MASTER'S THESIS 2020

Determining Room Occupancy with Machine Learning Techniques

Daniel Myhrman

ISSN 1650-2884
LU-CS-EX 2020-12

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2020-12

Determining Room Occupancy with Machine Learning Techniques

Daniel Myhrman

Determining Room Occupancy with Machine Learning Techniques

Daniel Myhrman
Daniel.Myhrman.629@student.lu.se

May 7, 2020

Master's thesis work carried out at
Sony Mobile Communications AB.

Supervisors: Pierre Nugues, Pierre.Nugues@cs.lth.se
Fredrik Karlsson, Fredrik.Karlsson@sony.com
Magnus Persson, Magnus.Persson@sony.com

Examiner: Jörn Janneck, Jorn.Janneck@cs.lth.se

Abstract

Today, meeting and conference rooms may be too big or small compared to the actual number of people using them and it would bring benefits to know to what extent they are used. In this thesis, occupancy count has been estimated by training models on passive infrared (PIR) sensor and booking data. I built a system with two artificial neural network models, where a binary classification model predicts the occupancy state of rooms, i.e. if it is empty or not. If a room is considered occupied, a subsequent regression model predicts the number of occupants.

Solutions exist today where cameras are used to calculate occupancy count, however these can be intrusive, costly and computationally expensive. PIR sensors placed in rooms can detect movement, but its data cannot directly be translated into occupancy count. To solve this and determine the occupancy level, I developed different neural network regression models and trained them on manually collected ground truth data from real meetings. The performance of the models were then compared and the best results were obtained with a bidirectional long-short term memory architecture. It reaches a mean squared error of 2.27 and mean absolute error of 0.94. It predicts the number of people with an error margin of one individual at 85% respectively 49% accuracy for occupants ranging from 1 to 7 and 8 to 14. At an error margin of two individuals the results for the same intervals are 94% respectively 66% accuracy.

Keywords: room occupancy, artificial intelligence, machine learning, deep learning, regression, PIR

Acknowledgements

I would like to thank Pierre Nugues at LTH for his continuous advice during the project. I would also like to thank Fredrik Karlsson and Magnus Persson at Sony for letting me do this interesting thesis and their help and feedback, and also the others in the Nimway team.

Another thank you to Mattea and Suleyman for helping me with the ground truth collection which made training the model possible.

Contents

Chapter 1
Acronyms

PIR Passive Infrared Radiation

API Application Programming Interface

IDE Integrated Development Environment

LSTM Long Short-Term Memory

BiLSTM Bidirectional Long Short-Term Memory

MSE Mean Squared Error

MAE Mean Absolute Error

RMSE Root Mean Squared Error

MLP Multi-layer Perceptron

SVM Support Vector Machine

Chapter 2
Introduction

To utilize space and resources efficiently, it would be beneficial for companies to know to what degree their meeting and conference rooms are used. The aim of this thesis was to determine the number of people in rooms by using machine learning techniques. I used PIR motion sensor and booking data together with static data, i.e. number of chairs in rooms, to predict the occupancy level of meeting and conference rooms where we collected ground truth data from real world meetings to train and test machine learning models.

2.1 Problem Formulation

The challenge of this thesis is to determine the number of people occupying meeting and conference rooms at a time and to count it automatically by using machine learning techniques. It may not be sure that meeting and conference rooms are the optimal size and can be too big or small compared to how they actually are used. Therefor it would be beneficial to have information about how many people occupy a room at a time. This information can also be studied and used to improve other aspects of an office, such as to reduce energy consumption and cleaning costs. The estimation of room occupancy with PIR sensor data is a task where the problem is to go from low level data from the sensors to higher level data of number of occupants. This a great fit for machine learning models that may find patterns in data which can be hard for a human to see. The aim of the project is to use and compare different neural network models since these have not nearly been explored as much as other machine learning models within occupancy level estimation. Neural networks have been shown to perform very well in other areas such as image recognition and natural language processing, and in specific in this thesis I wanted to study three different types of neural network models that have been shown in other areas to perform well, namely a feed forward model as a baseline, LSTM and a BiLSTM models and to train different models of these architectures and compare the results. There are many sources of data that more or less can explain the occupancy level of a room, for example cameras, CO2 sensors and Wi-Fi networks. However in super-

vised machine learning it is required to have answers to this type of data in the form of the number of people occupying the room at given times. This data is called the ground truth and when it is established it can be used to train and test models. But in this project, this data was not available and had to be collected to be able to train a machine learning model. The main objective of the thesis has been to study different machine learning techniques to be able to determine the number of people in meeting rooms in a way that gives the best results. In this thesis the main source of data is PIR motion sensors, explained in section 1.3, and could be combined with booking data from the Nimway team at Sony Mobile Communications.

2.2 PIR Sensors

In the project, the main source of data was PIR sensors. The PIR sensor can detect movement by sensing changes in infrared radiation. Infrared light is radiated from the human body, but can generally not be seen by the human eye. It works by having voltages change as it notices changes in radiation in its field of view (Chodon et al., 2013). In the project, I used two different types of sensors to get data about the amount of movement in rooms. The first one is a motion sensor, as shown in figure 1.1, and can detect motion in a room. The other one is a passage motion sensor (figure 1.2) which is mounted by the door frame of rooms and looking downwards to detect motion close to door openings.

Figure 2.1: Motion sensor

Figure 2.2: Passage motion sensor

2.3 Process of Machine learning

Machine learning is a set of techniques that can be used to train a model to thereafter be able to perform predictions. Models are trained by weighting features in training data to map them in the best way to corresponding targets. A simple example of a model could be one trained to determine if features correspond to an apple or orange. In such a case features could be weight, texture and color. By letting the model see large amount examples of what constitutes an apple and an orange, it should eventually be able to distinguish between the two.

When implementing a machine learning system, the first step is usually to study the raw data and select relevant features. Thereafter the data can be shuffled and split into training and test data. Doing the split early on in the pipeline is important to avoid data leakage where data enters the test set from the training set. Such a flaw could give good results on the test set, but may perform worse in practice. When the the split has been performed the data can be preprocessed so that the data is in a format that is optimal for the model to process. After preprocessing, the model can be trained when the data and targets, depending on the problem, have been separated. When the model has been trained it is possible to input new, unseen, data that the model let it predict targets. Initially, when predicting on the test set the results will probably not be perfect, therefor the model should be evaluated so that its performance can be improved.

2.4 Related Work

Within the field of machine learning, there is a range of different studies on room occupancy. Some study binary occupancy, i.e. if a room is vacant or not, and in depending on the study, different sources of data are used. The work of Hailemariam et al. (2011) studies occupancy detection in a one minute time window. They tested combining different features from light, CO2 and motion sensors, but achieved the best results by solely using motion sensors where they reached an accuracy of 98.4%.

Other studies have looked at counting the number of people in a room and in (Raykov et al., 2016) a single PIR sensors were used to predict the number of occupants. They reached an accuracy of 84% for occupants under eight individuals and 97% for occupants ranging from 8 to 14 with a 2 error margin, when predicting in 20 minute windows. The results do not reach the state of the art in occupancy counting which can be achieved using cameras. However, they show that it is possible to determine the occupancy to a certain degree with a single PIR sensor and that they can be used for tasks outside their ordinary use area of motion detection.

There is also a study, Jiang et al. (2012), on estimating occupancy levels using purely carbon dioxide (CO2) data. They predicted the number of occupants in an office room with 24 cubicles and 11 seats. In the study the performance of three different models were compared, i.e. an feature scaled extreme learning machine where..... reached an accuracy of 94% when the prediction error is less than 4 individuals.

