Degree Thesis

Computer Engineering, 180 credits

IR Image Machine Learning for Smart Homes

Degree Project Computer Engineering,
15 credits

Elias Josse, Amanda Nerborg

HALMSTAD
UNIVERSITY

Abstract

Sweden is expecting an aging population and a shortage of healthcare professionals in the near future. This amounts to problems like providing a safe and dignified life for the elderly both economically and practically. Technical solutions that contribute to safety, comfort and quick help when needed is essential for this future. Nowadays, a lot of solutions include a camera, which is effective but invasive on personal integrity. Griddy, a hardware solution containing a Panasonic Grid-EYE, an infrared thermopile array sensor, offers more integrity for the user. Griddy was developed by students in a previous project and was used for this projects data collecting. With Griddy mounted over a bed and additional software to determine if the user is on the bed or not a system could offer monitoring with little human interaction. The purpose was to determine if this system could predict human presence with high accuracy and what limitations it might have.

Two data sets, a main and a variational, were captured with Griddy. The main data set consisted of 240 images with the label “person” and 240 images with the label “no person”. The machine learning algorithms used were Support Vector Machine (SVM), k-Nearest Neighbors (kNN) and Neural Network (NN). With 10-Fold Cross Validation, the highest accuracy found was for both SVM and kNN (0.99). This was verified with both algorithms accuracy (1.0) on the test set. The results for the variational data set showed lower reliability in the system when faced with variations not presented in the training, such as elevated room temperature or a duvet covering the person. More work needs to be done to expand the main data set to meet the challenge of variations.

Keywords

Machine learning, infrared radiation, low resolution, Panasonic Grid-EYE, human detection

Sammanfattning

I Sveriges väntas i den närmaste framtiden en åldrande population och en brist på vårdpersonal. Detta innebär både ekonomiska och praktiska problem för att ge äldre ett säkert och värdigt liv. Tekniska lösningar som kan bidra med säkerhet, komfort och snabb hjälp vid behov är av essentiell vikt i framtiden. Idag innehåller många lösningar en kamera. Detta är en effektiv men integritetskränkande lösning. Griddy, som är en hårdvarulösning innehållande en Panasonic Grid-EYE, en infraröd termosensor, erbjuder mer integritet för brukaren. Griddy utvecklades av studenter i ett tidigare projekt och användes för datainsamling i detta projektet. Genom att montera Griddy över sängen och använda en tillhörande mjukvara, som avgör om brukaren är i sängen eller inte, skulle ett system kunna erbjuda övervakning med lite mänsklig inblandning. Syftet var att ta reda på om detta system skulle kunna avgöra brukarens närvaro med hög tillförlitlighet och vilka begränsningar systemet skulle ha.

Två datasamlingar samlades in med hjälp av Griddy. En huvudsaklig datasamling och en med variation. Den huvudsakliga datasamlingen bestod av 240 bilder med etiketten "person" och 240 bilder med etiketten "ingen person". Algoritmerna för maskininlärning som användes var Support Vector Machine (SVM), k-Nearest Neighbors (kNN) och Neural Network (NN). Med 10-Fold Cross Validation fanns den högsta tillförlitligheten med algoritmerna SVM och kNN (0.99). Detta verifierades med tillförlitligheten för testsamlingen hos SVM och kNN (1.0). För datasamlingen med variation visade resultaten på en lägre tillförlitlighet när systemet mötte variationer som det inte tränats med, såsom förhöjd rumstemperatur eller ett täcke över personen. Slutsatsen är att en huvudsaklig datasamling bör utökas med mer variation så att systemet tränas till att klara större utmaningar.

Acknowledgments

We would like to thank Phoniro for the opportunity to explore this field of science. Thank you Ludvig Åhlin and Johan Bröhne at Phoniro for support and guidance.

We would also like to thank Kevin Hernandez Diaz for all the support and guidance throughout the thesis work. Lastly, we would like to thank Wagner Ourique De Morais for helping us with equipment and premises for data collecting.

Content

1. Introduction

In this chapter, the problem will be presented followed by the purpose of the project. The limitations will be explained in regards to software, hardware and environment. The set requirements for the project will be explained and lastly, a brief outline will present an overview of the report.

1.1. Problem

Sweden's population is growing and is expected to keep doing so (SCB, 2018). People live longer in average. The proportion of the population that is between 20 to 64 years old are reducing. This can have big economical consequences for the country, since people between 20 to 64 years old constitutes the majority of the providing quota (SCB, 2017).

Sweden is also expecting a considerable shortage of healthcare professionals in only around 15 years' time from now. A large number of people were born during the 1940s in Sweden. These people are now expected to start being in need of elderly healthcare. The part of the Swedish population that is 80 years or older is expected to increase with 76 percent from the year of 2015 to the year of 2035. This means an increase from 500 000 people to 890 000 people. In contrast to this, Sweden is expecting a lower number of health care professionals in the year 2035 than today. The reason for this is a combination of lack of interest in healthcare education amongst young people and the fact that the average healthcare professional of today is older than 45 years (SCB, 2016).

The number of residences reserved for the elderly or disabled in Sweden at the end of 2018 was 135 769 (SCB, 2019). 123 of Sweden's 290 counties considered themselves having a shortage of residences for the elderly in 2019. 92 of the counties also answered that they did not think that the need for special residences for the elderly would be met in five years' time (Boverket, 2019).

The expectations surrounding Sweden's elderly in the future raises questions about how we will take care of all of them in a way that is both humane and economically possible. The human resources, the economical resources and the living solutions for the elderly will all have to be distributed with care.

Nowadays, a lot of elderly or disabled people get help from professionals in their own homes. This solution can only be presumed to be even more common in the future. Professional help in their own home can be both human and technological. Currently, a lot of solutions contain a mixture of both to optimize the personnel resources as well as safety.

One way to offer safety and distribution of personnel resources for an elderly person living in his own home is to monitor him. This way, the staff can visit him when he needs it the most

and leave him be when he does not need help. Monitoring can be done with different kinds of sensors with cameras being one of them. Cameras are an effective and cost-efficient solution. It lets a monitoring staff member see if the person is on his bed or not at night. By making this decision from the cameras monitoring instead of physically checking up on the person, time and money can be saved. In addition, an eventual situation where the person wakes up from human disturbance can be avoided.

Regular cameras are limited to data retrievals when there is sufficient lighting in the room. Monitoring with cameras also comes with a high price on personal integrity. The data that can be collected from a camera would ideally be collected without being able to identify the patient. One way of doing such a recording is to use a camera with a resolution so low that it makes recognition of a person impossible. The Grid-EYE sensor from Panasonic offers an eight by eight-pixel thermal image. A thermal image offers the ability to collect data even in dark environments (Panasonic, 2019). The low amount of pixels offers more integrity than a higher amount of pixels would. However, it could be impossible for the personnel to monitor by watching. Thus, this alternative solution would need the addition of machine learning software for predictions.

Phoniro is part of ASSA ABLOY, a global company with a leading position in the market of access solutions (Assa Abloy, 2019). Phoniro is a national company with innovative products and solutions designed specifically for home health care companies and senior living communities. Phoniro specializes in making the taking care of sick or elderly people easier and more effective, while also empowering by giving them greater independence with the use of modern technology (Phoniro).

This work is in collaboration with Phoniro. In the autumn of 2019, a project done by students from Halmstad University in collaboration with Phoniro resulted in the prototype “Griddy” (see Figure 1). This is a hardware prototype consisting of a Panasonic Grid-EYE infrared thermopile array sensor, a microprocessor and a Bluetooth module. Griddy can be plugged into a wall socket with 230 voltage and collect data with a speed of 1 or 10 frames per second. The data can then be transmitted to a computer via Bluetooth. This project takes over from here with the remaining problem of processing and interpreting the data that can be collected.

Figure 1. A prototype of Griddy with the Panasonic Grid-EYE infrared thermopile array sensor visible on the front.

1.2. Purpose

This project aims to determine how successful some well-known machine learning algorithms are at detecting if a person is lying in the bed or not using data collected by Griddy, as low-resolution infrared images.

How well do the algorithms handle data from an altered environment, not present during training? What data should be included in a data set used for the training of a future system?

1.3. Limitations

The system was developed using only free software Application Programming Interfaces (API). No hardware was developed. The hardware used was limited to the prototype Griddy provided by Phoniro and private laptops. The environment for data collection was restricted to the bedroom of the Halmstad Intelligent Home (HINT-room) and its single bed. The people used in the collected data was limited to the project's participants and volunteering students of Halmstad University.

Data collecting was constrained to involving only no person or one person in an image at a time. The person, when present, was always laying on the bed with his whole body on the bed. Many different positions laying down were collected. No other positions, as for example sitting up or standing, were collected. The conditions of the data collecting for the main data set was limited to the rooms standard temperature (20-21 degrees Celsius) and its standard lighting.

1.4. Requirements

The main set of collected data was required to include a minimum of 480 images of data where 240 was data with a person present and 240 was data with no person present. An extra set of collected data for variational testing was to include a source of heat other than a person. This was for representing a smaller pet laying on the bed. This object needed to have a minimum of 35 and a maximum of 40 degrees Celsius. It needed to be present on a minimum of 20 images with no person and 20 images with a person present. Another extra set of collected data for variational testing was to include a minimum of 20 images where a person was present with a duvet on top of him. The last extra set of collected data for variational testing was to include a minimum of 20 images where the room temperature was four degrees higher than the standard room temperature of 20-21 degrees Celsius. This gives a room temperature of 24-25 degrees which is sufficient to represent a hot summer night in Sweden.

The three machine learning algorithms Support Vector Machine, k-Nearest Neighbors and Neural Network were to be used with classification. The code was to be written in Python scripts using the Scikit Learn library. For each algorithm, the most relevant parameters must be changed to find the best ones for the task. Four different kernels were used for Support Vector Machine. A range of different k was to be used for k-Nearest Neighbors to find satisfactory results. Lastly, different numbers of neurons needed to be used for Neural Network. A range of one to 1024 neurons in the hidden layer was to be tried out. The three machine learning algorithms were to be compared with each other. The comparison was to include accuracy, sensitivity and specificity.

The system required to have an accuracy of at least 90% calculated on the main test data set for the best performing algorithm. The main test data set was 20% of the entire main data set. The other 80% was a training set.

1.5. Outline

The outline of the report is as follows: chapter 2 presents a background of machine learning, infrared sensors and previous related work. In chapter 3, the method is presented, including the data collecting, data exploration and algorithm testing. In chapter 4, the results for all algorithms can be found for the two datasets. Chapter 5 is the discussion of the method and results with regards to the background and related work. Finally, the work is concluded in chapter 6.

2. Background

In this chapter, machine learning will be described, as well as explore in depth some of the algorithms used in the project. It will also be presented how the data and the algorithms can be used and evaluated. After that, infrared sensors will be introduced to give context to the hardware involved. Lastly, related work in the field of machine learning, infrared image collecting and human detection will be presented.

2.1. Machine learning

Machine learning systems have the ability to learn and improve from experience. It is a subfield of computer science. Algorithms rely on collections of samples of some class. The samples can come from another algorithm, be handcrafted or come from nature (Burkov, 2019).

Machine learning can be categorized into supervised and unsupervised learning. Any supervised learning algorithm takes a feature vector as input and outputs information that can deduce the label for this feature vector. An unsupervised learning algorithm takes a feature vector as input, transforms it into another vector or value, and use this to solve the problem and find patterns in the features.(Alpaydin, 2010) (Marsland, 2015).

A machine learning algorithm can either use classification or regression. Some algorithms can be used with either one. In classification, a label gets automatically assigned to an unlabeled example. If the set of classes includes two classes, for example “person” and “no person”, it is called binary classification. Multiclass classification is all classification problems with three or more classes. Some machine learning algorithms only allow binary classification. Regression is predicting an unlabeled example real-valued label. A machine learning algorithm with regression takes a collection of valued examples as input and produces the model that can take unvalued examples as input and output a constructed value (Burkov, 2019).

Following is a more in-depth explanation of the three algorithms Support Vector Machine (SVM), k-Nearest Neighbors (kNN) and Neural Network (NN). The algorithms will be explained from a perspective of binary classification.

2.1.1. Support Vector Machine

SVM is one of the most popular machine learning algorithms (Marsland, 2015) and was first introduced by Cortes and Vapnik (1995). SVM is a common machine learning algorithm that can be used for a wide variety of problems including both classification and regression. In classification, it can be used for binary as well as multilabel problems. However, SVM is not so good for high dimensional tasks (Burkov, 2019).

The basis for SVM is to try and find the optimal separation between two classes. A case where the classes consist of samples containing two features is illustrated in Figure 2. It can be viewed as a collection D in equation 1.

$$D = \{(\boldsymbol{x}_i, y_i)\}_{i=0}^{L} \quad (1)$$

where

$\boldsymbol{x} = \left(a^{(1)}, a^{(2)}\right)$	$\boldsymbol{x}$ is a sample,
$y \in \{-1,1\}$	y is a label,
L	is the amount of samples (Marsland, 2015).

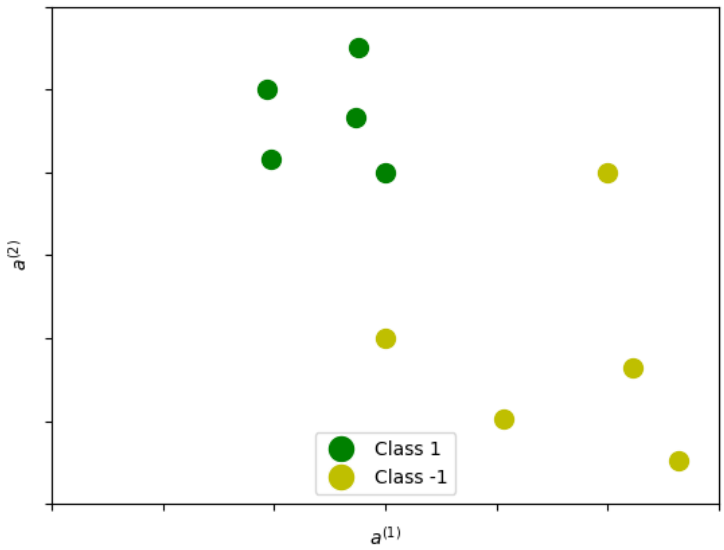

Figure 2. An example data set with two features $\left(a^{(1)} and\ a^{(2)}\right)$.

There are many different lines which can act as separation between these two classes. But there is only one line that is considered optimal. This is known as the maximum margin classifier. The maximum margin is the minimum distance between the decision boundary and its closest sample. These samples are known as the support vectors. The classes are most separated when the maximum margin is found (Marsland, 2015). This is illustrated in Figure 3.

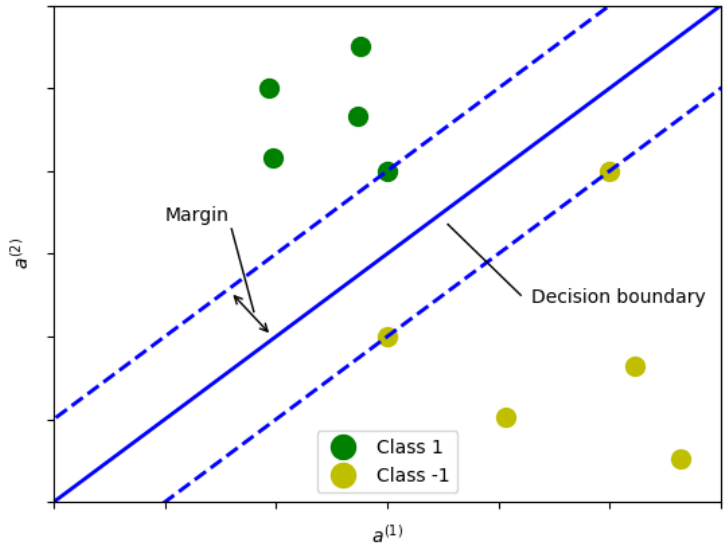

Figure 3. The example from Figure 2 is here shown with a maximum margin classifier.

The decision boundary is defined in equation 2.

$$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0 \tag{2}$$

where

$\boldsymbol{w}$	is known as a weight vector which is of the same shape as a sample $\boldsymbol{x}$. It is a normal to the separation line and therefore travels perpendicular to the line,
b	is a scalar value which is known as the bias,
$\boldsymbol{w}\cdot\boldsymbol{x}$	is the dot product of the two vertices,
$\frac{1}{\|\|w\|\|}$	is the calculated margin (Marsland, 2015).

