Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

Lena Kråkevik Haraldseid

Hidden Markov model for recognizing home appliances based on energy consumption measurements

Master's thesis in Electronics Systems Design and Innovation
Supervisor: Kimmo Kansanen
June 2023

NTNU
Norwegian University of
Science and Technology

Lena Kråkevik Haraldseid

Hidden Markov model for recognizing home appliances based on energy consumption measurements

Master's thesis in Electronics Systems Design and Innovation
Supervisor: Kimmo Kansanen
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

NTNU
Norwegian University of
Science and Technology

Abstract

This paper uses energy consumption measurements to make a system that can use energy measurements to predict if an appliance has been on or off during a given period for the purpose of building an energy saving system. A system like this will help the user's awareness and learn energy consumption habits for the purpose of energy saving. This will help a user to manage their consumption at home, as well as giving environmental benefits. With the knowledge of the consumption pattern a system like this might be able to automatically give saving tips and periodical reports.

However, it needs a lot of time and measurements before it will be able to work as needed. The overall goal is to use energy measurements from single appliances to learn the consumption pattern and usage for each appliance. There will be trained a Hidden Markov model for each appliance, based on individual measurements. After training, these models will be used on the total consumption data for the home to classify weather the appliance has been on or off during a period.

One of the biggest challenges are to get enough data to train the individual models, and then use this data to train the models for the total consumption data. To address this, the task is parted up in different parts with smaller goals to give a proof of concept. The total consumption data for a day includes a lot of different data, and it it hard to identify individual appliances. Future research should spend time training the model to fit the total consumption measurements, and look into making a good model for it. When you are able to recognize different appliances in the total consumption measurements, the system will be able to automatically give reports and tips on energy saving.

Sammendrag

Denne oppgaven bruker strømforbruksmålinger til å lage et system som kan bruke strømforbruk til å forutsi om et apparat har vært av eller på i løpet av en gitt periode for å lage et strømsparesystem. Et system som dette vil hjelpe brukeren med bevissthet og lære om strømforbruket i hjemmet for bruk for strømsparing. Dette vil hjelpe en bruker å administrere eget strømforbuk i tilleg til å gi miljøfordeler. Ved å vite forbruksmønsteret kan et system som dette automatisk gi strømsparetips og periodebaserte rapporter.

For å lage et system som dette trenger man mye tid og målinger før det vil fungere som ønsket. Det overordnete målet er å bruke strømmålinger fra enkelte apparater for å lære forbruksmønsteret og forbruket til hver apparat. Deretter trenes en Hidden Markov modell for hvert apparat, basert på de individuelle målingene. Etter trening, vil disse modellene bli brukt med total strømforbruk for hjemmet til å forutsi om apparatene har vært på eller av i løpet av en tidsperiode.

En av de største utfordringene er å få nok data til å trene de individuelle modellene, og deretter bruke denne dataen til å trene modellene for de totale forbruksdataene. For å løse dette problemet er oppgaven delt opp i flere mindre delmål for å gi et bevis på konseptet. Den totale forbruksdataen for en dag inkluderer mye ulik data og det er vanskelig å identifisere ulike apparater. Videre forskning burde bruke tid på å trene modellen til å kunne brukes på de totale strømforbruksmålingene, og jobbe med å lage en god modell for det. Når man klarer å kjenne igjen ulike apparater fra de totale målingene, kan systemet automatisk gi rapporter og tips til strømsparing.

Preface

This thesis concludes my master's degree in Electronic Systems Design and Innovation at the Norwegian University of Science and Technology during the spring of 2023. This paper presents the work on the master thesis, including a description of the system, solving of the system and then discussions. This thesis builds up on a similar project thesis from the fall of 2022, where parts of the measurement part of this paper were built.

I started this thesis with some experience with the system for measurements, cloud solution experience through part time job and close to no experience with Hidden Markov models. The challenges for the system were mostly to set it all up and automate processes to make sure the system worked as it should by itself. The Hidden Markov model also gave some challenges due to the fact that I had a lot of measuring data and little knowledge of the limitations and tweaking that had to be done to the model to get decent results.

I would like to thank my supervisor Prof. Kimmo Kansanen for always being available for help and support during the thesis with weekly meetings as well as being available at slack. He has helped me to think in new ways, provided feedback and tips as well as finding methods to solve my problem. If he hasn't had the answer he has always send me in the right direction to get help. I would also like to thank my mentor Markus Tacker from Nordic Semiconductor for great inspiration for the entire thesis and also with the work with the cloud solution in this paper. In the end I also would like to thank my fellow student Niels Bakmann for great help with the development of the Hidden Markov Model used in this thesis. He has always been eager to help with coding, debugging and discussions around how the model works.

Table of Contents

List of Figures

List of Tables

List of acronymes

AWS Amazon Web Services

API Application Programming Interface

HEMS Home Energy Management Systems

JSON JavaScript Object Notation

STFT Short Time Fourier Transform

1 Introduction

1.1 Motivation

With increasing energy prices, mortgage and groceries we need to find places to save money where we can. As for mortgage and groceries, they are essential, and will be something we always need and always will pay for. Energy consumption might be one of the things that can make the biggest changes. You will always need energy, but by using it in a smart way you can decrease the bills. It is important to be aware of your own consumption, to make sure you don't use more than needed. The energy price varies a lot during a day, and it is possible to save a lot of money if you pay attention to it. By checking the prices you can optimize the energy consumption and use energy at the ideal time of the day. Energy companies, like Tibber, will give you some statistics in their user app; but you will not get an overview over what kind of appliances that uses the energy - you get the total consumption and some big consumption categories like heating, behavior and 'always on'.

By using less energy, the energy companies can produce less energy. This will decrease the CO2 emissions and will save resources like oil and gas. In Norway we are a part of the European energy market, meaning that we share the energy. This means that we share the energy, and we don't have control over the production of the energy. This way, the energy consumption in Norway is a part of the power consumption in all Europe; including the fossil, renewable and nuclear power. This means that it is as important to decrease the energy consumption in Norway as well as in other European countries regardless of the direct emission effects here in Norway. [2]

One of the most efficient ways of decreasing the energy usage is by turning down the heating. Only one degree can have a big effect on energy saving, and it might be a smart move to have different temperature zones in different rooms based on the usage of that room. It is also possible to install sealing strips to prevent heat leakage around windows and doors. Another smart way is to use less hot water, meaning shorter showers and paying attention to how you use the hot water. Try to use less when you can. It is also more efficient to use the washing machine for clothes, in comparison to washing by hand when looking at the hot water usage.

Most people know the things mentioned above, but find it overwhelming to actually make a difference in their own consumption. They need to look for information and partially guess what the biggest consumers are and try to decrease the usage. By the time the bill comes they will not know exactly what worked or not, and the motivation might disappear. By making a system that includes all of this information, and learns the consumption patterns of the user on an appliance basis can give the user tips and feedback more often and be a motivational factor in energy saving.

1.2 Problem Description

This paper will take you through the process of making an energy saving system to help motivate users to decrease the energy consumption. In the first part of the paper we will address how to make a measuring system for a home, where you can measure different appliances. To store these measurements the paper will look into cloud storage, as that is a good solution for projects like these. For the system to be able to learn the habits and consumption patterns from the user, we need a part where the system will learn from the data it gets. To prevent the user from spending a lot of money on plugs for the measurement system the paper aims to find a solution where the system will learn patterns from one appliance at a time and after a certain period you can move on to a new appliance.

The overall plan for the paper is described in Figure 1.

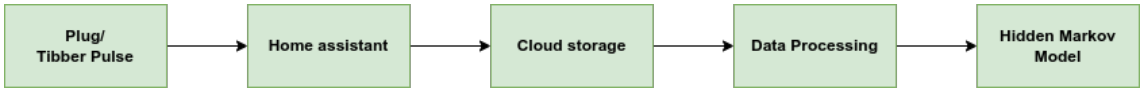


Figure 1: Entire system

To get the measurement the paper will try using energy plugs, which measures the consumption for one appliance and Tibber Pulse. The energy plugs are ideal for measuring one appliance at a time, and Tibber Pulse will be used for measuring the total consumption at every given time. The measurements will be send to Home Assistant which has an API for extracting the data. When the data is in Home Assistant it will be extracted and put into a database in the cloud. The data will then be extracted for data processing and a Hidden Markov Model will be used for learning the patterns of the appliances.

1.3 Goals

The goals for this thesis is listed below:

- Make an automated system for getting the Tibber Pulse measurement data, and putting it in a database.
- Automate the process of getting plug measurements into a database.
- Process both Tibber pulse data and plug data to be synchronized for comparison and the Hidden Markov Model.
- Implement and test how the Hidden Markov Model works for recognizing when and if an appliance has been on during a period.

1.4 Related work

Many researchers are trying to figure out how to decrease the energy consumption, by building smart home systems for saving money and the environment. One common thing is that a lot of these solutions are quite expensive, since you often have to install a lot of things in the home to make it work automatically.

The paper 'Impacts of home energy management systems on electricity consumption' [3] looks at the impact of HEMS systems at 10 Finnish households. Consumption data were gathered in those households before and after installing the HEMS. The households got to decide themselves how they wanted to decrease the energy consumption based on their own values. Some families had babies, meaning they wanted to keep the temperature quite high; while other families were more open to decrease the heat in their home. The results of this project was that in some households the HEMS brought the savings up to 30% for the households that values energy conservation over comfort. During this process the households were interviewed to figure out their values and see how they could decrease the energy consumption, and the installation had a cost of 2000-3500 euros. Even though not everyone decreased the consumption as much as 30%, the outcome of the project was good.

The negative parts about this project will be the price, time gone into interviews and all the installations. If we were able to detect which devices are on at a certain time only based on the

total consumption data we would save a lot of time and money on this. On the other hand, it might be harder to actually control all the appliances without having a system installed that can control everything.

The paper 'A method of appliance detection based on features of power waveform' [4] shows some interesting results. They are able to detect what is on by setting their system in 'learn-mode' where the appliance is put into the electrical outlet for a period while the system learns. The system extracts features during this learning period that are used for comparison later. The system consists of a current detector, microcomputer and transmitter that are equipped in the power outlet. This is somehow similar to my kind of measuring system, but by installing things in the power outlet you are able to get more features for the learning and comparison part. In addition you need someone to come and install this into the electrical plug, while in the system presented in this paper you will be able to just use an electrical plug for getting measurements.

2 Measurement system

2.1 Information sources

2.1.1 Tibber

Tibber is one of the information sources used in this paper. Tibber is a Norwegian power company providing an API for the users. For this paper we are collecting power consumption data from Tibber. The power consumption data comes from Tibber Pulse, which is a product from Tibber. Tibber Pulse gives live power consumption data every two seconds. In addition to the values we are using in this paper, Tibber provides a lot more information and measurements like, current and voltage for different phases, cost data for hour and month, estimated consumption, max hourly power etc. This makes it easy to integrate more information into our system if needed at any time.

2.1.2 Z-Wave plugs

Z-Wave plugs are used for measuring data from specific appliances. They give power consumption data from a day, but with no fixed times. When the appliance uses a lot of power, we get a lot of measurements. When the appliance is off or use less power, we get fewer measurements. These plugs are to be used in the electrical outlet, and they automatically sends their data through Z-Wave to a Raspberry PI that is set up with a program called Home Assistant. The two types of plugs used for this paper is as Fibaro Wall Plug and a Popp outdoor plug.

The Fibaro Wall Plug is a smart plug that can measure energy consumption up to 2500 W. Because of this limit, this plug is more ideal to use on small appliances and not the biggest consumers like a washing machine. It can be turned on and off externally via the Home Assistant App. This plug will show the consumption through a color on the actual plug as well as sending the measurements to Home Assistant.[5] The measurements the plug sends is given in Watt and kWh. If there is no changes in the power, the plug will send energy reports at least every hour. But, if the load changes by 80% you will get an immediate report.[6] The report will be sent up to 5 times per 30 seconds when the power changes by 15%, with a reporting threshold of 0.1kWh.[6] This plug has a circuit breaker threshold at 3000W.

The Popp outdoor plug has a max load at 3500 W, and this is the one used for the washing machine. It can also be used outside, meaning it is a quite good multipurpose plug. It communicates through Z-Wave like the other plug, and measures W, kWh and Ampere. For this paper we will focus on the Watt measurements. The reporting threshold is 50W on this one, and when disabled it will report every 10 minutes.[7] This plug has a circuit breaker threshold at 3600W.

