binned Distribution* mit zwei *Bins* und der *Average Root of Squares* verwendet. Die gewählte Intervallgröße beträgt zwei Sekunden, sodass sich die folgenden Intervalle ergeben:

$$
\begin{aligned}
I_1 = \{&(0, \text{Band}, \text{Gyroscope}, x \to 1.0, y \to 0.0), \\
&(0, \text{Band}, \text{Accelerometer}, x \to 1.0, y \to 1.0), \\
&(1000, \text{Band}, \text{Gyroscope}, x \to 2.0, y \to 2.0), \\
&(1000, \text{Band}, \text{Accelerometer}, x \to 0.0, y \to 1.0)\},
\end{aligned}
$$

$$
\begin{aligned}
I_2 = \{&(2000, \text{Band}, \text{Gyroscope}, x \to 3.0, y \to 0.0), \\
&(2000, \text{Band}, \text{Accelerometer}, x \to 1.0, y \to 3.0)\}
\end{aligned}
$$

Nun werden die Aggregatfunktionen auf I_1 angewendet:

$$\text{Avg}_{\text{Band, Gyroscope, x}} = (1+2)/2 = 1.5$$
$$\text{Avg}_{\text{Band, Gyroscope, y}} = (0+2)/2 = 1$$
$$\text{Avg}_{\text{Band, Accelerometer, x}} = (1+0)/2 = 0.5$$
$$\text{Avg}_{\text{Band, Accelerometer, y}} = (1+1)/2 = 1$$
$$\text{Bin}_{\text{Band, Gyroscope, x}}(0) = \#\{1\} = 1$$
$$\text{Bin}_{\text{Band, Gyroscope, x}}(1) = \#\{2\} = 1$$
$$\text{Bin}_{\text{Band, Gyroscope, y}}(0) = \#\{0\} = 1$$
$$\text{Bin}_{\text{Band, Gyroscope, y}}(1) = \#\{2\} = 1$$
$$\text{Bin}_{\text{Band, Accelerometer, x}}(0) = \#\{0\} = 1$$
$$\text{Bin}_{\text{Band, Accelerometer, x}}(1) = \#\{1\} = 1$$
$$\text{Bin}_{\text{Band, Accelerometer, y}}(0) = \#\{0\} = 0$$
$$\text{Bin}_{\text{Band, Accelerometer, y}}(1) = \#\{1\} = 0$$
$$\text{Aros}_{\text{Band, Gyroscope}} = (1/2) * (\underbrace{\sqrt{1^2+2^2}}_{x} + \underbrace{\sqrt{0^2+2^2}}_{y}) \approx 2.12$$
$$\text{Aros}_{\text{Band, Accelerometer}} = (1/2) * (\underbrace{\sqrt{1^2+0^2}}_{x} + \underbrace{\sqrt{1^2+1^2}}_{y}) \approx 1.2$$

Die Berechnung für I_2 erfolgt analog. Aus den obigen Features ergibt sich der folgende Datensatz, der von einem herkömmmlichen Klassifikationsalgorithmus verarbeitet werden kann:

Intervall	$\text{Avg}_{\text{Band, Gyroscope, x}}$	$\text{Avg}_{\text{Band, Gyroscope, y}}$	...	sampleClass
I_1	1.5	1	...	Jogging
I_2	...	...	...	Jogging

Die Intervallspalte dient nur der Illustration.

5.3. Nachbearbeitung der Daten

Bei der Transformation kann sich das Problem ergeben, dass nicht jede Zelle der resultierenden Tabelle einen Wert enthält. Dazu kann es beispielsweise kommen, wenn der Algorithmus `AverageTimeBetweenPeaks` *Nil* zurückgibt. Enthält eine Zelle keinen Wert, so wird in ihr der Durchschnittswert des gefüllten Teils der Spalte eingesetzt. Um dem Risiko vorzubeugen, dass ML-Algorithmen Spalten mit betragsmäßig hohen Werten fälschlicherweise auch automatish hohen Einfluss zuschreiben, werden alle numerischen Spalten mithilfe des z-Score-Verfahrens normalisiert. Dazu wird von den Werten jeder Spalte der Durchschnittswert dieser abgezogen, wonach das Resultat durch die Standardabweichung der Spalte geteilt wird. Der Betrag einer Zelle gibt dann an, wie viele Standardabweichungen vom Durchschnitt abgewichen wird.

5.4. Anwendung von ML-Klassifikationsalgorithmen

Um die Vergleichbarkeit mit [18] sicherzustellen, wurde die Java-basierte ML-Software WEKA zum maschinellen Lernen verwendet. Diese bietet fertige Implementierungen diverser ML-Algorithmen an, die sowohl mittels einer grafischen Benutzeroberfläche, als auch in Code verwendet werden können. Die in dieser Arbeit verwendeten Algorithmen entsprechen denen aus [18] und werden im Folgenden kurz beschrieben. Falls nicht anders angegeben, wurden für jedes Modell die standardmäßigen Hyperparameter verwendet. Dabei handelt es sich um Parameter, die der jeweilige Algorithmus nicht selbst lernen kann. TODO: Hyperparameter aktualisieren, falls nötig

5.4.1. Random Forest (RF)

Ein *Random Forest* ist ein Wald von Entscheidungsbäumen (erklärt in Kapitel 14.4 [4]), die randomisiert entstanden sind. Soll eine neue Instanz klassifiziert werden, treffen alle Entscheidungsbäume eine Entscheidung und die am häufigsten vorhergesagte Klasse gewinnt. WEKA nutzt eine Implementierung nach [5]. Experimentell haben sich die folgenden Hyperparameterwerte als sinnvoll erwiesen: Die Anzahl der Bäume in Wald wurde auf 50, die Anzahl der verwendeten Features pro Baum auf 10, sowie die maximale Tiefe auf 25 festgelegt.

5.4.2. J48-Entscheidungsbaum (J48)

J48 ist eine Implementierung des *C4.5*-Algorithmus[16]. Der Entscheidungsbaum wird anhand von Trainingsdaten T iterativ durch das Hinzufügen von Entscheidungsknoten erzeugt, die den Informationsgehalt der Teilmengen von T minimieren, die aus dem Split an jenem Entscheidungsknoten hervorgehen. Der Informationsgehalt umso höher, je mehr verschiedene Klassen sich in einer Menge von Instanzen befinden. Ein niedriger Informationsgehalt besagt demnach, dass nach einer Entscheidung an einem Entscheidungsknoten klarer geworden ist, welcher Klasse die Instanzen in den Mengen nach dem Split zuzuordnen sind.