In order to take the maximum margin into account, the problem could be defined as in equation 3 and equation 4.

$$\boldsymbol{x_i} \cdot \boldsymbol{w} + \boldsymbol{b} \geq \mathbf{1} \text{ for } y_i = 1 \tag{3}$$
$$\boldsymbol{x_i} \cdot \boldsymbol{w} + \boldsymbol{b} \leq -\mathbf{1} \text{ for } y_i = -1 \tag{4}$$

These equations can be combined into equation 5.

$$y_i(\boldsymbol{x}_i \cdot \boldsymbol{w} + \boldsymbol{b}) - \mathbf{1} \geq \mathbf{0}\ \forall_{\boldsymbol{i}} \tag{5}$$

For the support vectors equality holds true, see equation 6.

$$y_s(\boldsymbol{x}_s \cdot \boldsymbol{w} + \boldsymbol{b}) = \mathbf{1} \tag{6}$$

The goal is to find the right **w** and *b* for the largest margin. In other words, the smallest $||w||$ will result in the largest margin for the already established constraints. Furthermore, $||w||$ can be replaced by $\frac{1}{2}||w||^2$. A change that will make Quadratic Programming possible in order to optimize the problem later on. Something that Marsland (2015), Fletcher (2009) and Burkov (2019) do. The problem is now defined as equation 7.

$$\text{minimize}(\tfrac{1}{2}||w||^2) \text{ such that } y_i(\boldsymbol{x}_i \cdot \boldsymbol{w} + \boldsymbol{b}) - \mathbf{1} \geq \mathbf{0}\ \forall_{\boldsymbol{i}} \tag{7}$$

Lagrange multipliers (λ) can be used in order to find the right margin for the given constraints. Lagrange multipliers are positive values which are used to solve this type of equations with equality constraints. The '*' denotes the optimal value for the parameters when $\lambda \neq 0$ in equation 8.

$$\lambda_i^*(1 - y_i(\boldsymbol{x}_i \cdot \boldsymbol{w}^* + \boldsymbol{b}^*)) = \mathbf{0} \tag{8}$$

where

$\boldsymbol{x_i}$ is a support vector (Marsland, 2015).

This can be made into a Lagrangian function in equation 9.

$$\begin{aligned} G(\boldsymbol{w}, b, \boldsymbol{\lambda}) &= \frac{1}{2}||w||^2 - \sum_{i=1}^{L} \boldsymbol{\lambda_i}(\boldsymbol{y_i}(\boldsymbol{x_i} \cdot \boldsymbol{w} + b) - \mathbf{1}) \\ &= \tfrac{1}{2}||w||^2 - \textstyle\sum_{i=1}^{L} \boldsymbol{\lambda_i y_i}(\boldsymbol{x_i} \cdot \boldsymbol{w} + b)\ + \sum_{\boldsymbol{i=1}}^{\boldsymbol{L}} \boldsymbol{\lambda_i} \end{aligned} \tag{9}$$

The purpose here is to find where $\boldsymbol{w}$ and *b* are at its minimum and to find $\boldsymbol{\lambda}$ at its maximum. This can be done by calculating the partial derivatives of *G* with respect to $\boldsymbol{w}$ and *b*. Then set these to zero as for equation 10 and equation 11.

$$\frac{\partial G}{\partial \boldsymbol{w}} = 0 \Rightarrow \boldsymbol{w}^* = \textstyle\sum_{\boldsymbol{i=1}}^{\boldsymbol{L}} \boldsymbol{\lambda_i y_i x_i} \tag{10}$$

$$\frac{\partial G}{\partial \boldsymbol{b}} = 0 \Rightarrow \textstyle\sum_{\boldsymbol{i=1}}^{\boldsymbol{L}} \boldsymbol{\lambda_i y_i} = \mathbf{0} \tag{11}$$

Substitute the derivatives into $G(\boldsymbol{w}, \boldsymbol{b}, \boldsymbol{\lambda})$ and simplify to get equation 12.

$$G(\boldsymbol{w}^*, \boldsymbol{b}^*, \boldsymbol{\lambda}) = \textstyle\sum_{\boldsymbol{i=1}}^{\boldsymbol{L}} \boldsymbol{\lambda_i} - \frac{\mathbf{1}}{\mathbf{2}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{L}} \sum_{\boldsymbol{j=1}}^{\boldsymbol{L}} \boldsymbol{\lambda_i \lambda_j y_i y_j x_i} \cdot\ \boldsymbol{x_j} \tag{12}$$

with the constraints:

$$\lambda_i^* \geq \mathbf{0} \text{ and } \textstyle\sum_{\boldsymbol{i=1}}^{\boldsymbol{L}} \boldsymbol{\lambda_i y_i} = \mathbf{0}$$

Now the problem is known as a quadratic optimization problem. To find the largest λ, a quadratic programming solver can be used. Then $\boldsymbol{w^*}$ can be calculated from the previously derived expression.

The optimal b (b*) can be found by substituting the $\boldsymbol{w^*}$ into equation 6. This gives equation 13 and equation 14.

$$y_s\left(\sum_{m=1}^{C} \boldsymbol{\lambda_m y_m x_m} \cdot \boldsymbol{x_s} + \boldsymbol{b}\right) = \mathbf{1} \tag{13}$$

$$b^* = y_s - \sum_{m=1}^{C} \boldsymbol{\lambda_m y_m x_m} \cdot \boldsymbol{x_s} \tag{14}$$

where

s is the index of an arbitrary support vector,
C is a collection of support vector indices.

Both Marsland (2015) and Fletcher (2009) calculates the average b* across all of the support vectors, which is considered more stable.

In the end, the maximum margin classifier looks like equation 15.

$$y = z \cdot \boldsymbol{w}^* + \boldsymbol{b}^* \tag{15}$$

where

z is a new sample predicted to belong to class y.

These are the necessary calculations for finding the maximum margin classifier with a hard margin. A hard margin is useful when the data is linearly separable. It struggles when the data might contain noise, which would make it non-linearly separable. Such noise can for example be mislabeled data, which in turn would make a hard margin a lot worse. The solution to this is to use a soft margin classifier, which has penalty parameters. These parameters increase more when mislabeled data is placed further from the decision boundary (Marsland, 2015).

However, when the data is inherently non-linear, SVM can use something called the kernel trick. The kernel trick is composed of a set of functions where each one consist on calculating the dot product of sample vectors to create new features from existing ones. By doing this the whole data set is brought to a higher dimension where a decision boundary of a higher dimension than previously mentioned can be used to separate the classes (Fletcher, 2009).

Each data sample **x** is transformed into $\Phi(\mathbf{x})$, which turns the data sample into one of higher dimensionality with the use of existing features. By doing this to the maximum margin classifier it now looks like equation 16.

$$y = \Phi(z_i) * \boldsymbol{w}^* + \boldsymbol{b}^*, \boldsymbol{w}^* = \sum_{i=1}^{L} \boldsymbol{\lambda_i y_i \Phi(x_i)} \quad (16)$$

and the Lagrangian function turns into equation 17.

$$G(\boldsymbol{w}^*, \boldsymbol{b}^*, \boldsymbol{\lambda}) = \sum_{i=1}^{L} \boldsymbol{\lambda_i} - \frac{1}{2} \sum_{i=1}^{L} \sum_{j=1}^{L} \boldsymbol{\lambda_i \lambda_j y_i y_j \Phi(x_i) \cdot \Phi(x_j)} \quad (17)$$

Knowing which mapping functions to use can be difficult. Considering that calculating $\boldsymbol{\Phi(x_i) \cdot \Phi(x_j)}$ is also computationally expensive, this could be inefficient. Fortunately, here is where the kernel trick comes in. The result of the dot product in terms of $x_i \; and \; x_j$ have already been calculated, these are known as kernels $k(x_i, x_j)$. The most common kernels are shown in equation 18-21.

Linear

$$k(x_i, x_j) = (x_i \cdot x_j + a) \quad (18)$$

Polynomial

$$k(x_i, x_j) = (x_i \cdot x_j + a)^d \quad (19)$$

Radial Basis Function (RBF)

$$k(x_i, x_j) = e^{-\left(\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)} \quad (20)$$

Sigmoid

$$k(x_i, x_j) = \tanh(a x_i \cdot x_j - b)^d \quad (21)$$

These are the recommended kernels to use by Fletcher (2009), Marsland (2015), and Burkov (2019).

SVM has some advantages and some disadvantages. Because of its parameterized form, it is memory efficient. SVM is generally effective at reasonably sized data sets and worse at larger sized sets. It has fast prediction time. SVM was created with the ability to separate two classes (Cortes & Vapnik, 1995) and it is inherently binary, which makes it not so suitable for multiclass problems (Burkov, 2019).

2.1.2. k-Nearest Neighbors

kNN is one of the simplest machine learning algorithms. The core principle is that it weighs the evidence of the nearby samples most heavily. The first time the nearest neighbor rule was formulated was in 1951 by Fix and Hodges who investigated the k-Nearest Neighbor rule. This rule assigns, to an unclassified sample, the class that is most heavily represented among

its k nearest neighbors (Dudani, 1976). kNN keeps all training samples in memory. The fact that it does not allow discarding of the training data after the model is built makes it a non-parametric machine learning algorithm. It can be used for both classification and regression (Burkov, 2019).

The kNN algorithm predicts a label for an unseen input sample by looking at the close neighborhood of the input. The k stands for the number of neighbors that are closest to the unseen sample. The k should be an odd number in the case of two classes to avoid a tie (Kim, Kim & Savarese, 2012). When predicting, kNN calculates the sum of the weights of k closest neighbors to predict the new sample (Alpaydin, 2010). The weights can be either uniform- or distance-based. The uniform approach uses a full majority vote where every neighbor has equal value. For the distance-based one, weights are usually calculated as the inverse proportional Euclidean distance. This means that the neighbors closest to the new sample are weighted more heavily (Burkov, 2019).

The Euclidean distance between two points p and q can be calculated as in equation 22.

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(p^{(i)} - q^{(i)})^2} \qquad (22)$$

where

n is the number of features.

It is often the case that the samples belonging to the same class form a cluster (Gallego et al. 2018). Figure 4 shows an example of how kNN can classify a sample where there are two features. Depending on the value of k there will be different classifications. If k is set to one, the new sample will be classified as Class -1. If k is set to three, the new sample will be classified as Class 1.

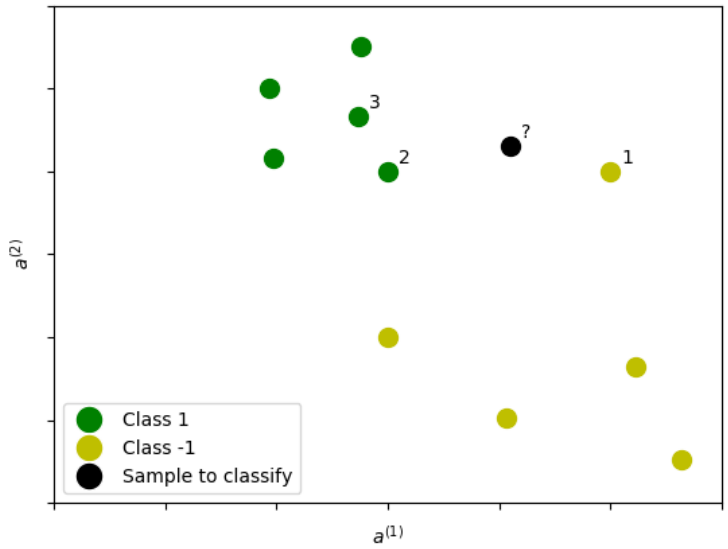

Figure 4. An example of how kNN can classify a new sample. Depending on the value of k there will be different classifications. K = 1 will give the new sample the label Class -1 whereas k = 3 will give the new sample the label Class 1.

The main advantage of kNN is that it is simple to understand and implement. Because its decision is based on a small neighborhood of similar objects it performs well with multiple classes (Kim, Kim & Savarese, 2012). The disadvantages of kNN are that it is slow at predicting (Burkov, 2019) (Gallego et al., 2018) and has a fairly high memory usage (Gallego et al., 2018). Another disadvantage is that it uses all features equally in computing for similarities. If there is only a small set of useful features for classification this will be a problem and can result in classification errors (Kim, Kim & Savarese, 2012).

2.1.3. Neural Network

Neural Network is designed to recognize patterns and received the name because it is modelled after the human brain (Alpaydin, 2010). NN consists of several layers from which the first one is called input layer, and it takes the features of a sample. The inputs are fed through one or more layers called hidden layers, until the last one, which is called output layer. In binary classification the output layer uses a decision function that gives a value of either 0 or 1 (Burkov, 2019) (Marsland, 2015). For example 0 “no person” and 1 “person”. The hidden layers allow NN to learn more complicated features. They consist of a number of neurons which purpose is to alter the features into a single value. For each neuron, the input is multiplied with a weight and summed together to a value that is passed onto the next neuron in the network as in equation 23.

$$
\begin{aligned}
a^{(1)} &= \left(w_{(1,1)}x^{(1)} + w_{(2,1)}x^{(2)} + \cdots + w_{(n,1)}x^{(n)}\right) + b \\
a^{(2)} &= \left(w_{(1,2)}x^{(1)} + w_{(2,2)}x^{(2)} + \cdots + w_{(n,2)}x^{(n)}\right) + b \\
&\cdots \\
&\cdots \\
&\cdots \\
a^{(k)} &= \left(w_{(1,k)}x^{(1)} + w_{(2,k)}x^{(2)} + \cdots + w_{(n,k)}x^{(n)}\right) + b
\end{aligned}
\qquad (23)
$$

where

$w_{(n,k)}$ is the weight,
b is the bias.

Note that each input of a neuron has its own weight. The bias is the value that is added to all of the neurons (Burkov, 2019). An example of a NN and its neurons can be viewed in Figure 5.

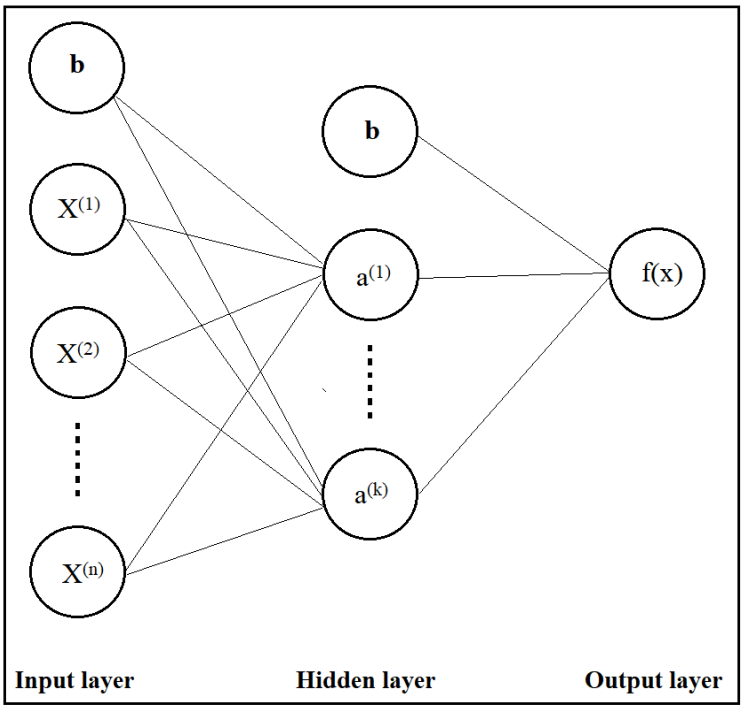

Figure 5. A model of a Neural Network with one hidden layer.

Each neuron is independent from other neurons in the same layer. During training, the weights and biases are adjusted to make the output of the network closer to the actual label of the given sample (Marsland, 2015).

If there is a pattern in the data this can be found by the NN. With pattern recognition the NN can then predict unseen samples correctly (Marsland, 2015).

Multilayer Perceptron (MLP) is a type of neural network. A basic form of MLP uses McCulloch and Pitts neurons, which are simple neurons that take an input and produces a

binary output of either 0 or 1, if they should fire or not fire. This depends on if *a* is larger than a given threshold θ as in equation 24.

$$\begin{aligned} a_{out} &= g(a) \\ g &= 1 \; if \; a > \theta \\ g &= 0 \; if \; a < \theta \end{aligned} \tag{24}$$

This way each neuron has its own activation function g which decides if the neuron should fire or not (Marsland, 2015).

It is hard to know what values the weights and biases of the neurons should have. That is where the learning part comes in. The learning part of the MLP lies in adjusting the weights during training so that they fire accordingly to the desired output. In an instance where a neuron fires correctly nothing is to be done. However, when it fires incorrectly the weights need to be adjusted. The weights are adjusted with the use of the learning rate parameter η. The value of η needs to be picked carefully. Having a large value means that the network can become unstable and not settle down. If it is too small the network will take longer to train. The neuron output is calculated as in equation 25.

$$y_j = g\left(\sum_{i=1}^{n} w_{ij} x^{(i)}\right) \tag{25}$$

where

y_j	is the neuron's output,
$x^{(i)}$	is an input,
w_{ik}	is a weight,
n	is the number of inputs.

If the desired output of a neuron is known as t_j an adjustment can be made to a weight according to equation 26.

$$w_{ij} - \eta\left(y_j - t_j\right) \Rightarrow \; w_{ij} \tag{26}$$

For a more advanced type of neuron the Sigmoid activation function can be used which is shown in equation 27.

$$a_{out} = g(a) = \frac{1}{1+\exp(-\beta a)} \tag{27}$$

The Sigmoid function offers the same limit from 0 to 1 but with a smoother transition in between.

