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

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 [11] 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 2.2) 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 [11], 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 [11] 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

TODO: Grafiken zur Erläuterung Während in der Einleitung nur generisch von "Sensordaten" die Rede war, werden diese nun konkretisiert. Zur Aufnahme werden das Smartphone OnePlus 3 sowie der Fitness-Tracker Microsoft Band 2 verwendet. 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 $g \approx 9.81\ m/s^2$[4][10]. Über diesen Sensor kann demnach die Orientierung des Objektes im Raum bestimmt werden.
Die Daten dieses Sensors werden von beiden Geräten aufgezeichnet.

2.1.2. Gyroskop

Ein Gyroskop misst die Rotationsgeschwindigkeit je physikalischer Achse[4]. Demnach bedeutet der Messwert $(x = 0, y = 0, z = 0)$, dass das Objekt relativ zur Erde nicht bewegt wird, während $(x = 1, y = 0, z = 0)$ bedeutet, dass das Objekt um die x-Achse gedreht 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.

TODO: Pedometer, Distance

2.2. Grundlagen der Machine Learning-Klassifikation

In den folgenden Abschnitten wird von einem grundlegenden Verständnis von Machine Learning-Klassifikation ausgegangen, weshalb wir diese Technik nun kurz erläutern.

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 *ist 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[8].*

Ein einfaches Beispiel kann wie folgt konstruiert werden: Sei $x = (x_1)$, wobei x_1 die Wohnfläche einer Wohnung oder eines Hauses in m^2 ist. Des Weiteren sei $D = \{(50, \text{Wohnung}), (75, \text{Wohnung}), (200, \text{Haus}), (3$
Eine sinnvolle Hypothesenfunktion wäre in diesem Fall beispielsweise gegeben durch:

$$h(x) = \begin{cases} \text{Wohnung}, & \text{wenn } x_1 < 200 \\ \text{Haus}, & \text{sonst} \end{cases}$$

Das beschriebene Klassifikationsproblem kann mittels Algorithmen des *Machine Learnings* (im Folgenden *ML*) gelöst werden, indem eine Verlustfunktion definiert wird, die ein solcher Algorithmus zu minimieren versucht. Eine solche Verlustfunktion bestraft die fehlerhafte Klassifikation einer Instanz, indem sie einen höheren Wert annimmt. Die Ausführung des ML-Algorithmus wird auch *Lernen* genannt.

Wenn ein ML-Algorithmus auf einem Datensatz arbeitet, besteht die Gefahr, dass er diesen gewissermaßen "auswendig lernt", d.h. sich diesem zu sehr anpasst und auf unbekannten Daten schlechte Vorhersagen liefert. Dieses Problem wird *Overfitting* genannt und kann erkannt werden, indem der Datensatz D in zwei disjunkte Mengen $D = \text{Train} \uplus \text{Test}$ aufgeteilt wird (*Train-Test-Split*). Der ML-Algorithmus wird nun auf den Trainingsdaten ausgeführt, woraufhin dessen Ergebnis mittels den Testdaten auf Overfitting geprüft wird.

Ein übliches Verfahren ist dabei die k-Kreuzvalidierung. Mit $D = D_1 \uplus ... \uplus D_k$ wird der ML-Algorithmus k mal ausgeführt, wobei in Iteration i gilt, dass $\text{Train} = D \setminus D_i$ und $\text{Test} = D_i$. Effektiv bedeutet dies, dass dem Algorithmus in jeder Iteration ein anderer Teil der Daten vorenthalten wird. Das Ergebnis der Kreuzvalidierung ist die Hypothese, die auf den jeweiligen Testdaten das beste Ergebnis liefert.

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 **??** achtzehn gelisteten Aktivitäten durchgeführt. Weiss et al. nahmen jeweils zwei Minuten pro Teilnehmer und Aktivität auf[11], 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 [11] 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 TODO: cite, 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 oder dies sogar bevorzugen. TODO: cite https://help.fitbit.com/articles/en_US/Help_article/1136

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 [11] 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

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* $(1000, \mathit{Band}, \mathit{Gyroscope}, \{x \to 0.0, y \to 1.0, z \to 1.0\})$.

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.