5.4.3. Instance-Based-Learner (IBk)

Der IBk-Algorithmus ist ein kNN-basierter Klassifikationsalgorithmus. Soll einer neuen Instanz $x \in \mathbb{R}^n$ eine Klasse zugeordnet werden, werden anhand eines Distanzmaßes für den Vektorraum $\mathbb{R}^n$ die k nächsten Nachbarn von x gesucht. x wird danach die Klasse zugeordnet, die unter den k Nachbarn am häufigsten vorkam. Weitere Details sind zu finden in [2]. k ist dabei ein Hyperparameter, der auf 3 eingestellt wurde.

5.4.4. Naive Bayes (NB)

Naive Bayes ist eine probabilistische Methode zur Klassifikation[11]. Mit $x = (x_1, ..., x_n)$ wird die Klasse C gesucht, die $\mathbb{P}(C|x_1, ..., x_n)$ maximiert. x besteht hier aus nominalen Attributen. Der Satz von Bayes besagt, dass $\mathbb{P}(C|x) = \frac{\mathbb{P}(C)\mathbb{P}(x|C)}{\mathbb{P}(x)}$. Da über C optimiert wird, kann der Nenner für diesen Ansatz ignoriert werden. Nimmt man *naiv* an, dass $x_1, ..., x_n$ voneinander unabhängig sind, kann somit statt $\mathbb{P}(C|x)$ auch folgender Term über C maximiert werden: $\mathbb{P}(C)\prod_{i=1}^{n}\mathbb{P}(x_i|C)$. Durch den Trainingsdatensatz sind $\mathbb{P}(C)$ als Prior der Klasse C, sowie

alle $\mathbb{P}(x_i|C)$ bekannt, sodass der Term durch einfaches Ausprobieren aller Klassen maximiert werden kann.

Enthält x numerische Attribute, müssen diese zunächst durch nominale Attribute ersetzt werden. Der Wertebereich aller x_i über den Trainingsdatensatz hinweg kann in eine feste Anzahl von Intervallen aufgeteilt werden. Für jedes Intervall I wird nun eine Indikatorvariable als Feature angelegt, die angibt, ob für eine Instanz x gilt, dass $x_i \in I$.

5.4.5. Multilayer Perceptron (MLP)

Bei dieser Methode handelt es sich um ein mehrschichtiges Netzwerk von sogenannten *Perzeptren*, auch *(künstliches) neuronales Netzwerk* genannt. Ein Perzeptron erhält n Eingaben $x_1, ..., x_n$ und gewichtet diese mit Parametern $\Theta_1, ..., \Theta_n$, die gelernt werden. Das Perzeptron berechnet $y = g(\sum_{i=1}^{n} \Theta_i x_i)$ als Ausgabewert, wobei g eine Aktivierungsfunktion ist, die den Ausgabewerte beschränkt, beispielsweise die Sigmoidfunktion.

Die Ausgabe eines Perzeptron kann einem Perzeptron der nächsten Schicht als Eingang dienen. Die letzte Schicht fungiert als Ausgabe des Netzwerks, dessen Fehler anhand eines Trainingsdatensatzes bestimmt werden kann. Jedes Perzeptron berechnet, wie die eigenen Parameter Θ angepasst werden müssen, um den Fehler zu minimieren, wofür der Fehler der Perzeptren der letzten Schicht an die vorherige Schicht weiter propagiert wird (*Backpropagation*). Zur Optimierung der Parameter wird ein stochastischer Gradientenabstieg verwendet. Zur Klassifikation können die Ausgangswerte der Perzeptren der letzten Schicht als Indikatorvariablen für die vorhandenen Klassen interpretiert werden. Weitere Informationen sind zu finden in [10].

6. Evaluation

Dieses Kapitel widmet sich der Auswertung der Ergebnisse anhand verschiedener Metriken. Eingebettet in der Transformationssoftware befindet sich ein Präprozessor, der die aufgenommen Daten vor der Transformation manipulieren kann. Auf diese Weise werden mehrere Trainingsdatensätze erstellt, die in der Evaluation daraufhin analysiert werden, wie genau ein Klassifikator nach dem Training mit ihnen Vorhersagen treffen kann.

Die folgenden Datensätze werden generiert:

1. *Combined*: Alle Daten werden ohne Veränderungen mit in die Transformation einbezogen.
2. *NoisyBandTimestamps*: Die Zeitstempel der Readings des Bands werden mit gaussschem Rauschen versehen
3. *Band combined*: Alle Daten des Smartphones werden vor der Transformation verworfen, sodass nur noch die Daten des Band verbleiben..
4. *Band accel*: Nur die Daten des Beschleunigungssensors des Microsoft Band 2 werden mit in die Transformation einbezogen.
5. *Band gyro*: Nur die Daten des Gyroskops des Microsoft Band 2 werden mit in die Transformation einbezogen.
6. *Phone comb.*: Alle Daten des Microsoft Band 2 werden vor der Transformation verworfen, sodass nur noch die Daten des Smartphones verbleiben.
7. *Phone accel*: Nur die Daten des Beschleunigungssensors des Smartphones werden mit in die Transformation einbezogen.
8. *Phone gyro*: Nur die Daten des Gyroskops des Smartphones werden mit in die Transformation einbezogen.
9. *SamplingRate1Hz*: Es werden Readings verworfen, als hätten Band und Smartphone jeweils nur Readings bei einer Rate von 1 Hz geliefert.
10. *SamplingRate5Hz*: Es werden Readings verworfen, als hätten Band und Smartphone jeweils nur Readings bei einer Rate von 5 Hz geliefert.

6.1. Effektivität der Datenkombination

Dieser Abschnitt widmet sich der Frage, wie effektiv die Kombination der Daten von Band und Smartphone hinsichtlich der Vorhersagegenauigkeit gegenüber einer einzigen Datenquelle

ist. Weiss et al.[18] werten in ihrem Papier die Genauigkeit von Modellen aus, die entweder auf Daten des Beschleunigungssensors eines Smartphones, des Beschleunigungssensors einer Smartwatch oder des Gyroskops einer Smartwatch zurückgreifen. Eine Kombination der Daten schlagen die Autoren vor, führen diese allerdings nicht durch.