The number of hidden layers and the number of neurons on each hidden layer must be chosen. One single hidden layer can be sufficient when using regular transfer functions, for

example the Sigmoid function according to Stathakis (2009). Two hidden layers can be required for more complicated problems as, for example, when labeling data with discontinuities. More than two hidden layers should not be used because there is no theoretical reason for it. The number of neurons in the hidden layer is often between the size of the input layer and the size of the output layer (Panchal et al., 2011). A reason for using a second hidden layer can be to reduce the total required number of hidden nodes (Stathakis, 2009). Too few neurons in the hidden layer can result in underfitting and to many can result in overfitting. If the data set is on the smaller side it can have too little information contained for a large number of neurons to process. Another problem with too many neurons is that the training time can be impossibly long (Panchal et al., 2011).

The advantages of NN is its ability to handle a huge number of training examples and its extremely fast prediction time. NN can naturally be extended to handle more than two classes, so-called multiclass problems. A disadvantage is that it is slow to train (Burkov, 2019).

2.2. Scikit Learn

Scikit Learn is a programming library for Python with focus on machine learning. It has implementation, documentation and guides for the most common machine learning tools and algorithms, such as kNN, SVM and NN (Scikit Learn).

2.3. Algorithm evaluation

SVM, kNN and NN all come with their pros and cons. NN can handle a huge number of examples and features whereas SVM has a much smaller capacity. If the data can be linearly separable SVM would work very well and if not, the conditions would suit NN better (Burkov, 2019). NN are slow to train. SVM and some types of NN are extremely fast in prediction time whereas kNN is slower. It is good practise to test out different algorithms on the validation set of the data (Burkov, 2019).

There are two things that needs to be done with the data. Firstly an estimation of the parameters for the machine learning algorithms. This is the training part. Secondly an evaluation of how well the machine learning algorithms work. This is the test part. The data is often divided into the three sets training set, validation set and test set. The training set is the biggest portion since it is used to build the model. To get a model that is good at predicting examples that the machine learning algorithm has not seen before it is important that not all data gets presented in the training stage. The validation set is used to choose the machine learning algorithms. The test set is used to assess the model's level of correct predictions. The training set usually is somewhere between 50% and 95% of the data. The validation set and the test set usually split the rest equally between them (Burkov, 2019) (Marsland, 2015).

2.3.1. Cross Validation

Cross Validation allows a comparison between different machine learning algorithms and gives a sense of how well they will work in practice. With Cross Validation all data will be used for testing. In k-Fold Cross Validation the data is divided into k blocks. The number of blocks is arbitrary. A commonly used k is ten which gives the Ten-Fold Cross Validation. When using Cross Validation the data set can be divided into training and test. Hence, the validation set can be left out. Dividing the data for Cross Validation and test is done to avoid and detect overfitting, both by the model and human actions (Burman, 1989) (Jung, 2018).

2.3.2. Performance

It is important to assess the model's performance. Some of the most widely used metrics and tools for assessing the classification model are confusion matrix, accuracy, sensitivity and specificity (Altman et al., 2013) (Burkov, 2019).

A confusion matrix can be used to summarize how successful a classification model is at predictions. The confusion matrix is a table with predicted label and actual label on its axis (see Figure 6). For binary classification it is 16abelled into four cells: true positive (TP), false positive (FP), true negative (TN) and false negative (FN). The distribution between TP, FP, TN and FN can be of big interest depending on the type of data and the consequences of the interpretation. For example, a false positive could potentially lead to a situation where a critical situation goes unnoticed and hence also unchecked (Burkov, 2019).

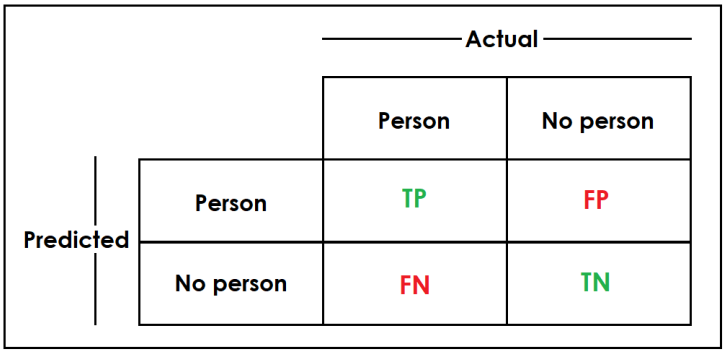

Figure 6. A binary confusion matrix with the two labels "person" and "no person".

The following calculations for accuracy, sensitivity and specificity are for binary classification. Accuracy describes how well the algorithm predicts the right instance across both labels. Sensitivity describes how well the algorithm predicts positive labels. Specificity describes how well the algorithm predicts negative labels (Altman et al., 2013) (Burkov, 2019). The calculations for accuracy, sensitivity and specificity can be viewed in equation 28-30.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (28)$$

$$Sensitivity = \frac{TP}{TP + FN} \tag{29}$$

$$Specificity = \frac{TN}{TN + FP} \tag{30}$$

2.4. Infrared sensors

Infrared radiation (IR) is electromagnetic radiation covering the wavelength longer than the visible but shorter than millimeter waves. Infrared detectors can be categorized into infrared thermal detectors and infrared photon detectors. Infrared thermal detectors can further be categorized into pyroelectric and thermopile (Rogalski, 2010) (Shetty et al., 2017).

Passive infrared sensors are electronic sensors that measure infrared light radiating from an object. They are most commonly used in security alarms and automatic lights. The passive infrared sensor detects the movement but has no way of knowing what kind of object it is that moved. An active infrared sensor can be used to distinguish different kinds of objects and also detect them in stationary positions as well as in motion (Rogalski, 2010) (Shetty et al., 2017).

Grid-EYE, the sensor used in Griddy, is an active infrared thermopile array sensor with 64 thermopile elements arranged in a two dimensional eight by eight grid format. The thermopile elements provide a temperature value each with an accuracy of a quarters degree Celsius. The sensor has a mixed-signal processing integrated circuit. The angle of vision is 60 degrees both horizontal and vertical and the distance of use is up to seven meters. The sensor's output temperature range is -20 degrees to 100 degrees Celsius, rounded to a quarter degree (Panasonic, 2019).

2.5. Related work

Interesting fields of previously done work are machine learning for image analysis and mainly low-resolution images and human detection. Data collecting from infrared sensors are of interest and mainly specifically the Panasonic Grid-EYE. Lastly, it is of interest to look into work done on comparing different machine learning algorithms, both with each other and with different parameters. Related work has previously been done in all of these fields.

Deep artificial neural network architectures can match human performance in recognition of low-resolution images. For example with MNIST, a handwritten digits database commonly used for the training of image processing systems. The original MNIST images are 20 by 20 pixels but there are additional datasets with 10, 12, 14, 16 and 18 pixels in width and height. Focusing on deep convolutional neural networks (DNN) with multiple layers can result in

down to a 0.23 percent fault rate in digit recognition on MNIST original images (Ciresan, Meier and Schmidhuber, 2012).

Skin detection is a way of detecting and tracking humans. For skin detection in pictures Khan, Hanbury and Stoettinger (2010) used color classification based on the machine learning algorithm Random Forest. This is a tree classifier and is popular because of its easy training procedure. In Random Forest, each tree depends on the values of a random vector. A new object from an input vector gets classified through being presented to each tree in the forest, which will all make a vote. The assigned classification is the one having the most votes. The framework Random Forest has quick training times and high generalization accuracy (Burkov, 2019). Khan, Hanbury and Stoettinger (2010) compared the results from Random Forest with, amongst others, the results from SVM and found the Random Forest approach to outperform the others.

Low-resolution thermal sensors have been widely used in indoor environments for human detection. The advantages are mainly the cheap pricing, preservation of privacy and the capability to detect human heat emission in dark environments as well as light (Trofimova et al., 2017) (Shetty et al., 2017). The Panasonic Grid-EYE has been used for this kind of human detection multiple times. Trofimova et al. (2017) did a study where data collected with the Grid-EYE was used to improve the accuracy of the algorithms by considering the temperature variation in the room from other sources than the human. Shetty et al. (2017) did a study in which the Grid-EYE was used to detect a human both in motion and motionless. The collected data was used to detect and track the human's position.

Chen and Wang (2018) examined how a system with the Grid-EYE together with an ultrasonic sensor (HC-SR04) could be used in fall-detection of a human. The sensor fusion method was proposed to derive the human's position and size. The algorithm used, SVM, had the task of differentiating between a fall and another kind of event, for example sitting or stooping.

Another solution with a low-resolution sensor has been done by Pontes et al. (2017) who used the OMRON D6T-44L thermal sensor, with four by four pixels, and machine learning. A prototype was installed in the bedroom ceiling for data retrieval. The task was to recognize body pose and the presence of a person. For the machine learning part decision trees were used. The results of the study indicated that data diversity was of great importance for system performance.

Liu, Lee and Lin (2010) developed a fall detection system where kNN was used to classify the body posture. To differentiate between a fall event and a lying down event time difference was taken into account. The human body silhouette was used to offer more protection of privacy. To determine the best number of neighbors the k-Fold Cross Validation was used.

3. Method

The specification of the project began with the demands and desires of Phoniro. In discussions with Phoniro, they described their need for a human recognition system with machine learning using Griddy. In the beginning, experiments were made using Griddy. The obtained knowledge from the experiments was used to form the method. Two data sets were captured with Griddy where each sample was in the form of an eight by eight matrix. The two sets were the main data set and the variational data set. The machine learning algorithms SVM, kNN and NN were chosen and evaluated with the main data set. The evaluation existed of a 10-Fold Cross Validation and confirmation on the test set. Lastly, all the algorithms were tested against the variational data set.

This chapter begins with a description of the sensor's image capturing followed by the initial pre-experiments. The data collecting will then be described, for both main and variational data set. Lastly, each algorithm's training and testing will be explained.

3.1. Image capture

Software already developed for Griddy included a Python program for visually displaying live images. These images came in an eight by eight format which can be visualized as equation 31.

$$A = \begin{bmatrix} A_{(1,1)} & .. & A_{(1,8)} \\ .. & .. & .. \\ A_{(8,1)} & .. & A_{(8,8)} \end{bmatrix} \quad (31)$$

Every pixel (equation 32) in the image was part of the capture range of the sensor (equation 33).

$$A_{(i,j)} \in \mathbb{D} \quad (32)$$

$$\mathbb{D} = \{-20, -19.75, .., 99.75, 100\} \quad (33)$$

Configurations to this program were made so that it could save the images when pressing a button. Each image was saved in accordance with dataset, date and label. The image was then written as serialized and standard matrix value format. Serialized means that the image could be stored and recovered back into its original state. Matrix format was for look-up convenience purposes. The files were stored on a cloud-storage service. The system of saving sensor output was kept throughout the project.

3.2. Pre-experiments

The initial experiments tested what the sensor could and could not capture. A simple experiment environment was built where the sensor could be used to capture data from an IR-emitting object, in this case a human hand. An isolated surface (two layers of cardboard) was used as a background to protect from interference. Figure 7 shows a picture taken from Griddy's perspective, using a regular cell phone camera, with a hand situated 30 centimeters from the sensor. The sensor output of this case is shown in Figure 8. Figure 9 shows the corresponding output for when the hand was not present.

Figure 7. Cell phone picture from the initial experimental tests. A hand was captured with the sensor at the same angle and distance as the image presents.

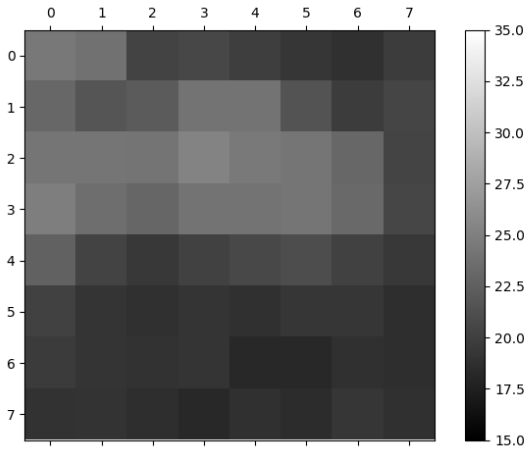

Figure 8. A visual representation of the sensor output of the hand shown in Figure 7. The pixels are displayed in degrees Celsius.

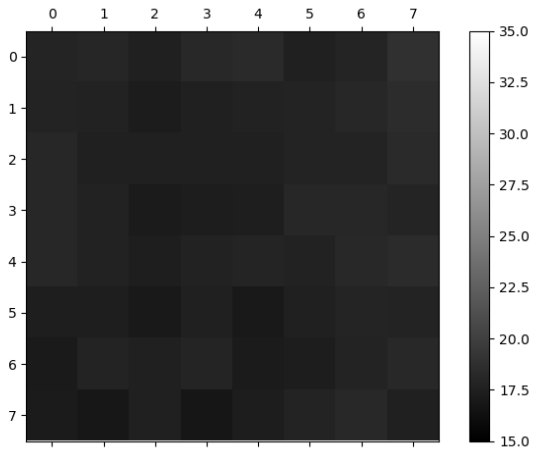

Figure 9. A visual representation of the sensor output of no hand present. The pixels are displayed in degrees Celsius.

Two further experiments were made with this environment while capturing images of the same hand. The first was limiting the amount of visible light exposed to the sensor. This was made by covering up the sensor and the environment, making it darker, to find out what effect it had on the output. The results, shown in Figure 9 and Figure 10, show no noticeable difference in the outputs. The amount of lighting was therefore not given further attention in the project.

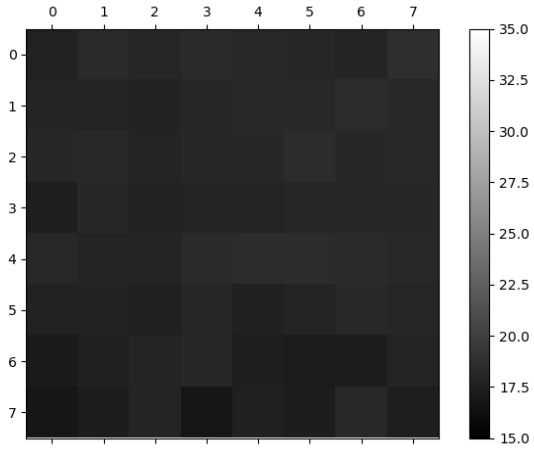

Figure 10. A visual representation of the sensor output of no hand present in the dark. The pixels are displayed in degrees Celsius.

The other experiment was done by covering up the hand with a glove. This was done to test whether covering up an object with some sort of fabric would change the output of the sensor and if it would change over time. The results were saved in two batches, one just after the glove had been put on the hand and the other after letting the glove stay on for five minutes. The resulting outputs are shown in Figure 11 and Figure 12. These outputs showed noticeable differences.

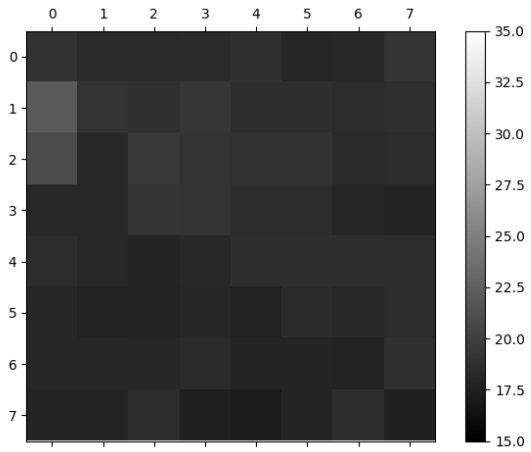

Figure 11. A visual representation of the sensor output of the hand covered by a glove. The glove had just been put on. The pixels are displayed in degrees Celsius.

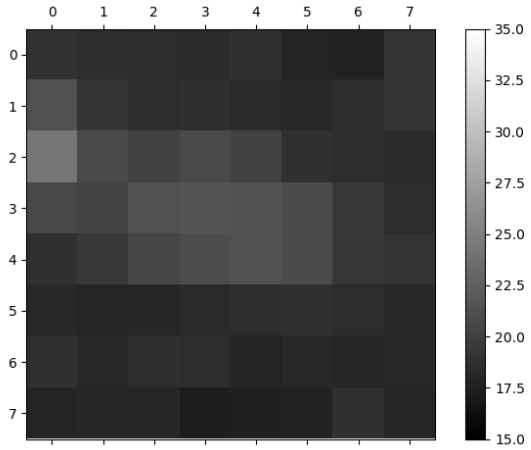

Figure 12. A visual representation of the sensor output of the hand covered by a glove. The glove has been on for five minutes. The pixels are displayed in degrees Celsius.

Differences between the sensor output cases have been calculated by first creating an average matrix for each individual case. Three images for each case were used. Then, from that matrix values for the lowest and highest cell were found.

The initial experiments gave results that helped to form the method. The outputs from the experiments with a hand showed that lighting was not something that needed to be accounted for in the main data set collecting. The temperature, on the other hand, was of great importance. For example, if the hand was to have approximately the same temperature as its surroundings, the sensor would not be able to capture it satisfactorily.