In Mohottige and Moors (2018), Wi-Fi data was used to estimate the occupancy level of university campus lecture theatres with machine learning. They used a regression model to predict the number of people in lecture theatres, and showed that Wi-Fi data such as signal strength, number of retries and amount of data sent can be used to estimate the number of people. Their system reached an RMSE of 25.4 and an MAE of 17.5 with a linear regression model, where occupants were between 0 and 250.

Another study from 2012, Mamidi et al. (2016) shows that number of people in a room can be determined by using a variety of data from motion, CO2, light, sound and door sensors. They trained and tested different models including linear regression, multi-layer perceptron (MLP) and support vector machines (SVM). And they tested a variety of different sets of many different features. One of the new ideas of of their approach was to filter out background noise by introducing additional CO2 features. This was introduced to let the model be able to consider CO2 coming from other rooms in the building. They received the best results with a MLP model with an root mean squared error (RMSE) of 0.82 while linear regression reached an RMSE of 0.89 on the same feature set with the number of occupants ranging between 0 and 10.

When it comes to counting the number of people in an office with a camera based approach there has been research done in Benezeth et al. (2011) where they reached an accuracy of 93% when counting the number of occupants in an office environment and 83% accuracy in a corridor environment. The corridor environment was harder to predict since there could be an overlap of people in the field of view, i.e. a person behind another.

Chapter 3

Data

3.1 Features

The features used for the models are based on the two type of motion sensors explained in section 2.2 together with booking data in the form of number of invites to meetings. Also, static data such as the number of chairs in a room is included as a feature. In total there are five features and a brief overview of them can be seen in table 3.1 below.

Table 3.1: Features and their type

Feature	Type
Motion event count	Integer
Passage event count	Integer
Size	Integer
Number of attendees	Decimal number
Binary occupancy status	Boolean

3.1.1 Sensor Data

The sensor data has many different features, however just a subset of them were used for training the model since many were not relevant for predicting room occupancy. The used features are listed below.

Motion Event Count

The sensors track the amount of motion that occurs in its field of view and the sum of these motion events are called event count. The event counts come either from a motion or passage motion sensor and *motion event count* is event counts from motion sensors.

Passage Event Count

Event counts that come from passage motion sensors, mounted at the top of door openings, are summed for each time step and room, the sum is the feature called passage event count.

3.1.2 Room Data

Size

The number of chairs in the meeting rooms is included as a feature and is named *size*. The rooms have sizes ranging from 4 to 14 chairs.

3.1.3 Booking Data

Number of Attendees

Booking data includes the number of people invited to the meeting along with start and end times for meetings. From this data the *number of attendees* feature could be derived which is the number of people invited to a meeting and can be anywhere from zero to around 38.

3.1.4 Occupancy Data

Binary occupancy status

Given that sensors can detect motion, data consisting of the binary occupancy state for rooms, i.e. if a room is occupied or not, can be derived and this feature can support cases where event counts are not available.

Chapter 4
Theory

4.1 Articial Neural Networks

Nature has been inspiration for many ideas and inventions, including artificial neural networks which have been designed by looking at the way the brain works. These networks are built up by units that are inspired by the neural networks in our brain built up by neurons (nerve cells). In a neural network, they are represented by units or cells that can build up layers in the network, these cells can then propagate data to other units.

4.1.1 Activation functions

The neurons that make up the layers of a neural network each have an activation function that determines both if there will be an output and the magnitude of the output to send to the next layer. These functions can be assigned to a whole layer, and can be linear like the ReLu or non-linear like the sigmoid activation function described below. Nwankpa et al. (2016)

Sigmoid

The sigmoid is among the most popular activation functions used in neural networks. (Ngah et al., 2016) The function gives an output from 0 to 1 and has a non-linear curve due to its properties (equation 3.1). The function is depicted in figure 3.1. A model that is to predict with the sigmoid function could be if it is to choose between different cases. An example could be to predict if a fruit is an apple, banana or orange, each of these fruit possibilities would be assigned a different value between zero and one and the one with the largest value would be the most probable.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{4.1}$$

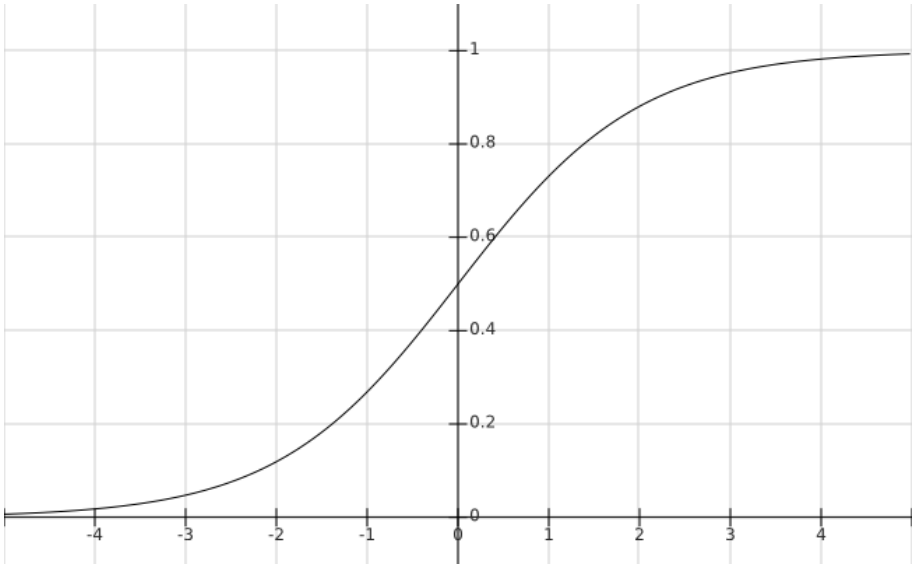

Figure 4.1: The sigmoid activation function.

ReLU

The rectified linear unit (ReLU), see figure 3.4, is an activation function that can be used in neural network layers. The functions output is either zero or greater, equation 3.2, and thus can be used if a models goal is to predict non-negative values and is used in this thesis to predict the number of occupants as the occupancy count can not be negative.

$$f(x) = x^{+} \tag{4.2}$$

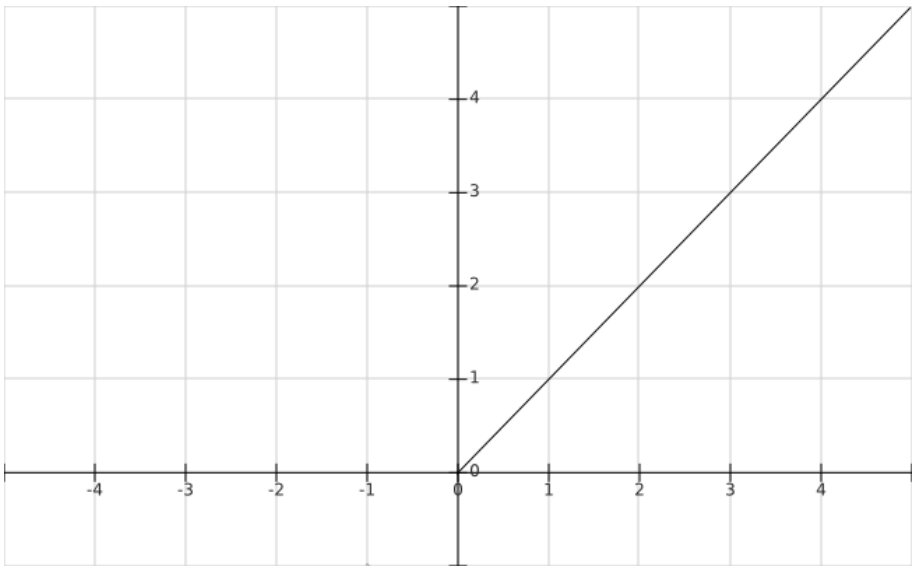

Figure 4.2: The ReLu activation function.