Tabelle 6.1 zeigt die Genauigkeit der persönlichen Modelle und Tabelle 6.2 die der unpersönlichen Modelle. Ein persönliches Modell basiert auf Daten des jeweiligen Benutzers, während ein unpersönliches Modell noch keine Daten des Nutzers gesehen hat, für den eine Vorhersage getroffen werden soll. Die *Genauigkeit* ist definiert als der Anteil der korrekt klassifizierten Instanzen, die dem Modell zum Test vorgelegt wurden.

Die erste Spalte der in diesem Abschnitt referenzierten Tabellen gibt den Algorithmus an, der zur Klassifikation verwendet wurde. Eine Erläuterung der Algorithmen ist in Abschnitt 5.4 zu finden. Fettgedruckt ist immer der jeweils höchste Wert einer Spalte.

6.1.1. Persönliche Modelle

Die Evaluierung der persönlichen Modelle erfolgt wie folgt: Für jeden Benutzer b wird eine 10-fache Kreuzvalidierung durchgeführt, wobei die Trainingsdaten nur Daten von b beinhalten. Anschließend wird der Durchschnitt der einzelnen Genauigkeiten über alle Benutzer gebildet, um einen globalen Genauigkeitswert zu erhalten. Um dieses Testverfahren zu ermöglichen, wurde die Weka-Software leicht angepasst: Die Methode zur Kreuzvalidierung wurde um einen Parameter $\texttt{foldSelector} : \texttt{Instanzen} \rightarrow \mathbb{N}$ erweitert, der für eine Instanz aus dem Datensatz zurückgibt, in welcher Iteration der Kreuzvalidierung diese aus dem Trainingsdatensatz ausgeschlossen und stattdessen im Testdatensatz verwendet wird.

Nutzt man nur je ein Gyroskop, ist die Qualität der persönlichen Modelle am schlechtesten. Im Gegensatz dazu haben bereits Modelle, die lediglich Daten von einem der Beschleunigungssensoren verwenden, eine gute Aussagekraft. Weiss et al.[18] stellten in ihrem Experiment fest, dass die Genauigkeit des *Phone accel*-Modells 17.8% über dem *Watch accel / Band accel*-Modell lag. Im eigenen Experiment ist der Unterschied trotz gleichem Experimentaufbau mit 1.6% weit geringer. Eine denkbare Erklärung wäre, dass die Probanden von Weiss et al. über verschiedene Aktivitäten hinweg mehr Variation bei ihrer Bewegung des Unterkörpers eingebracht haben könnten.

Vergleicht man *Band accel* als die beste Datenquelle mit den Kombinationsmöglichkeiten, so lässt sich feststellen, dass die Hinzunahme der anderen Sensoren des Band eine Genauigkeit von 97.1% erzielt, wohingegen die Kombination der beiden Smartphone-Sensoren lediglich eine Verbesserung von 0.7% gegenüber *Phone accel* und gegenüber *Band accel* sogar eine Verschlechterung erwirkt.

Hebt man jedoch diese künstlichen Einschränkungen auf, so wird eine Genauigkeit von 99.4% erreicht, sodass Benutzer nach dreiminütiger Aufnahme einer Aktivität davon ausgehen könnten, dass diese in Zukunft bis auf wenige Ausreißer automatisch erkannt wird. Insgesamt konnte gegenüber *Phone Accel* damit eine Verbesserung um 9.4% erreicht werden. Gegenüber dem besten Ergebnis eines einzelnen Sensors (*Band accel*) mit 91.6% liefert die Kombination aller Daten eine Verbesserung um 7.8%.

TODO: Describe the following two figs,put second fig in appropriate section

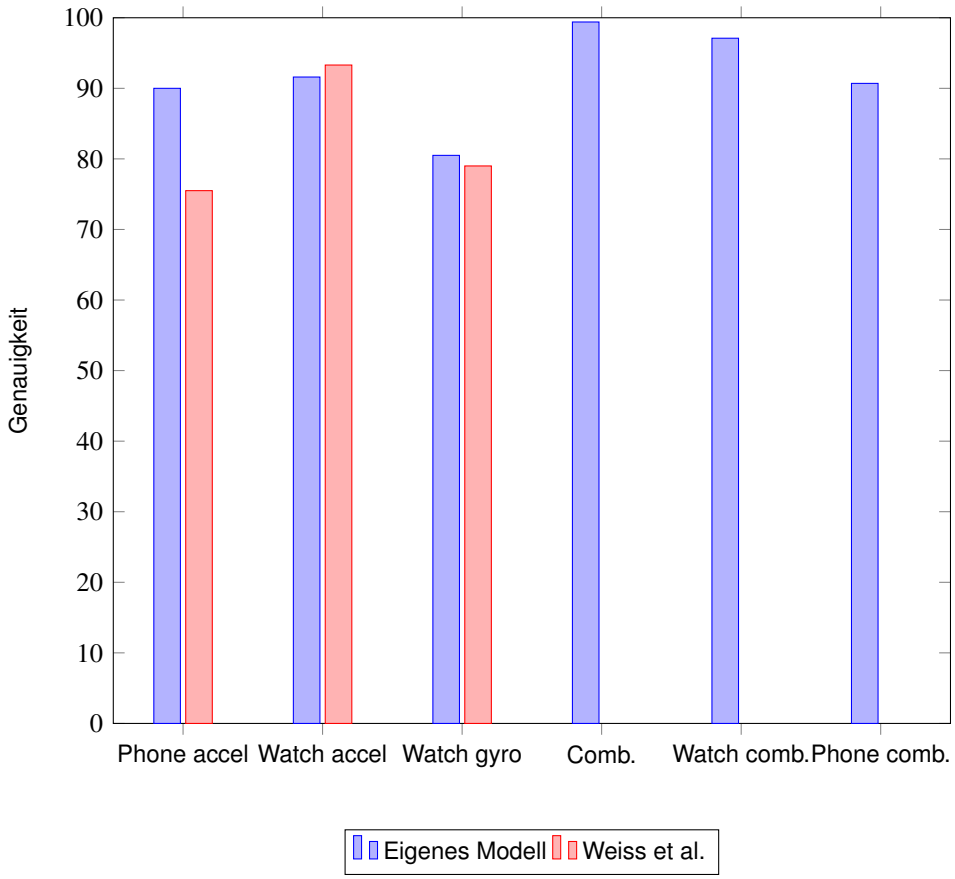

Abbildung 6.1.: Vergleich der Genauigkeiten der persönlichen Modelle mit Weiss et al.[18]. Die *Watch* ist in dieser Arbeit das Band.

