TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Ainar Assuküll 212088 IASM

INVESTIGATING BEHAVIORAL ANOMALIES USING MACHINE LEARNING

Master Thesis

Supervisor
Kalle Tammemäe
PhD

Tallinn 2022

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Ainar Assuküll 212088 IASM

KÄITUMUSLIKE ANOMAALIATE TUVASTAMINE KASUTADES MASINÕPPE MEETODEID

Magistritöö

Juhendaja
Kalle Tammemäe
PhD

Tallinn 2022

Author's declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Ainar Assuküll .....
(signature)

Date: 03.05.2022

Annotatsioon

Käesolev magistritöö käsitleb inimese võimalike käitumuslike anomaaliate või ohuolukorda sattumise tuvastamist tema oma kodus ühe toa piires. Töö fookuses on eakate inimeste heaolu ja iseseisva hakkamasaamise toetamine. Andurina inimeste tegevuse jälgimisel on kasutatud põhiliselt Omroni infrapuna maatrikstermosensorit (IRMT). IRMT sensorist tulev info loetakse arvutisse, kus luuakse konkreetsest ruumist termopilt ja seda analüüsitakse reaalajas, et tuvastada anomaaliaid ja ohuolukordi. Termopildi põhjal tuvastatakse erinevaid tegevusi, nagu televiisori vaatamine, toas liikumine või magamine. Hinnatakse võimalusi eristamaks järsku asendi muutust (kukkumist) tavalisest toas ringi liikumisest. Lisaks uuritakse uneanomaaliad võttes appi andmed aktiivsusmonitorist. Täiendavalt uuritakse masinõppe meetodeid sisendandmete lihtsustamiseks ja algoritmide efektiivsemaks tegemiseks.

Töö käigus luuakse arvutiprogramm, mis reaalajas, kasutades masinõppe meetodeid, analüüsib IRMT sensori väljundit ja ennustab anomaaliad. Lisaks luuakse masinõppemudel, mis kirjeldatakse C keeles realiseerimaks ja valideerimaks mudelit mikrokontrollersüsteemil. Täiendavaks tulemuseks on skriptid andmete visualiseerimiseks ja masinõppe mudelite analüüsimiseks Pythoni keeles.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 54 leheküljel, 10 peatükki, 26 joonist, 4 tabelit.

Abstract

This thesis investigates behavioral anomalies with focus on helping elderly people to live safer on their own. Main input device used in current study is Omron D6T matrix thermal sensor. The system consists of matrix thermal sensor, micro-controller and general purpose computer (desktop or mobile), connected to the Internet. Sensor is used to create a low resolution matrix thermal image of a specific room. Image is analyzed in the PC to detect human behavior e.g. watching TV, sleeping or moving around in the room. Investigated detecting and differentiating falling from moving around in the room. While person is watching TV, body temperature is assessed to detect possible fever. Sleep anomalies detection is made with help from activity sensor. Machine learning (ML) methods are used to make analysis easier and/or more efficient.

The result of the thesis is a computer program prototype, which uses data from the thermal sensor and implements machine learning model to detect anomalies. The ML model is also written in C code to run on a micro-controller (STM32F4). Additional result is python scripts, which are used to help visualize and analyze data to select machine learning models.

The thesis is in English and contains 54 pages of text, 10 chapters, 26 figures, 4 tables.

List of abbreviations and terms

IRMT sensor	Infrared matrix thermal sensor - Omron D6T-44L
Activity Monitor	PC application software name created during this thesis work
MCMC	Markov chain Monte Carlo
PIR sensor	Passive infrared sensor (Wikipedia)
MCU	Microcontroller unit (Wikipedia)
STM32	Family of 32-bit microcontroller integrated circuits by STMicroelectronics (Wikipedia)
COM	Communication port (Wikipedia)
GPS	Global Positioning System (Wikipedia)
SIM	Subscriber Identification Module (Wikipedia)
C#	Microsoft developed programming language (docs.microsoft.com)
.NET	.NET Framework - Microsoft software platform (dotnet.microsoft.com)
ML.NET	Microsoft open source machine learning framework (dotnet.microsoft.com)
Windows Forms	Microsoft open source graphics library (docs.microsoft.com)
UART	Universal Asynchronous Receiver-Transmitter (Wikipedia)
Bluetooth	Standardized wireless communication technology (Wikipedia)
SMS	Short Message Service (Wikipedia)
IBM	International Business Machines - American technology company (Wikipedia)
Gaussian distribution	Normal distribution (Wikipedia)
ID	Identity document (Wikipedia)
JSON	JavaScript Object Notation (json.org)
Python	General purpose programming language (python.org)
scikit-learn	Open source machine learning framework in Python (scikit-learn.org)

pandas	Open source data analysis and manipulation tool (pandas.pydata.org)
matplotlib	Data visualization tool(matplotlib.org)
numpy	Package for scientific computing with Python (numpy.org)

Table of Contents

List of Figures

List of Tables

1. Introduction

In given master thesis research is made to detect anomalies in human behaviour and possible danger situations. The thesis has three focus topics. The the main focus is on elderly people to live independently for longer and safer. Elderly could fear what happens if one falls and cannot get up on his/her own. Getting help quickly could be vital. One option is to monitor person and then assist if needed. For monitoring video camera can be used, but it has intrusive nature to persons privacy. Much less invasive is low resolution matrix thermal sensor (IRMT). This work is mainly focused on analysing the data from given sensor. Thermal image of the room is created, which is analyzed to monitor human behaviour and to detect anomalies and danger situations. The data is processed using machine learning methods on a personal computer (Windows PC) and also the machine learning model is developed for the microcontroller (STM32F4). The benefit of having the solution run on a smaller device is its lower price, size and energy consumption.

The second focus of current thesis is continuation of work done in given master thesis [1] by Christel Nilsen, where author investigated using IRMT sensor for detecting human in a specific room in different distances and positions.

The third focus is detecting sleep anomalies, based on the status received from the thermal image of the room.

1.1 Research questions

1. Which infrared sensors are investigated for monitoring human activity?

Infrared sensors are used for example detecting movement (PIR sensors) or measuring temperature. In given work author investigates detecting human behaviour using matrix thermal sensor (IRMT)

2. How to use machine learning methods to simplify detecting different human activities?

In the given work different machine learning methods and frameworks are studied to process data from the IRMT sensor to detect different activities by one person in a given

room.

3. Is use of machine learning meaningful to detect anomalies?

Based on the models from different machine learning methods, it will be studied if detecting anomalies is possible using these models.

4. Is IRMT sensor useful for detecting dangerous situations?

Different experiments are made using the author as a test subject to create dangerous situations and the IRMT sensor ability to help detect them.

5. Can machine learning model, which monitors human activity, run on a microcontroller (MCU)?

If the machine learning model works successfully on a PC, investigate if it can also be used efficiently on a MCU.

2. Literature Review

Overview of research made in the related subject. First thesises on similar subject from TalTech are provided and also research which was found from the internet. Chapters from 2.1 to 2.4 describe the use of IRMT sensor and 2.5 describes sleep anomalies research.

2.1 Localization methods for elderly people assistance

Christel Nilsen investigated in her master thesis [1] using of Omron IRMT to detect warm objects and their position in the room.
In her experiments 4x4 IRMT sensor is put in the upper corner of the room. It is used to investigate detecting different objects including a person. Also person's position in the room is investigated. The focus is having the end solution to work on a micro-controller. To calibrate the room thermal images are made with all the heat emitting objects both switched on and off (e.g. TV or a lamp). This way the statical items in a room are discovered. As a result the calibrated system does not start detecting human when e.g. lamp is switched on. Person is investigated as a dynamic object in the room. When a person moves the temperature changes in the thermal matrix rapidly. Binary matrix is created where '1' means that in the given point human is detected. Precondition is that there is only one person in a room and the human's temperature is always higher than the room temperature. Person's distance from the sensor investigated by measuring the temperature drop in different lengths. Test program is created which helps to analyze data from the IRMT sensor.
Author's work proves that IRMT sensor can be used to detect a person and his/her approximate position in the room. Current work will continue this work by using two sensor to get improved field of view and to detect more detailed situations with the help of the machine learning.

2.2 Developing internet of things and machine learning based bidirectional people counting system with passive infrared sensors

In this master thesis [2] author investigates PIR sensors for counting number of people in a room. This can be useful for e.g. adjusting the heating or ventilation. Author makes a working product prototype where two PIR sensors are connected to the Arduino and it

counts people with 93% accuracy. Counting is done by detecting a person moving either in or out the room. To count, 2 sensors are placed so that if one sensor detects movement and the second within specified time, the counter is increased, and the counter is decreased when the order of detection is the opposite. Author also investigates machine learning method Support Vector Regression (SVR), but because COVID-19 there is only limited field data collected and only initial predictions are made. Author mentions that more data is needed to make more accurate estimates with machine learning methods.
Author's work successfully monitors persons count in the room using a infrared sensor. This work will continue in given thesis with more focus on the machine learning.

2.3 Non-intrusive Human Activity Recognition with Low-Resolution Infrared Array Sensor Using Long Short-Term Memory Neural Network

In this paper [3] authors investigate 8x8 IRMT sensor to detect human activity recognition (HAR). A long short-term memory (LSTM) is used to extract human activity from the filtered low resolution thermal image. Their motivation to use the IRMT sensor is its relatively low cost and non-invasive nature.

Authors put IRMT sensor on the ceiling covering 3,3 x 2 meters detection area. Authors investigate detecting 4 activities (lying, standing, sitting and walking) plus empty room. 20 sequential frames of 8x8 image is collected to get one input for the model. Also J-filter and Butterworth filter is used to preprocess the input data. This improves accuracy from 84% with raw to 98 % with filtered data. Different recognition algorithms are tested and the LSTM gives the best overall result of 98%.
Authors prove that using IRMT sensor to recognise human activity is possible. Current work will continue this study and also try to detect anomalies.

2.4 Tracking Motion and Proximity using Thermal-sensor Array

In this paper[4] authors investigate detecting number of people and their direction of motion using Panasonic Grid-EYE 8x8 matrix thermal sensor. Their motivation to choose thermal image instead of video-camera is because of video-cameras intrusive nature. 8x8 thermal image can provide enough tracking info while preserving privacy of the subjects.

They saved 902 scenes (images from thermal sensor) with up to 4 people of different height. They subtracted background by subtracting per pixel long term temperature average. They found that usually it is the head of the person that has the highest temperature and therefore

finding local peaks in the thermal image helps identifying number of persons. They started with using K-means clustering for number of instances estimation. This approach had a problem distinguishing between 2 and 3 persons. They then used Support Vector Machine classification and then achieved 80% accuracy.

For motion and direction detection they used 2,5m x 2,5m view from the sensor mounted in the ceiling and only one person moving, without any other person on the scene. The sampling rate for the sensor was 10 samples per second. Authors extracted the time series from each pixel and computed a cross-correlation matrix with other cells. They used also delay between boundary cells to get the direction. With this approach they successfully identified 4 different directions of person walking in normal speed. For future work they suggested detecting also more complex scenes with multiple persons moving or more than 4 directions of moving.
This work was another example of researching human activity using IRMT sensor.

2.5 Markov chain and Monte Carlo for sleep time analysis

In given article [5] Will Koehrsen investigates Markov chain[6] and Monte Carlo methods [7] (MCMC) using own sleep data. Author takes input sleep data (time to fall asleep and the sleep duration) from the activity monitor around the wrist. Formula which represents time to fall a sleep is researched. Author creates a graph where 0 is being awake and 1 is asleep during last 60 nights (Figure [1]). From the graph it is visible, that time to fall

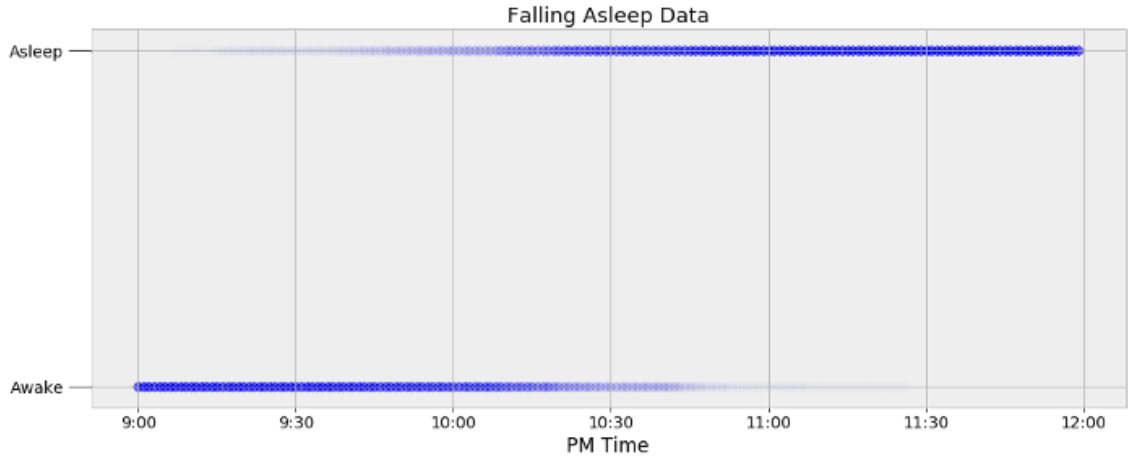

Figure 1. Will Koehrsen sleep start time with blue dots [5]

a sleep is usually after 10PM. Further it is evident that person does not fall asleep in the same time every day and the shape of the graph seems to represent logistic equation. 2.1

$$p(sleep|time) = \frac{1}{1 + e^{\beta * t + \alpha}} \tag{2.1}$$

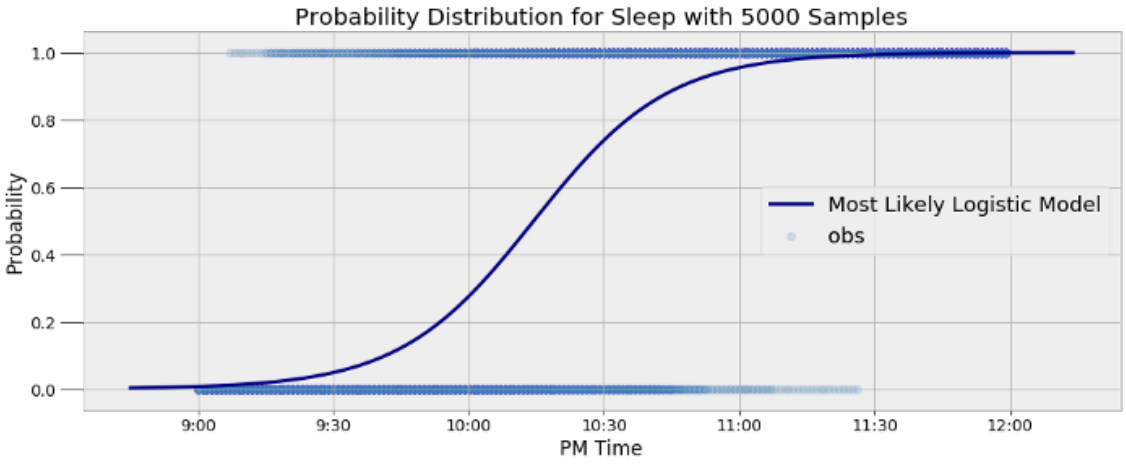

Figure 2. Logistic model for going to sleep [5]

The β and α are parameters which could be learned using MCMC methods. Author uses PyMC3 [8] library. The result is a logistic model (Figure [2]) which shows probability, that author sleeps at given time. At 22:14 the probability that author sleeps is over 50%.

Author continues by investigating the duration of the sleep and discovers that normal distribution does not describe this the best way. The reason is that while most of the time author sleeps around 8 hours, there are multiple times where author sleeps significantly more. To calculate the distribution graph asymmetric Metropolis-Hastingsi algorithm [9] is used. Resulting graph (Figure [3]), where the right side is not so steep. Average sleep duration is 7,67 hours.

