UiT
THE ARCTIC
UNIVERSITY
OF NORWAY

Faculty of Engineering Science and Technology

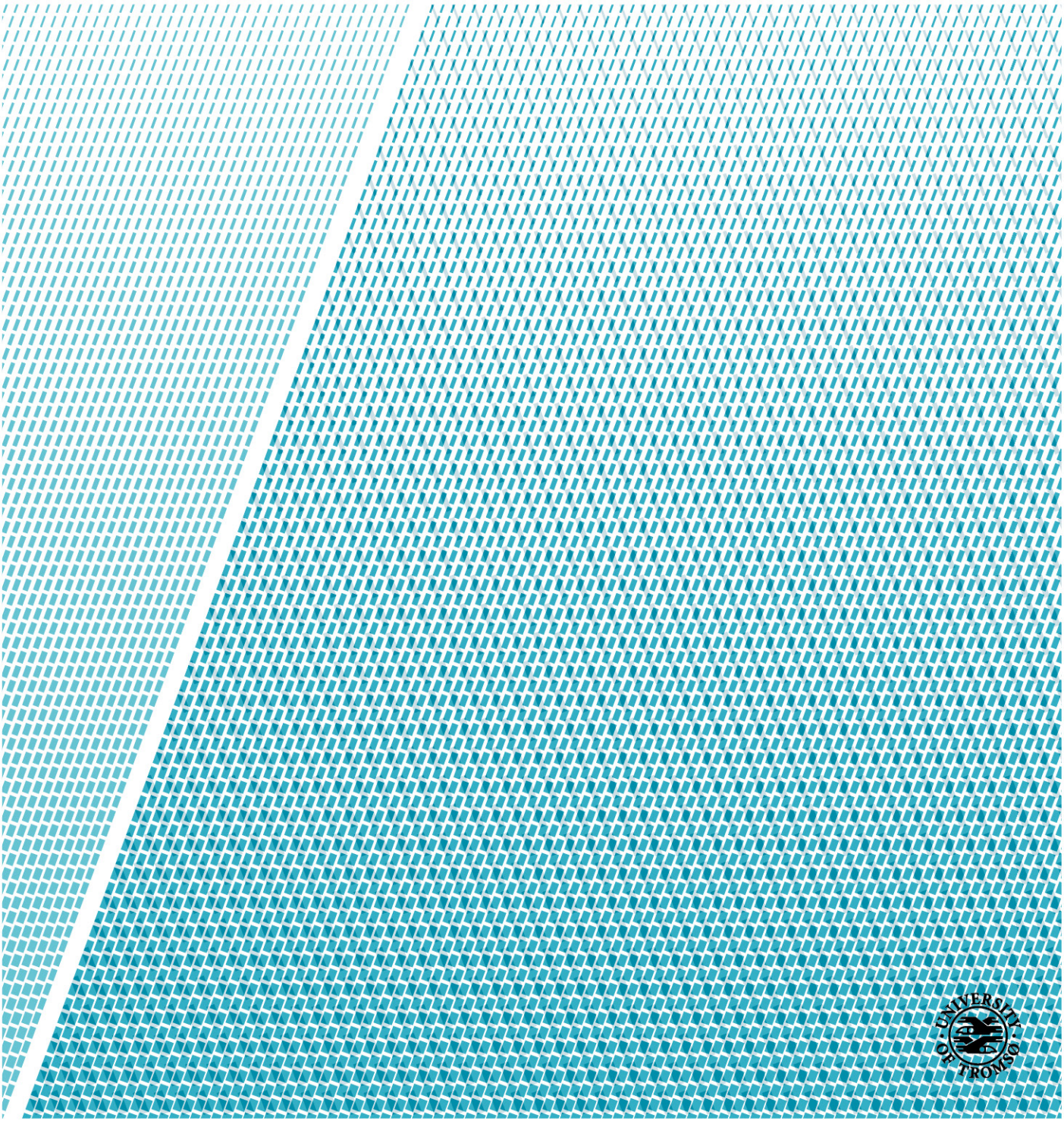
Department of Computer Science and Computational Engineering

Detection and prediction of falls among elderly people using walkers

—

Aleksei Degtiarev

Master's thesis in Computer Science - June 2018



Title: Detection and prediction of falls among elderly people using walkers

Author: Aleksei Degtiarev

Date: June 2018

Classification: Open

Pages: 120

Attachments: Zip file

Department: Department of Computer Science and Computational Engineering

Study: Master of Science, Computer Science

Student number: 166121

Course code: SHO6264 Diploma Thesis - M-IT

Supervisors: Bernt A. Bremdal, Asbjørn Danielsen

Principal: UiT - The Arctic University of Norway (Campus Narvik)

Principal contact: Bernt A. Bremdal

Keywords: CC2650, SensorTag 2.0, iPhone, Apple Watch, iOS, watchOS, Core ML, Core Motion, Core Bluetooth, Gyroscope, Accelerometer, Magnetometer, motion capture, data analysis, neural network, Bluetooth Low Energy, machine learning, walkers

Abstract (English): Falls of elderly people are big health burden, especially for long-term consequence. Yet we already have research, describing how exactly elderly fall and reasons of falls. We aimed to develop means that could not only detect falls and send alerts to relatives and doctors to conquer one of the biggest fears of elderly to fall and do not have the ability to call for help, but also tried to implement fall prevention system. This system based on "relatively safe walking patterns" that our system tries to detect during the walk. During the work we used SensorTag 2.0 CC2650 sensors, iPhone and Apple Watch to collect motion data (Gyroscope, Accelerometer and Magnetometer) and compared the accuracy of each device. As we chosen iPhone and Apple Watch to use Core ML framework to integrate the neural network model we generated using Keras into prototype app. The iPhone app perfectly detects falls, but it needs to collect data more accurately, to improve the machine learning model to improve the work of prediction falls. The Apple Watch app does not work acceptable, despite well prepared Keras model and requires revision.

Acknowledgements

I would like to thank my supervisor **Bernt A. Bremdal** for providing the basis for the project and giving me the opportunity to create my own vision on it and freedom in the choice of hardware, software and tools for this work. I would also like to thank him for his suggestions and feedback during the thesis work.

I would like to thank **Asbjørn Danielsen** for giving me valuable insight, advise on literature and hardware.

I also wish to extend a special thanks to **Igor Molchanov**, who helped with experiments and collecting motion data.

Finally, I would like to extend my sincere gratitude to everyone who provided me with advice, support and assistance throughout the thesis work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem description

Approximately 30% of people over 65 falls each year, and for those over 75 the rates are higher [1], [2]. Between 20% and 30% of those who fall suffer injuries that reduce mobility and independence and increase the risk of premature death. Fall rates among institution residents are much higher than among community-dwellers.

Another aspect of falling is connected with the psychological aftermath of falling. Although even few falls result could serious injury (head or spinal cord injury, joint dislocation, or fracture), the psychological sequelae [1] (sometimes termed "post-fall syndrome") can be severe and can lead to a loss of self-confidence in one's ability to perform routine daily tasks, as well as social withdrawal, depression, or confusion. These, in turn, can lead to self-imposed restrictions in activity, decreased mobility, and increased dependence.

For the last 5 years appeared mobile technologies allowing to capture the motion and handle the captured data on mobile device. The price of hardware is relatively low that allows using these devices by every person it's needed. In addition, a lot of people already have a minimum amount of devices for this purpose: Norway is the country with the largest smartphone penetration in the world for age group 55+ [3]. So, if the trend will continue the problem of falling of elderly could be solved just by mobile app, which would have minimal cost for a healthcare system.

1.2 Objectives

The main goal of this work is to develop means for detecting and predicting falls among elderly people using walkers. A concept for defining instabilities and risk of falls should be designed using machine learning technique. A risk of fall or fall should be identified and a user should

be notified using any kind of mobile device. Prevention of fall should be based on the detected risks. Such control could be embedded in the walker or as a kind of wearable or similar.

To implement this project should be also designed controlled experiment to train a wearable system to detect instabilities and risk of falls. For this purpose could be used sensors or smartphone or smartwatch or any combination of these devices. Communication with sensor and controlling device should be based on Bluetooth Low Energy. The project can be applied the same approach as Elisabeth Gangenes on her Master thesis [4]. Could be used any kind of software and hardware.

1.3 Research questions

This thesis focuses on researching work of sensors, its connectivity with the smartphone, collecting and researching data from sensors, and prototyping means for predicting and detecting the falls. Another focus is to research walk smartwatch and its possibilities for solving objectives we stated. And finally, it focuses on research of possibilities of mobile devices in relation to machine learning.

During the work on projects we faced with several questions:

1. Which sensors better to use: CC2650TK, built-in phone's sensors or watchs built-in sensors or combination of them?
2. Will be enough 10 Hz frequency of sensor CC2650TK for detecting/predicting falls?
3. Is it possible to develop autonomous Apple Watch app for predicting/detecting fall that could communicate with sensors? Is it possible to use for predicting/detecting only sensors of the watch?
4. Does the Apple Watch Series 3 have enough performance for Core ML framework to detect/predict falls?
5. What motion data could be excluded and how excluding them could improve performance?
6. What will be the battery life of devices using the app for predicting/detecting falls?

Chapter 2

State-of-the-art review

2.1 Classification

Prevention of falls has been a topic that is researched more than 30 years and considered important health issue in the United Kingdom (UK), Europe, North America and Australia. Falls prevention is a multidisciplinary problem that includes such discipline as occupational therapy, physiotherapy, general practice, nursing, geriatrics, gerontology health and social care.

In a paper by Julian Hamm et al. [5], reviewed the fall prevention interventions that used to prevent falls. One of the popular approaches of fall prevention is *exercise* intervention: *supervised*, when doctors conduct training for older adults and *unsupervised*, when elderly perform paper-based exercises or use 3D technologies and games for assistance. Another type of intervention is *risk fall assessment*, an approach that used to assess a number of risk factors that affect the likelihood of falling. For this purpose is used, for example, Berg balance scale, Timed Up and Go, Turn 180 test and others. As well as for exercise intervention, 3D technology and games have shown as a low-cost solution complement to traditional fall risk assessments and to account for low adherence rates of self-assessment of fall risks done at home for this kind of intervention. The third type of popular interventions is *education interventions*. They are developed to increase knowledge according to falls prevention and educate elderly people regarding their risk of falling and falls prevention strategies based on the available evidence-based literature. *Home assessments* based interventions are carried out and *assistive equipment* is prescribed to reduce falls within the home environment. To help elderly reduce the falls this type of interventions assumes inviting clinicians to patient's home and then clinicians propose adaptations, often via the installation of assistive equipment. $Technology-based\ interventions$ could be used in a wide range of falls prevention contexts and include diagnosing and treating fall risks, increasing adherence to interventions, detecting and predicting falls and notifying clinicians in case of falls. Technologies help elderly to be more independent as they help to

perform self-assessments assistive equipment provision. Another advantage of using technologies for solving this problem is that they are helping to save money to provide effective self-care as with ageing population it's going to be a challenge. And technologies could be key to reduce the cost of a healthcare system. All the approaches in scheme presented in figure 2.1.

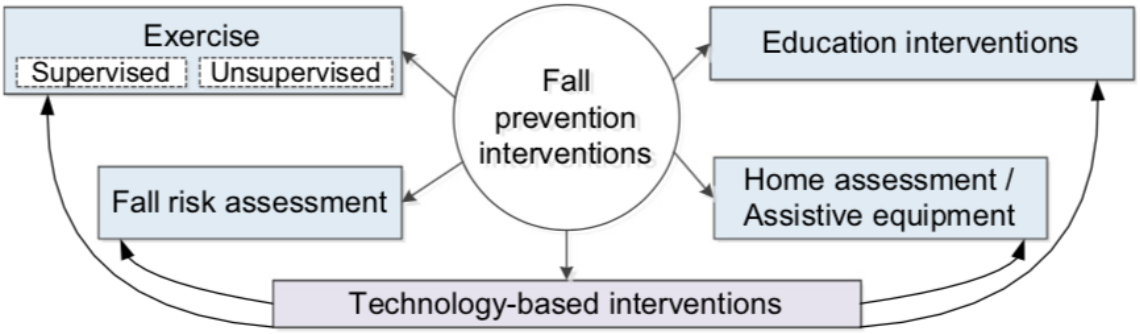


FIGURE 2.1: Overview of falls prevention interventions

Source: J. Hamm et al. / Journal of Biomedical Informatics 59 (2016) 319345

All the falls prevention technology systems could be classified by several parameters. The conceptual model of falls prevention technology presented on figure B.1. The model is separated into two parts: falls prevention technology systems in practice (in the top part of the figure) and technology deployment, which presents the range of falls technology systems proposed in the literature, the types of the user interface which they use, the information sources they exploit and their respective collaborative functions.

2.2 Representation

There are a lot of fall prevention intervention systems implemented for preventing falls. They could be divided into 4 main groups by prevention type:

- $Pre-fall\ prevention$ intervention systems - focused on supporting the prevention of falls by targeting risk factors, which if present, are known to be the cause of falls.
- $Post-fall\ prevention$ intervention systems are used in the first instance to screen elderly for fall risks after they have experienced a fall.
- *Fall injury prevention* intervention systems (FIPIs) aim to detect and respond to falls after they have occurred and prevent or minimize fall-related injuries that may occur as a consequence of falling.

- *Cross falls prevention* intervention systems (CFPIs) target the full range of interventions covered by Pre-FPIs, Post-FPIs and FPIs, thus providing an integrated approach to the delivery of falls prevention interventions to patients.

As our topic of thesis refers most to the first group and used wearables, sensors and smartphones, we consider and review only such types of systems. The compilation of projects we present in table 2.1 is based on paper by Julian Hamm et al. [5].

System	Some details	Intrinsic Fall risk factors	App type	Sensor location	Sensor purpose	Deployment environment	Multimodal interaction	Collaboration
Chou et al. [6]	from bed	Fun	S	U+C	Co+Bs	-	Nii+Ts	Async
Ferreira et al. [7]	exercises	Fun+Bal	G	U	Co	He	NUI+Ts	Sync
Geraedts et al. [8]	exercises	Fun	VR	U	Bs	He	NUI+Ts	Sync
Horta et al. [9]	similar concept	Fun	-	U	Co	-	NUI+Ts	Async
Danielsen Asbjørn et al. [10] [11]	similar concept	Fun	-	U+C	Co	-	-	Async
Majumder et al. [12]	smartshoe	Fun	S	U	Co	-	Nii + Ts	Async
Otis and Menelas [13]	smartshoe	Eh (Extrinsic)	S	U	Co	-	Nii	Asynch

Table 2.1: Pre-fall prevention interventions

Abbreviations used in table:

- Async - Asynchronous
- Bal - Balance impairments
- Bs - Bespoke sensor

- Co - Co-opted
- C - Context
- Eh - Environmental hazards
- Fun - Functional ability deficit(s)
- He - Home environment
- Nii - Non-interactive interface
- NUI - Natural User Interface
- S - Static
- Sync - Synchronous
- Ts - Touch screen
- U - User-worn
- VR - Virtual Reality

The first project in the table is concerned on preventing falls from bed, the second and third ones are based on providing special exercises for elderly, fifth and sixth ones represents special smart shoe, that could help to prevent falls.

The forth has the similar concept as the project we work on. Authors detect falls "by measuring a user body acceleration and position, getting acceleration values from the accelerometer and then transforms the values in G-force. Through the magnitude of the G-force at a given moment and inactivity immediately after, considering the position of the device, a fall may be detected." Authors also report that such technology cannot detect all the falls. They conclude that "to avoid a fall, it is better for users health and a solution to increase the accuracy rate of the mobile solution." That's is our goal of the thesis - increase accuracy using machine learning technique. The fifth has also the similar concept, but not considered the case, when elderly own their smartphone.

2.3 Other systems

Another type of interventions is post fall technologies. For example, HIPSAFE [14] represents belt that inflates with air in the moment of fall before ground impact to protect the hips of elderly (Figure 2.2). It mostly designed to protect seniors from hip fractures. But the biggest problem of using such kind of devices is social. The elderly do not want to wear hip protectors

[14]. Hip protectors can reduce the risk of fractures, but they dont make people fall less often, they just slightly reduce the risk of pelvic fractures. The problem is needed to be solved for companies that produce hip protectors is to motivate people to wear them.

FIGURE 2.2: HIPSAFE belt

Source: senior.helite.com/en/my-hipsafe/

Also, there are other projects that not based on smartphone and wearables, for example for detecting falls from bed, Danielsen Asbjørn et al. in paper [15] use the thermal camera and in papers [16], [17] use roof-mounted infrared array combined with an ultrasonic sensor.

Chapter 3

Method

3.1 Introduction

The method applied consists of multiple steps that involve qualifying the equipment needed, conduct fall experiments, create a machine learning platform and carry out testing. The overall method can be divided into the following steps:

- Assess requirement instrumentation and equipment
- Prepare and conduct fall experiments
- Pre-analysis of data from fall experiments
- Train and validate machine learning system
- Develop prototype

Each of these steps consists of one or more sub-activities. Each of these constitute part of the method developed and applied. They will be described briefly in the next paragraphs. Flow chart showing the working process on details presented in figure A.1.

3.2 Assess requirement instrumentation and equipment

3.2.1 Choosing software and hardware rig

At first, needs to be chosen a platform for development (software: iOS or Android and hardware: type of wearables) and research software frameworks needed for development. Then, needs to

be developed means for collecting the data from different sources: phone and wearable and collected data. After this step needs to research collected data using any kind of Machine Learning technique. And finally, needs to be developed prototype app that could predict and detect falls.

In the beginning, we needed to choose a mobile platform on which develop all the system: iOS or Android. In previous work implemented by Elisabeth Gangenes [4] was chosen Android, but for these project, we choose iOS for several reasons.

First of all, reliability: Blanco's report [18] shows that average Android device performed considerably worse than an average iOS device. In Q2 2017, the Android device failure rate worldwide was 25 percent, which was more than twice as high as the failure rate of iOS devices (12 percent) in the same period. As we are going to work with health data, fault tolerance of device is the most important aspect of the chosen platform.

Secondly, the CoreML framework for iOS/watchOS apps [19], presented on WWDC 2017 could be used to easily integrate machine learning models into the app [20], which theoretically allows us to detect and prevent failings not only on the mobile phone but also on the smartwatch.

Finally, Apple Watch has deep communication with iOS platform as well as convenient API for programming apps. So, our chose of development platform defined the wearable kind.

Another option needed to make is the kind of Bluetooth motion sensors. My assistant supervisor Asbjørn Danielsen recommended to use the Texas Instruments SimpleLink Multi-Standard SensorTag 2.0 CC2650TK wireless MCU, that's why we use such kind of wearable.

The walkers we used for experiments are Gemino 20 Lightweight Rollator (Figure 3.1) I have recommended by supervisor Bernt A. Bremdal.

FIGURE 3.1: Gemino 20 Lightweight Rollator

For iOS development, we used Xcode, for handling and researching data we used PyCharm. To make some editing of data we used Excel. As hosting for sources we used GitHub.

3.2.2 Review software and hardware rig

Gemino 20 Lightweight Rollator

Gemino 20 Lightweight Rollator (Figure 3.1) is stable and firm foldable rollator produced by Handicare which is very easy to operate by the user. It has the ability to correct height adjustment, which is important for achieving the best upright walking position. Height adjustment could be performed without the use of any tools, just by pulling out the handle and making the adjustment release to handle. The height of the push handle is automatically and safely locked in the fixed position. Full technical specifications presented in table 3.1.

Parameter	Values
Max. User Weight	150 kg
Maximum User Height	150 - 200 cm
Overall Length	65 cm
Total Width	60 cm
Total Weight	7.4 kg
Width Between Push Handles	47 cm
Seat Height	62 cm
Overall Height	78 - 100 cm
Folded Height	80 cm
Folded Length	65 cm
Folded Width	23 cm
Turning Radius	84 cm
Colours	Grey

TABLE 3.1: Technical specifications

Source: sunrisemedical.eu

Bluetooth Low Energy

Bluetooth Low Energy [21] is a wireless personal area network technology designed and marketed by the Bluetooth Special Interest Group (Bluetooth SIG) aimed at novel applications in the healthcare, fitness, beacons, security, and home entertainment industries. Compared to Classic Bluetooth, Bluetooth Low Energy is intended to provide considerably reduced power consumption and cost while maintaining a similar communication range. The full comparison between Bluetooth Low Energy and Bluetooth presented on table 3.2.

	Bluetooth Low Energy (LE)	Bluetooth Basic Rate/Enhanced Data Rate (BR/EDR)
Optimized For...	Short burst data transmission	Continuous data streaming
Frequency Band	2.4GHz ISM Band (2.402 - 2.480 GHz Utilized)	2.4GHz ISM Band (2.402 - 2.480 GHz Utilized)
Channels	40 channels with 2 MHz spacing (3 advertising channels/37 data channels)	79 channels with 1 MHz spacing
Channel Usage	Frequency-Hopping Spread Spectrum (FHSS)	Frequency-Hopping Spread Spectrum (FHSS)
Modulation	GFSK	GFSK, $\pi/4$ DQPSK, 8DPSK
Power Consumption	$\sim$ 0.01x to 0.5x of reference (depending on use case)	1 (reference value)
Data Rate	LE 2M PHY: 2 Mb/s LE 1M PHY: 1 Mb/s LE Coded PHY (S=2): 500 Kb/s LE Coded PHY (S=8): 125 Kb/s	EDR PHY (8DPSK): 3 Mb/s EDR PHY ($\pi/4$ DQPSK): 2 Mb/s BR PHY (GFSK): 1 Mb/s
Max Tx Power	Class 1: 100 mW (+20 dBm) Class 1.5: 10 mW (+10 dbm) Class 2: 2.5 mW (+4 dBm) Class 3: 1 mW (0 dBm)	Class 1: 100 mW (+20 dBm) Class 2: 2.5 mW (+4 dBm) Class 3: 1 mW (0 dBm)
Network Topologies	Point-to-Point (including piconet) Broadcast Mesh	Point-to-Point (including piconet)

TABLE 3.2: Bluetooth type comparison

BLE standard provides developers with a huge amount of flexibility, including multiple power levels, from 1 mW to 100 mW, as well as multiple security options up to government grade [22] and multiple PHY options that support data rates from 125 Kb/s to 2 Mb/s.

To transfer data back and forth we use GATT (Generic Attribute Profile) [23], which uses the concept called Services and Characteristics. It simplifies the usage of generic data protocol called the Attribute Protocol (ATT), which used to store Services, Characteristics and related data in a simple lookup table using 16-bit IDs for each entry in the table. Once a dedicated connection is established between two devices, GATT comes into play, showing that you advertising process governed by GAP already finished.

The most important is that connections are exclusive: Peripheral can only be connected to one central device (smartphone, etc) at a time. Advertising stops as it turns out that Peripheral connected to the Central device. After this, other Central devices no longer be able to see it and connect until the existing connection is broken.

We use this type of communication to connect together next couples: SensorTag 2.0 and iPhone, Apple Watch and iPhone. As programming framework we use Core Bluetooth [24] for first couple and Watch Connectivity [25] for second.

SensorTag 2.0 CC2650TK

SensorTag 2.0 CC2650TK is device based on the SimpleLink ultra-low power CC2650 wireless MCU for quick and easy prototyping of IoT devices. It supports development for Bluetooth low energy and allows to load new images or firmware directly over-the-air. It includes 10 low-power micro-electro-mechanical systems and other items in a tiny red package [22] :

- IR Thermopile Temperature Sensor TMP007 (Texas Instruments)
- 9-axis Motion Sensor MPU-9250 (Invensense)
- Multi-Standard Wireless MCU CC2650 (processor, Texas Instruments)
- Digital Humidity Sensor HDC1000 (Texas instruments)
- PCB antenna
- Altimeter/Pressure Sensor BMP280 (Bosch Sensortec)
- Ambient Light Sensor OPT3001 (Texas Instruments)
- Buzzer
- DevPack Expansion Connector
- JTAG Debug/Programming Interface
- CR2032 Battery Clip
- 4M Serial Storage
- Magnet Sensor MK24 (Meder)
- Digital Microphone SPH0641LU (Knowles)
- Solder point for AAA battery pack
- red and green LEDs

For collecting motion data (Gyroscope, Accelerometer and Magnetometer) we use 9-axis Motion Sensor MPU-9250 (Invensense) and for labelling sensors we use LEDs.

iPhone

iPhone is a line of smartphones produced by Apple Inc. It uses iOS as operating system software. For this project, we use iPhone 5s and iPhone 8. Each has following sensors:

- Touch ID fingerprint sensor
- Three-axis gyro
- Accelerometer
- Magnetometer
- Proximity sensor
- Ambient light sensor
- Barometer (only iPhone 8)

We use only motion sensors: Three-axis gyro, Accelerometer and Magnetometer. All these sensors support up to 100 Hz frequency.

iOS

iOS is a mobile operating system created and developed by Apple Inc. for iPhone, iPad, and iPod Touch. As we use iPhone as the Central device for this project, we develop our apps for iOS 11. As the language for development, we use Swift 4. For our apps we use several frameworks for development:

- *CoreMotion* - framework for working with motion sensors
- *CoreBluetooth* - framework for working with BLE
- *CoreData* - framework for working with object graph manager, which has an ability to persist object graphs to a persistent store, on a disk.
- *CoreML* - framework for integrating machine learning model to app
- *WatchKit* - framework for working with Apple Watch
- *WatchConnectivity* - framework for exchanging data between iPhone and Apple Watch

CoreML [19] framework helps to integrate machine learning models into app.

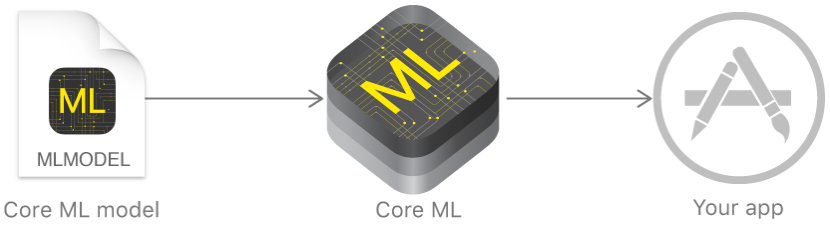


FIGURE 3.2: Integrating trained model scheme

Source: Apple's documentation

A trained model is the result of applying a machine learning algorithm to a set of training data. The model makes predictions based on new input data. For example, a model that's been trained on a region's historical house prices may be able to predict a house's price when given the number of bedrooms and bathrooms.

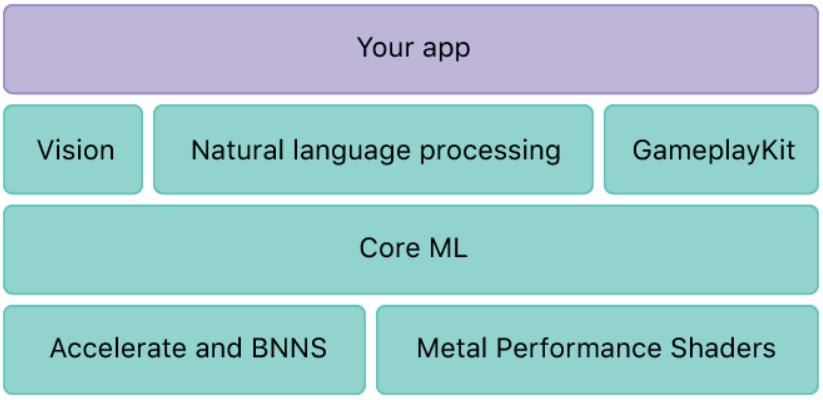


FIGURE 3.3: Core ML layer in app

Source: Apple's documentation

Core ML is optimized for on-device performance, which minimizes memory footprint and power consumption. Running strictly on the device ensures the privacy of user data and guarantees that your app remains functional and responsive when a network connection is unavailable. Core ML support following models and third-party frameworks [19]:

Model type	Supported models	Supported frameworks
Neural networks	Feedforward, convolutional, recurrent	Caffe v1 Keras 1.2.2+
Tree ensembles	Random forests, boosted trees, decision trees	scikit-learn 0.18 XGBoost 0.6
Support vector machines	Scalar regression, multiclass classification	scikit-learn 0.18 LIBSVM 3.22
Generalized linear models	Linear regression, logistic regression	scikit-learn 0.18
Feature engineering	Sparse vectorization, dense vectorization, categorical processing	scikit-learn 0.18
Pipeline models	Sequentially chained models	scikit-learn 0.18

Table 3.3: Models and third-party frameworks supported by Core ML Tools

Apple Watch

Apple Watch is a line of smartwatches produced by Apple Inc. It uses watchOS as operating system software which has deep integration with iOS. For this project, we use Apple Watch Series 3. It has following sensors:

- Barometric altimeter
- Heart rate sensor
- Accelerometer
- Gyroscope
- Ambient light sensor

We use only motion sensors: Gyroscope and Accelerometer. All these sensors, as well as iPhone's ones, support up to 100 Hz frequency.

watchOS

watchOS is the mobile operating system of the Apple Watch, developed by Apple Inc. It is based on the iOS operating system and has many similar features. It actually has all the frameworks listed in iOS section. We use watchOS 4 for our project.

3.3 Prepare and conduct fall experiments

3.3.1 Develop apps for collecting data

Developing app for collecting data from SensorTags 2.0 (DataCollector)

To collect motion data from several SimpleLink Multi-Standard SensorTags 2.0 CC2650TK we need to develop a special iOS app for that purpose that could communicate with several SensorTags and record data into internal storage of phone and export all the data into *.csv files. Another requirement for the app is a possibility to label each sensor by a unique label to do not mix data working with several sensors. For that purpose, we are going to use LEDs build-in SensorTag. We mark the first sensor by red LED, second by the green LED, third by a combination of red and green LEDs as presented in figure 3.4.

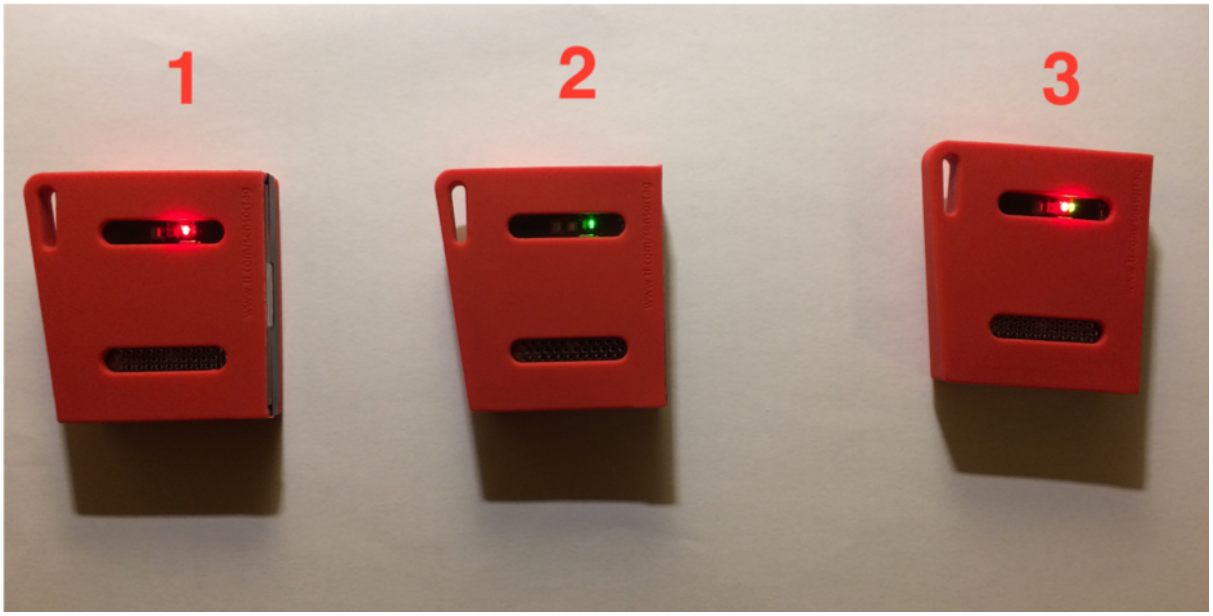


FIGURE 3.4: Labelling SensorTags

We usually use such terms as client and server to describe the entities which want data or which has the data. But when we work with BLE we use a bit different terms: as for device that has the data we use term Peripheral [26]; for the device that we want to receive data from Peripheral, we use Central. In iOS development terms: iOS app will be Central, which interacts with one or more Peripherals to receive information that could be handled, analyzed, or stored. As for our case, the Peripherals would be SensorTags 2.0 CC2650TK. Another aspect of working with BLE devices is that we want to know how to get the data we are interested in out of the Peripheral. For this purpose, we use Services and Characteristics of each Peripheral. Characteristics are representing "properties" of the device which could be read from and written to. Services are

represent set of characteristics. Peripherals are usually contained several services that we need to examine to find out which characteristics are available with which to interact.

At first, to start reading/writing data on the device we should discover it. As soon as it turns out that a Bluetooth device is switched on and is in range, it periodically sends out a little signal that lets interested devices know that its alive and kicking. This process named Advertising; the time interval between signals is named "advertising interval".

As for our case the Central is an iOS app, it listens for those advertisements, and we can specify inside our app exactly which Services we are interested in. In the process of discovering, iOS finds in the area that supports needed Services. This is important because the amount of BLE devices that are in range could be many and we need to have a possibility to filter them. For this purpose, we use framework power to filter the devices that are not broadcasting the services we are interested in.

The framework we use for working with BLE devices is called Core Bluetooth [24], which provides the classes needed for our iOS app to communicate with devices that are equipped with Bluetooth low energy wireless technology. The framework is not concerned with working with standard Bluetooth protocol, it is specially designed for working with BLE devices.

The main classes and delegates of the framework we use to develop an app in which Centrals interact with Peripherals are:

- $CBCentralManager$ - manages and interacts with $Peripherals$
- $CBPeripheral$ - an abstraction of the Peripheral that wraps the functionality surrounding the retrieval and updating of data in the remote device
- $CBService$ - represent services
- $CBCharacteristic$ - represent characteristics
- $CBCentralManagerDelegate$ - delegate of $CBCentralManager$
- $CBPeripheralDelegate$ - delegate of $CBPeripheral$

The names of classes and delegates in the framework begin with "CB" which shows that they are all the part of Core Bluetooth framework. The work of framework is based on work of delegates as about all of the interactions between the Central and Peripheral are asynchronous and non-blocking. So, the usage of delegates in this scenario allows for the calling of a method on one of the Core Bluetooth Framework objects, and it will get back at some undetermined time in the future by way of a delegate method. Initially, the class we are most interested in is the $CBCentralManager$. It works as the coordinator of the dialogue with the Peripheral

devices, and its this class that one uses as the starting point when building an app that will act as a Central.

All the Services and Characteristics in Core Bluetooth are defined by either a 16-bit UUID (defined by the Bluetooth LE specification and are listed in the Bluetooth Developer Portal in the Services or Characteristics sections) or a 128-bit UUID (for proprietary Services and Characteristics) [26]. A UUID (Universally Unique Identifier) is basically just a number.

The basic workflow developing with Core Bluetooth should be following:

Algorithm 1 Workflow Core Bluetooth

create an instance of a $CBCentralManager$
if Bluetooth services are powered on **then**
    start scanning for $CBPeripherals$ with desired Services
    if found a Peripheral we want to connect to **then**
        stop scanning
        connect to the $CBPeripheral$ we have found
        inspect the $CBPeripheral$ for available $CBServices$ (ask the Peripheral if it supports specific services or ask it to return all available Services that it supports)
        if found needed Peripherals Services **then**
            if found needed $CBCharacteristics$ **then**
                read from or write to the values of those Characteristics

In our case, for the device of SensorTag 2.0 [27] we need to get an access to movement sensor data: Gyroscope, Accelerometer and Magnetometer and also to IO Service to have a possibility to label each of the Peripheral.

To start receiving updates of motion data periodically or notifications if naming in Bluetooth low energy terms, we need using data from table: 3.4 and table 3.5 do following actions [27] :

1. Configure: enable each sensor and setting accelerometer range; I used 16G as the most sensitive.
2. The numbers start at 1 with every call to the enumerate environment.
3. Set up needed period.
4. Write 0x0001 to Notification UUID to start getting notifications.

Type	UUID	Access	Size (bytes)	Description
Data	AA81*	R/N	18	GyroX[0:7], GyroX[8:15], GyroY[0:7], GyroY[8:15], GyroZ[0:7], GyroZ[8:15], AccX[0:7], AccX[8:15], AccY[0:7], AccY[8:15], AccZ[0:7], AccZ[8:15], MagX[0:7], MagX[8:15], MagY[0:7], MagY[8:15], MagZ[0:7], MagZ[8:15]
Notification	2902	R/W	2	Write 0x0001 to enable notifications, 0x0000 to disable.
Configuration	AA82*	R/W	2	One bit for each gyro and accelerometer axis (6), magnetometer (1), wake-on-motion enable (1), accelerometer range (2). Write any bit combination top enable the desired features. Writing 0x0000 powers the unit off.
Period	AA83*	R/W	1	Resolution 10 ms. Range 100 ms (0x0A) to 2.55 sec (0xFF). Default 1 second (0x64).

TABLE 3.4: Movement Sensor Attribute table

Bits	Usage
0	Gyroscope z axis enable
1	Gyroscope y axis enable
2	Gyroscope x axis enable
3	Accelerometer z axis enable
4	Accelerometer y axis enable
5	Accelerometer x axis enable
6	Magnetometer enable (all axes)
7	Wake-On-Motion Enable
8:9	Accelerometer range (0=2G, 1=4G, 2=8G, 3=16G)
10:15	Not used

TABLE 3.5: Movement Sensor Configuration table

To label each SensorTag by desired color we need using data from table 3.6 and table 3.5 do following actions [27] :

1. Switch IO Service of each *Peripheral* into remote mode
2. Enable LED of the correspondent color of each *Peripheral*

Type	UUID	Access	Size (bytes)	Description
Data	AA65*	R/N	1	Depending on mode set in configuration characteristic.
Configuration	AA66*	R/W	1	0: local mode, 1: remote mode, 2: test mode

TABLE 3.6: IO Service Attribute table

Mode	Name	Description
0	Local	In local mode the application itself controls the use of the LEDs. By default the green led blinks when advertising, and is also used at start-up to indicate a successful self test. The red LED is used to indicate any error in the power on self test, under normal operation it is not used. The buzzer is not used by any built in functionality of the firmware.
1	Remote	In remote mode the BLE host overrides the IO usage and can activate the LEDs and the buzzer directly.
2	Test	In test mode the values of the power on self test can be read.