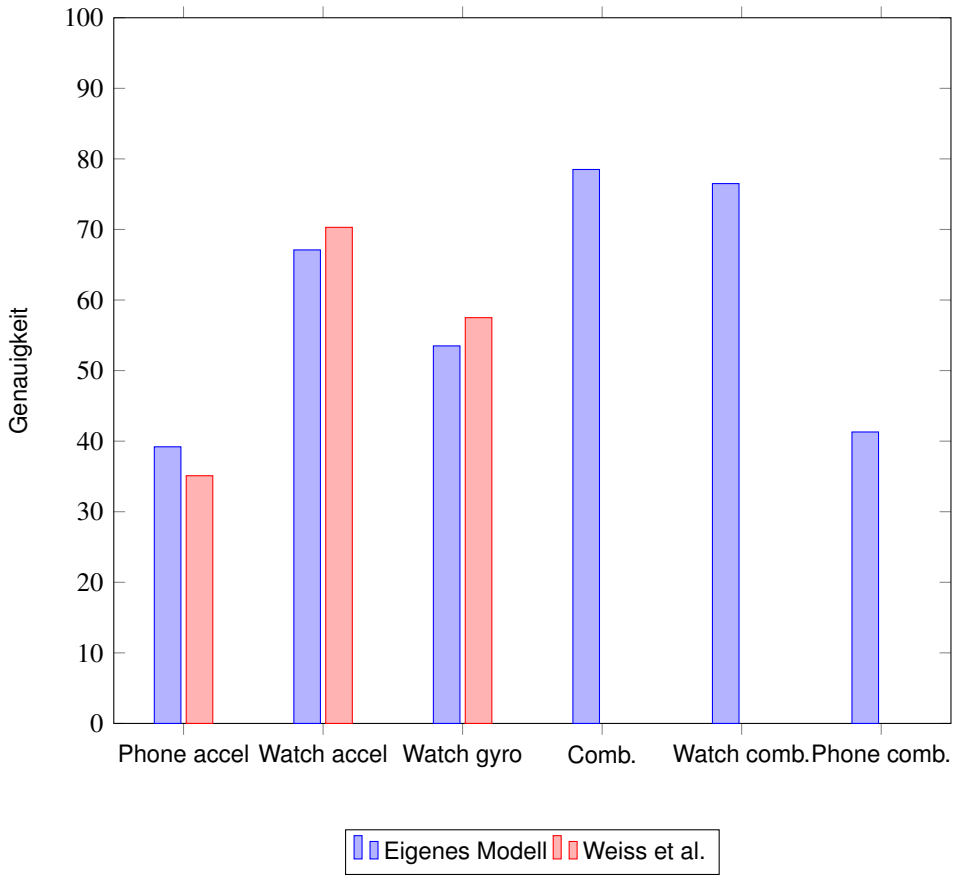

Abbildung 6.2.: Vergleich der Genauigkeiten der unpersönlichen Modelle mit Weiss et al.[18]. Die *Watch* ist in dieser Arbeit das Band.

Algo.	Phone accel	Phone gyro	Band accel	Band gyro	Comb.	Band comb.	Phone comb.
RF	88.7	**68.2**	**91.6**	**80.5**	**99.4**	**97.1**	**90.7**
J48	**90.0**	64.3	86.5	72.7	92.8	90.1	89.7
IB3	62.2	44.8	77.0	59.4	82.3	76.9	60.1
NB	87.6	60.4	90.9	78.8	96.2	92.2	85.5
MLP	78.9	51.5	88.9	67.8	94.8	90.7	75.1
∅	81.5	57.8	87.0	71.8	93.1	89.4	80.2

Tabelle 6.1.: Genauigkeit der persönlichen Modelle in Prozent

Algo.	Phone accel	Phone gyro	Band accel	Band gyro	Comb.	Band comb.	Phone comb.
RF	**39.2**	**35.1**	**75.2**	**61.7**	**78.5**	**76.5**	**41.3**
J48	32.9	29.1	61.6	49.2	59.7	61.3	32.6
IB3	24.3	22.6	60.9	45.0	52.9	58.0	26.0
NB	31.8	30.0	67.3	56.9	60.9	62.3	35.0
MLP	28.8	29.3	70.3	53.5	54.9	74.3	31.5
∅	81.5	29.2	67.1	53.3	61.4	66.5	33.3

Tabelle 6.2.: Genauigkeit der unpersönlichen Modelle in Prozent

6.1.2. Unpersönliche Modelle

Für die Evaluierung der unpersönlichen Modelle wird eine Kreuzvalidierung über die Nutzer durchgeführt, das heißt der Datensatz D wird für alle Benutzer b aufgeteilt in $D = \text{Train} \uplus \text{Test}$ mit $\text{Train} = \{\text{Intervall ist nicht von } b\}, \text{Test} = D \backslash \text{Train}$.

Das schlechteste Ergebnis wird mit 35.1% bei den unpersönlichen Modellen erzielt, wenn nur das Gyroskop des Smartphones verwendet wird. Nur wenig besser sind die Modelle, die nur den Beschleunigungssensor des Smartphones (39.2%) oder beide Sensoren des Smartphones (41.3%) nutzen. Für den praktischen Gebrauch sind diese Modelle aufgrund ihrer Ungenauigkeit untauglich. Schon rund 20% besser sind die Modelle, die nur das Gyroskop des Bands nutzen, wobei der Wechsel zum Beschleunigungssensor eine weitere Verbesserung um fast 14% ermöglicht. Offenbar erklären die Daten des Beschleunigungssensors einen Großteil der Daten, die auch durch das Gyroskop und die anderen Sensoren des Band erklärt werden, sodass eine Kombination aller Band-Sensoren lediglich ein Verbesserung um rund 1% ermöglicht. Selbiges gilt auch für die Hinzunahme der Daten des Smartphones: Es wird eine Verbesserung um 2% erzielt, was darauf schließen lässt, dass die Armbewegungen unter den Probanden ähnlich sind, die Bewegungen auf Hüfthöhe hingegen weniger. Gegenüber dem besten Ergebnis eines einzelnen Sensors (*Band accel*) mit 75.2% liefert die Kombination aller Daten eine Verbesserung um 3.3%.