In conclusion author finds this sleep data as a good data-set for learning Markov chains and Monte Carlo methods. This thesis work will also study detecting sleep/awake anomalies.

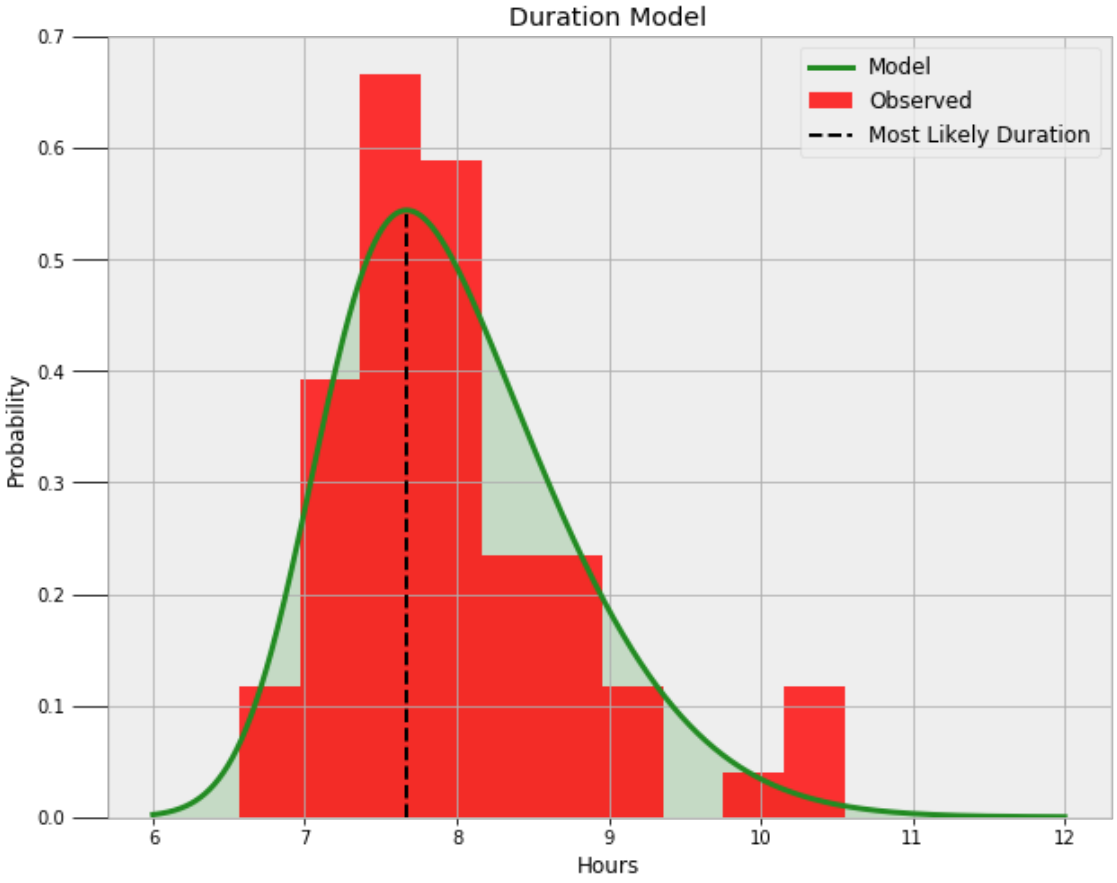

Figure 3. Sleep duration model [5]

3. Market Status

In given chapter author gives overview of current market status for technical devices which help aging people to live more safely and independently in their own homes. Author focuses on devices which help detect dangerous situations and can work autonomously.

3.1 Video camera

With today's wide range of affordable IP cameras it is quite often a practice that close ones put a camera to their elderly relative home. This is to monitor their health and to assure that everything is under control. The camera gives an accurate overview of the situation in the placed room. In the article [10] Gareth Williams describes his experience using a video camera to care for his mother and father. Author gives advice both on technical and legal side. The legal part goes for the United States and the United Kingdom context. The biggest negative side of the camera based solution is its privacy invasive nature. Not many people would like to be monitored 24 hours a day. Legally it also can be problematic, especially if the camera is put secretly, although the intentions might be good.

3.1.1 Giraff avatar

In the article [11] video call device is introduced. It was taken into use 2010 in a Swedish nursing home and what is special about in this device, is that it looks as much as possible as a human. The size of a screen provides visibility for the full size person head and shoulders. Device can be in standing up or sitting down position. It can turn the head unit, providing the caller full view of the room. The main benefit is that for the client in the nursing home it feels more like talking to a person than on the video call on a phone. When developing the device, clients privacy is taken into consideration e.g. this device cannot be used to record a video.
This work gives an example of a technical solution using video camera to monitor and help elderly without actual person having to be in the room. Given thesis will also monitor a person, but with non-intrusive IRMT sensor.

3.1.2 Proctorio

Proctorio [12] is an add-on for the Google Chrome browser, which enables to autonomously check that person who makes an exam does not cheat. It is used when the test is made using a personal computer, either at school or in the students home. Solution uses web-camera to analyze examinee and also the surroundings in the room. Program has following features:

- User identification using persons ID card and facial recognition.
- Examinee desk and room analysis to detected unauthorized use of aid.
- Examinee behaviour analysis using machine learning methods and also optional expert opinion.
- Tests originality proofing, gives an estimate on how likely the test is a plagiarism.
- Locking the PC for any other activity e.g. user cannot open new Internet browser tab.

Connection to authors given thesis is that Proctorio also uses machine learning to detect behavioral anomalies, in this case cheating in an exam.

Opinions

In the Internet there are many opinions (mostly from students) about using the Proctorio during the exams. In the Reddit group discussion [13] participants were interested on how effective the program is and how easy it would be to cheat. Answers were very different, some said that even the slightest looking in the other direction caused problems and others that they cheated without any consequences. Some came to the conclusion that it depends on the program settings and also what the teacher considers as cheating.
In the Daily Illini article [14] students raise concern that Proctorio intrudes their privacy. It also needs high speed Internet and a personal computer, but many students use only a mobile phone. Some professors have taken the Proctorio into use, but because of mentioned concerns not all.

3.2 Movement sensors

Given thesis focuses on assisting elderly in non-invasive way. Movement sensors for given purpose are mostly either fall detectors (3.3) or sensors attached to moving objects e.g. refrigerator door. In the following chapters some of the sensor systems are introduced and their working principles explained.

3.2.1 Lively passive remote monitoring sensors

In given article [11] elderly assistance and monitoring systems are introduced. In San Fransisco a company named Lively started selling remote monitoring devices in 2013. Sensors with accelerometer are placed on different devices which move in persons home, e.g. key-chain or refrigerator door. Over *Bluetooth* movement detection is sent to a base station. Base station sends data to a central server which compares and analyses data against expected movement. Expected data is gotten by monitoring the client during the first week of device use. Clients relatives can monitor remotely to see if the client situation is as expected, shown as a green status. When detecting danger situation e-mail or SMS is sent to a client's contact. When designing the system clients privacy is taken into consideration, e.g. person monitoring the client cannot know how often client uses toilet.

3.3 Fall detection sensors

Falling at home, especially in case of elderly, is quite common and might result serious consequences. Every year 36 million elderly people (65+) fall in the United States alone, resulting in over 32 000 deaths [15]. Due to unexpected medical condition or due to bad health in general, it might be hard or even impossible to stand up again. Getting help could be vital. To solve this problem of getting help, the market provides fall detection sensors. These are mostly attached to the person and could send a message to either persons relative or to medical personnel.

3.3.1 General working principles

Fall detectors are generally attached to a person and the accelerometer [16] inside detects falling and then the device raises the alarm. Device usually has a button to cancel the alarm if a person manages to get up or it is a false alarm. More advanced versions have internal algorithms for detecting if person can get up on its own. The main disadvantage with these kind of devices is the constant need to be attached to a person, which one can forget or could just be uncomfortable.

3.3.2 Devices available on the market

Following table [1] introduces some of the devices available in the market which are considered best for year 2022 from this article [17] and also best rated (4+) found from amazon web store.

Table 1. Fall detection sensors comparison

Name	Info	Pros	Cons	Price
BlueStar SeniorTech Sentry in-Home Medical Alert	Two device solution. Base station which is connected to a landline and sensor which is worn around neck or wrist.	Can call help with pressing one button and relatively long range (180 m)	Requires landline and available only in the United States	$29.95 a month
SureSafeGO 2 'Anywhere' Alarm	Mobile device solution with mobile SIM card and GPS tracking. Device can be used to contact response center 24/7.	Unlimited range, no base station. Built-in GPS	Battery needs regular charging (every 4-th day)	£149.95 for device plus £18.99 a month
GreatCall Lively	Mobile device worn around neck or wrist, which uses smart phone over Bluetooth as a base station.	Relativley cheap, no charging required	Smart phone has to be in Bluetooth range and only works in the United States	$49.99 for device plus $14,99 a month
Buddi	Mobile device which looks like a wristwatch. Device needs a base station which could be smart phone or a dedicated base station.	Wireless charging. Adjustable fall sensitivity.	Requires regular charging. Customer support only in the UK	£99 device, £149 base station. £1.99 - £4.99 a month
AMG Emergency Call Button [18]	Small mobile device which can be carried with a collar. Has GPS inside and makes alarm call to up to 3 persons. Can be configured via SMS	Unlimited range due to no base station and built in GPS. Monthly cost for only regular SIM card. Low battery warning SMS	Requires charging.	€121.80, no monthly payment, but own SIM card is needed

4. Machine Learning

Author uses Machine learning methods to analyze input data and make the prediction. Use of machine learning has proven efficient [19] in solving problems which include relatively big amount of input data. In the given chapter author gives general introduction to machine learning methods, frameworks and tools used in this work.

Machine learning (ML) is the study of computer algorithms that can improve automatically through experience and by the use of data [20].
There are 3 sub-types in machine learning:

1. Supervised learning [21], which means that during learning input data is labelled. We know result/prediction of each input data set.
2. Unsupervised learning [22], where input data is not labelled at all, mostly clustering is used to group similar data.
3. Reinforcement learning [23], where only part of the input data is labelled and using intelligent agent who is rewarded when predictions are getting better.

Given work uses supervised learning. During learning different states like watching TV are simulated, state is selected from the list and program stores data from the IRMT sensor with the correct label. Later stored data with all different labels is used when creating the machine learning model. First the model is created and used on a PC and then using the same input data MCU versions are created. Input data is also analyzed and visualized using different python based frameworks.

4.1 Monte Carlo method

Monte Carlo methods are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results [7]. Focus is on using randomness for getting deterministic result. Author has created sleep anomalies detection algorithm (Figure [16]) inspired by the Monte Carlo methods.

4.2 ML.NET

Open-source machine learning framework, which is meant for .NET developers, using either C# or F# programming language. It is installed as an add-on for the Visual Studio [24]. Using of ML.NET is made relatively simple, with one mouse-click wizard is opened which guides the user through the machine learning model learning process. The ML model is selected by the framework automatically, user only needs the input data and can select the maximum time for training. Learning can be done on the user PC, but there is also option to delegate this to the cloud. At the time of this writing cloud computing option was only available for *Image classification*. For given work *Text classification* scenario was chosen, because this enables to predict same type of input data-set into 2+ categories. Given framework was used because of authors experience with the .NET development and its easy user interface.

4.3 Python frameworks: scikit-learn, pandas, matplotlib ja numpy

List of Python frameworks used in given thesis

- scikit-learn [25] open-source ML framework in Python.
- pandas [26] data analysis tool in python.
- matplotlib [27] visualization tool in Python.
- numpy [28] tool written in C to be used for scientific calculations in python.

4.3.1 Machine learning algorithms

Following machine learning algorithms were used using Python when analyzing input data. They were selected to compare the accuracy of different algorithms.

- Naive Bayes classifier [29] - is statistics based one of the simplest machine learning methods. Precondition is that input data vector values are independent and with equal scale. Given work uses Naive Bayes with normal distribution, which is defined by given function 4.1

$$p(x = v|C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(v-\mu_k)^2}{2\sigma_k^2}} \tag{4.1}$$

  x - continuous attribute
  μ_k - mean of the values in x
  σ_k^2 - Bessel corrected variance

v - observed value

C_k - x associated class

π - Pii

e - Euler's number

- Random Forest classifier [30] - multiple decision trees are used and prediction which get the most votes is chosen. Preconditions for the model are:
  1. Inputs need to have a correlation with the output.
  2. Multiple independent decision trees can be made from the inputs.
- Logistic regression - Generalization of linear regression. Uses logistic model. [31].

4.4 Tensorflow Keras

Keras is a neural network library in Tensorflow which is an open source library for machine learning. Author used it to create machine learning model which can be converted to *Tensorflow Lite*, which in turn can be realised in C code and then run on a MCU.

4.5 NanoEdge AI Studio

Tool to create edge AI solutions for STM32 devices [32]. It is an automated machine learning solution, meaning that no coding is required. It uses the sensors e.g. accelerometer present in the STM32. Author used it to create machine learning model for STM32 device.

5. Application software

One of the main end products from the thesis work is a Windows program called Activity Monitor which reads input data from the IRMT sensor and makes live predictions based on a machine learning model. Input data needed for the machine learning model is also created using the program to read data from the IRMT sensor during the learning phase. Activity Monitor system overview (Figure [4])

5.1 Used devices

2 x Omron D6T-44L	Two 4x4 IRMT sensors, which create 4x8 thermal image of the room [33]. Sensors are equipped with I2C interface
TM4C123GXL	IRMT sensors are connected to TM4C123G LaunchPad Evaluation Kit with TI MCU
Windows PC	Used for developing and running Activity Monitor which is connected to the MCU via virtual COM port
Garmin Vivomove	Sleep and activity monitor for getting sleep data

IRMT sensors and the TM4C123GXL were provided by the supervisor and are used as an input device.

5.1.1 Omron D6T-44L sensor

Two Omron D6T-44L IRMT sensors are used, which are described in the product homepage [33] as being good for human presence detection (Figure [5]).

5.2 Used Software

Visual Studio 2019 Community	Windows application development in C#
ML.NET	Microsofti open-source machine learning framework, which is used to train the ML model and in real time predict the output
CCS Code Composer Studio	C language MCU code development environment

The Jupyter Notebook	Open-source programming environment which supports Python
Live Home 3D	Home and interior design application, which is used to create model of the room
Microsoft Visio	Program for diagrams

5.3 Room plan

To test method author uses this room (Figure [6]) for experiments. With purple is the placement of the IRMT sensor shown.

Figure [7] is view from the first and in Figure [8] second IRMT sensor (Sensors, with view of 45 degrees, are in 22,5 degree angle to each other, for the optimal view of the room)

5.4 Activity Monitor GUI and menu structure

Figure [9] shows the program main user interface.