TABLE 3.7: IO Service Configuration table

For storing collected data we use *Core Data* framework [28]. *Core Data* is a framework for managing the model layer objects in an app. It provides generalized and automated solutions to common tasks associated with object lifecycle and object graph management, including persistence. *Core Data* is not an ORM or object-relational mapper. Nor is it a database. Instead, *Core Data* is an object graph manager which also has the ability to persist object graphs to a persistent store, on a disk. To store all the data we need we developed model for *Core Data* presented on figure 3.5. To store the session we use several entities:

- *Session* - main object, where we store session; $recordID$ is used to mark each session, for example, $recordID = 0$ use for walking, $recordID = 1$ for falling.
- *Sensor* - sensor identificator; the sensor with red LED has id=1; the sensor with green LED has id=2; the sensor with the combination of red and green LEDs has id=3

- *SensorData* - used to store data combination of 3 sensors (Gyroscope, Accelerometer and Magnetometer) and their timestamp; we receive the data from all each sensor simultaneously: timestamp is same for same Peripheral; each Peripheral has I's own timestamp for each sensor data, which could not be synchronized with each other
- *Characteristic* - used for storing such characteristics as Gyroscope, Accelerometer and Magnetometer
- *CharacteristicName* - used to label each of characteristics by name

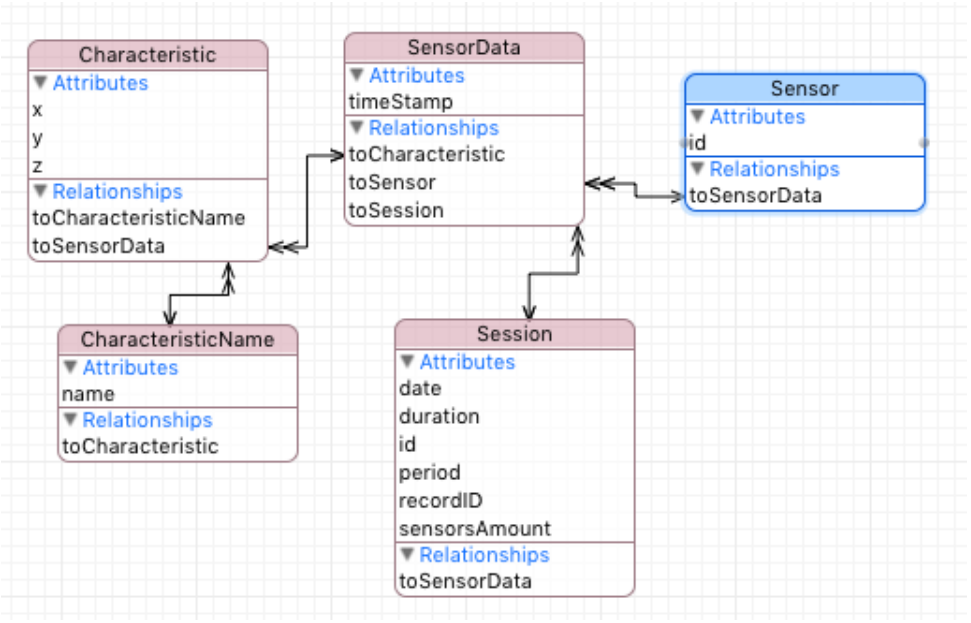
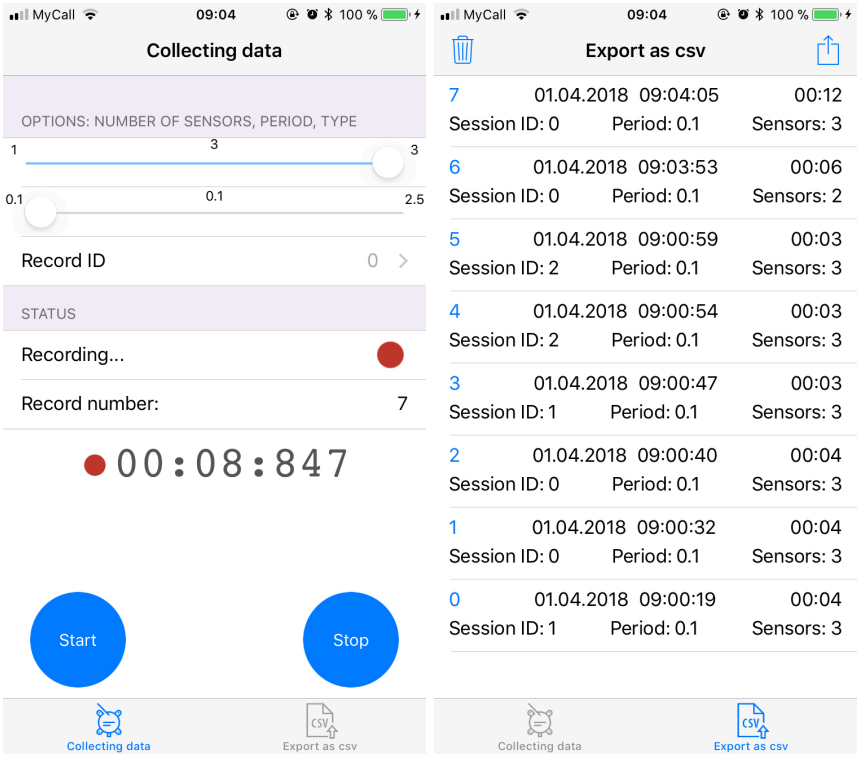


FIGURE 3.5: Core Data DataCollector

In the end, we implement the functionality of exporting all the data from object graph manager to *.csv file. That gives *.csv file containing a table that contains data for each of the sensors for each of the sessions. Finally, we get the app that contains all needed features for our experiments. Screenshots of this app presented in Figure 3.6.



(A) Record/searching View Controller (B) Editing/Exporting View Controller

FIGURE 3.6: App's screenshots (DataCollector)

Working on the app we implemented several constructs needed to provide all the functionality:

- *NSManagedObject* classes - provide work of each object of Core Data (Figure 3.7)

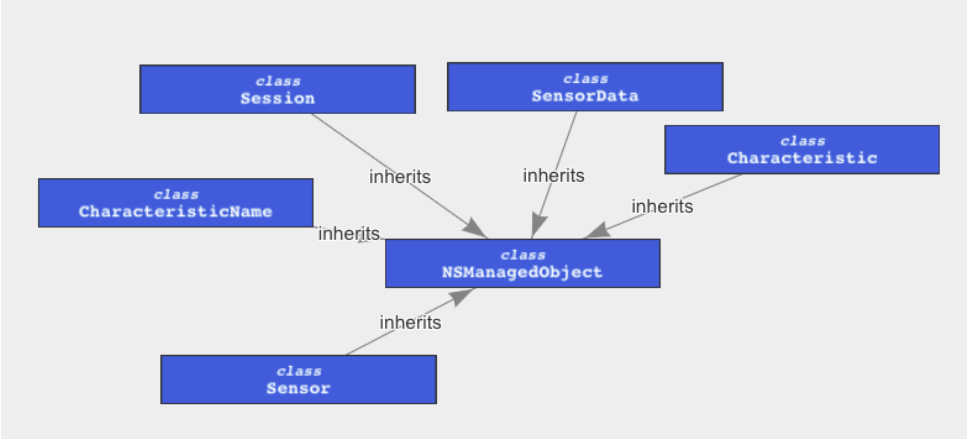


FIGURE 3.7: UML - NSManagedObjects

- *CollectingDataVC* class - provides all the functionality for collecting data (Figure 3.8)
- *ExportDataVC* class - provides all the functionality for exporting data (Figure 3.8)

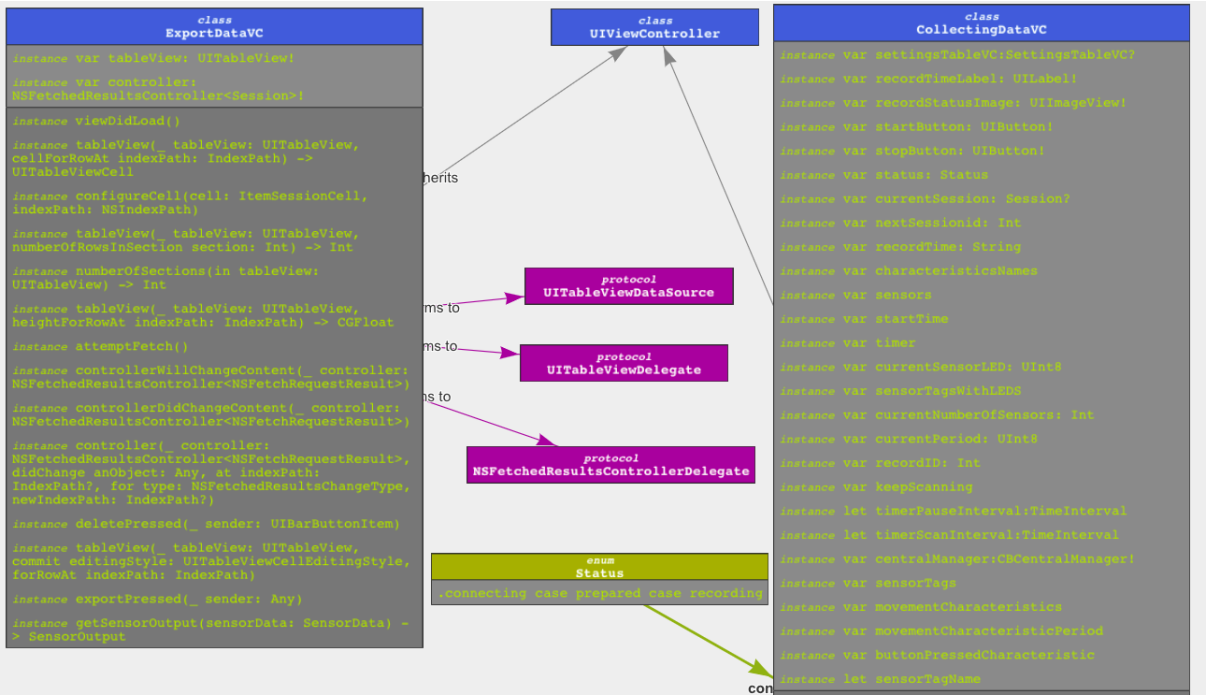


FIGURE 3.8: UML - ExportDataVC & CollectingDataVC

- *CircleButton* class - used for deforming Start/Stop buttons (Figure 3.9)
- *RecordIDVCDelegate* protocol - used for synchronization between *CollectingDataVC* and *RecordIDVC* (Figure 3.9)
- *SettingsTableVCDelegate* protocol - used for synchronization data between *CollectingDataVC* and *SettingsTableVC* (Figure 3.9)

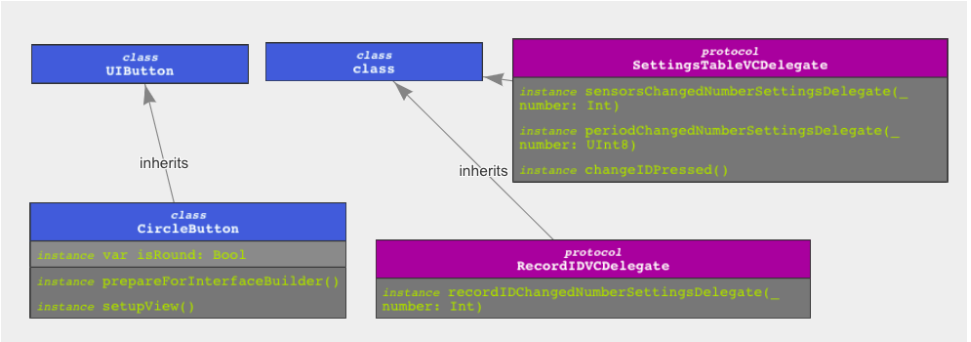


FIGURE 3.9: UML - CircleButton, RecordIDVCDelegate, SettingsTableVCDelegate

- *SessionIDCell* class - used for viewing recordIDs (Figure 3.10)
- *ItemSessionCell* class - used for viewing Sessions in table (Figure 3.10)

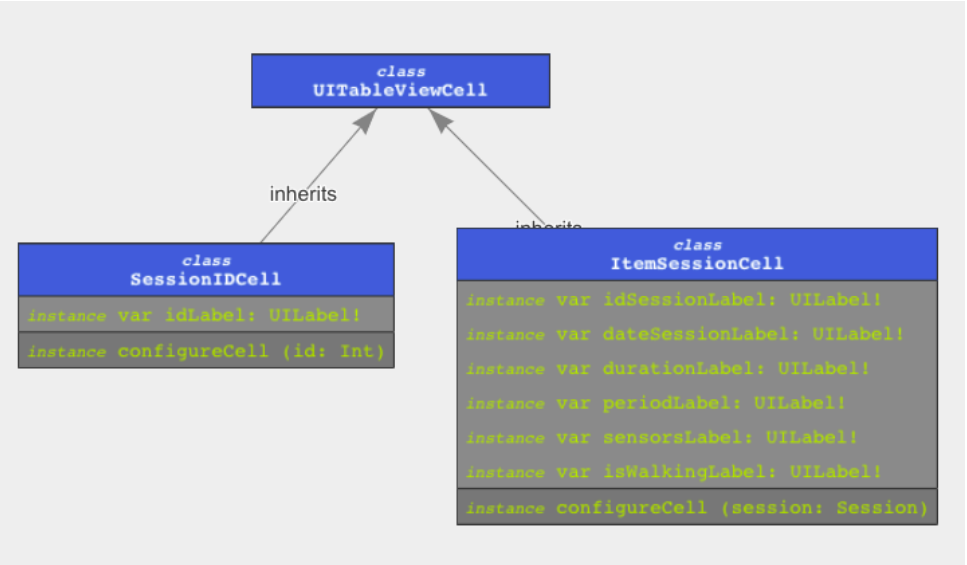


FIGURE 3.10: UML - SessionIDCell and ItemSessionCell

- *RecordIDVC* class - used to recordID changing (Figure 3.11)
- *SettingsTableVC* class - used to implement settings (Figure 3.11)

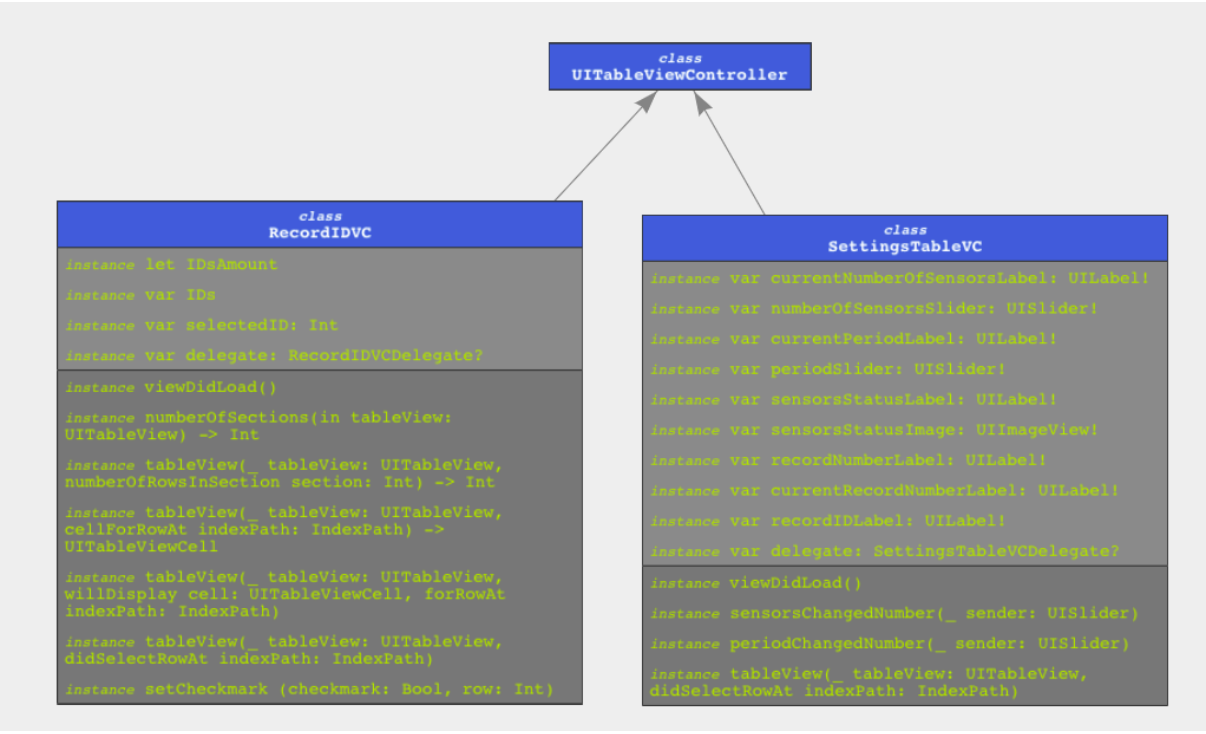


FIGURE 3.11: UML - RecordTableVC and SettingsTableVC

- *SensorOutput* class - for temporary storing SensorTag output data (Figure 3.12)
- *Device* class - singleton class that stores all const UUID values we use in app (Figure 3.12)

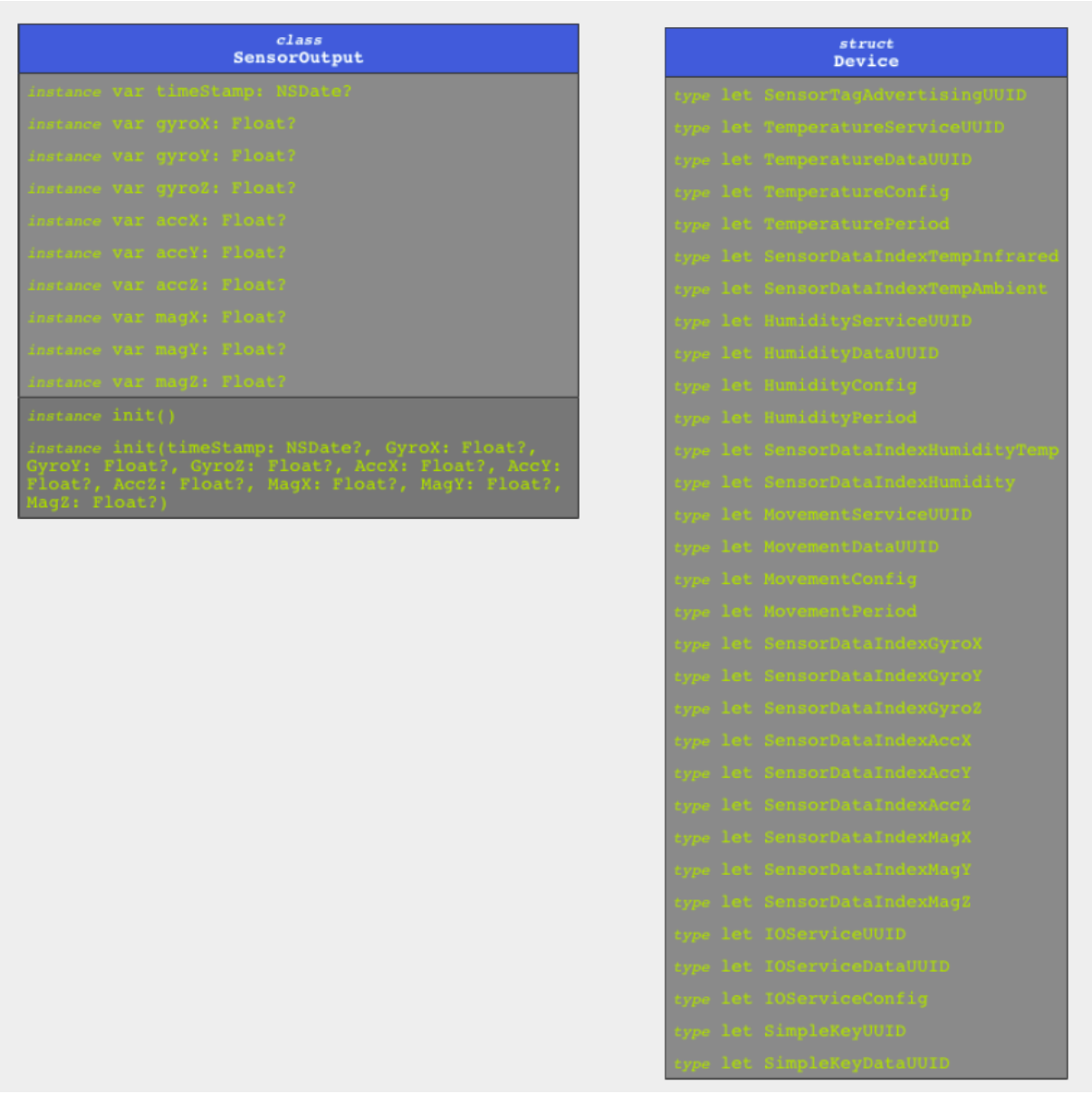


FIGURE 3.12: UML - SensorOutput and Device

In View Controllers the scheme of work looks in following way. Main View Controller has Tab Bar controller allowing separate functionality into two parts: Collecting data / Settings and Exporting data/Deleting. Full scheme presented on Figure 3.13.

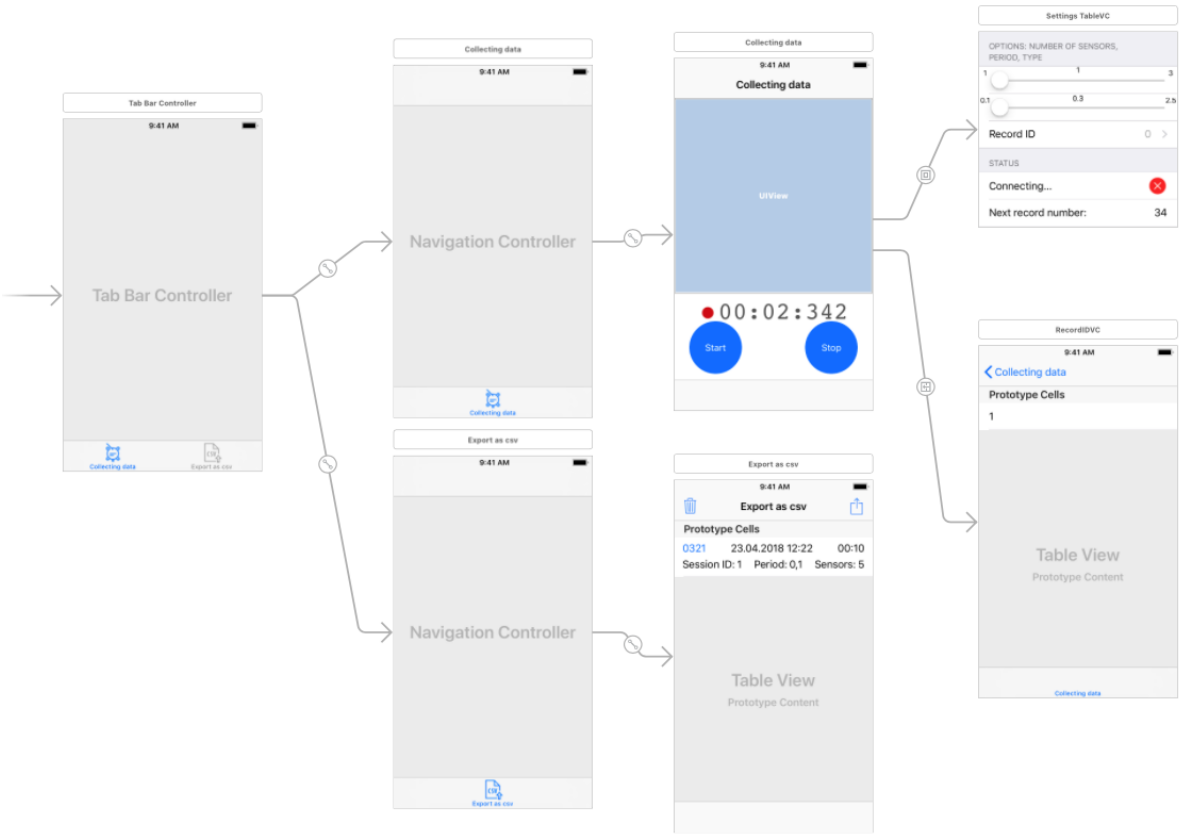


FIGURE 3.13: DataCollector ViewController Scheme

Developing app for collecting data from iPhone and Apple Watch (MotionCollector)

To collect motion data from internal sensors of iPhone and Apple Watch we need to develop another app. The app should have 3 regimes: collecting data from iPhone, collecting data from the watch and collecting data from phone and watch simultaneously.

At first, we develop all the functionality, excluding work of Apple Watch. To implement this we use Core Motion framework [29], which reports motion - and environment-related data from the onboard hardware of iOS devices, including from the accelerometers and gyroscopes, and from the pedometer, magnetometer, and barometer. We use this framework to access hardware-generated data from in-built iPhone's Gyroscope, Accelerometer and Magnetometer and save the data into Core Data [28] as we implemented in the previous app. The basic workflow developing with Core Motion goes following way:

Algorithm 2 Workflow Core Motion

```
1: if isAccelerometerAvailable == True, isGyroAvailable == True, isMagnetometerAvailable
   == True then
2:     accelerometerUpdateInterval = 1.0 / neededFrequency
3:     startAccelerometerUpdates()
4:     gyroUpdateInterval = 1.0 / neededFrequency
5:     startGyroUpdates()
6:     magnetometerUpdateInterval = 1.0 / neededFrequency
7:     startMagnetometerUpdates()
8:     create scheduledTimer loop:
9:     get all needed values: GyroX, GyroY, GyroZ, AccX, AccY, AccZ, MagX, MagY, MagZ
10:    goto loop
```

As iPhone supports up to 100 Hz frequency for motion sensors, we need to optimize work with memory. For this purpose, we store all the data from the sensor in RAM during a recording session. As well as user pressed stop button all the data we save to object graph manager Core Data. This optimization reduces CPU of phone load from 70% to 5%.

We store data using almost the same Core Data model as in the first app but we change *sensorsAmount* property which defines amount of sensors of session into sensorType as for now we need to maintain 3 types of sessions: with data only from iPhone, with data only from Apple Watch and session which includes data from iPhone as well as data from Apple Watch. Data Core model for this app presented on figure 3.14.

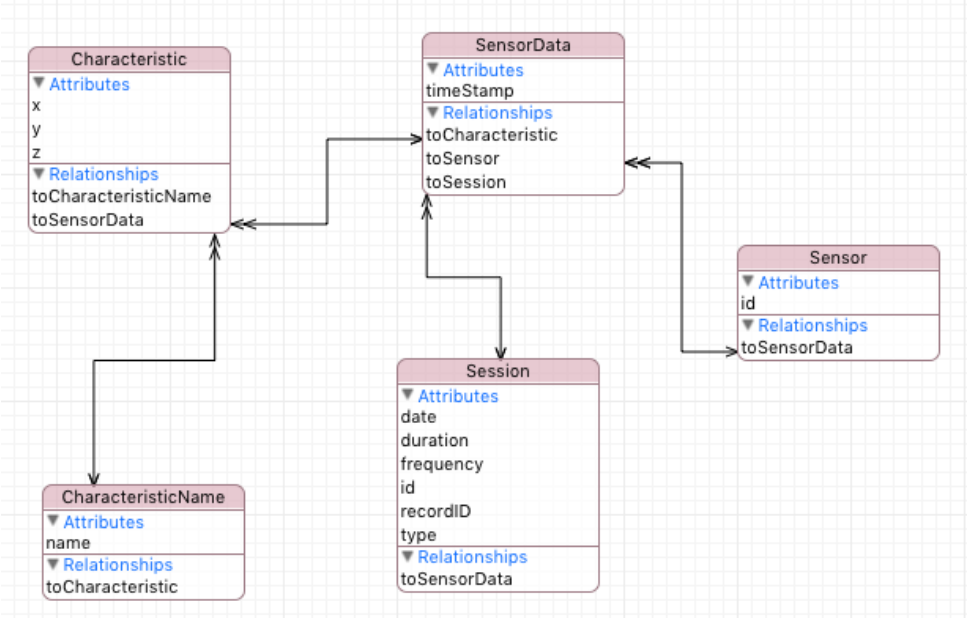
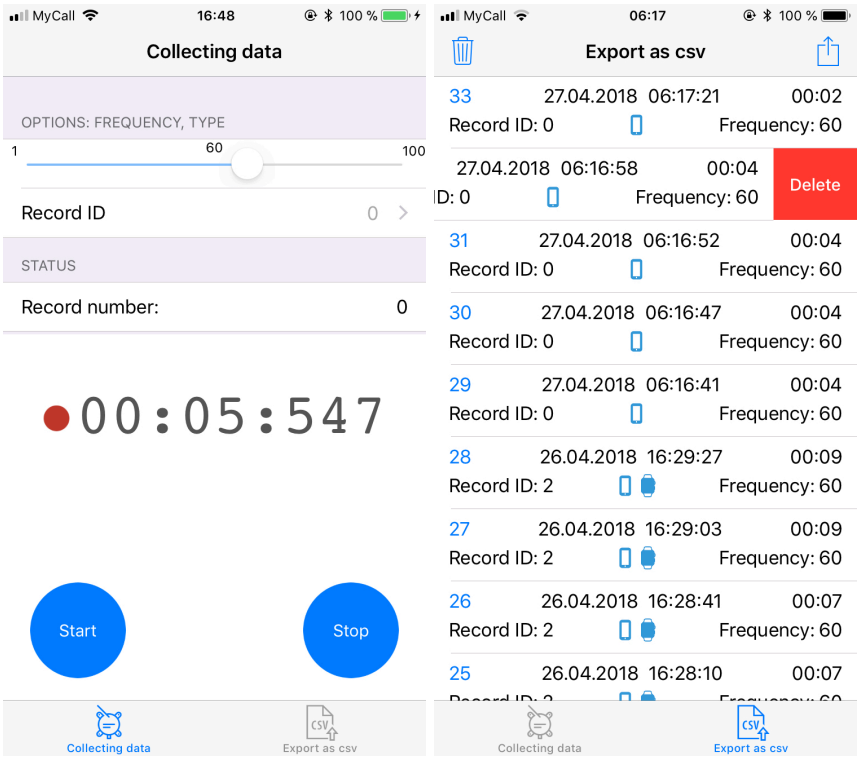


FIGURE 3.14: Core Data MotionCollector

Exporting data into *.csv file works in the same principle as in the first app. Finally, we get the app that could record data from in-built sensors and export collected data. The View Controller scheme is almost the same as in the first app with the exception of lack of not used items in settings. Screenshots of this app presented on Figure 3.15.



(A) Record/Searching ViewController (B) Editing/Exporting ViewController

FIGURE 3.15: App's screenshots (MotionCollector)

Starting working on Apple Watch app we meet several challenges:

1. We have limited resources: any modern iPhone or Mac has 10x or more computing power of Apple Watch, so there is no scope to waste system resources.
2. We have limited screen space: even the larger Apple Watch only has a resolution of 312x390 (121,680 pixels), which is small even compared to an old iPhone as iPhone 4.
3. We have limited user attention: Apple recommends complete the app's task in under two seconds.

Apple Watch relies on a phone to be fully functional. The app for Apple Watch is store inside iOS app. As soon as an iOS app would be installed on phone, the phone tries to find paired Watch. As soon as Watch will be found, the Watch app would be installed.

There is a limitation on background work of watch. This means that if we do not bypass it, collecting data will stop as soon as user turn the wrist that switches watch to sleep mode. $WatchKit$apps cannot use background execution except for 3 use cases [30] :

1. Network operations using $URLSession$
2. Playing audio using $WKAudioFilePlayer$ or $WKAudioFileQueuePlayer$
3. Run a workout using $HKWorkoutSession$

The last one is the most suitable for us. For this purpose, we use $HealthKit$ framework. Comparing to the IOS app, $WatchKit$ app needs not only check the availability of hardware and start updates, but also create and run $HKWorkoutSession$ in order to provide background work.

For exchanging data between phone and watch we use $WatchConnectivity$ framework and create $WCSession$ on phone and watch for this purpose. The limitation of this framework is concerned with maximum message size that watch can send and receive. There are three ways of exchange dictionaries between watch and phone :

1. Real-time messages or sending a message is the way of immediately transferring data from one device to the other. If either device is not reachable, this fails and it is needed to try again later. In the case of calling this from watchOS, it will launch the iOS app in the background if it isn't already run. In the case of calling this repeatedly, new data will deliver after old data. For this method limit dictionary size of data is 65,536 bytes (65.5 KB).
2. Guaranteed messages or transferring user info is a way of guaranteed data gets delivered at some points in the future. It might not be now, and this won't wake iOS app in the background, but it does ensure that data gets delivered. In the case of calling this repeatedly, new data will deliver after old data. For this method limit dictionary size of is 65,536 bytes (65.5 KB).
3. Application states or updating application context is for sending high-priority data that contains core application settings and information. In the case of calling repeatedly, new data replace old data. For this method limit dictionary size of is 262,144 bytes (262.1 KB).

As we use 60 Hz frequency, size of Double type is 8 bytes, size of Date is 8 bytes, we have two characteristics: Gyroscope and Accelerometer and 3 axes for each, we can calculate the size of stream of data to let us know whether or not we can keep within the limit of guaranteed messages to deliver data from watch to phone. We have:

$$8 * (2 * 3 + 1) * 60/1024 = 3.28125 \ (KB/\text{sec})$$

So, using this method we could only deliver data for up to 20 second, which is not enough.

There is another way to deliver data: sending files. The limit of files is not documented in *WatchKit* documentation. Then, we implement new class *SensorOutput* that adopts protocol *Codable*, that allows us to serialize/deserialize data of supporting types to store our data in the file.

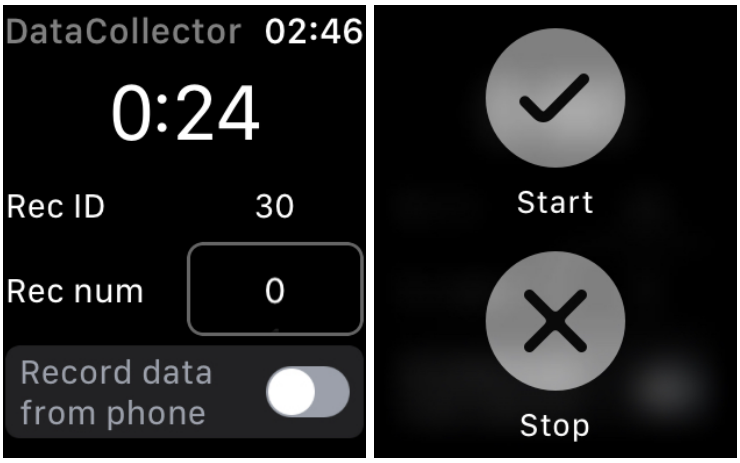
Another feature of the app we need to implement is exchanging messages between watch and phone to synchronously start the record in the case of the regime of recording simultaneously recording on iPhone and Apple Watch. In the case of using this regime, the record number which is used for identifying the type of record should be chosen on watch.

To start work on Watch app we need to think through the process how exactly the watch will be interacting with the phone. The whole workflow of working of the app should be following:

Algorithm 3 Workflow of Start and Finish recording of data on Apple Watch and iPhone

```
1:  procedure START RECORDING ON WATCH
2:      Request last record id from Phone
3:      if Received last record id then
4:          Update session id
5:          Send to phone record id and request for starting recording on phone
6:          Create new HKWorkoutSession on watch
7:          Start recording data into RAM
8:  procedure STOP RECORDING ON WATCH
9:      Stop recording data on watch
10:     Send request to stop recording on phone
11:     Serialize all recording data and save into file
12:     Send file to phone and request feedback as soon as it would be handled
```

After receiving the file on the phone, the phone starts the process of deserialization and saves all the received data into Core Data. And finally, we have all the data from phone and watch stored in single place, from which we can export for further analysis. Screenshots of Apple Watch app presented in figure 3.16. Start/Stop menu is running using *ForceTouch*, a technology developed by Apple that enables touchscreen to distinguish between different levels of force being applied to its surface.



(A) Main Interface controller (B) Start/Stop menu

FIGURE 3.16: Watch app's screenshots (MotionCollector)

Working on this app we used modified constructs from the previous app as well as create new ones:

- *CollectingDataVC* class - provides all the functionality for collecting data (Figure 3.17)
- *ExportDataVC* class - provides all the functionality for exporting data (Figure 3.17)

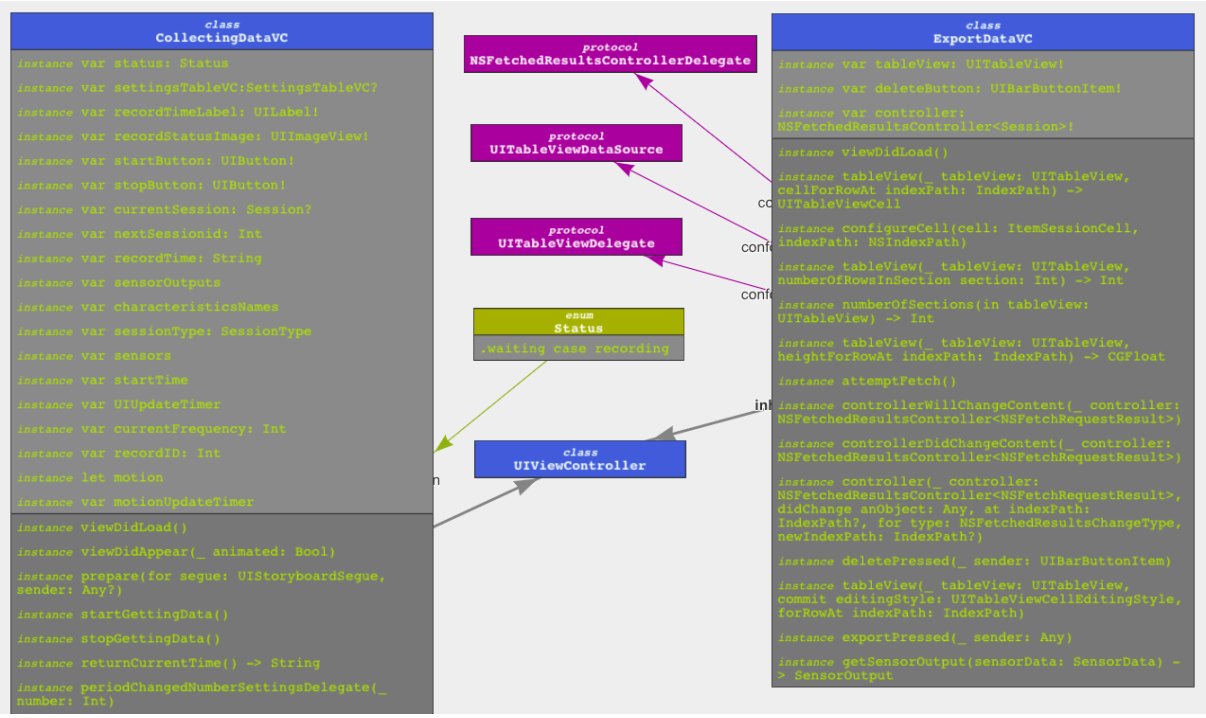


FIGURE 3.17: UML - ExportDataVC & CollectingDataVC

- *NSManagedObject* classes - provide work of each object of Core Data (Figure 3.18)

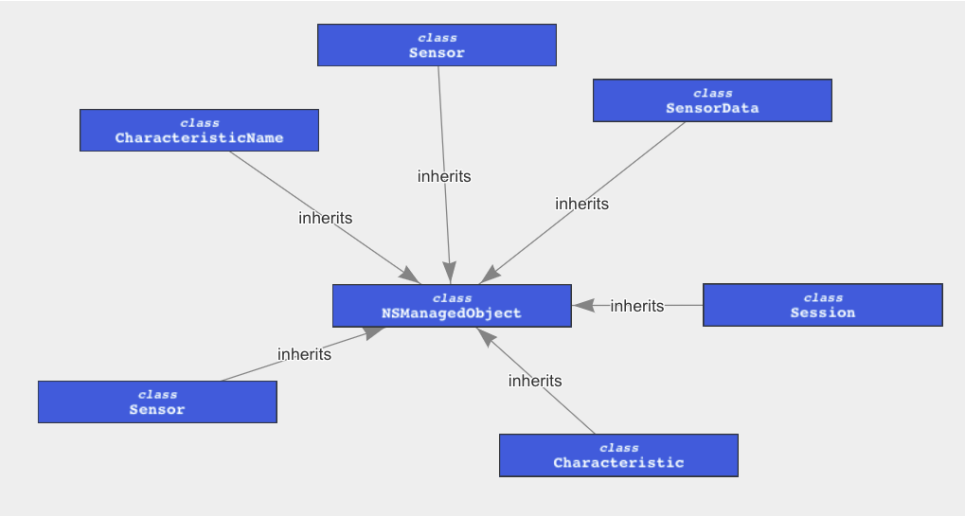


FIGURE 3.18: UML - NSManagedObjects

- *CircleButton* class - used for deforming Start/Stop buttons (Figure 3.19)
- *SessionIDCell* class - used for viewing recordIDs (Figure 3.19)
- I*temSessionCell* class - used for viewing Sessions in table (Figure 3.19)

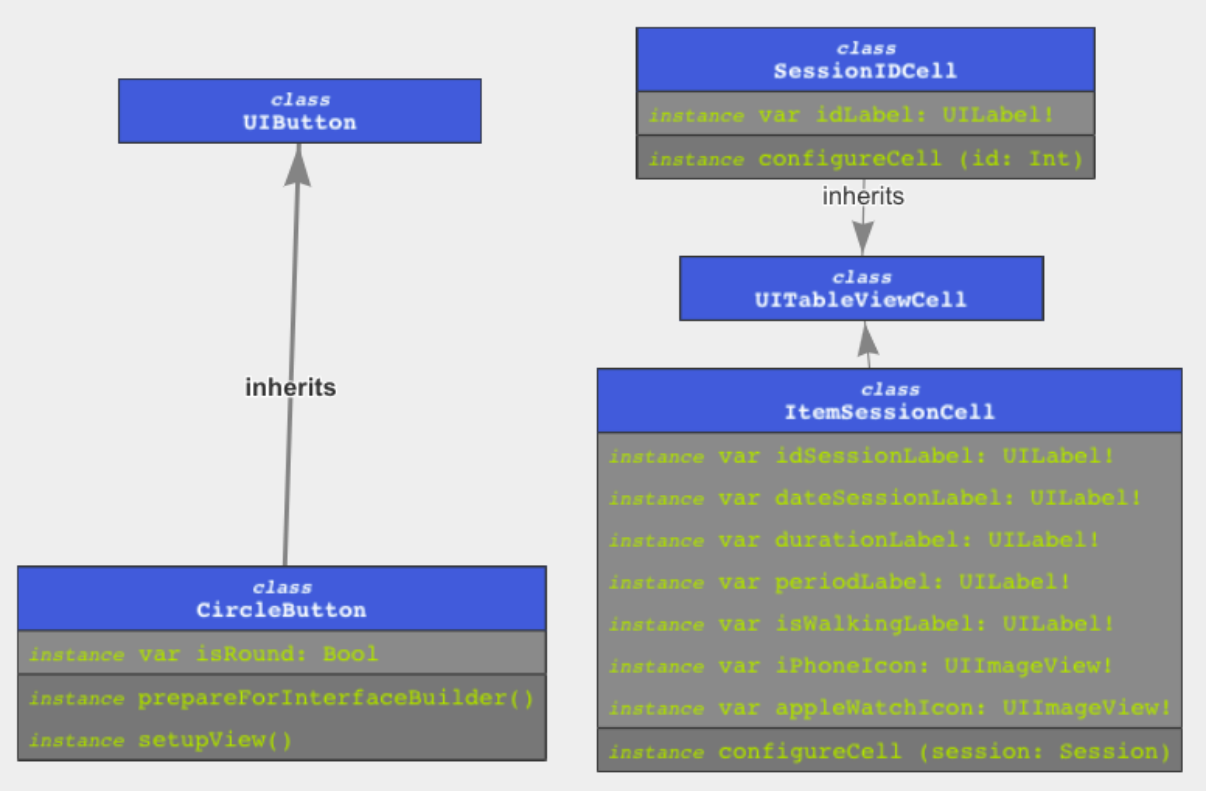


FIGURE 3.19: UML - CircleButton, SessionIDCell, ItemSessionCell

- *SettingsTableVCDelegate* protocol - used for synchronization data between *CollectingDataVC* and *SettingsTableVC* (Figure 3.20)

- *SettingsTableVC* class - used to implement settings (Figure 3.20)
- *RecordIDVCDelegate* protocol - used for synchronization between *CollectingDataVC* and *RecordIDVC* (Figure 3.20)
- *RecordIDVC* class - used to recordID changing (Figure 3.20)

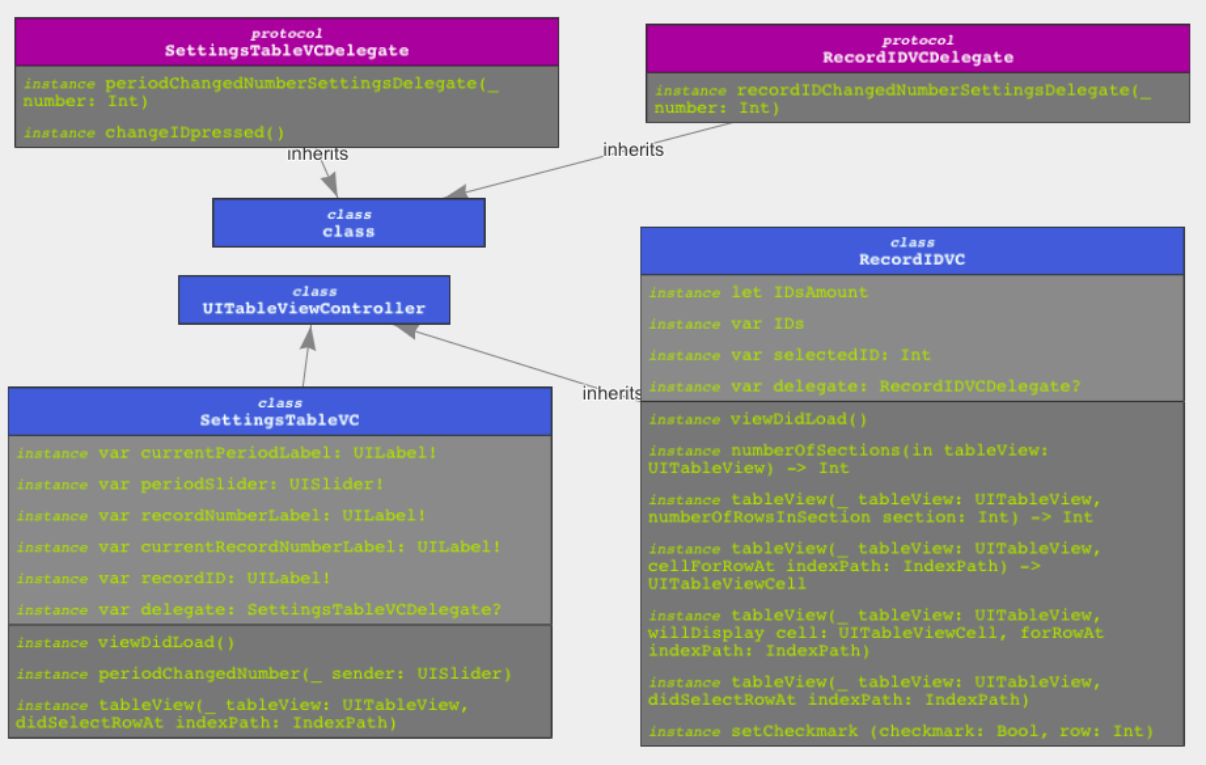


FIGURE 3.20: UML - SettingsTableVCDelegate, SettingsTableVC, RecordIDVC, SettingsTableVCDelegate

- *MainIC* class - for controlling main Interface controller of watch app, that starts/stops sessions and sends data to phone (Figure 3.21)
- *SensorOutput* class - for temporary storing SensorTag output data (Figure 3.21)
- *SessionContainer* - container using which we serialize data before sending from watch to phone and from which we deserialize data on phone before saving into Core Data (Figure 3.21)

FIGURE 3.21: UML - MainIC, SensorOutput, SessionContainer

In View Controllers the scheme of work looks in the same as in fist app. The difference is only in absence of settings that do not need in the case of using built-in sensors of iPhone and Apple Watch. Full scheme presented on Figure 3.22.