6.2. Genauigkeit bei ungenauen Zeitstempeln

Die Problemstellung, Daten mehrerer Quellen miteinander zu kombinieren, wirft die Frage auf, inwiefern die Synchronisierung der Daten einen Einfluss auf die Genauigkeit der resultierenden Modelle besitzt. Smartphone und Band besitzen je eine eigene Uhr, sodass

Algo.	Comb. (Pers.)	Verrauscht (Pers.)	Comb. (Unpers.)	Verrauscht (Unpers.)
RF	**99.4**	**99.2**	**78.5**	**78.3**
J48	92.8	93.5	59.7	57.3
IB3	82.3	81.5	52.9	52.9
NB	96.2	95.7	60.9	60.5
MLP	94.8	94.2	54.9	55.4
∅	93.1	92.8	61.4	60.9

Tabelle 6.3.: Genauigkeit der Modelle mit verrauschten Band-Zeitstempeln in Prozent

die Datenquellen zum selben Zeitpunkt t *Readings* mit Zeitstempeln $t + \varepsilon_{\text{Smartphone}}$, bzw. $t + \varepsilon_{\text{Band}}$ mit $|\varepsilon_{\text{Smartphone}} - \varepsilon_{\text{Band}}| > 0$ liefern könnten. Wird die Differenz zu groß, schadet die Kombination der Quellen möglicherweise mehr als sie nützt.

Ein erster Ansatz, um dieses Problem zu bekämpfen, war daher zunächst, die beiden Uhren via Bluetooth und einem Verfahren wie das *Network Time Protocol (NTP)* [14] zu synchronisieren. Leider ermöglicht das Microsoft Band SDK nicht die Ausführung beliebigen Codes auf dem Band, sondern nur das Abonnieren von Sensordaten-Events, weshalb kein direkter Zugriff auf die Uhr des Gerätes möglich ist. Die Events sind zwar mit Zeitstempeln versehen, jedoch kann die Übertragungszeit nicht ermittelt werden. Eine manuelle Synchronisierung der Uhren ist daher nicht möglich.

Um zu prüfen, inwiefern die Modelle durch eine mögliche, nicht detektierbare Differenz der Uhren beeinträchtigt werden, generiert die Transformationssoftware zusätzlich zum normalen vollständigen Datensatz auch einen, in dem die Zeitstempel des Bands additiv mit gausschem Rauschen versehen wurden. Die Parameter des gausschen Rauschens wurden auf einen Mittelwert von 0 und eine Standardabweichung von 500ms festgelegt. Die Ergebnisse der Auswertung werden in Tabelle 6.3 aufgeführt. Betrachtet man die jeweils besten Modelle, so lässt sich sowohl bei den persönlichen, als auch bei den unpersönlichen Modellen eine Verschlechterung um lediglich 0.2% feststellen. Auch die anderen Modelle sind im Durchschnitt mit Verschlechterungen um 0.3%, bzw. 0.5% hinreichend resistent gegen das Rauschen.

Um zu prüfen, ob eine Standardabweichung von 500ms sinnvoll ist, wurde über drei Tage hinweg um jeweils 10, 15 und 18 Uhr geprüft, mit welchem zeitlichen Abstand das Umspringen von der vollen Stunde auf die nächste Minute erfolgte. Eine Verzögerung war in keinem der Fälle mit dem Auge beobachtbar, weshalb davon auszugehen ist, dass die Uhren aufgrund der bereits bestehenden Synchronisierung durch die vom Hersteller gelieferte Band-App für Android in der Regel weniger als 500ms voneinander abweichen. Dies führt zu der Annahme, dass die Genauigkeit der Zeitstempel im Rahmen dieser Arbeit keinen negativen Einfluss auf die Ergebnisse hat und somit nicht weiter betrachtet werden muss.

6.3. Genauigkeit bei einer niedrigen Sampling-Rate

Da beide verwendete Geräte primär akkubetrieben sind, muss bei einer marktreifen Umsetzung einer Aufnahmesoftware auch bedacht werden, dass das Sammeln der Daten hinsichtlich der Energieressourcen nicht kostenlos ist. Eine Rolle beim Energieverbrauch spielt auch die Sampling-Rate (Abtastrate): Ist sie hoch eingestellt, können die entsprechenden Sensoren

Gerät	Sensor	Durchsch. Sampling-Rate
Smartphone	Accelerometer	200 Hz
Smartphone	Gyroskop	200 Hz
Fitness-Tracker	Accelerometer	8 Hz
Fitness-Tracker	Gyroskop	8 Hz
Fitness-Tracker	Laufgeschwindigkeit	1 Hz
Fitness-Tracker	Hautwiderstand	5 Hz

Tabelle 6.4.: Sampling-Raten der Sensoren

Algo.	5 Hz (Pers.)	1 Hz (Pers.)	5 Hz (Unpers.)	1 Hz (Unpers.)
RF	**98.6**	**95.3**	**70.1**	**62.1**
J48	93.8	87.8	53.4	48.7
IB3	71.0	14.6	43.5	7.8
NB	91.6	75.5	53.3	44.2
MLP	90.0	78.9	63.6	51.2
∅	89.0	70.4	56.8	42.8

Tabelle 6.5.: Genauigkeit der Modelle mit reduzierter Sampling-Rate in Prozent

nicht in einen Energiesparmodus wechseln und außerdem müssen von der anfordernden Anwendung mehr Daten verarbeitet werden, wodurch wiederum Anforderungen an den Prozessor gestellt werden [12]. Beim Band kommt hinzu, dass eine höhere Bluetooth-Datenrate in Kauf genommen werden muss, die insbesondere auf den bauartbedingt kleinen Akku des Fitness-Trackers einen negativen Einfluss hat. Aus diesem Grund ist es untersuchenswert, inwiefern die Sampling-Rate einen Einfluss auf die Genauigkeit der Modelle besitzt, um möglicherweise Energie sparen zu können.

Die Sampling-Raten, mit denen die Aufnahmesoftware Daten aufgezeichnet hat, befinden sich in Tabelle 6.4. Um Sampling-Raten miteinander zu vergleichen, wurden dieselben Rohdaten verwendet wie im normalen, vollständigen Modell. Um eine Sampling-Rate von k Hz zu simulieren, wurden nach einem behaltenen Reading $(t_0 \,\text{secs}, \text{sensor}, \text{source}, ...)$ alle weiteren Readings $(t \,\text{secs}, \text{sensor}, \text{source}, ...)$ mit $t \in [t_0, t_0 + 1/k]$ gelöscht.