All pre-experiment resulted in a bit of noise showing in the matrices. This was assumed to be the result of mainly the low resolution of the sensor but also airflow and people's movement. This meant that there would be a threshold value for how small differences in temperature that could be correctly interpreted as variation due to a human heat source and not noise.

The experiments with a glove showed that a hand in a glove gives different values in temperature depending on how long the glove has been on. This meant that future variational data collecting with a duvet would need to take the time of the duvet being on the person into account. Further tests with a duvet need to be done to calculate how long time it takes before the person has heated up the duvet enough for the temperature change to pass a threshold.

3.3. Main data collecting

The data set was created inside the bedroom of the HINT-room at Halmstad University which can be seen in Figure 13. A bed was used which measured 0.9 meters in width and 2.0 meters in length. Griddy was placed in the ceiling 2.0 meters over the bed, facing the bed. Griddy was set in a fixed position so all of the outputs could be taken from the same sensor position every time. Pythagoras theorem was used to calculate which surfaces that would be captured with the sensor (see Figure 14). Given the distance from the bed to the placement of the sensor in the ceiling and the angle of the sensors field of view (30 degrees from the center) the captured width in level with the bed was calculated to 2.3 by 2.3 meter. The sensors placement of 0.85 meters horizontally from the top of the bed gives the beds placement in the field of view that can be seen in Figure 15 and Figure 16. The calculations were roughly confirmed by looking at images where a person was captured at the border of the sensor view.

Figure 13. The bed used in data collecting. The bedroom is located in the HINT-room at Halmstad University. Griddy is visible in the ceiling over the bed.

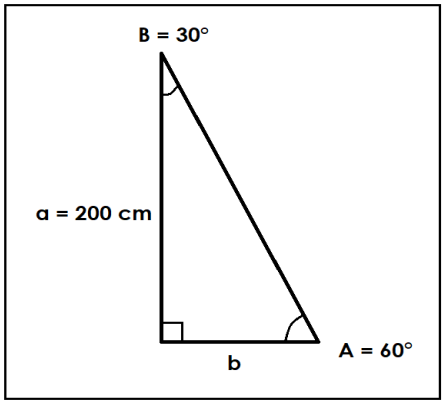

Figure 14. The sensor's field of view (b) at the level of the bed given the sensors height over the bed (a) and the angle of the sensor (B), represented as a right triangle.

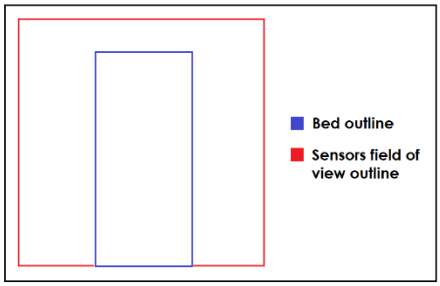

Figure 15. An illustration of the sensor's field of view at the level of the bed as seen from the top.

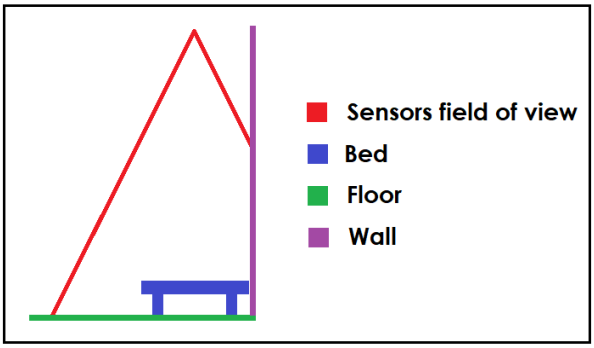

Figure 16. An illustration of the sensor's field of view as seen from the foot of the bed.

For the label "no person", captures were made from the environment with no human present in the view of the sensor. However, the surroundings of the range of the sensor varied. Sometimes a person was present just outside the range and sometimes not. Another variation was the time since a person had been in the range. Some data was collected from a maximum 30 seconds after a person had been in the bed. Other data was collected before any person had

entered the bed at all that day. For the label “person”, six different test people were presented. The people were asked to vary between different sleeping positions to simulate realistic human positioning in the bed. The test group consisted of both male and female adults with different heights and body types.

The entire data set was collected from four different days across four weeks. Data for “person” and “no person” were equally distributed over the four days. Every day the data collecting alternated with 20 captures for a label before switching labels. The order of labels for collecting varied from day to day. In total 480 images were collected of which 240 was with the label “person” and 240 was with the label “no person”. The schema for data collecting was developed with the need for data diversity in mind. It was the aim to get a dataset with as much diversity as possible when it comes to body shape, body temperature, body position and the surroundings temperature fluctuations.

Figure 17 and Figure 18 show the mean temperature for the entire main data set for the label “no person” respectively “person”. This was done by calculating the average for each individual pixel.

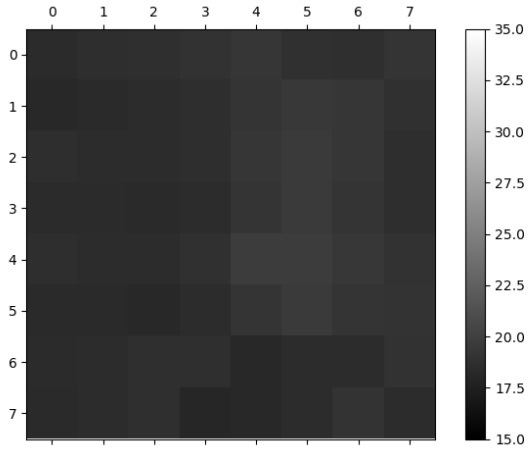

Figure 17. Mean temperature in degrees Celsius for images with the “no person” label in the main data set.

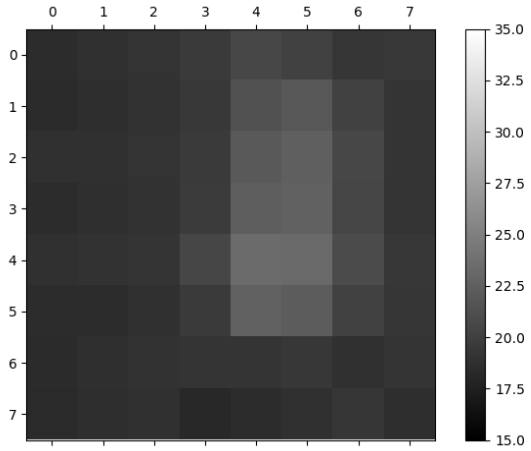

Figure 18. Mean temperature in degrees Celsius for images with the “person” label in the main data set.

The main data sets minimum and maximum temperatures for each label are represented in Figure 19. A minimum and a maximum temperature are presented for every sample. Likewise, the main data sets mean value and standard deviation temperatures are represented in Figure 20. Each point was calculated from all the pixels in an image. The data forms mainly two clusters and this is what could be expected given the difference in temperature between a person and the rooms itself. The data is also linearly separable, which was also expected. Different days and different test people result in small differences but these are not nearly as separated as the difference between a person in the frame and no person in the frame.

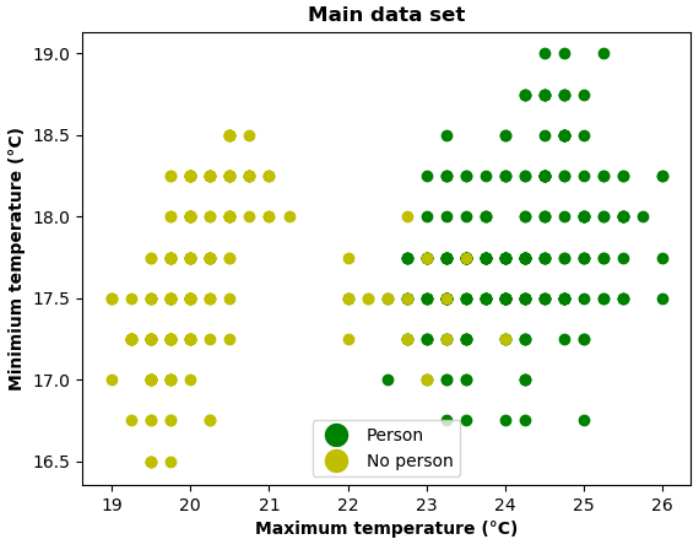

Figure 19. A scatter plot of the entire main data sets minimum and maximum temperature. The two labels are represented as different colors.

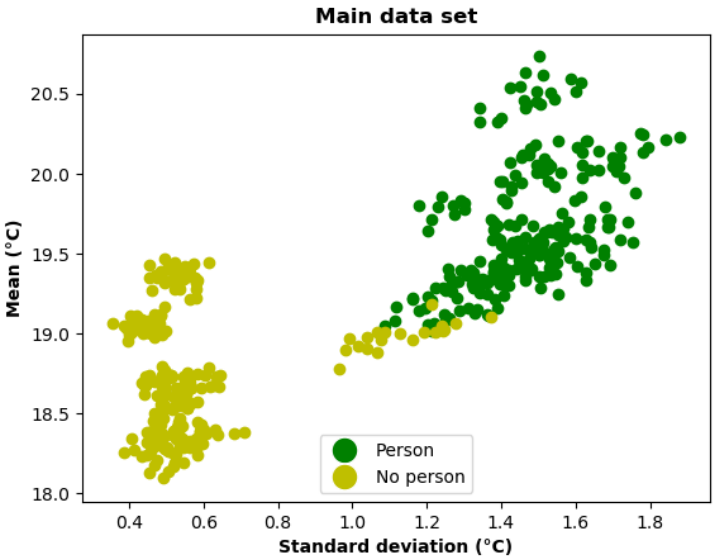

Figure 20. A scatter plot of the entire main data sets mean and standard deviation temperature. The two labels are represented as different colors.

A boxplot can be used to summarize the data set by displaying several main features. The box plot show the same data as a barplot. The line is the median value. Hence, 50 percent is on each side of the line. Within the box, 25 percent of the data is on each side of the line. Outliers are plotted as dots or small circles (Frigge, Hoaglin & Iglewicz, 1989). A boxplot for the main data set can be seen in Figure 21.

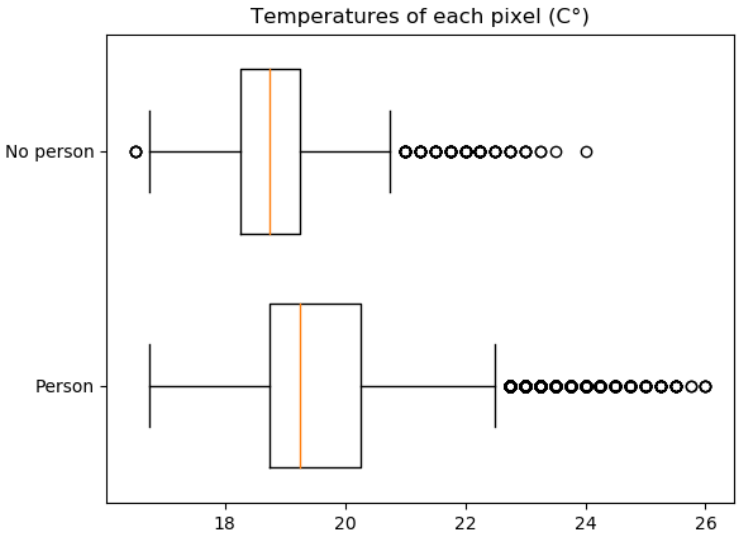

Figure 21. A boxplot of the main data sets temperatures for each pixel.

3.4. Variational data collecting

The variational data collection took place at the same location and under the same basic conditions as for the collection of the main data. The entire variational data set was collected during one day. Three separate variations were done which included an increased room temperature, a hot non-human object present and a duvet covering the person. These variations were chosen because they are expected to occur frequently in users' homes. The reason for having a variational data set is to test the algorithms on data not present in the main data set. There would be no need for a system that could not handle the most frequently occurring variations. There would also be no need for a system that had unknown limitations when it comes to frequently occurring variations.

For each of the three variations, 20 images were collected with a person and 20 images were collected without a person. The names of the labels were the same as for the main data set, "person" and "no person".

For the first variation, increased room temperature, an extra portable radiator was used to increase the room temperature from the standard temperature of 20-21 degrees Celsius to 24-25. See Figure 22 for a mean temperature of the images with no person and Figure 23 for a mean temperature of the images with a person present in the bed.

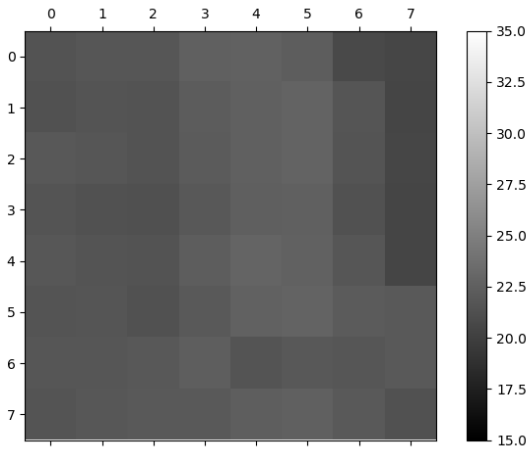

Figure 22. Mean temperature in degrees Celsius for images with increased room temperature (24-25 degrees Celsius) with the label “no person”.

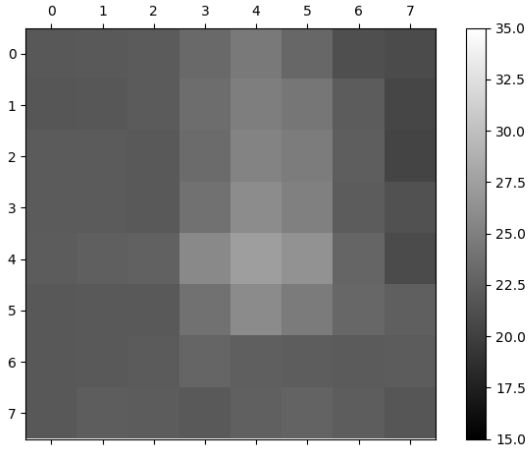

Figure 23. Mean temperature in degrees Celsius for images with increased room temperature (24-25 degrees Celsius) with the label “person”.

For the second variation, an IR-emitting object was present to represent a smaller pet. The object used was a water bottle filled up with warm water of around 37 degrees Celsius. The bottle was placed on top of the bed in various positions. See Figure 24 for a representation of the temperature in an image with the water bottle present and no person present. A mean value has not been calculated since the bottle was moved around in various positions for different images.

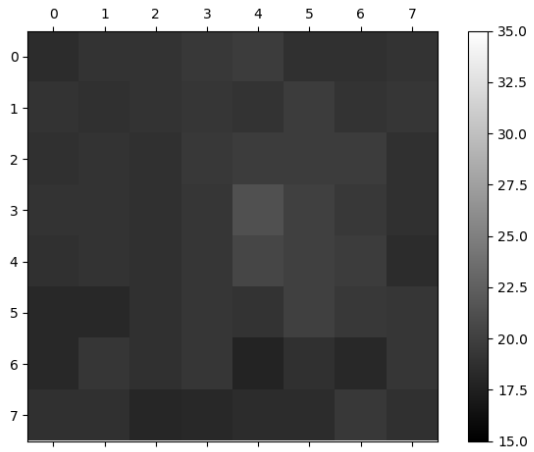

Figure 24. The temperature in degrees Celsius for an image with an IR-emitting object with the “no person” label.

The last variation was the use of a duvet on the person lying in the bed. The person was covered up to the neck by the duvet. Ten images with the label “person” were collected right after the person had gotten into bed and covered himself by the duvet. Another ten images were collected after five minutes and another ten images were collected after ten minutes. See Figure 25 for a mean temperature of the images with the duvet after it had just been put on the person. Figure 26 and Figure 27 represent the mean temperature of the images with the duvet after it had been on for five respectively ten minutes.

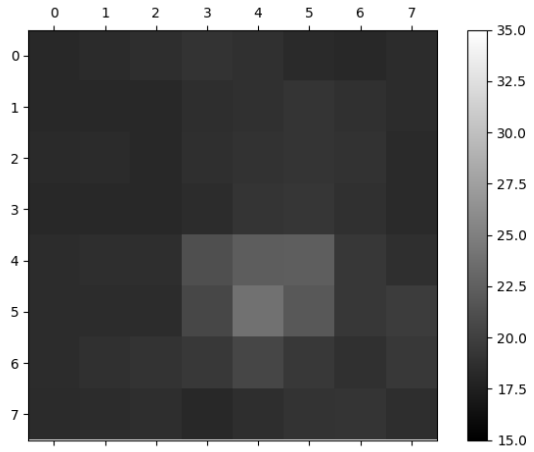

Figure 25. Mean temperature in degrees Celsius for images with a duvet that has just covered up the person.