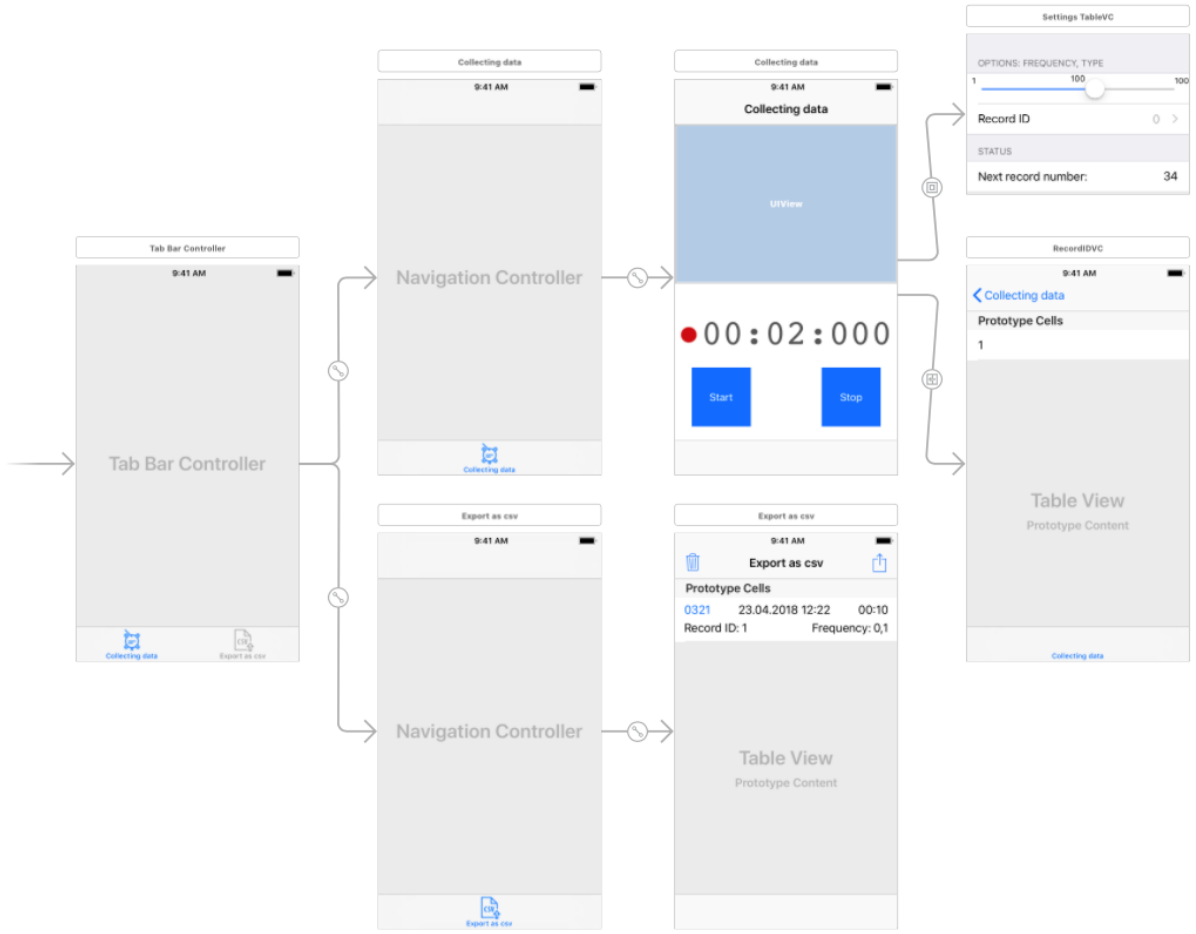


FIGURE 3.22: MotionCollector ViewController Scheme

3.3.2 Conduct fall experiments and collect data

General conditions and limitations

To start record data we need to review falling patterns of elderly people using walkers. There are five examples [31]:

1. Incorrect weight shifting while standing and turning. It happens in following way: while initiating a turn, person rotates the walker and upper body 180 degrees, while the feet remain stationary (typical of Parkinson-like freezing). Despite eventual steps, a backward fall ensues (Figure 3.23 A).

2. Incorrect weight shifting while walking forward: while stepping around the dog, the person establishes too narrow a base of support, causing a sideways fall (Figure 3.23 B).

3. Trip while walking and turning: while playing ball, the man initiates a turn by crossing his left leg in front of his right. He loses balance during the next step, after the toe of his right foot collides with his left heel, resulting in a backward fall (Figure 3.23 C).

4. Trip while walking forward: the woman seems to attempt to steer around the foot of a lifting device, but trips on the obstacle (Figure 3.23 D).

5. Loss of support with the external object while sitting down. We can note that the wheelchair rolls backwards on contact, and is unable to provide the support necessary to complete the transfer (Figure 3.23 E).

6. Fall when raising up/sitting down in bed or chair using a walker for support the walker moves, and we have a fall (Figure 3.24).

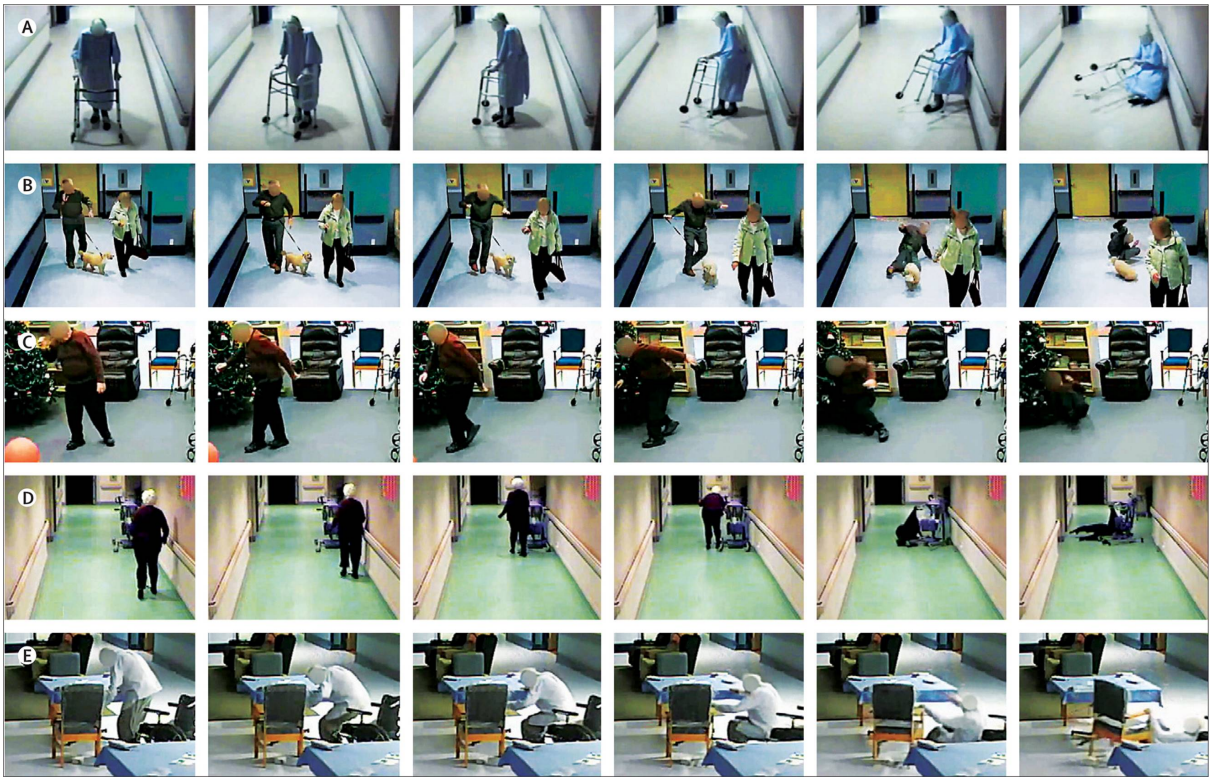


FIGURE 3.23: Falls patterns 1

Source: Lancet. 2013 January 05; 381(9860)

FIGURE 3.24: Falls patterns 2

Source: grayandwhitelaw.com

For collecting data we separate data into three groups and labelled each group by its own record id:

1. Safe walking patterns or normal walk and other movements
2. Relatively safe walking patterns or initiations [31]
3. Unsafe walking patterns of falls

For first one we record simple walk, standing, turning and sitting implemented safely (without movement that could provoke a fall). For the second one, we record patterns that could lead to a fall. We use these patterns to predict the falls as notification to a user that his or her current body position could lead to fall can help him or her to avoid falling changing the position. And for the last group, we record all patterns described above. Each pattern records 1-3 times. These patterns we use to detect falls.

For each of the type of collecting data (using SensorTags and iPhone and Apple Watch) data was recorded by me and my groupmate Igor Molchanov.

Collecting data using SensorTags 2.0 CC2650TK

To collect data using SensorTags we attach 3 of them in following way (Figure 3.25) :

1. To upper part of walkers
2. To lower part of walkers
3. Put into pocket

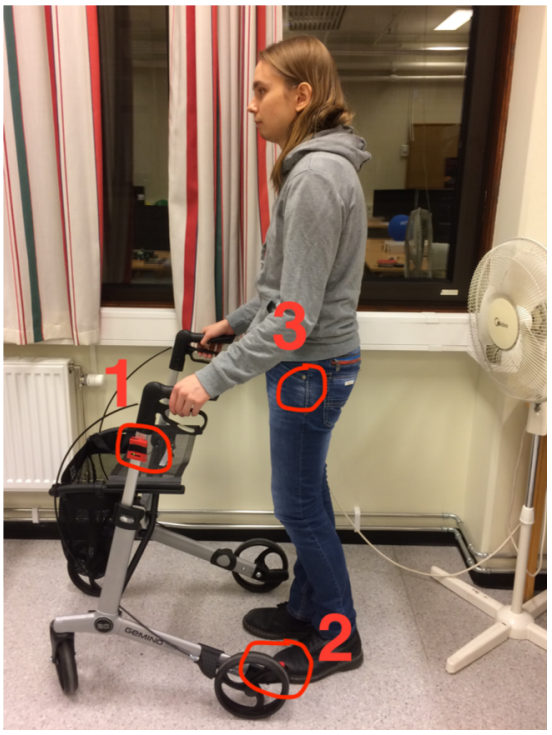


FIGURE 3.25: Attached sensors

For identification the number of each of SensorTag we use labelling by LED as we developed (Figure 3.4). For recording we used following settings:

- *Session period*: 0.1 (maximum possible)
- *Amount of SensorTags*: 3
- $Wake - On - Motion$: disabled (not available via UI, default value in app)
- *Accelerometer range*: 16G (not available via UI, default value in app)

Using such placement of SensorTags and settings described above we implement all the walking patterns and collect the data.

Collecting data using in-built iPhone and Apple Watch sensors

To collect data using iPhone and Apple Watch we dress the watch on the left hand and put a phone into the pocket of jeans. All the control of processes of recording we make using a watch.

For recording we use following settings:

- *Session frequncy*: 60 (for phone and watch)
- *Session type*: watch and phone

3.4 Pre-analysis of data from fall experiments

3.4.1 Research collected data from SensorTags

After collecting all 3 type of patterns we received 7429 data samples that include data collected by me and by Igor Molchanov. The data has following structure:

Relatively

SessionID	SessionDate	SessionDuration	SessionPeriod	AmountOfSensors	RecordID
0	2018-04-26 16:34:00.4030	05:01	0.1	3	0
0	2018-04-26 16:34:00.4030	05:01	0.1	3	0
0	2018-04-26 16:34:00.4030	05:01	0.1	3	0
0	2018-04-26 16:34:00.4030	05:01	0.1	3	0
0	2018-04-26 16:34:00.4030	05:01	0.1	3	0

Timestamp1	timeIntervalSince1970_1	GyroX1	GyroY1	GyroZ1	AccX1	AccY1	AccZ1	MagX1	MagY1	MagZ1
2018-04-26 16:34:00.4760	1524753240.47638	1.15967	0.62561	0.183105	7.83789	2.12695	-0.15625	-512.0	276.0	441.0
2018-04-26 16:34:00.5940	1524753240.59378	1.20544	0.572205	0.137329	7.83984	2.13867	-0.115234	-512.0	276.0	441.0
2018-04-26 16:34:00.6440	1524753240.64393	1.20544	0.587463	0.312805	7.87109	2.12305	-0.0683594	-509.0	275.0	435.0
2018-04-26 16:34:00.7770	1524753240.77731	1.28174	0.595093	0.221252	7.85938	2.13281	-0.132812	-509.0	275.0	435.0
2018-04-26 16:34:00.8940	1524753240.89391	1.19781	0.617981	0.328064	7.87305	2.12695	-0.0605469	-509.0	275.0	435.0

Timestamp2	timeIntervalSince1970_2	GyroX2	GyroY2	GyroZ2	AccX2	AccY2	AccZ2	MagX2	MagY2	MagZ2
2018-04-26 16:34:00.4360	1524753240.43642	1.50299	-0.923157	0.1297	7.75	-2.21289	-0.164062	-525.0	-167.0	-824.0
2018-04-26 16:34:00.5100	1524753240.50978	1.06812	-0.923157	0.495911	7.77148	-2.20117	-0.167969	-525.0	-167.0	-824.0
2018-04-26 16:34:00.6280	1524753240.6281	1.31226	-0.869751	0.259399	7.74414	-2.18945	-0.0996094	-525.0	-167.0	-824.0
2018-04-26 16:34:00.7270	1524753240.72718	1.26648	-0.831604	0.213623	7.75586	-2.16602	-0.158203	-525.0	-167.0	-824.0
2018-04-26 16:34:00.8110	1524753240.81054	1.38092	-0.892639	0.228882	7.73242	-2.17969	-0.138672	-525.0	-167.0	-824.0

Timestamp3	timeIntervalSince1970_3	GyroX3	GyroY3	GyroZ3	AccX3	AccY3	AccZ3	MagX3	MagY3	MagZ3
2018-04-26 16:34:00.4410	1524753240.44143	-0.556946	0.892639	-0.205994	-1.37598	-1.38867	0.330566	1164.0	289.0	264.0
2018-04-26 16:34:00.5440	1524753240.54388	-0.335693	0.762939	-0.556946	-1.35449	-1.39893	0.34375	1167.0	291.0	273.0
2018-04-26 16:34:00.6270	1524753240.62729	-0.427246	0.526428	-0.839233	-1.354	-1.39746	0.333496	1166.0	293.0	270.0
2018-04-26 16:34:00.7440	1524753240.74401	0.205994	1.10626	-0.938415	-1.35254	-1.41357	0.330078	1164.0	301.0	273.0
2018-04-26 16:34:00.8440	1524753240.84389	0.419617	1.00708	-1.25885	-1.34766	-1.38672	0.320312	1159.0	290.0	267.0

FIGURE 3.26: SensorsTags Data Structure

At first, we made analysis to make sure that we will not face hard-to-analyze issues when training machine learning model on this dataset (C.1). Listing shows information about collected data: amount of data for each type, used data types, amount of null values for each column, information according distribution data and amount of data for each session record type. The dataset contains both meta-information in columns $SessionID$, $SessionDate$, $SessionDuration$, $SessionPeriod$, $AmountOfSensors$, $RecordID$, $Timestamp$(1-3) and $timeIntervalSince$1970_(1-3) as well as sensor data in $GyroX$(1-3), $GyroY$(1-3), $GyroZ$(1-3), $AccX$(1-3), $AccY$(1-3), $AccZ$(1-3), $MagX$(1-3), $MagY$(1-3), $MagZ$(1-3). The number in the end of the name of the column is indicates the number of SensorTag used for collecting data according the scheme of attachment as depicted on figure 3.25. Each data sample is labeled with a value from $SessionID$, $AmountOfSensors$, $SensorPeriod$ and $RecordID$ columns.

Apart from numerical sensor data, the dataset contains data in $RecordID$ column which acts as a label for each row in the dataframe.

$RecordID$ column represents categorical variable with following possible values:

- 0: “safe walk pattern”
- 1: “relatively safe walk pattern” or initiation [31] ’
- 2: “unsafe walk pattern”

The dataset contains 6032 “safe” data samples as well as 613 “relatively safe” data samples and 613 “unsafe” data samples.

Since we have samples from 3 types of patterns, it makes sense to explore the distribution of numerical data separately for each pattern. The result is presented on figures C.1 - C.27. It’s could be concluded that data distribution of $GyroY$(1-3), $GyroZ$(1-3), $AccX$(1-3), $AccY$(1-3) and $AccZ$(1-3) are symmetrical, but have different size of peaks for each type of walk pattern in the most cases. In some cases ($AccX3$ and $AccY3$) the distribution is bimodal. $MagX$(1-3), $MagY$(1-3) and $MagZ$(1-3) are multimodal in the most of cases. Only $MagX$(1-3), $MagY$(1-3) and $MagZ$(1-3) have outliers.

To see how patterns different in accordance to each characteristics we build Sensor data plots (Figures C.28 - C.36). On these plots, we separate curves into 3 zones. Each zone represents one of three possible patterns we research. As we can see in general the characteristics that mostly subject to change the pattern of itself when the pattern of walk changes are $Gyro$(1-3) and Acc(1-3) for all the axis. The pattern change does not affect the behaviour of Mag(1-3) for all the axis. So, we can conclude, that we can not use Magnetometer for predictions. Also should be mentioned, that on the plots C.35 and C.36 first half of values of each zone is opposite to second half. It can be explained that SensorTag was attached in different positions when data was collected by me and by Igor Molchanov.

It is impossible to make predictions based on a single sample [32]. Only taking into account several data samples make it possible to define the pattern. To let us know how much samples do we need to use for detecting patterns we build plots (Figures C.37 - C.63) on which we depict all the sessions starting each session from the start of the plot. We also left the section on the plot on which the curve changes the most rapidly. Using this plot we can decide, what amount of sample do we need to take for an acceptable level of defining the pattern. As seen from the plots, 5-6 samples should be enough. That would give us a 6 by 6 ($Gyro$ + Acc all axis) matrix as an input for our neural network, but it could be power and performance inefficiently as we use the mobile device. So at first, we need to test, how prediction will work using only one characteristic and then add another for additional accuracy.

As we recorded using 10 Hz frequency (maximum frequency supported by SensorTag), 6 sample will equivalent to 0.1 * 6 = 0.6 seconds time observation. It means that minimum period needed to define the pattern will be 0.6 seconds.

3.4.2 Research collected data from iPhone and Apple Watch

Now we repeat the way we analyzed the data collected using SensorTags for data collected using iPhone and Apple Watch. After collecting all 3 type of patterns we collected 73574 data samples that include data collected by me and by Igor Molchanov. The data has following structure:

SessionID	SessionDate	SessionDuration	SessionFrequency	RecordID	Timestamp	timeIntervalSince1970	GyroX	GyroY
0	2018-04-26 16:00:32.4520	05:07	60	0	2018-04-26 16:00:32.5750	1524751232.57482	0.114449347846155	0.42063430995311
0	2018-04-26 16:00:32.4520	05:07	60	0	2018-04-26 16:00:32.5920	1524751232.59166	0.0688690794733401	0.390462292699167
0	2018-04-26 16:00:32.4520	05:07	60	0	2018-04-26 16:00:32.6090	1524751232.60864	0.00736656989126818	0.312997328175378
0	2018-04-26 16:00:32.4520	05:07	60	0	2018-04-26 16:00:32.6260	1524751232.62562	-0.0488508965075415	0.257071477915932
0	2018-04-26 16:00:32.4520	05:07	60	0	2018-04-26 16:00:32.6420	1524751232.64222	-0.113524431999571	0.201090233905812

GyroZ	AccX	AccY	AccZ	MagX	MagY	MagZ	WatchTimestamp
0.014215421266625	0.239654541015625	0.964920043945312	0.0238189697265625	-445.621887207031	2368.05053710938	2540.11059570312	2018-04-26 16:00:32.3630
-0.00310870794044949	0.260238647460938	0.964080810546875	0.030242919921875	-445.621887207031	2368.05053710938	2540.11059570312	2018-04-26 16:00:32.3780
-0.0182439850475878	0.276962280273438	0.965118408203125	0.0352020263671875	-445.621887207031	2368.05053710938	2540.11059570312	2018-04-26 16:00:32.3930
-0.0302207531018922	0.283294677734375	0.973342895507812	0.041015625	-445.697021484375	2368.04736328125	2540.11157226562	2018-04-26 16:00:32.4090
-0.0508005967415885	0.279800415039062	0.97808837890625	0.0431365966796875	-445.772155761719	2368.04418945312	2540.11254882812	2018-04-26 16:00:32.4240

WatchtImeIntervalSince1970	WatchGyroX	WatchGyroY	WatchGyroZ	WatchAccX	WatchAccY	WatchAccZ
1524751232.36313	-0.251138031482697	0.118928894400597	0.3846595287323	0.415756225585938	0.319625854492188	-0.778305053710938
1524751232.37848	-0.0423198081552982	0.152088448405266	0.446978569030762	0.424835205078125	0.382247924804688	-0.835861265659332
1524751232.39349	0.00936263520270586	0.114960975944996	0.379976868629456	0.425674438476562	0.434127807617188	-0.906600952148438
1524751232.40864	-0.216994613409042	0.0220312271267176	0.329766184091568	0.402938842773438	0.440200835466385	-0.988128662109375
1524751232.42372	-0.545621156692505	-0.0640782862901688	0.293155699968338	0.39117431640625	0.424163848161697	-0.989151000976562

FIGURE 3.27: iPhone and Apple Watch Data Structure

As for the previous dataset, at first, we made analysis to make sure that we will not face hard-to-analyze issues when training machine learning model on this dataset (D.1). The code was the same as for the first dataset, so the listing shows the same type of information: the amount of data for each type, used data types, amount of null values for each column, information according to distribution data and amount of data for each session record type. This dataset contains both meta-information in columns $SessionID$, $SessionDate$, $SessionDuration$, $SessionFrequency$, $RecordID$, $Timestamp$, $WatchTimestamp$, $timeInterval-Since1970$ and $WatchtimeIntervalSince1970$ as well as sensor data in $GyroX$, $GyroY$, $GyroZ$, $AccX$, $AccY$, $AccZ$, $MagX$, $MagY$, $MagZ$, $WatchGyroX$, $WatchGyroY$, $WatchGyroZ$, $WatchAccX$, $WatchAccY$, $WatchAccZ$. The word "Watch" at the start of the name of the column is indicate belonging data to Apple Watch. Each data sample is labeled with a value from $SessionID$,, $SensorFrequency$ and $RecordID$ columns.

$RecordID$ column is also as in previous dataset represents categorical variable:

- 0: "safe walk pattern"
- 1: "relatively safe walk pattern" '
- 2: "unsafe walk pattern"

The dataset contains 46890 "safe" data samples as well as 12496 "relatively safe" data samples and 12496 "unsafe" data samples.

Since we have samples from 3 types of patterns, it makes sense to explore the distribution of numerical data separately for each pattern. The result is presented on figures D.1 - D.15. t's could be concluded that data distribution of $Gyro$, Acc, $WatchGyro$ and $WatchAcc$ for all the axis are symmetrical, but have different size of peaks for each type of walk pattern in the most cases. In some cases ($AccY$, $AccZ$, $WatchAccY$, $WatchAccZ$) the distribution is bimodal for one or several walk patterns. $MagX$, $MagY$ and $MagZ$ are multimodal in the most of cases. Only $MagX$, $MagY$ and $MagZ$ have outliers.

To see how patterns are different in accordance with each characteristic we build Sensor data plots (Figures D.16 - D.20) as well as we did it for dataset collected from SensorTags. On these plots, we separate curves into 3 zones. Each zone represents one of three possible patterns we research. As we can see in general the characteristics that mostly subject to change the pattern of itself when the pattern of walk changes are $Gyro$, Acc, $WatchGyro$ and $WatchAcc$ for all the axis. In the case of $WatchGyro$ and $WatchAcc$ we see that the behaviour of curves almost does not changes when the plot passes from 'safe" pattern to "relatively safe" pattern. This means, that we cannot actually distinguish between 'safe" pattern and "relatively safe" using the Apple Watch motion sensor, The pattern change does not affect the behaviour of Mag for all the axis. So, we can conclude, that we can not use Magnetometer for predictions. It could be also mentioned that on the plot D.17 the first half of values of each zone is opposite to the second half. It can be explained that iPhone was attached in different positions when data was collected by me and by Igor Molchanov.

In a similar way as according to the data collected using SensorTags, It is impossible to make predictions based on a single sample. We need to take into account several data samples to make it possible to define the pattern. To let us know how much samples do we need to use for detecting patterns we build plots (Figures D.21 - D.35) on which we depict all the sessions starting each session from the start of the plot. On the plots, we left only the section on which the curve changes the most rapidly. Using this plot we can decide, what amount of sample do we need to take for an acceptable level of defining the pattern. As seen from the plot, 10-12 samples should be enough. That would give us a 12 by 6 ($Gyro$ + Acc all the axis for iPhone and $WatchGyro$ + $WatchAcc$ all the axis for Apple Watch) matrix as an input for our neural network, but it could be power and performance inefficiently as we use the mobile device, especially for Apple Watch. So at first, we need to test, how prediction will be work using only one characteristic and then ad another for additional accuracy.

As we recorded using 60 Hz frequency (maximum frequency supported by iPhone and Apple Watch is 100Hz), 12 sample will equivalent to 0.06 * 12 = 0.72 seconds time observation. It means that minimum period needed to define the pattern will be 0.72 seconds.

3.5 Train and validate machine learning system

As our goal is to create and train machine learning model that could predict type users' walking pattern, we need to choose machine learning framework. Our choice is limited by the list of frameworks supported by Core ML (Table 3.3).

It is always the challenge of variety trying to solve the problem using machine learning. A variety of models the one can use is vast. As it is always time and resources to evaluate only a couple of models, choosing the most appropriate one is important.

We choose Keras framework because feed-forward neural networks in Core ML are supported by this framework. This framework is developed with a focus for swift experimenting and can be used on top of TensorFlow, CNTK, or Theano, therefore, was a perfect option for us. We selected TensorFlow backend for Keras as the one we already made some experiences with and was comfortable understanding its concepts and that allows experiment fast and concentrate on design and not the implementation of the model.

Neural network for data from SensorTags

In feedforward neural networks everything starts from the input layer. As we already decided to use 6 samples (or 6*2 for 2 characteristics) for input layer, now we need to decide on a hidden layer. Starting neural network that contains only one hidden layer, evaluating its performance and move further, adding an additional hidden layer if needed we find out that three hidden layers are optimal for our case as appending more hidden layers had either no effect on accuracy or reduced it. We used 10-fold cross-validation on test data here and in all further experiments to derive accuracy numbers. Its arguable, whether or not such the neural network we use can be considered a deep one, but most that we able to find an optimal number of hidden layers.

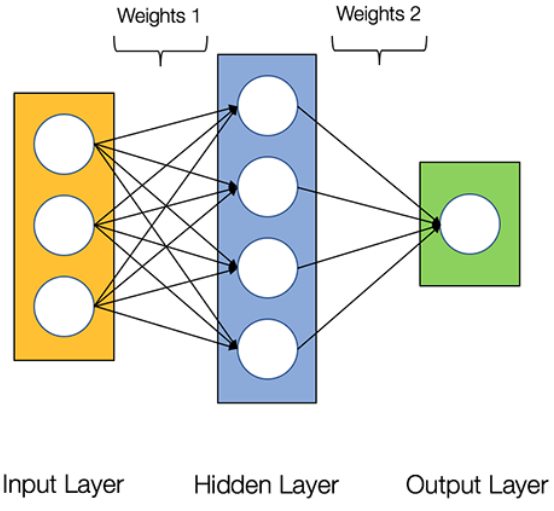


FIGURE 3.28: Layers in neural network

As we have other hyperparameters such as the number of neurons in hidden layers and their activation functions, we also tweak them. The search procedure showed that the highest prediction accuracy had a network with 15 neurons in each hidden layer and rectified linear unit activation function. We selected categorical cross-entropy loss function for its ability to increase a networks learning speed independently of defined learning rate. ADAM optimizer was chosen for our model for computational efficiency and delivering adequate results for the kind of problem we trying to solve.

As a result (Figures C.64 - C.93) we can partially prove our statement, that using Mag(1-3) for all the axis is impossible to define the pattern and this characteristic is not affected by changing the pattern. Only in the case of $MagZ3$ it gives adequate accuracy, but this accuracy is relatively small compared to *Gyro* and *Acc*. As for *Gyro* and *Acc*, the result of prediction is approximately the same for all the SensorTags, but the best accuracy has SensorTag that was in a pocket the process during data recording. If we train our model using two characteristics $GyroX$ and $AccX$ (6 samples for each or 12 in total) to improve the accuracy (Figures C.90 - C.93) the result will be almost the same as using only one characteristic.

As the biggest accuracy has the neural network build using the data collected from the SensorTag that was in a pocket, it could be concluded that for our main goal - to detect and predict the falls among people using walkers it is possible to use only iPhone's in-built sensors.

Neural network for data collected from iPhone and Apple Watch

As we already decided, for input layer we use 12 samples (or 12 * 2 for 2 characteristics), now we need to decide on a hidden layer. Experiments show the same result as for previous data - the most optimal is using 3 hidden layers. We also use 10-fold cross-validation on test data here and in all further experiments to derive accuracy numbers as well as we done for data collected using SensorTags. As for other available hyperparameters: a number of neurons in hidden layers and their activation functions, after a search we stopped on the network with 15 neurons in each hidden layer and rectified linear unit activation function. We selected categorical cross-entropy loss function for its ability to increase a networks learning speed independently of defined learning rate. ADAM optimizer as well as in the previous case was chosen for our model for computational efficiency and delivering adequate results for the kind of problem we trying to solve.

As a result (Figures D.36 - D.52) we can prove our statement, that using Mag for all the axis is impossible to define the pattern and this characteristic is not affected by changing the pattern. As for other characteristics, *Acc* and $WatchAcc$ shows the better result in comparison with *Gyro* and $WatchGyro$. Comparing with result that we get using data collected from SensorTag, the result is slightly worse, but we need to take into account that frequency is bigger in 10 times,

that means, that sensitivity of iPhone's and Apple Watch's sensors is better and adding more samples could improve the results.

If we train also our model using two characteristics $GyroX$ and $AccX$ (12 samples for each or 24 in total) to improve the accuracy (Figure D.42) the result is better compared to use only one characteristic. As for $WatchGyroX$ and $WatchAccY$ in the case of using together (24 samples for prediction) the result is slightly worse, compared to best one from its couple.

Combining two characteristics as well as in the case of data collected using sensor tags we choose two best characteristics of the different type. Now we can start preparing the model for integration to iOS app and implementing a prototype.

3.6 Develop prototype (SafeWalk)

As we saw during working on prediction, that although we can use SensorTags for prediction and detection, but following the logic "less equipment required, the better" we decided to continue our work on the prototype using only models collected using iPhone and Apple Watch. Other reasons for this choice are better accuracy and less minimum period needed for defining the pattern (as the frequency is higher) of iPhone's and Apple Watch's sensors.

To utilize the models we received inside the iOS application which defines the walking pattern of user we need to prepare them according to the following path:

1. Serialize our trained model to JSON and save its weights. Keras actually provides a straightforward API to achieve this.
2. Save models weights to *.h5 file on disk.
3. Convert my saved model to Core ML format using Apples Core ML Tools Python
4. Importing Core ML

To import Core ML model to our iOS app we need just to drag&drop it to Xcode project. The values used for creating model metadata for output and input description are now visible in the Core ML model's window in Xcode (Figure 3.29).

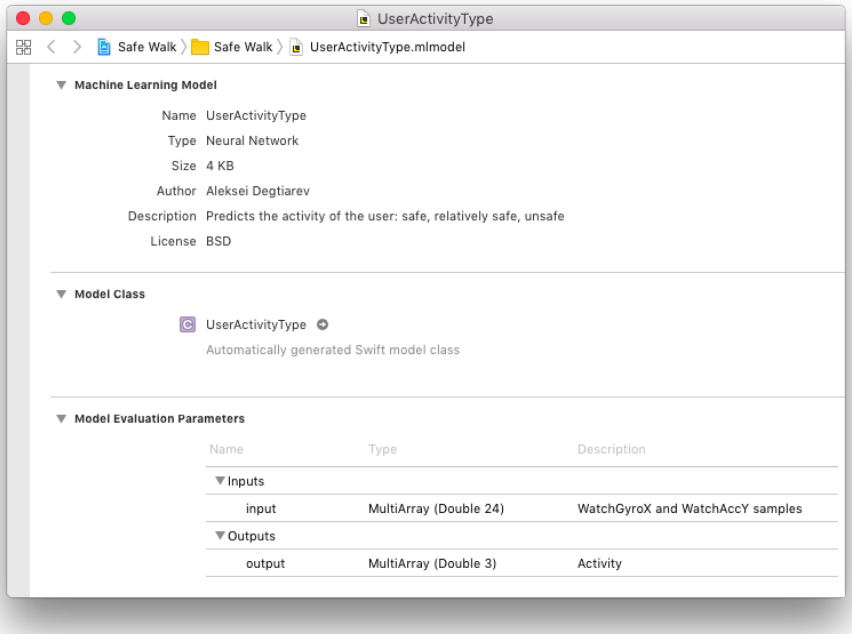
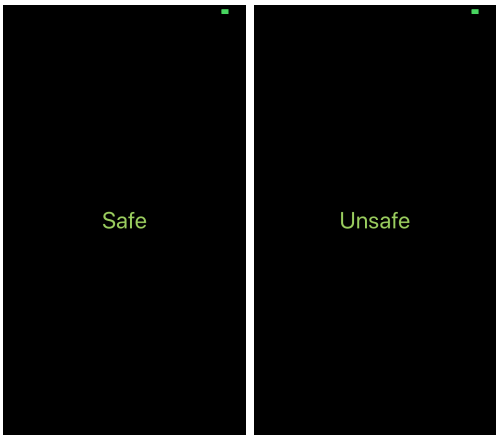


FIGURE 3.29: Core ML model window in Xcode

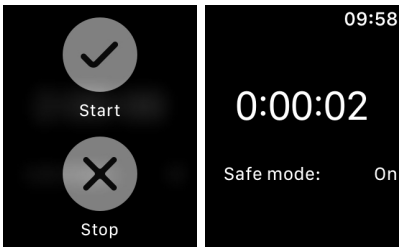
In the same way, imported the model for the watch. Xcode automatically generates custom API for imported Core ML model. So, now we ready to feed the model with sensor data and read predictions.

We implemented simple prototype apps for iOS and WatchOS. We accessing the motion sensor in the same way as we done it during the work on apps for collecting the data. After getting access to real-time motion data we feed the model and get a result. Apps have a single view with the text label, showing the current state. If state is unsafe, device vibrates (Figure 3.30, 3.31).