Nun werden die besten Modelle miteinander verglichen. Zur Erinnerung: *Comb. (Pers.)* erzielte eine Genauigkeit von 99.4% und *Comb. (Unpers.)* eine Genauigkeit von 78.5%. Reduziert man die Sampling-Rate des persönlichen Modells auf 5 Hz, so reduziert sich die Genauigkeit um 0.8%. Eine dramatischere Verschlechterung wird durch die Reduzierung auf 1 Hz verursacht, durch die sich die Genauigkeit um 4.1% verschlechtert. Noch stärker beeinträchtigt werden durch die Reduzierung der Sampling-Rate die unpersönlichen Modelle. Beschränkt man diese auf 5 Hz, so ergibt sich gegenüber *Comb. (Unpers.)* eine Verschlechterung um 8.4%. Bei einer Beschränkung auf 1 Hz beträgt die Verschlechterung sogar 16.4%, sodass zu einer Reduzierung der Sampling-Rate nur bei den ohnehin bereits genauen persönlichen Modellen zu raten ist.

Algo.	Comb. (Pers.)	Überlappung (Pers.)	Comb. (Unpers.)	Überlappung (Unpers.)
RF	**99.4**	**99.2**	**78.5**	**73.4**
J48	92.8	96.3	59.7	57.5
IB3	82.3	81.8	52.9	50.7
NB	96.2	95.8	60.9	59.7
MLP	94.8	95.0	54.9	59.4
∅	93.1	93.6	61.4	60.9

Tabelle 6.6.: Genauigkeit der Modelle mit Intervallüberlappung in Prozent

Algo.	Comb. (Unpers.), verschmolzen	Comb. (Unpers.), nicht verschmolzen
RF	**87.1**	**78.5**
J48	73.7	59.7
IB3	63.9	52.9
NB	72.4	60.9
MLP	77.6	54.9
∅	74.9	61.4

Tabelle 6.7.: Genauigkeit der Modelle bei Verschmelzung der Ess- und Trinkaktivitäten in Prozent

6.4. Genauigkeit bei Überlappung der Intervalle

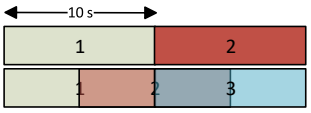

Abbildung 6.3.: Intervallüberlappung

Bao et al. nutzten 2004 auf der Basis bereits bestehender Werke für die Transformation eine Intervallüberlappung von 50% [3]. Weiss et al. probierten dies laut ihrem Paper nicht aus [18]. Um zu prüfen, ob dies eine weitere Verbesserung der Genauigkeiten erzielen könnte, wurden für die Intervalle $(t_0, t_0+10), ((t_0+10)-10*0.5, (t_0+10)-10*0.5+10), ...$ Features generiert und wie zuvor evaluiert. Eine Illustration der Überlappung befindet sich in Abbildung 6.3.

Anders als erhofft verschlechterten sich die persönlichen RF-Modelle wie in Tabelle 6.6 ersichtlich durch die Überlappung um 0.2%, während sich die unpersönlichen Modelle sogar um 5.1% verschlechterten.

6.5. Genauigkeit bei Verschmelzung der Ess- und Trinkaktivitäten

Betrachtet man die Konfusionsmatrix der vollständigen unpersönlichen RF-basierten Modelle in Tabelle 6.8, so lässt sich feststellen, dass die Klassen, die sich mit Essen oder Trinken

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	< Output für >
149	0	0	0	0	0	2	0	0	8	0	0	0	2	0	0	0	0	**a = Brushing Teeth**
0	153	0	7	0	0	0	0	0	0	0	0	0	1	0	0	0	0	**b = Clapping**
0	0	126	0	0	0	0	0	0	0	0	1	17	0	0	0	0	26	**c = Climbing Stairs**
0	11	0	147	0	0	0	0	0	1	0	0	0	5	0	0	0	0	**d = Basketball**
0	0	0	0	133	6	0	9	0	0	0	0	0	0	12	0	3	0	**e = Drinking**
1	0	0	0	5	135	3	15	4	0	0	0	0	0	0	0	1	0	**f = Eating Chips**
3	0	0	0	0	11	106	3	24	0	3	0	0	0	2	0	9	0	**g = Eating Pasta**
3	0	0	0	14	26	15	75	27	0	6	0	0	0	7	1	3	0	**h = Eating Sandwich**
7	0	0	0	0	13	39	44	84	0	0	0	0	0	3	0	2	0	**i = Eating Soup**
6	0	0	0	0	0	0	0	0	148	0	0	3	2	0	0	0	0	**j = Folding Clothes**
0	0	0	0	0	5	7	6	2	0	139	0	0	0	0	0	4	0	**k = Handwriting**
0	0	2	0	0	0	0	0	0	0	0	166	0	0	0	0	0	0	**l = Jogging**
0	0	13	0	0	0	0	0	0	0	0	1	150	0	0	0	0	2	**m = Soccer Ball**
0	0	0	7	0	0	0	0	0	0	0	0	3	145	0	0	0	0	**n = Playing Catch**
4	0	0	0	1	2	2	1	1	0	4	0	0	0	136	8	5	0	**o = Sitting**
1	0	0	0	0	0	0	1	0	0	0	0	1	0	15	142	0	0	**p = Standing**
2	0	0	0	1	1	4	1	0	0	27	0	0	0	0	0	125	0	**q = Typing**
0	0	66	0	0	0	0	0	0	0	0	0	19	0	0	0	0	84	**r = Walking**

Tabelle 6.8.: Konfusionsmatrix der unpersönlichen RF-Modelle

beschäftigen, nur ungenau erkannt werden und somit die Durchschnittsgenauigkeit entsprechend senken. Zudem liegt die fehlerhaft vorhergesagte Klasse in $298/324 \approx 92\%$ der Fälle im Bereich des Essens und Trinkens. Zurückzuführen ist dies auf die Tatsache, dass sich diese Aktivitäten untereinander ähneln, da die zentralen Bewegungsmerkmale bei all diesen Klassen das Sitzen sowie das Führen der Hand zum Mund sind.

Tabelle 6.7 zeigt, wie sich die Genauigkeit der unpersönlichen Modelle verändert, wenn man diese Aktivitäten zu einer einzigen Aktivität verschmilzt. Bei den RF-Modellen ist eine Verbesserung um 8.6% zu erkennen, wovon ein Teil auch darauf zurückzuführen sein könnte, dass es insgesamt weniger Klassen gibt und die Wahrscheinlichkeit einer Fehlklassifikation somit sinkt. Tabelle 6.9 zeigt die neue Konfusionsmatrix der RF-Modelle und es wird deutlich, dass aufgrund der verschmolzenen Aktivität *Eating* die Fehlerrate stark gesunken ist.