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

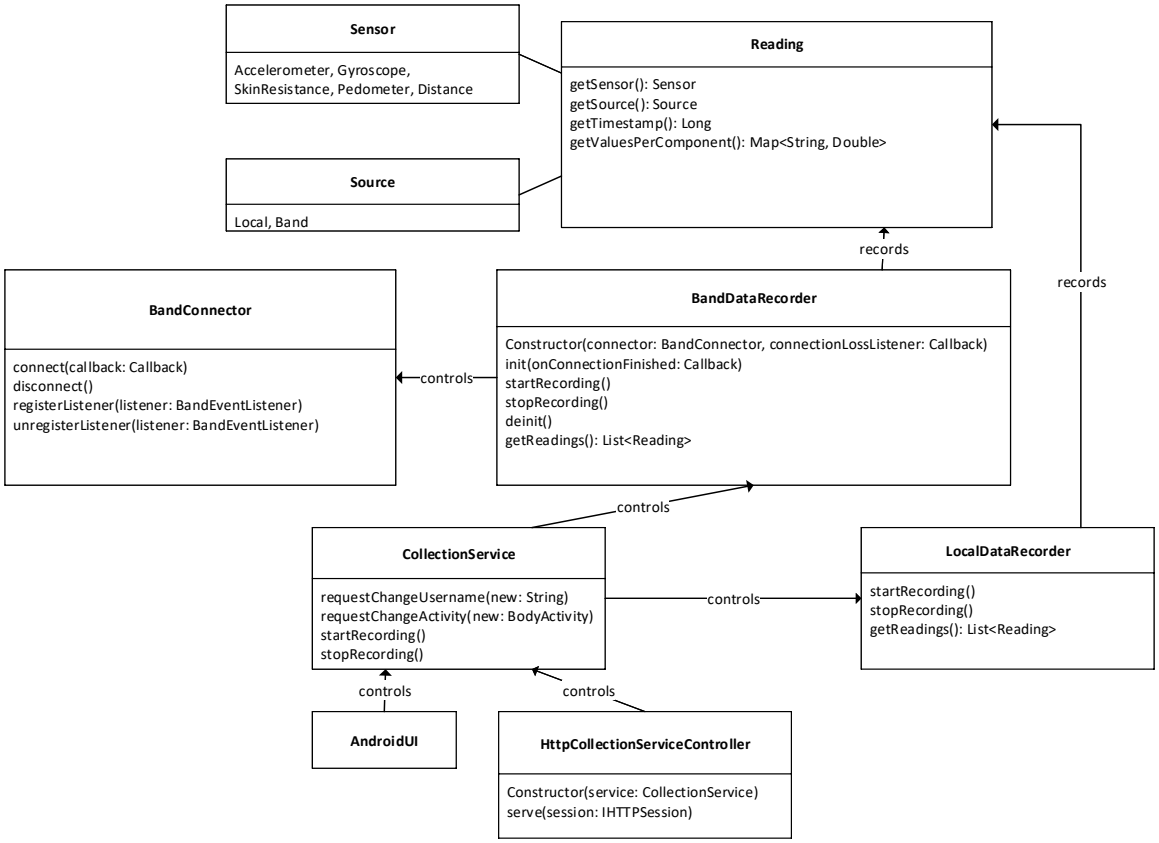

Abbildung 5.1.: Struktur der Aufzeichnungssoftware

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 unserem Fall 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.[7] 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 von uns gesammelten Daten als hinreichend. 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.

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.

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

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)

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 Instanzerzeugung verwenden wir als Aggregatfunktionen den Durchschnitt, die *Binned Distribution* mit zwei *Bins* und den *Average Root of Squares*. 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. Anwendung von ML-Klassifikationsalgorithmen

Um die Vergleichbarkeit mit [11] 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 [11] 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

5.3.1. Random Forest

Definition 3 (Entscheidungsbaum)**.** *Ein Entscheidungsbaum ist ein Baum, dessen innere Knoten, auch Entscheidungsknoten genannt, je ein Entscheidungskriterium beinhalten, um eine Menge von Instanzen zu unterteilen und schließlich genau einer Klasse zuordnen zu können. Ein solches Kriterium könnte beispielsweise lauten:* $Avg_{Band,\ Gyroscope,\ x} > 0$*. Ein Blattknoten gibt die Klasse an, die der Entscheidungsbaum einer Instanz zuordnet.*