4.1.2 Feed Forward Networks

One of the type of models developed and tested in the thesis is a feed forward neural network model. It is the most simple version of a neural network and in such a model the units are fully connected to all units in the following layer. The units are however not connected to each other within the layer. The unit data is propagated forward to the next layer, and feedback does not exist. (Schmidt et al., 1992) These are also called dense layers as the connections are dense (each unit is connected to the all units in the next layer). The first layer which receives the feature data is called the input layer and the last layer is the output layer, and the layers inbetween are called hidden layers (as shown in figure 4.3).

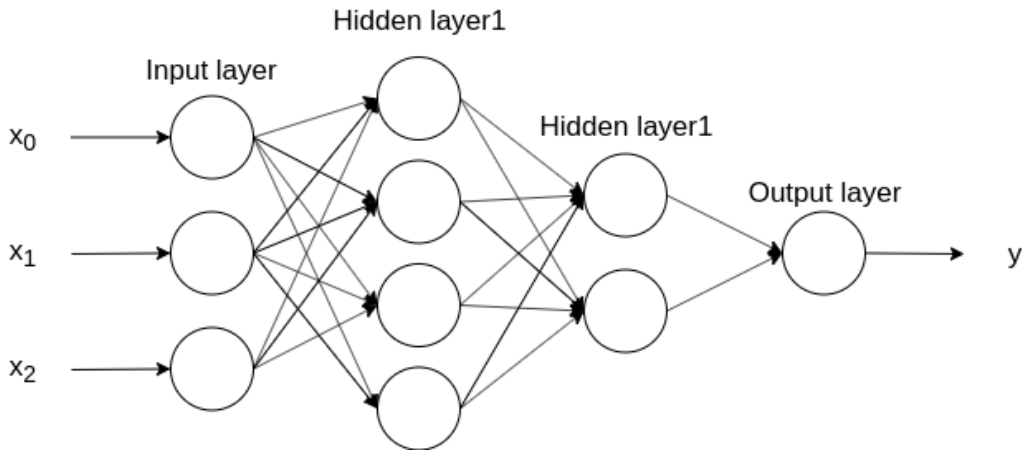

Figure 4.3: A feed forward network with two hidden layers.

4.1.3 Long Short-Term Memory

Another type of model that has been used in the thesis is the long short-term memory (LSTM) is a neural network layer that can take sequences of data as input and has a way to remember what it has seen before and use that for deciding what to do with new data, see equation 3.3 to 3.7. The input that enters the architecture can be kept within the cell or be sent out or forgotten through the forget gate. So it can read and write data to its cell like in a memory chip of a computer (Graves, A. and Schmidhuber, J., 2019).

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \tag{4.3}$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \tag{4.4}$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \tag{4.5}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ (W_c x_t + U_c h_{t-1} + b_c) \tag{4.6}$$

$$h_t = o_t \circ \sigma_h(c_t) \tag{4.7}$$

Where σ_g is a sigmoid, x_t is the input tensor, i_t is the input gate, f_t the forget gate, o_t the output gate, c_t the cell state and h_t the output tensor. Furthermore, W and U are weight matrices and b is the bias. A schema of the LSTM cell is shown in figure 3.3 where the green parts are the respective gates and blue arae gatings.

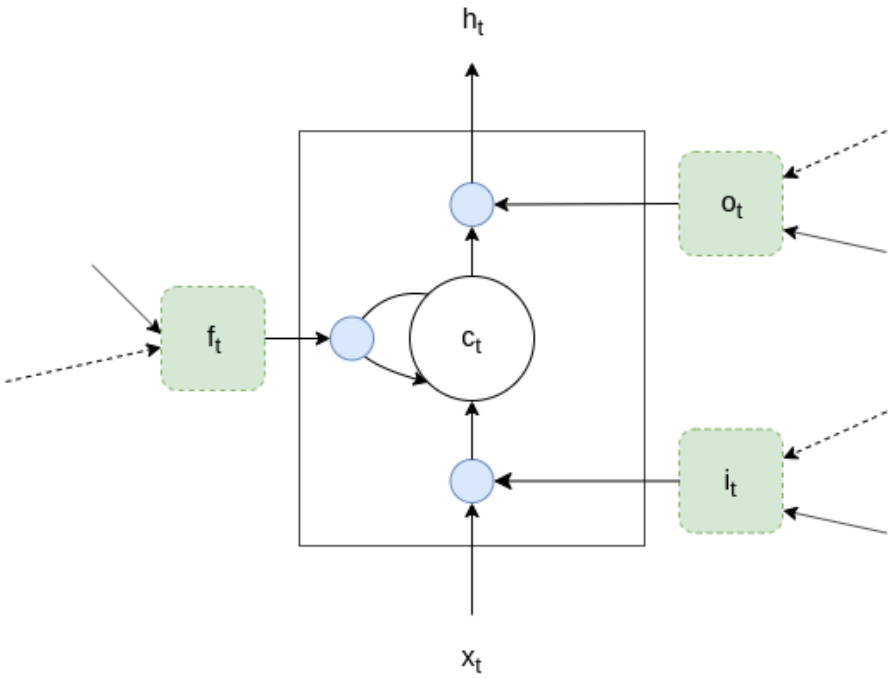

Figure 4.4: The LSTM cell.

(Gers et al., 1999)

4.1.4 Bidirectional LSTM

The LSTM layer can be bidirectional, and is then named a bidirectional long short-term memory (BiLSTM) and has been found to perform better than LSTM networks. Therefor this model has been included in the project and compared to the two other types of neural network models explained above. The concept behind a BiLSTM is to stack two LSTMs on top of each other, where one processes data from start to end and the other from end to start. And the two LSTM stacked on each other therefor form a BiLSTM layer which can process data through time in both directions. The input is connected to both LSTM units, which in turn are connected to the output. The BiLSTM knows what was before and after each point in a sequence and is thus able to predict based on context (Graves, A. and Schmidhuber, J., 2019). An illustration of the network is shown in figure 3.2.

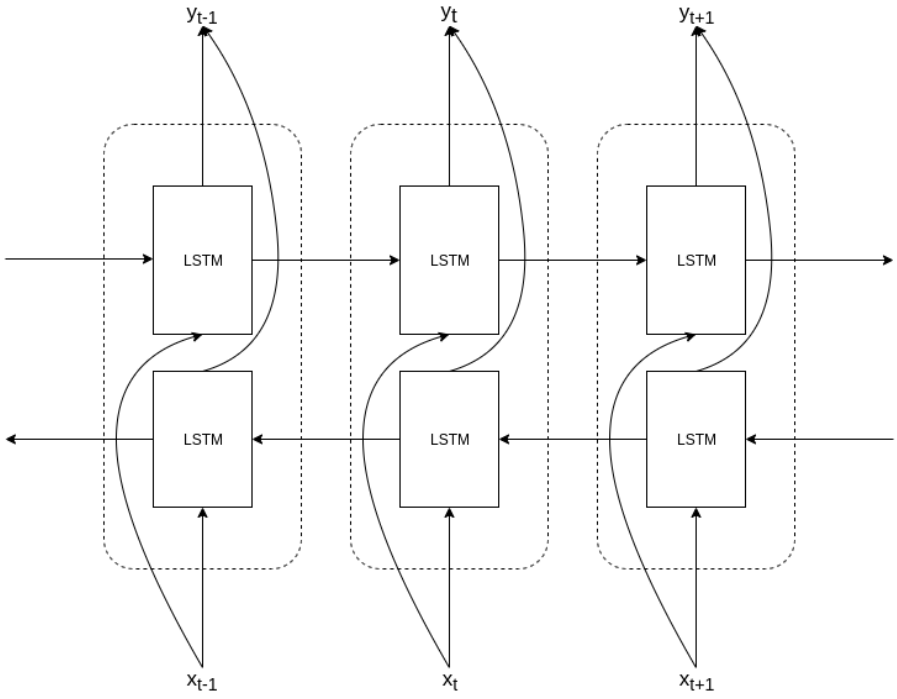

Figure 4.5: The BiLSTM with forward and backward layers depicted.