Aus diesen Daten lässt sich schließen, dass bei der Verwendung von unpersönlichen Modellen darauf geachtet werden sollte, dass die zu erkennenden Aktivitäten nicht zu feingranular voneinander getrennt werden, da dies die Klassifikation stark erschweren kann.

6.6. Genauigkeit in Abhängigkeit von der Teilnehmeranzahl

Während die persönlichen Modelle lediglich auf die Bewegungsdaten einer einzelnen Person zurückgreifen, sind für die unpersönlichen Modelle Daten anderer Personen erforderlich. Daraus ergibt sich die Frage, wie die Genauigkeit der unpersönlichen Modelle von der Anzahl der Personen, auf denen dieses basiert, abhängig it. Um dies zu ermitteln, wurden aus dem aus 10 Personen bestehenden vollständigen Datensatz D für $i \in \{2,...,10\}$ je $N_i := \min(10, \binom{10}{i})$ verschiedene, zufällig gezogene Datensätze $D_{i,j}$ mit $j \in \{1,...,N_i\}$ generiert. Ein Datensatz $D_{i,j}$ enthält Daten von $i \geq 2$ verschiedenen Personen, um die Aufteilung in Train- und Testsets zu ermöglichen, da ein Testset für die unpersönlichen Modelle genau eine Person enthalten soll und die Mengen disjunkt sein müssen (vgl. Abschnitt 6.1.2). Die Genauigkeit in Abhängigkeit von der Anzahl der Personen i ist dann definiert durch $G(i) := \sum_{j=1}^{N_i} \frac{\text{Genauigkeit mit } D_{i,j}}{N_i}$. Die Berechnung der Genauigkeit mit Datensatz $D_{i,j}$ erfolgt nach Abschnitt 6.1.2. TODO: Visuelle Illustration des Ziehens von Datensätzen Abbildung 6.4 zeigt den Plot der Funktion G. Es ist zu erkennen, dass die Genauigkeit langsam konvergiert, wobei für $i > 10$ noch etwas höhere Genauigkeiten zu erwarten sind. Die Verifikation dieser Hypothese durch weitere Aufnahmen war im Rahmen der zeitlichen Beschränkungen einer Bachelorarbeit nicht möglich.

TODO: Check if this figure is in the right place

6.7. Optimierung der Hyperparameter

Um die unpersönlichen Modelle weiter zu verbessern, wurden die Hyperparameter des RF-Modells optimiert, da dieses Verfahren bis dato das beste Ergebnis geliefert hat. Als Baseline dient das RF1-Modell, das dieselben Hyperparameter besitzt wie das RF-Modell in Abbildung 6.2. Zu bemerken ist, dass die folgenden Genauigkeiten je nach Seed für den Zufallsgenerator um $\pm 1.5\%$ schwanken, weshalb die Genauigkeit des RF1-Modells im hier gezeigten Durchlauf des Tests rund 1.45% über der des RF-Modells in Abbildung 6.2 liegt.

Die variierten Hyperparameter sind die folgenden:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	< Output für ∨
143	0	0	0	5	12	0	0	0	1	0	0	0	0	**a = Brushing Teeth**
0	155	0	5	0	0	0	0	0	1	0	0	0	0	**b = Clapping**
0	0	126	0	0	0	0	1	13	0	0	0	0	30	**c = Climbing Stairs**
0	5	0	136	0	5	0	0	0	18	0	0	0	0	**d = Basketball**
2	0	0	0	830	0	2	0	0	0	16	0	7	0	**e = Eating**
2	0	0	0	0	155	0	0	0	2	0	0	0	0	**f = Folding Clothes**
0	0	0	0	30	0	124	0	0	0	0	0	9	0	**g = Handwriting**
0	0	0	0	0	0	0	168	0	0	0	0	0	0	**h = Jogging**
0	0	6	0	0	0	0	1	157	1	0	0	0	1	**i = Soccer Ball**
0	0	0	7	0	2	0	1	2	143	0	0	0	0	**j = Playing Catch**
1	0	0	0	23	0	2	0	0	0	130	8	0	0	**k = Sitting**
0	0	0	0	5	0	0	0	0	0	17	138	0	0	**l = Standing**
1	0	0	0	37	0	18	0	0	0	0	0	105	0	**m = Typing**
0	0	60	0	0	0	0	0	26	0	0	0	0	83	**n = Walking**

Tabelle 6.9.: Konfusionsmatrix der unpersönlichen, verschmolzenen RF-Modelle

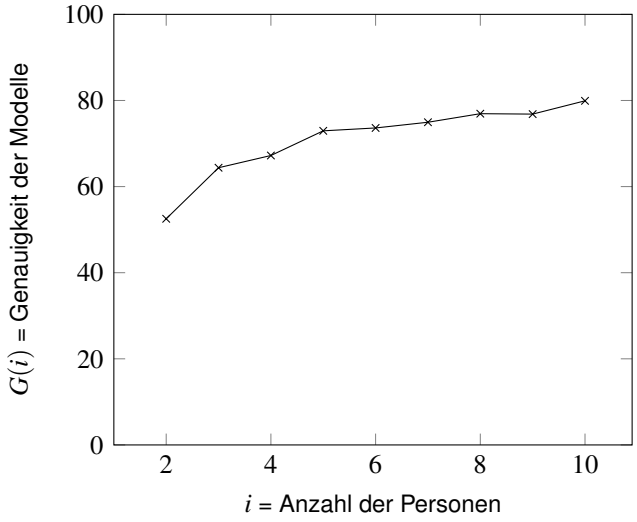

Abbildung 6.4.: Genauigkeit in Abhängigkeit von der Personenanzahl

- *numIterations* gibt die Anzahl der Entscheidungsbäume im Wald an. Ein zu niedriger Wert sorgt dafür, dass die zufällig generierten Bäume den Datensatz zusammen nicht gut genug abdecken können.

- *numFeatures* gibt die Anzahl der Features an, die jeder Entscheidungsbaum bei einem Split betrachtet. Wird ein zu hoher Wert verwendet, ist die Entwicklung des Baums weniger vom Zufall getrieben. Ist der Wert zu niedrig, so sinkt die Wahrscheinlichkeit, dass ein Split an einem Feature durchgeführt wird, sodass der Information Gain groß ist.

- *maxDepth* bestimmt die maximale Tiefe der Entscheidungsbäume im Wald. Ein niedriger Wert sorgt dafür, dass der Baum sich den Trainingsdaten nicht zu sehr anpasst und overfittet. Dabei muss beachtet werden, dass ein zu niedriger Wert wiederum dafür sorgen kann, dass zu wenige Features betrachtet werden können, um eine genaue Klassifizierung zu ermöglichen.