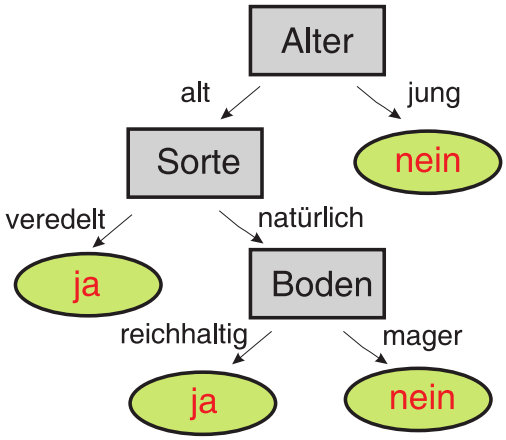

Abbildung 5.2.: Ein binärer Entscheidungsbaum mit Klassen *Ja* und *Nein*, ob ein Apfelbaum Früchte tragen wird
TODO: cite https://de.wikipedia.org/wiki/Datei:Entscheidungsbaum.svg

Ein *Random Forest* ist ein Wald von Entscheidungsbäumen, 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 [3]. 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.

5.3.2. J48-Entscheidungsbaum

J48 ist eine Implementierung des *C4.5*-Algorithmus[9]. 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.

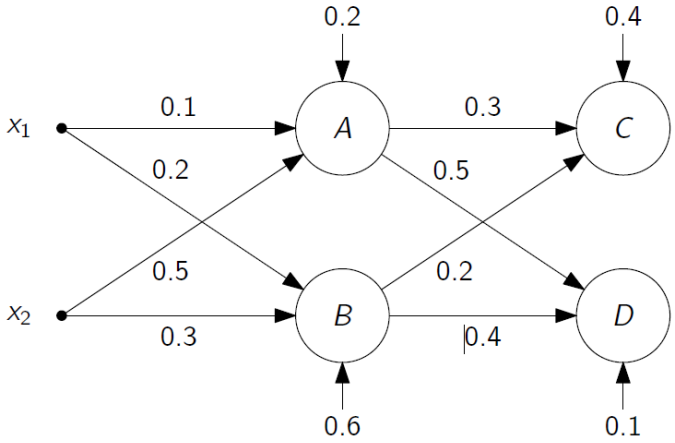

Abbildung 5.3.: Ein zweischichtiges neuronales Netzwerk aus [2]

5.3.3. 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 [1]. k ist dabei ein Hyperparameter, der auf 3 eingestellt wurde.

5.3.4. Naive Bayes

Naive Bayes ist eine probabilistische Methode zur Klassifikation[6]. 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 Einsetzen 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 $x_i \in I$.

5.3.5. Multilayer Perceptron

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. Weitere Informationen sind zu finden in [5]. Ein Beispiel ist in Abbildung 5.3 zu finden.

Zur Klassifikation können die Ausgangswerte der Perzeptren der letzten Schicht als Indikatorvariablen für die vorhandenen Klassen interpretiert werden.

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. *All*: Alle Daten werden ohne Veränderungen mit in die Transformation einbezogen.
2. *NoisyBandTimestamps*: Die Zeitstempel der Readings des Bands werden mit gaussschem Rauschen versehen
3. *OnlyBand*: Alle Daten des Smartphones werden vor der Transformation verworfen.
4. *OnlyBandAccelerometer*: Nur die Daten des Beschleunigungssensors des Microsoft Band 2 werden mit in die Transformation einbezogen.
5. *OnlyBandGyroscope*: Nur die Daten des Gyroskops des Microsoft Band 2 werden mit in die Transformation einbezogen.
6. *OnlyLocal*: Alle Daten des Microsoft Band 2 werden vor der Transformation verworfen.
7. *OnlyLocalAccelerometer*: Nur die Daten des Beschleunigungssensors des Smartphones werden mit in die Transformation einbezogen.
8. *OnlyLocalGyroscope*: 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

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

Abbildung 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

Abbildung 6.2.: Genauigkeit der unpersönlichen Modelle in Prozent

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

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

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

Abbildung 6.4.: Genauigkeit der Modelle mit reduzierter Sampling-Rate in Prozent

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

Abbildung 6.5.: Genauigkeit der Modelle mit Intervallüberlappung in Prozent

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

[2] Michael Baumann. Neuronale netze. Slides, AIS-BI, 2016.

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

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

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

[6] 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.

[7] 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.

[8] 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.

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

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

[11] 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