2.2 Technical platforms

2.2.1 Raspberry Pi

A Raspberry Pi is a micro-controller that works like a small computer. The task of the Raspberry Pi for this paper is to continuously run a program called Home Assistant to make sure the plugs can send data to Home Assistant. Since the program needs to run continuously it is easy, efficient and cheap to use a Raspberry Pi compared to a computer.

2.2.2 Git

Git is used for the code in this paper for version control. By using git you are able to keep track of the coding history and all the changes. This makes it easy to go back to working code if you break something, and overall gives you better control over the code base with its changes.

2.2.3 Amazon Timestream

Amazon Timestream is a time series database provided by Amazon. One of the biggest advantages for using Timestream as a database is the scalability. It automatically scales to meet the users need. It is also cost-effective, since you only pay for the storage you use. That makes it perfect for storing data, when you don't know exactly how much space you need or how much you will use the database. It is also easy to use, with both a console you can use in the browser; and it's easy to connect to it and integrate it in your code. If needed you can add scheduling for queries. Timestream also includes time series analytics, which might be useful depending on what you use it for. [8]

2.2.4 Amazon EventBridge

Amazon EventBridge is a serverless service, which uses events to connect application components together. This makes it easier for the user to build scalable event-driven applications. [9] For this paper Amazon EventBridge is used as a daily trigger to fetch the measurements from Home Assistant.

2.2.5 Amazon Lambda

AWS Lambda is a compute service that lets you run code without provisioning or managing servers. [10] The Lambda function in this paper is responsible for fetching the measurements from Home Assistant, and putting the data in the correct place for storing. Like other services you only pay for the compute time of the Lambda function, and the function supports several coding languages. [10]

2.2.6 Amazon S3

Amazon S3 is short for Amazon Simple Storage Service. It is an object storage service that offers industry-leading scalability, data availability, security, and performance. [11] This paper uses Amazon S3 as storage for the entire objects that comes from Home Assistant. This is a lot of data, with some necessary and unnecessary things that might be handy some day. Because of this we want to keep the data, but puts it in another place than Timestream. This is because it is both cheaper and some of the data/measurements are not suited to be placed in Timestream.

2.2.7 Home Assistant

Home Assistant is an home automation application where you can display measurements from Z-Wave plugs and other services like Tibber. The dashboard in Home Assistant gives you all the important information needed from the different measurements you have added, and Home Assistant also provides an API for easy extraction of the data. It is possible to run Home Assistant

on a Raspberry Pi, which makes it easy to run continuously.[12] Home Assistant will automatically detect devices in your home that can be added to the application, making the pairing process easy.[13] It is also possible to automate tasks in Home Assistant.[14] This can be tasks as turning on or off the light at certain times of a day. Home Assistant is also able to detect whether you are home or not, which might be very useful in a home automation system.[15]

2.3 Communication Protocols

2.3.1 Z-Wave

Z-Wave is a wireless communication protocol, primarily used in smart home networks.[16] It allows smart devices to connect to each other, and send information. The communication is two-way with mesh networking and message acknowledgement. A Z-Wave system consists of different Z-Wave devices and one hub. The hub is connected to the internet, the devices are not. Figure 2 shows an example of a Z-Wave system.

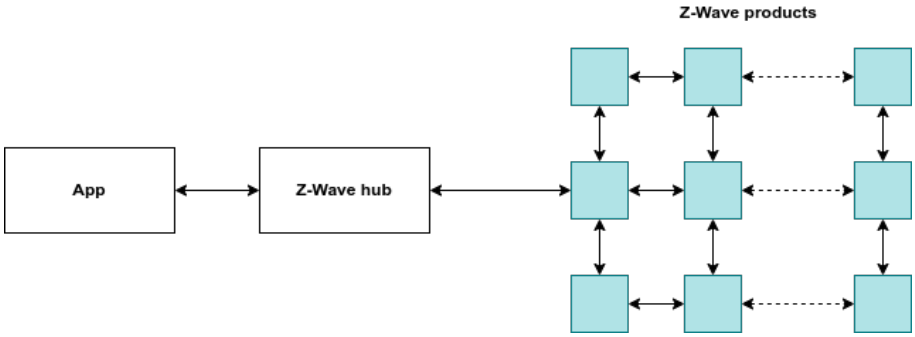


Figure 2: Example Z-Wave system

In the figure we can see the Z-Wave products to the right, which represents Z-Wave plugs that will be used in this paper. This could also be other smart home products. These Z-Wave products can all talk to each other. Normally the product that is closest to the Z-Wave hub will communicate with the hub. This communication is also two-way. By sending the measurements to an App like Home Assistant you have a place to see the measurements and do what you want with them.

2.3.2 HTTPS

HTTPS is used by the lambda function when getting measured data from the API of Home assistant. For sending data between web browsers and web sites, HTTP is the primary protocol.[17] HTTPS is the secure version of HTTP which is encrypted, and stands for Hypertext Transfer Protocol Secure.[17] Encryption is used to ensure security when sending sensitive data. The measurement data from Home Assistant is sensitive in the way that it includes the address and information about the user that should not be publicly accessible.

HTTPS uses the encryption protocol TLS (Transport Layer Security) to encrypt the communication. TLS uses an asymmetric public key infrastructure for encryption, which is a system where you need a private and a public key.[17] The private key is controlled by the website owner, and is kept private. The key is used to decrypt encrypted information from the public key. The

public key is publicly available for people who want to interact with that web site in a secure way. [17]

2.3.3 MQTT

The lambda function that gets the data from Home Assistant and puts it in Amazon Timestream has a rule that triggers the function every day. The rule action writes the data from an MQTT message into the Timestream Table. [18]

MQTT is a messaging protocol/set of rules, used for machine-to-machine communication. [19] It is used a lot with smart sensors, wearables, IoT devices that have to transmit and receive data over a resource-constrained network with limited bandwidth. [19] The MQTT protocol has a lot of benefits, like the fact that it is lightweight and efficient as well as scalable. It is also reliable, secure and well-supported. [19] These are all valuable benefits when building a small or large system depending on MQTT. In systems like home automation systems the security is very important, and MQTT supports encryption of messages and device authentication.

MQTT works in a way where a client can establish a connection with something called an MQTT broker. A client has two options, either subscribe to a specific topic or publish messages to a specific topic. The broker works as a forwarder, and will forward the messages it gets to everyone that has subscribed to the topic. [19]

3 System Description

This paper will go through the process of making an energy saving system, with focus on making a machine learning learning model for detecting which appliances has been on during a period. The aim is to make a system for homes, without making it very expensive. It should also be possible for non engineers to set it up. The system will consist of different parts, where the measurement part is one important part. The measurement system will measure appliances and the total consumption for the household. The ideal system would have to measure the total consumption, and also measure different appliances for a while. The different appliances should be measured enough to train a model, so that the model will be able to detect the appliance in the total consumption data. The appliances can be measured by using a plug, and this plug can be passed around to different appliances after the previous appliance is trained enough.

By using this technique you will not have to buy a lot of plugs to measure the consumption, but you can reuse one throughout the house with the different appliances you want to measure. For this paper we focus on detecting whether one appliance is on, and that will be the washing machine. This is to test whether the method works, and a starting point for the overall system. If we are able to detect whether one appliance has been one, we will be able to detect more appliances. The washing machine is chosen as an example in this paper because it is an appliance with a high consumption, and also an interesting consumption pattern. It is a real appliance that makes sense starting with, and an appliance that is important for a system like this.

The overall system contains of different parts. First, we have the actual measuring of the data, and then we need a system for sending the data to Home Assistant; the program we use for getting the data. Home Assistant has an application were it is possible to see the measurements, and also an API for getting the measurements. Home Assistant only stores the data for 10 days, so we need to find a place for storing the data. Amazon Timestream will be used for this. When the data is put in Amazon Timestream, it is easy to access the data needed and process it.

3.1 Measurement system

The measuring itself will consist of two different measure-devices. One of them are Z-Wave plugs, which are plugs that you can put directly into the electrical outlet to measure specific appliances. The other measure-device will be a Tibber Pulse for measuring the total consumption for the household. The Z-Wave plugs will send the measurements over Z-Wave to a Raspberry Pi running Home Assistant. It is also possible to integrate Tibber Pulse with Home Assistant to get all the measurements the same place. Figure 3 shows how the Z-Wave system is set up for this paper.

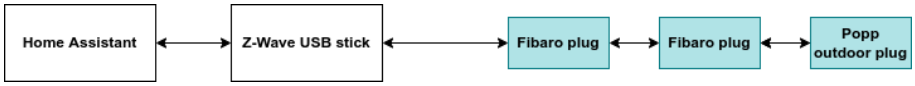


Figure 3: Actual Z-Wave system

The system consists of three different Z-Wave plugs, two fibaro wall plugs and one Popp outdoor plug. They can all talk to each other, and report to the Z-Wave hub which is a Z-Wave USB stick. This stick is connected to a Raspberry Pi running Home Assistant. The measurements from the sticks will show up in Home Assistant.

Initially the Home Assistant program will only work locally within the household. To be able to send the data to cloud storage, the data needs to be accessible from the outside. To fix this it

is possible to use router forwarding to forward the port were Home Assistant is running. In this way it will be possible to reach the data from the cloud, and automate the process of measuring the data and putting it in Amazon Timestream.

By default ipv4 addresses are dynamic and is always changing. [20] This makes port forwarding hard, but it can be solved by using duckDNS. With duckDNS you can pick a hostname of your choice. The hostname will point back to your home connection even if the IP address changes from time to time. [21] DuckDNS sets up the connection and generates secure certificates automatically via LetsEncrypt.

3.2 Data storing

The data will be stored in two different storage spaces on AWS, one for storing all the data from Home Assistant and one for storing the data needed for the HMM. Home Assistant gives a lot of data including weather, GPS locations and different measurements we might not need at the moment. To make sure we don't loose data we might need later we put this data in another storage. It is cheaper this way, and we still have it accessible if needed later. Figure 4 displays what the cloud solution will look like.

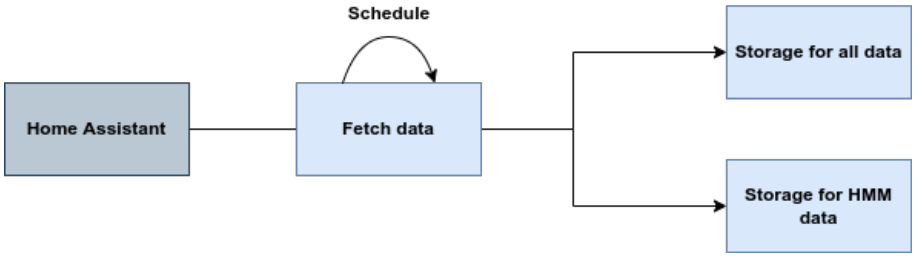


Figure 4: Cloud solution description

The figure shows how we need to fetch the data from Home Assistant at a given schedule, which might be once a day, and after fetching it it should be send to storage.

3.3 Data processing

The data processing step starts with getting the data from Timestream. This can be done by querying the wanted data. In the query you specify what kind of data you want, from which database, and from what time you want the data. For this project we use data for one day at a time. It is then possible to receive the data in a 'time: value' format. This is benefitial for further processing since we need both time and value to process the data.

The data from the Z-Wave plugs and Tibber Pulse comes with different timing. Tibber Pulse comes with measurevalues every 2. second, while the plugs are quite different. When the energy consumption are stable, they report values every 10 minutes, but when the energy consumption is changing, they report much more, even multiple reported values every second. To be able to compare these two data types, or even compare plug data with plug data, we need them to have the same time interval.