(Schuster and Paliwal, 1997)

4.2 Linear Regression

To be able to study the weights of the features, I trained a linear regression model. This is since this model gives weights in a format where I directly could draw conclusions around what features that the model prioritized.

Linear regression is statistical model that can be used for problems that involve predicting values. It is performed by fitting a line to data points so that the line explains the data as good as possible, which then can be used to predict new values based on new data. The line can be modelled with

$$y = \beta_0 + \beta_1 x_1 + ... + \beta_p x_n + \epsilon_i \tag{4.8}$$

Where y is the dependent variable, β_i influences the variable x_i. For a linear regression model, the goal is to minimize the sum of squared errors,

$$\sum_{i=1}^{n} [(y_i - f(x_i))]^2 \tag{4.9}$$

where y_i is the prediction, and $f(x_i)$ is the data point. In simple linear regression only one independent variable x_1 is included, unlike equation 3.8 where multiple variables are included, the line is easy to illustrate. Figure 3.5 shows a regression line fitted to data points.

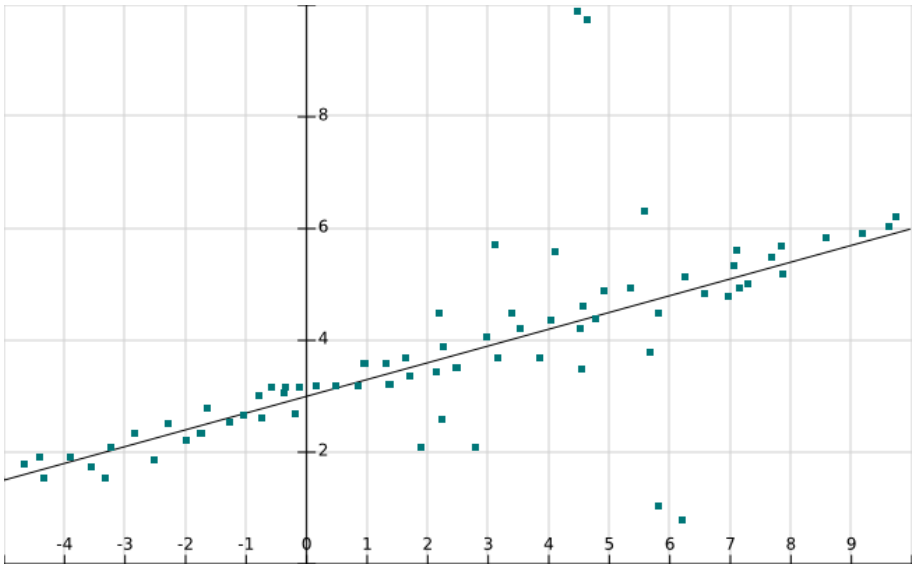

Figure 4.6: Linear regression example

Chapter 5
Method

The main goal from the beginning of the project has been to make some kind of general prototype as soon as possible and iterate. So during the first week a simple *Scikit-learn* model was created where all the steps from training to evaluation were included. However, the data sent into the model was very primitive and the occupancy values were simply random integers representing the occupancy count, as the ground truth data had not yet been collected.

5.1 Tools and Technologies

5.1.1 Keras

Keras is a a deep learning library where neural network models can be programmed and does not require the user to go into much detail. It is a simple API for creating and prototyping neural network models and I used the library for this purpose during the project.

5.1.2 Scikit-learn

For preprocessing, there are several tools in the python library Scikit-learn. I used its functionality for shuffling and splitting data in the project, as I wanted the data to be in random order and split the data in a train and test set.

5.1.3 Imbalanced-learn

During the data collection most rooms were empty or had few people occupying them resulting in skewed ground truth data. The imbalanced-learn library consists of different algorithms to over and under-sample data, and as a result reducing or increasing the count of

certain targets to get a more balanced dataset (Lemaitre, G. and Nogueira, F. and Aridas C. K., 2019).

5.1.4 Pandas

I used the Python library *Pandas* to analyze and sort data. For example, with the describe function I could get a summary of the data such as the most frequent values, amounts of occurrences etc. The `head` and `tail` functions print a number of rows in the beginning respectively end of the dataset, which I used to view actual values in the data. Further, there is functionality to drop specific columns and locate rows. Moreover, since the project includes a lot of code which I had to run to process data, compilation and loading time of the IDE was long. Pandas also includes functionality to save and read data, for example the `toPickle` and `readPickle` methods, which saved me a lot of time by reducing the amount of code that I had to run.

5.1.5 Numpy

Numpy is a Python library for handling arrays and performing common mathematical calculations. The project dealt with multidimensional data in the form of [sample, time step, feature] and the library has therefor been useful to be able to handle and process the data.

5.2 Data Collection

Initially, there existed no ground truth, i.e. the number of people in the rooms. So we collected the ground truth data by walking around the office and manually note the occupancy level.

I collected the data by noting the occupancy of rooms in regular 11 minute intervals and had help from two other workers from the team at Sony. The rooms were of varying size from small rooms with four chairs to conference rooms with 14 chairs. We performed the collection between 10 A.M. to around 16 P.M. during four days. This data was used for developing a working model and later on in the project I collected data during five additional days since more data was needed to improve results. For the case of a room being empty, data could have been taken from nighttime hours where registered sensor activity is nonexistent, but there were basically always some room that was empty and therefor the data collection naturally included empty rooms.

5.3 Preprocessing

To be able to get the data in compatible format, preprocessing was performed. Some of the features were converted into numerical values since they initially were in nonnumerical format.

Since some parts of the preprocessing required generating large variables like pandas dataframes and dictionaries, this kind of large data was pickled (cached) during the first runs to be able to run the code without the large wait times while testing and debugging.

The original data files were first read into Pandas files to be able to get valuable information about the data and also so that the data efficiently could be sorted with respect to specific columns.

After we had collected the ground truth it was transferred to spreadsheet files so that they easily could be read with Pandas. Since the data was collected from specific rooms, dates and times in the office, the training data had to be preprocessed so that the times and rooms of the data collection process were used.

The data consisted of large amount of rooms, but naturally the room data that was used for training was only the ones where the ground truth was collected. Also, the data with timestamps before, after and between the data collection periods were cut off, so that only the needed data was used, which also decreased run time in the IDE as less data was processed.

The data had to be sorted by timestamp so that the data could be inserted in the same order as the ground truth collection. The total data frame was divided into 20 different data frames for each room that the data was collected from to avoid unnecessary data traversal.

Also worth to note is that there were many parts of the ground truth data that was empty and therefor translated into *NaN* in Python. These points were removed since the actual occupancy at these intervals had not been noted.

5.3.1 Feature and Target Mapping

To be able to couple the ground truth data to the feature data, a list was created where each element represents the occupancy level per room for an eleven minute window (EMTW). Since the ground truth target values, i.e. the occupancy count, was collected in eleven minute intervals the training data had to be mapped to these. Most intervals corresponded to large amounts of data points, and to be able to map the training data to the collected targets, two main approaches were tested. The first approach was to take all of the data perand take the sum of the *event count* features and means of the *booking* feature per EMTW and map them to respective target when training. This approach resulted in a two-dimensional input in the form of [sample, features]. The second training approach however generated better results and consisted of having input to the model where each target was mapped to an array with 11 elements where each corresponds to a minute in the EMTW. Each of these array elements were constituted by the feature values for respective minute. So the input was in the form [sample, time step, features].