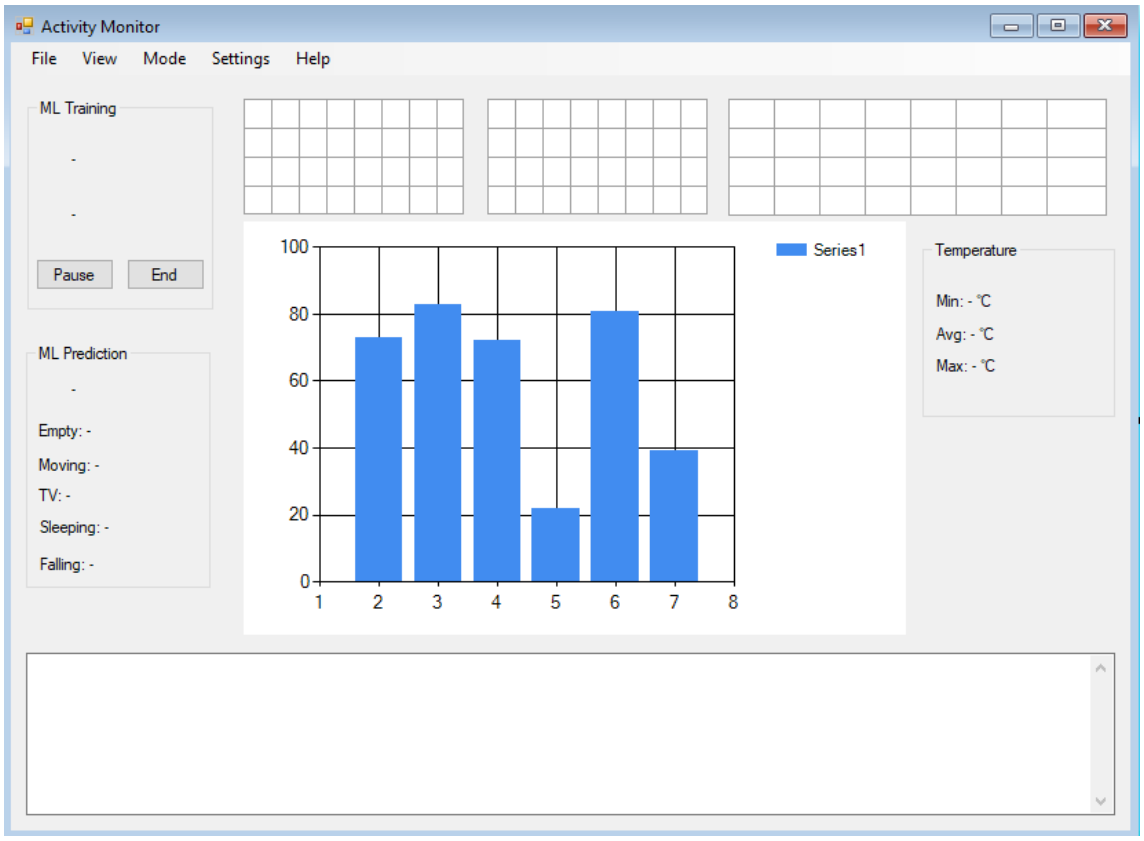

Figure 9. Activity Monitor GUI

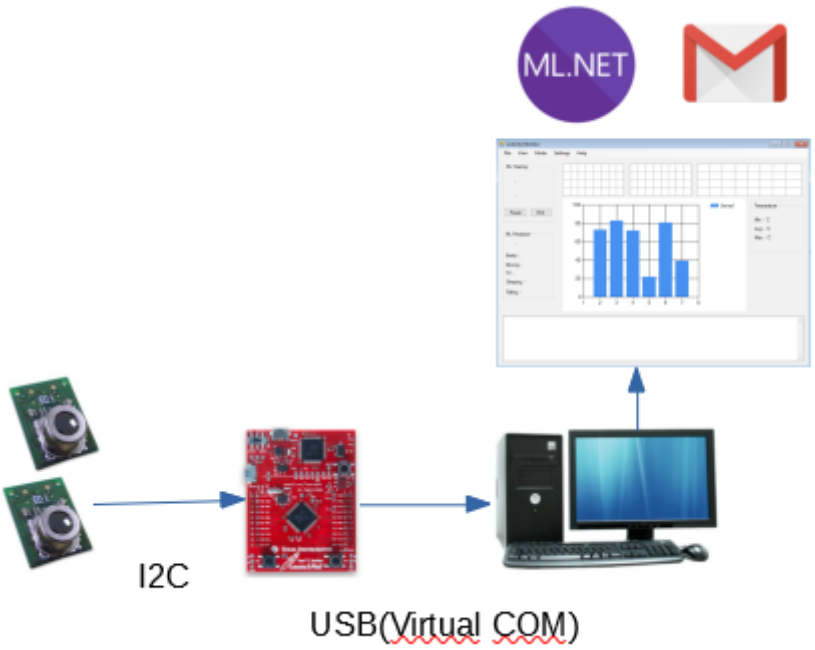

Figure 4. System overview

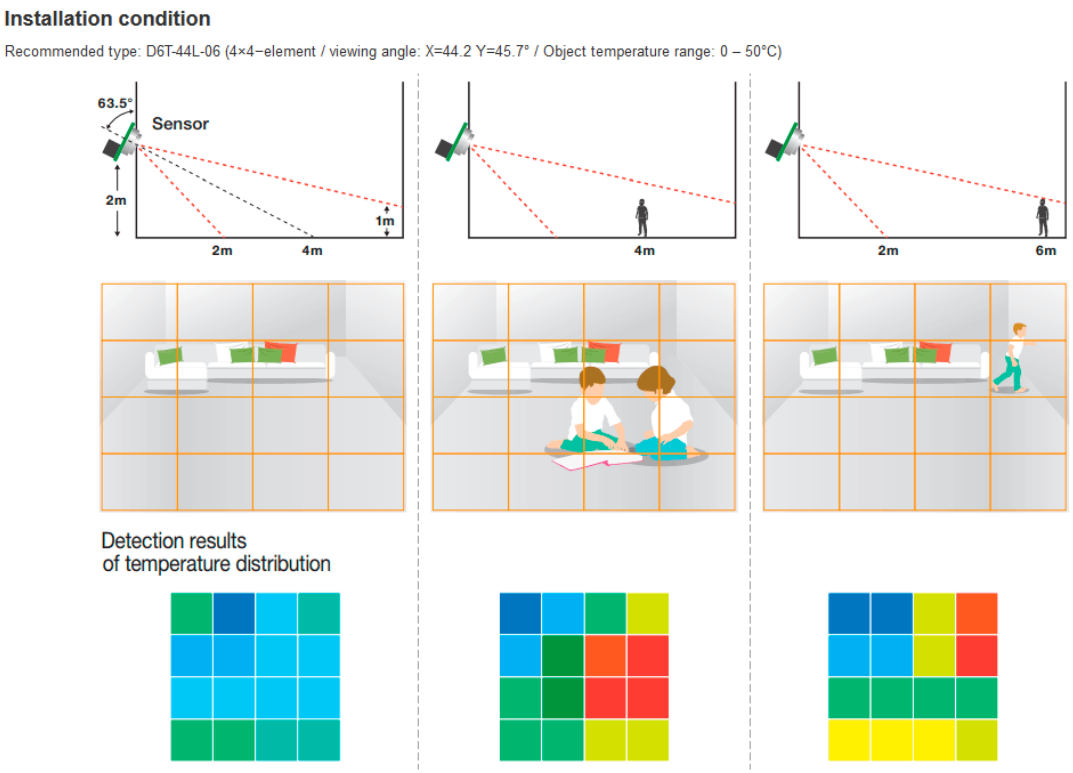

Figure 5. Sensor introduction in the omron.com [33]

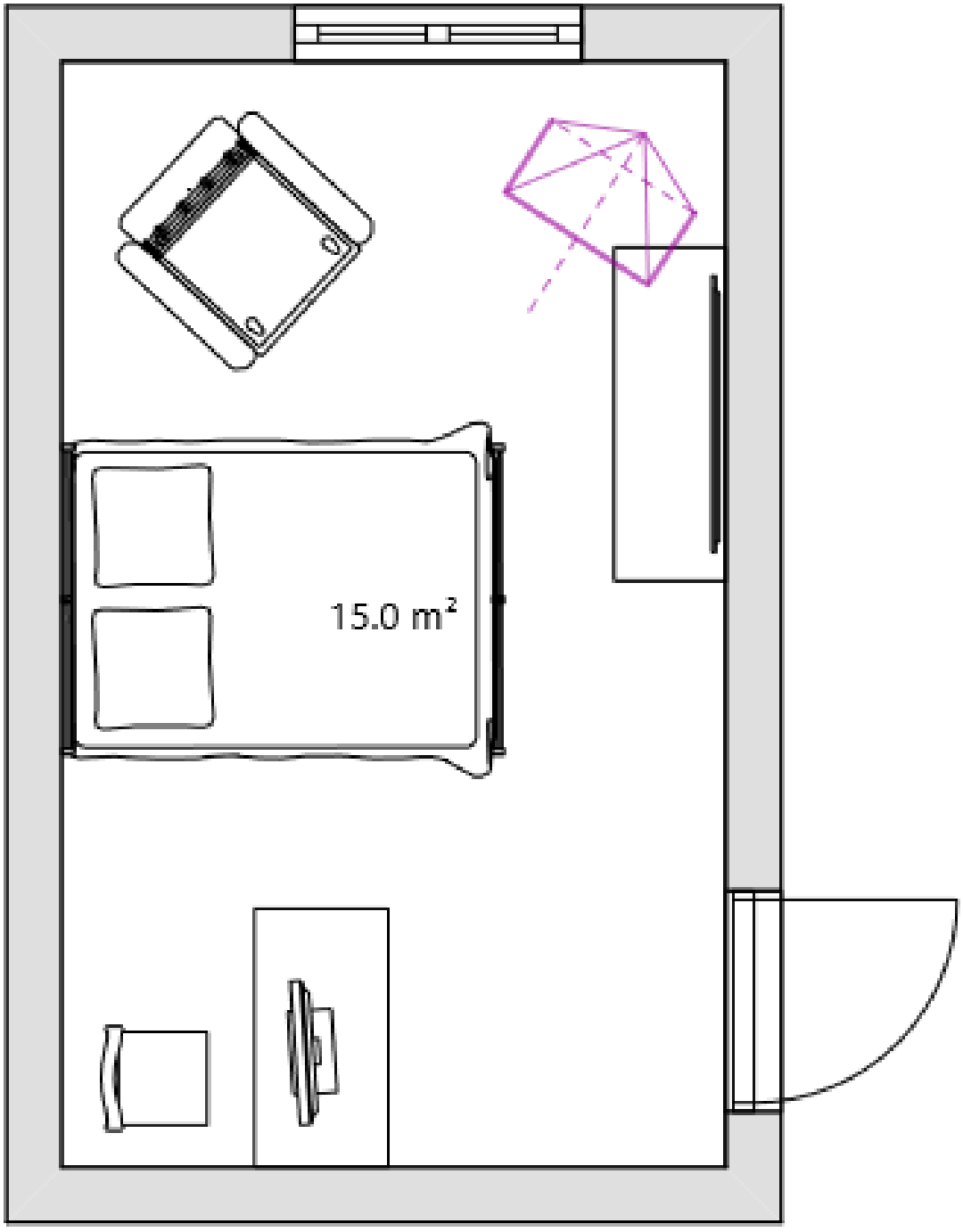

Figure 6. Room plan

Figure 7. View from IRMT sensor 1

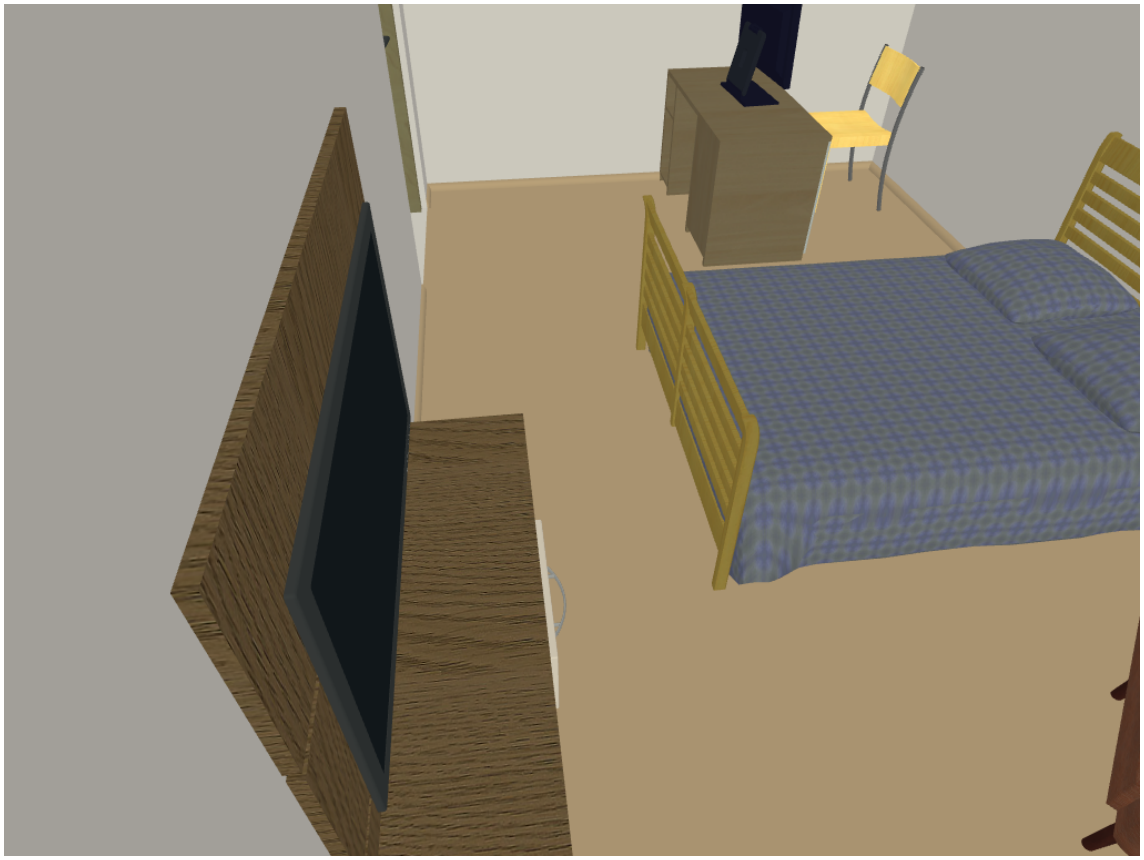

Figure 8. View from IRMT sensor 2

In the following the menu structure of the program is described

- File
    - Start - Open the connection to the IRMT sensor over Virtual COM port
    - Save log - Saves log to a file
    - Exit - Closes connection and closes the program
- View - Views
    - Log - Program log
    - Raw input - Direct input from the sensor
    - Input - Sensor input in degrees
    - Filtered Input - Filtered sensor input in degrees
    - Movement row - Input change in 1 second interval
    - Input table - Sensor input visualized as a table
        - Black and White -Above and below 25 °C
        - Colored - Colorized table
        - Numbers - Number values in degrees in a table
    - Sleep graph - Sleep graph
    - Temperature - Min, Max and average room temperature
    - ML Prediction - Machine learning prediction
- Mode
    - ML Learning - Machine learning input data
        - Empty room
        - Person moving
        - Person watching TV
        - Person sleeping
        - Falling - Fall simulation
    - Sleep
        - Garmin data - Analyze json input from Garmin activity monitor.
        - Live monitoring - Live sleep monitoring from IRMT sensor
    - Test - Program test and helper functions
        - Convert temp - Temperature conversion
        - Prediction - Test machine learning prediction from the input file
        - Room temp - Find room temperature from the input file
        - Calibrate room temp
- Settings
    - General
    - Reset to default
- Help
    - About

5.5 Reading data

Omron IRMT sensor output is read via I2C with MCU and from there is sent over UART to Windows PC, which is input for the Activity Monitor application. Since there are two sensors, observing adjacent spaces, two lines of data are needed for one input image.

Example input:
I2C1: 12 01 e3 00 e4 00 e4 00 e5 00 e6 00 e5 00 e5 00 e4 00 e5 00 e6 00 e4 00 e5 00 e5 00 e5 00 e4 00 e1 00 37
I2C0: 13 01 e7 00 e7 00 e5 00 e3 00 e7 00 e8 00 e7 00 e4 00 e9 00 e9 00 e7 00 e4 00 ec 00 ea 00 e5 00 e5 00 cc
I2C1 is the left sensor and the I2C0 the right followed by high-byte low-byte value pairs of temperature values in decidegree Celcius (d°C).