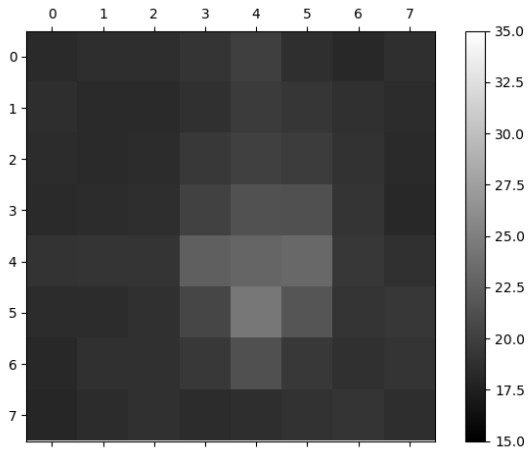

Figure 26. Mean temperature in degrees Celsius for images with a duvet that had covered up the person for five minutes.

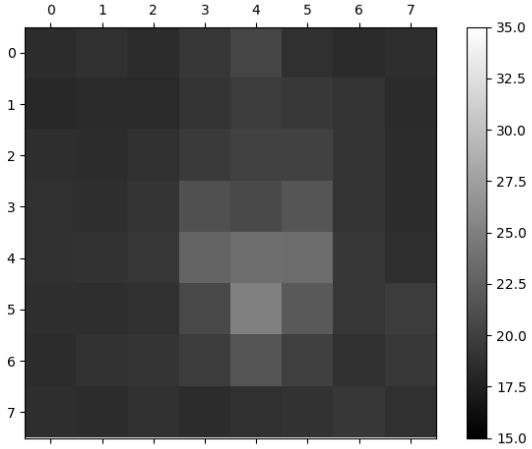

Figure 27. Mean temperature in degrees Celsius for images with a duvet that had covered up the person for ten minutes.

The main data sets minimum and maximum temperatures for each image are represented in Figure 28. The data formed several clusters and this was what could be expected given the different variations' differences. The data was also linearly separable which also was expected. The person heats up parts of the frame even with a blanket given the time allowed. The water bottle heats up parts of the frame but not nearly to the same extent as for the person.

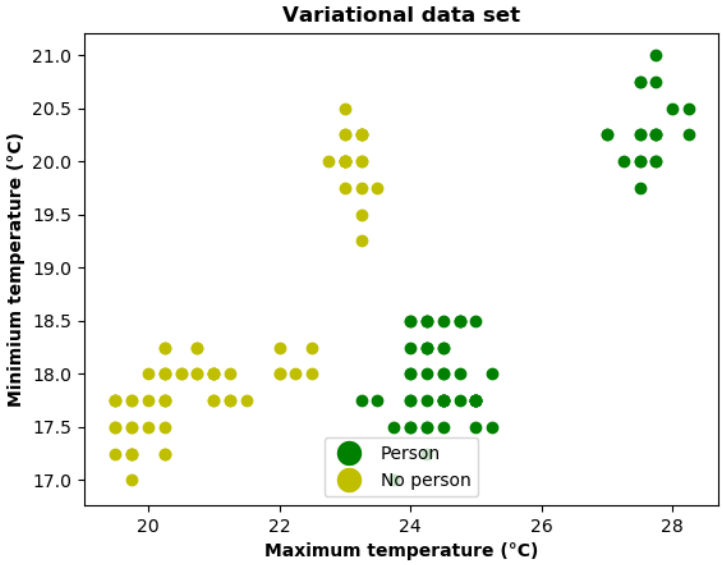

Figure 28. A scatter plot of the entire variational data sets minimum and maximum temperatures. The two labels are represented as different colors.

3.5. Algorithm training and testing

For the machine learning part, the images and labels were seen as a collection. This can be viewed in equation 34.

$$\mathbb{X} = \left\{\left(a_i, y_i\right)\right\}_{i=1}^{480} \tag{34}$$

Every image was considered a sample in the collection and every pixel a feature, resulting in the feature vector in equation 35.

$$a_i = \left(a^{(1)}, a^{(2)}, \ldots, a^{(63)}, a^{(64)}\right) \tag{35}$$

Each label is a scalar of either 0 or 1 as shown in equation 36.

$$y_i \in \{0, 1\} \tag{36}$$

The main data set was divided into 80% training data set and 20% test data set. Each set consisted of equal parts of each label which were randomly distributed between the sets. This was done by using the Scikit Learn ‘train_test_split’ method which takes in data samples and split percentage to output the two resulting arrays. By first sorting into two groups, one for each label, and then running both through the function, two equal 80/20 splits were created, one for each label. They were added to create the final train/test split (Scikit Learn).

SVM, kNN and NN were chosen as the algorithms to be tested and compared. SVM was chosen for its compatibility with linearly separable data. The kNN was chosen because it fits well with the clustering that could be seen in the data. NN was chosen because it, according to related work, performs very well on low-resolution images.

In the Scikit Learn library, all the algorithms followed the same pattern of creation. Firstly, the package which the algorithm belonged to was imported. When creating the model, the preferred parameters could be put in. Most of the parameters were optional meaning that they were set to a default value. Most of the parameters were set as default during testing. When the object is created training is done by calling 'fit' while providing an array of samples and an array consisting of corresponding labels for the samples. One sample consisted of 64 values, one for each pixel in an image. The labels were either 0 (no person) or 1 (person) (Scikit Learn).

When making predictions the 'predict' function is called on the object with an array of previously unseen samples. It returns a vector which length is the number of samples given consisting of the predicted labels. Then, by comparing a cell of the predicted array with the same position in the actual labels array, four different results could be realized. If the predicted value were correct it was TP for 'person' and TN for 'no person'. If the predictions were wrong it would be FP for a predicted value 'person' but in reality, it was 32abelled as 'no person'. Likewise, if it was a wrong prediction on an actual person, the result would be FN.

In the tests, 10-Fold Cross Validation was chosen to be used. The main reason was the fairly small data set. 10-Fold Cross Validation was used using the function 'Kfold' on the training set. This returned the training set split into ten new train test splits. The same split was used for all algorithms and all parameter tweaks. This means that all algorithms used exactly the same data folds. By iterating through these new splits and creating a new model object for each iteration, every sample could be predicted once. Then the algorithms were trained using the training set and made predictions on the unseen test set. This method of testing resulted in two different confusion matrices which could be used to compare the algorithms and different parameter options against each other. Accuracy from the 10-Fold Cross Validation was compared domestically on all three algorithms (Scikit Learn).

The best settings for each algorithm were verified with the test set. The most successful one went on to represent that algorithm against the others in the variational test. In the variational test, the algorithms were trained with the entire main data set and made predictions one the entire variational set.

The main data set was evaluated on accuracy. The accuracy was expected to be high for the best parameter settings. For this reason, the sensitivity and specificity was not displayed as results for the main data set. The accuracy was evaluated for the variational data set as well. It was not expected to be as high. The reason for expecting lower accuracy for the variational data set is because the data shows a displacement from the main data, which the model is

trained on. It was also expected to be larger variance in the accuracies. For these reasons, the sensitivity and the specificity was evaluated and displayed.

3.5.1. Support Vector Machine

The class ‘sklearn.svm.SVC’ (Scikit Learn) was chosen to represent the SVM algorithm. Among its parameters, the ‘kernel’ were tested for different options while others were left to default. The different kernel options tested were: ‘linear’, ‘poly’, ‘sigmoid’ and ‘rbf’ (Scikit Learn).

The different kernels were chosen because those are the recommended ones from the previously mentioned sources when exploring SVM.

3.5.2. k-Nearest Neighbors

The library class which was chosen for the kNN classifier was ‘sklearn.neighbors.KneighborsClassifier’ (Scikit Learn). For this class, the value of k and weights parameters were chosen. The k started at one and incremented by two each test until there was no change in the accuracy of the predictions. Odd k:s were used to prevent ties. This was done for the weights ‘uniform’ as well as ‘distance’.

3.5.3. Neural Network

For a Neural Network classifier, ’sklearn.neural_network.MLPClassifier’ (Scikit Learn) was chosen, which is a Multilayer Perceptron classifier. The parameter was the number of neurons in one hidden layer. The number of neurons were chosen according to equation 37.

$$neurons = 2^{i},\ i \epsilon \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\} \tag{37}$$

The numbers of neurons were enough to show a wide range of different test results for comparing. Even though only one neuron is expected to result in underfitting, it is still used in the test. This is because it could give knowledge to when the network starts to learn. Only one layer was tested because the data exploration showed that the data was in large linearly separable.

4. Result

In this chapter the results for all algorithms will be presented for the two data sets. Firstly, the resulting accuracy for the main data set will be presented for each algorithm's settings. This will be followed up with a validation of the best setting. For the variational data, the algorithms with their best settings will be compared with regards to accuracy, sensitivity and specificity. The results for the complete variational data set will first be shown followed by each subset individually.

4.1. Main data

For SVM the best result in accuracy with 10-Fold Cross Validation could be found with the linear, polynomial and RBF kernels. Their resulting accuracy was 0.99 when rounded to two decimals. All kernels resulting accuracy can be found in Figure 29. From this, the kernel chosen to be used on the variational data set was the linear since a more complex kernel didn't achieve higher accuracy. For verification with the test set was the accuracy with linear kernel 1.0. This can be found in Figure 30.

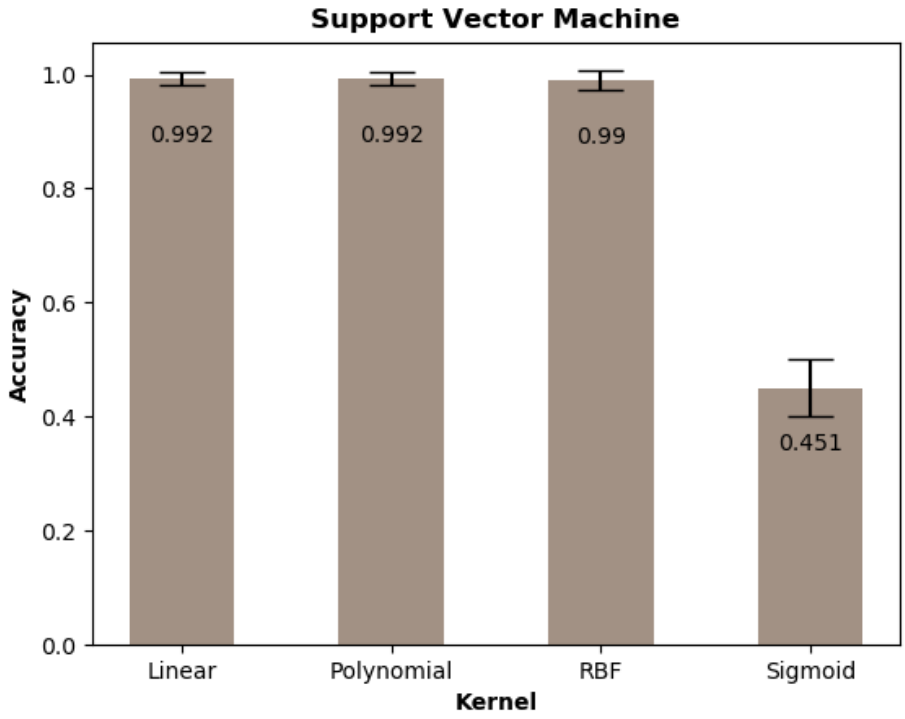

Figure 29. The resulting accuracy for Support Vector Machine. For each of the four kernels linear, polynomial, RBF and sigmoid the resulting accuracy is shown for the 10-Fold Cross Validation. The standard deviation is shown with an error bar.

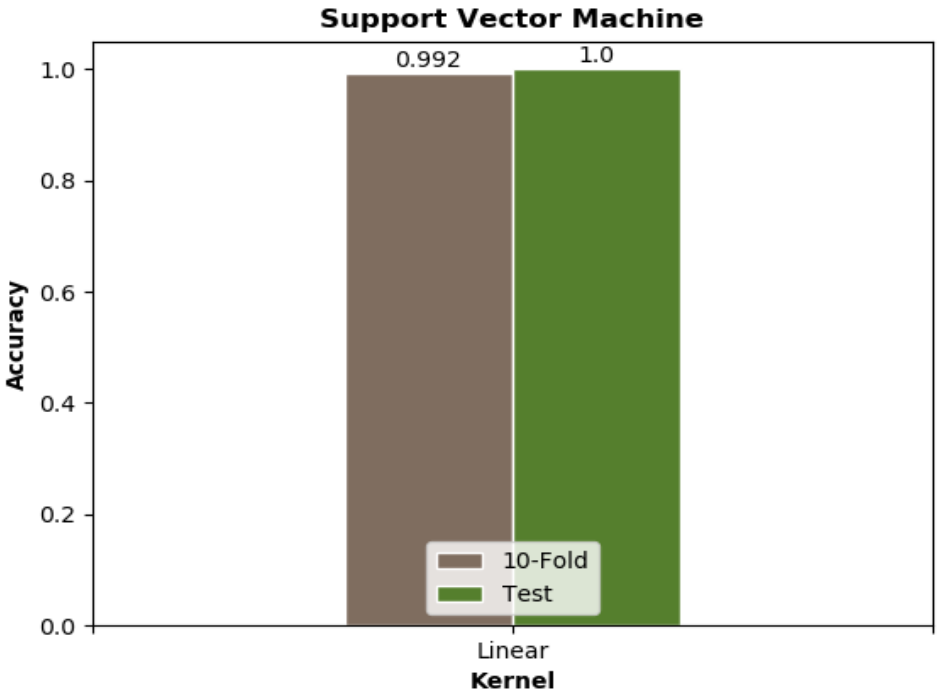

Figure 30. The resulting accuracy for Support Vector Machine with the chosen kernel linear. The resulting accuracy is shown for the 10-Fold Cross Validation and verified with the test set.

For kNN the accuracy was the same for both weights in the 10-Fold Cross Validation. Their resulting accuracy was 0.99 when rounded to two decimals. Resulting accuracy for different k can be found in Figure 31 for the weight uniform. From this, the k chosen to be used on the variational data set was 1 since a higher k didn't achieve higher accuracy. The weight chosen to be used was the uniform. Since k was set to 1 it didn't matter which weight to use. For verification with the test set the accuracy was 1.0 with k being 1. This can be found in Figure 32. The corresponding results for the weight distance can be found in Figure 33 and Figure 34.

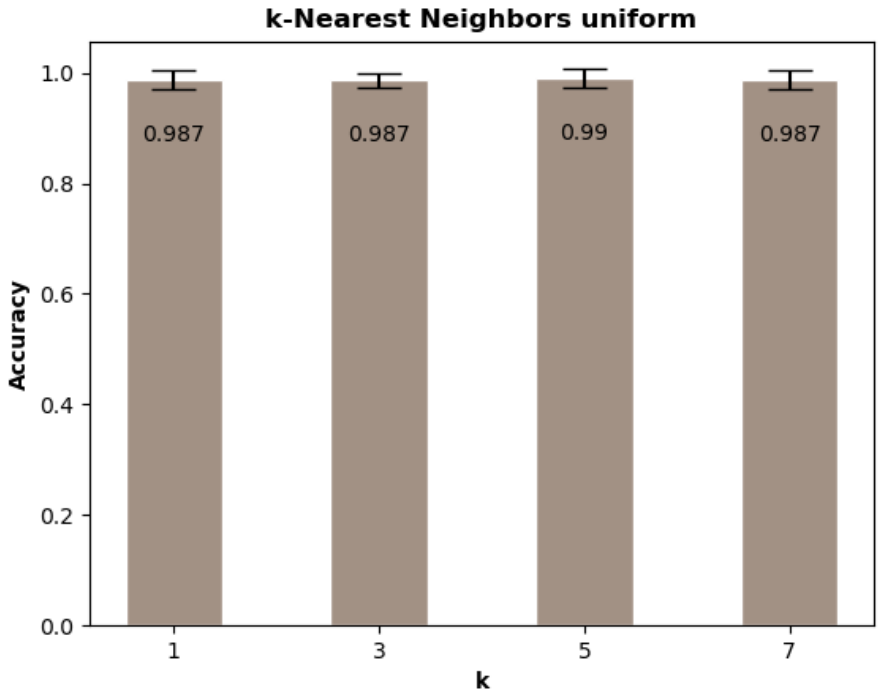

Figure 31. The resulting accuracy for k-Nearest Neighbors with the weight uniform. For each k the resulting accuracy is shown for the 10-Fold Cross Validation. The standard deviation is shown with an error bar.