5.3.2 Empty rooms with detected activity

Sometimes when we observed a room to be empty, the sensor still had detected activity in the room. This complicated the process for the model to learn how to predict a room as empty, so these samples were removed. The reason these samples existed can be due to a misjudgement of the occupancy during the ground truth collection or not noticing that someone walked into the room for a short while.

5.3.3 Feature selection

The data of the sensors contain many features, but there are just a few of them that are relevant to the task of determining room occupancy. The features that initially were used were *timestamp* and *event count* which included both passage and motion sensor event counts. The timestamp feature was the timestamp of an event count and was transformed into an int. However this feature, was later ablated due to not contributing to performance. Only these features were used since the initial goal was to have a low amount of features in the early stage of the project to simplify the process of developing a working pipeline. Later on the amount of features could be increased so that an ablation study of the features could be performed.

Later on the dimensionality adding more features and separating *event count* into *motion event count* and *passage event count* among others, which ultimately improved the models performance.

Ablation study

For the features that were available and relevant, I studied how the features impacted the model's performance by removing one feature at a time and thereafter studying the results. This process is called an ablation study. So the performance with different combinations of features were studied. An ablation study with the total amount of the available features in the raw data was not performed since the majority of features in the sensor data were irrelevant to the task of determining room occupancy. However an ablation study has been performed on the features that were seen as relevant by experimentally removing a feature at a time, training the model on this feature set and then studying the model's performance. The results of the study was that a timestamp, invites in the 11 minute interval that was before each 11 minute interval, and a "not occupied" feature could be removed.

Feature Transformation

To study the weights of each feature in a feature set, linear regression models were trained and their weights were in such a form that it was easy to study how each feature was weighted. When doing this I could see that the *Motion Event Count* and *Passage Event Count* had low weights even though they intuitively are relevant features. This could be that they occur often or that they were too large and had an uneven distribution. This was handled by transforming the features and testing different approaches. The best approach was to take the logarithm

of *Motion Event Count* and the square root of *Passage Event Count*, as shown in 5.1 and 5.2.

$$ecm_a = \log|ecm| \tag{5.1}$$

$$ecp_a = \sqrt{ecp} \tag{5.2}$$

where $\boldsymbol{ecm}$ and $\boldsymbol{ecp}$ is the *Motion Event Count* respectively *Passage Event Count* features.

5.3.4 Outlier detection

I tested removing outliers in the data by using the median absolute deviation (MAD) since it has been shown to be efficient for outlier detection, Leys, C. and Ley, C. and Klein, O and Bernarda P. and Licata, L. (2013) where an outlier is a data point lying far away from the other points. MAD was tested for this purpose to find out if any outliers according to this method was present in the data. It was used on the *Motion Event Count* and *Passage Event Count* features,

$$MAD = M(|x_i - M_j|) \tag{5.3}$$

$$x_i < M + 3MAD \tag{5.4}$$

where $\boldsymbol{M_j}$ is the median of the samples and $\boldsymbol{x_i}$ is the sample value. Feature samples with a value over the maximum value (equation 4.4) is considered an outlier. The approach did however not locate any outliers in the data. And removing the data points that I saw as potential outliers did not give any better results, but if such data points for some reason would be present they are handled to different degree through the feature transformations detailed in equations 5.1 and 5.2.

5.3.5 Feature Imputation

During the project, feature imputation was tested which means that empty or missing data points can be replaced by values so that the dataset is continuous and can be analyzed statistically as a whole. It was performed on missing data and outlier feature values. The methods tried were multivariate, median and mean feature imputation, where multivariate imputation takes other features into account to generate a missing value and the others impute the median respectively the mean of a feature. Results were however not improved with any of the three methods, so feature imputation was ultimately not used.

5.3.6 Over-sampling

Larger meetings were less common in the ground truth data, resulting in less data from these and an imbalanced dataset. The initial distribution of targets after one week of data collection was heavily skewed towards zero but also one and two as shown in figure 4.1. The zero target was under-sampled and the other targets over-sampled with the *Imblanaced-learn* library for the model to be less biased towards a certain occupancy. The algorithms tested for

over-sampling was the "Synthetic Minority Over-sampling Technique" (SMOTE) and Random over-sampling, described below. Oversampling was tried when the data was in the form of [samples, features] but none of the algorithms tested improved results. Moreover, no libraries were found to support three-dimensional data as in the case of an LSTM, which requires an additional time step dimension.

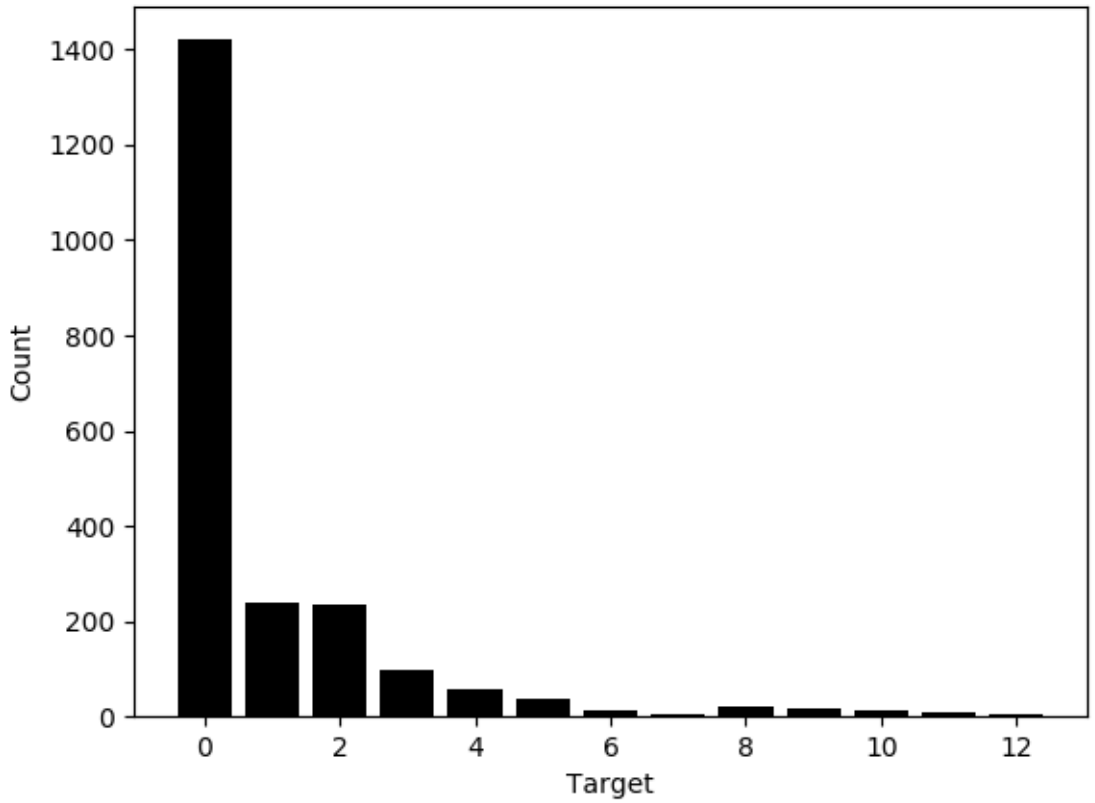

Figure 5.1: Target distribution in ground truth data after initial data collection

SMOTE

Since some target samples were very few, especially those over 12, the SMOTE approach could be used to over-sample them. The SMOTE works by constructing new data points based on data from the nearest neighbors data and creates synthetic samples between the points. N. V. Chawla (2002)

Random over-sampling

Beyond the SMOTE approach, I also tried another technique that could be utilized, namely the random oversampling technique, with an algorithm that over-samples targets based on

random samples taken from the distribution. But this neither gave any improved results.