(A) State Safe (B) State Unsafe

FIGURE 3.30: App's screenshots (SafeWalk)



(A) State Safe (B) State Unsafe

FIGURE 3.31: Watch app's screenshots (SafeWalk)

During development we implemented several constructs needed to provide all the functionality:

- *ViewController* class - provides all the functionality of main view of iOS app (Figure 3.32)
- *InterfaceController* class - provides all the functionality of main Interface controller of watch app, that starts/stops sessions of predictions (Figure 3.32)

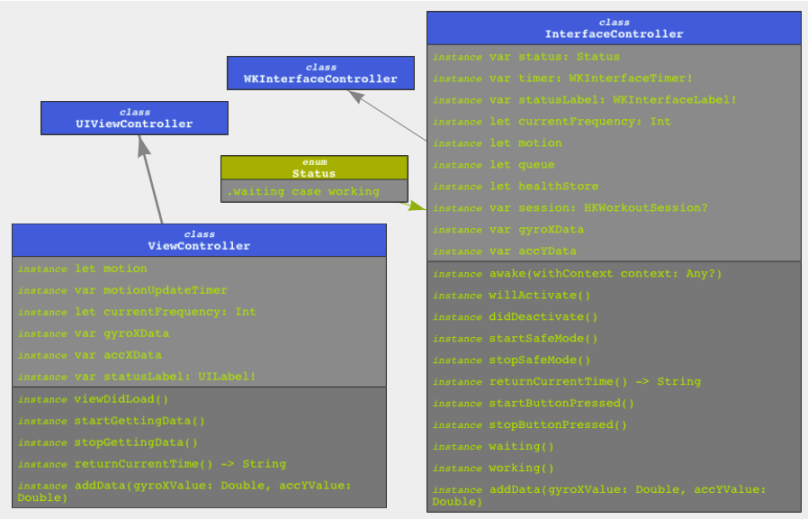


FIGURE 3.32: UML - ViewController and InterfaceController

- *Classifier* and *WatchClassifier* provide the work of models prepared for iPhone and Apple Watch respectively (Figure 3.33)

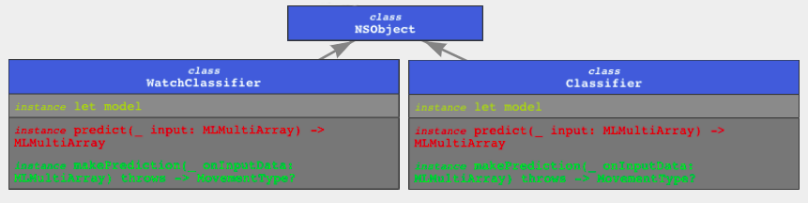


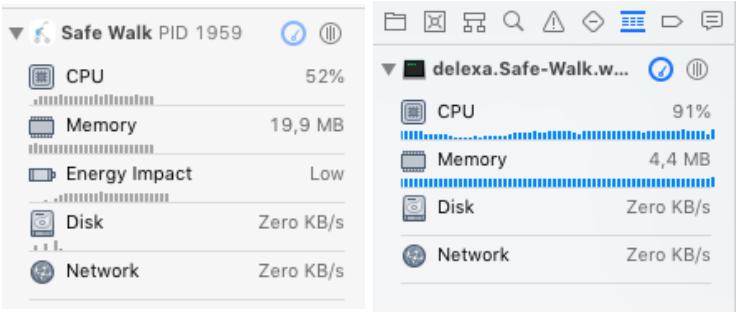
FIGURE 3.33: UML - Classifier and WatchClassifier

Chapter 4

Results

The prototype for iPhone works stably, distinguishes between “safe” and “unsafe” patterns perfectly. It works the same on iPhone 5s as on iPhone 8, despite of the best performance of the last one - optimization is not required. But sometimes the app has problems with distinguishing between “safe” and “relatively safe” patterns. This could be explained that data labelled as “relatively safe” in our dataset used for training the model was collected insufficiently accurate. For example, when a user starts and ends recording he or she spends short time period for pressing the button to start and to stop.

Another problem with the app is CPU load (Figure 4.1a) during feeding the model with sensor data. This leads to increase of battery consumption. This problem can be solved with optimizing the frequency.



(A) Phone CPU load (B) Watch CPU load

FIGURE 4.1: CPU load

The app for Apple Watch works unstable: actually almost never defines the right pattern. The problem could be connected with the performance of processor used in Apple Watch (Apple S3 for the generation I used). As we can see in figure 4.1b, the CPU load of the watch is 91% which is very high. In addition, it is not allowed to work in the background with such load,

as the limit of CPU load for apps working in the background is 80% [33] and in the case when observed 89% CPU load over 60 seconds, the app crashes.

The work of watch app could be optimized by limiting frequency of motion sensors. As we concluded in our research that it is impossible to predict falls or detect "relatively safe" walk pattern using motion data from Apple Watch, it would be not a problem as detecting falls is less sensitive to decreasing frequency. Another way of using Apple Watch for this purpose is just to use them as notification device: all the prediction work will be done on iPhone and when the body position become "relatively safe" the user will receive notification on watch. This approach significantly saves energy, as in this case watch do not need to work with motion data in the background.

Chapter 5

Discussion of results

The testing prototype showed that it is actually possible to use only smartphone for prediction and detection of falls. As for now, it is not the ideal solution, but it works. Our research also showed us that only Accelerometer and Gyroscope could be used for detection and prediction the falls. It is possible to use only one characteristic: Accelerometer or Gyroscope to predict and detect falls, but using them together increases accuracy.

The accuracy could be improved by changing the way of preparing the data for the neural network. The following way of collecting and preparing data could solve such problem: simultaneously with the recording of motion data, record the video on camera of all the moments, and when all the types of pattern will be recorded, basing on video (which should be synchronized by time with motion records) prepare manually the data for creating the model, but for this purpose needed to be developed special app that could consider motion data on the tape of time with synchronized video for accurately preparing sessions, watching the videos and cutting needed moments. An example of software that works on similar principle could be $Kinect Studio$ (Figure 5.1): it records the movement of the person using $Kinect$ sensor as well as video of the movements and allows to cut the records to get needed moments of movements.

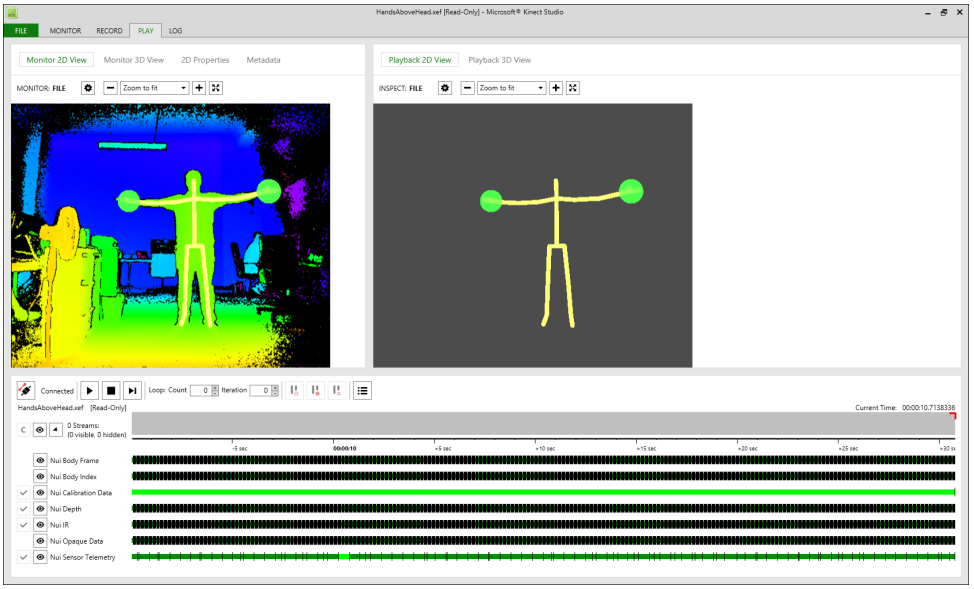


FIGURE 5.1: Kinect Studio

Source: kinect.github.io

Although, the performance of Apple Watch is enough for collecting data autonomously and model, built using data collected from Apple Watch shows acceptable accuracy for detecting falls (only detecting falls, not predicting as shown on plots D.19 and D.20), there is problem with it when we trying to use *CoreML* framework. Formally, it compiles and runs, but not defines the current state and as shown in chapter 4, the CPU load of the watch is 91% and actually not works correctly, which is unacceptable for usage. The problem should be researched more in detail, because it could be optimization problem as well as framework's bug. Another way of optimizing the work of Apple Watch as falls detector/predictor is refuse of using machine learning technique on it and use fall detection algorithms [34], [35].

SensorTags 2.0 that we used in our experiments could be also used for our main purpose - detection and prediction of falls, but it has several problems and for some of them there are ways for the solution:

- Low frequency (10 Hz) allows detect/predict falls but gives longer delay compared to phone's sensors. This problem could be solved by using another type of sensors, for example, *MetaWear : BudgetConscious Tiny Programmable BLE Sensors* [36] can stream sensor data from 1 Hz to 100 Hz, which is comparable to the phone's sensor.
- Poor battery life. In the case of working in maximum frequency (10 Hz) sensor works no longer than 48 hours (maximum ideal conditions) [37]. It could be a problem for elderly to change the battery every 1-2 days. This could be solved in 2 ways: optimization of frequency: decreasing it when the person sleeps or sits, and increase in the moments of

movements. Another way is to use battery with more capacity, but it could also increase weight.

- Receiving data not guaranteed by BLE: we cannot be 100% sure that no packet from the sensor would be lost. We can check if the packet was lost, but can not get rid of losing packets.

It is actually enough to use only one sensor and we do not need to solve the problem of synchronization of sensors [38], which does not work perfectly. As showed our experiments, it does not matter, where the sensor will be attached (at least among the places we used, which presented in figure 3.25): to walkers or lie in the pocket, the data it receives could be used for prediction and detection, but with data from the sensor that lies in pocket it can predict and detect falls with slightly higher accuracy then compared to other sensors.

Chapter 6

Further development

Before this project can help elderly people using walkers it should be extended and improved and I would like to propose a plan for further development:

1. Develop software described in chapter 4, that could combine video and motion data in a single interface to improve the accuracy of data used to build models.
2. Provide new data collecting with more people.
3. Prepare data for building model using new software.
4. Improve CPU load issues of iOS app (for example, by reducing the frequency, but needed to be tested).
5. Try to optimize work of Watch app or left it the role of the device for getting notifications.
6. Add more functionality of app: settings, registration/authorization, sending GPS coordinates, etc.
7. Implement backend.
8. Implement Web-service for doctors.
9. Implement app's functionality for relatives.

One more thing could be implemented to improve defining walk pattern, to notify user for efficiently and to make predictions more personal is to implement Real-Time Machine Learning [39]. This means is to record the data when the person falls to use them for improving the model.

The project could be also ported to Android platform. But as there is a diversity of Android devices with varying degrees of reliability [18], the app should be tested on concrete Android devices and compiled the list of recommended devices to use for this project.

Chapter 7

Conclusion

The main goal of this work was to develop means for detecting and predicting falls among elderly people using walkers. This implies to choose hardware and software, develop apps for collecting motion data, analyze data, build machine learning model, convert this model to Core ML format and develop a prototype and integrate *Core ML* model to an app.

We actually created two experimental rigs:

- Walkers, three CC2650 sensors and iPhone where iPhone is the only controller, all data collecting from sensors.
- Walkers, Apple Watch and iPhone, where data collecting from both devices.

For each of rigs was developed its own application. The second rig has two apps: for iPhone and for Apple Watch.

As soon as apps were developed, we conducted experiments of collecting data. We divided activities into 3 types: "safe walking" patterns or normal walk, "relatively safe" walking patterns or initiations and "unsafe" walking patterns or falls. All these activities we recorded by me and Igor Molchanov on both rigs. For the first kind of activity we recorded 5-minute walk, for the second we recorded process of changing body from "safe" to "unsafe" position (about 25 different records each), for third - falls (about 25 different records each).

After collecting data, we conduct analysis to let us know whether or not collected data suits for predictions and which characteristics it is better to use. We concluded that magnetometer does not suit for prediction purpose and that we cannot predict fall (or detect "unsafe walking pattern") using Apple Watch. We concluded, that for prediction we can use only Gyroscope and Accelerometer.

For prediction purpose, we built Keras neural network as it is compatible with *Core ML* framework. For research purpose, we built the neural network using each of characteristics of each axis from each device we collected. To increase accuracy we also combined 2 parameters (one axis for gyro and one axis for accelerometer).

We concluded, that BLE sensors we used do not allow to predict fall for acceptable time (as the frequency is low) and decided to build prototype app only using data collected from the phone and watch. The app for phone ideally detects falls, but sometimes have errors in defining "relatively safe" walking patterns. Apple Watch app almost never identifies patterns correctly and needed to be revisioned.

Through the prototype app we developed, we proven that it is possible to build means for detecting and predicting falls using only data from phone and Core ML framework with relatively high accuracy.

Bibliography

[1] O'Loughlin J et al. Incidence of and risk factors for falls and injurious falls among the community-dwelling elderly. *American journal of epidemiology*, pages 342–54, 1993.

[2] PhD; Dirk Cambier PT PhD Tine Roman de Mettelinge, PT. Understanding the Relationship Between Walking Aids and Falls in Older Adults: A prospective cohort study. *Journal of GERIATRIC Physical Therapy*, 2015.

[3] Simo Hosio Denzil Ferreira Theodoros Anagnostopoulos Anabela Berenguer, Jorge Goncalves and Vassilis Kostakos. Increasing fall risk awareness using wearables: A fall risk awareness protocol. Journal of Biomedical Informatics. *IEEE Consumer Electronics magazine*, pages 104 – 110, 2017. ISSN 2162-2248/17.

[4] Elisabeth Razafimandimby Gangenes. Motion Capturing Machine Learning: Bluetooth motion sensors. *Master's Thesis in Computer Science SHO-6264-December-2016*, 2016.

[5] Anita Atwal Ioannis Paraskevopoulos Julian Hamm, Arthur G. Money a. Fall prevention intervention technologies: A conceptual framework and survey of the state of the art. *Journal of Biomedical Informatics*, 2016.

[6] Ming-Yih Lee Kin Fong Lei Wen-Cheng Chou, Wen-Yen Lin. Design and Assessment of a Real-Time Accelerometer-Based Lying-to-Sit Sensing System for Bed Fall Prevention. *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*, 2013.

[7] B. N. Ferreira, V. Guimares, and H. S. Ferreira. Smartphone based fall prevention exercises. *IEEE 15th International Conference on e-Health Networking, Applications and Services (Healthcom 2013)*, 2013.

[8] Zhang Wei-Bulstra Sjoerd Geraedts Hilde AE, Zijlstra Wiebren and Stevens Martin. Adherence to and effectiveness of an individually tailored home-based exercise program for frail older adults, driven by mobility monitoring: design of a prospective cohort study. *BMC Public Health*, 2014.

[9] Joel J. P. C. Rodrigues Sudip Misra Edgar T. Horta, Ivo C. Lopes. Real time falls prevention and detection with biofeedback monitoring solution for mobile environments. *IEEE 15th*

International Conference on e-Health Networking, Applications and Services (Healthcom 2013), 2013.

[10] Bernt Arild Bremdal Asbjørn Danielsen, Hans Olofsen. Predicting Bedside Falls using Current Context. *Increasing fall risk awareness using wearables: A fall risk awareness protocol*, 2016.

[11] Bernt Arild Bremdal Asbjørn Danielsen and Hans Olofsen. Fall Risk Assessment and Prevention Using Wearables. *Ambient Assisted Living*, 2015.

[12] Akm Jahangir Alam Majumder, Ishmat Zerin, Miftah Uddin, Sheikh I. Ahamed, and Roger O. Smith. smartprediction: A Real-time Smartphone-based Fall Risk Prediction and Prevention System. *Proceedings of the 2013 Research in Adaptive and Convergent Systems*, 2013.

[13] M. J. D. Otis and B. A. J. Menelas. Toward an augmented shoe for preventing falls related to physical conditions of the soil. *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2012.

[14] Romina Brignardello-Petersen Nancy Santesso, Alonso Carrasco-Labra. Hip protectors for preventing hip fractures in older people. *Cochrane Database of Systematic Reviews*, 2017.

[15] Danielsen Asbjørn and Torresen Jim. Recognizing Bedside Events Using Thermal and Ultrasonic Readings. *Sensors*, 2017.

[16] Danielsen Asbjørn and Bernt A. Bremdal. Predicting Bedside Falls using Current Context. *Computational Intelligence (SSCI), 2017 IEEE Symposium Series on*, 2017.

[17] Asbjørn Danielsen. Non-intrusive Bedside Event Recognition Using Infrared Array and Ultrasonic Sensor. *Ubiquitous Computing and Ambient Intelligence*, 2016.

[18] Blancco. State of Mobile Device Performance and Health. *Trend Report: Q2 2017*, pages 1 – 16, 2017.

[19] Apple. Core ML, 2017. URL https://developer.apple.com/documentation/coreml. [Online; accessed 16-May-2018].

[20] Apple. Deploying to Core ML, 2017. URL https://apple.github.io/turicreate/docs/userguide/activity_classifier/export_coreml.html. [Online; accessed 16-May-2018].

[21] Bluetooth SIG. Radio Versions, 2018. URL https://www.bluetooth.com/bluetooth-technology/radio-versions. [Online; accessed 16-May-2018].

[22] Andreas Knecht. modum.io Software on the CC2650 SensorTag Portability, Security and General Architecture. *University of Zurich SOFTWARE PROJECT Communication Systems Group*, 2017.

[23] Kevin Townsend. Introduction to Bluetooth Low Energy, 2015. URL https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gatt. [Online; accessed 16-May-2018].

[24] Apple. Core Bluetooth, 2017. URL https://developer.apple.com/documentation/corebluetooth. [Online; accessed 16-May-2018].

[25] Apple. Watch Connectivity, 2017. URL https://developer.apple.com/documentation/watchconnectivity. [Online; accessed 16-May-2018].

[26] Evan K. Stone. Zero to BLE on iOS, 2017. URL https://www.cloudcity.io/blog/2015/06/11/zero-to-ble-on-ios-part-one/. [Online; accessed 16-May-2018].

[27] Texas Instruments contributors. CC2650 SensorTag User's Guide, 2018. URL http://processors.wiki.ti.com/index.php/CC2650_SensorTag_User's_Guide. [Online; accessed 16-May-2018].

[28] Apple. Core Data, 2017. URL https://developer.apple.com/library/content/documentation/Cocoa/Conceptual/CoreData/index.html. [Online; accessed 16-May-2018].

[29] Apple. Core Motion, 2017. URL https://developer.apple.com/documentation/coremotion. [Online; accessed 16-May-2018].

[30] Apple. Leveraging iOS Technologies, 2017. URL https://developer.apple.com/library/content/documentation/General/Conceptual/WatchKitProgrammingGuide/iOSSupport.html#//apple_ref/doc/uid/TP40014969-CH21-SW10. [Online; accessed 16-May-2018].

[31] Fabio Feldman-PhD* Yijian Yang MD Rebecca Schonnop BSc Pet Ming Lueng MSc Thiago Sarraf MSc Joanie Sims-Gould PhD Prof. Stephen N Robinovitch, PhD* and MSc Marie Loughin. Video capture of the circumstances of falls in elderly people residing in long-term care: an observational study. *Lancet. 2013 January 05; 381(9860)*, page 47 54, 2013.

[32] Viktor Malyi. Run or Walk. *Towards Data Science*, 2016.

[33] Apple. Energy Efficiency Guide for iOS Apps, 2017. URL https://developer.apple.com/library/content/documentation/Performance/Conceptual/EnergyGuide-iOS/WorkLessInTheBackground.html. [Online; accessed 16-May-2018].

[34] Ning Jia. Detecting Human Falls with a 3-Axis Digital Accelerometer. *AnalogDialog*, 2009.

[35] Yan Zhao Falin Wu, Hengyang Zhao and Haibo Zhong. Development of a Wearable-Sensor-Based Fall Detection System. *International Journal of Telemedicine and Applications*, 2015.

[36] MbientLab Inc. MetaWear: Budget Conscious Tiny Programmable BLE Sensors, 2015. URL https://www.kickstarter.com/projects/guardyen/tiny-programmable-ble-sensors-that-wont-break-your. [Online; accessed 16-May-2018].

[37] MobileModding Tech blog. TI SensorTag 2 Power consumption analysys, 2015. URL http://mobilemodding.info/2015/06/ti-sensortag-2-power-consumption-analysys/. [Online; accessed 16-May-2018].

[38] Stefan Hey Andre Bideaux, Bernd Zimmermann and Wilhelm Stork. Synchronization in wireless biomedical-sensor networks with Bluetooth Low Energy. *Current Directions in Biomedical Engineering 2015; 1:7376*, 2015.

[39] Stephanie Wang Alexey Tumanov William Paul Johann Schleier-Smith Richard Liaw Mehrdad Niknami Michael I. Jordan Ion Stoica Robert Nishihara, Philipp Moritz. Real-Time Machine Learning: The Missing Pieces. *HotOS '17 Proceedings of the 16th Workshop on Hot Topics in Operating Systems*, pages 106–110, 2017.

Appendix A

Flowchart of working process

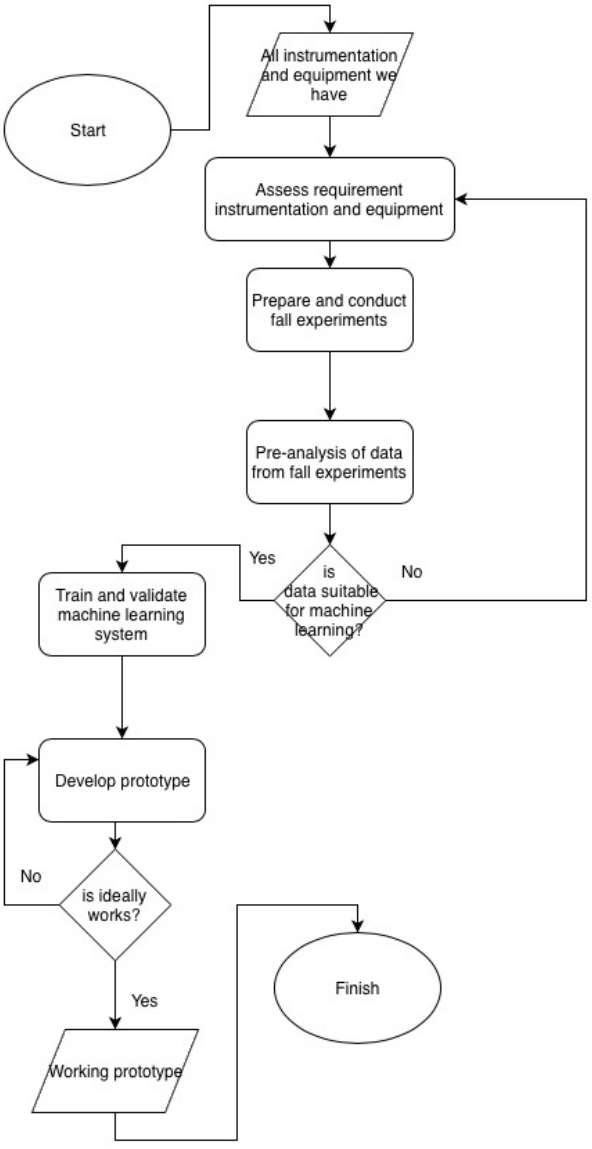


FIGURE A.1: Flowchart of working process

Appendix B

Conceptual model of falls prevention technology

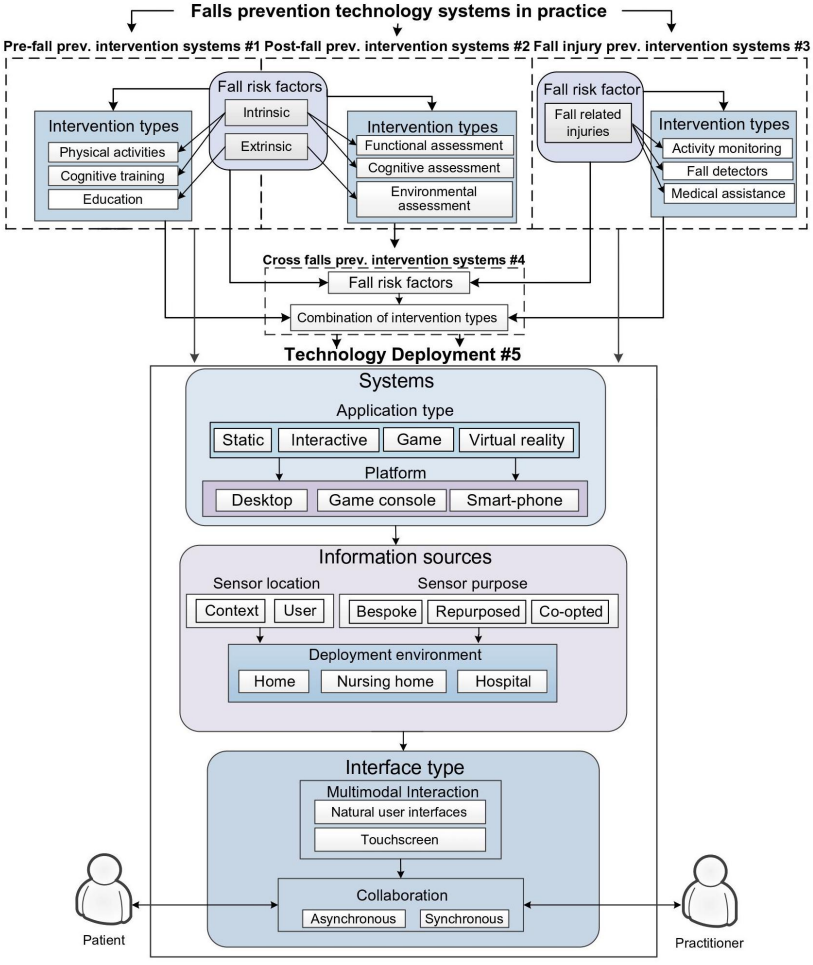


FIGURE B.1: Conceptual model of falls prevention technology

Source: J. Hamm et al. / Journal of Biomedical Informatics 59 (2016) 319345

Appendix C

Analysis of data collected from 3 SensorTags

Console output

```
/Library/Frameworks/Python.framework/Versions/3.6/bin/python3 /Users/delexa/
    Desktop/ModelGenerator/analyze.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7429 entries, 0 to 7428
Data columns (total 39 columns):
SessionID                   7429 non-null int64
SessionDate                 7429 non-null object
SessionDuration             7429 non-null object
SessionPeriod               7429 non-null float64
AmountOfSensors             7429 non-null int64
RecordID                    7429 non-null int64
Timestamp1                  7421 non-null object
timeIntervalSince1970_1     7421 non-null float64
GyroX1                      7421 non-null float64
GyroY1                      7421 non-null float64
GyroZ1                      7421 non-null float64
AccX1                       7421 non-null float64
AccY1                       7421 non-null float64
AccZ1                       7421 non-null float64
MagX1                       7421 non-null float64
MagY1                       7421 non-null float64
MagZ1                       7421 non-null float64
Timestamp2                  7415 non-null object
timeIntervalSince1970_2     7415 non-null float64
GyroX2                      7415 non-null float64
GyroY2                      7415 non-null float64
GyroZ2                      7415 non-null float64
```

```
27 AccX2                          7415 non-null float64
   AccY2                          7415 non-null float64
29 AccZ2                          7415 non-null float64
   MagX2                          7415 non-null float64
31 MagY2                          7415 non-null float64
   MagZ2                          7415 non-null float64
33 Timestamp3                     7417 non-null object
   timeIntervalSince1970_3        7417 non-null float64
35 GyroX3                         7417 non-null float64
   GyroY3                         7417 non-null float64
37 GyroZ3                         7417 non-null float64
   AccX3                          7417 non-null float64
39 AccY3                          7417 non-null float64
   AccZ3                          7417 non-null float64
41 MagX3                          7417 non-null float64
   MagY3                          7417 non-null float64
43 MagZ3                          7417 non-null float64
   dtypes: float64(31), int64(3), object(5)
45 memory usage: 2.2+ MB
   The dataset contains 7429 data samples and 39 data columns
47 SessionID                       0
   SessionDate                     0
49 SessionDuration                 0
   SessionPeriod                   0
51 AmountOfSensors                 0
   RecordID                        0
53 Timestamp1                      8
   timeIntervalSince1970_1         8
55 GyroX1                          8
   GyroY1                          8
57 GyroZ1                          8
   AccX1                           8
59 AccY1                           8
   AccZ1                           8
61 MagX1                           8
   MagY1                           8
63 MagZ1                           8
   Timestamp2                     14
65 timeIntervalSince1970_2        14
   GyroX2                         14
67 GyroY2                         14
   GyroZ2                         14
69 AccX2                          14
   AccY2                          14
71 AccZ2                          14
   MagX2                          14
73 MagY2                          14
   MagZ2                          14
```

```
Timestamp3                  12
timeIntervalSince1970_3     12
GyroX3                      12
GyroY3                      12
GyroZ3                      12
AccX3                       12
AccY3                       12
AccZ3                       12
MagX3                       12
MagY3                       12
MagZ3                       12
dtype: int64
SessionID   SessionPeriod   AmountOfSensors      RecordID  \
count   7429.000000          7429.0            7429.0   7429.000000
mean       2.109032             0.1               3.0      0.293579
std        5.012527             0.0               0.0      0.646925
min        0.000000             0.1               3.0      0.000000
25%        0.000000             0.1               3.0      0.000000
50%        0.000000             0.1               3.0      0.000000
75%        0.000000             0.1               3.0      0.000000
max       20.000000             0.1               3.0      2.000000

timeIntervalSince1970_1        GyroX1        GyroY1        GyroZ1  \
count              7.421000e+03  7421.000000  7421.000000  7421.000000
mean               1.524754e+09     4.798598     1.808747     0.485961
std                5.010783e+02    11.749951     5.073385     3.542819
min                1.524753e+09  -122.505000   -87.867700   -57.296700
25%                1.524753e+09    -0.793457    -0.068665    -0.213623
50%                1.524754e+09     3.913880     1.510620     0.480652
75%                1.524754e+09    10.406500     3.425600     1.136780
max                1.524755e+09   173.775000   139.137000    72.120700

AccX1         AccY1      ...       timeIntervalSince1970_3  \
count   7421.000000   7421.000000     ...                    7.417000e+03
mean       7.826345      2.151173     ...                    1.524754e+09
std        0.597147      0.728369     ...                    5.014980e+02
min       -6.791020    -16.000000     ...                    1.524753e+09
25%        7.750000      2.000000     ...                    1.524753e+09
50%        7.843750      2.189450     ...                    1.524754e+09
75%        7.937500      2.378910     ...                    1.524754e+09
max       15.999500     15.999500     ...                    1.524755e+09

GyroX3        GyroY3        GyroZ3         AccX3         AccY3  \
count   7417.000000   7417.000000   7417.000000   7417.000000   7417.000000
mean      -1.185217      0.286809     -0.712268     -0.831244      0.268602
std       26.789268     30.590364     17.011276      0.537666      1.674093
min     -179.329000   -250.000000   -167.015000     -6.511230     -3.319820
25%      -17.654400    -14.831500     -7.591250     -1.205570     -1.550780
```

```
50%      -7.240290     0.282288     2.502440    -0.791016     1.403810
75%      12.321500    14.511100     8.483890    -0.479980     1.886230
max     249.992000   249.992000   171.577000     2.874020     8.816410

AccZ3          MagX3          MagY3          MagZ3
count   7417.000000    7417.000000    7417.000000    7417.000000
mean      -0.166580     923.383983     276.965485      64.720911
std        0.531404     210.614916      81.803291     265.499163
min       -5.143550     429.000000    -116.000000    -538.000000
25%       -0.443848     733.000000     225.000000    -190.000000
50%       -0.156738     814.000000     266.000000      86.000000
75%        0.088867    1134.000000     325.000000     315.000000
max       11.610800    1332.000000    1216.000000    1072.000000

[8 rows x 34 columns]
Dataset contains 6032 "safe" data samples as well as 613 "relatively safe" data
    samples and 613 "unsafe" data samples
SessionID                SessionDate SessionDuration  SessionPeriod  \
0            0  2018-04-26 16:34:00.4030             05:01            0.1
1            0  2018-04-26 16:34:00.4030             05:01            0.1
2            0  2018-04-26 16:34:00.4030             05:01            0.1
3            0  2018-04-26 16:34:00.4030             05:01            0.1
4            0  2018-04-26 16:34:00.4030             05:01            0.1
5            0  2018-04-26 16:34:00.4030             05:01            0.1
6            0  2018-04-26 16:34:00.4030             05:01            0.1
7            0  2018-04-26 16:34:00.4030             05:01            0.1
8            0  2018-04-26 16:34:00.4030             05:01            0.1
9            0  2018-04-26 16:34:00.4030             05:01            0.1
10           0  2018-04-26 16:34:00.4030             05:01            0.1
11           0  2018-04-26 16:34:00.4030             05:01            0.1
12           0  2018-04-26 16:34:00.4030             05:01            0.1
13           0  2018-04-26 16:34:00.4030             05:01            0.1
14           0  2018-04-26 16:34:00.4030             05:01            0.1
15           0  2018-04-26 16:34:00.4030             05:01            0.1
16           0  2018-04-26 16:34:00.4030             05:01            0.1
17           0  2018-04-26 16:34:00.4030             05:01            0.1
18           0  2018-04-26 16:34:00.4030             05:01            0.1
19           0  2018-04-26 16:34:00.4030             05:01            0.1

AmountOfSensors  RecordID                Timestamp1  \
0                      3         0  2018-04-26 16:34:00.4760
1                      3         0  2018-04-26 16:34:00.5940
2                      3         0  2018-04-26 16:34:00.6440
3                      3         0  2018-04-26 16:34:00.7770
4                      3         0  2018-04-26 16:34:00.8940
5                      3         0  2018-04-26 16:34:00.9440
6                      3         0  2018-04-26 16:34:01.0770
7                      3         0  2018-04-26 16:34:01.1940
```

```
    8                   3           0   2018-04-26  16:34:01.2440
171 9                   3           0   2018-04-26  16:34:01.3770
    10                  3           0   2018-04-26  16:34:01.4940
173 11                  3           0   2018-04-26  16:34:01.5440
    12                  3           0   2018-04-26  16:34:01.6770
175 13                  3           0   2018-04-26  16:34:01.7940
    14                  3           0   2018-04-26  16:34:01.8440
177 15                  3           0   2018-04-26  16:34:01.9770
    16                  3           0   2018-04-26  16:34:02.0940
179 17                  3           0   2018-04-26  16:34:02.1440
    18                  3           0   2018-04-26  16:34:02.2760
181 19                  3           0   2018-04-26  16:34:02.3940

183 timeIntervalSince1970_1     GyroX1     GyroY1   ...      \
    0                1.524753e+09  1.159670   0.625610   ...
185 1                1.524753e+09  1.205440   0.572205   ...
    2                1.524753e+09  1.205440   0.587463   ...
187 3                1.524753e+09  1.281740   0.595093   ...
    4                1.524753e+09  1.197810   0.617981   ...
189 5                1.524753e+09  1.159670   0.617981   ...
    6                1.524753e+09  1.342770   0.488281   ...
191 7                1.524753e+09  1.121520   0.648498   ...
    8                1.524753e+09  1.297000   0.656128   ...
193 9                1.524753e+09  1.205440   0.663757   ...
    10               1.524753e+09  1.121520   0.602722   ...
195 11               1.524753e+09  0.923157   0.541687   ...
    12               1.524753e+09  1.106260   0.709534   ...
197 13               1.524753e+09  1.159670   0.534058   ...
    14               1.524753e+09  1.312260   0.457764   ...
199 15               1.524753e+09  1.266480   0.595093   ...
    16               1.524753e+09  1.251220   0.572205   ...
201 17               1.524753e+09  1.335140   0.831604   ...
    18               1.524753e+09 -3.196720   0.068665   ...
203 19               1.524753e+09  2.792360  -2.403260   ...

205 timeIntervalSince1970_3     GyroX3     GyroY3     GyroZ3     AccX3  \
    0                1.524753e+09  -0.556946   0.892639  -0.205994 -1.375980
207 1                1.524753e+09  -0.335693   0.762939  -0.556946 -1.354490
    2                1.524753e+09  -0.427246   0.526428  -0.839233 -1.354000
209 3                1.524753e+09   0.205994   1.106260  -0.938415 -1.352540
    4                1.524753e+09   0.419617   1.007080  -1.258850 -1.347660
211 5                1.524753e+09  -1.251220  -1.243590  -1.716610 -1.367190
    6                1.524753e+09  -1.419070  -0.312805  -1.571660 -1.351560
213 7                1.524753e+09  -4.241940  -2.120970  -1.403810 -1.358890
    8                1.524753e+09  -7.225040  -5.874630   1.358030 -1.363770
215 9                1.524753e+09  -6.172180  -4.707340   3.341670 -1.336430
    10               1.524753e+09   1.655580  -2.204900  -1.388550 -1.261720
217 11               1.524753e+09   6.950380 -21.087600 -15.228300 -1.246580
```

```
    12              1.524753e+09  19.966100 -17.639200 -25.688200 -1.189940
219 13              1.524753e+09  10.505700 -16.464200 -43.830900 -0.898438
    14              1.524753e+09  -0.480652  42.488100 -46.836800 -0.930664
221 15              1.524753e+09   6.050110  54.336500 -27.275100 -1.095210
    16              1.524753e+09  20.431500  34.812900 -20.843500 -1.262700
223 17              1.524753e+09  23.498500  -4.783630  -4.943850 -1.002930
    18              1.524753e+09  27.359000  10.147100   0.892639 -0.739258
225 19              1.524753e+09  -8.621220  22.979700   9.017940 -1.012700

227 AccY3       AccZ3    MagX3   MagY3   MagZ3
    0  -1.38867  0.330566  1164.0   289.0   264.0
229 1  -1.39893  0.343750  1167.0   291.0   273.0
    2  -1.39746  0.333496  1166.0   293.0   270.0
231 3  -1.41357  0.330078  1164.0   301.0   273.0
    4  -1.38672  0.320312  1159.0   290.0   267.0
233 5  -1.39990  0.309570  1164.0   291.0   275.0
    6  -1.40234  0.327148  1167.0   291.0   273.0
235 7  -1.39600  0.335449  1164.0   289.0   266.0
    8  -1.40430  0.291992  1163.0   290.0   267.0
237 9  -1.45264  0.272949  1163.0   300.0   267.0
    10 -1.48389  0.314453  1157.0   294.0   255.0
239 11 -1.52881  0.412109  1154.0   290.0   251.0
    12 -1.53516  0.525391  1148.0   296.0   255.0
241 13 -1.62354  0.460938  1166.0   288.0   270.0
    14 -1.53613  0.245605  1159.0   276.0   283.0
243 15 -1.65576  0.362793  1145.0   260.0   268.0
    16 -1.61133  0.448730  1112.0   249.0   271.0
245 17 -1.56934  0.519043  1132.0   245.0   354.0
    18 -1.63916  0.538086  1172.0   232.0   378.0
247 19 -1.41846  0.447754  1183.0   234.0   371.0

249 [20 rows x 39 columns]

251 Process finished with exit code 0
```