5.5.1 Pre-processing data

It is a known difficulty in machine learning to get the input data in correct format before inputting it to the ML model. Most of the input values are around room temperature which was about 23 °C. But since the room temperature can change depending on the weather and heating, author observed with initial tests that ML model which worked fine one day was useless on another. To filter out room temperature changes, input data is filtered by deducting the minimum temperature value from each value from the thermal image.

To be able to detect movement from the input data and to satisfy the Markov property [6], author collected multiple of images (4 in one experiment, but this can be configured in the Activity Monitor application) and then set them all together as one data line for the input of the ML. Making the each new frame input to the rolling window, where the oldest is thrown away. Multiple images are needed for detecting movement and for better accuracy. More about it in the experiments (Chapter 8.1.2)

5.5.2 ML model training

Reading input data

To save and label input data for the machine learning model, Activity Monitor has the following options:

- 0 - Room is empty
- 1 - Person moving around

- 2 - Person watching TV (expressing long set of behaviors in sitting position)
- 3 - Person sleeping
- 4 - Person falling

To simplify the learning other heat emitting objects e.g. radiator, pet or multiple people, were not allowed in the room.

To start learning user selects from the menu *ML Learning* type of the learning e.g. *Empty room*. Program gives 3 seconds delay time to create needed situation. (Figure [10]). Learning can be paused or ended. After all the interested states are learned, the program creates a formatted text file, which can be used for ML model training input data.

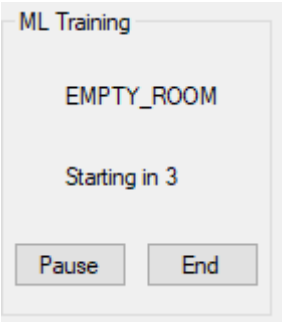

Figure 10. Starting of learning

Training the ML model

To train the model author used Visual Studio wizard for ML.NET. In first step type of the model is chosen (Figure [11]). For given task *Text classification* suited the best.

Next step is environment selection (Figure [12]). For given task only local PC can be selected but e.g. for *Image classification* cloud computing is also an option.

To continue training data needs to be selected. For that text file, which was created by the Activity Monitor, is chosen. First column shows prediction. (Figure [13]).

Maximum time for training can be set in seconds (Figure [14]). Suitable maximum time 30 seconds was found experimentally, which was enough for given size of input data. Training result for given input data was 100% and it was calculated in 12 seconds (Figure [15]).

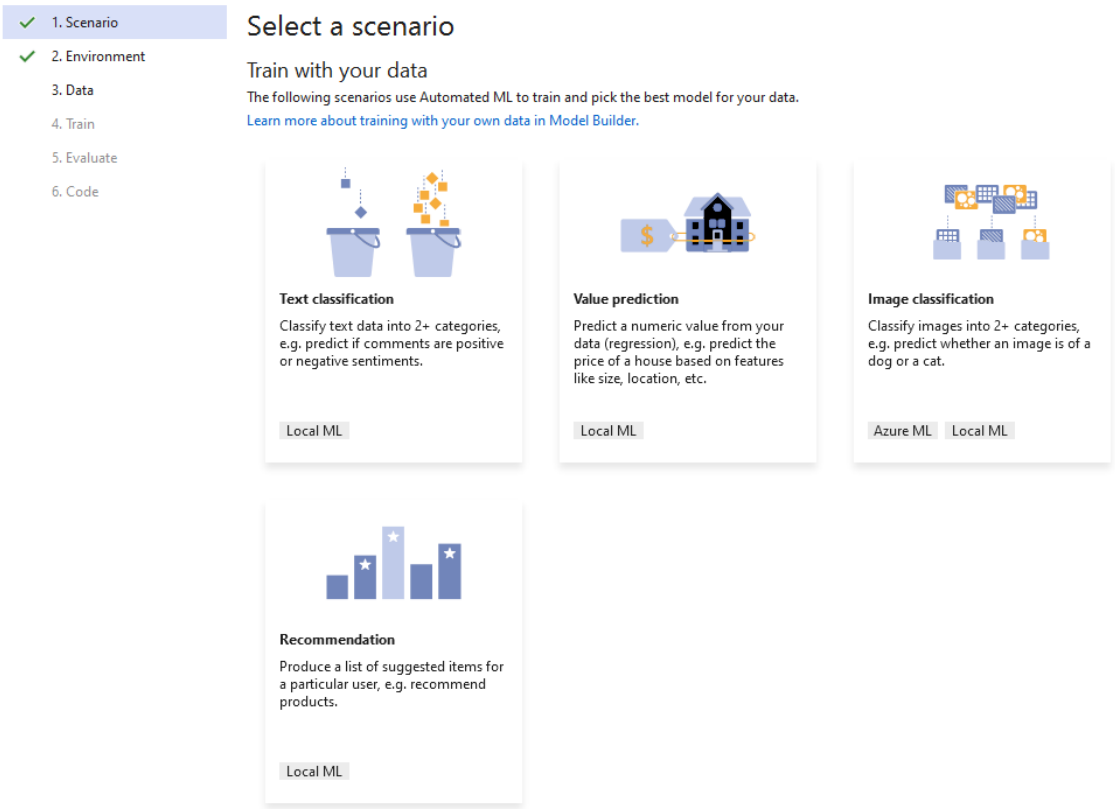

Figure 11. ML.NET Scenario selection

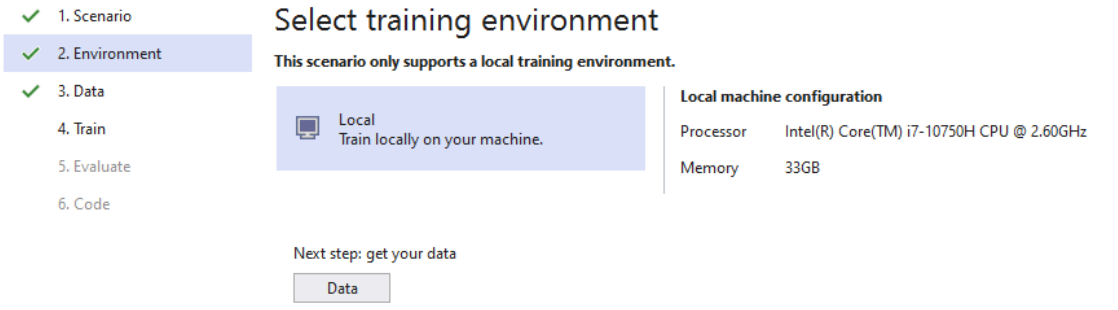

Figure 12. ML.NET training environment

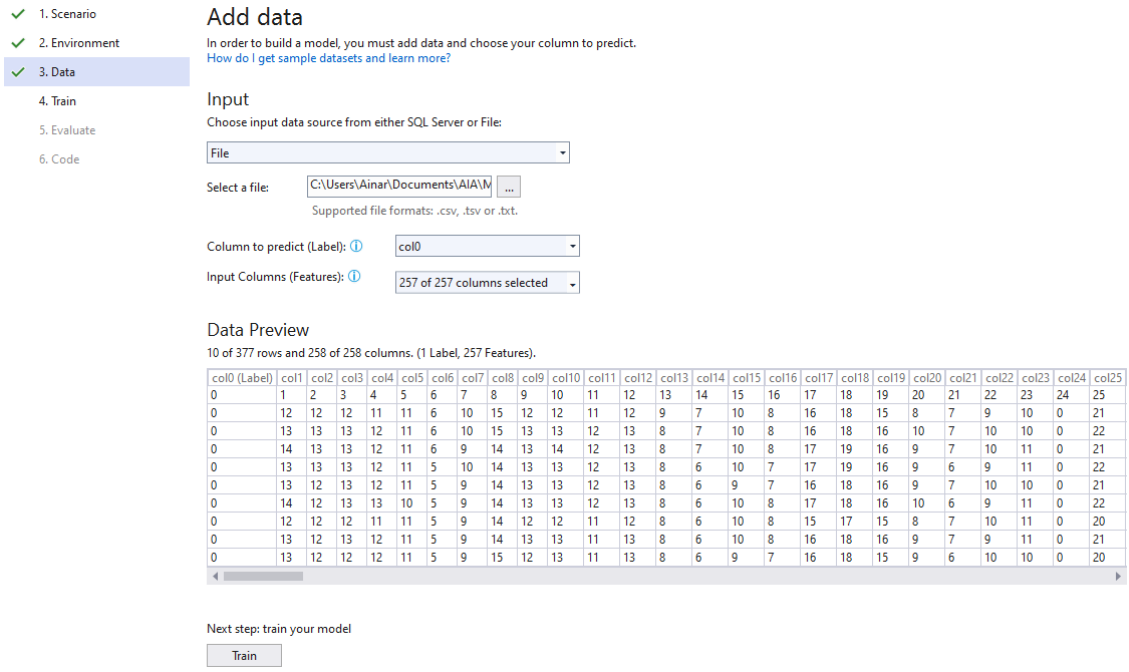

Figure 13. ML.NET input data select

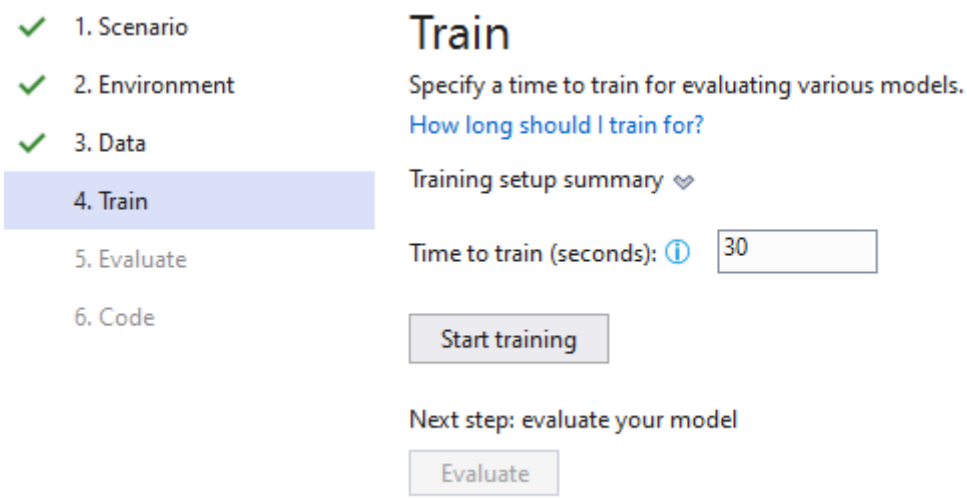

Figure 14. ML.NET training time

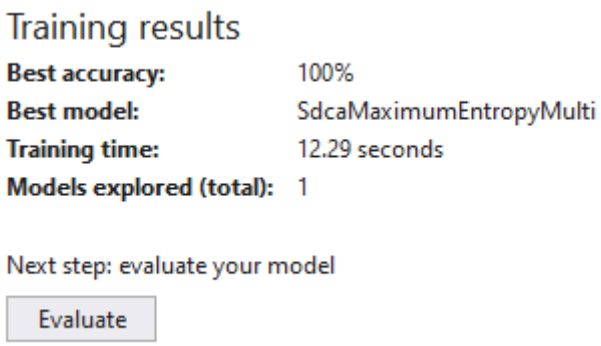

Figure 15. ML.NET training result

5.6 Detecting anomalies

Author investigated two different anomalies i.e. falling and sleep anomalies. Fall detection is based on direct input from the IRMT sensor to the ML model. For sleep anomalies first sleeping or not sleeping is detected from the IRMT sensor and then based on a longer period anomaly is detected as difference from expected behaviour.

5.6.1 Fall detection

Detecting person falling was more complicated than the other human activity detection. Reason is harder to get the proper training data, when compared to e.g. watching TV.

To start author made experiments, where author moved around in the room and made sudden drops to the floor. Then author manually investigated the saved data file and selected data lines which had warm object moving from upper row to lower one. This data was easily distinguishable by the ML.NET machine learning. Model accuracy was 100%. Author wanted to improve the Activity Monitor and added extra function which detects warm object moving down, so when user trains falling these data lines are stored automatically. No need for manually selecting data lines.

```
/// <summary>
/// Detect if the movement row has warmer values in the lower row
/// </summary>
/// <param name="mRow">Movement vector</param>
/// <param name="threshold">Value which is considered as threshold for human moving</param>
/// <returns></returns>
public static bool DetectMovingDown(int[] mRow, int threshold)
{
  if (mRow == null)
  {
    return false;
  }

  for (int i = 1; i < SENSOR_COLUMS; i++)
  {
    for (int j = 0; j < (SENSOR_COLUMS * NOF_SENSORS); j++)
    {
      if (mRow[i * 8 + j] - mRow[((i - 1) * 8) + j] > threshold && mRow[((i - 1) * 8) + j] < -1)
```

```
      {
        return true;
      }
    }
  }
  return false;
}
```

Raising fall alarm and cancelling the alarm

When the falling is detected during program operational mode, then it is not yet known if the person can get up on its own or was it just incorrect interpretation of sensor data (false alarm). To reduce false alarms author added a 10 second timeout, and only if during that 10 second no new state e.g. movement around in the room is detected, only then alarm is raised. Activity Monitor has a support for contact person, who gets notification when fall event is detected. User has to first configure both the sender and the receiver e-mail.

5.6.2 Sleep anomaly detection

Using the machine learning model created in the Activity Monitor, we know if a person is either sleeping or not sleeping. This info is the input to the sleep anomalies detection algorithm, which also runs on the Activity Monitor

Algorithm description

Author made a sleep anomalies detection algorithm which is inspired by the Monte Carlo methods (Figure [16]).

With the Activity Monitor it is possible to detect if a person is sleeping or not. As a next step author investigated sleep anomalies. Garmin Vivomove activity monitor [34] was used to get the sleep data. Author had used given monitor for longer period and this way actual data from one year can be used. Data can be loaded in JSON format where one file is for one day. In each file there is a sleep starting minute and then sleep level for every minute. Since author made a sleep algorithm interval one minute in the Activity Monitor, it was possible to implement additional functionality in the program to accept data in JSON format for sleep anomalies detection algorithm. Given graph [17] shows algorithm result for first 10 nights of year 2019.