By using linear interpolation we can synchronize the data by changing the format to give us data for every 1 minute, every 5 minute or whatever needed. The formula for linear interpolation is given by:

$$\text{Linear interpolation}(y) = y_1 + (x - x_1)\frac{y_2 - y_1}{x_2 - x_1} \tag{1}$$

x_1 and y_1 are the first coordinates, and x_2 and y_2 are the second coordinates. x is where you would like to perform the interpolation and y is the interpolated value. [22]

For us this means that x_1 is the x-axis (time) and y_1 is the y-axis (value), and together they are the first measurement. x_2 and y_2 are another 'time: value' pair. As an example we can look at the following measurements:

```
"1": 2,
"5": 6,
```

If we wanted to know the value at minute 3, x_1 and y_1 would be 1 and 2, while x_2 and y_2 would be 5 and 6. The value y would be the value at minute 3; meaning x is 3. This gives the following equation for minute 3:

$$\text{Linear interpolation}(y) = 2 + (3 - 1)\frac{6 - 2}{5 - 1} = 4 \tag{2}$$

This is how the values will be calculated using linear interpolation. Even though we are missing some of the values in the washing machine measurements we can estimate the value at any given time by using this method. By doing this to both the washing machine data and the Tibber pulse data we can be sure that the data is synchronized and it can be compared.

3.4 Hidden Markov Model

3.4.1 Putting together the model

For the Hidden Markov Model, the first thing needed is the training and test data. The data is collected from Timestream, and then put into separate arrays for training and testing.

Then it's time to get ready for making the model. One can begin by making a matrix describing the model for the washing machine. For this case, we have a Hidden Markov Model consisting of two different models. One model for describing the appliance, and one model for describing everything else. The HMM model then will have two different models, were one is called the 'Good model'. The Good model is the model describing the appliance I want to detect. The other part is the 'Bad model', which is a model describing anything that is not the appliance. The different fields in this matrix represents different states. Figure 5 displays how the transition matrix of the model can look like.

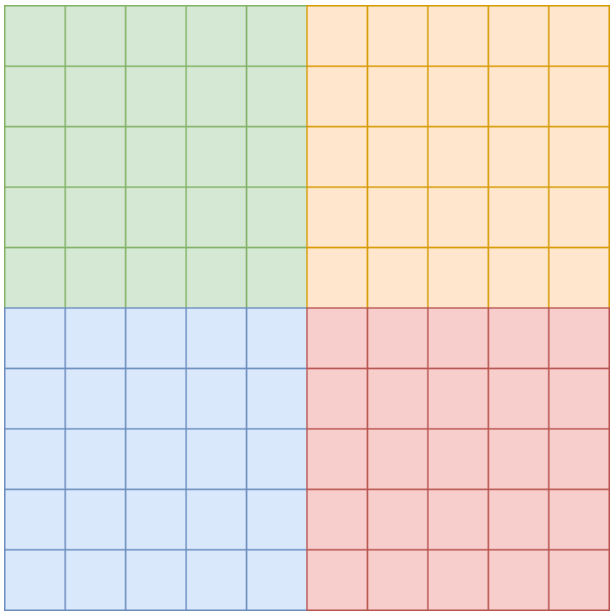


Figure 5: Color coded matrix to understand how it's made

The top left green part is representing the Good model, that should be trained on the washing machine data. The red part is the Bad model and will be trained on the rest of the data, meaning all the data when the washing machine is off. To make it possible to transition between the two models, we have to fill in some fields in the orange and green fields as well. The orange gives the possibility to go from the Good model to the Bad model, and the blue fields gives the possibility to go from the Bad model to the Good model. The green field represents a washing machine cycle, meaning it should go through the different states one by one. To represent this with transition values the green field can't be filled with values. From each field you should be able to go back to yourself or transition to the next state. The Figure 6 describes the possible transitions for the Good model.

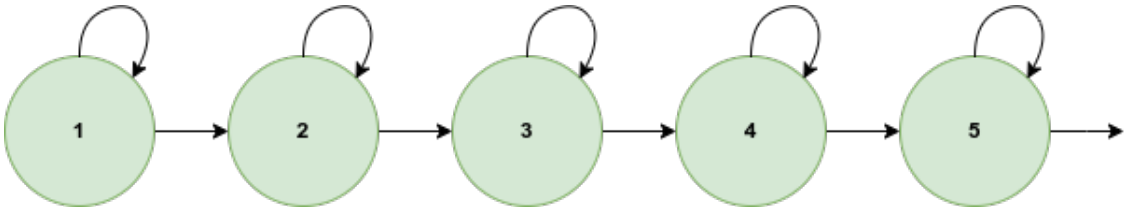


Figure 6: State transition possibilities for the green model.

This figure shows how it is possible to get from the first state and move on to the rest of the states. When you are in one state you can either go back to yourself or continue to the next state. It is not possible to go back to the previous state. This is because when we have a cycle we want to recognize it won't be possible to go back. To get into the first state it is either possible to start in the first state, or go from the bad model, through the blue field and then into state 1. Figure 7 shows the ideal transition matrix for a model like this.

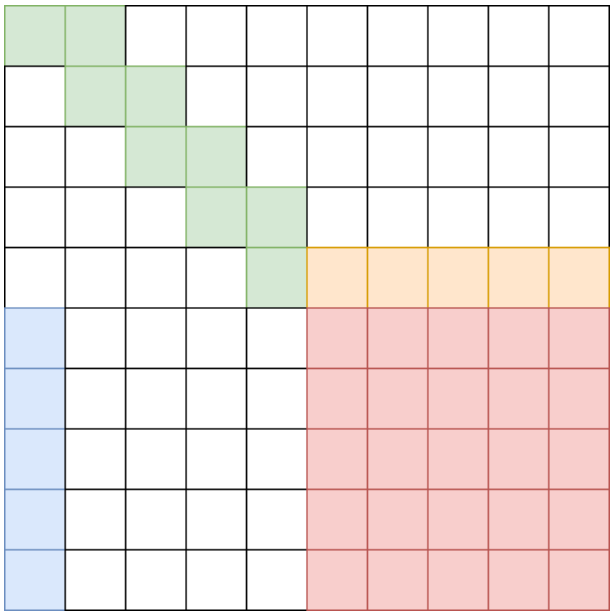


Figure 7: Color coded matrix with correct format

In this matrix there is a lot of white fields. These are the fields that should not be filled with values, meaning that's where it is not possible to go. The green fields are now a diagonal, stating that each field can only move to itself or the next field. When it comes to the bottom right of the green fields, the only option is to go into the orange fields and then go into the red fields which is the bad model. To get from the bad model to the good model, the only option is to go into the blue fields. The blue field is only filled within the first column since you should only be able to enter the Good model in the first state. Also for the orange field you are only able to go out of the Good model after the cycle so this one is only filled at the bottom row.

Both the Good model and the Bad model needs to be trained to work as they should, and they will be trained individually. First, the Good model is trained with washing machine measurement data. The Bad model is then trained with the data when the washing machine is off. After both models are trained, we make a new model were we combine them. That looks like the matrix shown above in Figure 7. The values in the orange and blue fields are set during training and evaluation of the model. The green model are trained, but the transition matrix need to be set to some values based on training to match the diagonal pattern. During training the entire green field is filled with values, that's why it needs fixing afterwards. In addition to setting the transition matrix we also add the values mean and conv from the training to the finished model, to make sure we bring the trained data for the finished model.

3.4.2 Training the model

The model is trained by using an algorithm called Baum-Welch. In reality we don't have to do anything to make this algorithm do it's thing other than initializing the model and writing a line of code

```
model.fit(trainingArray)
```

This line of code will train the model with the training data given. The model should be trained until convergence, so it is important to keep track on the training to make sure it is trained

enough but not too much. This can be done by using the following for-loop.

```
def trainToConvergence(model, trainingArray):
    iterations = 100
    tol = 1e-4
    prev_score = None
    for i in range(iterations):
        model.fit(trainingArray)
        curr_score = model.score(trainingArray)
        if prev_score != None and abs(prev_score-curr_score) < tol:
            break
        prev_score = curr_score
    return conv_arr
```

This function makes sure the model is trained until convergence. We are setting a upper limit for training iterations, and a tolerance for what we see as a point where the point converges. We then check the difference between the score for each training, and if two values are so close that the difference is smaller than the tolerance; then the training has converged.

During the training there is a lot of things happening with the Baum-Welch algorithm. The algorithm is a forward-backward algorithm, and its purpose is to tune the parameters of the HMM.[23] It will tune the transition matrix, emission matrix and initial state distribution. In the initial phase the matrices are either set to known values or set randomly. The algorithm in itself is an expectation-maximization algorithm; where the E step includes forward and backwards formulas that gives us the expected hidden states given the observed data and gives parameter matrices.[23] The Maximization step will update the formulas and tune the parameter matrices with the new observations to find the best fit.[23] The two steps iterates until the parameters have converged.

3.4.3 Testing the model

For testing the model, the measurements not used in training will be used for testing. Then we know that the model has not seen those measurements before, and we can be certain that the results are correct.

We are using the Viterbi algorithm for testing the model. It is an dynamic programming approach, because it uses the previously calculated result in the next calculation. That is a good fit since in HMM any state depends on it's first previous state.[24] The Viterbi algorithm uses the trained parameter matrices to find the choice of states to make the joint probability reach maximum.[24]. It finds the most likely choice based on the test data and the trained model.

During testing I will use daily data from multiple days, both when the appliance has been on and days where the appliance has been off. First I will look at the measured data, and see where I expect the model to go into the good and bad model. The data should go into the good model when the appliance is on, and into the bad model when the appliance is off. From looking at Figure 7, the good model has the states 0-4 and the bad model has the states 5-9. When the washing machine is on we should see states from 0-4, and when the washing machine is off we should see states from 5-9. The measurements from the plugs might have the form shown in Figure 8.

```
"1": 0.0,
"2": 0.0,
"3": 0.0,
"4": 0.0,
"5": 60.0,
"6": 2447.0,
"7": 2298.0,
"8": 122.0,
"9": 77.0,
"10": 222.0,
"11": 164.0,
"12": 355.0,
"13": 80.0,
"14": 0.0,
"15": 0.0
```


Figure 8: Example of values when the washing machine goes on.

This figure shows 15 different measurements of an appliance. The time is shown at the left, and the value at the right. It starts at 0, indicating that the appliance is off, and at measure number 5 we can see that the appliance goes on, and it stays on until measure number 13. At measurement 14 and 15 it is off again. That means that the appliance has been on at 9 measurements, and off at 6 measurements. The expected outcome of a HMM with the measurements in Figure 8 could be the values shown below:

Predicted states: 7 8 8 9 0 1 2 3 3 3 3 4 4 5 6

The important part of the results are to see that the first 4 states is 5,6,7,8 or 9, and that the states when the appliance is on is either 0,1,2,3 or 4. In the states shown above this looks good, and would be rated as a perfect result. To further rate the results we have multiple ways to do this.

The first way of doing it would be to simply give the good and the bad model a percentage score based on how many correct predictions it has made. For the example above both models would get a 100% score. This is calculated by dividing the actually correct predicting divided by all the possible correct predictions. For this example it looks like:

$$\frac{\text{actually correct predictions}}{\text{possible correct predictions}} * 100 = \frac{9}{9} * 100 = 100\% \tag{3}$$

The model were able to predict all 9 of the 9 total predictions, giving 100% score for the model. This is what we aim for in testing the models.

Another way of rating the results is to use a confusion matrix, were we have four different inputs. The different inputs are true positive, false negative, false positive and true negative. Figure 9 shows how the confusion matrix is made.

		Predicted values: Positive Appliance ON	Predicted values: Negative Appliance OFF
Actual values	Positive Appliance ON	True positive	False negative
	Negative Appliance OFF	False positive	True negative

Figure 9: Confusion matrix explanation. [1]

The true positive values is where the actual value and the predicted values are the same, meaning how many values we have predicted in the good model is actually when the appliance is on. The false negative is when the model has predicted that the appliance is off, but it is really on. The false positive is where the model predicts that the appliance is on, but the appliance is actually off. The true negative is where the model predicts that the appliance is off and it is actually off. For the example shown above in Figure 8 and the predicted states:

Predicted states: 7 8 8 9 0 1 2 3 3 3 3 4 4 5 6

we get the following confusion matrix shown in Figure 10.