Listing C.1: Console output

Sensor data distribution plots

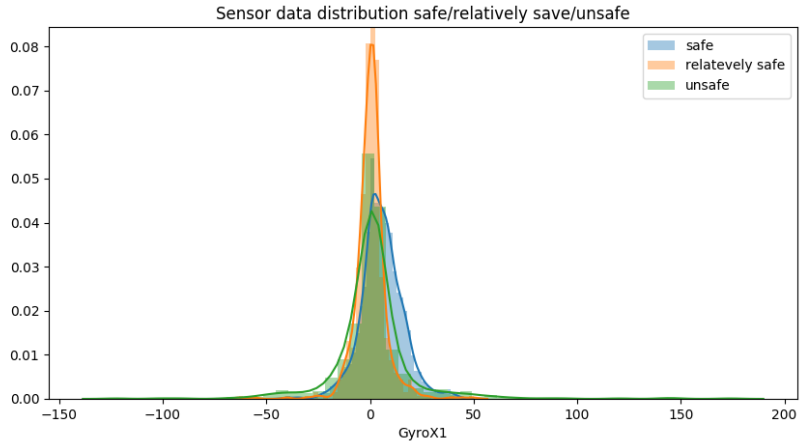


FIGURE C.1: Sensor data distribution GyroX1

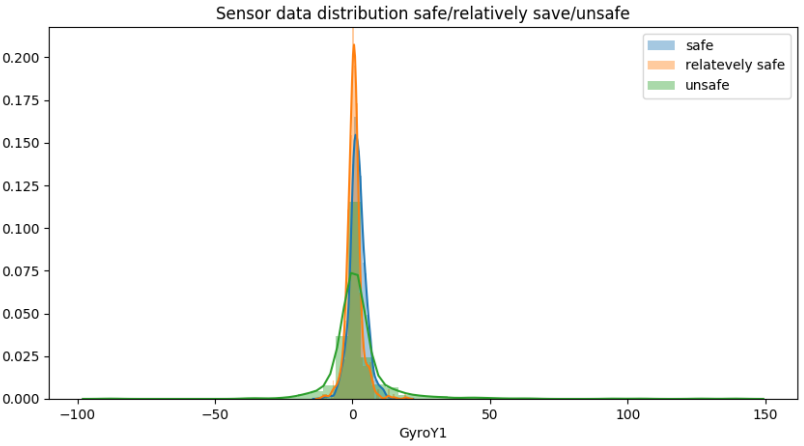


FIGURE C.2: Sensor data distribution GyroY1

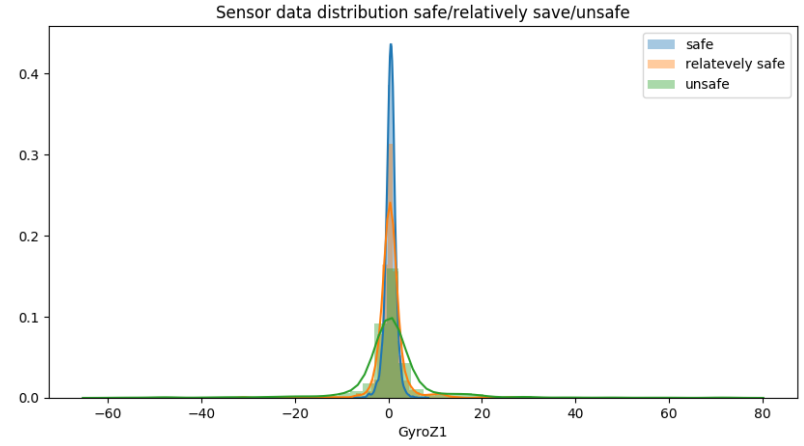


FIGURE C.3: Sensor data distribution GyroZ1

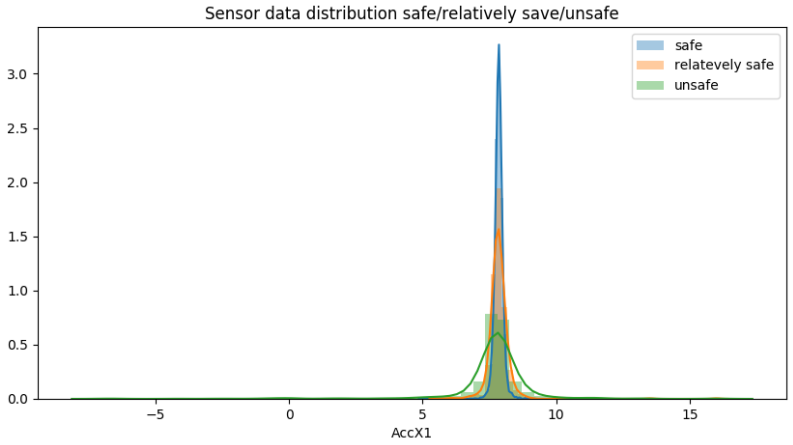


FIGURE C.4: Sensor data distribution AccX1

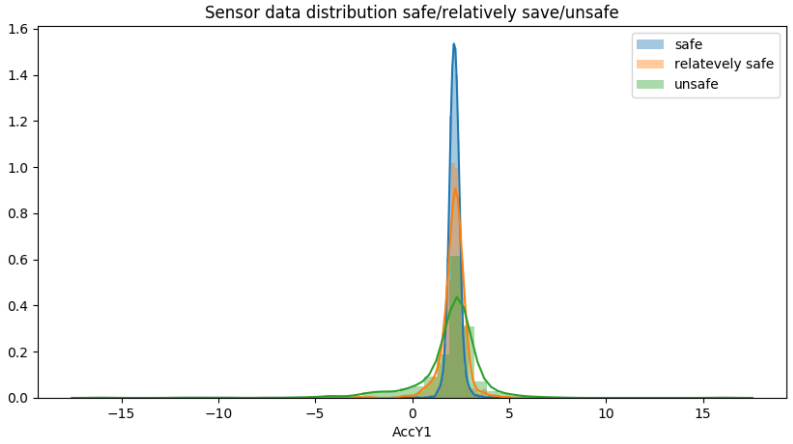


FIGURE C.5: Sensor data distribution AccY1

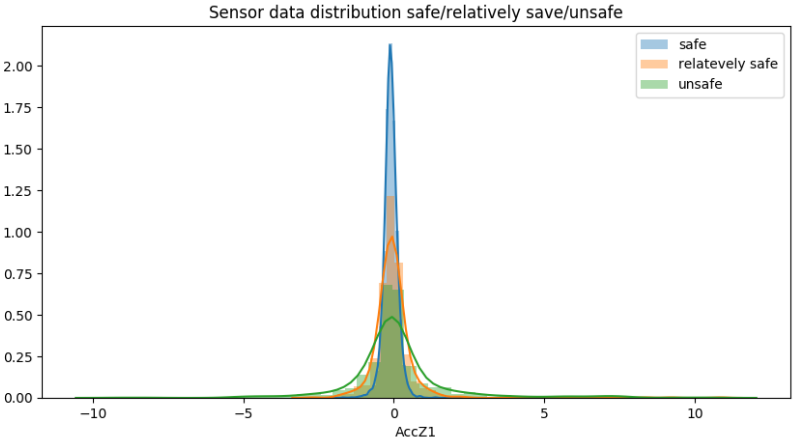


FIGURE C.6: Sensor data distribution AccZ1

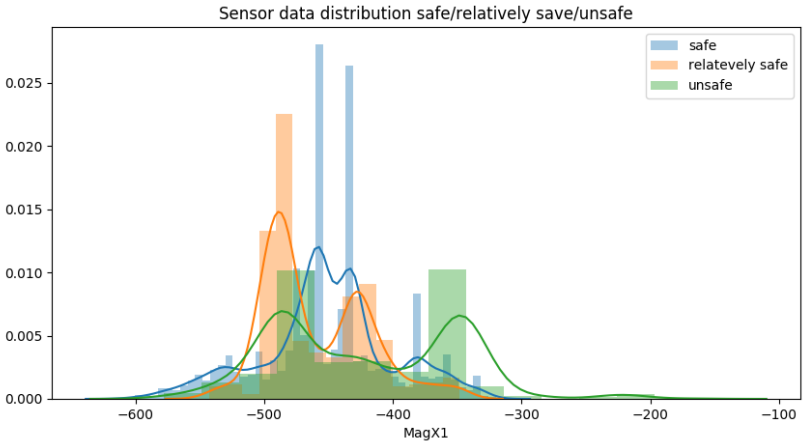


FIGURE C.7: Sensor data distribution MagX1

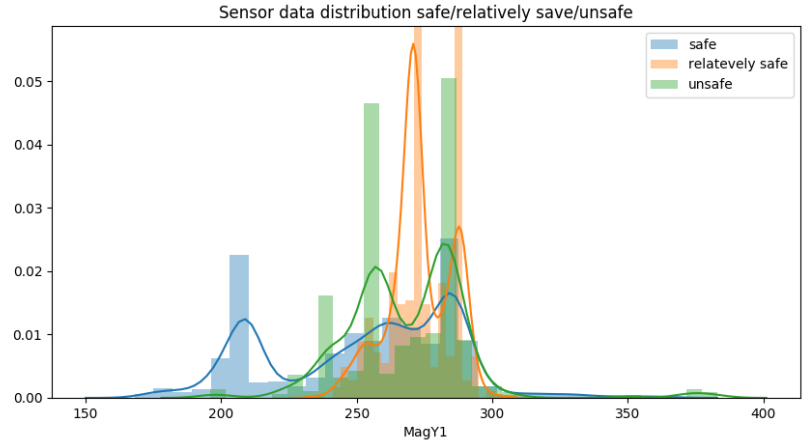


FIGURE C.8: Sensor data distribution MagY1

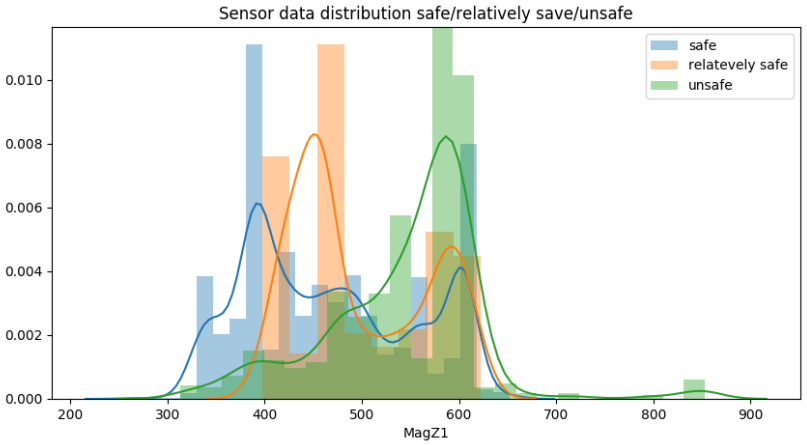


FIGURE C.9: Sensor data distribution MagZ1

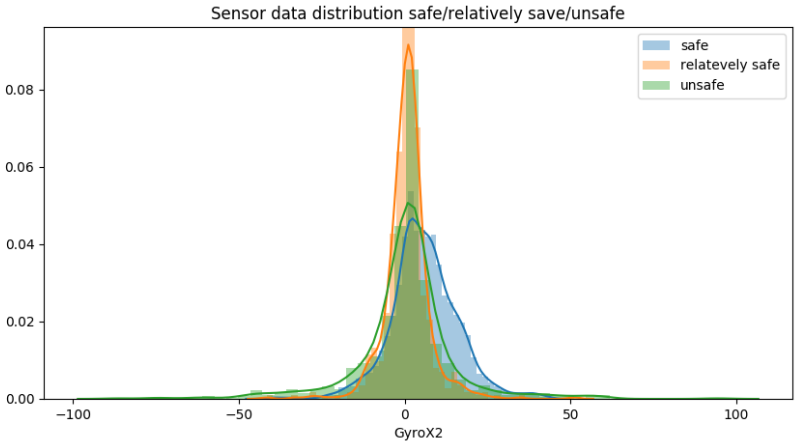


FIGURE C.10: Sensor data distribution GyroX2

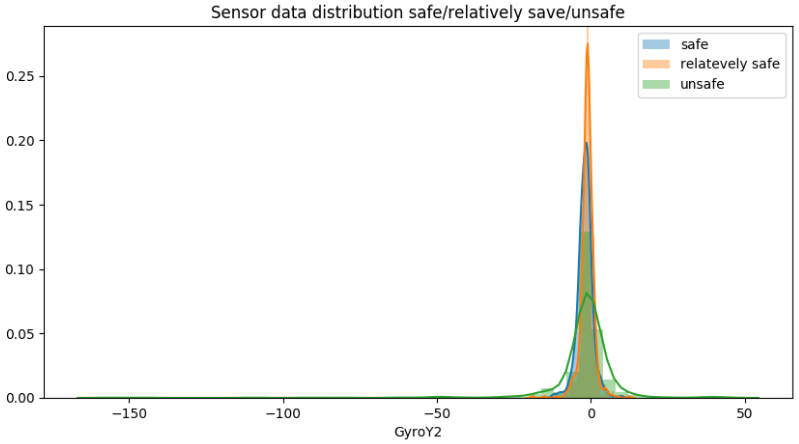


FIGURE C.11: Sensor data distribution GyroY2

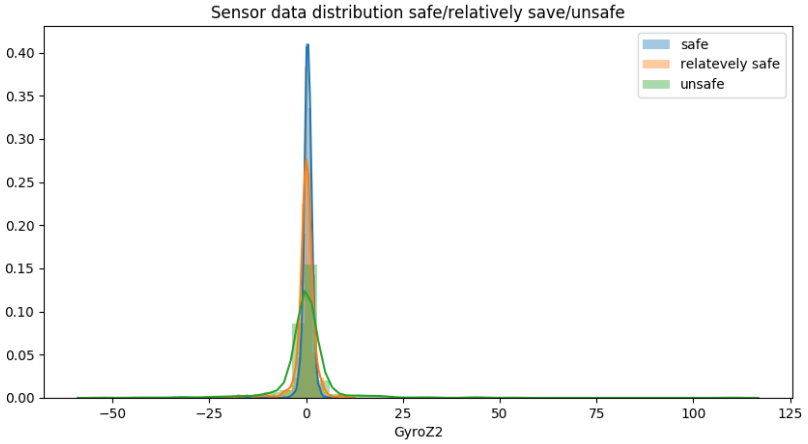


FIGURE C.12: Sensor data distribution GyroZ2

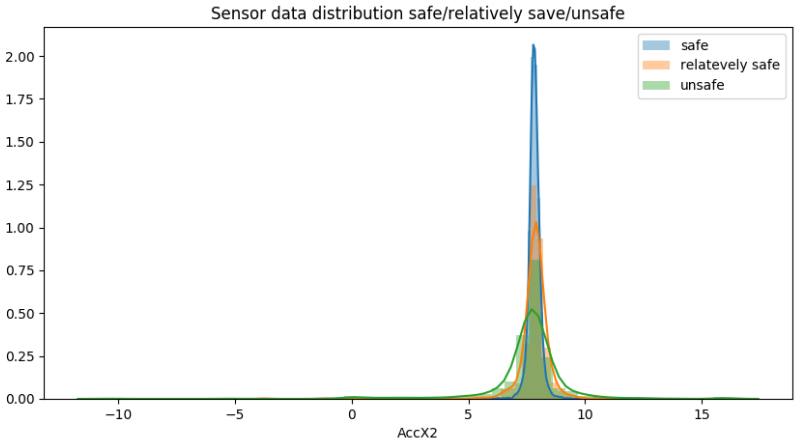


FIGURE C.13: Sensor data distribution AccX2

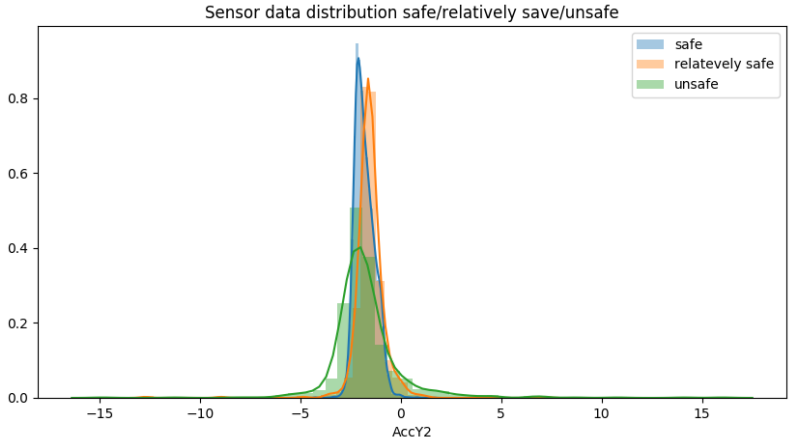


FIGURE C.14: Sensor data distribution AccY2

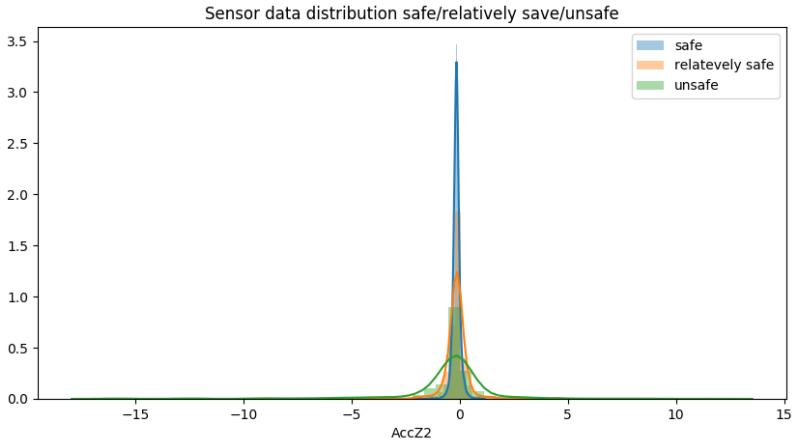


FIGURE C.15: Sensor data distribution AccZ2

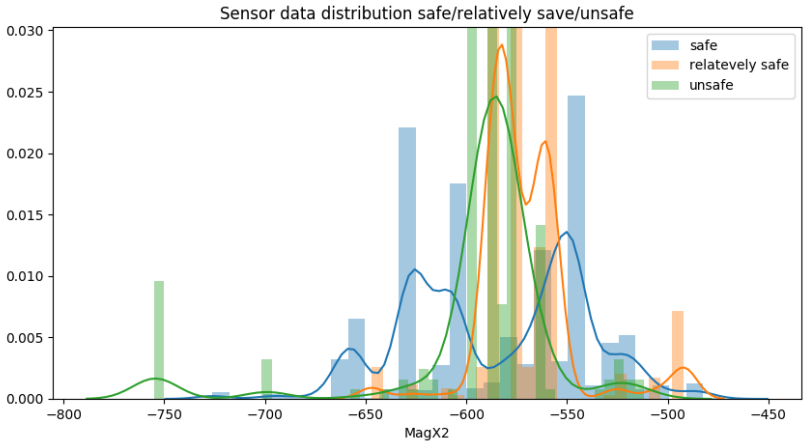


FIGURE C.16: Sensor data distribution MagX2

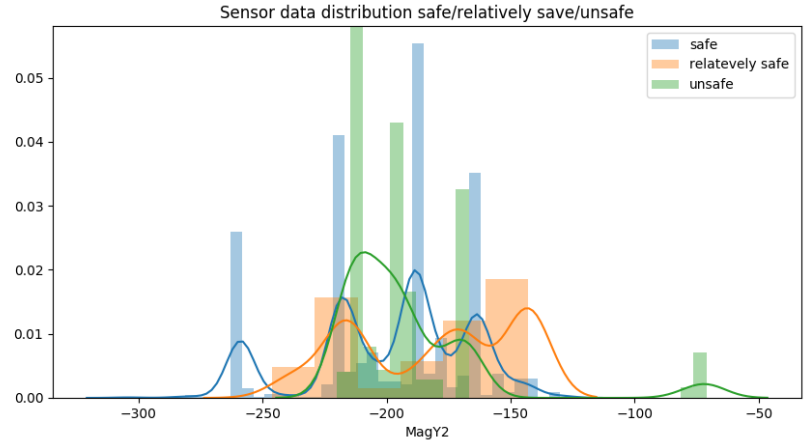


FIGURE C.17: Sensor data distribution MagY2

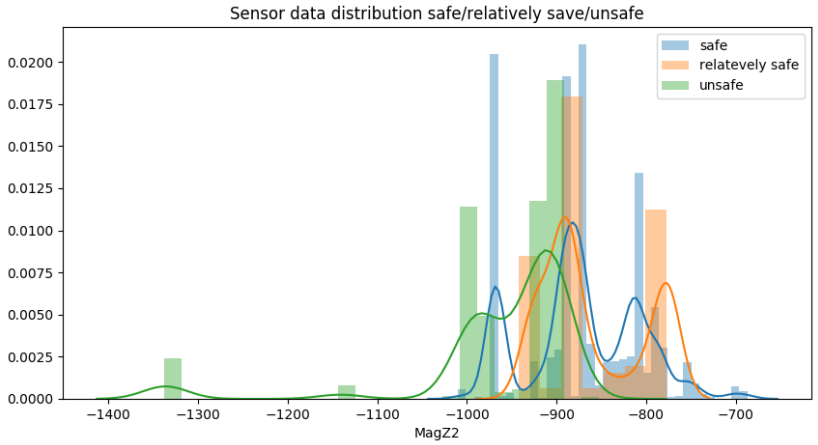


FIGURE C.18: Sensor data distribution MagZ2

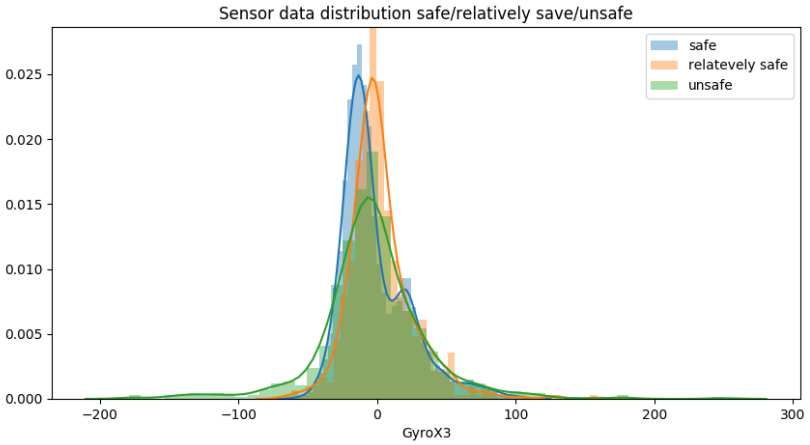


FIGURE C.19: Sensor data distribution GyroX3

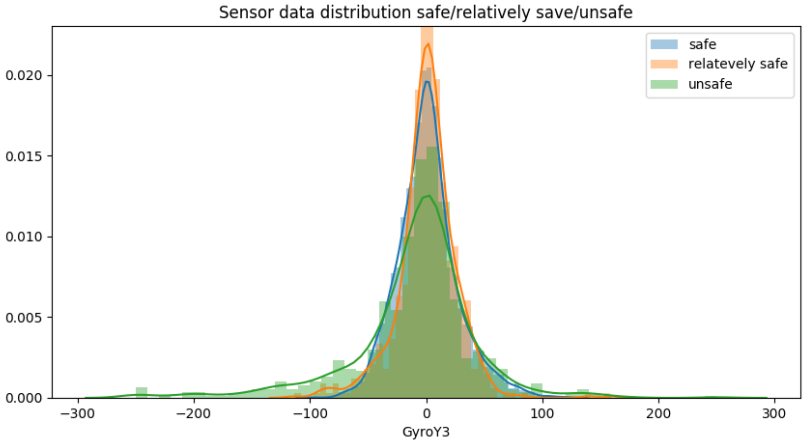


FIGURE C.20: Sensor data distribution GyroY3

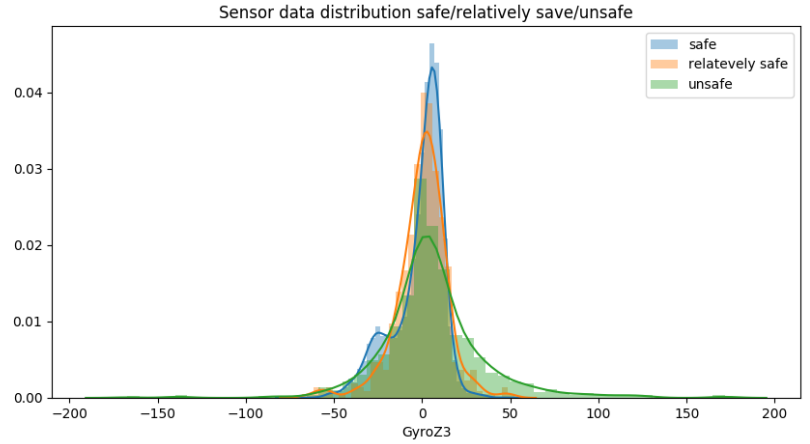


FIGURE C.21: Sensor data distribution GyroZ3

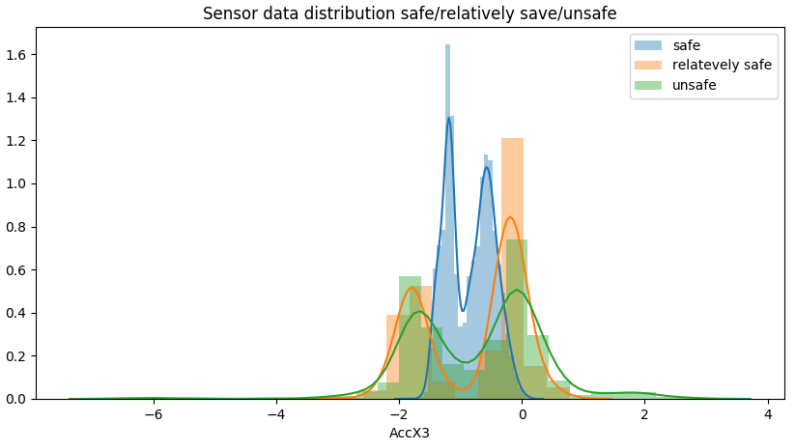


FIGURE C.22: Sensor data distribution AccX3

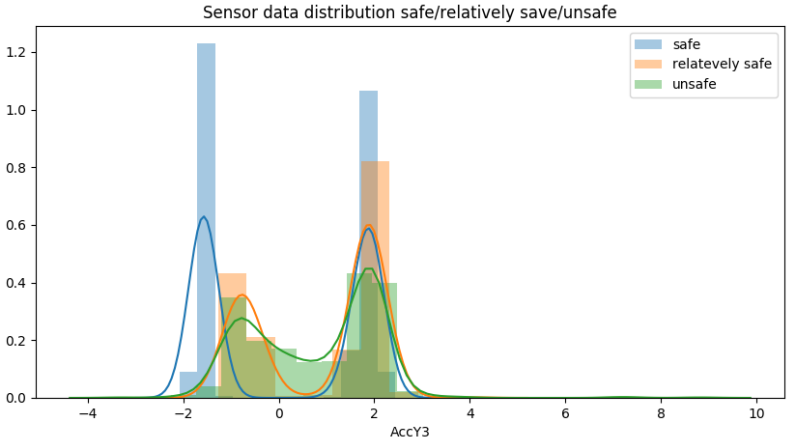


FIGURE C.23: Sensor data distribution AccY3

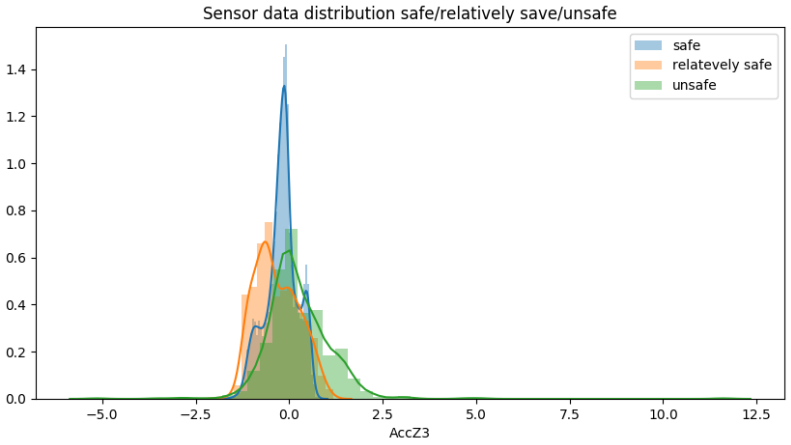


FIGURE C.24: Sensor data distribution AccZ3

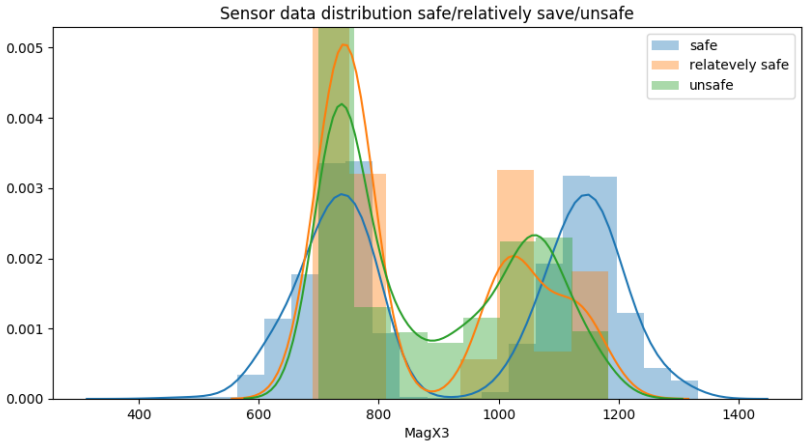


FIGURE C.25: Sensor data distribution MagX3

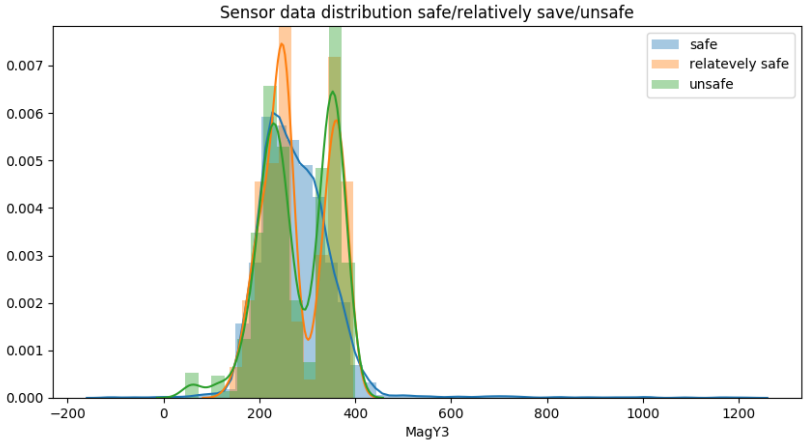


FIGURE C.26: Sensor data distribution MagY3

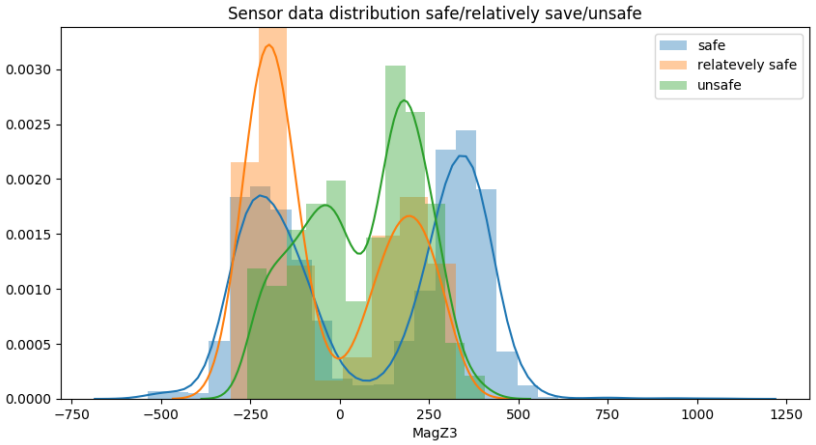


FIGURE C.27: Sensor data distribution MagZ3

Sensor data plots

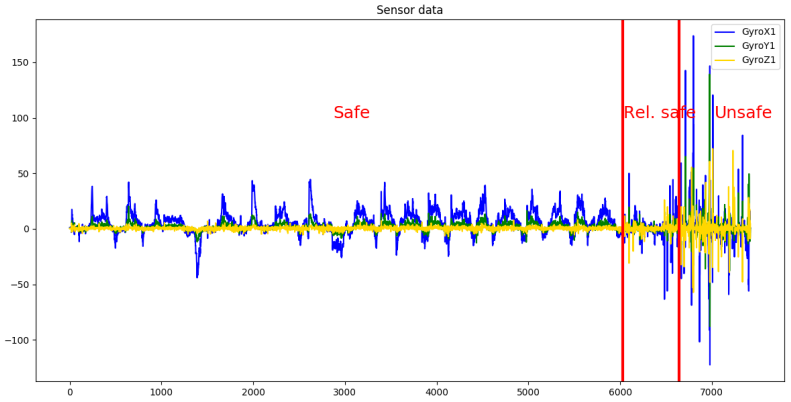


FIGURE C.28: Data plot GyroX1, GyroY1, GyroZ1

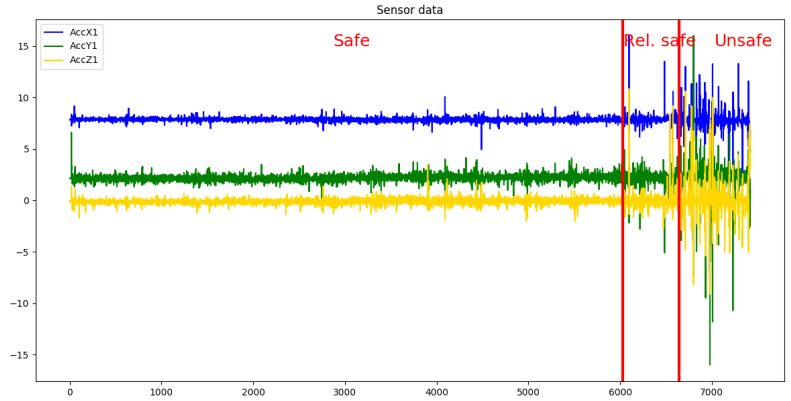


FIGURE C.29: Data plot AccX1, AccY1, AccZ1

FIGURE C.30: Data plot MagX1, MagY1, MagZ1

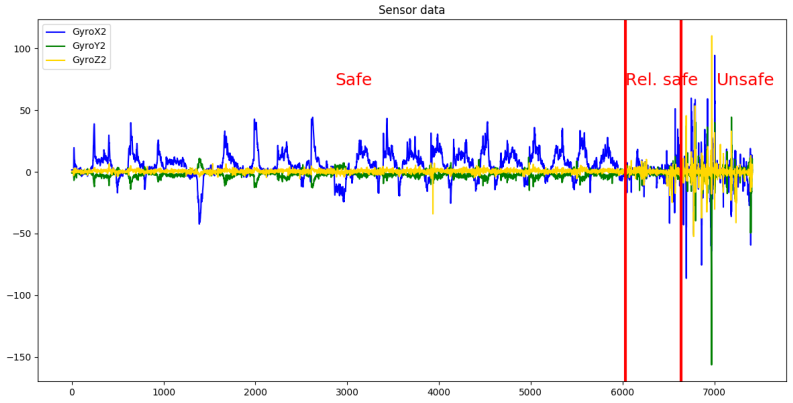


FIGURE C.31: Data plot GyroX2, GyroY2, GyroZ2

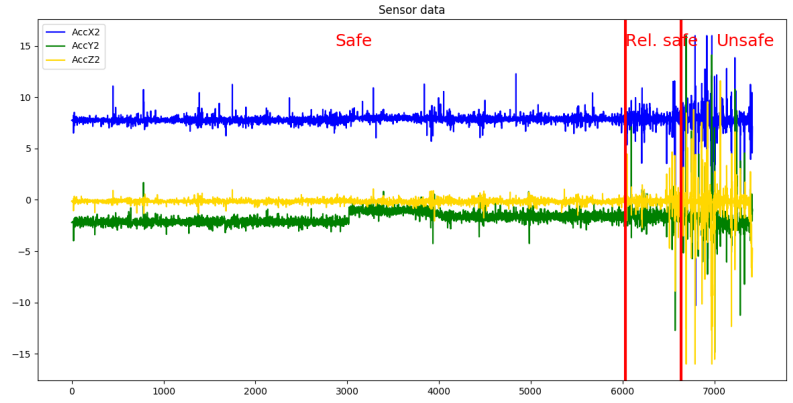


FIGURE C.32: Data plot AccX2, AccY2, AccZ2

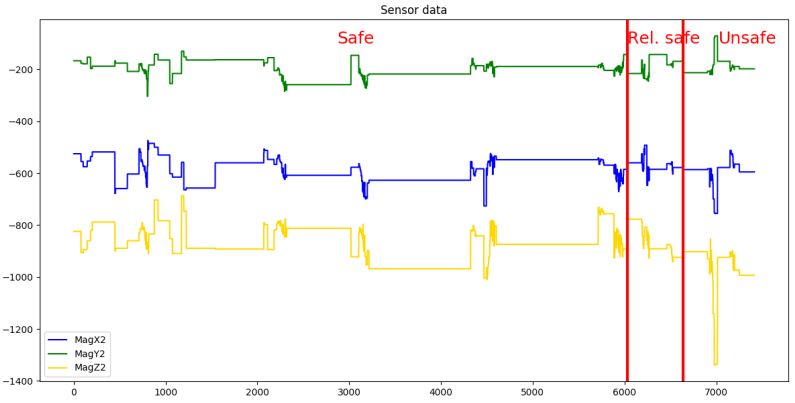


FIGURE C.33: Data plot MagX2, MagY2, MagZ2

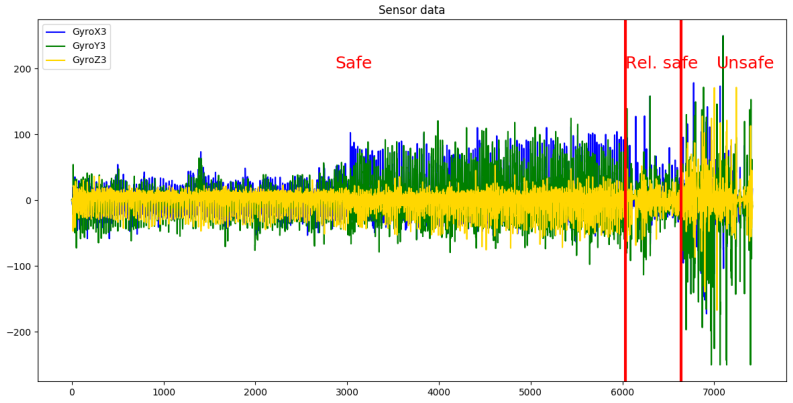


FIGURE C.34: Data plot GyroX3, GyroY3, GyroZ3

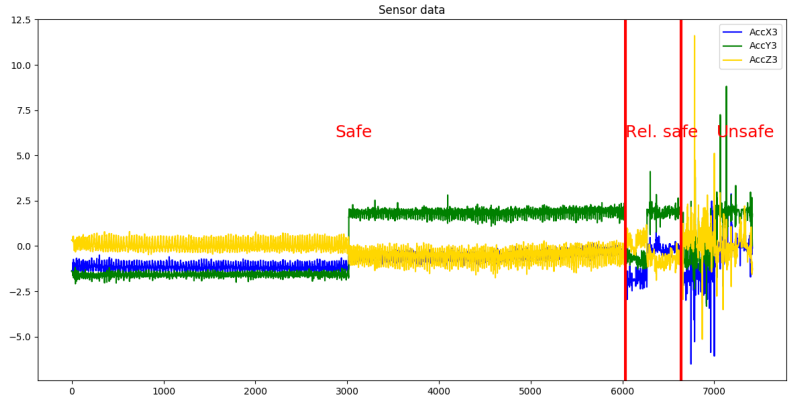


FIGURE C.35: Data plot AccX3, AccY3, AccZ3

FIGURE C.36: Data plot MagX3, MagY3, MagZ3

Comparison single characteristic of all sessions

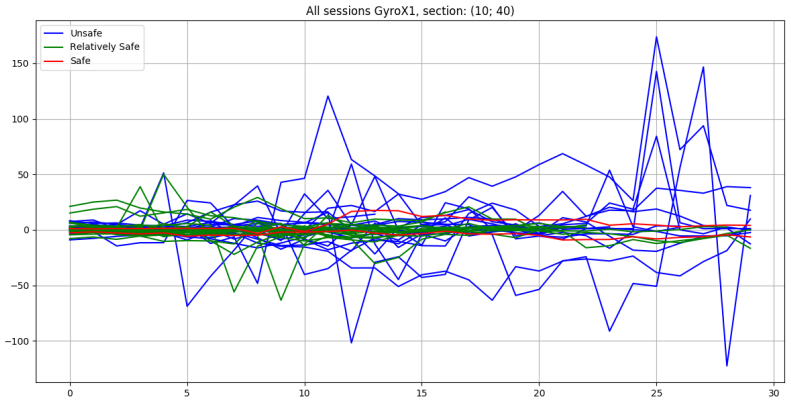


FIGURE C.37: GyroX1, all sessions comparison

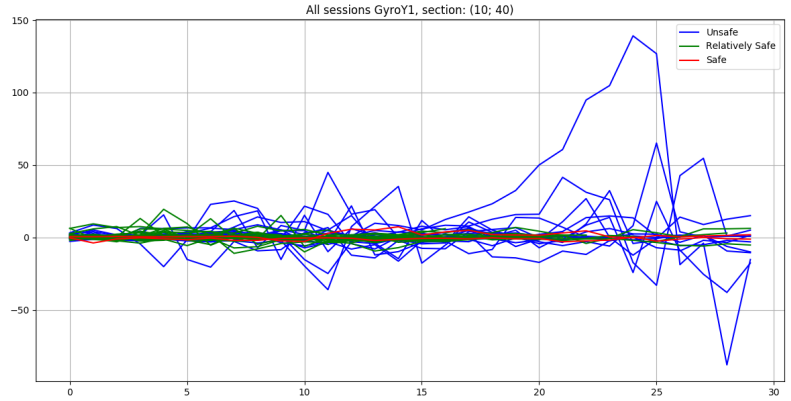


FIGURE C.38: GyroY1, all sessions comparison

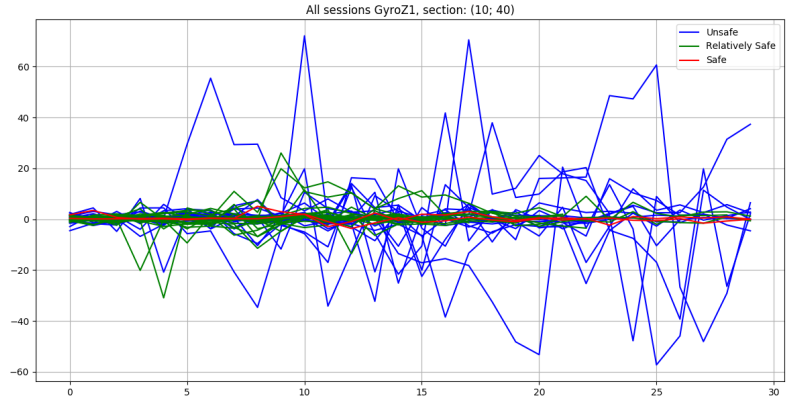


FIGURE C.39: GyroZ1, all sessions comparison

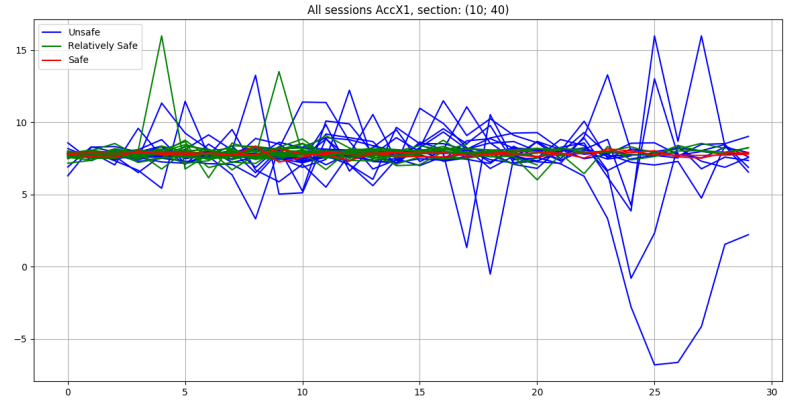


FIGURE C.40: AccX1, all sessions comparison

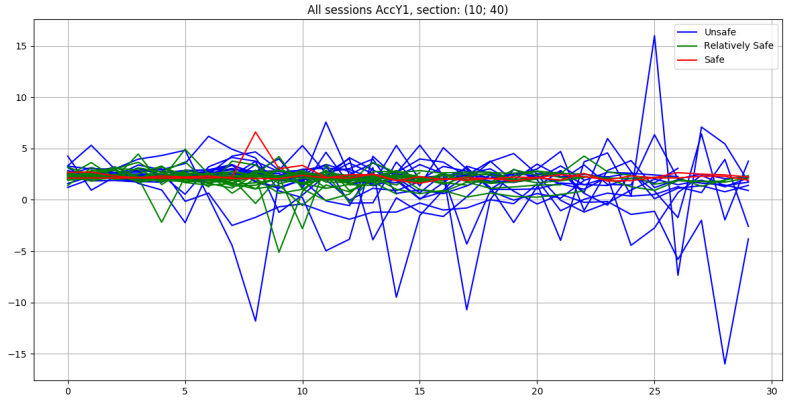


FIGURE C.41: AccY1, all sessions comparison

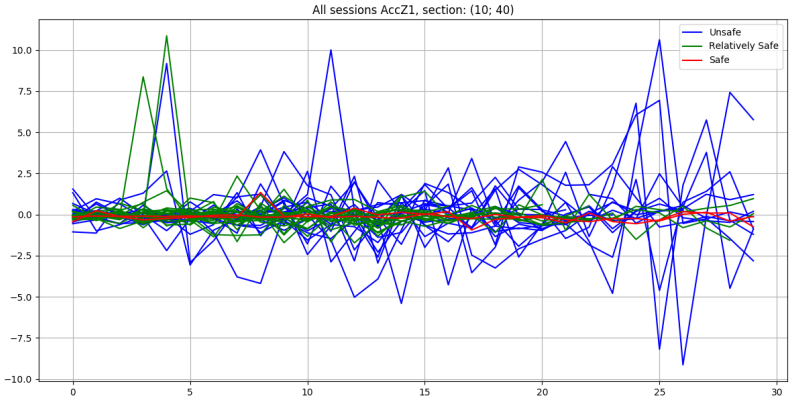


FIGURE C.42: AccZ1, all sessions comparison

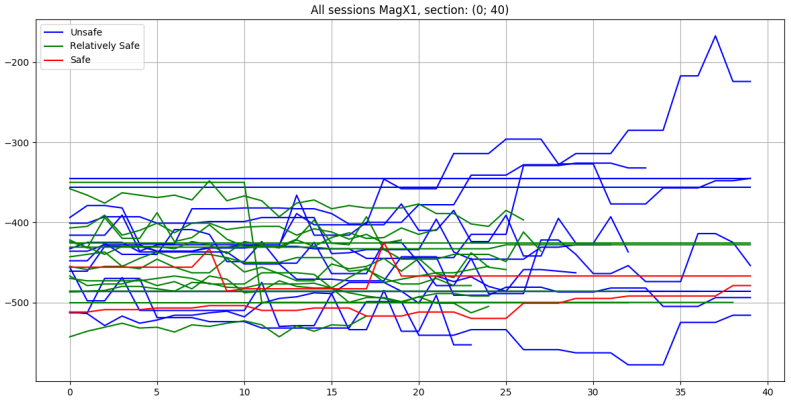


FIGURE C.43: MagX1, all sessions comparison