Time axis has minutes (1440) in twenty-four hours, and in other axis the expectation strength if a person is sleeping. Graph is made so that in every minute it is evaluated if person is sleeping. This is done by assessing if more than half of the *is sleeping* values within one minute from the machine learning model are true. Program holds data for every minute of the day. After every new minute passing the value for the given minute

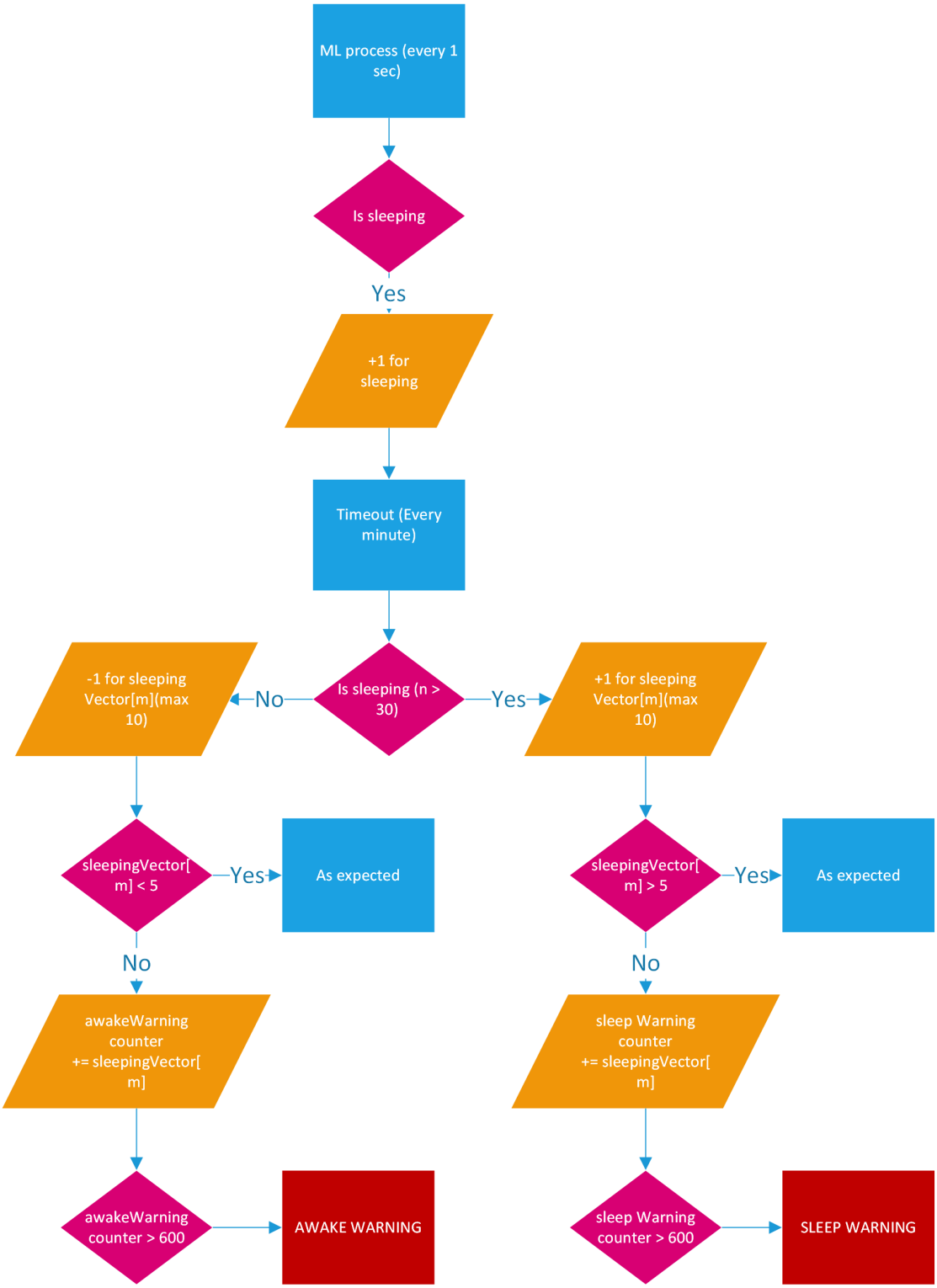

Figure 16. Sleep/Awake anomaly detection algorithm

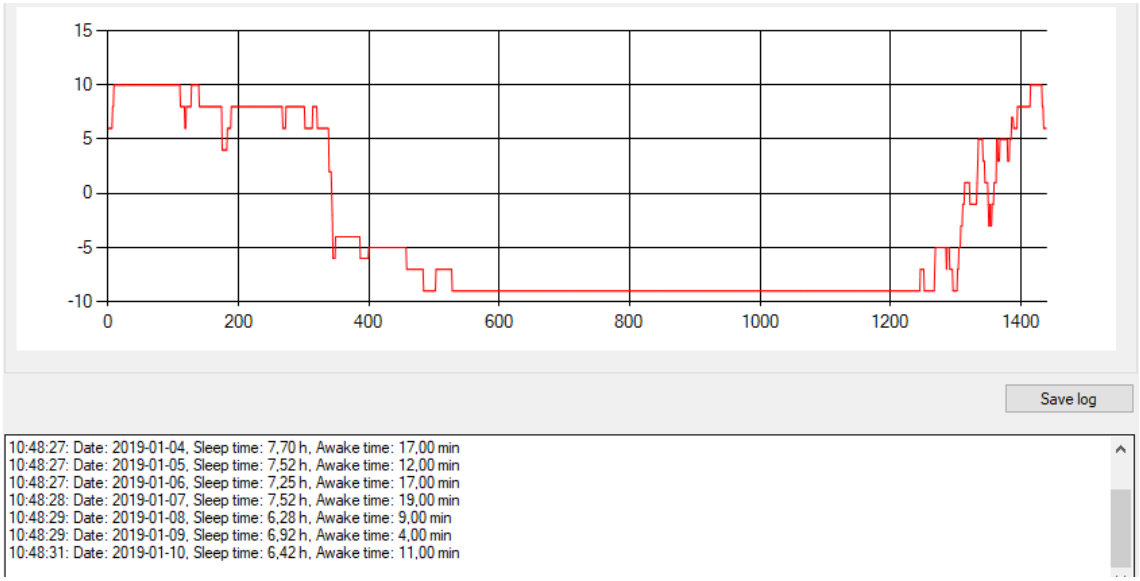

Figure 17. First 10 nights in 2019, red line showing when author was sleeping during 24h, in the lower window program log

is updated. If the person is sleeping value is added by 1 and if not subtracted by 1. If the value reaches maximum (+-10 by default) then it is not updated anymore.
To detect sleep anomaly, every minute program checks the expectation if a person is sleeping (value above 5) or not sleeping (below 5) matches the current value, if not the current value is added to the sum value for given anomaly. If the sum reaches configured maximum (default is 600, which represents 1 hour of maximum values) anomaly is detected. The sum value for either *is sleeping* or *not sleeping* anomaly is reset every time the expected value matches current value.
In the following table [4] there is description of the constants used in the algorithm and what effect they have.

Calibrating parameters

Author tried to detect anomalies (in period 01.2019-07.2020) with assumption that 1% of days could be considered as anomaly. 1% was chosen because then there would be about 3-4 days in a year where the warning is raised. Although the author had rather regular sleeping behaviour, these days should indicate something author remembers was outside of normal sleep/awake days. Author either slept when should not or did not when should have. Preliminary parameters:

```
const int AWAKE_WARNING = 600;
const int SLEEP_WARNING = 600;
const int MAX_VAL_FOR_MINUTE = 10;
```

Table 4. Constants used in the sleep anomalies algorithm

Name	Info	Effect
AWAKE_WARNING	Limit for awake warning	Limit, when alarm is given when person is awake when should be asleep
SLEEP_WARNING	Limit for sleep warning	Limit, when alarm is given when person is asleep when should be awake
MAX_VAL_FOR MINUTE	Maximum value for any given minute	Increasing the value, it takes more time for algoritm to get the max values, but it contains more previous is asleep values.
SLEEP_EXP THRESHOLD	Time when it is expected for the person to sleep	Increasing the value makes the exact time to go sleep less relevant
AWAKE_EXP THRESHOLD	Time when it is expected for the person to be awake	Increasing the value makes the exact time to be awake less relevant

```
const int SLEEP_EXP_THRESHOLD = 5;
const int AWAKE_EXP_THRESHOLD = 5;
```

Calibrating parameters more under experiments 8.3

5.7 Testing the program

To test the program, manual testing and unit-testing was used.

5.7.1 Manual testing

Creating the program and after adding new features, it was manually tested with different states and inputs. Also special testing options added to the program e.g. predict data from saved file input. So instead of testing that e.g. the prediction for watching TV is still working author ran previously saved test file to see how the program behaves.

5.7.2 Unit-testing

To make the testing more efficient especially against regressions, that old features does not need to be manually re-tested after every update, author added unit tests, which test smaller components. After every code change they can be run automatically. *MSTest.TestFramework* was used for unit-testing.

6. Micro-controller software

When the experiments with IRMT sensor and machine learning model running on the PC proved successful, author investigated running the model also on a micro-controller. For that author used single board controller with STM32F429 MCU [35].

6.1 Used devices

Windows PC	Used for developing the ML model and code for the MCU.
STM32F429	MCU used to implement machine learning model on the edge device. It has Cortex-M4 core clocking frequency at 180 MHz, 192 KiB RAM and 2.00 MiB flash.

6.2 Used Software

STM32CubeMX	Windows application for graphical configuration of the STM32 microcontrollers (Figure [18])
X-CUBE-AI	Artificial Intelligence Pack version 5.2.0 for STM32 MCUs
NanoEdgeAIStudio	Automated Machine Learning (ML) tool for STM32 developers
System workbench for STM32	Eclipse like IDE for STM32
Tensorflow Keras	Deep learning API written in Python
Tensorfloe Lite	Open source deep learning framework for microcontrollers

6.3 ML model creation with X-CUBE-AI

Author took the same input data file which was created during the learning process with the Activity Monitor. Then author used Tensorflow Keras framework in Python to create a ML model. Author had the limitations of the MCU to consider. This meant that the maximum RAM is 192 KiB and maximum flash size is 2.0 MiB (which is not that problematic as the RAM run out first)

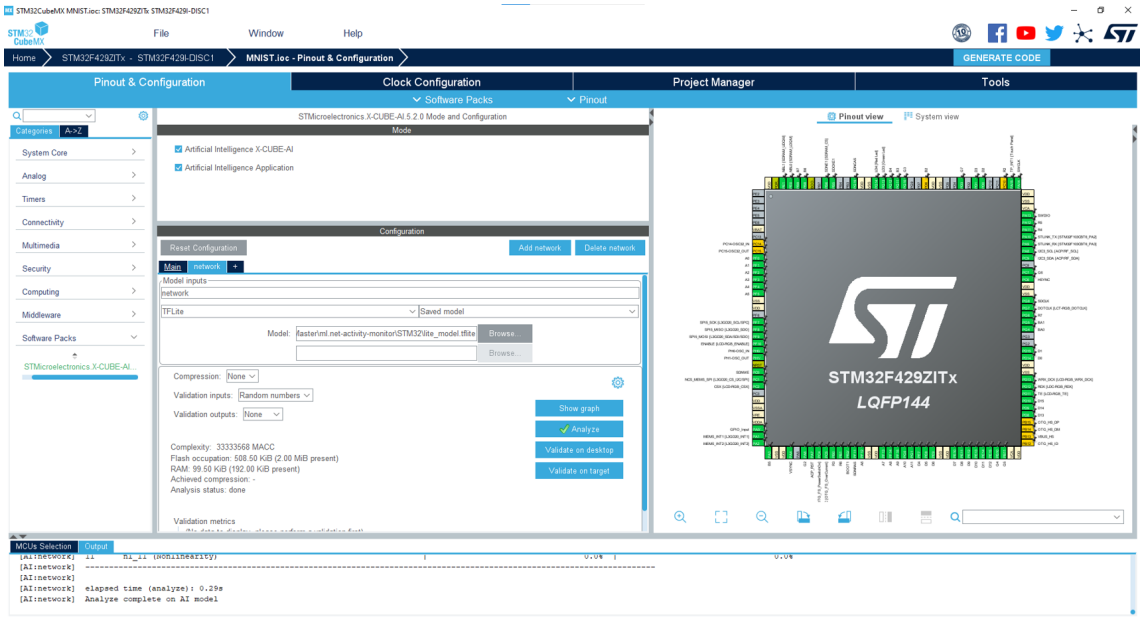

Figure 18. STM32CubeMX configuration view

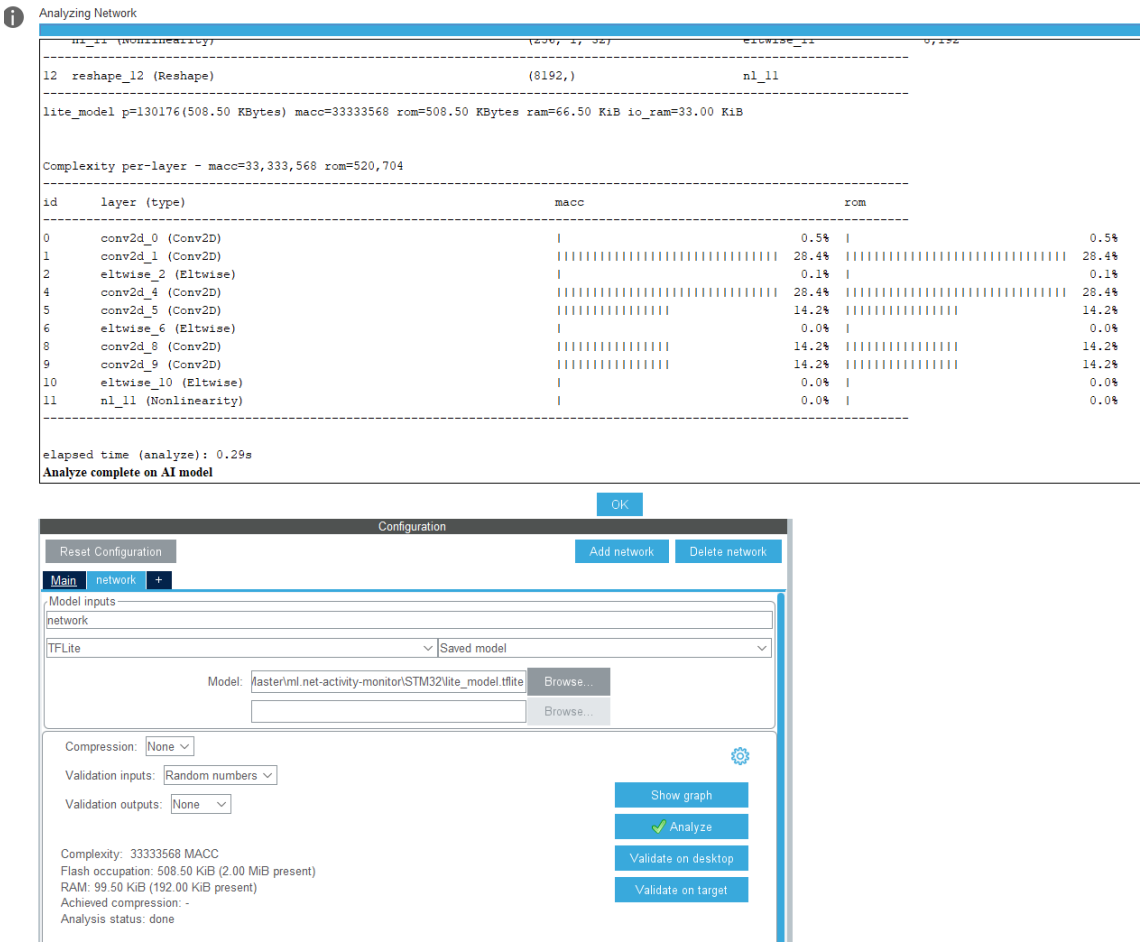

Figure 19. STM32CubeMX X-CUBE-AI plugin

Steps to create model for STM

- Create and train using Tensorflow Keras
- Convert model to Tensorflow Lite
- Analyze model using X-CUBE-AI in STM32CubeMX (Figure [19])
- Generate C code from the ML model in STM32CubeMX
- Use generated ML model using System workbench for STM32