		Predicted values: Positive Appliance ON	Predicted values: Negative Appliance OFF
Actual values	Positive Appliance ON	9	0
	Negative Appliance OFF	0	6

Figure 10: Confusion matrix example case 1

In this figure we see that out of 9 actual positive values, 9 were predicted positive, and out of 6 actual negative values; 6 were predicted negative. By looking at another case, were we have wrong predictions, we get a different confusion matrix. Figure 11 shows the confusion matrix for the following states:

Predicted States = 3 4 8 9 9 1 2 3 3 3 3 5 5 5 6

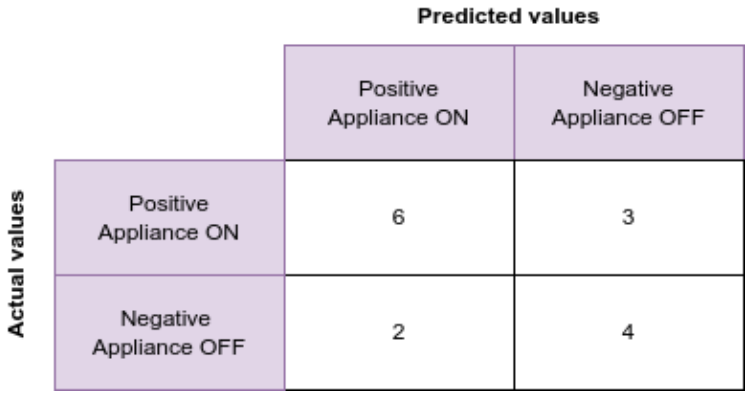


Figure 11: Confusion matrix example case 2

In this figure we can see that out of 9 actual positive values, 6 were predicted positive and 3 were predicted negative. For the 6 actual negative values, 2 were predicted positive and 4 were predicted negative. This gives us 5 wrongly predicted values. By looking at the score in this way, it is possible to distinguish how the two models are doing. It is possible to decide whether it is most important that the good model are doing well or the bad. The choice you have to make is to decide whether the false negative or the false positive is worse. Ideally we want both those to be 0, but in reality that's hard.

For our case, the result we want is to figure out if the appliance has been on during a day. It is not crucial to know exactly how long it has been on, and because of that a false positive might be worse than a false negative. If we get one false negative value during a day, that it not that important. But, if we get a false positive value on a day where the appliance has been off all day it is more crucial. If there is a different amount of positive and negative actual values it might be useful to use percentage when comparing the false positive and false negative values.

4 Problem Solving

4.1 Measurement system

The measuring system consists of many different parts. It all starts with power measuring plugs that sends measuring data via Z-wave to a raspberry Pi running Home Assistant. The data is collected from Home Assistant using a lambda function in AWS and is then sent to Amazon Timestream. By storing the data in Timestream, it is easy to get the data based on different parameters after what you need.

4.1.1 Power measuring plugs

The power measuring plugs are sending measuring data to the Raspberry Pi over Z-Wave. These plugs are used on certain appliances, and can only measure one appliance at a time. Figure 12 shows a days measurements from the washing machine.

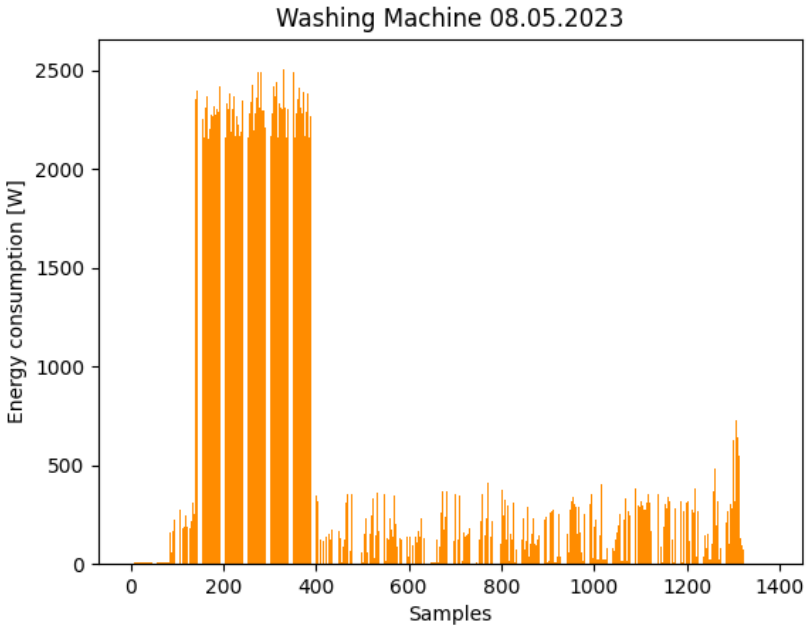


Figure 12: Washing machine measurements from 08.05.2023

This figure shows the energy consumption through a day for the washing machine. The y-axis shows the consumption in Watts and the x-axis shows the number of samples. Figure 13 shows a part of the figure above to show that the measurements stays high during the peak in the plot around approximately sample 150-400.

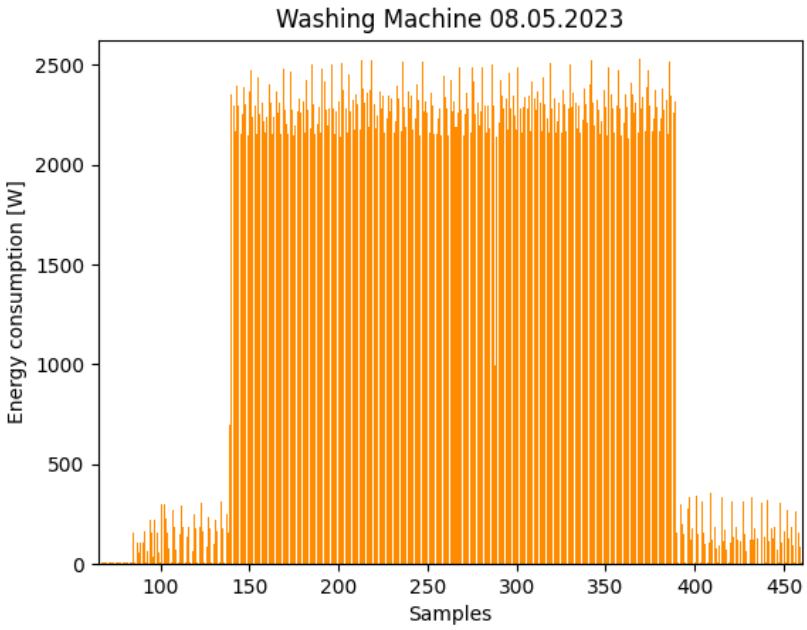


Figure 13: Washing machine measurements from 08.05.2023 zoomed in.

This figure shows the measurements while the washing machine runs when it is on the top to show how the measurements looks like. From Figure 12 it looks like it goes up and down, but this figure shows that it stays high during the entire peak.

Figure 14 shows 4 different washing machine measurements from different days to display what they look like.

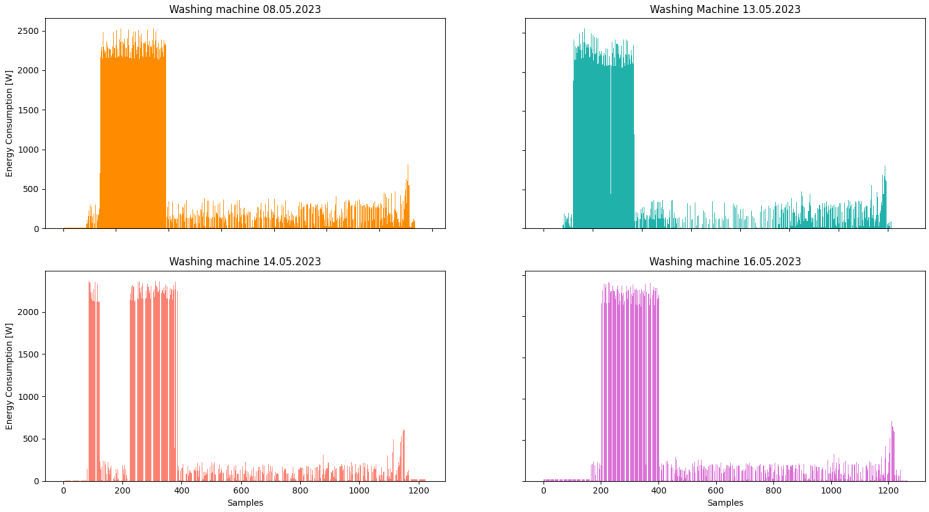


Figure 14: Washing machine measurements from four different days.

We can now see different washing machine measurements from different days. The two sub figures at the top describes a washing machine cycle where the wash is on 40 degrees. For the third figure to the bottom left, the washing machine ran two times that day. One 30 degrees cycle and

one 40 degrees cycle. The bottom right sub figure shows another day with a 40 degrees cycle.

Figure 15 shows the measurements from Tibber pulse from 8.05.2023.

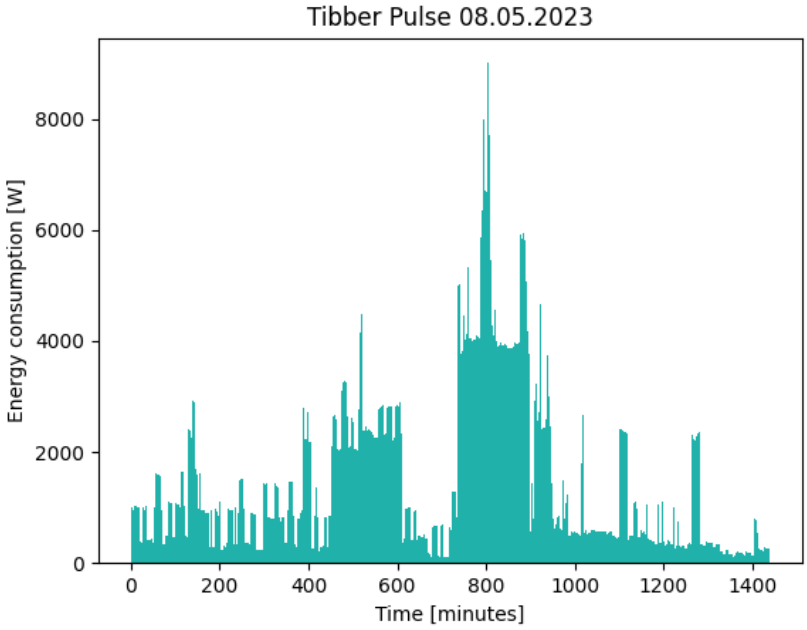


Figure 15: Tibber Pulse measurements from 08.05.2023

This figure shows the Tibber Pulse measurements from one day. The Tibber pulse data is already synchronized since we get regular measurements every 2 seconds. It is possible to see that the measurements shows multiple peaks during a day.

4.1.2 Home Assistant

Home Assistant provides a local web page with an overview over the measured values from the Z-Wave plugs. In addition it is possible to connect Home Assistant to Tibber, meaning that you will have access to all the data from Tibber in Home Assistant. That makes it easy since it's possible to get all the data you need from one API. The downside of using Home Assistant locally, is that you can't reach the data from any other place. In addition it wont be possible to set up scheduled API fetching from the Cloud, because the cloud has no access to the API.

To fix this problem, port forwarding is used. When using port forwarding, the router forwards a given port openly to the internet. It is then possible for outside traffic to get into your network. DuckDNS is used for setting a host name making sure the home connection is constantly reachable. By using this method you are able to access Home Assistant from anywhere, and it is also possible to integrate with cloud solutions for your application.

4.1.3 Amazon Web Services

For storing the data on the cloud, four different services are used from AWS. Eventbrigde is used for triggering the data fetching, Lambda is used for actually fetching the data and Timestream and S3 is used for actually storing the data.

Amazon Eventbridge is in charge of triggering the function that fetches the data. By adding an event schedule, it invokes the lambda function at the given time. This time is set to be every morning.

Amazon Lambda function is a function that is used for fetching the data from the Home Assistant API. It can be triggered to run anytime by triggering it in the Amazon console. Otherwise it runs every time it gets triggered by Amazon Eventbridge, which is every morning. The Lambda function fetches the wanted data, and puts the data in Amazon Timestream and Amazon S3. The code for fetching the data and putting it in Timestream is shown in Appendix A. [25] [26]

The data from Home Assistant contains a lot of measurements, including measured data, Tibber data, GPS locations, weather data and more. For the current task we need the washing machine measurements and the Tibber pulse data. The other kinds of data might be interested later, but they are not needed right away. Tibber Pulse data and washing machine data is put in Timestream, making it easy to extract the data later for processing. The rest of the data is put in Amazon S3. It might be handy to keep this kind of data for later, in case we want to add extra features for our model or in case we would like to train more models. The data is accessible in S3 as well, but it is cheaper storage and it will take a bit more time to get it.