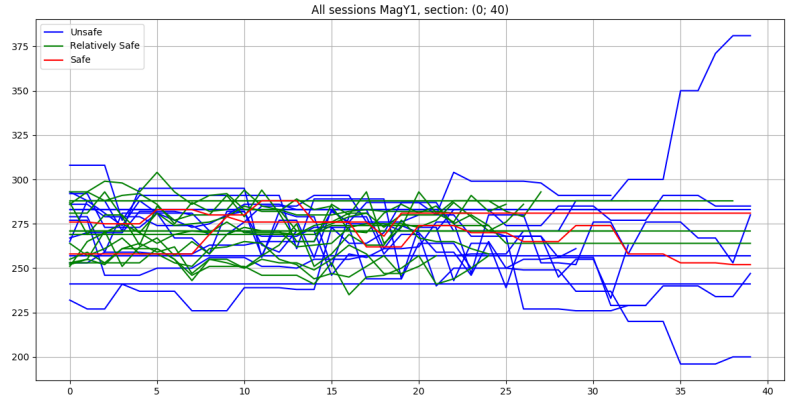


FIGURE C.44: MagY1, all sessions comparison

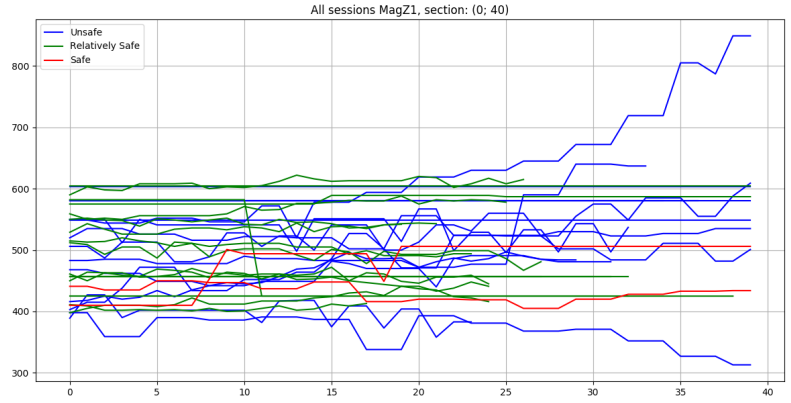


FIGURE C.45: MagZ1, all sessions comparison

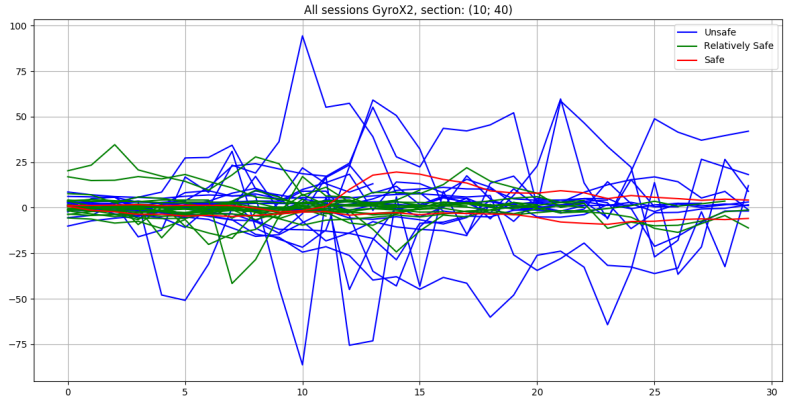


FIGURE C.46: GyroX2, all sessions comparison

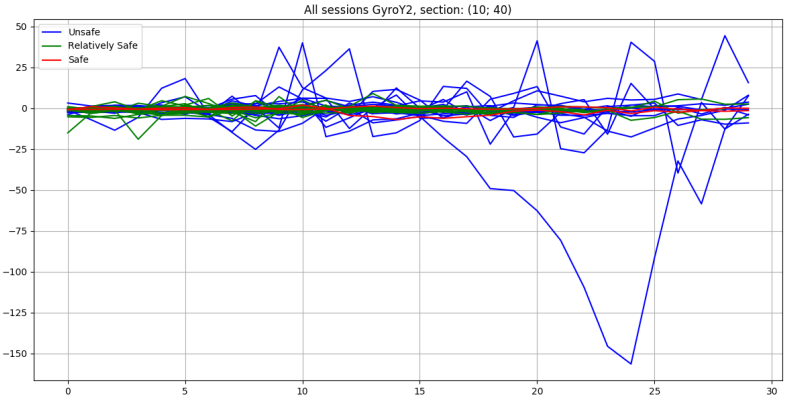


FIGURE C.47: GyroY2, all sessions comparison

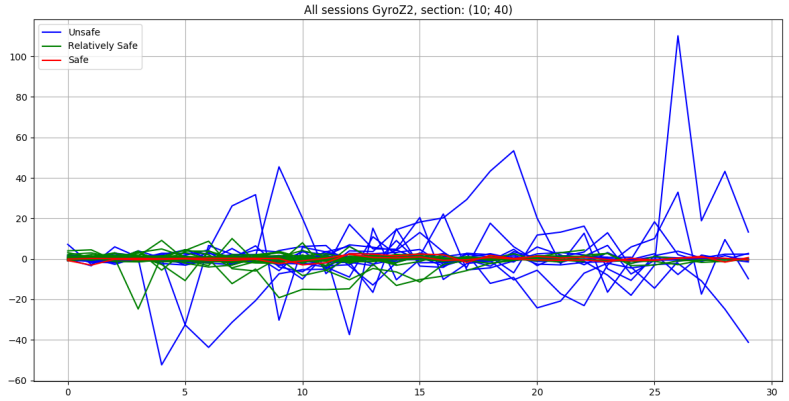


FIGURE C.48: GyroZ2, all sessions comparison

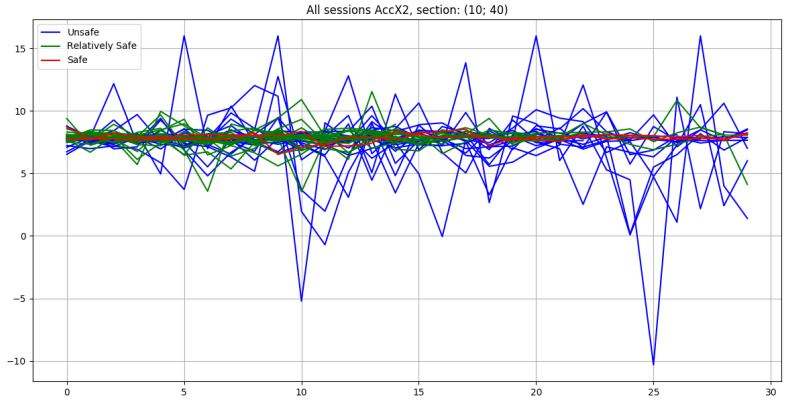


FIGURE C.49: AccX2, all sessions comparison

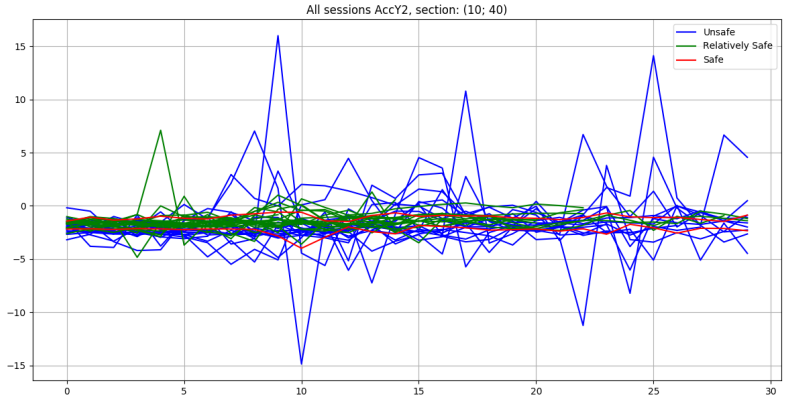


FIGURE C.50: AccY2, all sessions comparison

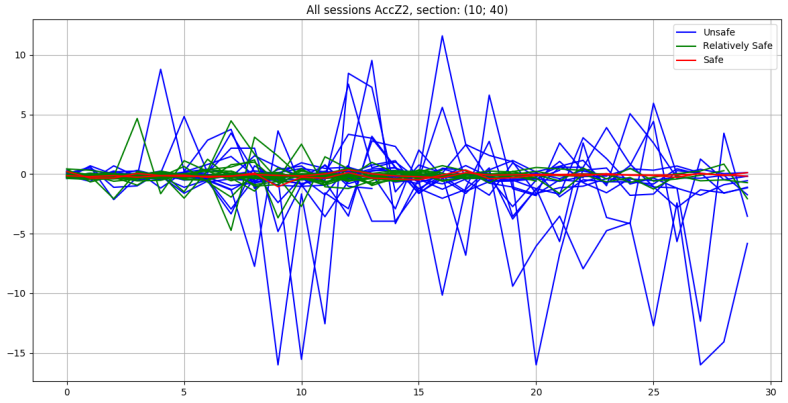


FIGURE C.51: AccZ2, all sessions comparison

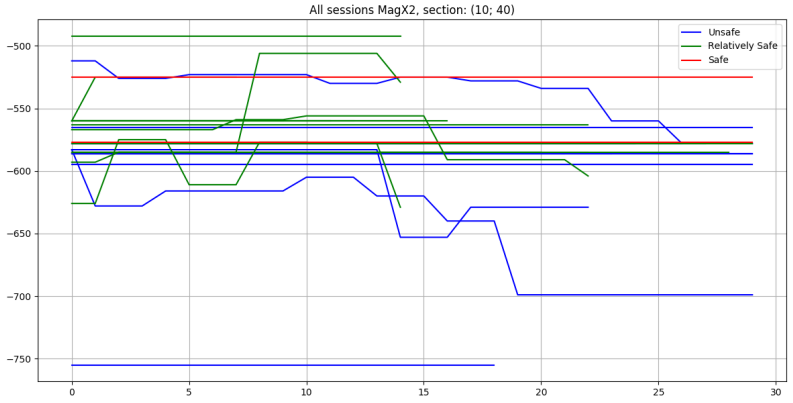


FIGURE C.52: MagX2, all sessions comparison

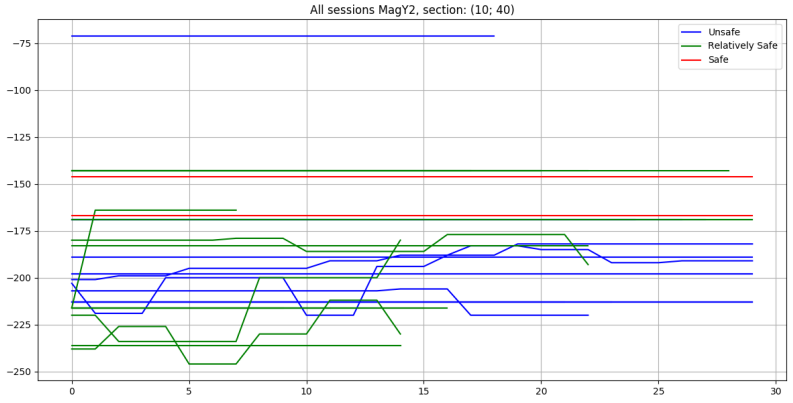


FIGURE C.53: MagY2, all sessions comparison

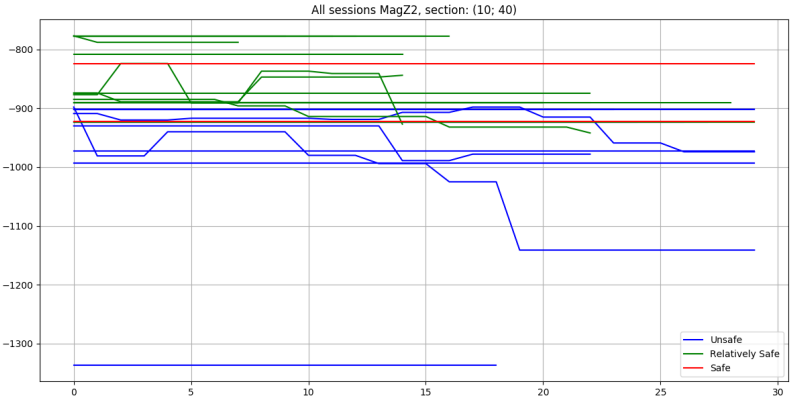


FIGURE C.54: MagZ2, all sessions comparison

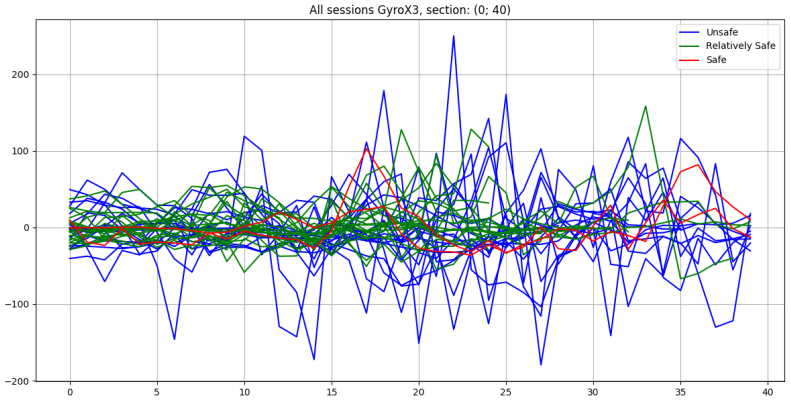


FIGURE C.55: GyroX3, all sessions comparison

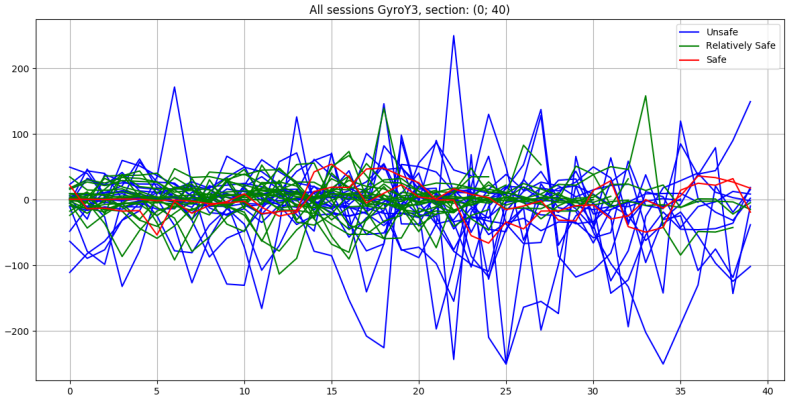


FIGURE C.56: GyroY3, all sessions comparison

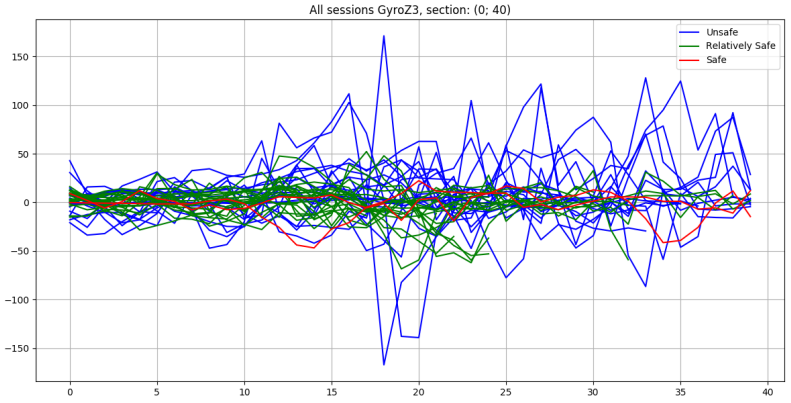


FIGURE C.57: GyroZ3, all sessions comparison

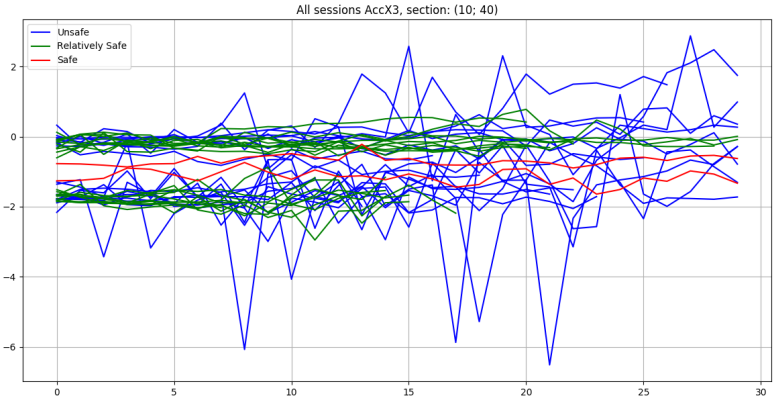


FIGURE C.58: AccX3, all sessions comparison

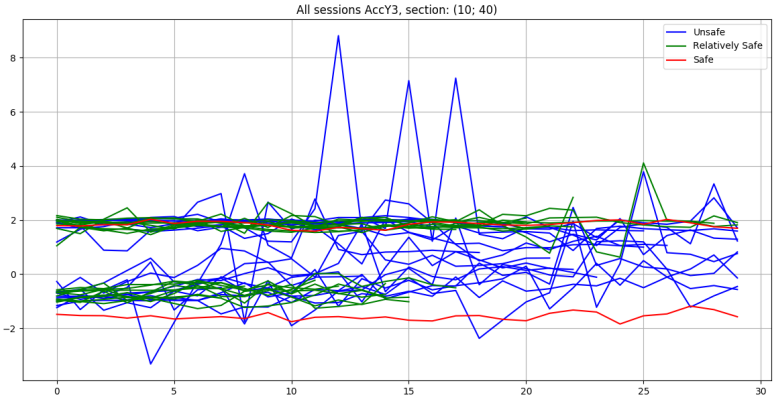


FIGURE C.59: AccY3, all sessions comparison

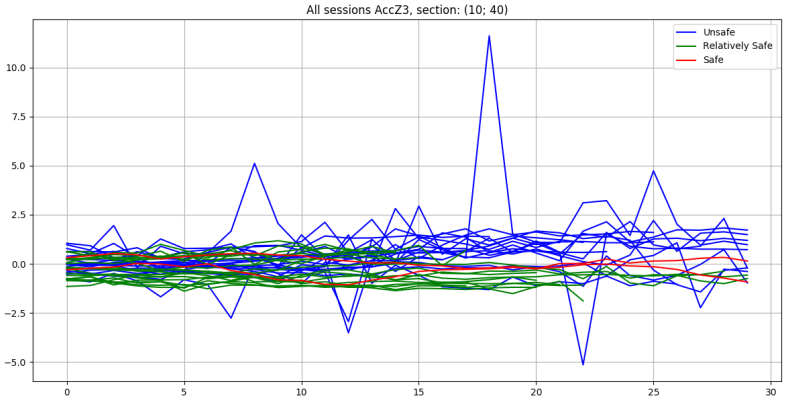


FIGURE C.60: AccZ3, all sessions comparison

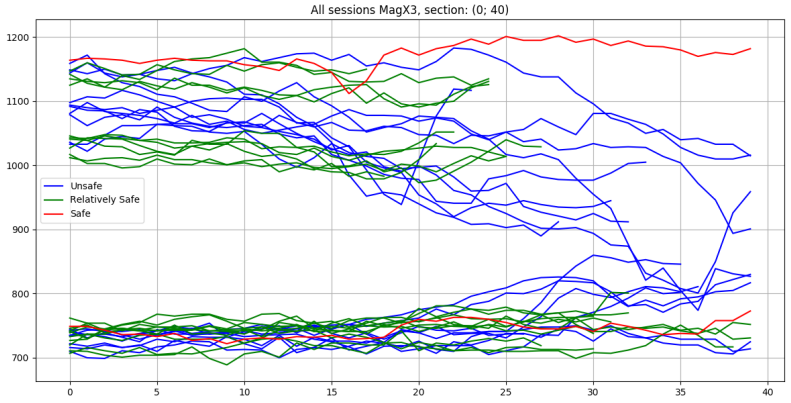


FIGURE C.61: MagX3, all sessions comparison

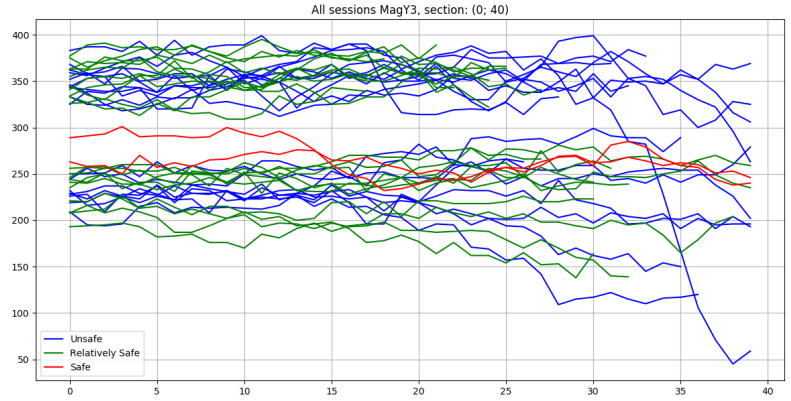


FIGURE C.62: MagY3, all sessions comparison

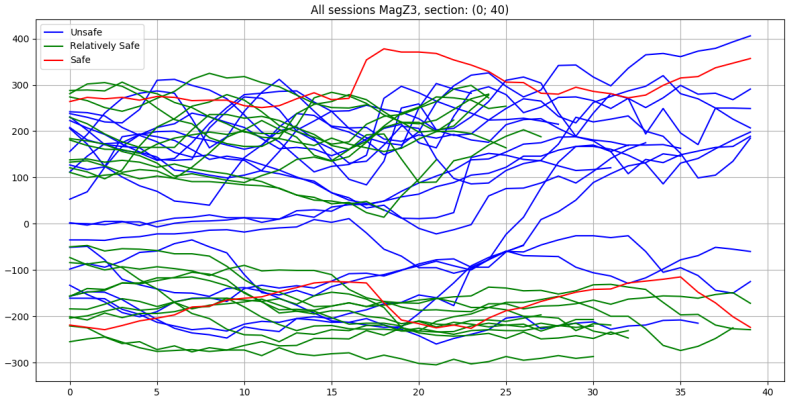


FIGURE C.63: MagZ3, all sessions comparison

Prediction plots by 6 samples using neural network

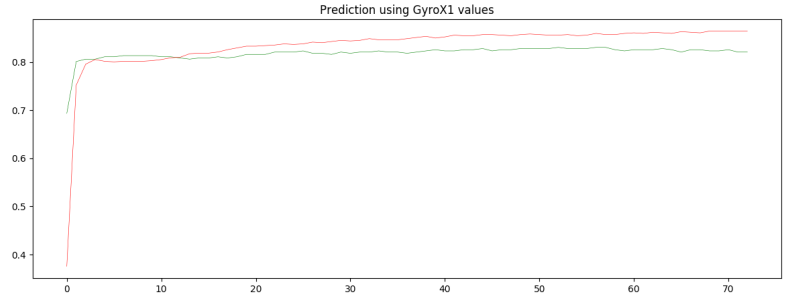


FIGURE C.64: Prediction using GyroX1 values, result: 80.41% (5.76%)

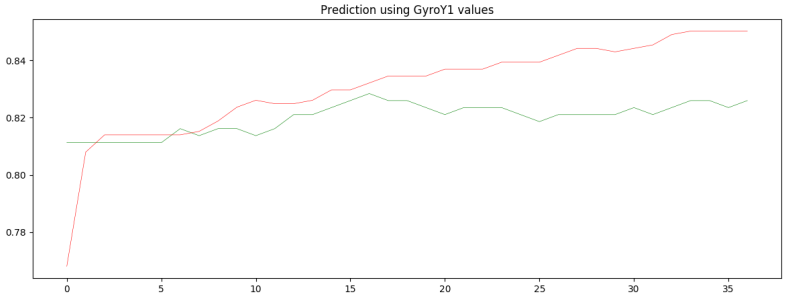


FIGURE C.65: Prediction using GyroY1 values, result: 80.41% (4.96%)

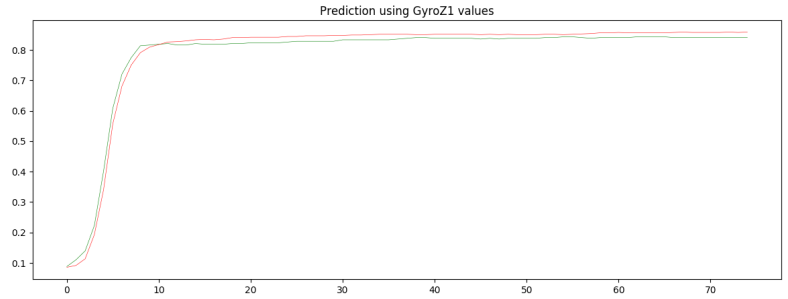


FIGURE C.66: Prediction using GyroZ1 values, result: 80.64% (5.04%)

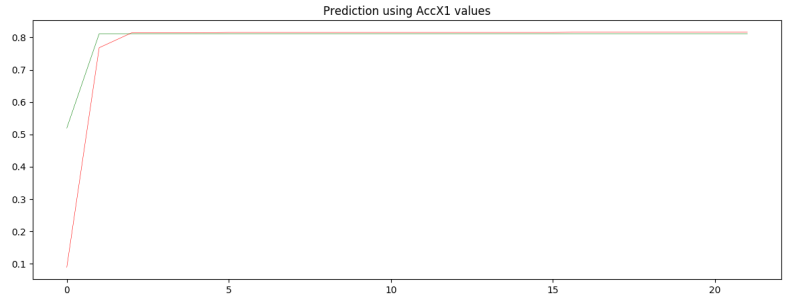


FIGURE C.67: Prediction using AccX1 values, result: 83.84% (5.35%)

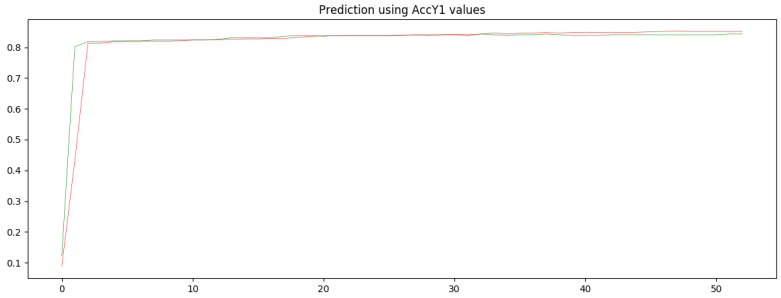


FIGURE C.68: Prediction using AccY1 values, result: 82.87% (6.72%)

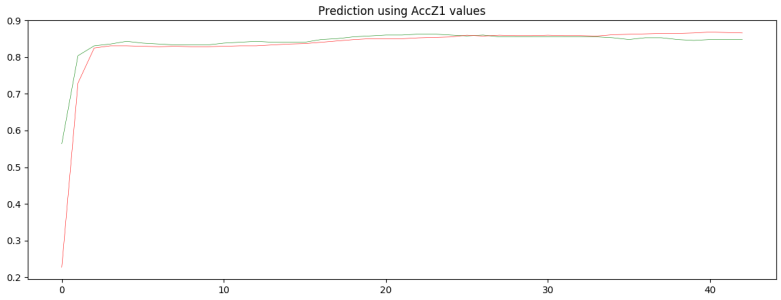


FIGURE C.69: Prediction using AccZ1 values, result: 83.35% (4.93%)

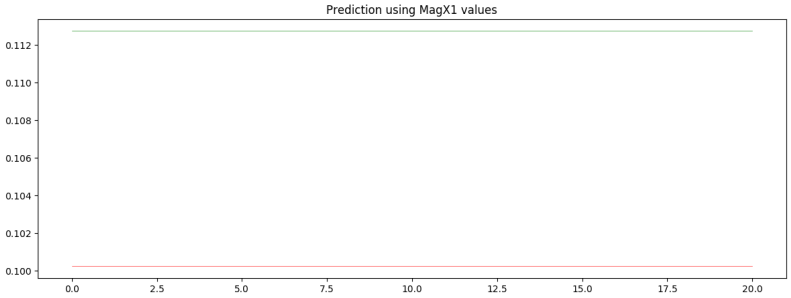


FIGURE C.70: Prediction using MagX1 values, result: 36.84% (33.52%)

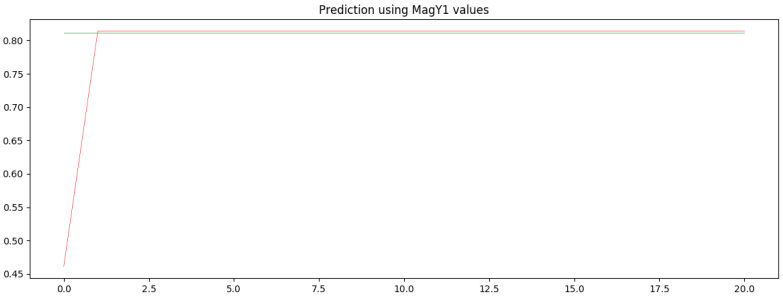


FIGURE C.71: Prediction using MagY1 values, result: 31.24% (34.59%)

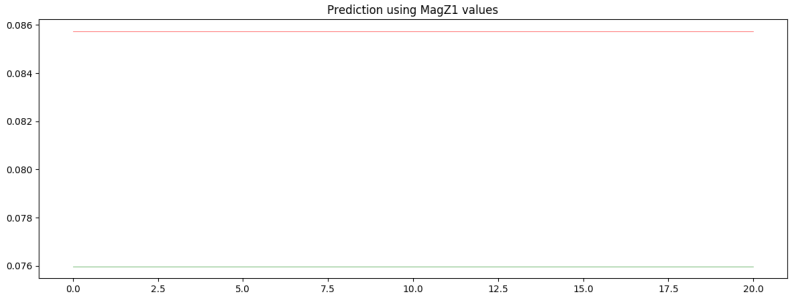


FIGURE C.72: Prediction using MagZ1 values, result: 39.32% (35.04%)

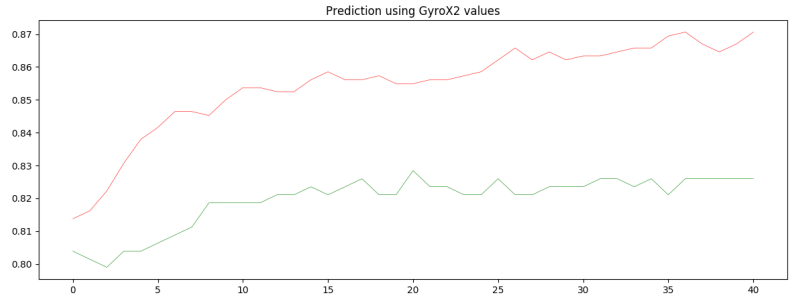


FIGURE C.73: Prediction using GyroX2 values, result: 82.35% (4.90%)

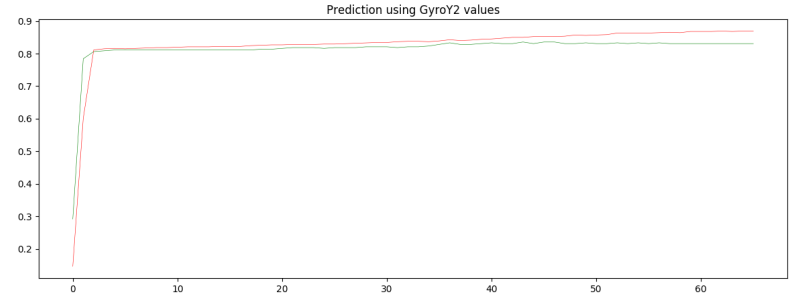


FIGURE C.74: Prediction using GyroY2 values, result: 82.86% (5.44%)

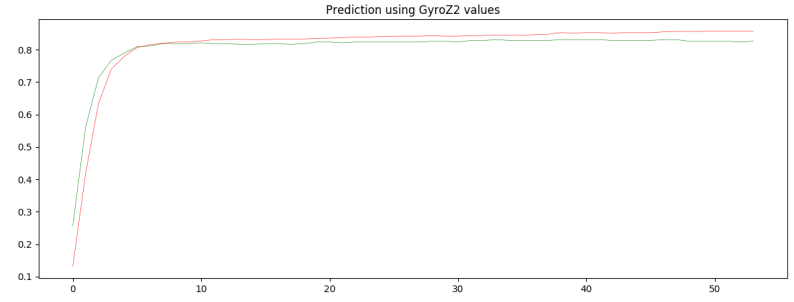


FIGURE C.75: Prediction using GyroZ2 values, result: 79.64% (3.39%)

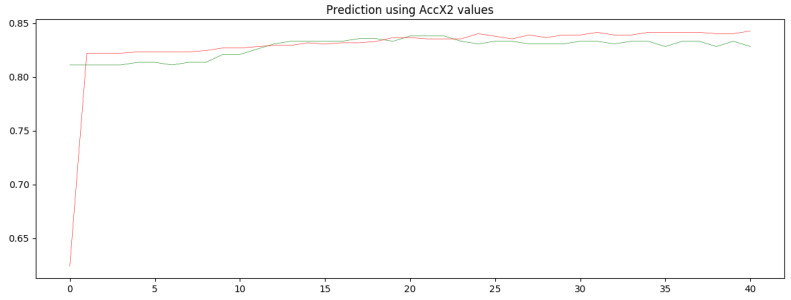


FIGURE C.76: Prediction using AccX2 values, result: 83.59% (4.74%)

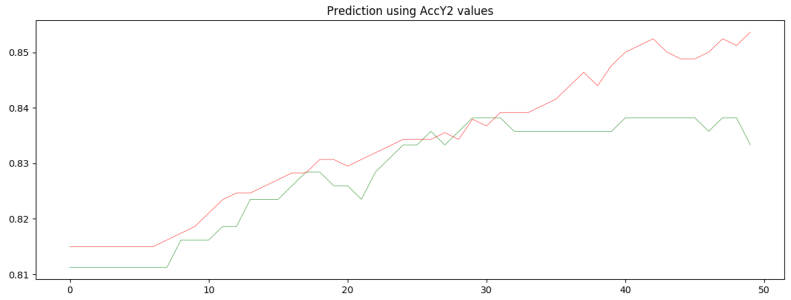


FIGURE C.77: Prediction using AccY2 values, result: 83.34% (4.06%)

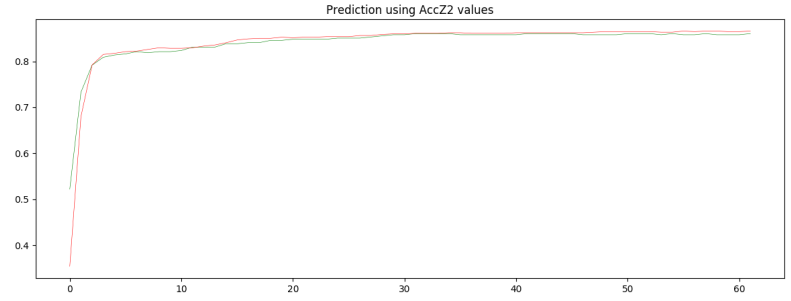


FIGURE C.78: Prediction using AccZ2 values, result: 86.51% (5.86%)

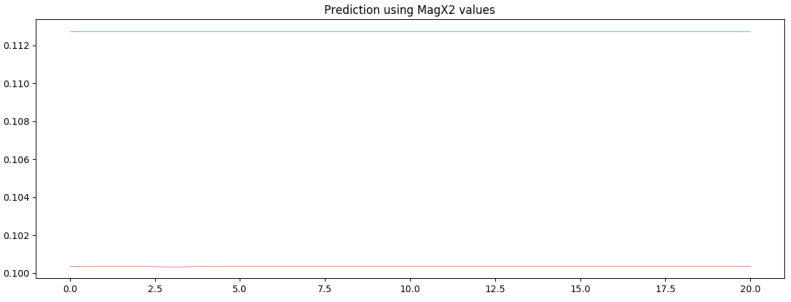


FIGURE C.79: Prediction using MagX2 values, result: 20.52% (27.13%)

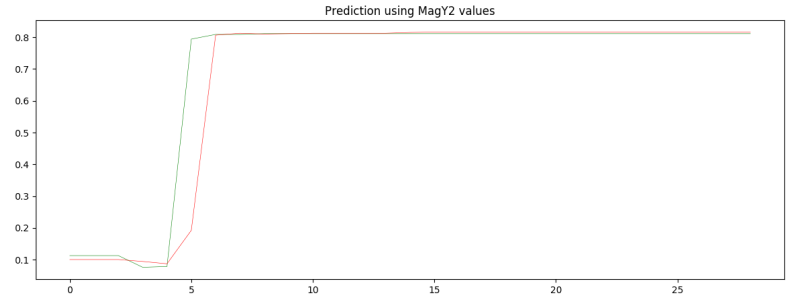


FIGURE C.80: Prediction using MagY2 values, result: 53.12% (36.79%)

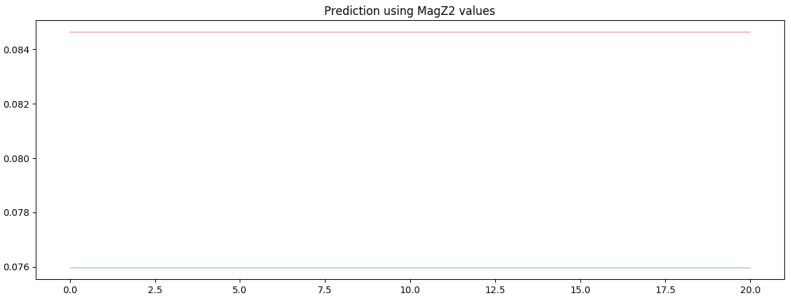


FIGURE C.81: Prediction using MagZ2 values, result: 40.02% (36.45%)