5.4 Model Development

As mentioned, the work during the project has been carried out iteratively and the goal was to get a working pipeline early on. A simple Scikit-learn model was developed initially and after a neural network model was created. Since the thesis task of predicting the number of people in rooms can be modelled as a regression problem, the loss of the model was set to *mean squared error* (MSE) see 4.2 below where Y_i is the true value and $\hat{Y}$ is the prediction.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y})^2 \tag{5.5}$$

5.5 Evaluation

When the model has performed predictions on the test set, the evaluation is done by comparing the predictions with the observed targets by using different metrics. The main metric for evaluating the model is the mean squared error (MSE), see equation 4.5. The benefit of using MSE as an evaluation metric is that it penalizes large errors. However the MAE has also been used and is explained in 4.4.2 below.

5.5.1 Mean Absolute Error

The mean absolute error (MAE) sums the absolute errors and then takes the mean of all errors,

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}| \tag{5.6}$$

where y_i is the actual value and $\hat{y}$ is the prediction. This metric shows the average absolute error, but can however be misleading if the targets are distributed unevenly or if most targets are low, as this would give a low MAE.

(Cort, J., W. and Matsuura, K., 2019)

5.6 Post-processing

Since their are two models constituting the system, the second model receives data where the first model has predicted that a room is occupied. Therefor, if the second model predicts 0 occupants, i.e. empty room, the prediction is altered to 1 since it probably has made an inaccurate prediction. But since it sees the occupancy as low, it is set to 1.

Chapter 6
Results

The developed system consists of two models. The first one (Model 1) classifies if a room is either empty or occupied. Data of rooms that are predicted as occupied are forwarded to the subsequent regression model (Model 2) where the number of occupants is predicted. To get the best results possible for Model 2, many different architectures were trained and evaluated. The networks that were chosen are respectively feed forward, LSTM and BiLSTM models. Different models and respective results for Model 2 can be seen in table 6.1.

Table 6.1: Results for Model 2 trained when trained for different number of epochs.

Architecture	MSE	MAE	Epochs
Feed forward	2.73	1.11	70
LSTM	2.67	1.05	70
BiLSTM	2.49	1.02	70
Feed forward	2.59	1.08	100
LSTM	2.30	0.95	100
BiLSTM	2.31	0.96	100
Feed forward	2.54	1.05	130
LSTM	2.33	0.94	130
BiLSTM	2.40	0.97	130
Feed forward	2.56	1.07	150
LSTM	2.52	0.99	150
BiLSTM	2.32	0.97	150
Feed forward	2.31	1.03	175
LSTM	2.47	0.96	175
BiLSTM	**2.27**	**0.94**	175
Feed forward	2.49	1.05	200
LSTM	2.63	0.99	200
BiLSTM	2.43	0.95	200

The best performing archiceture was the BiLSTM and the lowest MAE was achieved by training it for 175 epochs. The factor that determined what model to present, is the one with lowest MSE. This is since the MSE is a better metric on how well the model performs as it penalizes large errors. But the best performing naturally also has the lowest MAE.

6.1 Model 1

The initial model is a binary classifier in the form of a feed forward network and predicts if a room is either empty or occupied. It has been trained 5 epochs which was enough to get good results. Its layers are described in the layers section underneath and it is also depicted in Figure 6.1 below.

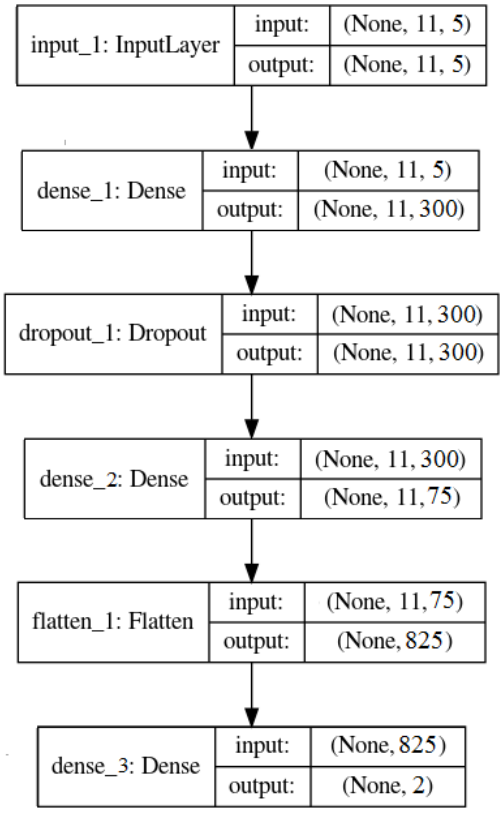

Figure 6.1: The model architecture with its layers.

6.1.1 Layers

Input

This is the input layer where the data enters in the form [sample, time step, feature], after this layer it enters the first Dense layer named Dense1. The number of samples can vary, however time step is 11 and there are five features in total.

Dense1

This layer has 300 neurons and its activation is ReLu.

Dropout1

The subsequent layer after the first dense layer is a dropout layer with a dropout rate of 0.25 to randomly dropout neurons from the prior layer to hinder overfitting.

Dense2

The second dense layer has 75 neurons and like the Dense1 it has a ReLu activation function

Flatten

This layer is used to transform the data into an appropriate dimension that can be handled by the output layer. As mention, the input has three dimensions and this layer converts it into two dimensions with a feature dimensionality of 825.

6.1.2 Output

Two neurons are assigned to the output layer of the model as it is a classification model. The activation function is a sigmoid. The first neuron represents an empty rooms, and the other one represents an occupied rooms.

6.2 Model 2

The model that performed the best was a BiLSTM architecture as depicted in figure 6.2. It has several layers between input and output. There are two BiLSTM layers in the model, where each of them are followed by a dropout layer to prevent overfitting. The last dropout layer is flattened to prepare for the output layer which is a dense layer with one unit. The models output is the predicted occupancy of a room within an 11 minute interval. All of the layers of the model are described below. The model was tested on different numbers of epochs, but the model performed the best at around 100 epochs. Also, learning rates between 0 and 0.000001 were studied to see which could benefit the model the most and the best performing one was concluded to be 0.0005.

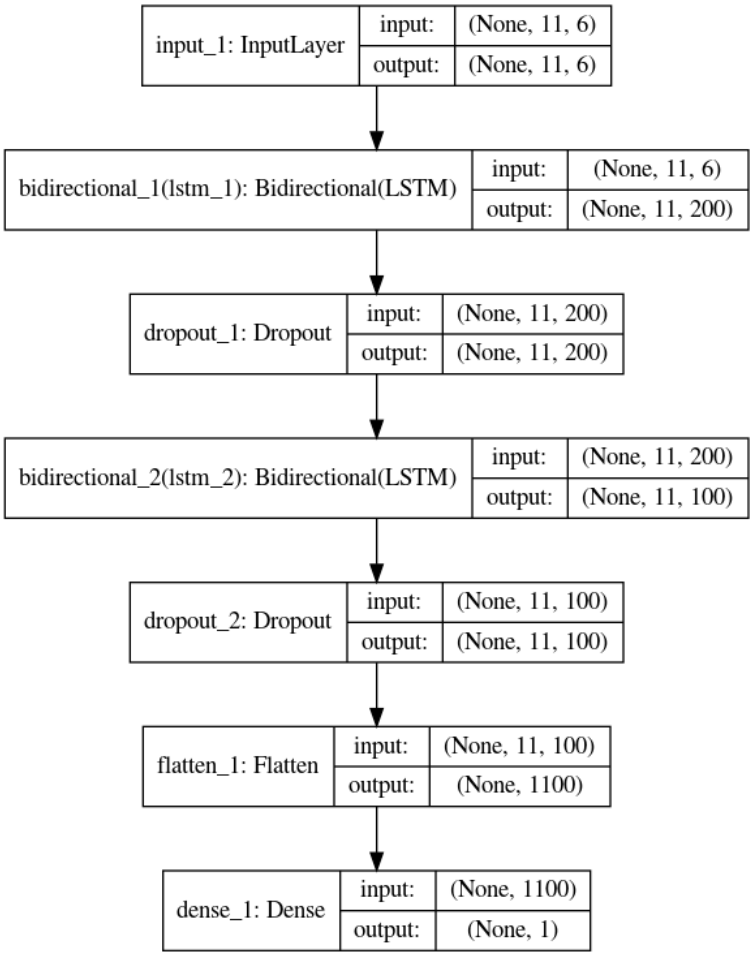

Figure 6.2: The model architecture with its layers.