The entire cloud solution is shown in Figure 16.

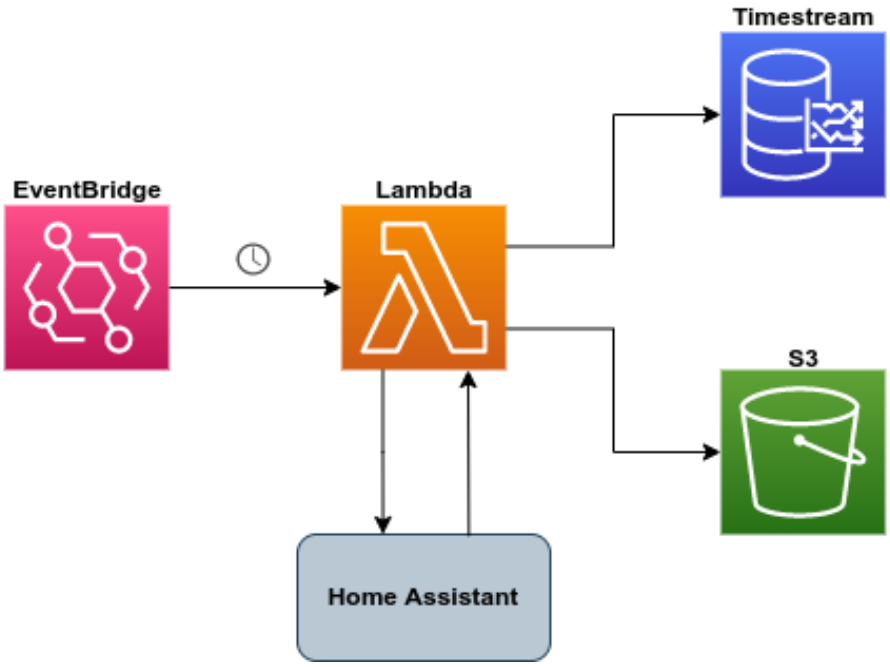


Figure 16: Cloud solution

This figure shows the four AWS services and how the are connected. Eventbridge triggers the Lambda every morning, at 6:00 am, and the lambda then fetches data from Home Assistant and gets the data. The entire object with all the data then goes to storage in S3, while the measurements we are using for the Hidden Markov Model gets send to Timestream.

4.2 Data Processing

4.2.1 Getting data from timestream

The data can be queried from Timestream by using the TimestreamQueryClient, and specifying the data wanted. To get the time and measure values from Tibber pulse, the query would be:

```
SELECT
  time, measure_value::VARCHAR
  FROM "ProjectDB"."projectTable"
  WHERE measure_name = 'sensor.power_hans_osnes_veg_16'
  AND date(time) = '2023-03-18'
```

First you select the values you want, that being time and the measure value, and then the database and table you are getting it from. The measure value is defined by using the name of the measure from Home Assistant. In the end you specify which day you would like to get the data from. This could also be another time period, but for this project we use days. For getting the data from the washing machine, we simply change the measure_name to the name of the washing machine sensor:

```
SELECT
  time, measure_value::VARCHAR
  FROM "ProjectDB"."projectTable"
  WHERE measure_name = 'sensor.popp_smart_outdoor_plug_ip44_rated_electric_consumption_w'
  AND date(time) = '2023-03-18'
```

This can be made info files with data on the following format:

```
{
  "2023-03-18 00:00:00.018000000": 1486,
  "2023-03-18 00:00:01.218000000": 1486,
  "2023-03-18 00:00:02.271000000": 1490,
  "2023-03-18 00:00:04.495000000": 1479,
  ...
  "2023-03-18 23:59:59.642000000": 1776,
}
```

This is an easy format to work with later, and we have a good overview over the data for each day before further data processing. For the washing machine, the measurement data looks like this:

```
{
  "2023-03-18 00:00:00.890000000": 26,
  "2023-03-18 00:03:09.890000000": 26,
  "2023-03-18 00:13:54.474000000": 26,
  "2023-03-18 00:24:39.071000000": 26,
  ...
  "2023-03-18 23:52:15.665000000": 21,
}
```

We can see that the time period of the measurements are different. For the Tibber Pulse measurements we get a lot of measurements every 1-2 seconds, while for the washing machine we get measurements for approximately every 10 minutes.

4.2.2 Linear Interpolation

The measurement data from the Z-Wave plug are quite messy, with different time intervals between measures. You might get multiple measurements for one seconds, and other times there is 10 minutes before you get the next measurement. It is hard to work and process the data when it looks like this, and to make the data easier to work with, linear interpolation is used to get measurements every minute.

The data from Tibber Pulse also has it's problems. It consists of approximately 40 000 measurements every day, with one measurement approximately every second second. By using linear interpolation on this data we can decrease the amount of data, which also makes it easier to process. By having a common time period for measurements it is also possible to compare them, and look at them together.

For the data processing it is first important to make sure the data starts at the same time. To make sure of this we can establish a starting point; which we choose to be 00:00. Some days the washing machine gives out the first measure at 00:10, so this is important to set. For every day we add a starting value at 00:00 for 0W. Then a new dictionary is made to store the values 'time: value', to make it easier to calculate the time. After this I make an array that is called widthArray, which is an array to see how long the different measurements lasts. A for-loop is in charge of checking the duration of each measurements and adds them to widthArray.

The array x_axis_minutes contains the minutes for when the different measurements are measured. It's a cumulative array. When these arrays are made, we use linear interpolation to find the approximate values for every minute by using x_arr and values for interpolation. The values we are using are values from the dictionary newDict, and x_arr contains the values from x_axis_seconds, corrected for width.

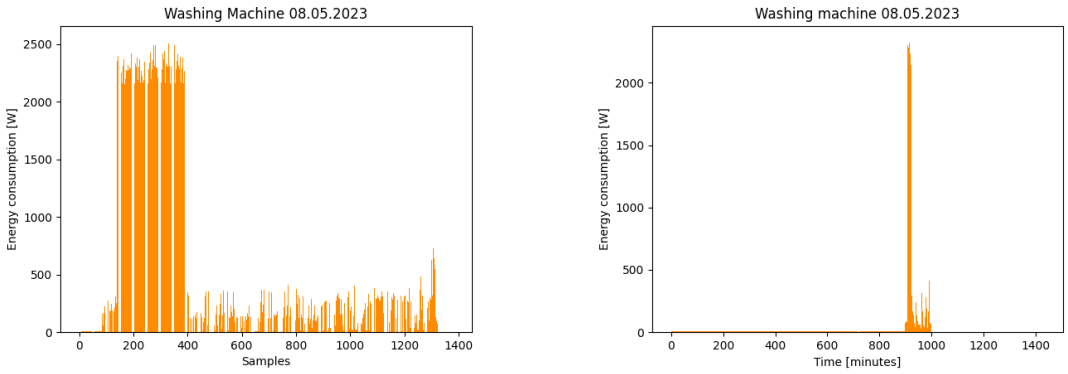
The code is explained below in a form of pseudo code:

```
dictFromTimestream = {.....}
add value in measurements at 00:00: 0W
new dictionary newDict = {} #Dictionary to store data
for values in dictFromTimestream:
    add 'time': value to dictionary #only use hour:minute:second of time
initialize widthArray = []
for key in newDict:
    add duration of each measurements to widthArray #the difference in time
initialize x_axis_minutes = []
for value in newDict:
    add the minutes each measurement starts to x_axis_minutes
x_arr = []
use widthArray to center the measurements to the middle of the timevalue for plotting
put the new centered times in a new array x_arr
LI_Xarr = []
LI_Values = []
Use x_arr and newDict-values in linear interpolation
```

```
interpolate and put in the new arrays with the wanted step value
return LI_Xarr, LI_Values
```

This code is an attempt of explaining how we start with the dictionary from Timestream, extracts the format we want from the long time string, finds how long the measurements lasts, and centers the measurements for correct plotting. We are then left with an array with time in minutes, and the values from the dictionary newDict. These values are used for the linear interpolation together with a step value, deciding how often we want measurements. This can be 1 minute or 10 minutes or anything else dependent on what is needed. The code is also attached in B.

When this processing step is used on both Tibber Pulse data and the Z-Wave data we are able to compare the measurements. Figure 17 shows the washing machine measurement before and after interpolation with 1 min intervals.



(a) Before interpolation

(b) After interpolation

Figure 17: Washing machine before and after linear interpolation

We can see that the washing machine is now placed where it should be during the day, and the length of the run looks better since it is now not lasting the entire day.

Figure 18 shows the Tibber Pulse data together with Z-Wave data 08.05.2023 with an interpolation value at 1.

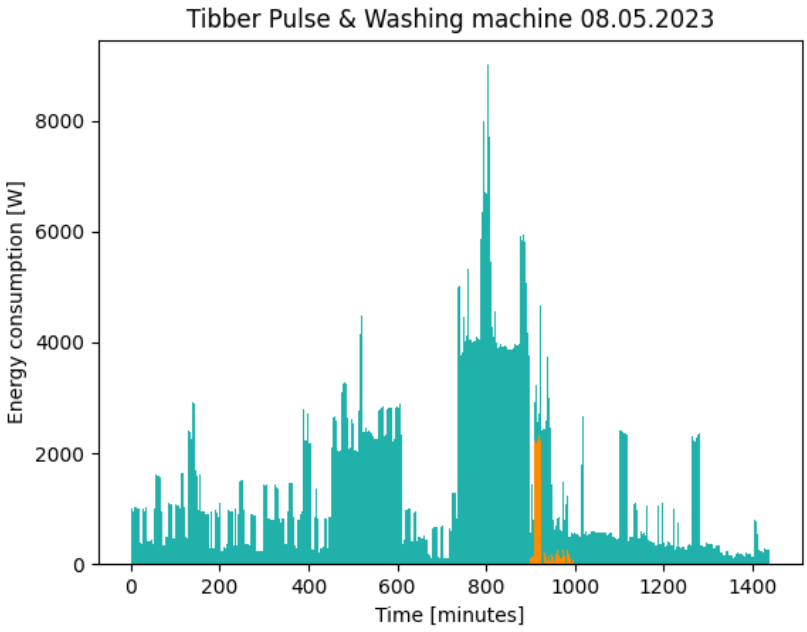


Figure 18: Tibber Pulse and Washing machine - 1 min interpolation

In this figure we can see both the Tibber Pulse measurements and the washing machine measurements together in one plot. This plot has one measurement every 1 minute, and we can see that it is quite detailed and it looks like the washing machine measurements fits quite well where it is.

Figure 19 shows data from the same day, but with an interpolation value of 5.

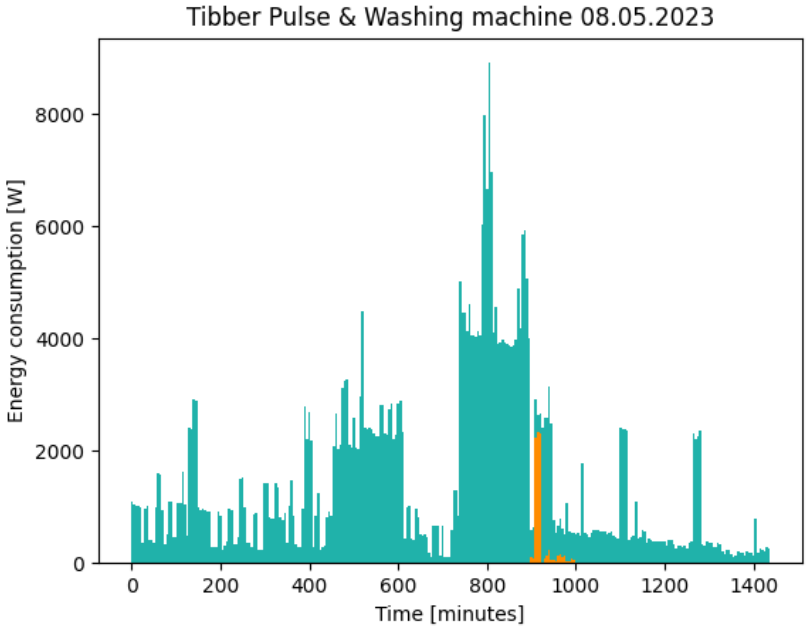


Figure 19: Tibber Pulse and Washing machine - 5 min interpolation

This figure looks quite like the one before, but we can see that it is not that detailed since the measurements are only every 5 minutes. We can see that we missed a couple of the high peaks around the time where the washing machine is on. But still, we can see that the washing machine measurements fits quite well where it is.