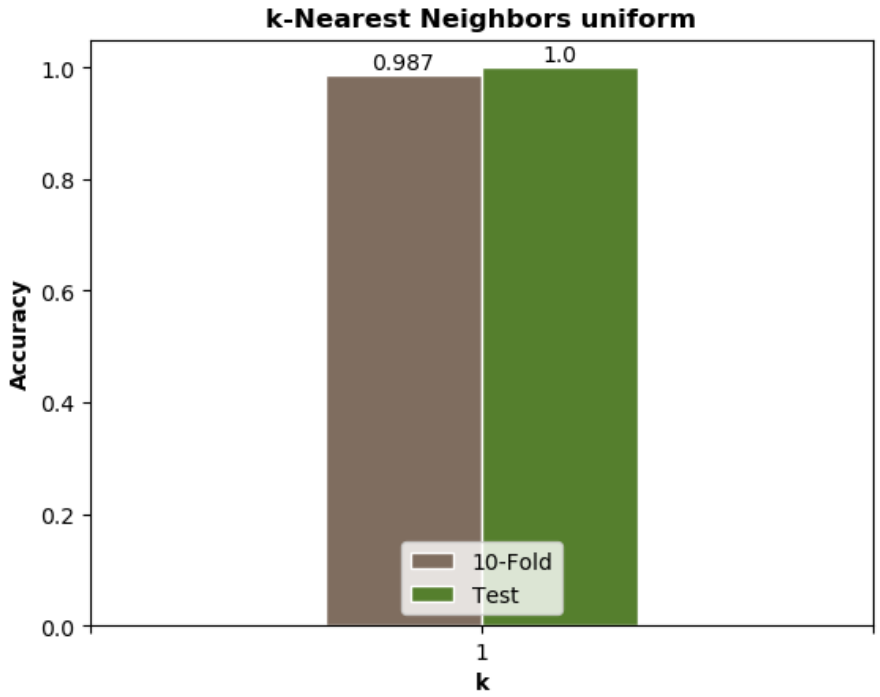

Figure 32. The resulting accuracy for k-Nearest Neighbors with the weight uniform. The chosen k is 1. The resulting accuracy is shown for the 10-Fold Cross Validation and verified with the test set.

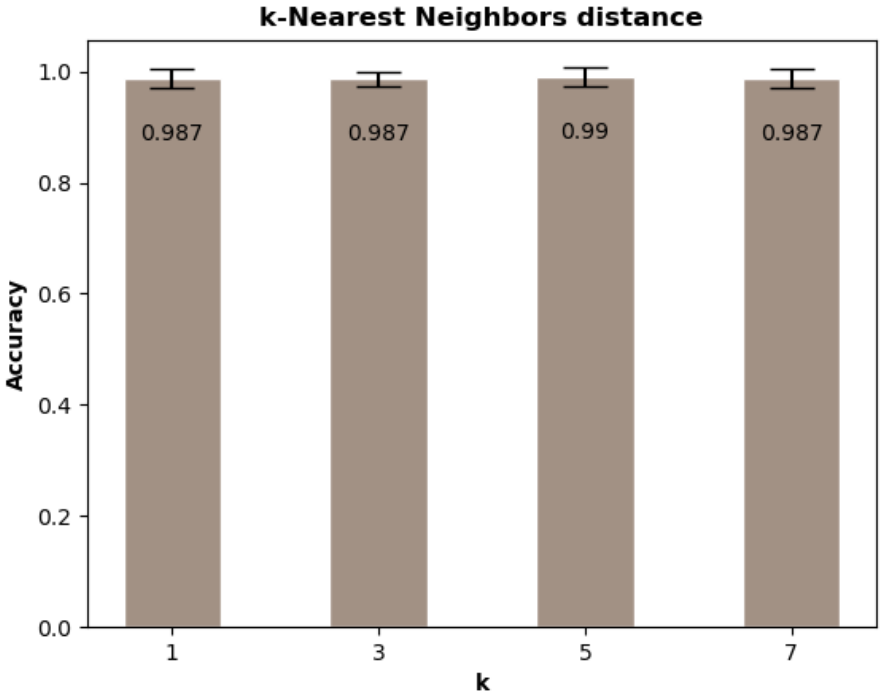

Figure 33. The resulting accuracy for k-Nearest Neighbors with the weight distance. For each k the resulting accuracy is shown for the 10-Fold Cross Validation. The standard deviation is shown with an error bar.

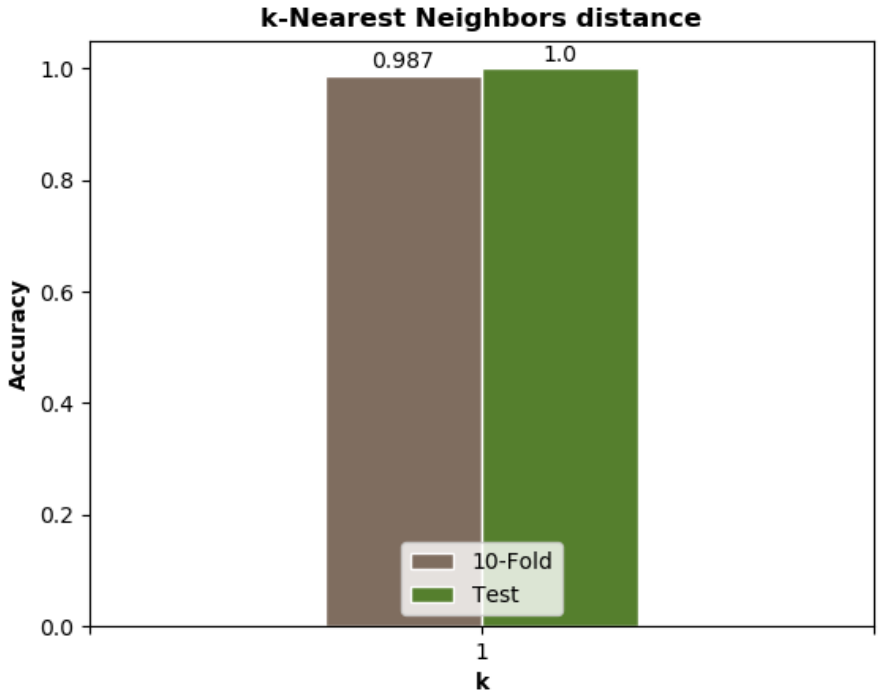

Figure 34. The resulting accuracy for k-Nearest Neighbors with the weight distance. The chosen k is 1. The resulting accuracy is shown for the 10-Fold Cross Validation and verified with the test set.

With 10-Fold Cross Validation the resulting accuracy for NN was the highest for 128 neurons in the hidden layer. It's resulting accuracy was 0.97 when rounded to two decimals. All calculated numbers of neurons resulting accuracy can be found in Figure 35. The number of neurons chosen to be used on the variational data set was 128. For verification with the test set the accuracy was 0.99. This can be found in Figure 36.

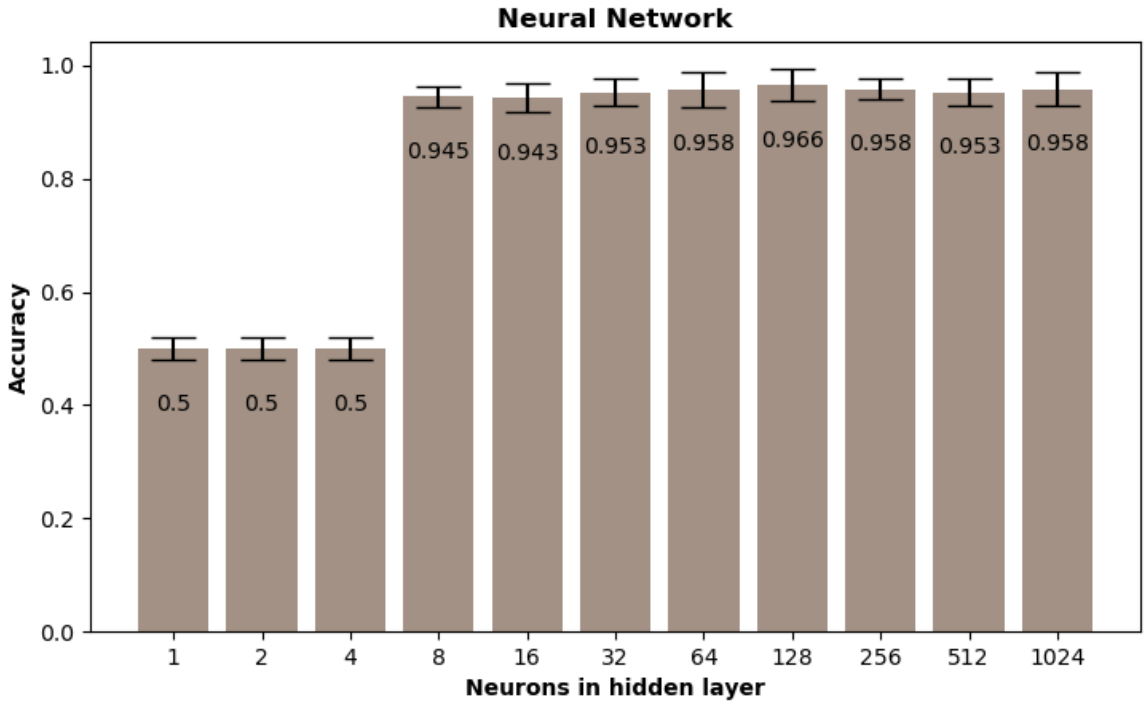

Figure 35.. The resulting accuracy for Neural Network. For each number of neurons, the resulting accuracy is shown for the 10-Fold Cross Validation. The standard deviation is shown with an error bar.

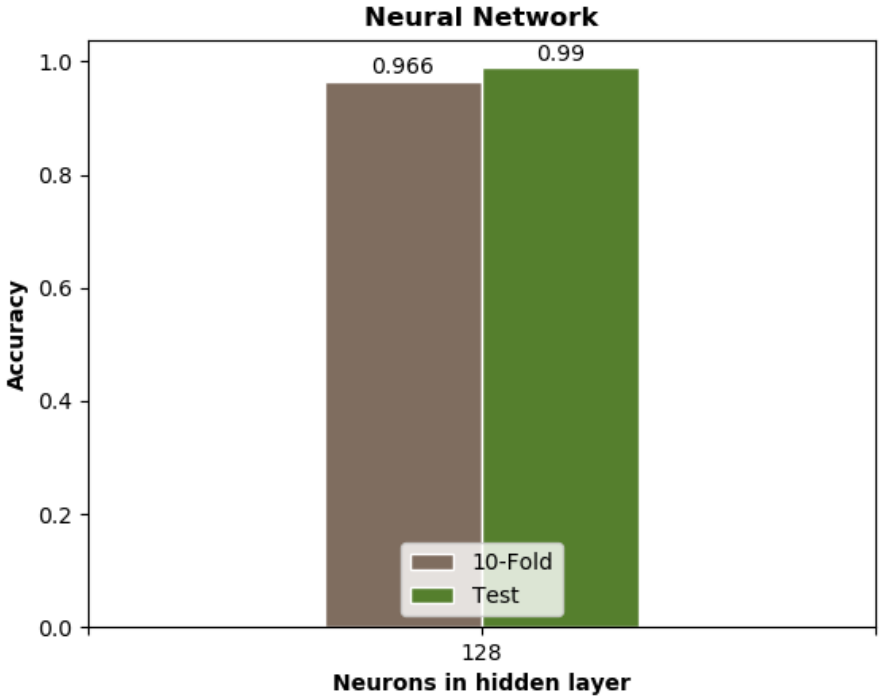

Figure 36. The resulting accuracy for Neural Network. The chosen number of neurons is 128. The resulting accuracy is shown for the 10-Fold Cross Validation and verified with the test set.

4.2. Variational data

Here are the results for the variational data set presented for the chosen parameters of the three algorithms.

For kNN the represented weight is uniform and the k is 1. The kernel in SVM is linear. The number of neurons in the hidden layer in NN is 128. Figure 37 gives an overview of how the algorithms performed on the entire variational data set in regards to accuracy. The best accuracy could be found with NN. This was 0.94 when rounded to two decimals. Figure 38 gives an overview of how the algorithms performed on the entire variational data set in regards to sensitivity and specificity. The best sensitivity (0.87) was for both kNN and NN. The best specificity (1.0) was for NN.

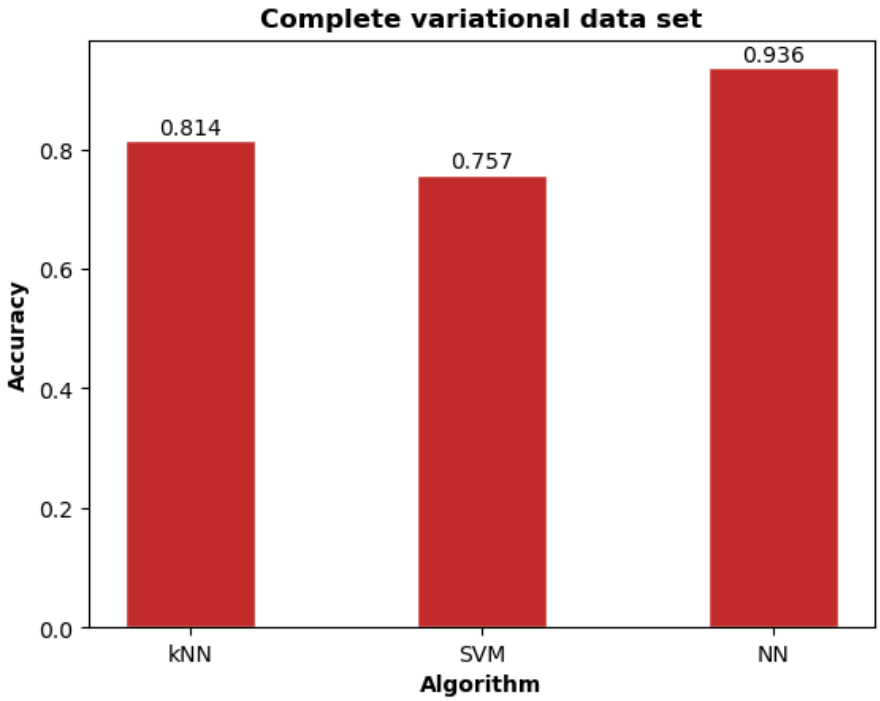

Figure 37. The resulting accuracy for the three algorithms on the entire variational data set.

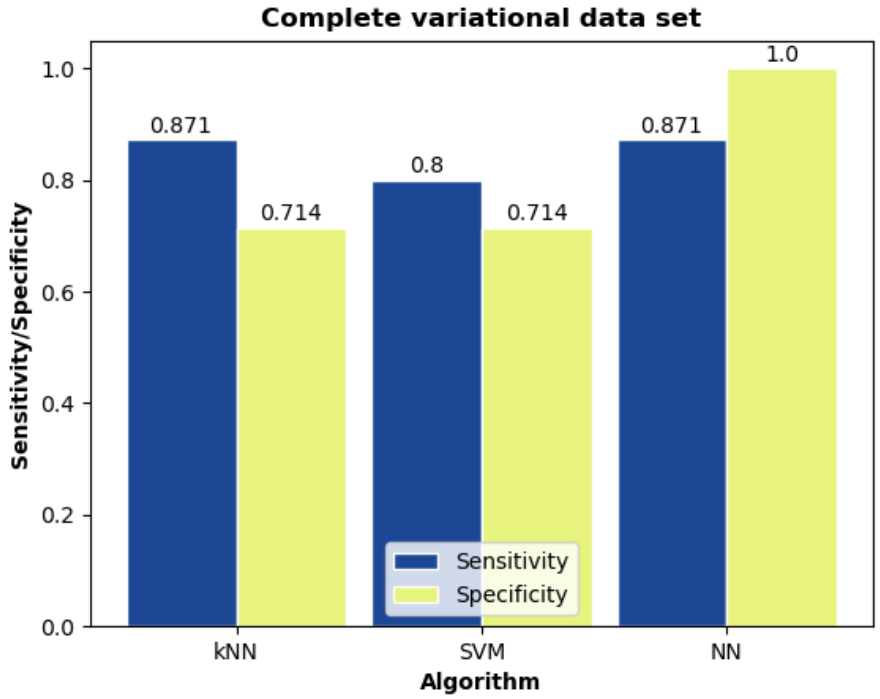

Figure 38. The resulting sensitivity and specificity for the three algorithms on the entire variational data set.

The data set with an additional heat source in the form of a filled water bottle resulted in 1.0 in accuracy for both kNN and SVM. The resulting accuracy can be found in Figure 39. The

sensitivity was 1.0 for both kNN and SVM. The specificity was 1.0 for all three algorithms. The resulting sensitivity and specificity can be found in Figure 40.