6.2.1 Layers

Input

The input to the model is a three dimensional array, where the first axis (axis zero) constitutes the samples. A sample corresponds to an 11 minute interval in which the data was noted per room. Axis one contains time steps, each corresponding to one minute of data. The third axis is an array containing the five feature values: *Motion event count*, *Passage event count*, *Room size*, *Number of attendees* and *Binary occupancy status*.

BiLSTM1

The input layer is connected to a Bidirectional LSTM (BiLSTM) layer with 100 units. It has return sequences and activation set to the ReLu function.

Dropout1

The first dropout layer has a rate of 0.25 to reduce overfitting on the training set by randomly dropping out neurons of the BiLSTM1 layer.

BiLSTM2

After the first dropout layer comes another BiLSTM layer which is similar to the first BiLSTM layer except that it only has 25 units.

Dropout1

The second dropout layer after the second BiLSTM has a rate of 0.1, again to reduce the chances of overfitting on the training set.

Flatten

This layer is to flatten the input into the right dimensions for the next layer. The data is in the form [sample, time step, features], but for the next layer (Dense) to handle the data its dimensionality has to be reduced to two dimensions.

Dense

Since determining room occupancy is a regression task, the output layer is a dense layer with one unit and uses the ReLu activation function.

6.3 Evaluation

For evaluating the result, predictions are divided into different target intervals. The mean percentages of predictions with an error between 0 to 1 respectively 0 and 2 individuals have been calculated for targets within the intervals in the test set as shown in table 6.2. The method used to be able to evaluate the results is cross validation.

6.3.1 Cross Validation

To be able to study how the models of the system perform on different data, each model is trained and tested 10 times on separate data. The total data is divided into 10 folds and the models are trained on 9 of the folds where one fold per trained model is put aside for testing. Thereafter the averages of the results are calculated. After cross validating Model 1, the accuracy of predicting an empty room is 100% and when predicting if a room is occupied it is 100% after editing predictions that are unreasonable. This means changing an empty room prediction to occupied if motion or passage event counts exists. The average performance for Model2 is an MSE of 2.27 and MAE of 0.94.

Table 6.2: Model 1: Accuracy of binary occupancy status.

Occupancy	Accuracy
Empty	100%
Occupied	100%

Table 6.3: Model 2: Accuracy within different error margins and number of people.

Amount	0-1 error margin	0-2 error margin	0-3 error margin
1 to 7	85%	94%	98%
8 to 15	49%	66%	74%

Table 6.4: Model2: Accuracy within different error margins and number of people. Smaller intervals than table 5.2.

Amount	0-1 error margin	0-2 error margin	0-3 error margin
1 to 3	93%	97%	98%
4 to 6	66%	90%	98%
7 to 10	49%	68%	78%
11 to 15	49%	64%	76%

Chapter 7
Discussion and Conclusion

7.1 Discussion

In related work, many different approaches have been studied to determine the occupancy state of a room, i.e. if a room is occupied or empty, and occupancy level with machine learning techniques. The work of Hailemariam et al. (2011) studied occupancy state detection with light, CO2 and motion sensors and reached an accuracy of 98% in one minute time windows. This thesis reaches 100% for predicting occupancy state, but a mayor difference is that the time windows are 11 minutes long making it an easier problem to solve as the windows consist of more data.

To my knowledge, the research that shares most similarity with this thesis when considering the task to solve and the type of input data is the study by Raykov et al. (2016) which used PIR sensor data to predict the number of occupants in meeting rooms. The work reached an accuracy of 97% respectively 84% for when the occupancy count is under eight respectively over seven when the error is two or less in a 20 minute window. In my thesis I reach 94% and 66%, however the prediction windows are different as in this thesis it is 11 minutes. Furthermore, the distribution of target values in their data is unknown and a difference in distribution most likely leads to different results. This thesis had a target count that declines as targets get larger, as depicted in figure 5.1, since larger meetings were less usual during the data collection.

A study by Jiang et al. (2012) estimates the occupancy level with only CO2 data and had a result of 94% accuracy for a prediction error less than 4 individuals. The room had 24 cubicles and 11 open seats, so the number of people that could be occupying the room could be larger than the maximum number of people in a room of 15 as in this thesis. CO2 data therefor seem to be a good way to determine the occupancy level in an office environment as the accuracy is high even for relatively large number of possible occupants.

As the number of people in a room increases, use of motion sensors may not be as efficient since they may not be able to capture all movement due to loss of line of sight. This is since

the infrared radiation emitted from one individual may be blocked by another for example if one is standing up, but this could be solved by adding more sensors to cover different angles in the room. In research done by Mohottige and Moors (2018) they predicted the occupancy level in a lecture theatres where the number of occupants ranged from 0 to 250. They solved the problem by using Wi-Fi based data and reached an RMSE of 25.4 and a MAE of 17.5. The results cannot directly be compared to the work of this thesis but in relative terms the MAE is 7% of the maximum occupancy level, while the MAE of this thesis equals 6%, so by this comparison the results are similar. However this kind of comparison may just be an indication as there are many other factors contributing to the results such as the distribution of targets.

In research by Mamidi et al. (2016) a feature set with data from different kinds of sensors including light, sound, motion and CO2 data. They tested different models including SVMs, MLPs and linear regression. Their results indicated that the MLP performed the best and achieved a RMSE of 0.82 where the number of occupants were between 0 and 10. The MSE of this thesis for the best performing model is 2.27 and by taking the quare root it results in an RMSE of 1.51. So in comparison their result is much better. However, there are many aspects that make the results differ. Especially the fact that the study used many different sources of data and an maximum occupancy level of 10 individuals, while this thesis includes room occupancy level of up to 15 individuals and data only derived from motion sensors and bookings where the latter not always is in line with the actual occupancy level.

It can be interesting to name the study by Benezeth et al. (2011) on people counting with cameras. Even though it is a different type of project as it includes image processing, the aim of people counting is still the same. In the study, they counted the exact number of people with 93% accuracy in an office environment, while this thesis is at that level of accuracy when occupants range from 1 to 7, but with an error margin of 2 individuals. This infers that a camera setup is superior to the approach of this thesis and they show that cameras can be an effective tool for people counting. However, a camera solution introduces possible problems to consider such as increased need of computational power, but also cost and intrusiveness.