Figure 20 shows the measurements if we use 10 minute interpolation value.

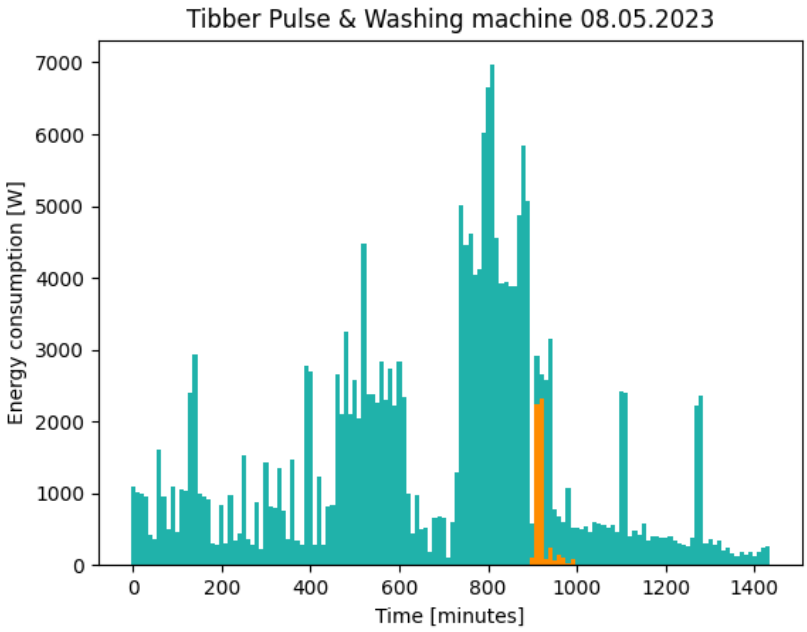


Figure 20: Tibber Pulse and Washing machine - 10 min interpolation

In this figure we have measurements every 10 minutes, and we can see how it is even less detailed but still looks okay in form of placing of the washing machine data.

One can do too much interpolation as well, and Figure 21 shows one example of this.

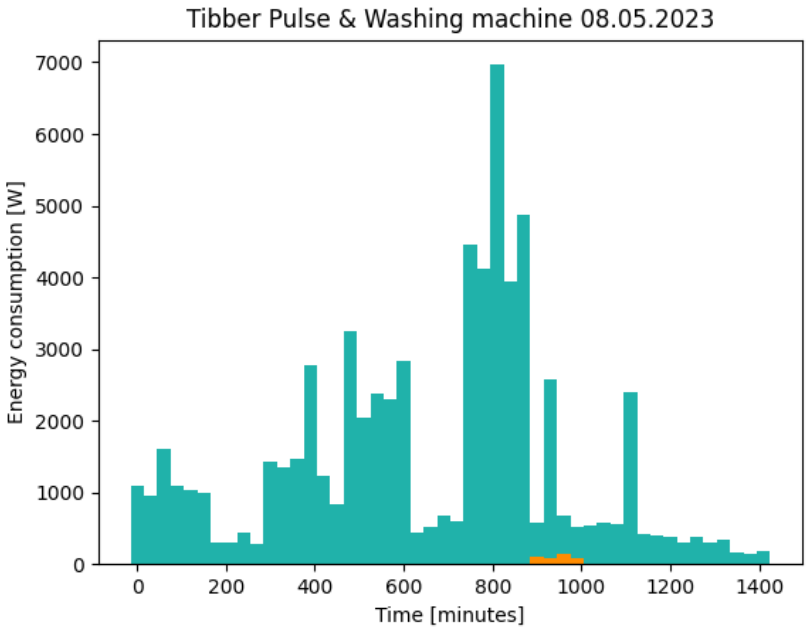
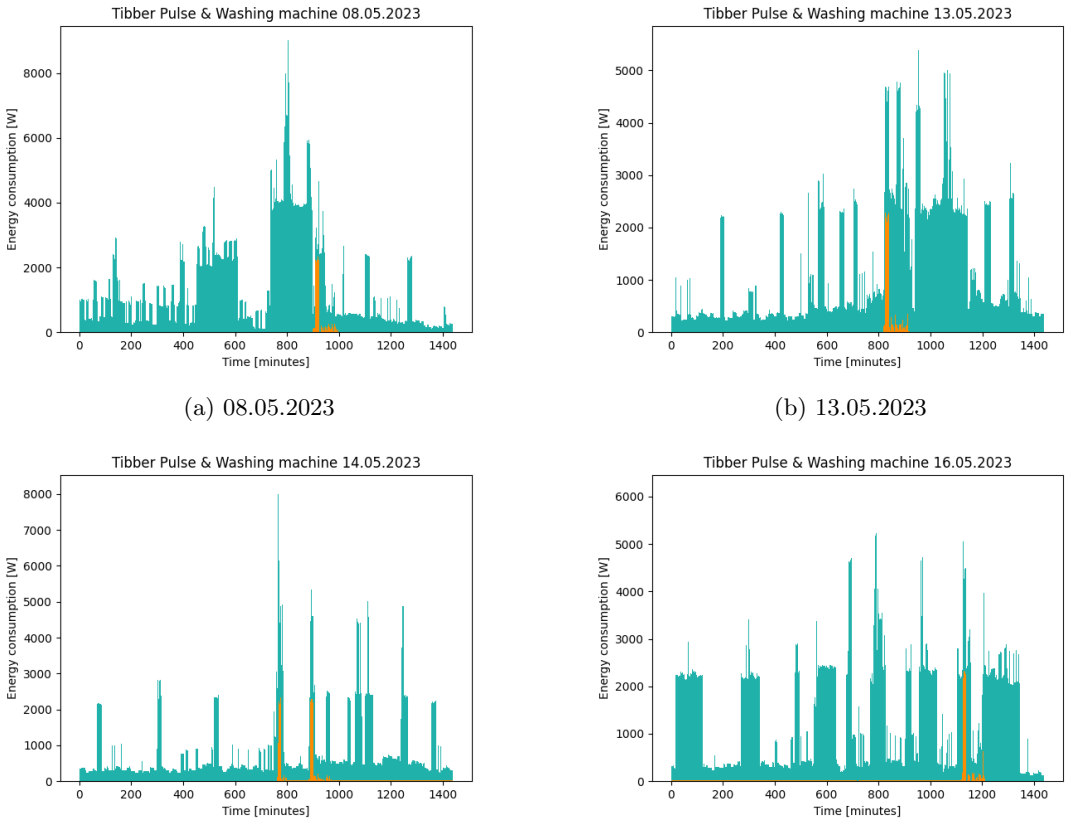


Figure 21: Tibber Pulse and Washing machine - 30 min interpolation

This figure shows measurements for every 30 minutes. In this figure we can see that the washing machine measurements are quite low, and it doesn't have the peak it usually has. This is because the peak of the washing machine is approximately 20 minutes for this day, meaning that during interpolation it might be hard to catch the exact time when the value is on top, and in this case we have probably estimated this value based on lower values before and after the peak; making the peak disappear. This is important to take into consideration when using linear interpolation, at least when you want to keep the characteristics of the measurements.

Figure 22 shows the Tibber Pulse measurements together with the washing machine measurements for four different days.



(a) 08.05.2023

(b) 13.05.2023

(c) 14.05.2023

(d) 16.05.2023

Figure 22: Tibber Pulse and washing machine data for four different days in May.

This figure shows the Tibber Pulse data and washing machine data for four different days in May. These are the same days that were displayed in Figure 14. This figure shows that both the washing machine and Tibber Pulse data is quite different from day to day, but also that the washing machine data looks like is belongs in the place it is put. This means that the linear interpolation works, and puts the measurements in the correct place. This is quite easy to see in the bottom left figure where the washing machine is on at two times during the day. The two peaks fits perfectly in with the two high peaks in the Tibber pulse measurements.

4.2.3 Compare different appliances

Now we are able to put multiple measurements into one plot, to see how different measurements affect the total consumption. Figure 23 shows a day with measurements from Tibber Pulse, washing machine and 2 panel ovens.

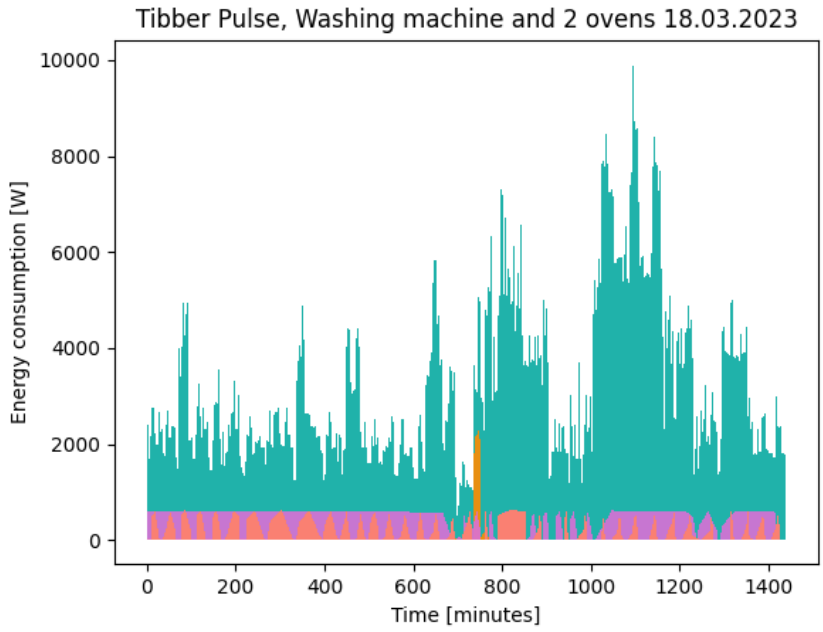
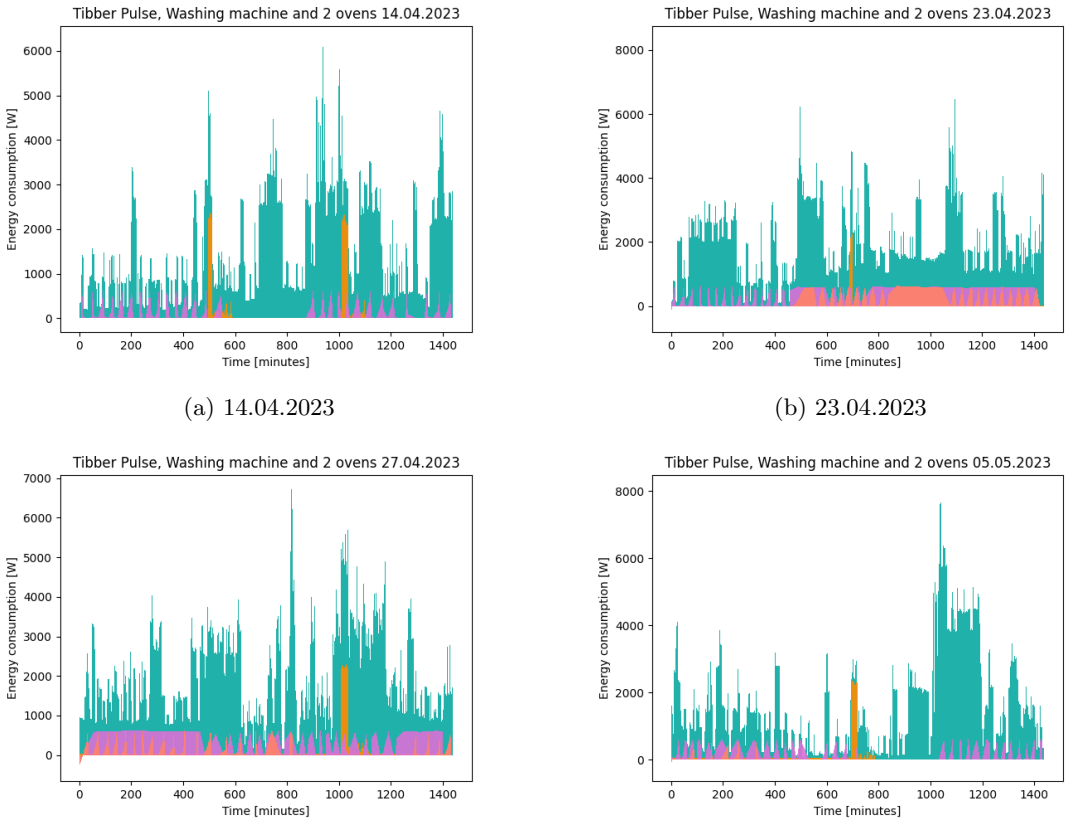


Figure 23: Tibber Pulse, washing machine and 2 ovens.