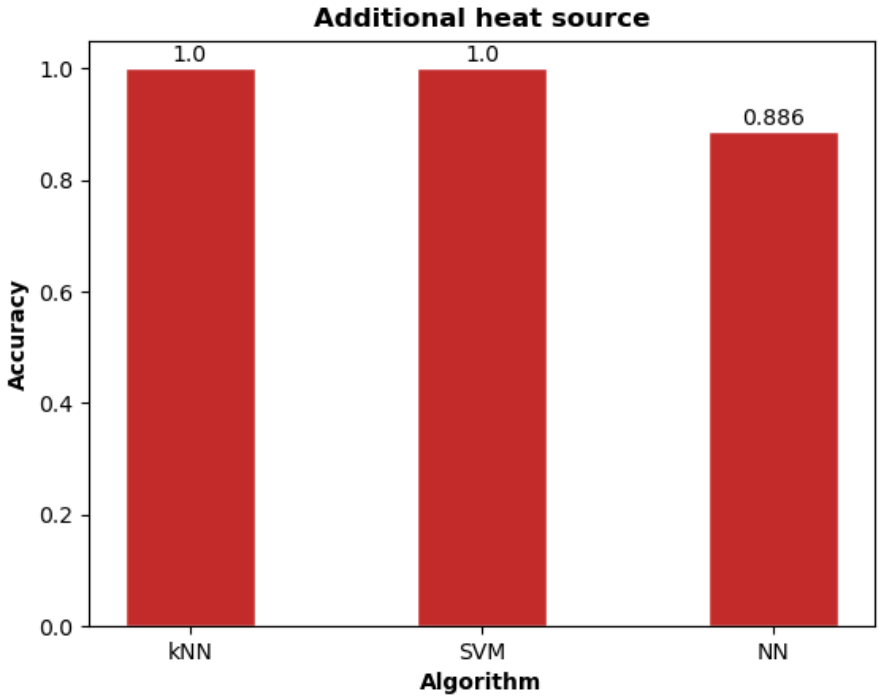

Figure 39. The resulting accuracy for the three algorithms on the variational data subset with a water bottle as an additional heat source.

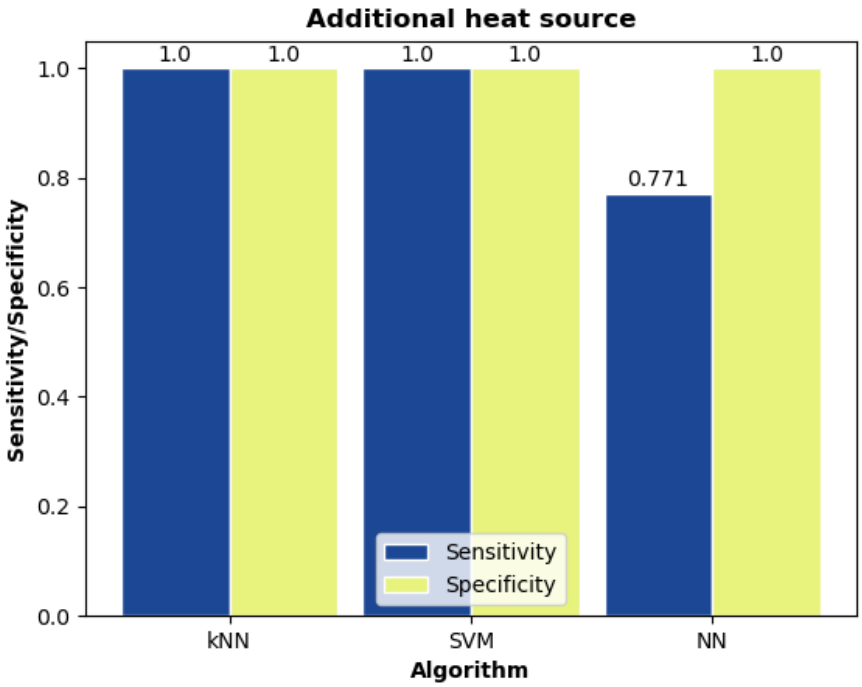

Figure 40. The resulting sensitivity and specificity for the three algorithms on the variational data subset with a water bottle as an additional heat source.

The data set with an increased room temperature (24-25 degrees Celsius) resulted in 0.9 as the highest accuracy for NN. The resulting accuracy for all algorithms can be found in Figure 41. The sensitivity was 1.0 for both kNN and SVM. The specificity was 1.0 for NN. The resulting sensitivity and specificity can be found in Figure 42.

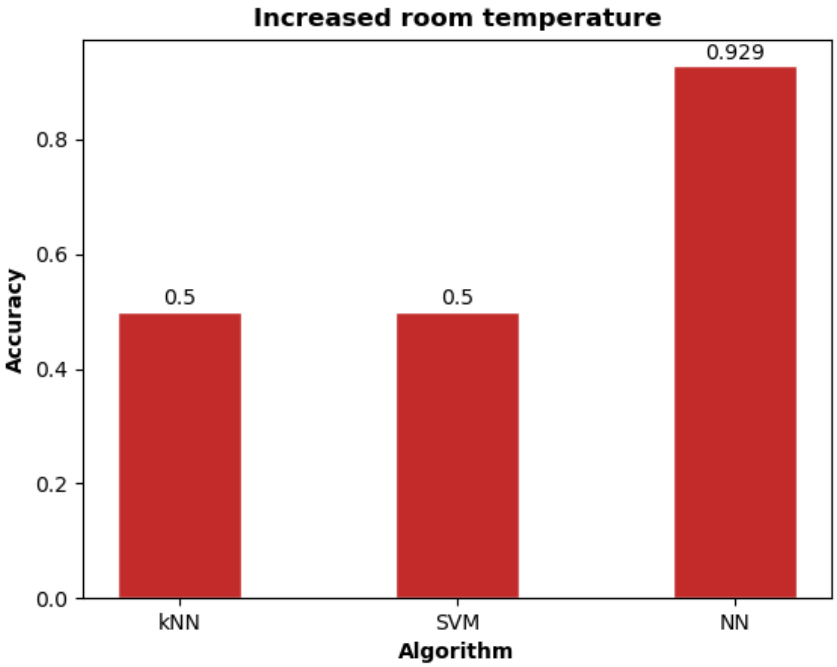

Figure 41. The resulting accuracy for the three algorithms on the variational data subset with increased room temperature to 24-25 degrees Celsius.

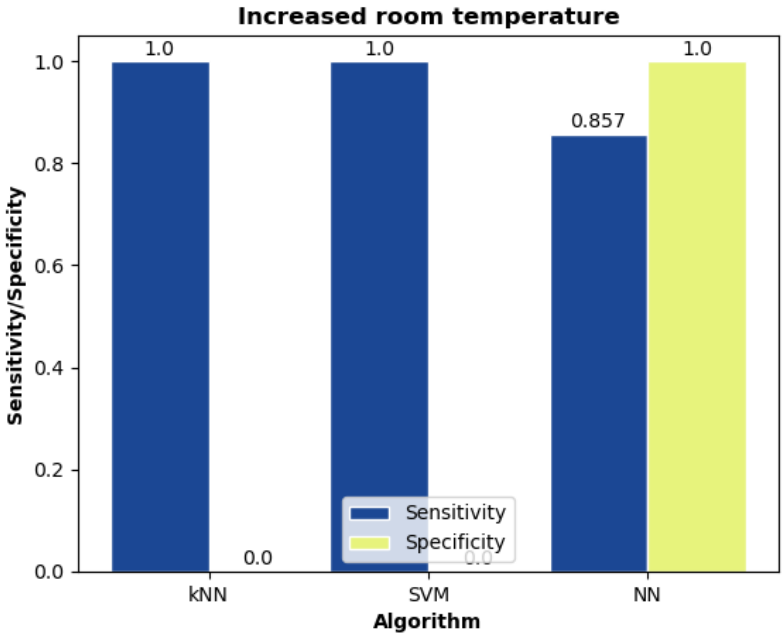

Figure 42. The resulting sensitivity and specificity for the three algorithms on the variational data subset with increased room temperature to 24-25 degrees Celsius.

The results from the data set with a duvet over the test person were divided into three parts with different times. With the duvet just put on was the highest accuracy for NN (0.9). The resulting accuracy for all algorithms can be found in Figure 43. The sensitivity was 0.9 for NN. The specificity was 1.0 for all three algorithms. The resulting sensitivity and specificity can be found in Figure 44.

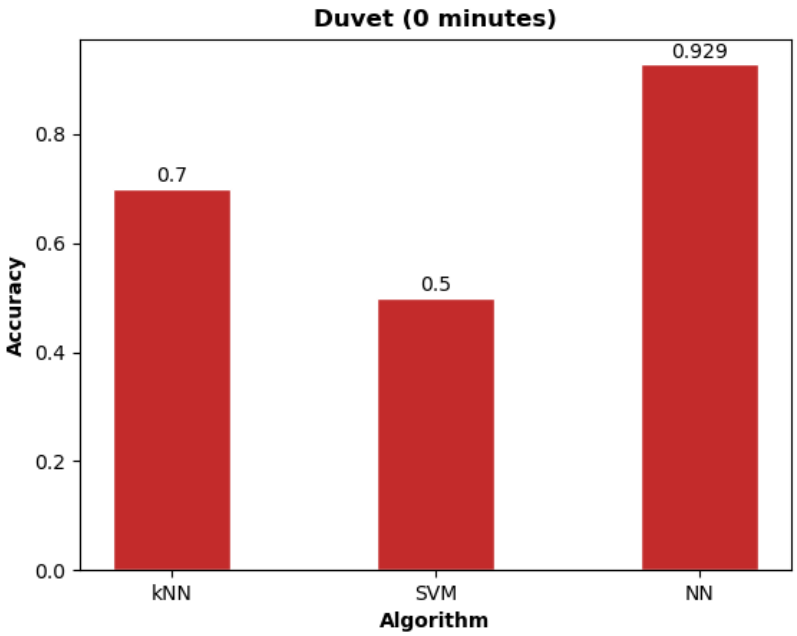

Figure 43. The resulting accuracy for the three algorithms on the variational data subset with a duvet covering the test person. The duvet had just been put on when data collecting started.

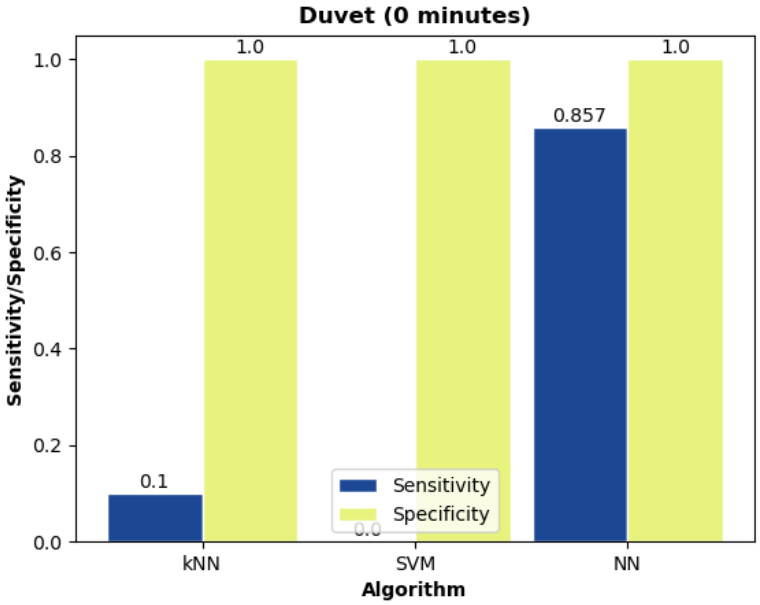

Figure 44. The resulting sensitivity and specificity for the three algorithms on the variational data subset with a duvet covering the test person. The duvet had just been put on when data collecting started.

With the duvet being on for five minutes the highest accuracy was for kNN (1.0). The resulting accuracy for all algorithms can be found in Figure 45. The sensitivity was 1.0 for kNN. The specificity was 1.0 for all three algorithms. The resulting sensitivity and specificity can be found in Figure 46.

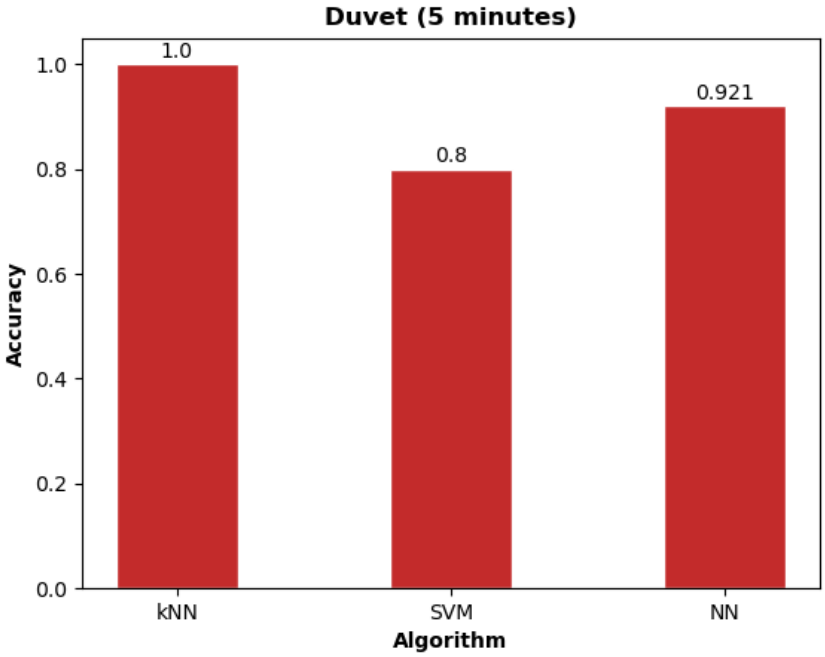

Figure 45. The resulting accuracy for the three algorithms on the variational data subset with a duvet covering the test person. The duvet had been on for 5 minutes.

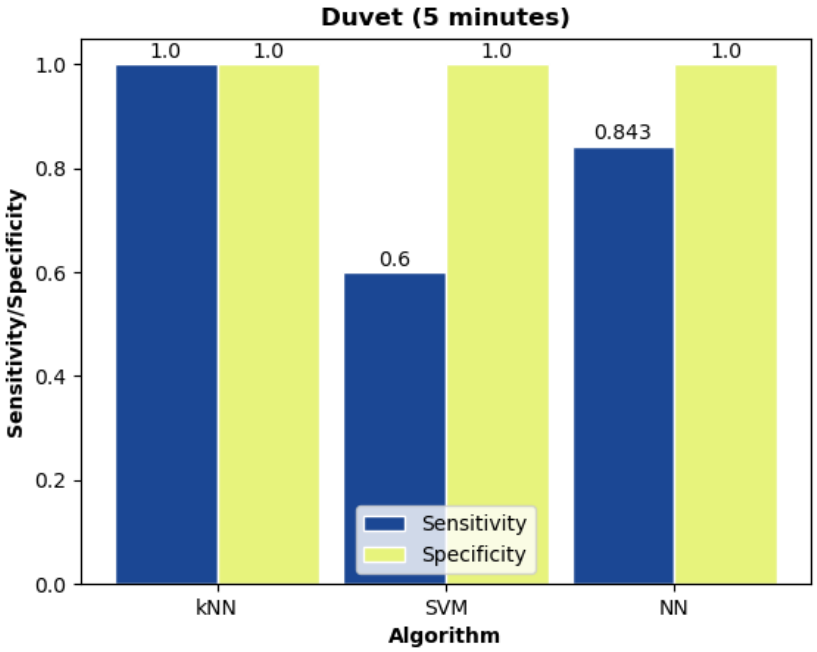

Figure 46. The resulting sensitivity and specificity for the three algorithms on the variational data subset with a duvet covering the test person. The duvet had been on for 5 minutes.

With the duvet being on for ten minutes the highest accuracy was for kNN and SVM (1.0). The resulting accuracy for all algorithms can be found in Figure 47. The sensitivity was 1.0 for kNN and SVM. The specificity was 1.0 for all three algorithms. The resulting sensitivity and specificity can be found in Figure 48.

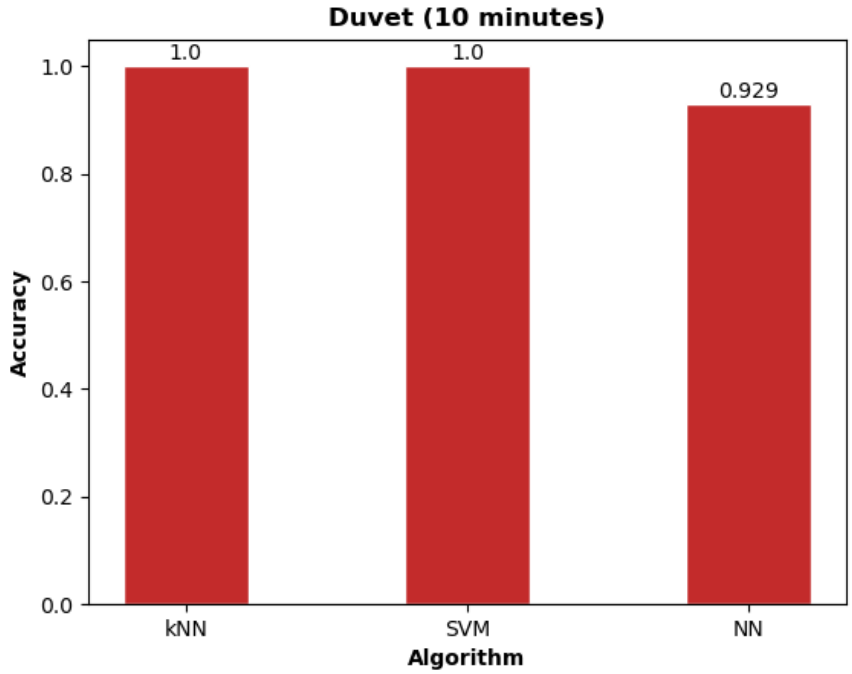

Figure 47. The resulting accuracy for the three algorithms on the variational data subset with a duvet covering the test person. The duvet had been on for 10 minutes.