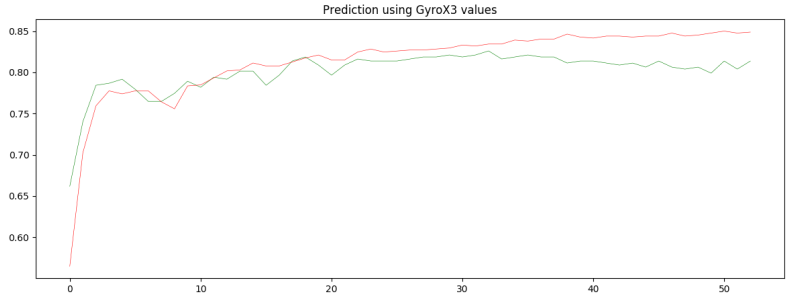


FIGURE C.82: Prediction using GyroX3 values, result: 77.68% (7.85%)

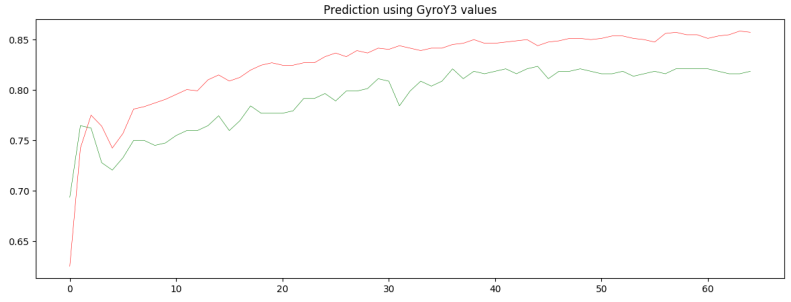


FIGURE C.83: Prediction using GyroY3 values, result: 77.68% (5.12%)

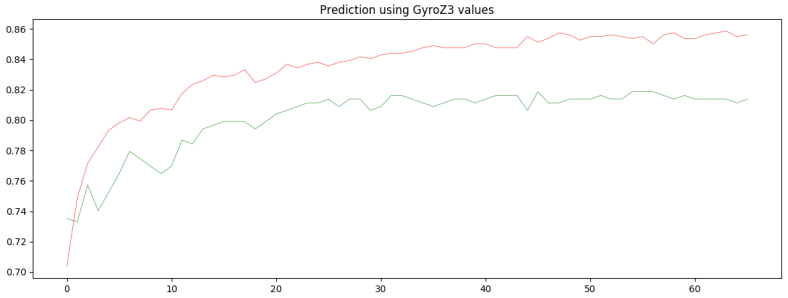


FIGURE C.84: Prediction using GyroZ3 values, result: 78.91% (6.80%)

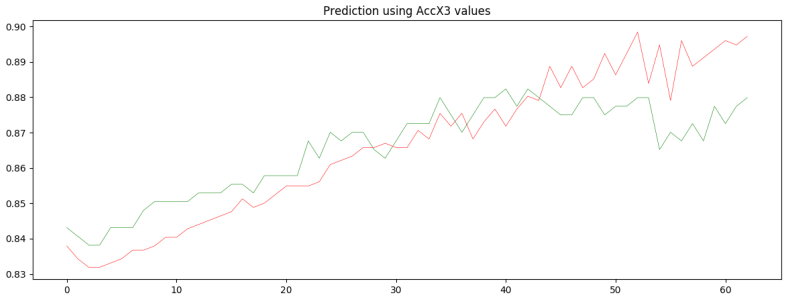


FIGURE C.85: Prediction using AccX3 values, result: 87.74% (6.55%)

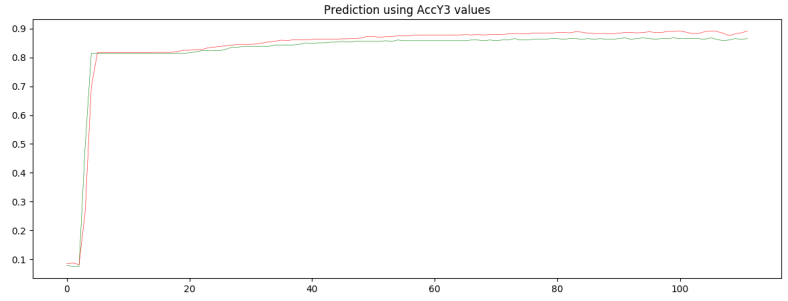


FIGURE C.86: Prediction using AccY3 values, result: 84.06% (5.19%)

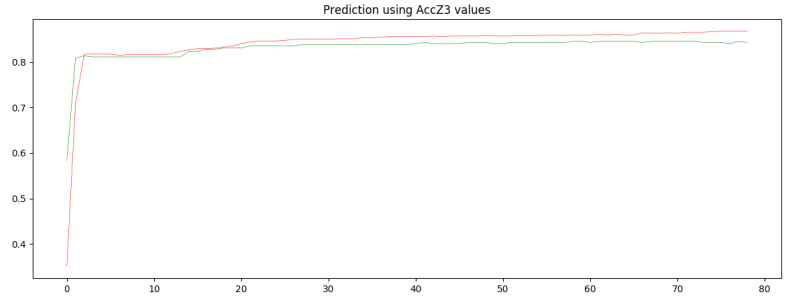


FIGURE C.87: Prediction using AccZ3 values, result: 84.30% (4.81%)

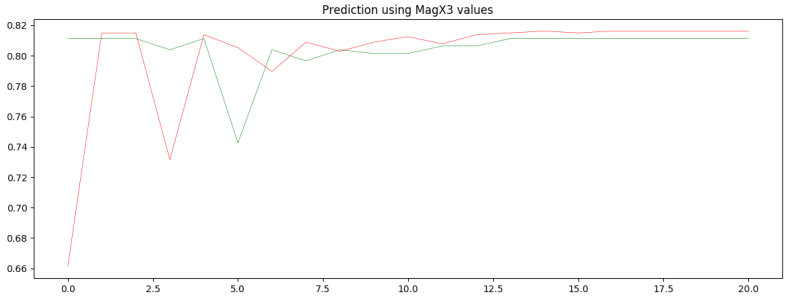


FIGURE C.88: Prediction using MagX3 values, result: 29.55% (34.17%)

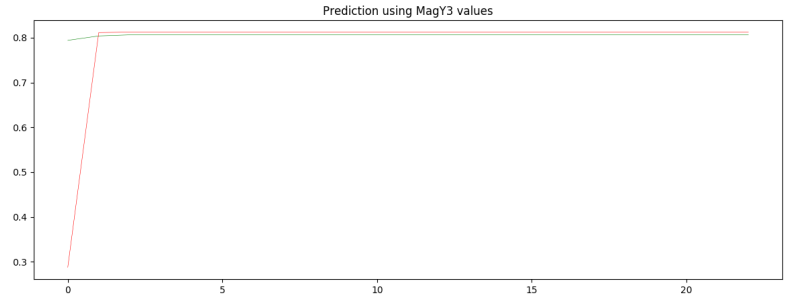


FIGURE C.89: Prediction using MagY3 values, result: 40.69% (34.67%)

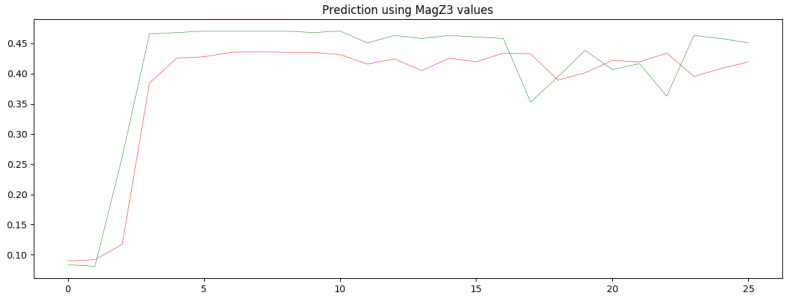


FIGURE C.90: Prediction using MagZ3 values, result: 75.63% (5.77%))

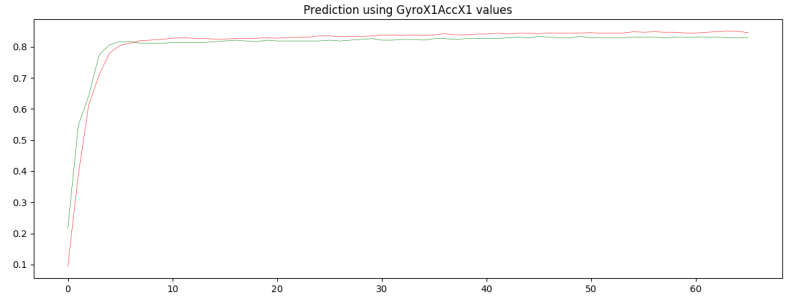


FIGURE C.91: Prediction using GyroX1 and AccX1 values, result: 80.64% (5.39%)

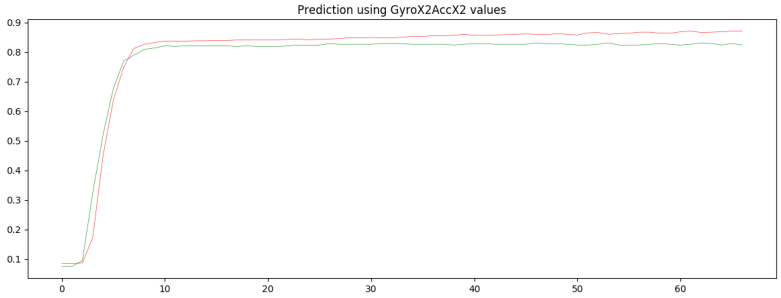


FIGURE C.92: Prediction using GyroX2 and AccX2 values, result: 83.80% (5.37%)

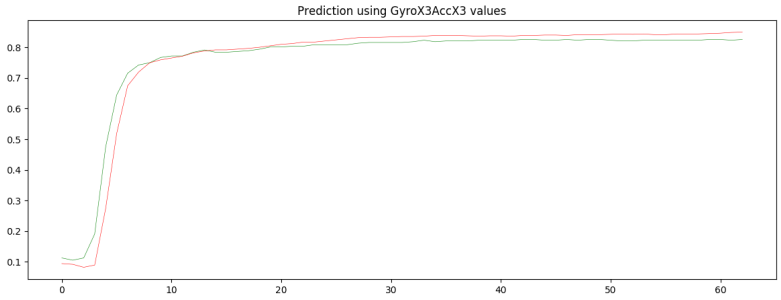


FIGURE C.93: Prediction using GyroX3 and AccX3 values, result: 85.12% (6.28%)

Appendix D

Analysis of data collected from in-built iPhone and Apple Watch motion sensors

Console output

```
/Library/Frameworks/Python.framework/Versions/3.6/bin/python3 /Users/delexa/
    Desktop/ModelGenerator/analyze.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 73573 entries, 0 to 73572
Data columns (total 24 columns):
SessionID                    73573 non-null int64
SessionDate                  73573 non-null object
SessionDuration              73573 non-null object
SessionFrequency             73573 non-null int64
RecordID                     73573 non-null int64
Timestamp                    67070 non-null object
timeIntervalSince1970        67070 non-null float64
GyroX                        67070 non-null float64
GyroY                        67070 non-null float64
GyroZ                        67070 non-null float64
AccX                         67070 non-null float64
AccY                         67070 non-null float64
AccZ                         67070 non-null float64
MagX                         67070 non-null float64
MagY                         67070 non-null float64
MagZ                         67070 non-null float64
WatchTimestamp               73573 non-null object
WatchtImeIntervalSince1970   73573 non-null float64
WatchGyroX                   73573 non-null float64
```

```
WatchGyroY                    73573 non-null float64
WatchGyroZ                    73573 non-null float64
WatchAccX                     73573 non-null float64
WatchAccY                     73573 non-null float64
WatchAccZ                     73573 non-null float64
dtypes: float64(17), int64(3), object(4)
memory usage: 13.5+ MB
The dataset contains 73573 data samples and 24 data columns
SessionID                         0
SessionDate                       0
SessionDuration                   0
SessionFrequency                  0
RecordID                          0
Timestamp                      6503
timeIntervalSince1970          6503
GyroX                          6503
GyroY                          6503
GyroZ                          6503
AccX                           6503
AccY                           6503
AccZ                           6503
MagX                           6503
MagY                           6503
MagZ                           6503
WatchTimestamp                    0
WatchtImeIntervalSince1970        0
WatchGyroX                        0
WatchGyroY                        0
WatchGyroZ                        0
WatchAccX                         0
WatchAccY                         0
WatchAccZ                         0
dtype: int64
          SessionID  SessionFrequency      RecordID  timeIntervalSince1970  \
count  73573.000000           73573.0  73573.000000           6.707000e+04
mean       3.650891              60.0      0.555503           1.524751e+09
std        6.827412               0.0      0.795353           9.955023e+02
min        0.000000              60.0      0.000000           1.524750e+09
25%        0.000000              60.0      0.000000           1.524750e+09
50%        0.000000              60.0      0.000000           1.524751e+09
75%        2.000000              60.0      1.000000           1.524752e+09
max       28.000000              60.0      2.000000           1.524753e+09

              GyroX         GyroY         GyroZ          AccX          AccY  \
count  67070.000000  67070.000000  67070.000000  67070.000000  67070.000000
mean      -0.007527     -0.014304      0.006841      0.156397     -0.053040
std        0.396976      0.522330      0.336815      0.262515      0.920346
min       -9.733911     -9.350921     -3.720490     -7.007690     -8.075256
```

```
25%      -0.197357    -0.255608    -0.128766    -0.012161    -0.948441
50%      -0.009513    -0.007902    -0.016014     0.139290    -0.093346
75%       0.147421     0.182670     0.081850     0.253723     0.973648
max       5.313863    22.365086     6.960622     7.504089     8.004822

              AccZ         MagX         MagY         MagZ  \
count  67070.000000  67070.000000  67070.000000  67070.000000
mean      -0.224298   -821.137326   2351.779068   2544.714715
std        0.261519    623.156775     27.002165     7.640330
min       -5.975113  -1918.711670   2304.219727   2537.750488
25%       -0.303616  -1795.484619   2309.559326   2539.945068
50%       -0.218887   -458.846252   2367.477539   2540.272705
75%       -0.063572   -432.115746   2368.635864   2556.660889
max        4.069229   -253.117111   2376.392090   2558.171631

       WatchtImeIntervalSince1970     WatchGyroX     WatchGyroY     WatchGyroZ  \
count                7.357300e+04  73573.000000  73573.000000  73573.000000
mean                 1.524751e+09      0.000180     -0.023876     -0.004918
std                  9.938303e+02      0.632220      0.337101      0.408254
min                  1.524750e+09    -21.260214     -9.755253     -6.625195
25%                  1.524750e+09     -0.137497     -0.125230     -0.135379
50%                  1.524751e+09     -0.014599     -0.010576      0.001319
75%                  1.524752e+09      0.094476      0.087081      0.155269
max                  1.524753e+09     21.818573      6.052173      8.516460

          WatchAccX     WatchAccY     WatchAccZ
count  73573.000000  73573.000000  73573.000000
mean       0.369578      0.322029     -0.728343
std        0.273462      0.393525      0.279427
min       -4.127808    -15.992234    -14.162827
25%        0.284286      0.168198     -0.893875
50%        0.411102      0.361084     -0.823074
75%        0.550217      0.604874     -0.542801
max       13.100601     10.012497      6.332337

Dataset contains 46890 "safe" data samples as well as 12496 "relatevely safe"
    data samples and 12496 "unsafe" data samples

   SessionID          SessionDate SessionDuration  SessionFrequency  \
0          0  2018-04-26 16:00:32.4520            05:07                60
1          0  2018-04-26 16:00:32.4520            05:07                60
2          0  2018-04-26 16:00:32.4520            05:07                60
3          0  2018-04-26 16:00:32.4520            05:07                60
4          0  2018-04-26 16:00:32.4520            05:07                60
5          0  2018-04-26 16:00:32.4520            05:07                60
6          0  2018-04-26 16:00:32.4520            05:07                60
7          0  2018-04-26 16:00:32.4520            05:07                60
8          0  2018-04-26 16:00:32.4520            05:07                60
```

```
119 9            0   2018-04-26  16:00:32.4520            05:07                60
    10           0   2018-04-26  16:00:32.4520            05:07                60
121 11           0   2018-04-26  16:00:32.4520            05:07                60
    12           0   2018-04-26  16:00:32.4520            05:07                60
123 13           0   2018-04-26  16:00:32.4520            05:07                60
    14           0   2018-04-26  16:00:32.4520            05:07                60
125 15           0   2018-04-26  16:00:32.4520            05:07                60
    16           0   2018-04-26  16:00:32.4520            05:07                60
127 17           0   2018-04-26  16:00:32.4520            05:07                60

129      RecordID                  Timestamp  timeIntervalSince1970     GyroX  \
    0           0   2018-04-26  16:00:32.5750           1.524751e+09  0.114449
131 1           0   2018-04-26  16:00:32.5920           1.524751e+09  0.068869
    2           0   2018-04-26  16:00:32.6090           1.524751e+09  0.007367
133 3           0   2018-04-26  16:00:32.6260           1.524751e+09 -0.048851
    4           0   2018-04-26  16:00:32.6420           1.524751e+09 -0.113524
135 5           0   2018-04-26  16:00:32.6610           1.524751e+09 -0.163319
    6           0   2018-04-26  16:00:32.6760           1.524751e+09 -0.188750
137 7           0   2018-04-26  16:00:32.6940           1.524751e+09 -0.219468
    8           0   2018-04-26  16:00:32.7100           1.524751e+09 -0.236479
139 9           0   2018-04-26  16:00:32.7270           1.524751e+09 -0.227973
    10          0   2018-04-26  16:00:32.7440           1.524751e+09 -0.212068
141 11          0   2018-04-26  16:00:32.7610           1.524751e+09 -0.200369
    12          0   2018-04-26  16:00:32.7770           1.524751e+09 -0.183296
143 13          0   2018-04-26  16:00:32.7940           1.524751e+09 -0.160993
    14          0   2018-04-26  16:00:32.8110           1.524751e+09 -0.169483
145 15          0   2018-04-26  16:00:32.8270           1.524751e+09 -0.169528
    16          0   2018-04-26  16:00:32.8440           1.524751e+09 -0.189672
147 17          0   2018-04-26  16:00:32.8610           1.524751e+09 -0.200226

149       GyroY     GyroZ  ...          MagY         MagZ  \
    0  0.420634  0.014215  ...   2368.050537  2540.110596
151 1  0.390462 -0.003109  ...   2368.050537  2540.110596
    2  0.312997 -0.018244  ...   2368.050537  2540.110596
153 3  0.257071 -0.030221  ...   2368.047363  2540.111572
    4  0.201090 -0.050801  ...   2368.044189  2540.112549
155 5  0.154738 -0.073429  ...   2368.050537  2540.110596
    6  0.120278 -0.083103  ...   2368.034424  2540.115234
157 7  0.087935 -0.098162  ...   2368.034424  2540.115234
    8  0.084778 -0.095090  ...   2368.031006  2540.116211
159 9  0.100871 -0.098284  ...   2368.034424  2540.115234
    10 0.140669 -0.096176  ...   2368.031006  2540.116211
161 11 0.178249 -0.102628  ...   2368.034424  2540.115234
    12 0.186672 -0.118521  ...   2368.021240  2540.118896
163 13 0.209206 -0.119492  ...   2368.014893  2540.120850
    14 0.212437 -0.121709  ...   2368.014893  2540.120850
165 15 0.218970 -0.113201  ...   2368.008301  2540.122559
    16 0.222173 -0.121914  ...   2368.014893  2540.120850
```

```
17   0.179066  -0.131431     ...        2368.001709   2540.124512

                 WatchTimestamp   WatchtImeIntervalSince1970   WatchGyroX  \
0   2018-04-26 16:00:32.3630                  1.524751e+09    -0.251138
1   2018-04-26 16:00:32.3780                  1.524751e+09    -0.042320
2   2018-04-26 16:00:32.3930                  1.524751e+09     0.009363
3   2018-04-26 16:00:32.4090                  1.524751e+09    -0.216995
4   2018-04-26 16:00:32.4240                  1.524751e+09    -0.545621
5   2018-04-26 16:00:32.4390                  1.524751e+09    -0.898000
6   2018-04-26 16:00:32.4540                  1.524751e+09    -1.178968
7   2018-04-26 16:00:32.4690                  1.524751e+09    -1.045339
8   2018-04-26 16:00:32.4850                  1.524751e+09    -0.592919
9   2018-04-26 16:00:32.5000                  1.524751e+09    -0.483998
10  2018-04-26 16:00:32.5150                  1.524751e+09    -0.687511
11  2018-04-26 16:00:32.5300                  1.524751e+09    -0.566808
12  2018-04-26 16:00:32.5450                  1.524751e+09    -0.077885
13  2018-04-26 16:00:32.5600                  1.524751e+09     0.341297
14  2018-04-26 16:00:32.5760                  1.524751e+09     0.380301
15  2018-04-26 16:00:32.5950                  1.524751e+09     0.215465
16  2018-04-26 16:00:32.6050                  1.524751e+09     0.039867
17  2018-04-26 16:00:32.6210                  1.524751e+09     0.157158

     WatchGyroY  WatchGyroZ  WatchAccX  WatchAccY  WatchAccZ
0     0.118929    0.384660   0.415756   0.319626  -0.778305
1     0.152088    0.446979   0.424835   0.382248  -0.835861
2     0.114961    0.379977   0.425674   0.434128  -0.906601
3     0.022031    0.329766   0.402939   0.440201  -0.988129
4    -0.064078    0.293156   0.391174   0.424164  -0.989151
5    -0.110774    0.348198   0.340195   0.290054  -0.922485
6    -0.107791    0.356047   0.338486   0.280914  -0.895691
7    -0.092344    0.329540   0.369049   0.388519  -0.794418
8    -0.033741    0.320642   0.375153   0.505051  -0.799042
9    -0.022736    0.268441   0.402039   0.464737  -0.846954
10   -0.035012    0.269091   0.407852   0.388168  -0.838440
11   -0.032540    0.223395   0.396469   0.476639  -0.808060
12   -0.010667    0.187023   0.392410   0.525406  -0.813080
13    0.061721    0.153365   0.408936   0.543961  -0.792542
14    0.087955    0.107310   0.405014   0.502380  -0.823151
15    0.058433    0.106029   0.389954   0.382782  -0.858536
16    0.035453    0.115367   0.396179   0.388687  -0.840485
17    0.020057    0.105061   0.407516   0.466400  -0.796265

[18 rows x 24 columns]
```