This figure shows four different measurements during one day. We have the Tibber Pulse in seagreen, the washing machine in orange, and the two ovens in an orchid and salmon color at the bottom. We see that Tibber Pulse is full of different peaks as always, and the washing machine is in one of those peaks. The ovens works a bit different. One of them is on the entire night, then goes a bit on an off during the day, and in the evening it is mostly on the entire evening. The other oven goes up and down regularly. They act different because they are placed in different rooms. The oven that is mostly on all the time is in a room that gets regularly fresh air, but the other one is in a more closed room.

Figure 24 shows 4 other days with the same kind of measurements.



(a) 14.04.2023

(b) 23.04.2023

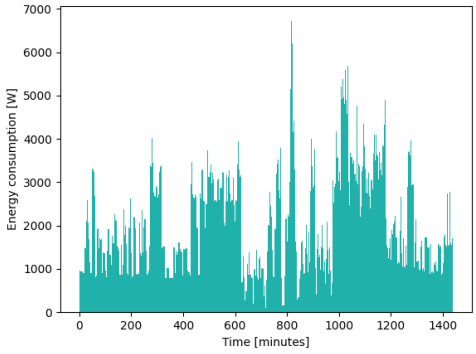
(c) 27.04.2023

(d) 05.05.2023

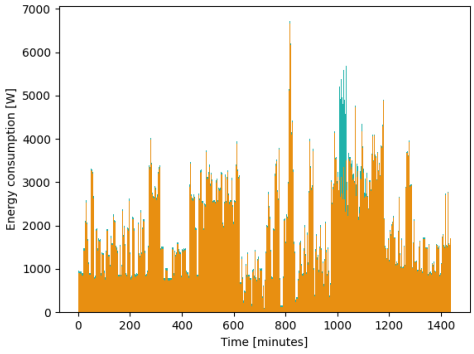
Figure 24: Tibber Pulse, washing machine and oven data for four different days.

This figure shows four different days of measurements from Tibber Pulse, washing machine and two ovens. It can be seen that all of the measurements are quite different all days. The washing machine is the appliance that is most similar for each run, but still it can be quite different between different washes. The ovens are also quite different in how they work. Sometimes they are constantly on, sometimes constantly off and sometimes they go up and down. They are controlled by a temperature, and their job is to make sure the room is in the same temperature at all times. Because of this it makes sense that they are quite different from day to day and minute to minute. A lot of factors can change the temperature in a room, like people being in the room, sun, outdoor temperature and cooking.

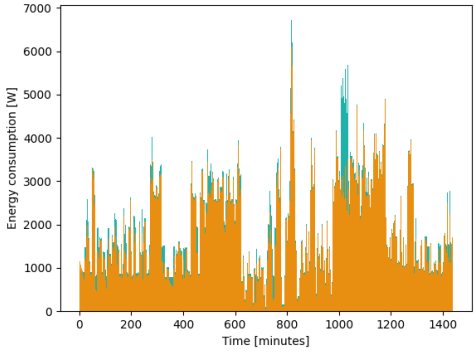
By subtracting one and one appliance it is possible to figure out what the Tibber Pulse measurements are built up from. All appliances in the house should sum up to what Tibber Pulse measures. Figure 25 shows an example of this from 27.04.2023.



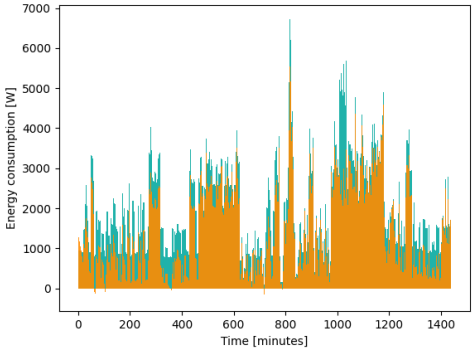
(a) Tibber pulse



(b) Tibber pulse minus washing machine.



(c) Tibber pulse minus washing machine and oven2



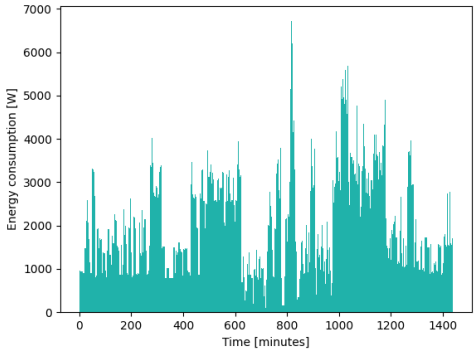
(d) Tibber pulse minus waching machine, oven1 and oven2.

Figure 25: Tibber Pulse and Tibber pulse minus different appliances.

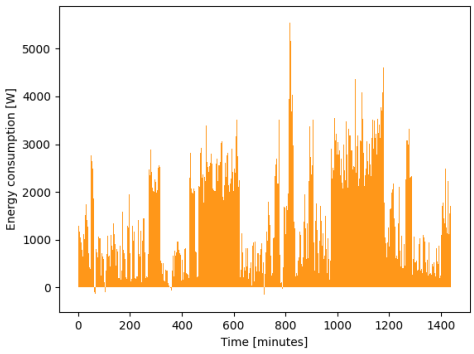
In this figure we can see Tibber Pulse measurements in seagreen, and Tibber Pulse measurements minus different appliances in orange. In the first figure we can see the Tibber Pulse measurements by itself. In the second figure we can see Tibber Pulse measurements in seagreen and Tibber Pulse minus washing machine measurements in orange. We can then see that the peak where the washing machine is running is gone, and during the rest of the day the measurements are a bit lower. This is because the plug for the washing machine usually shows some value, and not 0W when the washing machine is off. For this current day it shows 45W during the day, meaning it will decrease the overall measurement with 45W.

In Figure 25c we can see the Tibber Pulse measurements minus oven2 measurements as well as minus the washing machine measurements. This is the oven you can see goes on and off constantly in Figure 24c. This does not decrease the overall measurements that much, but it is definitely possible to see a change from Figure 25b. In Figure 25d we can see Tibber Pulse measurements and Tibber Pulse measurements minus washing machine, oven1 and oven2. In this figure we can see that we have removed more of the initial measurements, but there is still a lot left that is not identified.

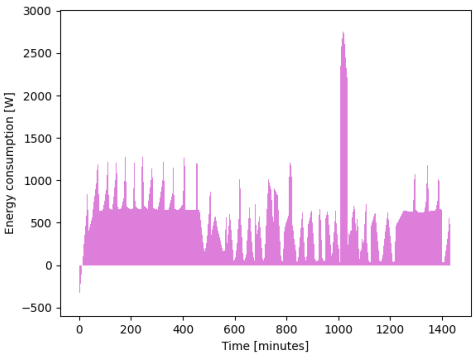
Figure 26 shows the Tibber Pulse measurements from 27.04.2023 and the final measurements when washing machine, oven1 and oven2 is subtracted.



(a) Before interpolation



(b) After interpolation



(c) After interpolation

Figure 26: Washing machine before and after linear interpolation

Figure 26a shows the overall Tibber pulse measurements from 27.04.2023, while Figure 26b shows the Tibber pulse measurements minus the washing machine, oven1 and oven2. Here we can see that the measurements are overall lower and the 'washing machine peak' is gone. Figure 26c shows the washing machine, oven1 and oven2 data summed together. The washing machine can be recognized in the big peak, while the ovens are the small peaks and the other measurements during the day. In the second and third figure it is possible to see that some measurements are negative, something that should be impossible. This could be due to the interpolation, or due to the fact that the plugs measures a value of 45W for the washing machine the whole day even though it is off.

By measuring even more appliances and doing the same, we might get further on breaking down the total consumption. Even though this process with the washing machine and ovens didn't give us that much, it is interesting to see how much these appliances affects the total power consumption. By looking at the power consumption for the panel ovens we can see how much power is used on heating and predict how much we will save in the summer when they are off.

4.2.4 Short Time Fourier Transform

In an attempt of figuring out what kind of data to train the Hidden Markov model with, the Short Time Fourier transform were tested for some different measurements. The hope was to figure out if any of the appliances stands out and if we might be able to use them for prediction. The results are shown in Figure 27, with Tibber Pulse.

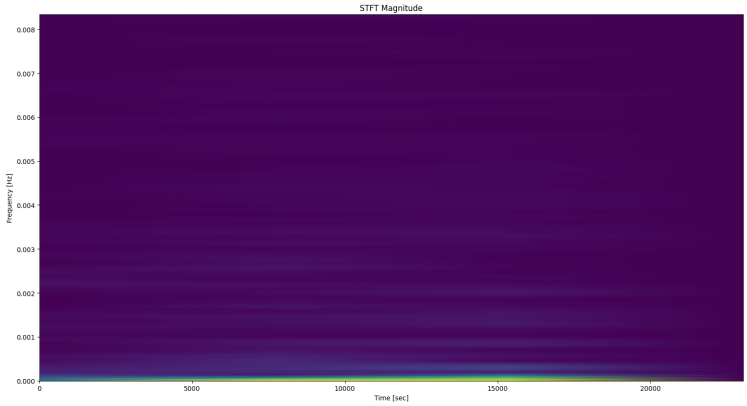


Figure 27: STFT of Tibber Pulse

In the Tibber Pulse STFT we are not able to see that much noticeable. That makes sense since there is a lot of things happening, and not that much reoccurring things. Figure 28 shows the STFT of the washing machine.

Figure 28: STFT of washing machine

In the Washing machine STFT we can see that we get a higher value at a certain place in the measurements. This means that it might be possible to recognize the washing machine using STFT. Figure 29 shows the STFT for oven 1.

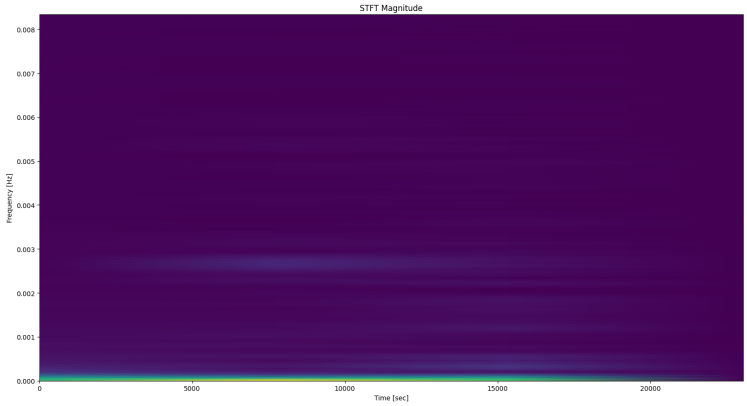


Figure 29: STFT of panel oven 1

In this figure we can see some lines, indicating that we might be able to recognize the oven by using STFT. Figure 30 shows the STFT of oven 2.

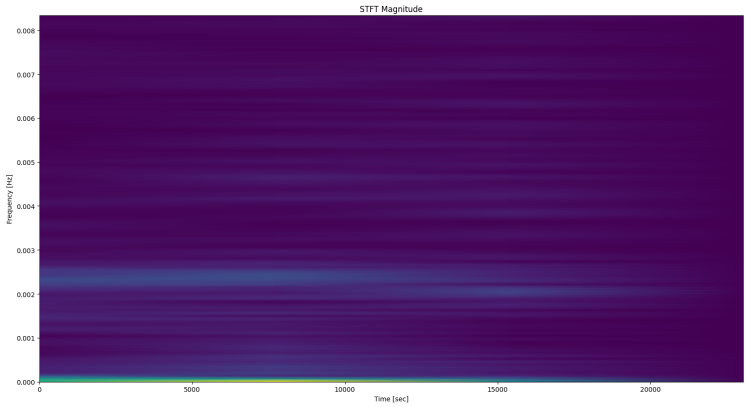


Figure 30: STFT of panel oven 2

This figure shows multiple lines at certain frequencies. This is the 'best' result from the STFT, meaning this would most likely be the easiest appliance to recognize by using STFT.