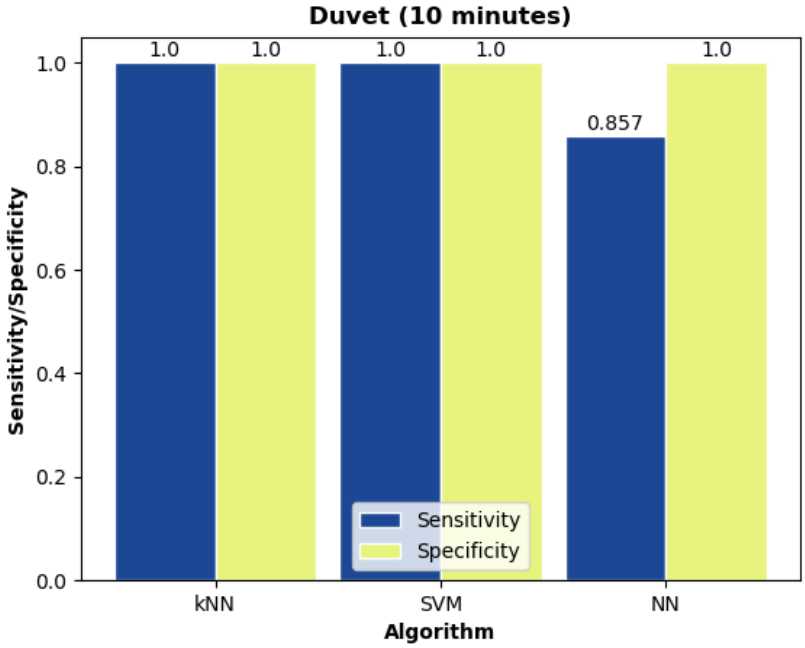

Figure 48. The resulting sensitivity and specificity for the three algorithms on the variational data subset with a duvet covering the test person. The duvet had been on for 10 minutes.

5. Discussion

To determine if Griddy could be a good alternative for monitoring of an elderly person on his bed it is not only important to answer if his presence can be sure of. It is equally as important to determine the proficiency of the system. Different users can have different requirements for the system. It is possible that the system can meet some potential users' requirements but not all. In this case, it would be important to determine which requirements the system can meet sufficiently.

Regardless of what difficulties the system faces, a false negative will always be more acceptable than a false positive. It is better that the system wrongfully decides the bed to be empty than giving the personnel false assurance that the user is in bed. The personnel should rather be directed to manually check on the user one too many times than risk missing a potentially dangerous situation. Sensitivity is therefore much more important than specificity in the evaluation of the system.

The requirement set on the system, to have an accuracy of at least 90% calculated on the main test data set for the best performing algorithm, have been met for all three algorithms. However, the results on the variational data set show that all three algorithms perform with lower accuracy when subjected to variations not presented in the training. In regards to sensitivity, the variational data sets results show troubling numbers. A further developed system in the future could contain more variation in its training data to try to resolve this issue.

In Sweden, sleeping under a duvet can be considered the standard and not a variation. Therefore, this is a highly important scenario to examine in more detail. As the results showed the duvet only has to have been on for a couple of minutes before the body had heated it up sufficiently for the sensor to distinguish the body's shape. This can result in some sort of exception or small delay for a working system.

For some users, it might be the case that a pet is sleeping in the bed alongside the owner. In this case it is of great importance that the system does not wrongfully determine the bed as occupied in the event where the user has left but the pet stayed. As the results show the system has the potential to handle pets. However, a more in-depth analysis is needed to determine which kinds of pets and of what sizes the system can handle.

The temperature in different bedrooms can vary quite a bit, therefore, it is important to determine the range in which the system performs as expected. For this purpose, an increased room temperature of 24-25 degrees Celsius was presented to the system. The results showed that a wide range of temperatures is a problem for the existing system. Improvements to the system need to be done in this regard and until then the system should not be used in any other room temperature than the standard of 20-21 degrees Celsius. A lower temperature has not been tested out and the system's abilities in this regard are therefore unknown.

Integrity and safety are important when it comes to personal information, such as the data on an elderly's whereabouts. The security of the system depends highly on the sensitivity. The integrity depends on how detailed the data is and how it will be stored. The data collected throughout the project was handled safely and with integrity in mind. The data was stored on a cloud and on the computers. The people used in the data collecting were anonymous. It was not possible to visually distinguish different test people from each other by comparing images. This does not mean that it is impossible. It would also not be entirely impossible to determine what activities take place, activities that might be uncomfortable to have identified. These are aspects important to have in mind when offering such a system.

The position of the sensor might seem controversial given that a user might feel uncomfortable laying on a bed with a sensor facing right down at him. This could in a future system be changed if wished. One example would be to have the sensor attached to the wall, facing the bed horizontally instead. Another solution would be to approach the problem from the other end. A couple of sensors in different positions could cover the whole home except the bed. If the system would find the home to be unoccupied the conclusion would be that the person is on the bed. This idea presumes that it is certain that the person has not left his home, for example since the front door has not set off an alarm.

In a potential future system the integrity would be improved compared to a system with a regular camera. The low resolution gives fewer details of for example the person's position, body type and gender. This would make it harder to identify a person. Since the collected data would not need any manual check it would not need to be stored over time. Discarding the collected data after it has been interpreted results in more integrity and more security. Such a system could also benefit the elderly care economically when the personnel resources could be more effectively distributed. This could in turn have a positive impact on the environment when unnecessary car rides to the elderly's home are prevented.

Comparing different algorithms aiming to recognize people on a bed with the Panasonic Grid-EYE has been done before. This work contributes with two new things. The first thing is the variations, which are expected to occur in the elders' everyday life. A system that has not explored the limitations of such variations would be of little help. The second new thing that this work presents is a proposal of what data a data set, for a future system, should be trained with. The data diversity needed has been explored.

After looking at the exploration of the main data set, the graphs on figure 19 and figure 20 in particular, it is possible that this data could have been successfully used to build a machine learning model based on these features only, instead of the entirety of the 64-pixel image. This is something that could be explored to see if the used amount of features were redundant. If this is the case, the integrity could perhaps be improved even more.

6. Conclusion

Sweden is expecting an aging population and a shortage of healthcare professionals in the near future. Technical solutions that contribute to safety, comfort and quick help when needed is essential. Griddy could be part of a solution that offers safety and integrity for the user. With Griddy mounted over a bed and additional software to determine if the user is on the bed or not a system could offer monitoring with little human interaction. The purpose was to determine if this system could predict human presence with high accuracy and what limitations it might have. The goal was to have an accuracy of at least 90%.

The system performed with an accuracy of 0.99 for 10-Fold Cross Validation. The best performing algorithms were SVM and kNN. The results was verified with an accuracy of 1.0 on the test set for both algorithms. All three algorithms met the goal of having an accuracy of at least 90% calculated on the main test data set.

Comparing different algorithms and different parameters has been done a lot before. The Panasonic Grid-EYE sensor has been used before for very similar data collecting as the main data set. What has been missing is the consideration of variations that occurs in everyday life.

The algorithms, with their best performing parameters, were tested on the variational data set. The best performing algorithm on the entire variational data set was NN with an accuracy of 0.94 and a sensitivity of 0.87. This showed that variations are challenging for the system but it can handle all three variations to some extent. It might be the most important variation to consider a duvet over the person in bed. The system is able to handle the duvet satisfactorily if given a bit of time. After five minutes gave kNN both accuracy and sensitivity of 1.0. In a potential system for elderly healthcare, it would be of great importance to have especially a high sensitivity in all potential situations. Therefore, it is important to include variational data in a main data set when training a system to monitor a bed.

There are more variations to consider to expand the work further. One way is to use data augmentation techniques to artificially create more variations. Since different algorithms perform well in different situations it could also be worth combining them in a future, extended, system.

7. References

Alpaydin, E 2010, *Introduction to machine learning*, MIT press, retrieved 9 March 2020, <https://kkpatel7.files.wordpress.com/2015/04/alppaydin_machinelearning_2010.pdf>.

Altman, D, Machin, D, Bryant, T & Gardner, M 2013, *Statistics with Confidence Confidence Intervals and Statistical Guidelines*, John Wiley & Sons, retrieved 12 April 2020, <https://ebookcentral.proquest.com/lib/halmstad/detail.action?docID=1813669>.

Assa Abloy 2019, *Om oss*, Assa Abloy, retrieved 2 March 2020, <https://www.assaabloy.com/sv/com/about-us/>.

Boverket 2019, *Bostadsmarknadsenkäten 2019*, Boverket, retrieved 19 April 2020, <https://www.boverket.se/contentassets/44b828c304f24b46ba69a1f293c24a97/bostadsmarknadsenkaten-2019.pdf>.

Burkov, A 2019, *The Hundred-Page Machine Learning Book*, Andriy Burkov, retrieved 2 March 2020, <http://www.mlebook.com/wiki/doku.php>.

Burman, P 1989, 'A comparative study of ordinary cross-validation, v-fold cross-validation and the repeated learning-testing methods', *Biometrika,* vol. 76, no. 3, pp. 503-514.

Chen, Z & Wang, Y 2018, 'Infrared-ultrasonic sensor fusion for support vector machine-based fall detection', *Journal of Intelligent Material Systems and Structures,* vol. 29, no. 9, pp. 2027-2039.

Ciresan, D, Meier, U & Schmidhuber, J 2012, 'Multi-column Deep Neural Networks for Image Classification', *Proceedings of the 2012 Computer Society Conference on Computer Vision and Pattern Recognition,* IEEE, Providence, United States, pp. 3642-3649, <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6248110>.

Cortes, C & Vapnik, V 1995, 'Support-Vector Networks', *Machine Learning,* vol. 20, no. 3, pp. 273-297.

Dudani, SA 1976, 'The Distance-Weighted k-Nearest Neighbor Rule', *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-6, no. 4, pp. 325-327.

Fletcher, T 2009, *Support Vector Machines Explained,* Tutorial paper, London's Global University, retrieved 24 April 20202, <http://sutikno.blog.undip.ac.id/files/2011/11/SVM-Explained.pdf>.

Frigge, M, Hoaglin, DC & Iglewicz, B 1989, 'Some implementations of the boxplot', *The American Statistician,* vol. 43, no. 1, pp. 50-54.

Gallego, AJ, Calvo-Zaragoza, J, Valero-Mas, JJ & Rico-Juan, JR 2018, 'Clustering-based k-nearest neighbor classification for large-scale data with neural codes representation', *Pattern Recognition*, vol. 74, no. 1, pp. 531-543.

Jung, Y 2018, 'Multiple predicting K-fold cross-validation for model selection', *Journal of Nonparametric Statistics*, vol. 30, no. 1, pp. 197-215.

Khan, R, Hanbury, A & Stoettinger, J 2010, 'Skin Detection: A Random Forest Approach', *Proceedings of the 2010 International Conference on Image Processing*, IEEE, Hong Kong, China, pp. 4613-4616, <http://allan.hanbury.eu/lib/exe/fetch.php?media=khan_et_al_icip.pdf>.

Kim, J, Kim, BS & Savarese, S 2012, 'Comparing Image Classification Methods: K-Nearest-Neighbor and Support-Vector-Machines', *Proceedings of the 2012 International Conference on Computer Engineering and Applications*, WSEAS, Stevens Point, Wisconsin, United States, pp. 48109-2122, <https://pdfs.semanticscholar.org/6bad/2168f4f3a0d3f51d33a2d85a7efb4f76f412.pdf>.

Liu, CL, Lee, CH & Lin, PM 2010, 'A fall detection system using k-nearest neighbor classifier', *Expert systems with applications*, vol. 37, no. 10, pp. 7174-7181.

Marsland, S 2015, *Machine learning: An Algorithmic Perspective*, CRC press, retrieved 9 march 2020, <https://doc.lagout.org/science/Artificial%20Intelligence/Machine%20learning/Machine%20Learning_%20An%20Algorithmic%20Perspective%20%282nd%20ed.%29%20%5BMarsland%202014-10-08%5D.pdf>.

Panasonic 2019, *Infrared Array Sensor Grid-EYE*, Panasonic, retrieved 2 March 2020, <https://industrial.panasonic.com/cdbs/www-data/pdf/ADI8000/ADI8000C66.pdf>.

Panchal, G, Ganatra, A, Kosta, YP & Panchal, D 2011, 'Behaviour Analysis of Multilayer Perceptrons with Multiple Hidden Neurons and Hidden Layers', *International Journal of Computer Theory and Engineering*, vol. 3, no. 2, pp. 332-337.

Phoniro, *Våra innovativa IT-lösningar frigör tid och borgar för god kvalitet*, Phoniro, retrieved 2 March 2020, <https://www.phoniro.com/sv/om-phoniro/>.

Pontes, B, Cunha, M, Pinho, R & Fuks, H 2017, 'Human-Sensing: Low Resolution Thermal Array Sensor Data Classification of Location-Based Postures', *Proceedings of the 2017 Distributed, Ambient and Pervasive Interactions*, DAPI, Vancouver, Canada, pp. 444-457, <https://link.springer.com/chapter/10.1007/978-3-319-58697-7_33>.

Rogalski, A 2010. *Infrared Detectors*, CRC Press, retrieved 25 May 2020, <https://books.google.se/books?hl=sv&lr=&id=0VUJSafhaK0C&oi=fnd&pg=PP1&dq=+Infr

ared+Detectors+rogalski&ots=F_s2YL_r4b&sig=1VjxivB2iNssf-ppMrPLOXBsFw0&redir_esc=y#v=onepage&q=Infrared%20Detectors%20rogalski&f=false>.

Scikit Learn, *Getting Started*, Scikit Learn, retrieved 14 April 2020, <https://scikit-learn.org/stable/getting_started.html>.

Scikit Learn, *User Guide - Nearest Neighbors*, Scikit Learn, retrieved 14 April 2020, <https://scikit-learn.org/stable/modules/neighbors.html>.

Scikit Learn, *User Guide - Support Vector Machines*, Scikit Learn, retrieved 14 April 2020, <https://scikit-learn.org/stable/modules/svm.html>.

Scikit Learn, *User Guide - Neural network models (supervised)*, Scikit Learn, retrieved 14 April 2020, <https://scikit-learn.org/stable/modules/neural_networks_supervised.html>.

Scikit Learn, *User Guide - Cross-validation: evaluating estimator performance,* Scikit Learn, retrieved 14 April 2020, <https://scikit-learn.org/stable/modules/cross_validation.html>.

Scikit Learn, *User Guide - Metrics and scoring: quantifying the quality of predictions,* Scikit Learn, retrieved 28 April 2020, <https://scikit-learn.org/stable/modules/model_evaluation.html>.

Shetty, AD, Shubha, B & Suryanarayana, K 2017, 'Detection and Tracking of a human Using the infrared thermopile Array sensor –"grid-EYE"', *Proceedings of the* 2017 *International Conference on Intelligent Computing, Instrumentation and Control Technologies,* ICICICT, Kannur, Kerala, India, pp. 1490-1495, <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8342790>.

Stathakis, D 2009, 'How many hidden layers and nodes?', *International Journal of Remote Sensing*, vol. 30, no. 8, pp. 2133-2147.

Statistics Sweden (SCB) 2016, *Stora insatser krävs för att klara 40-talisternas äldreomsorg*, Statistics Sweden, retrieved 10 March 2020, <https://www.scb.se/hitta-statistik/artiklar/2016/Stora-insatser-kravs-for-att-klara-40-talisternas-aldreomsorg/>.

Statistics Sweden (SCB) 2019, *Drygt 4,9 miljoner bostäder i landet*, Statistics Sweden, retrieved 10 March 2020, <https://www.scb.se/hitta-statistik/statistik-efter-amne/boende-byggande-och-bebyggelse/bostadsbyggande-och-ombyggnad/bostadsbestand/pong/statistiknyhet/bostadsbestandet-2018-12-31/>.

Statistics Sweden (SCB) 2018, *Så gör SCB en befolkningsframskrivning*, Statistics Sweden, retrieved 7 April 2020, <https://www.scb.se/hitta-statistik/artiklar/2018/sa-gor-scb-en-befolkningsframskrivning/>.

Statistics Sweden (SCB) 2017, *Konsekvenser för försörjningskvoten*, Statistics Sweden, retrieved 19 april 2020, <https://www.scb.se/hitta-statistik/artiklar/2017/Farre-maste-forsorja-fler/>.

Trofimova, AA, Masciadri, A, Veronese, F & Salice, F 2017, 'Indoor Human Detection Based on Thermal Array Sensor Data and Adaptive Background Estimation', *Journal of Computer and Communications,* vol. 5, no. 4, pp. 16-28.

Elias Josse
Computer Engineer

Amanda Nerborg
Computer Engineer

HALMSTAD UNIVERSITY

PO Box 823, SE-301 18 Halmstad
Phone: +35 46 16 71 00
E-mail: registrator@hh.se
www.hh.se