The PIR sensor has the benefit of being relatively cheap, however it is hard to achieve perfectly accurate results, even with booking data which I had access to. The booking data was however not always in line with the actual occupancy and seemed to be more of an indicator to some extent of occupancy level for the model. The BiLSTM model was the one that gave the best results, as compared to the model with LSTM respectively the feed forward dense model. However the LSTM performed better than the dense model. This can be since the input to the model is in the form [sample, time step, features] and LSTM has memory to be able to handle the sequences of features in a time window of time steps, unlike the feed forward dense model which does not have recurrent connections or memory similar to the LSTM layer's units. The bidirectional model in addition to the LSTM model can process the time steps both directions. This architecture has been found in other applications to perform better than unidirectional LSTMs. The architecture may however not be perfect since in other applications when using LSTM models such as in language technology, specifically named entity recognition, each time step is predicted and its weights inputed into other LSTM units to give it data about previous time steps and base the prediction partially on this. A different type of input to the model than the one I present could therefor be tested in future work. Such an approach could be to let each time step corresponds to a certain occupancy count to be predicted. If meetings last on average 45 minutes, 4 time steps corresponding to

4 time windows of 11 minutes could be tested as input to a model.

The evaluation metric results for MAE is slightly larger compared to the relatively similar study by Raykov et al. (2016), however it is hard to compare this metric with other studies as it includes the prediction of empty rooms in its results thus including zero values, which in my project were the easiest to predict, and thus could be contributing with low errors and could therefor lower the total mean absolute error. It would however be interesting to compare the results with the study if they would have included the MSE of their results as this metric penalizes predictions with large errors. This may affect the metrics as the predictions could be less varying in magnitude compared to if the dataset was completely balanced with a large share of high occupancy counts. However, when comparing the results in percentage-terms they still have an edge on larger occupancy levels, but as mentioned this thesis lacked these samples. However I still think the results for this is still acceptable for purposes where perfect accuracy is not a make or break such as for allocation of cleaning resources or air-conditioning. For these cases it should not be critical to know if there were 13, 14 or 15 people in the room during a meeting, since knowing that the occupancy level was large should be good enough.

7.2 Conclusion

In this thesis, it has been shown that it is possible to determine the occupancy level of meeting rooms to a certain degree using PIR sensor and booking data features together with the number of chairs in a room. Occupancy level data was collected by manually noting the occupancy of the rooms and training a model on it. The results for mean squared error is 2.27 and for mean absolute error it is 0.94. The model will therefor on average always have some error when predicting and results are therefor not perfect, especially for larger occupancy levels. However, I show that the problem of occupancy can be performed by using a deep neural network approach while many other studies use other machine learning models such as HMMs and linear regression to predict the occupancy level. In conclusion it is hard to say if neural networks or other models perform the best on the problem of determining room occupancy level, unless the same exact dataset is used, however I show that neural networks, and especially BiLSTMs, is a possible option for solving this type of problem in addition to other machine learning models. Neural networks have been shown to perform well in many different applications from image recognition to natural language processing, and for me it was natural to try it in the area of room occupancy. Also, to my knowledge there is only one study looking at the occupancy level in meeting and conference rooms, i.e. the research by Raykov et al. (2016), where other studies look at predicting the number of occupants in places such as a lecture theatre where there can be hundreds of individuals or larger open office spaces. Also many look at binary occupancy status (occupancy detection) based on motion sensors and other data while I predicted the occupancy count in meeting and conference rooms with up to 15 chairs. I hope that this thesis eventually can be helpful to others in the future who aim to solve this problem or other machine learning regression problems.

Bibliography

Benezeth, Y., Laurent, H., Emile, B., and Rosenberger, C. (2011). Towards a sensor for detecting human presence and activity. *Energy and Buildings, Elsevier*, 43:305–314.

Chodon, P., Adhikari, D. M., Nepal, G. C., Biswa, R., Gyeltshen, S., and Chencho (2013). Passive infrared (pir) sensor based security system. *International Journal of Electrical, Electronics & Computer Systems*, 14(2):1–5.

Cort, J., W. and Matsuura, K. (2019). Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *My Journal*, 3.

Gers, F., A., Schmidhuber, J., and Cummins, F. (1999). Learning to forget: Continual prediction with lstm.

Graves, A. and Schmidhuber, J. (2019). Framewise phoneme classification with bidirectional lstm and other neural network architectures. *My Journal*, 3.

Hailemariam, E., Goldstein, R., Attar, R., and Khan, A. (2011). Real-time occupancy detection using decision trees with multiple sensor types.

Jiang, C., Masood, M. K., S. Y. C., and Li, H. (2012). Indoor occupancy estimation from carbon dioxide concentration.

Lemaitre, G. and Nogueira, F. and Aridas C. K. (2019). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 3.

Leys, C. and Ley, C. and Klein, O and Bernarda P. and Licata, L. (2013). Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*, 49.

Mamidi, S., Chang, Y., and Maheswaran, R. (2016). Adaptive learning agents for sustainable building energy management.

Mohottige, I. P. and Moors, T. (2018). Estimating room occupancy in a smart campus using wifi soft sensors.

N. V. Chawla, K. W. Bowyer, L. O. H. W. P. K. (2002). Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 3:321–357.

Ngah, S., A., B. R., Embong, A., and S., R. (2016). Two-steps implementation of sigmoid function for artificial neural network in field programmable gate array.

Nwankpa, E. C., Ijomah, W., Gachagan, A., and Marshall, S. (2016). Activation functions: Comparison of trends in practice and research for deep learning.

Raykov, Y. P., Özer E., and Boukouvalas, A. (2016). Predicting room occupancy with a single passive infrared (pir) sensor through behavior extraction.

Schmidt, W. F., Kraaijveld, M. A., and Duin, R. P. W. (1992). Feed forward neural networks with random weights.

Schuster, M. and Paliwal, K., K. (1997). Bidirectional recurrent neural networks.

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2019-10-31

EXAMENSARBETE Determining Room Occupancy with Machine Learning Techniques
STUDENT Daniel Myhrman
HANDLEDARE Pierre Nugues (LTH), Fredrik Karlsson (Sony)
EXAMINATOR Jörn Janneck (LTH)

Automatisk Uträkning av Antal Mötesdeltagare utan Kamera

POPULÄRVETENSKAPLIG SAMMANFATTNING **Daniel Myhrman**

Mötesrum planeras och byggs i olika storlekar, men det är inte säkert att de används i tillräckligt stor utsträckning eller att de till och med är för små. I detta arbete har jag utvecklat ett system som automatiskt räknar ut antal personer i rum baserat på data från bland annat rörelsesensorer.

För att bestämma antalet personer som befinner sig i ett mötes- eller konferensrum har jag byggt ett system med två modeller. Data från rörelsesensorer (Figure 1) och antalet inbjudna deltagare till möten har matchats med manuellt insamlad data i form av det faktiska antalet personer i rummen. Datan matades in stora mängder i modellen för få den att lära sig att på egen hand veta antalet personer i nya situationer.

Modellerna är utvecklade med hjälp av metoder inom ett område som heter maskininlärning. Mer specifikt har jag använt mig av neurala nätverk som är olika typer av modeller som är inspirerade av hur våra egna hjärnor fungerar med kopplingar mellan nervceller.

Den första modellen i systemet förutspår om rummet var ledigt eller upptaget. Ifall det var upptaget räknar den påföljande modellen ut antalet individer som var i rummet. För att kunna veta hur väl modellen presterar, lades delar av datan åt sidan som testdata och flera modeller har tränats och testats på olika data. Sedan räknade jag ut medelvärdet av resultaten hos de olika modellerna. Systemet klarar av att bestämma antalet personer (från 0 till och med 15) i rummen med lite under en person fel i snitt. Det visar att det går att bestämma antalet personer i mötesrum med hjälp av bland annat rörelsesensorer till en viss utsträckning, men att det inte är helt exakt. Det kan däremot användas för ändåmål där exakta resultat inte är kritiskt, som till att omfördela utrymme i ett kontor eller för att veta var man ska städa.

Figure 1: Rörelsesensor som använts i projektet.