Model created for the the STM32

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 256, 1, 64)        640
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 256, 1, 64)        36928
_________________________________________________________________
batch_normalization (BatchNo (None, 256, 1, 64)        256
_________________________________________________________________
dropout (Dropout)            (None, 256, 1, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 256, 1, 64)        36928
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 256, 1, 32)        18464
_________________________________________________________________
dropout_1 (Dropout)          (None, 256, 1, 32)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 256, 1, 32)        128
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 256, 1, 64)        18496
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 256, 1, 32)        18464
_________________________________________________________________
batch_normalization_2 (Batch (None, 256, 1, 32)        128
_________________________________________________________________
activation (Activation)      (None, 256, 1, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 8192)              0
=================================================================
Total params: 130,432
Trainable params: 130,176
Non-trainable params: 256
Validation evaluation results: loss -  0.434 accuracy -  0.936
```

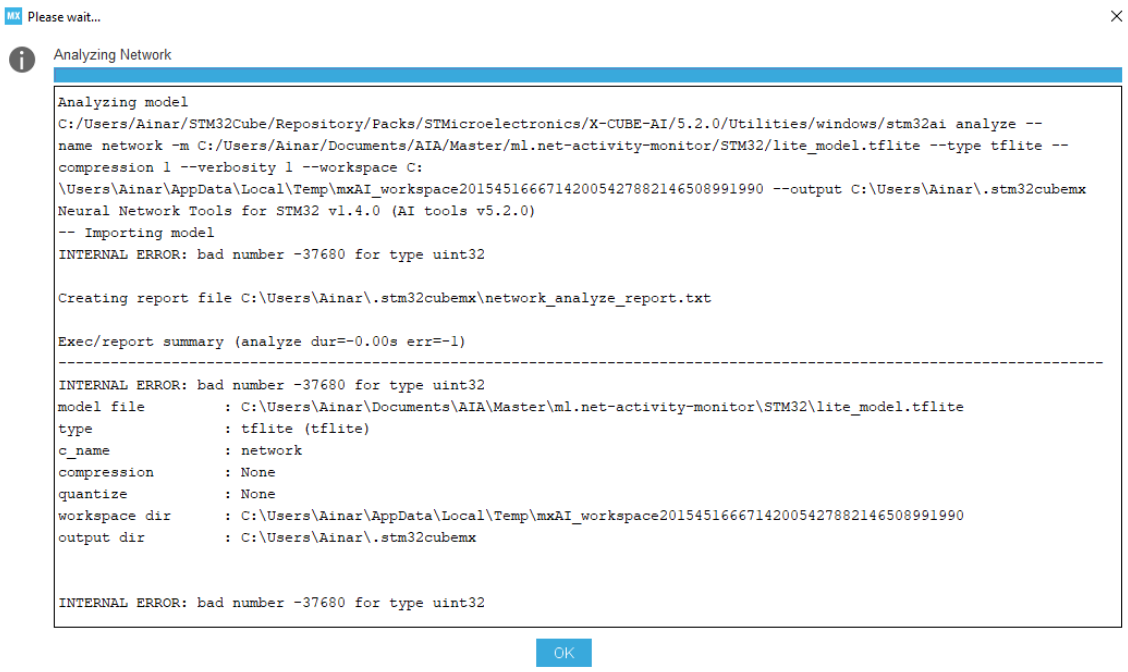

Figure 20. STM32CubeMX X-CUBE-AI analyze error

Model creation is an iterative process, reason is that resulting model might not fit to the MCU, but also not every layer/combination is supported by the X-CUBE-AI. Example of an error caused by connecting two Dense layers [36] (Figure [20])

6.4 STM32 program with ML

After model is successfully created and also the analyze from STM32CubeMX is successful, then the ML model C code can be created. STM32 project to receive the C code has to exist first. Author made one based on a previous school project. The generated C code has the main interface put into app_x-cube-ai.h file and the ML model goes by default to network.c file.

Author added 5 random data samples from initial input file for ML training to the C code. Then programmed GUI code to visualize the thermal image and the ML prediction. Resulting STM32 program runs through the inputs one by one and shows the result on the LCD screen (Figure [21]).

6.4.1 Running the STM32 program

Video about the code running on the MCU through all the input samples is uploaded along with the source code to GitHub (Chapter 10).
Like mentioned before author hard-coded 5 test samples to the STM32 code to verify the

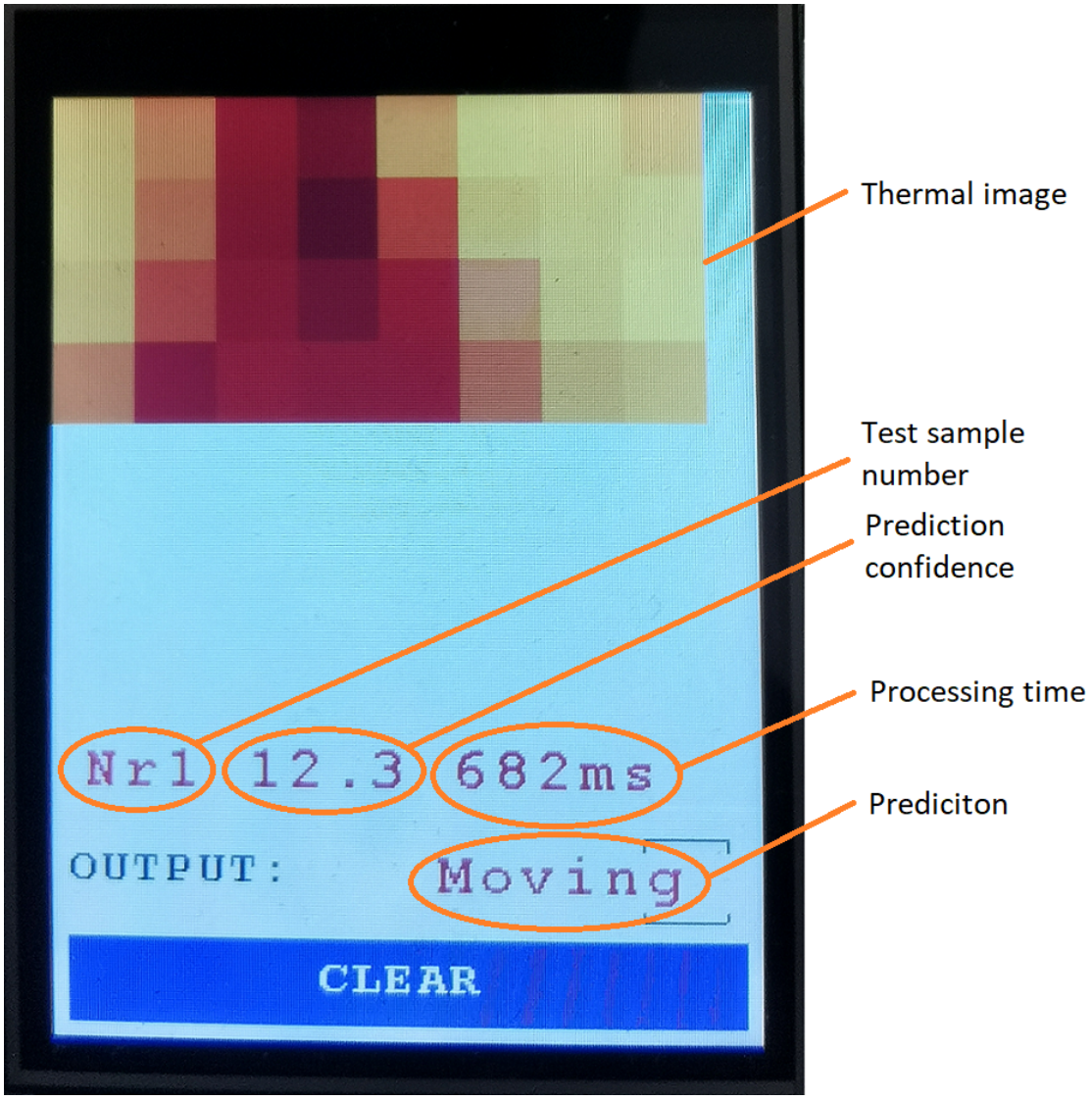

Figure 21. STM32 program GUI

accuracy and the speed of the ML model.

Processing time

The processing time for the ML model was 682 milliseconds for all inputs. Author did not see any time fluctuation.

Accuracy

When author run through all the test samples, it was noticed that all but prediction for the falling were correct. To see how much the prediction was wrong, author modified the GUI code so that it printed the prediction confidence for the highest and also for the expected value (Figure [22]). This showed that the difference was not over 5.6 for regular moving around in the room when it was 5.1 for falling. Author tried couple of other samples for falling, they all were predicted wrongly, but the error was not big. This suggests that adding additional code checks could make the fall detection work e.g. if the falling prediction is within 10-20% of regular moving prediction.

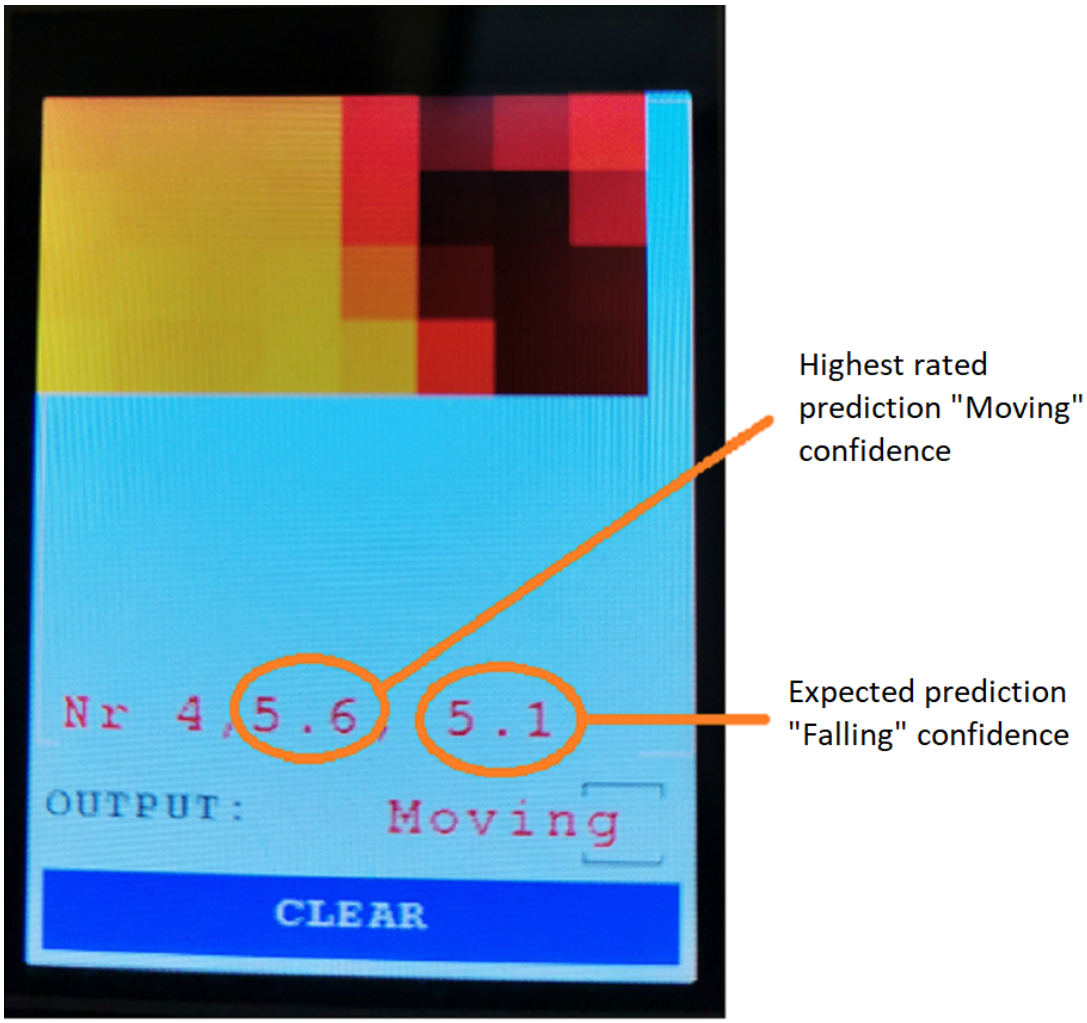

Figure 22. STM32 program prediction for falling

6.5 ML model creation with NanoEdgeAIStudio

Author investigated creation of the model using the NanoEdgeAIStudio, because it creates the model and compiles C code in 5 simple steps using a wizard. A lot fewer steps compared to previous ML model creation for the MCU, where different programs and frameworks had to be used. Here only one program is needed. Unfortunately the STM32F4, which author had, was not supported by the tool at the time of thesis writing. It was grayed out in the target selection combo-box(Figure [23]). Author asked about it from the support e-mail, but did not get any reply.

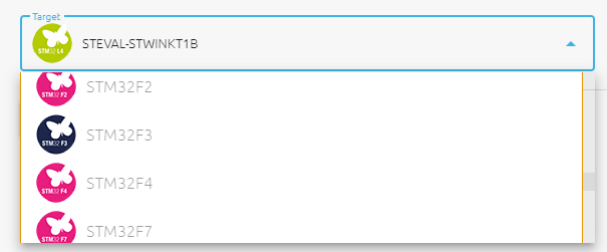

Figure 23. STM32F4 not selectable

Author continued to study the program by selecting target which was supported (STEVAL-STWINKT1B) and a generic sensor. In the next step (Figure [24]) input signals need to be imported. Author modified the test data which was made by the Activity Monitor to be suitable for given program. This meant that all observed predictions had to be in a separate file. Then program analyses input files to see if they are suitable for the model creation, e.g. if they fit into the RAM (with previous model creation, only after the model was created, one could see the required RAM). Also there are input signal visualization and filtering options.

Next step is the creation of the machine learning model (Figure[25]). User only selects how many CPU cores are allocated and the rest is done by the program.

The result of the model creation was 31 different ML libraries, with the best ones having accuracy 88%, memory usage for RAM 1,2 kB and required flash 12,3 kB. User can get more info about each of the resulting model/library (Figure[26]). In the view one can see the type of the model, in given case the support vector machine (SVM) gave the best result. Also confusion matrix is shown to see where the wrong predictions went. Basically all of the ML models failed to successfully detect falling, but were close to 100% on all other predictions.

Confusion matrix had only one prediction of falling, which was wrong, but does not indicate

Figure 24. NanoEdgeAIStudio Signals view

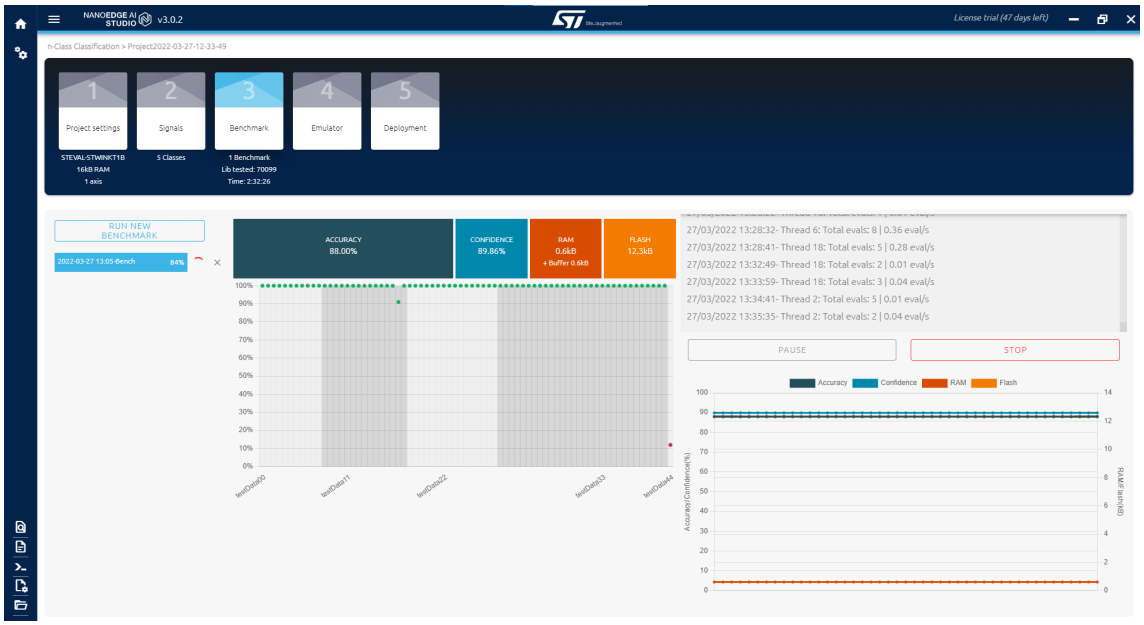

Figure 25. NanoEdgeAIStudio Benchmark view

Figure 26. NanoEdgeAIStudio Benchmark libraries view

that all of the predictions for falling are wrong. Author tried to predict falling using the best library with the emulator, which is the next step in the program wizard (Figure[27]). For test author made test file with 5 data lines, 2 for moving around in the room and 3 for falling. Author modified randomly input data values by increasing/decreasing 3 values in each line but one, which remained identical to the one in the learning phase. This was to not have exact match from training data. The result was 100% accuracy.

Last step in the program is deployment, which compiles the machine learning library to C library. There are few configuration possibilities like compiler flags and option to have multiple libraries deployed to the same STM32 board. The view also includes library integration C code example (Figure[28])

Although the required STM32F4 could not be selected, author found the program very promising. Reason is that it had very few simple steps to create the model from input data ready to be deployed to the MCU board.

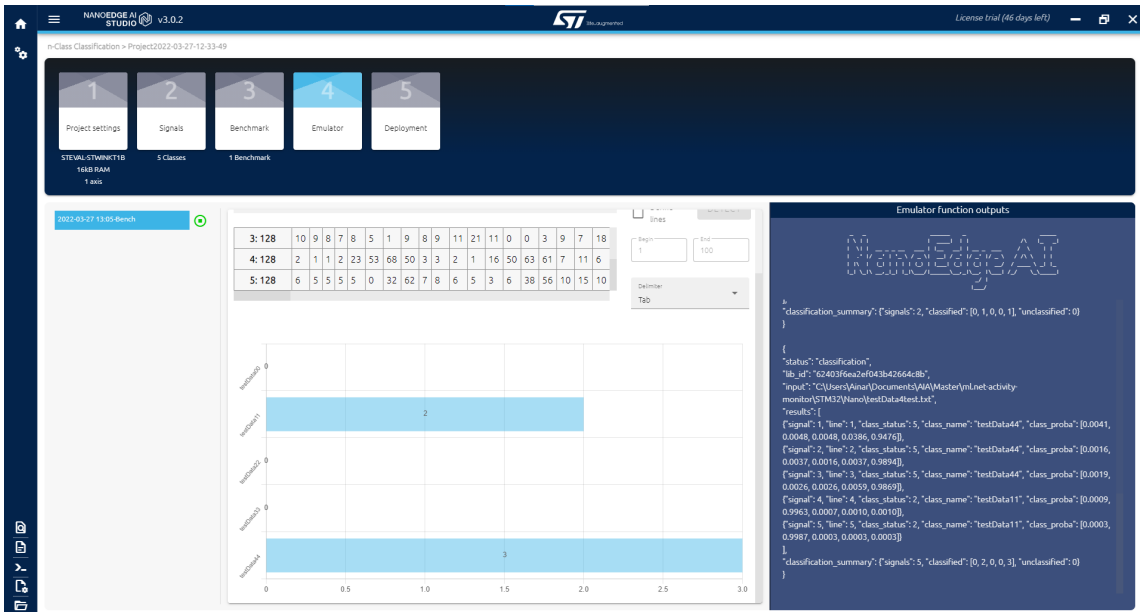

Figure 27. NanoEdgeAIStudio Emulator view

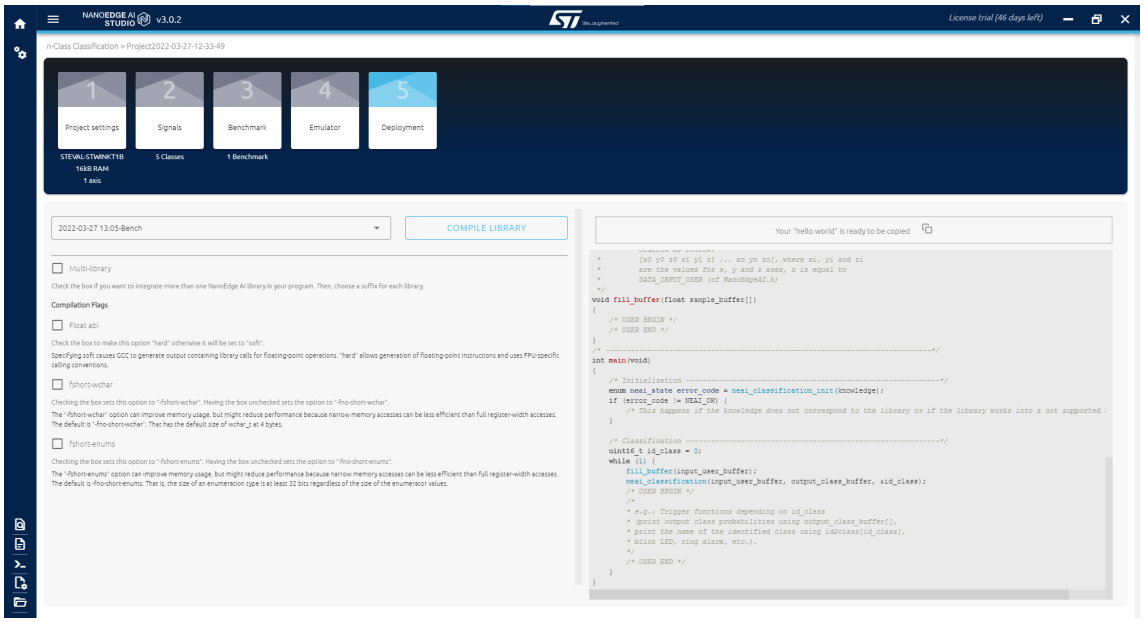

Figure 28. NanoEdgeAIStudio Deployment view

7. Analysis scripts

In the Activity Monitor program, the wizard in the ML.NET selects best machine learning model automatically. Author decided to analyze the input data and test different machine learning models to see their accuracy. Author used Jupyter Notebook and machine learning frameworks in python to analyze input data. Reason for using python is its wide use in the machine learning and data processing. Python is considered the most popular language for ML [37].

7.1 Processing input data

Input data is the values stored using the Activity Monitor with IRMT sensor. To load the data for analysis pandas framework is used.
Example to validate the input data and its structure the following methods were used:

```
#see how many rows and columns
df.shape
#see the first 5 rows
df.head(5)
#see the last 5 rows
df.tail(5)
#check that values are not null
df.isnull().values.any()
#Data correlation shows how each input column is dependent on
#the other, where 1 means 100%
df.corr()
```

7.2 Different machine learning models

Naive Bayes classification

```
from sklearn.naive_bayes import GaussianNB

nb_model = GaussianNB()
nb_model.fit(X_train, y_train.ravel())
```

Random forest classification

```
from sklearn.ensemble import RandomForestClassifier

rf_model = RandomForestClassifier(random_state=42, n_estimators=10)
rf_model.fit(X_train, y_train.ravel())
```

Logistic regression

```
from sklearn.linear_model import LogisticRegression

lr_model = LogisticRegression(C=0.7, solver='liblinear', random_state=4
lr_model.fit(X_train, y_train.ravel())
```

7.2.1 Comparing ML models

To know which ML model gave the best result, author compared following metrics using metrics library from the scikit-learn [38]:

- Accuracy of training and test data
- Confusion Matrix
- Classification Report

Accuracy

To know the model accuracy author compared the training data accuracy and also the test data accuracy. Author used *accuracy_score* method

```
predict_train = model.predict(X_train)
metrics.accuracy_score(y_train, predict_train)

predict_test = model.predict(X_test)
metrics.accuracy_score(y_test, predict_test)
```

Confusion Matrix

Confusion matrix shows the summary of predictions in a matrix. Author used it to see where the wrong predictions went. *confusion_matrix* is part of the sklearn.metrics

```
print("Confusion␣Matrix")
print("{0}".format(metrics.confusion_matrix(y_test, predict_test)))
```

Example confusion matrix 7, which shows that one prediction for falling is wrong and is predicted as sleeping:

Table 7. Random forest classifier confusion matrix

Empty	Moving	TV	Sleep	Fall
20	0	0	0	0
0	24	0	0	0
0	0	21	0	0
0	0	0	46	0
0	0	0	1	1

Classification Report

classification_report method shows additional metrics about the ML model predictions:

```
print("Classification Report")
print(metrics.classification_report(y_test, predict_test))
```

- Precision - percentage of positive predictions which are actually correct for given label
- Recall - number of predictions for given label divided by all the predictions for given label
- F1-score - precision and recall combined to one value

8. Experiments

8.1 Input data analysis

Author made experiments with the python scripts described in 7. For input data author used the values stored using the Activity Monitor with IRMT sensor. Author used version where 8 consecutive thermal images were saved to get one row of input data with 256 (8x32) values 5.5.1.

8.1.1 Finding best ML model

Naive Bayes classification

```
Training data accuracy: 0.9430
Test data accuracy: 0.7788
Confusion Matrix
[[ 0 20  0  0  0]
 [ 0 20  0  0  4]
 [ 0  0 21  0  0]
 [ 0  0  0 45  1]
 [ 0  0  0  0  2]]
Classification Report
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        20
           1       0.50      0.83      0.62        24
           2       1.00      1.00      1.00        21
           3       1.00      0.98      0.99        46
           4       0.29      1.00      0.44         2

    accuracy                           0.78       113
   macro avg       0.56      0.76      0.61       113
weighted avg       0.70      0.78      0.73       113
```

Random Forest classification

```
Training data accuracy: 1.0000
Test data accuracy: 0.9912
Confusion Matrix
[[20  0  0  0  0]
 [ 0 24  0  0  0]
 [ 0  0 21  0  0]
 [ 0  0  0 46  0]
 [ 0  0  0  1  1]]
Classification Report
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        20
           1       1.00      1.00      1.00        24
           2       1.00      1.00      1.00        21
           3       0.98      1.00      0.99        46
           4       1.00      0.50      0.67         2

    accuracy                           0.99       113
   macro avg       1.00      0.90      0.93       113
weighted avg       0.99      0.99      0.99       113
```

Logistic Regression