Listing D.1: Console output

Sensor data distribution plots

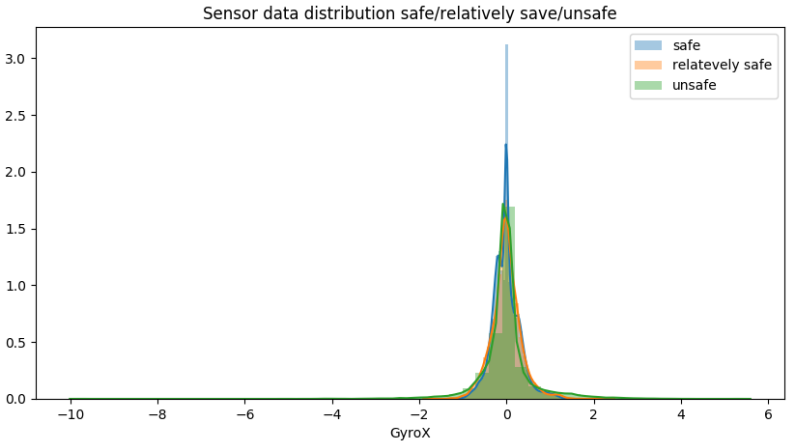


FIGURE D.1: Sensor data distribution GyroX

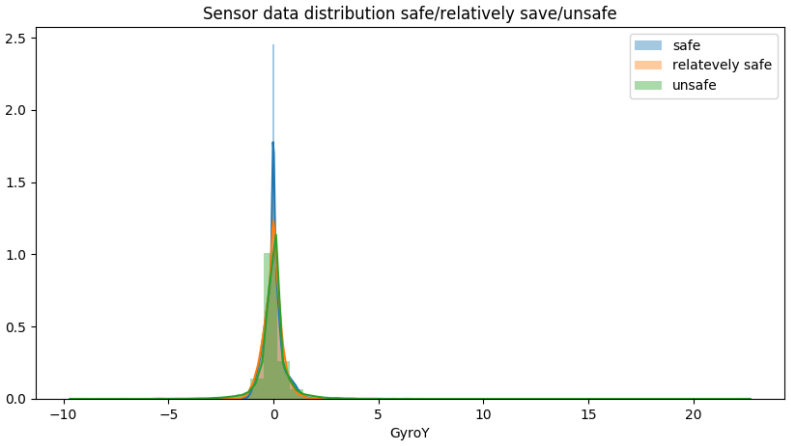


FIGURE D.2: Sensor data distribution GyroY

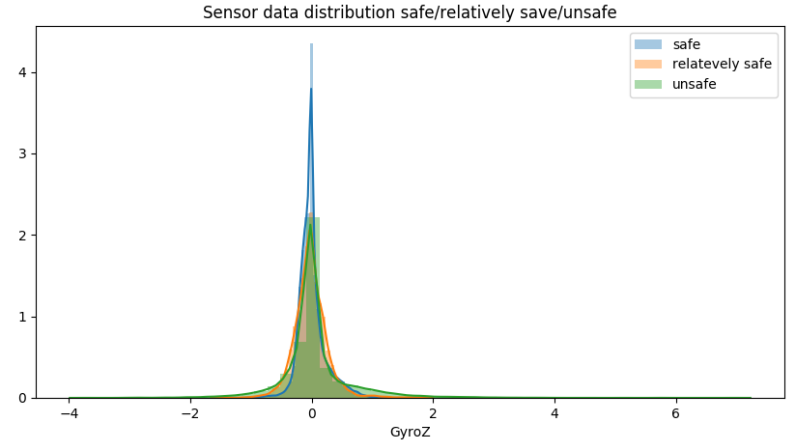


FIGURE D.3: Sensor data distribution GyroZ

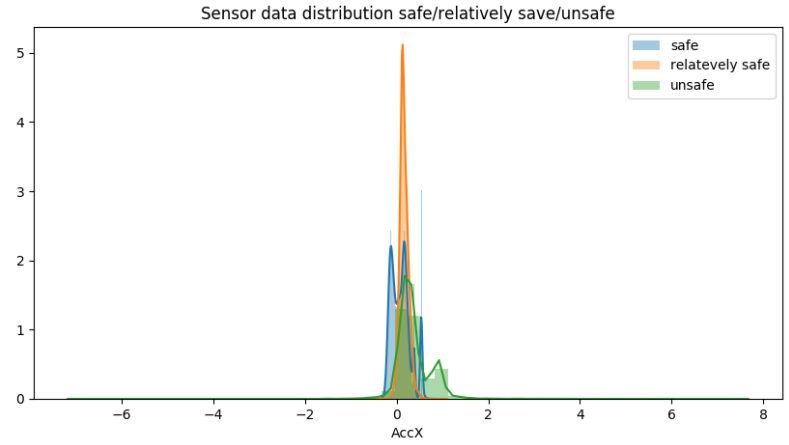


FIGURE D.4: Sensor data distribution AccX

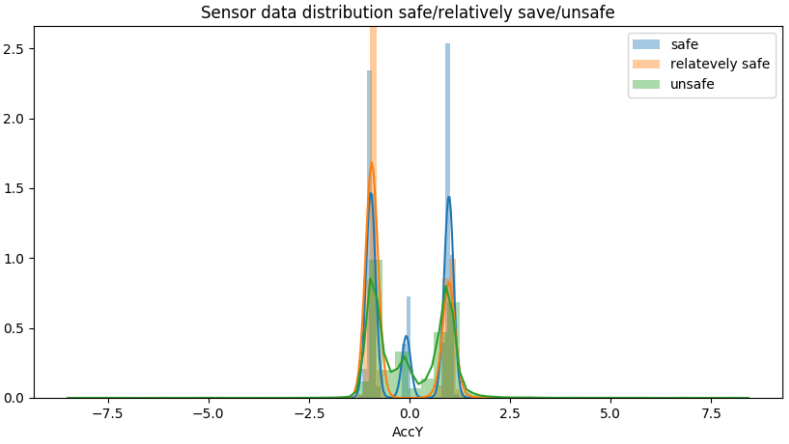


FIGURE D.5: Sensor data distribution AccY

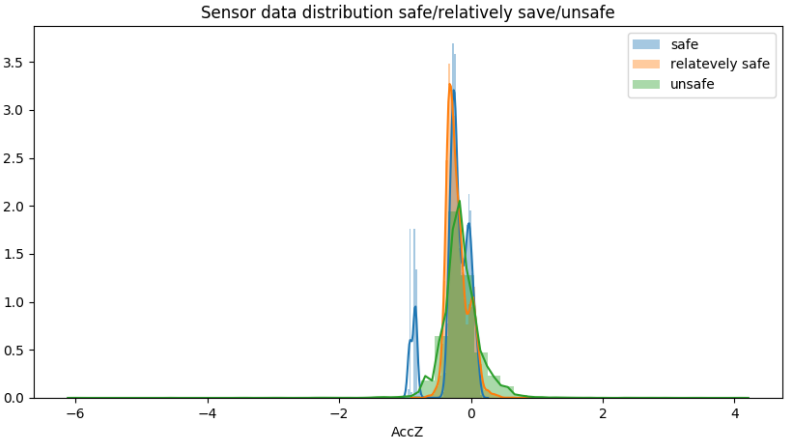


FIGURE D.6: Sensor data distribution AccZ

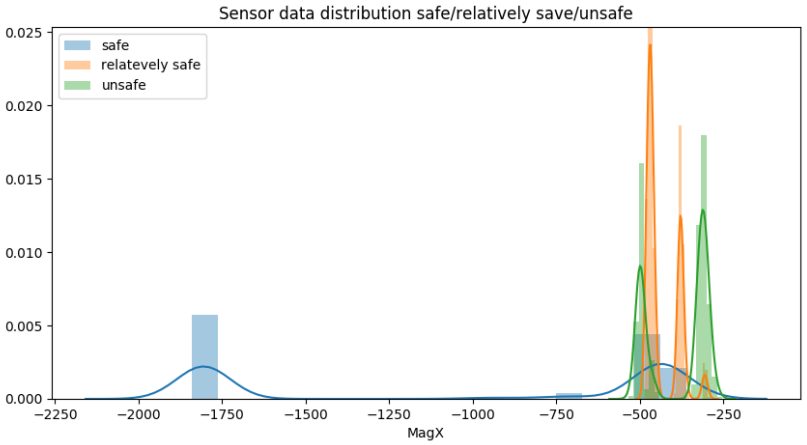


FIGURE D.7: Sensor data distribution MagX

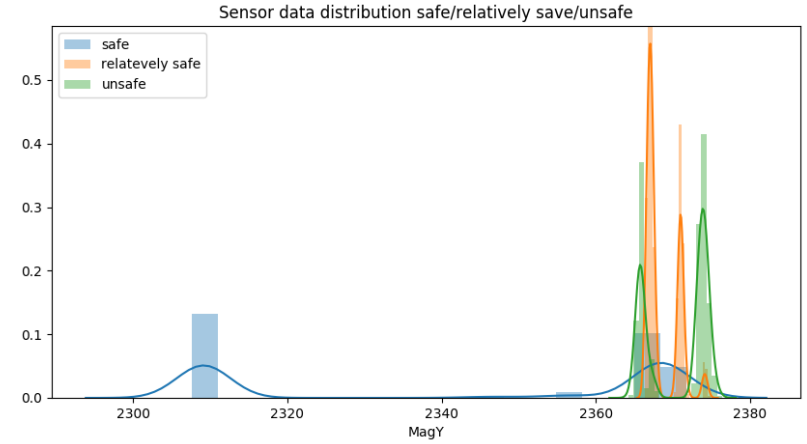


FIGURE D.8: Sensor data distribution MagY

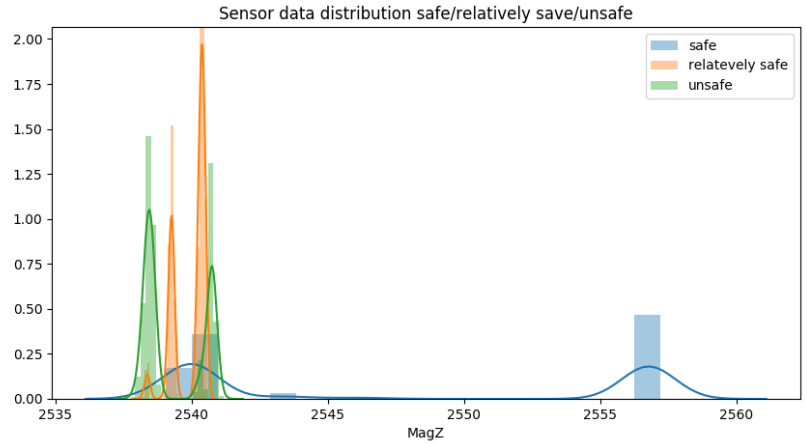


FIGURE D.9: Sensor data distribution MagZ

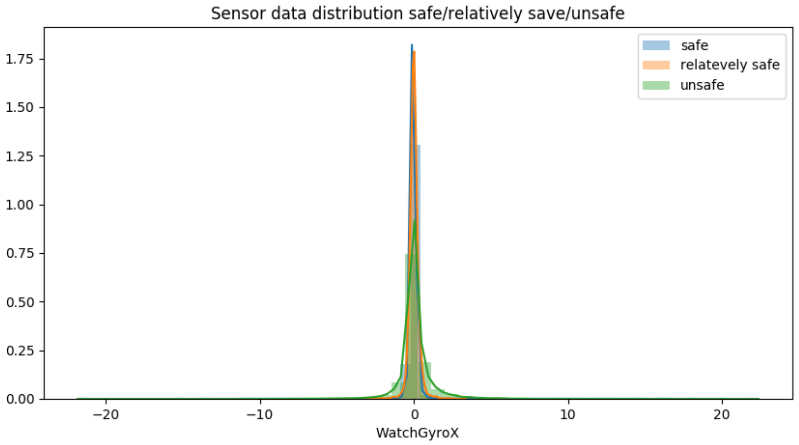


FIGURE D.10: Sensor data distribution WatchGyroX

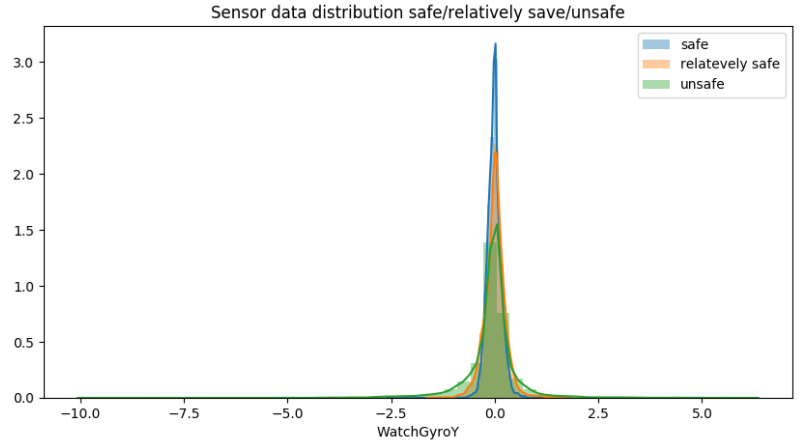


FIGURE D.11: Sensor data distribution WatchGyroY

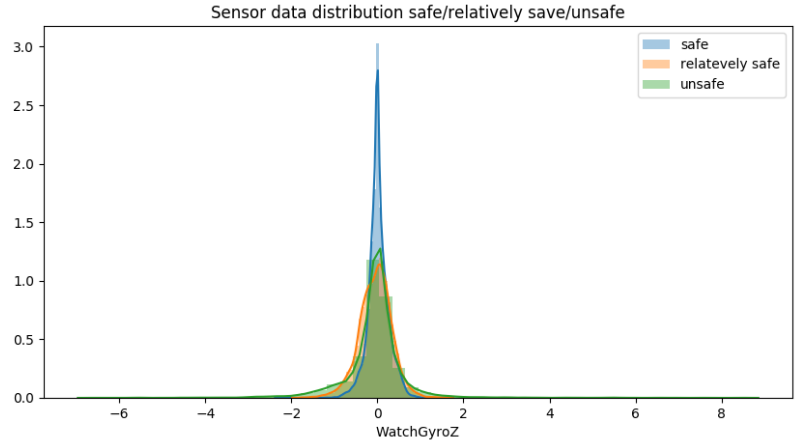


FIGURE D.12: Sensor data distribution WatchGyroZ

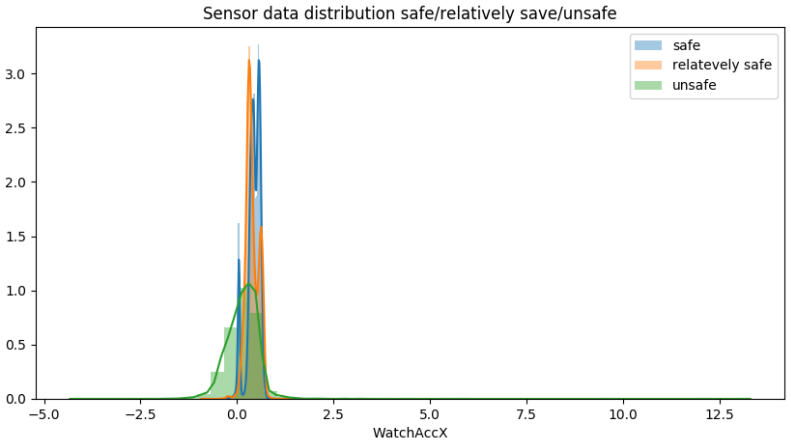


FIGURE D.13: Sensor data distribution WatchAccX

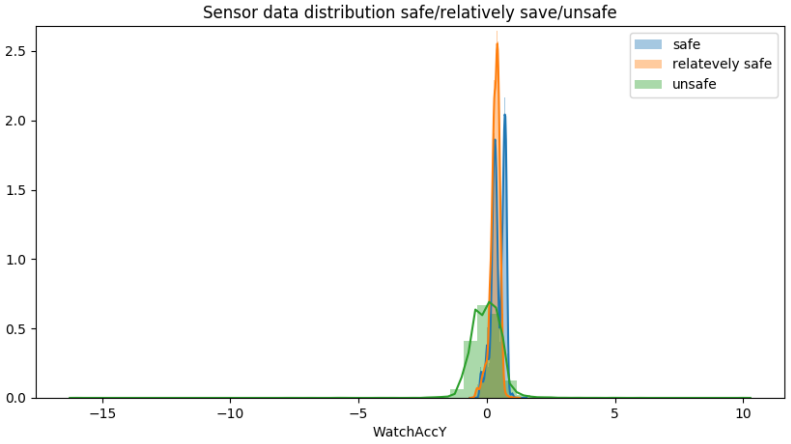


FIGURE D.14: Sensor data distribution WatchAccY

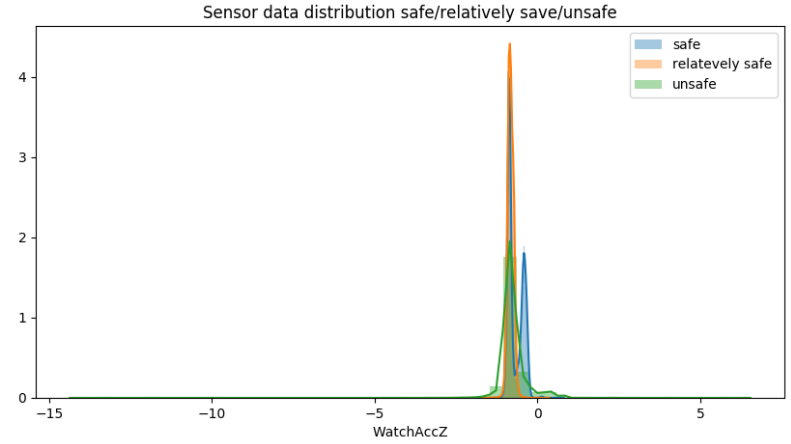


FIGURE D.15: Sensor data distribution WatchAccZ

Sensor data plots

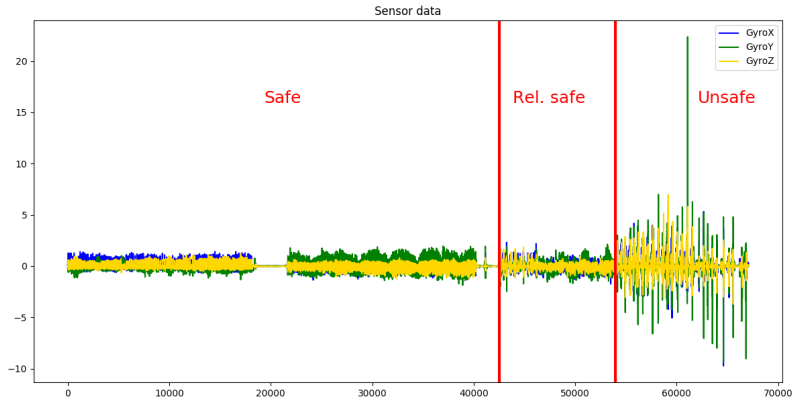


FIGURE D.16: Data plot GyroX, GyroY, GyroZ

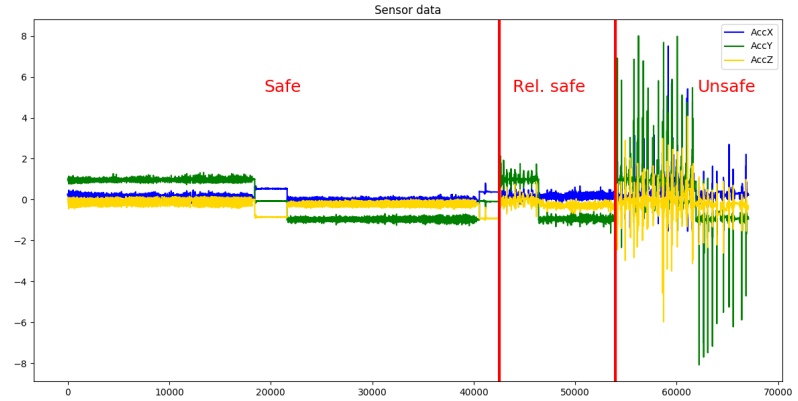


FIGURE D.17: Data plot AccX, AccY, AccZ

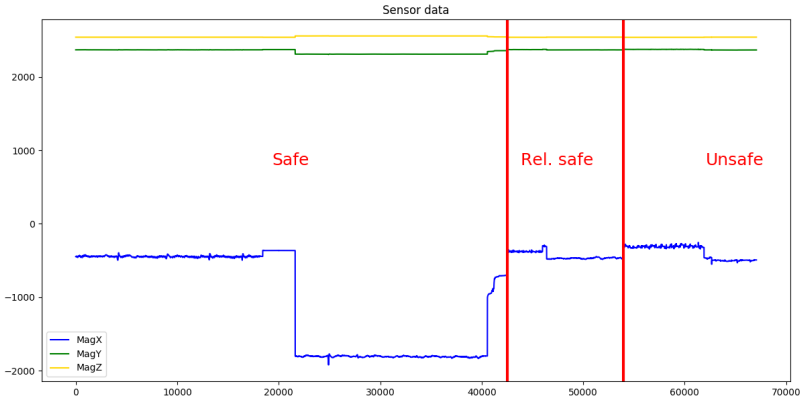


FIGURE D.18: Data plot MagX, MagY, MagZ

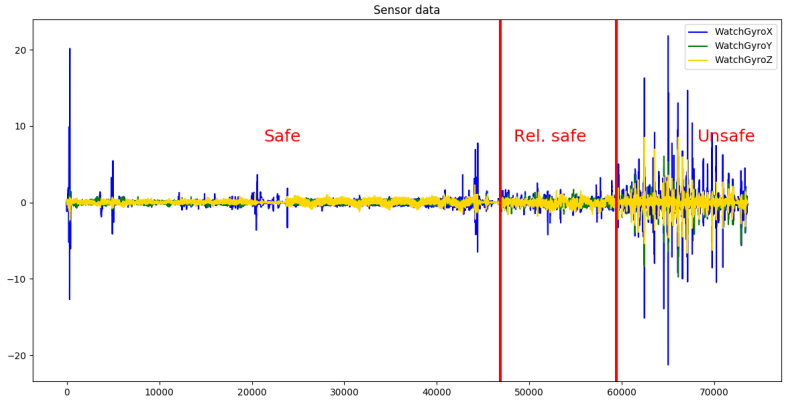


FIGURE D.19: Data plot WatchGyroX, WatchGyroY, WatchGyroZ

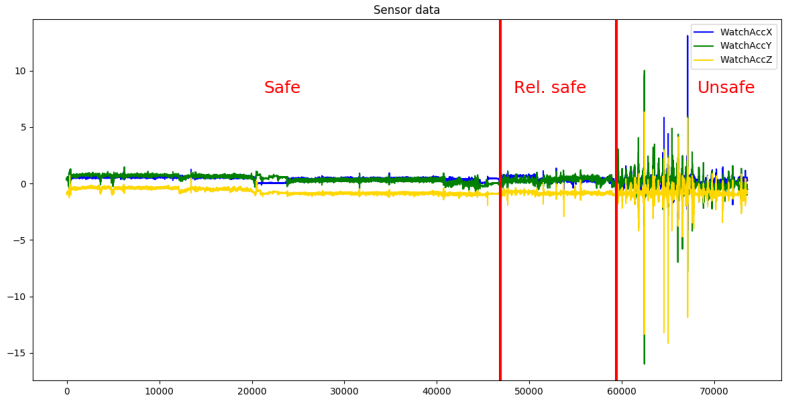


FIGURE D.20: Data plot WatchAccX, WatchAccY, WatchAccZ

Comparison single characteristic of all sessions

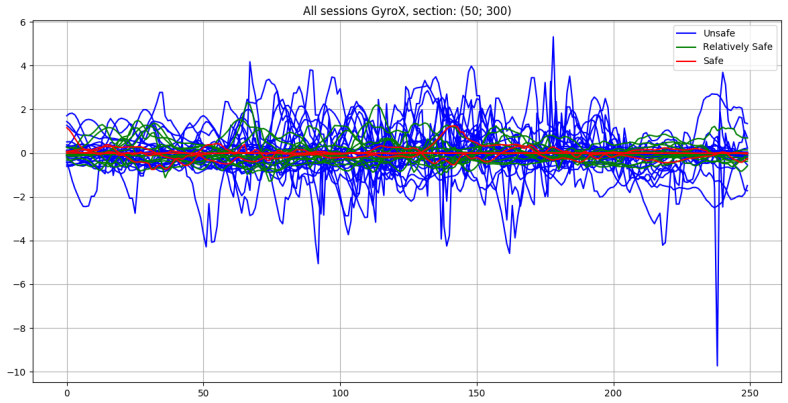


FIGURE D.21: GyroX, all sessions comparison

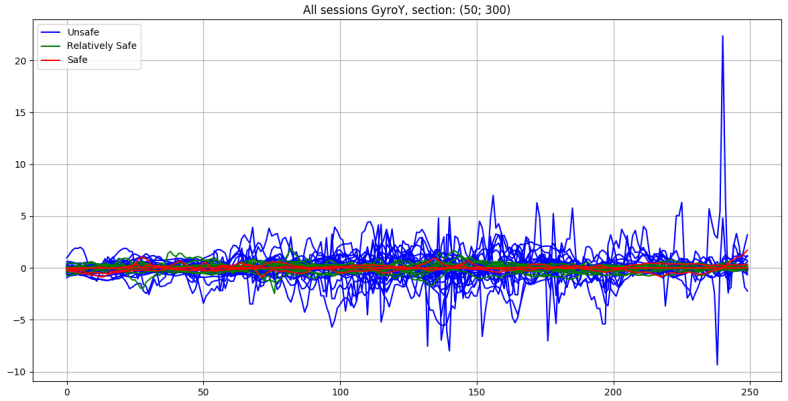


FIGURE D.22: GyroY, all sessions comparison

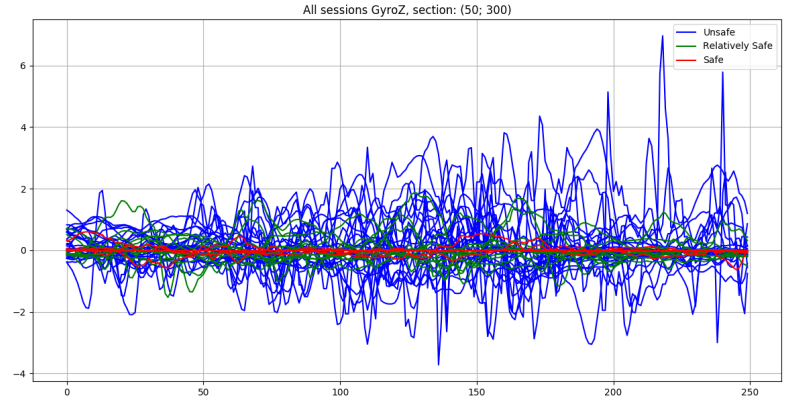


FIGURE D.23: GyroZ, all sessions comparison

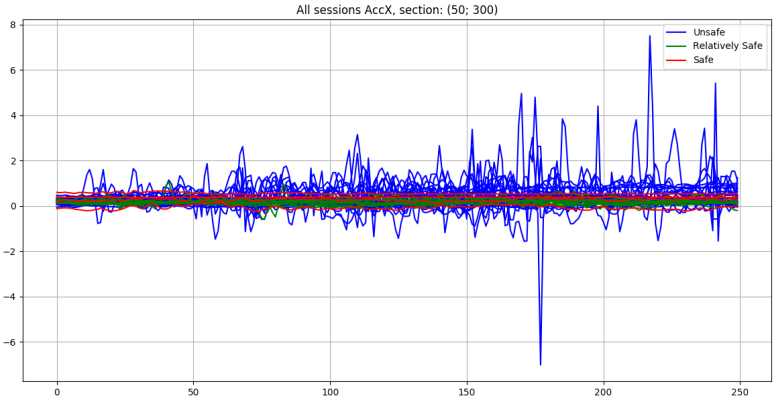


FIGURE D.24: AccX, all sessions comparison

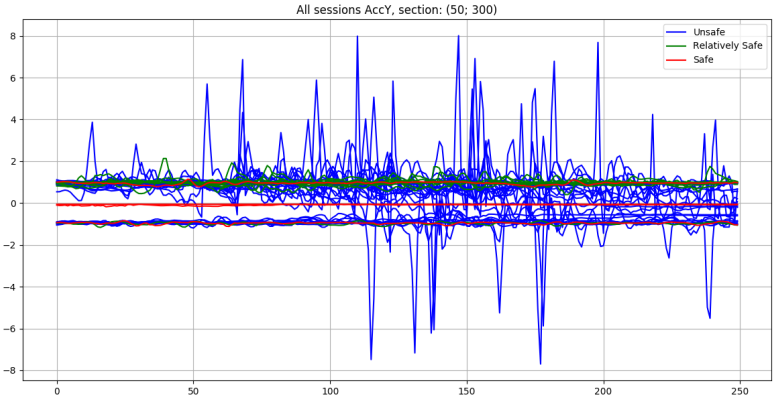


FIGURE D.25: AccY all sessions comparison

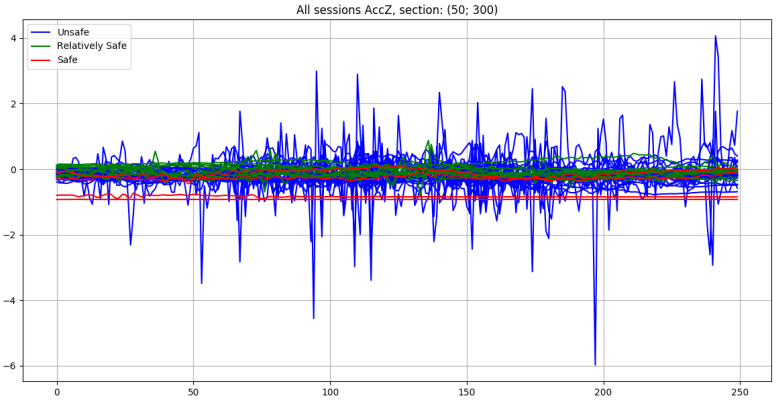


FIGURE D.26: AccZ all sessions comparison

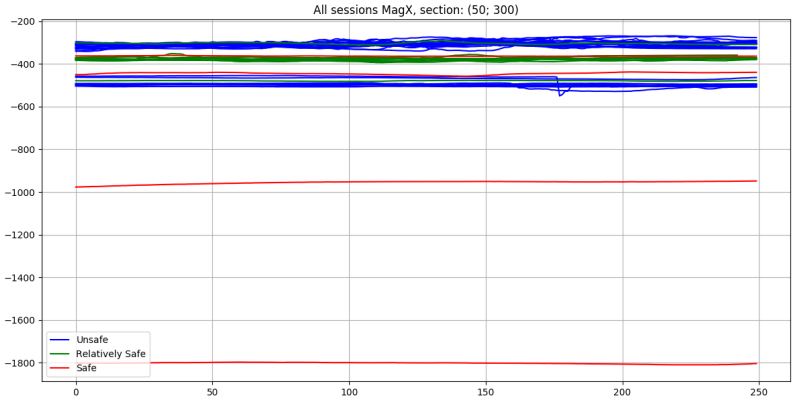


FIGURE D.27: MagX all sessions comparison

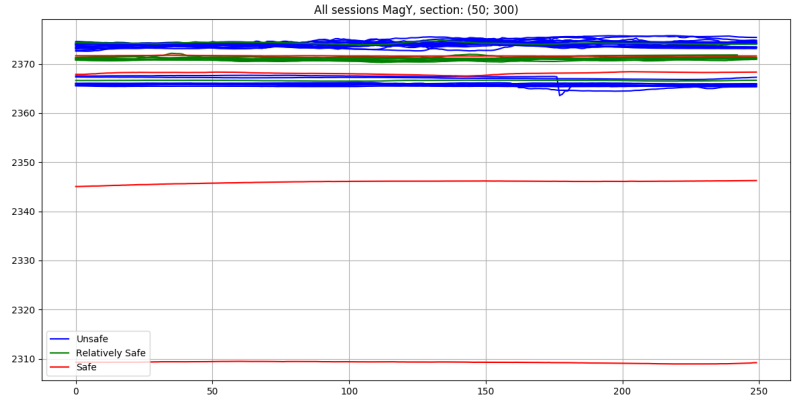


FIGURE D.28: MagY all sessions comparison

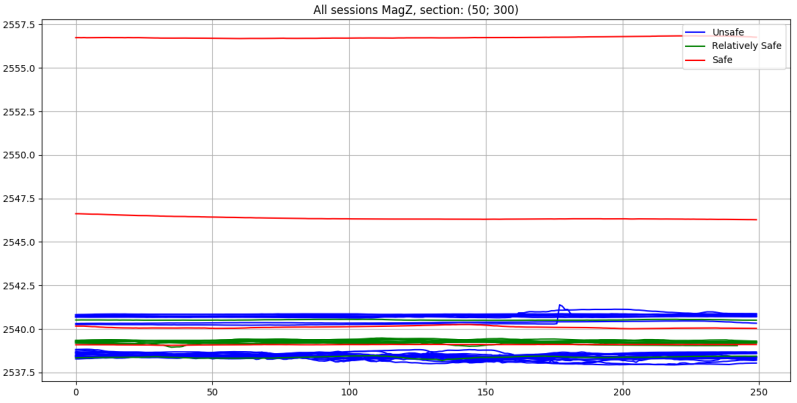


FIGURE D.29: MagZ all sessions comparison

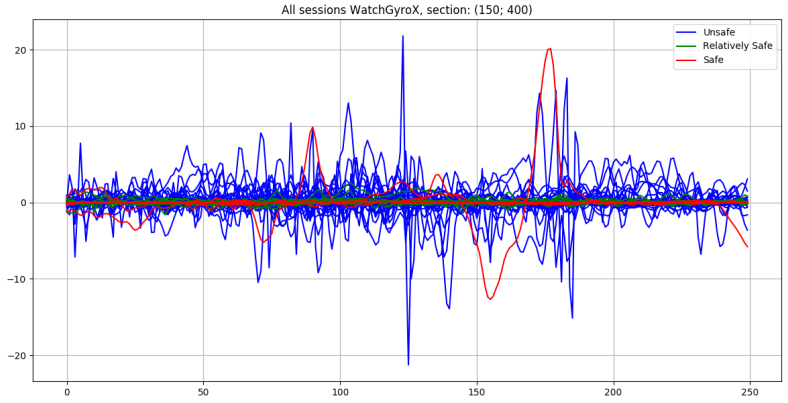


FIGURE D.30: WatchGyroX, all sessions comparison

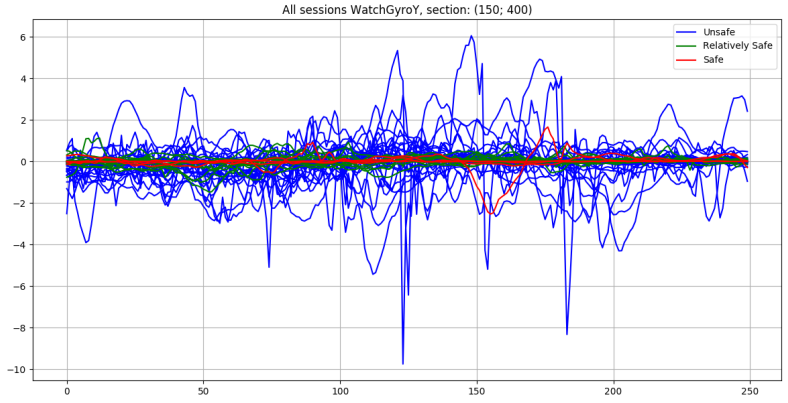


FIGURE D.31: WatchGyroY, all sessions comparison

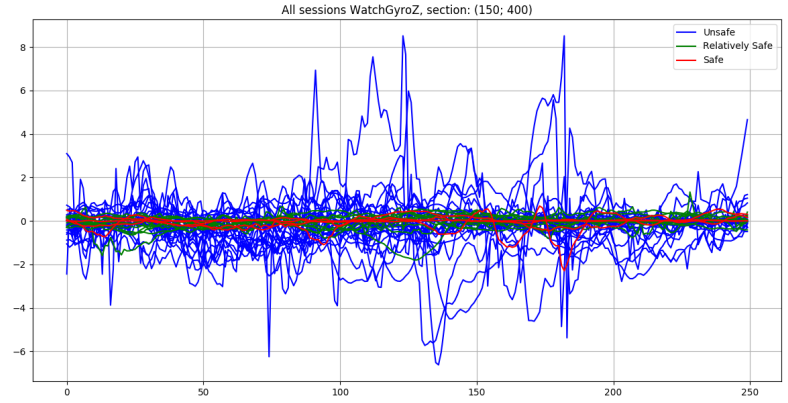


FIGURE D.32: WatchGyroZ, all sessions comparison

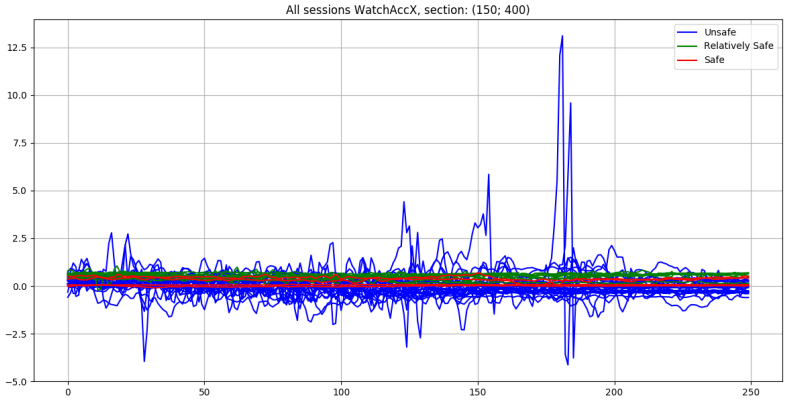


FIGURE D.33: WatchAccX, all sessions comparison

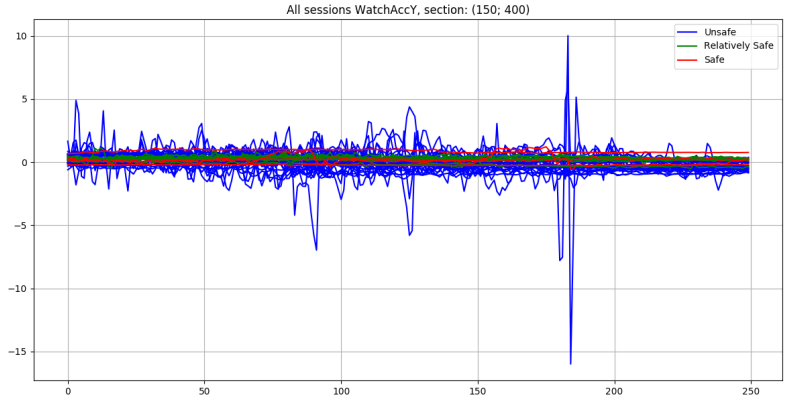


FIGURE D.34: WatchAccY all sessions comparison

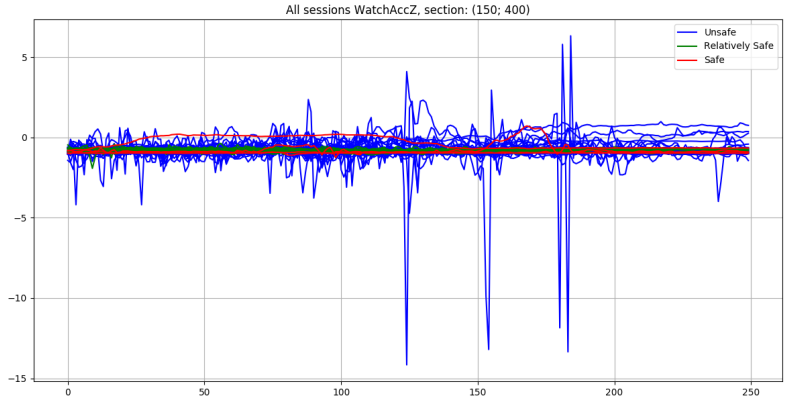


FIGURE D.35: WatchAccZ all sessions comparison

Prediction plots by 12 samples using neural network

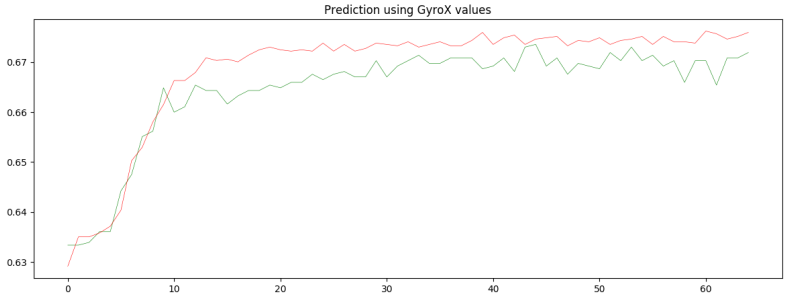


FIGURE D.36: Prediction using GyroX values, result: 65.94% (3.12%)

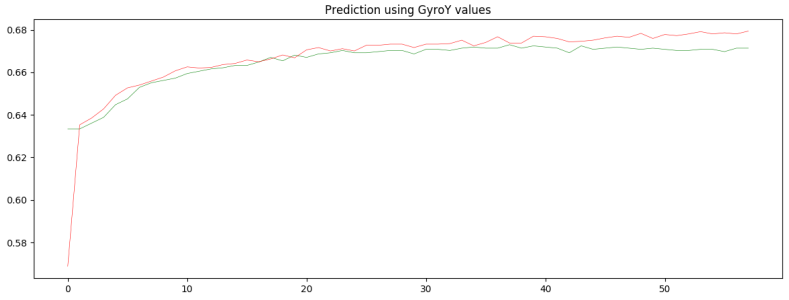


FIGURE D.37: Prediction using GyroY values, result: 65.34% (3.83%)

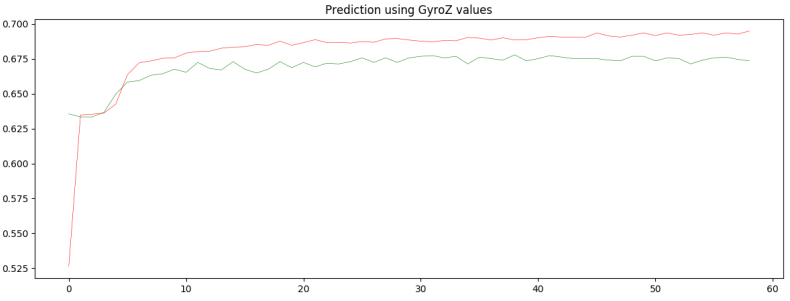


FIGURE D.38: Prediction using GyroZ values, result: 67.62% (2.80%)

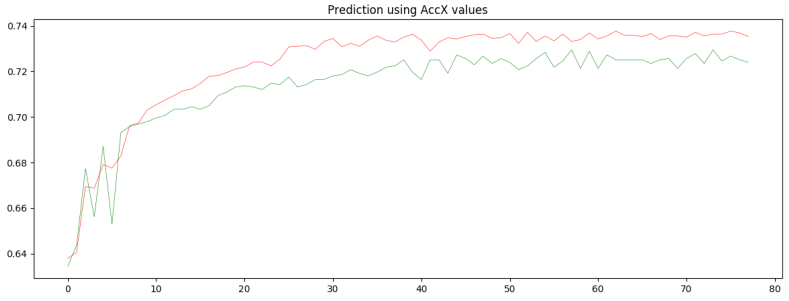


FIGURE D.39: Prediction using AccX values, result: 72.50% (2.24%)

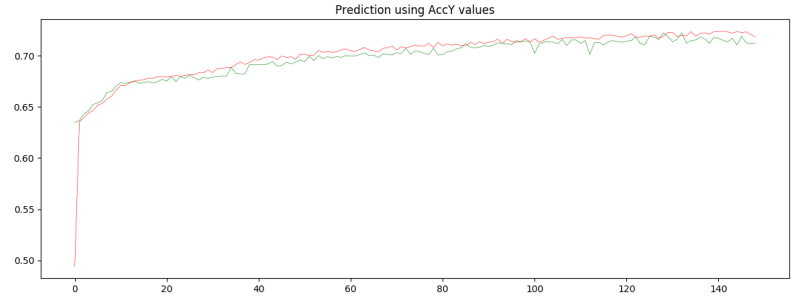


FIGURE D.40: Prediction using AccY values, result: 71.09% (3.78%)

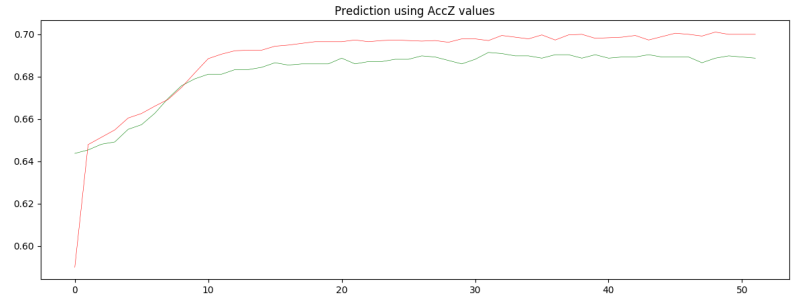


FIGURE D.41: Prediction using AccZ values, result: 70.44% (3.11%)

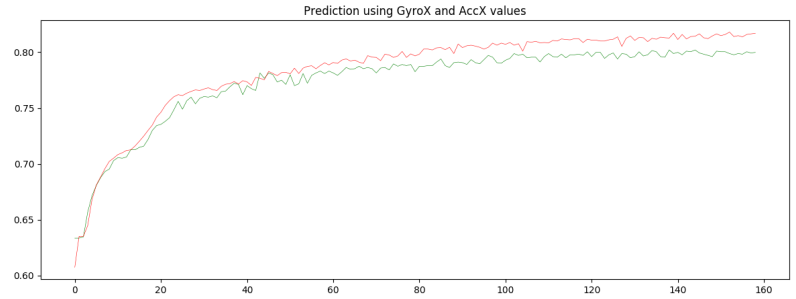


FIGURE D.42: Prediction using GyroX and AccX values, result: 77.38% (2.80%)

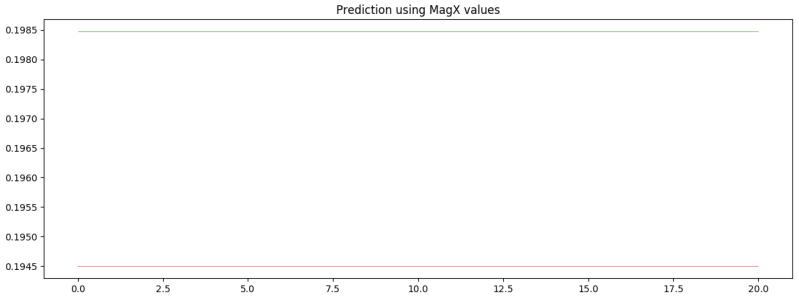


FIGURE D.43: Prediction using MagX values, result: 36.06% (21.72%)

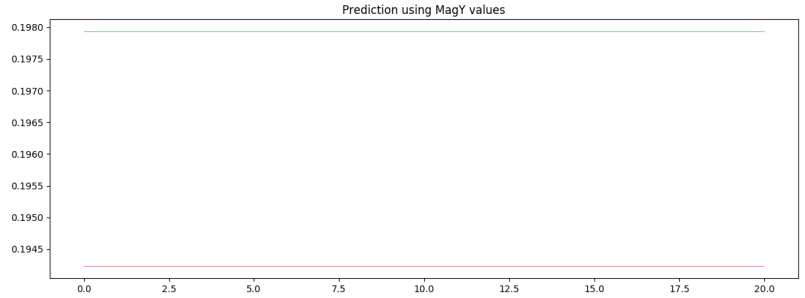


FIGURE D.44: Prediction using MagY values, result: 41.05% (22.91%)

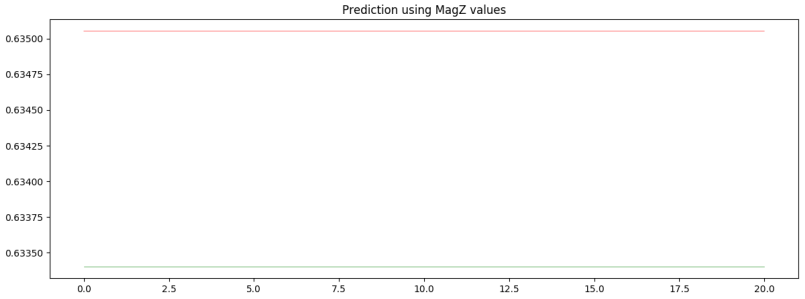


FIGURE D.45: Prediction using MagZ values, result: 45.15% (20.87%)

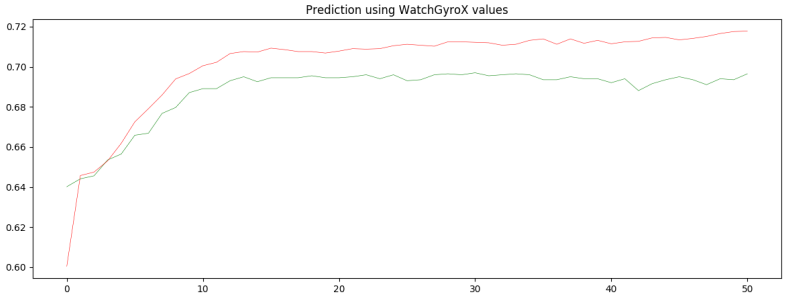


FIGURE D.46: Prediction using WatchGyroX values, result: 69.21% (2.97%)

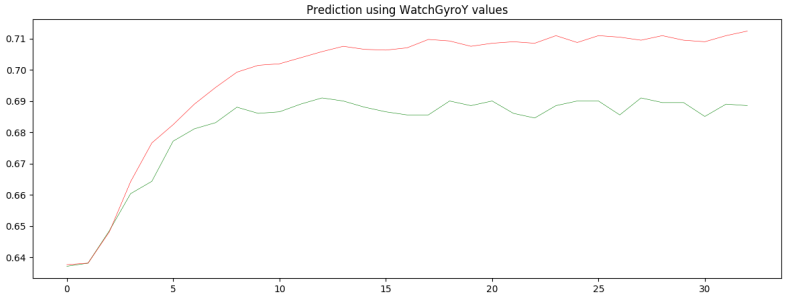


FIGURE D.47: Prediction using WatchGyroY values, result: 68.96% (3.85%)

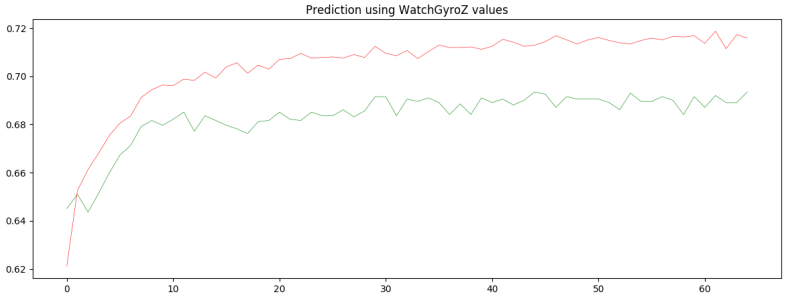


FIGURE D.48: Prediction using WatchGyroZ values, result: 68.61% (3.71%)

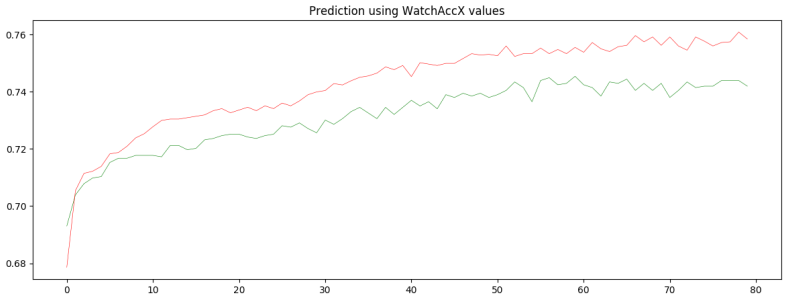


FIGURE D.49: Prediction using WatchAccX values, result: 72.96% (2.85%)

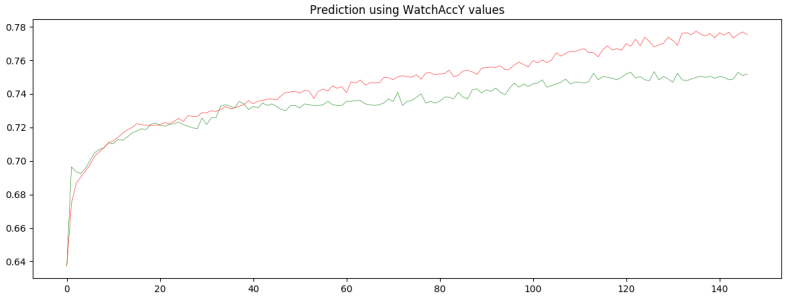


FIGURE D.50: Prediction using WatchAccY values, result: 76.92% (2.91%)

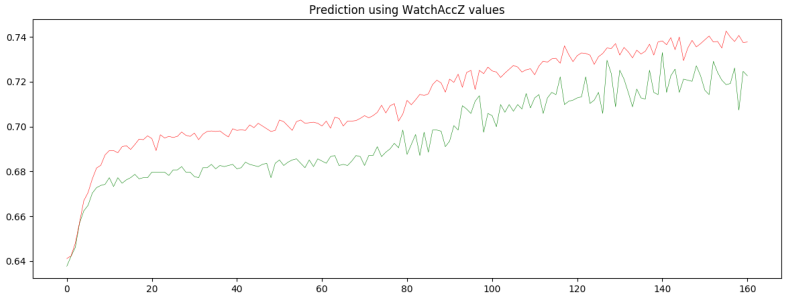


FIGURE D.51: Prediction using WatchAccZ values, result: 69.35% (4.21%)

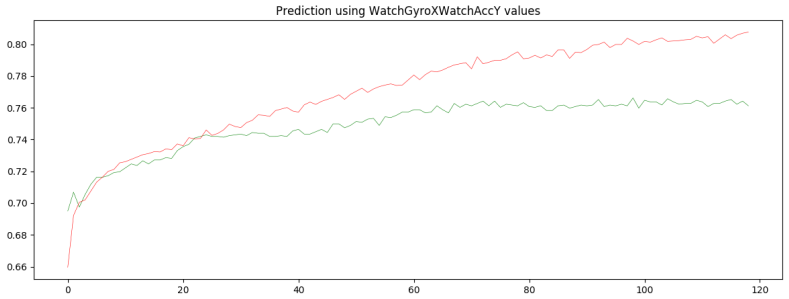


FIGURE D.52: Prediction using WatchGyroX and WatchAccY values, result: 74.45% (2.88%)

Appendix E

Source code

DataCollector (Collecting data using SensorTag 2.0 CC2650TK)
https://github.com/degtiarev/DataCollector

MotionCollector (Collecting data using iPhone and Apple Watch)
https://github.com/degtiarev/MotionCollector

MotionDataHandler (Handling data, building plots and models)
https://github.com/degtiarev/MotionDataHandler

MotionDataHandler (Prototype of app (iPhone & Apple Watch) for detection and prediction of falls among elderly people using walkers)
https://github.com/degtiarev/SafeWalk

Appendix F

Thesis description document

Faculty of Engineering Science and Technology
Department of Computer Science and Computational Engineering
UiT The Arctic University of Norway

Detection and prediction of falls among elderly people using walkers

Aleksei Degtiarev
Thesis for Master of Science in Computer Science

Problem description

This proposal is meant to extend previous work at UiT Narvik on fall risk assessment carried out by Elisabeth Gangenes (2016) and PhD work by Asbjørn Danielsen (2015-2016).

The idea is to create a means for detecting and possible prevent falls among elderly people that use walkers. A concept for detecting instabilities and risk of falls using machine learning should be designed. A risk of fall or fall should immediately emit an alarm signal on the spot and by means of a message emitted from the smart device. In addition, the design should include a specification of how falls can be automatically controlled and prevented based on the detected risks. Such control could be embedded in the walker or as a kind of wearable or similar.

A controlled experiment should be designed to train a wearable system to detect instabilities and risk of falls. This could be by means of a smart watch or smart phone or both. A set of sensors should be identified for the purpose and connected to the wearable device. Communication with between sensor and controlling device should be based on Bluetooth. The same approach as Elisabeth Gangenes can be applied. The student is at liberty in the choice of machine learning technique. But it is recommended that a cluster technique, a boosted CART or a LTSM network is applied. The trained model should be converted to a core ML model format.

The thesis will consist of four sub-tasks each with a distinct goal:

1. Review state-of-the-art literature and design the concept
2. Create an experimental rig and conduct experiments to generate a training and test set for machine learning.
3. Demonstrate the accuracy of fall risk detection when using a walker based on sampling from task 2 above
4. Create a pilot test with subjects using a walker

Dates

Date of distributing the task:	<12.01.2018>
Date for submission (deadline):	<1.6.2018>

Contact information

Candidate	Aleksei Degtiarev delexa0@gmail.com
Advisor at UiT-IVT	Bernt A. Bremdal bernt.a.bremdal@uit.no
Advisor at UiT-IVT	Asbjørn Danielsen Asbjorn.danielsen@uit.no

General information

This master thesis should include:

- Preliminary work/literature study related to actual topic
  - A state-of-the-art investigation
  - An analysis of requirement specifications, definitions, design requirements, given standards or norms, guidelines and practical experience etc.
  - Description concerning limitations and size of the task/project
  - Estimated time schedule for the project/ thesis
- Selection & investigation of actual materials
- Development (creating a model or model concept)
- Experimental work (planned in the preliminary work/literature study part)
- Suggestion for future work/development

Preliminary work/literature study

After the task description has been distributed to the candidate a preliminary study should be completed within 3 weeks. It should include bullet points 1 and 2 in “The work shall include”, and a plan of the progress. The preliminary study may be submitted as a separate report or “natural” incorporated in the main thesis report. A plan of progress and a deviation report (gap report) can be added as an appendix to the thesis.

In any case the preliminary study report/part must be accepted by the supervisor before the student can continue with the rest of the master thesis. In the evaluation of this thesis, emphasis will be placed on the thorough documentation of the work performed.

Reporting requirements

The thesis should be submitted as a research report and could include the following parts; Abstract, Introduction, Material & Methods, Results & Discussion, Conclusions, Acknowledgements, Bibliography, References and Appendices. Choices should be well documented with evidence, references, or logical arguments.

The candidate should in this thesis strive to make the report survey-able, testable, accessible, well written, and documented.

Materials which are developed during the project (thesis) such as software / source code or physical equipment are considered to be a part of this paper (thesis). Documentation for correct use of such information should be added, as far as possible, to this paper (thesis).

The text for this task should be added as an appendix to the report (thesis).

General project requirements

If the tasks or the problems are performed in close cooperation with an external company, the candidate should follow the guidelines or other directives given by the management of the company.

The candidate does not have the authority to enter or access external companies' information system, production equipment or likewise. If such should be necessary for solving the task in a satisfactory way a detailed permission should be given by the management in the company before any action are made.

Any travel cost, printing and phone cost must be covered by the candidate themselves, if and only if, this is not covered by an agreement between the candidate and the management in the enterprises.

If the candidate enters some unexpected problems or challenges during the work with the tasks and these will cause changes to the work plan, it should be addressed to the supervisor at the UiT or the person which is responsible, without any delay in time.

Submission requirements

This thesis should result in a final report with an electronic copy of the report including appendices and necessary software, source code, simulations and calculations. The final report with its appendices will be the basis for the evaluation and grading of the thesis. The report with all materials should be delivered according to the current faculty regulation. If there is an external company that needs a copy of the thesis, the candidate must arrange this. A standard front page, which can be found on the UiT internet site, should be used. Otherwise, refer to the "General guidelines for thesis" and the subject description for master thesis.

The advisor(s) should receive a copy of the the thesis prior to submission of the final report. The final report with its appendices should be submitted no later than the decided final date.