Tabelle 6.10 zeigt die Genauigkeiten in Abhängigkeit von den Hyperparametern. In einigen Fällen wurde für *numFeatures* der WEKA-Standardwert $f = \lfloor log_2(n-1)+1 \rfloor = \lfloor log_2(208)+1 \rfloor = 8$ verwendet wird, wobei n die Anzahl der Features einer Instanz ist.

Gegenüber der Baseline ist das RF8-Modell das beste, allerdings nur weniger als 1.3% besser, sodass die Differenz unter den Veränderungen durch Schwankungen liegt. Es ist jedoch festzustellen, dass der automatisch gewählte Wert f für *numFeatures* die besten Ergebnisse erzielt. Eine Erhöhung der maximalen Tiefe auf 50 oder gar 75 erzielt keine nennenswert besseren Ergebnisse als die oben gewählte Tiefe von 25. Auch für die Anzahl der Bäume gilt, dass bereits 50 annähernd so gut sind wie 100 oder mehr. Positiv zu vermerken ist somit, dass in diesem Anwendungsfall von Random Forests auch kleine Werte eine gute Genauigkeit liefern, wobei der Leistungsbedarf entsprechend niedriger ist.

	numIterations	numFeatures	maxDepth	Genauigkeit
RF8	75	f	50	81.229
RF7	100	f	75	80.7925
RF4	100	f	50	80.6917
RF6	150	f	50	80.1545
RF9	100	f	30	80.0537
RF1	50	10	25	79.953
RF2	200	∞	∞	73.5729
RF5	200	∞	50	73.2371
RF3	200	∞	50	72.6998

Tabelle 6.10.: Hyperparameteroptimierung

6.8. Einfluss von Feature-Selection

Betrachtet man die *Comb.*-Spalte in Tabelle 6.2, so fällt auf, dass die Genauigkeiten der übrigen Modelle dem RF-Modell deutlich unterlegen sind. Da dies bei den persönlichen Modellen nicht der Fall ist, ist es untersuchenswert, ob die anderen Algorithmen nicht durch eine Vorauswahl der Features (*Feature Selection*) höhere Genauigkeiten liefern können, da eine solche Auswahl prinzipiell auch durch den Zufall im Random Forest erfolgt.

Zur Feature Selection wurde das Verfahren *CfsSubsetEval*[8] verwendet, das laut WEKA-Dokumentation wie folgt arbeitet:

> "[CfsSubsetEval] ermittelt den Wert einer Untermenge der Features, indem die individuelle Vorhersagefähigkeit eines jeden Features mitsamt dem Grad der Redundanz zwischen ihnen betrachtet wird. Feature-Untermengen, die stark mit der Klasse korrelieren und dabei eine niedrige Interkorrelation haben, werden bevorzugt."
>
> *– aus [1] übersetzt.*

Begonnen wurde mit der leeren Menge, der nach und nach Features hinzugefügt worden sind, bis die Bewertung durch *CfsSubsetEval* sank.

Unter Einsatz der Feature Selection entstanden die Ergebnisse aus Tabelle 6.11. Gegenüber Tabelle 6.2 sind die Genauigkeiten der IB3-, NB- und MLP-Modelle wesentlich besser, wodurch auch der Durchschnittswert angehoben wird, jedoch schlägt weiterhin kein Algorithmus den RF-Algorithmus. Diesen hat die Feature Selection des Weiteren nicht verbessert, sodass sich die Feature Selection in diesem Anwendungsszenario als nicht sinnvoll erwiesen hat.

Algo.	Comb. (Unpers.)
RF	78.2
J48	60.0
IB3	59.6
NB	67.8
MLP	68.3
∅	66.8

Tabelle 6.11.: Genauigkeit mit Feature Selection

7. Conclusions

TODO: Check template for suggestions on how to write this

Anhang

A. What goes in the appendices

B. Used Acronyms

C. Literaturverzeichnis

[1] Weka javadoc: Cfssubseteval, 02 2017.

[2] D. Aha and D. Kibler. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[3] Ling Bao and Stephen S Intille. Activity recognition from user-annotated acceleration data. In *International Conference on Pervasive Computing*, pages 1–17. Springer, 2004.

[4] Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[5] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[6] Fitbit. How accurate are fitbit trackers? https://help.fitbit.com/articles/en_US/Help_article/1136. Abgerufen am 20.01.2017.

[7] Google Inc. *Android Sensors Overview*, 12 2016.

[8] M. A. Hall. *Correlation-based Feature Subset Selection for Machine Learning*. PhD thesis, University of Waikato, Hamilton, New Zealand, 1998.

[9] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2009.

[10] Simon Haykin. Neural networks, a comprehensive foundation. -, 1994.

[11] George H. John and Pat Langley. Estimating continuous distributions in bayesian classifiers. In *Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 338–345, San Mateo, 1995. Morgan Kaufmann.

[12] A. Krause, M. Ihmig, E. Rankin, D. Leong, Smriti Gupta, D. Siewiorek, A. Smailagic, M. Deisher, and U. Sengupta. Trading off prediction accuracy and power consumption for context-aware wearable computing. In *Proc. Ninth IEEE Int. Symp. Wearable Computers (ISWC'05)*, pages 20–26, October 2005.

[13] Jennifer R. Kwapisz, Gary M. Weiss, and Samuel A. Moore. Activity recognition using cell phone accelerometers. *SIGKDD Explor. Newsl.*, 12(2):74–82, March 2011.

[14] D Mills, J Martin, J Burbank, and W Kasch. Rfc5905. *Network Time Protocol Version*, 4:2010–06, 2010.

[15] AY Ng. Cs229 lecture notes on machine learning. Technical report, Technical report, Stanford University, Department of Computer Science, Stanford, USA, 2011. 39, 116, 140, 2011.

[16] Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[17] B. N. Taylor and A. Thompson. The international system of units (si). National Institute of Standards and Technology, Special Publication 330, Gaithersburg, MD, 2008.

[18] G. M. Weiss, J. L. Timko, C. M. Gallagher, K. Yoneda, and A. J. Schreiber. Smartwatch-based activity recognition: A machine learning approach. In *Proc. IEEE-EMBS Int. Conf. Biomedical and Health Informatics (BHI)*, pages 426–429, February 2016.

Erklärung der Urheberschaft

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Ort, Datum

Unterschrift