```
Training data accuracy: 1.0000
Test data accuracy: 0.9823
Confusion Matrix
[[20  0  0  0  0]
 [ 0 24  0  0  0]
 [ 0  0 21  0  0]
 [ 0  0  0 46  0]
 [ 0  1  0  1  0]]
Classification Report
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        20
           1       0.96      1.00      0.98        24
           2       1.00      1.00      1.00        21
           3       0.98      1.00      0.99        46
           4       0.00      0.00      0.00         2
```

accuracy			0.98	113
macro avg	0.79	0.80	0.79	113
weighted avg	0.97	0.98	0.97	113

Result

The best result for training data gave Random Forest (RF) and Logistic Regression (LR) with 100% accuracy, but for the test data RF was better 99,1% vs 98,2%. When looking into the wrong predictions, then the RF had one wrong prediction for falling which was predicted as sleeping. For LR both of the predictions for falling went into wrong labels (sleeping and moving).

8.1.2 Analyzing the amount of data needed

Author made an assumption that multiple sequential thermal images are needed in one data line to detect movement. To verify this assumption, author used python scrips. To verify the stability author used Activity Monitor and reduced the number of data rows used for machine learning and then observed prediction probability for each prediction.

Number of images needed for accuracy

When input data from the IRMT sensor using Activity Monitor was collected, the initial setup saved values from 8 thermal images to one data row. Author used this input data and reduced number of images as an input to python scripts to see how much it affects prediction accuracy. The best ML model (Random Forest) from the previous section was used. The result was that almost no effect when reducing the number of input data from 256 to 32 (8 images to 1). The training data accuracy 100% and test data accuracy 99,1% remained the same. Author thinks that the reason for this is due to different locations in the room used for e.g. watching TV or moving in the room. The moving around in the room gave never identical thermal image. In the following chapter 8.1.2 author experiments what happens if moving around can overlap static positions.

Number of images needed for stability

In the previous chapter author observed that multiple thermal images were not needed to detect moving. Author set-up new test, with 4 states: empty room, standing on the left side, standing on the right and moving from left to right. Author modified Activity Monitor to support this. To train the model author first used 4 consecutive thermal images (128 data lines) and then tried the ML model accuracy while testing these different states.
Author made a test where he first stood on the left side, then moved around in the room,

stopped at the right side and then left the room. In the following graph one can see better accuracy and stability from having four thermal images compared to one (Figure [29]) in one data line. Graph shows that standing in either *left* or *right* has close to 100% accuracy on both images but upper one is more stable. With *moving* state there are inaccuracy's in both graphs but again upper one has more correct predictions. And regarding *empty room* the lower graph shows it wrongly as *moving*.

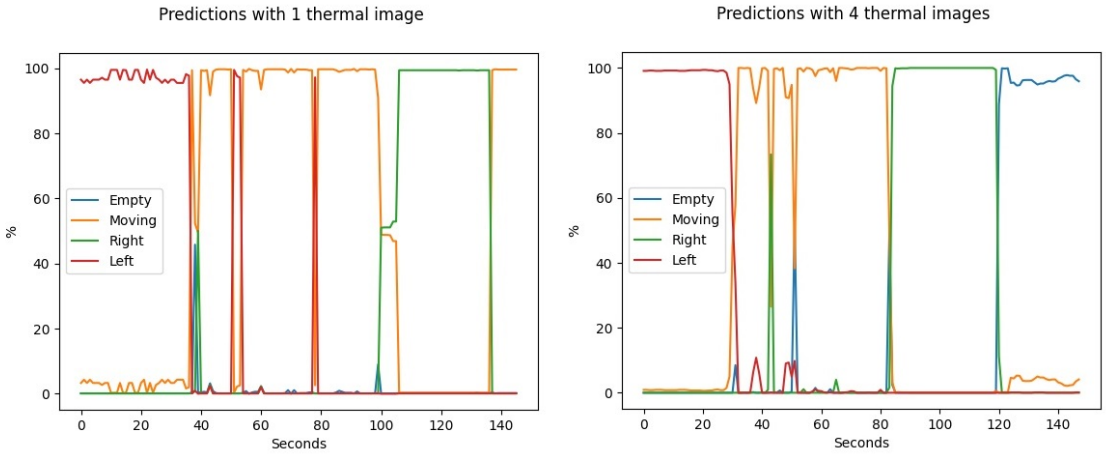

Figure 29. Prediction confidence with 4 thermal images

8.2 Measuring body temperature

Christel Nielsen [1] found in her thesis, that when a person is close (about 10 cm from the sensor) body temperature is measured 35 °C and from 1 meter distance 30 °C. Sensor does not get the infrared only from the person but also from the surrounding environment and the average will be smaller.

Author decided to see if using the IRMT sensor in live monitoring mode could be used for measuring persons temperature. Idea was to measure temperature while a person is watching TV (which is about 1,5 meters from the sensor). To know how much is distance affecting temperature changing in given setup, author measured maximum body temperature from 3 different distances 0,5, 1 and 1,5 meters. Author's body temperature at the time of the test was 36,6 °C. In the table [8] it is possible to see that average change in temperature is 0,45°C per half a meter. Since the change between distances is constant it is reasonable to assume that when persons temperature is changing the resulting measurement is changing. To test persons temperature change author took cup of hot tea and sat on a TV watching position. Maximal average temperature was 32,34 °C. Since the cup of tea is smaller than persons head, but hotter, it is not clear if we could use given setup to measure reliably fewer, but the experiments suggest there could be a possibility, which could be further investigated. Then the system has to take into consideration other heat emitting objects like the actual cup of hot tea.

Table 8. Body temeprature measurement results

Distance	Value	Difference	Results
0,5 m	31,27 °C	0	39
1 m	30,82 °C	0,45 °C	22
1,5 m	30,37 °C	0,45 °C	54

8.3 Calibrating parameters for sleep anomaly detection

Preliminary parameters (5.6.2) in sleep anomaly algorithm (Figure [16]) were chosen without deep analysis and gave this result (Figure [30]). Red line shows if the author is expected to be asleep (value 10) or be awake (value -10) in every minute of a full day.

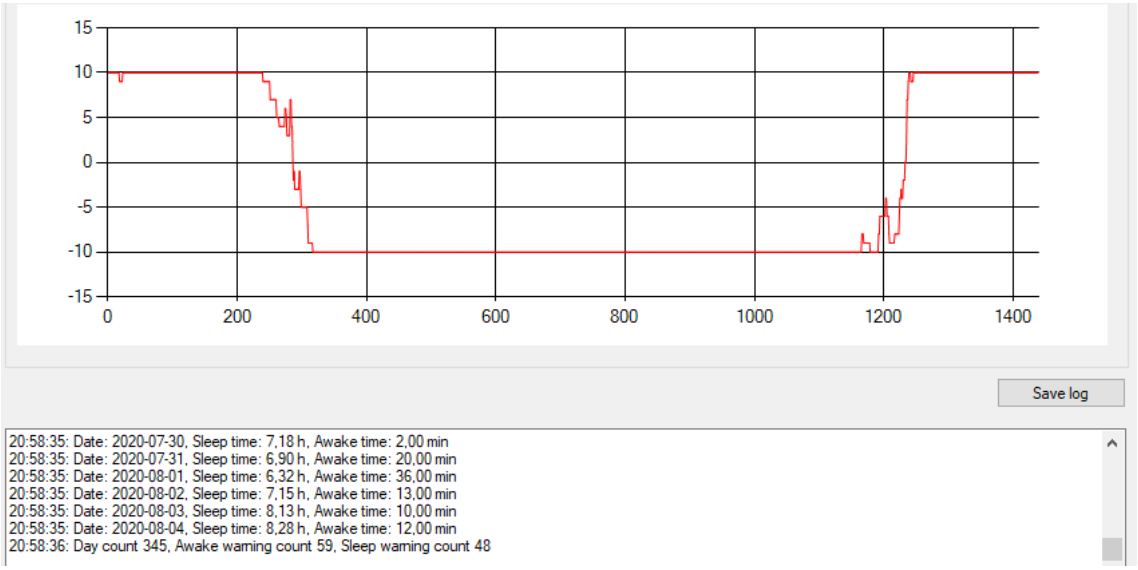

Figure 30. Resulting sleep expectation for one day shown in red after 345 full days simulation

Out of 345 analysed days the awake anomaly was detected 59 times and the sleep anomaly 48 times.For example on 2020-07-22 author woke up half past 4 which made the program raise the awake anomaly.

Figure 31. Awake warnings between 09-12.2019

That many anomalies (Figure [31]) did not seem reasonable to the author, especially because no real anomaly occurred in given period. And by anomaly author means e.g. not sleeping at all during the night or not waking up for many hours after the normal time. Author raised the threshold for raising an alarm from 600 to 1200, which means that 2 full hours of maximum wrong value. After rerunning the algorithm with test data, the result was 14 awake warnings and 6 sleep warnings. Warnings made more sense now e.g. awake warning on 2019-09-04, was caused by awaking up for going to an early flight, and another for author going to a party and went to a sleep more than 3 hours later than normal. Sleep warnings came when author slept longer than usual in the weekends e.g. on 2020-03-28 author slept until 10 a clock in the morning. The last does not seem like a real warning and then author increased only the sleep warning to 1800, which results to 3 hours of maximum wrong value, and then there were no sleep warnings.
More test data from multiple test subjects would be needed to adjust the algorithm which could work on a wider population.

9. Summary

Author investigated human behavioral anomalies in one specific room using mainly thermal image read from the IRMT sensor. Author successfully detected different situations using ML i.e. watching TV, moving around in the room or sleeping in the bed. Author wrote application software (Activity Monitor) on Windows PC which uses ML.NET framework to create the ML model. Given program managed to monitor one person in one specific room and detect 2 types of anomalies - falling and sleep anomaly.

Fall detection was possible using machine learning with extra checks in the code (detecting warm object moving down and waiting 10 seconds for safe situation) before raising the alarm. The accuracy of the created ML model in ML.NET was 100%. After analysing input data with Python scripts using *scikit-learn*, *pandas* and *numpy* frameworks, the best result for test data was 99%. The difficult part was getting fall detection to work, hence the extra checks in the code.

To investigate sleep anomaly (is person sleeping when should not or vice versa) author used the prediction from the ML model, if the person is sleeping or not. Then created an algorithm inspired by Monte Carlo methods to detect anomaly. To verify the algorithm author used sleep data from his Garmin activity monitor. The verification worked, but defining when is persons *is sleeping* status considered as an anomaly was not investigated further.

Additionally author used the ML training data from the IRMT sensor and created ML model using Tensorflow for the STM32 MCU. The model accuracy was 93,6%. Author created a prototype program for the STM32 which showed successful predictions for 4 out 5 predicted states. Falling was not detected properly, but when analyzing prediction values it could be possible with additional code checks.

Finally author investigated NanoEdgeAIStudio, which is an automated ML model creation tool for STM32 devices. The model accuracy 88% was lower than with previous tools, but still promising. Author did not have the required device to test given model, but the emulator in the tool predicted falling 3 out of 3 tries.

9.1 Future work

The prototype PC program (Activity Monitor) could with correct adjustment and in similar conditions, as with the experiments, successfully fulfill its task of detecting fall and sleep anomalies. But since the tests were only performed in lab like environment with only author as a test subject and no additional heat emitting objects, further research should be done. E.g. experimenting with other heat emitting objects in the room like other person, pet or a radiator. Also there are now higher resolution thermal matrix sensors in the market like 32x32 [33], which supports having more detailed image of the room.
Author also implemented ML model on a STM32 MCU and a proof of concept was made. Initial tests showed that the ML model for given task could run on a MCU. Fall detection was only successful on an emulator, which could be further researched on the actual MCU hardware.

10. Kokkuvõte

Autor uuris inimese käitumuslikke anomaaliad ühes kindlas ruumis, kasutades põhiliselt IRMT sensorist loetud termopilti. Autoril osutus võimalikuks tuvastada erinevaid olukordi, nagu televiisori vaatamine, toas ringi liikumine või voodis magamine võttes abiks masinõppe meetodid. Töö käigus loodud arvutiprogrammis (Activity Monitor) oleva masinõppe mudeli valis ML.NET raamistik automaatselt. Lisaks analüüsiti andmeid kasutades Pythoni skripte scikit-learn, pandas ja numpy raamistikus ning andmeid visualiseeriti kasutades matplotlib raamistikku. Uuriti ka masinõppe mudeli realiseeerimist mikrokontrolleri peal.

Anomaaliatest uuriti kukkumist ja uneanomaaliat.

Inimese kukkumine suudeti tuvastada masinõppe ja sellele lisatud lisakontrolliga (soojussignaali liikumine ülevalt alla, ning häire antakse alles kui 10. sekundi jooksul pole tuvastatud ohtutut olukorda). Masinõppe mudel suudeti siis luua 100%-ilise täpsusega. Sisendandmete analüüs, kus osa andmeid jäeti testiks kõrvale andis tulemuse 99% juures. Katsed kukkumist tavalisest toas liikumist eristada, aga andsid aegajalt tavalise liikumise peale ka häire, lisakontrollid aitasid vältida valehäireid.

Uneanomaalia tuvastamiseks võeti appi andmed aktiivsusmonitorist ja loodi Monte Carlo meetoditest ja Markovi ahelatest inspireeritud anomaalia tuvastamisalgoritm. Antud algoritm andis katsete tulemusel anomaalia kui inimene magas paar tundi kauem või läks paar tundi hiljem magama. Erinevad konfiguratsiooniparameetrid võimaldvad algrotimi vastavalt vajadustele seadistada.

Lisaks uuris autor masinõppe mudeli realiseerimist STM32 mikrokontrolleri peal. Mudeli täpsuseks tuli 93,6%. Autor teostas programmi prototüübi, mis suutis ennustada neli olekut viiest. Kukkumist ei suudetud tuvastada, aga analüüsides tulemusi, leidis autor, et kukkumise tuvastamine oleks võimalik kui rakendada lisakontrolle lisaks masinõppe mudeli väljundile.

Lõpetuseks uuris autor ka NanoEdgeAIStudio programmi, mis võimaldab luua masinõppemudeli automaatselt STM32 mikrokontrollerile. Mudeli täpsuseks tuli 88%. Täpsus on

küll madalam kui eelnevate meetoditega, aga see-eest programmis oleva emulaatori peal tuvastati kukkumine kolmel korral kolmest katsest.

Võimalikuks edaspidiseks arenduseks võiks olla kogu lahenduse töölesaamine mikrokontrolleri peal.

Bibliography

[1] Christel Nilsen. "Localization methods for elderly people assistance". MA thesis. TalTech, 2015. URL: https://digikogu.taltech.ee/et/Item/8c8d39db-56de-42e3-85ae-75389dfc1c02.

[2] Emre Arslan. "Developing internet of things and machine learning based bidirectional people counting system with passive infrared sensors". MA thesis. TalTech, 2021. URL: https://digikogu.taltech.ee/et/Item/139cddaf-ceb6-4044-9db9-63bba1cff189.

[3] Yin C. Chen J. Miao X. Jiang H. Chen D. "Device-Free Human Activity Recognition with Low-Resolution Infrared Array Sensor Using Long Short-Term Memory Neural Network". In: (2021). URL: https://www.mdpi.com/1424-8220/21/10/3551.

[4] Anthony Rowe Chandrayee Basu. "Tracking Motion and Proxemics using Thermal-sensor Array". In: (2015). URL: https://arxiv.org/abs/1511.08166.

[5] Will Koehrsen. "Markov Chain Monte Carlo in Python". In: *Medium* (2018). URL: https://towardsdatascience.com/markov-chain-monte-carlo-in-python-44f7e609be98.

[6] Wikipedia. *Markov property*. 01-02-2022 [Accessed: 01-02-2022]. URL: https://en.wikipedia.org/wiki/Markov_property.

[7] Wikipedia. *Monte Carlo method*. 19-01-2022 [Accessed: 22-02-2022]. URL: https://en.wikipedia.org/wiki/Monte_Carlo_method.

[8] *PyMC3 Documentation*. [Accessed: 22-07-2020]. URL: https://docs.pymc.io/.

[9] Wikipedia. *Metropolis–Hastings algorithm*. 26-06-2020 [Accessed: 30-07-2020]. URL: https://en.wikipedia.org/wiki/Metropolis%E2%80%93Hastings_algorithm.

[10] Gareth Williams. *Looking after mom and dad*. 23-07-2020 [Accessed: 21-10-2020]. URL: https://lookingaftermomanddad.com/video-surveillance-for-elderly-monitoring-and-safety-a-helpful-survey/?cn-reloaded=1.

[11] Keith Kirkpatrick. “Sensors for Seniors”. In: *COMMUNICATIONS OF THE ACM* 57.12 (2014), pp. 17–19.

[12] *A Comprehensive Learning Integrity Platform*. [Accessed: 29-07-2020]. URL: https://proctorio.com/.

[13] Reddit. *r/USF - Is Proctorio even effective?* [Accessed: 29-07-2020]. URL: https://www.reddit.com/r/USF/comments/7shh2i/is_proctorio_even_effective/.

[14] Laszlo Richard Toth. “Students raise concerns over virtual proctoring”. In: *The Daily Illini* (2020). 17-04-2020 [Accessed: 29-07-2020]. URL: https://dailyillini.com/news/2020/04/17/proctorio-concerns/.

[15] Centers for Disease Control and Prevention. *Keep on Your Feet-Preventing Older Adult Falls*. 16-12-2020 [Accessed: 28-03-2022]. URL: https://www.cdc.gov/injury/features/older-adult-falls/index.html.

[16] Wikipedia. *Accelerometer*. 04-02-2022 [Accessed: 11-02-2022]. URL: https://en.wikipedia.org/wiki/Accelerometer.

[17] Mark Pickavance. *Best Fall Detection Sensors of 2022*. 16-07-2021 [Accessed: 11-02-2022]. URL: https://www.techradar.com/best/best-fall-detection-sensors.

[18] Amazon.de: DIY Tools. *AMG Emergency Call Button with GPS Transmitter for Seniors Safe at Home and on the Go with Lanyard and Charging Station Splash-proof Modern and Discreet*. [Accessed: 28-03-2022]. URL: https://www.amazon.de/gp/product/B01N12P33I/ref=ppx_yo_dt_b_asin_title_o01_s00?ie=UTF8&amp&psc=1&language=en_GB&currency=EUR.

[19] Machine Learning Mastery. *Machine Learning is Popular Right Now*. 09-12-2020 [Accessed: 28-03-2022]. URL: https://machinelearningmastery.com/machine-learning-is-popular/.

[20] McGraw Hill. *Machine Learning textbook*. [Accessed: 28-03-2022]. URL: http://www.cs.cmu.edu/~tom/mlbook.html.

[21] Wikipedia. *Supervised learning*. 08-04-2020 [Accessed: 20-04-2020]. URL: https://en.wikipedia.org/wiki/Supervised_learning.

[22] Wikipedia. *Unsupervised learning*. 08-04-2020 [Accessed: 21-04-2020]. URL: https://en.wikipedia.org/wiki/Unsupervised_learning.

[23] Wikipedia. *Reinforcement learning*. 15-04-2020 [Accessed: 21-04-2020]. URL: https://en.wikipedia.org/wiki/Reinforcement_learning.

[24] Microsoft. *ML.NET Tutorial: Get started in 10 minutes: .NET*. [Accessed: 26-03-2020]. URL: https://dotnet.microsoft.com/learn/ml-dotnet/get-started-tutorial/scenario.

[25] *scikit-learn*. [Accessed: 05-07-2020]. URL: https://scikit-learn.org/stable/.

[26] *pandas*. [Accessed: 05-07-2020]. URL: https://pandas.pydata.org/.

[27] *Visualization with Python*. [Accessed: 05-07-2020]. URL: https://matplotlib.org/.

[28] *NumPy*. [Accessed: 05-07-2020]. URL: https://numpy.org/.

[29] Wikipedia. *Naive Bayes classifier*. 18-02-2022 [Accessed: 22-02-2022]. URL: https://en.wikipedia.org/wiki/Naive_Bayes_classifier.

[30] Tony Yiu. "Understanding Random Forest". In: *Medium* (2019). URL: https://towardsdatascience.com/understanding-random-forest-58381e0602d2.

[31] Tanel Kaart. *Logistiline regressioon*. [Accessed: 26-11-2020]. URL: http://ph.emu.ee/~ktanel/bin_tunnuste_analyys/pt31.php.

[32] STMicroelectronics. *Automated Machine Learning (ML) tool for STM32 developers*. [Accessed: 23-03-2022]. URL: https://www.st.com/en/development-tools/nanoedgeaistudio.html.

[33] Omron. *D6T MEMS Thermal Sensors*. [Accessed: 03-05-2022]. URL: https://components.omron.com/us-en/solutions/sensor/mems-thermals-sensors.

[34] Garmin e-pood. *Vivomove*. [Accessed: 08-08-2020]. URL: https://garmin.com.ee/et/arhiiv/672-vivomove.html.

[35] STMicroelectronics. *STM32F429*. [Accessed: 06-02-2022]. URL: https://www.st.com/en/microcontrollers-microprocessors/stm32f429-439.html.

[36] Keras. *Dense layer*. [Accessed: 06-02-2022]. URL: https://keras.io/api/layers/core_layers/dense/.

[37] GeeksforGeeks. *Top 5 Programming Languages and their Libraries for Machine Learning in 2020*. 06-11-2021 [Accessed: 29-03-2022]. URL: https://www.geeksforgeeks.org/top-5-programming-languages-and-their-libraries-for-machine-learning-in-2020/.

[38] scikit. *Metrics*. [Accessed: 29-03-2022]. URL: https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics.

Appendices

Appendix 1 - Source code

Three types of source code implemented for given work. All of them are uploaded to *github* repository: ml.net-activity-monitor

1. Windows desktop program in C# containing few thousand lines of code.
2. Python scripts for data analysis and visualization.
3. Python scripts for converting ML model to be used on MCU