4.3 Hidden Markov Model

The data previously collected from Timestream and then processed as described in the chapters above, will be used for training and testing the Hidden Markov Model. We start by making the good model, which will describe the appliance you want to detect, in my case the washing machine. The model is made using a Gaussian Hidden Markov Model, and during initialization we choose how many components we want, covariance type and how many iterations we want for the training. These are all thing you have to consider and test to make sure you get the best result. The code for the HMM can be found in Appendix C. [27]

The Good Model is trained with a lot of training data consisting of values from the washing machine when it's on. After training the model you get a transition matrix, mean values and

covariance values. For a model like this, we want the transitions to be going forward or back to itself; not backwards and not jumping around. Because of this we need to restrict the transition matrix to be on the form showed in Figure 7. To get the perfect transition matrix one needs to try different values, and it might be smart to look at the trained matrix and use some parts of it. The most important is to make sure those white fields stay 0, with no possibility of entering it. Figure 31 shows an example of a trained transition matrix for the Good model.

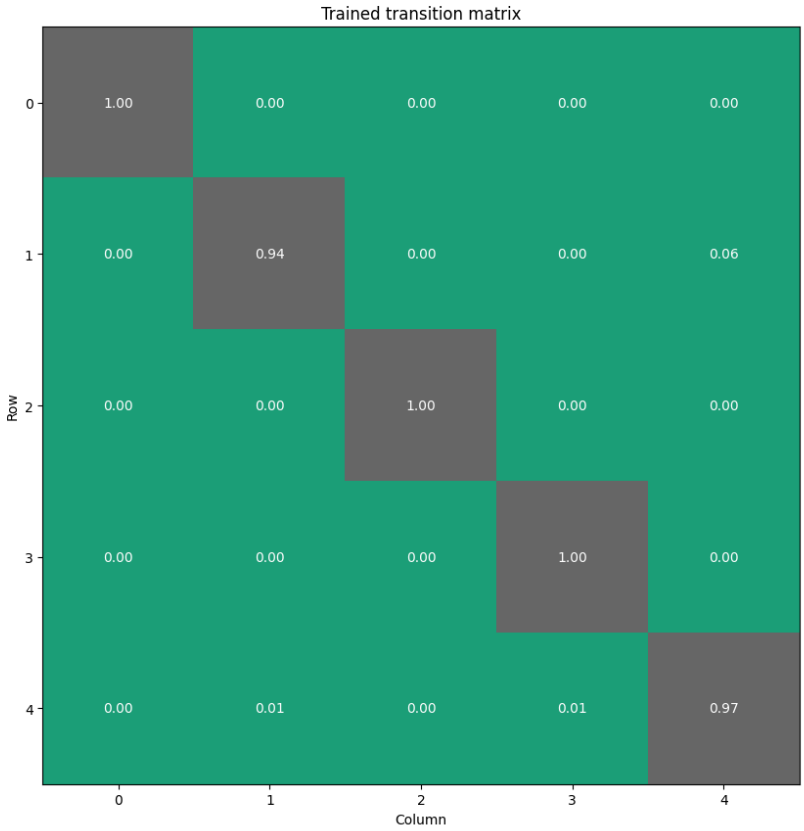


Figure 31: Trained transition matrix for the Good Model

This is an example of a trained transition matrix for the Good model, with washing machine training data. We can spot some problems right away. In the first state, on the top left, the model will be stuck in that state. This is because the transition possibility is 100% for staying in that position. We need to change this so that it can go to the next state. This is the same for other transition probabilities as well.

It is now time to train the Bad model, with all the data when the washing machine is off. This gives us a new transition matrix, mean values and covariance values. For a model like this, we don't care where the transitions go, and we won't do anything to change the transition matrix. We leave it as it is. The job of the bad model is to recognize everything that is not the washing machine. Figure 32 shows an example of a trained transition matrix for the Bad model.

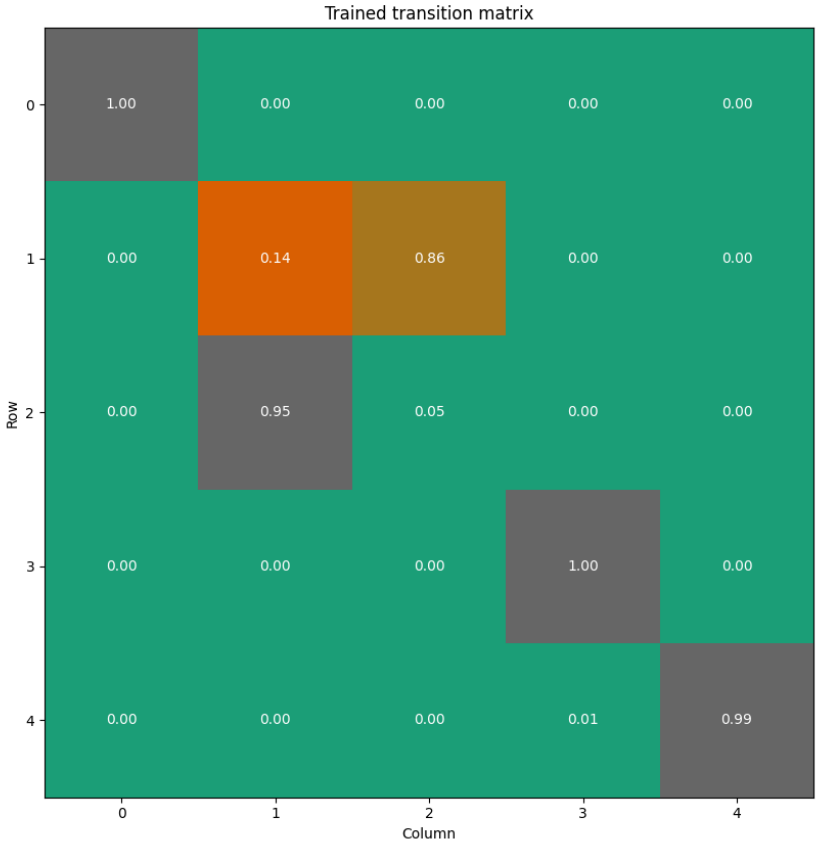


Figure 32: Trained transition matrix for the Bad Model

In this model it can also be seen that some possibilities are 1 (100%), which is not good. We would like to see a model where the possibilities are spread over the model giving possibility to go from and to every state.

The next step is to put these models together to one big model. Then it's needed to add the transition probabilities between the different models as well, to make sure it can transition between the good model and the bad model. Figure 33 shows an example of the entire transition matrix.

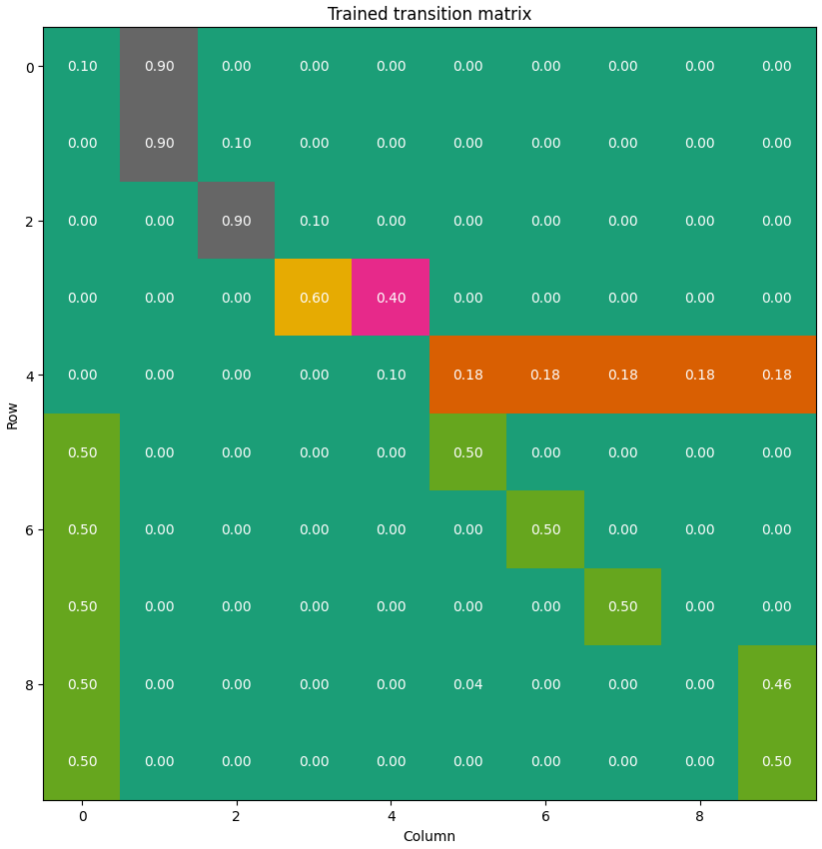


Figure 33: Trained transition matrix for entire model

This is a trained transition matrix, which combines the two matrices shown above. The good matrix is at the top left of the matrix, while the bad matrix is at the bottom right. The orange numbers shows the probability from going from the good model to the bad model, while the green numbers at the bottom left shows the probability of going from the bad model to the good model.

This entire matrix shows another training run than the one showed above, and the Good matrix is changed a bit to make sure the structure is correct. That means that the structure should be like the green fields in Figure 7, and the probabilities needs to be less than 1 (100%) to make sure the model can go from one state to another. The transition probabilities in this good model is set by testing and training results.

When testing a model like this, I'm sending in test data from two days, where one of the days had a washing machine run. In advance, I can see that the washing machine lasted for 1 hour and 40 minutes, and I can see the starting time. The model will put every measurement, which is every minute, into a state after predicting where it belongs. For this case, we have one day without using the washing machine, and on the second day, the washing machine was turned on at 14:48. In the results we then want to see that the entire first day is predicted to be in the bad model. The results showed this, and the model predicted state 9 the entire first day. On the second day, the model also predicted state 9 until the washing machine was turned on. When the washing machine turned on, the state was predicted as 8 and then 5. Later it changes between 9,8,5 and 6 during the washing machine cycle. That means that the model never enters the Good model,

which is where it should be when the washing machine is on.

During the testing of this model the results are quite bad, and in an attempt of making it better we have changed the data with an interpolation value of 10. In this way we get measurements every 10 minutes, and the results should improve.

4.3.1 Amount of training data

In this chapter we will look into how much data is needed for good results. It is beneficial for the overall system if we don't need that much data. In that case it would be possible to move the energy measuring plugs around to different appliances quite fast, and you could get a working system faster. It is also unnecessary to run the training with too much data as it can take a lot of time to get results.

To test the amount of data needed for a good result, the accuracy of the predictions for different amount of training data is tested. First out we have 50 training days for the Bad model, and 18 training days for the Good model. The model is trained and the outcome is predicted 10 different times to see the results. This is done since the outcome can be quite different each time, so I want to make sure I don't casually end up with one bad model that's ruining the results.

We are now testing with a model that have 3 states for the Good Model, and 3 states for the Bad model. The amount of states are chosen based on the fact that trying with 5 states per model gave quite bad results. The expected result is that we want it to go into the Good model, which is state 0,1 or 2 during the washing, which is the values at 234-243. Below I will show one run of the model to show the outcome of the first run with 50 training data for when the washing machine has been off and 18 training days where the machine has been on.

States: [5 3 5 0 1 2 2 2 2 2 2 2 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5

Here we can see that at the start the values are classified into state 5 and 3, which is great since the washing machine is not on. When the washing machine goes on, at value 234, the state changes to 0, and goes through 1 and 2 while the machine is on. When the washing machine goes off, at 244, the model predicts the states to be 5. This is a good prediction that gives the exact time the washing machine is on and off. The results from training and testing the model 10 times with these amount of training data is shown in Table 1.

	Good model	Bad model	Total results
1	90%	100%	95%
2	90%	100%	95%
3	90%	84.2%	87.1%
4	90%	84.2%	87.1%
5	100%	15.8%	57.9%
6	90%	84.2%	87.1%
7	100%	84.2%	92.1%
8	100%	100%	100%
9	100%	100%	100%
10	90%	99.6%	94.8%

Table 1: Score for training with 50 days for the Bad model and 18 days for the Good model.

The total result from the Good model is 95% correct predictions, and the results from the Bad model is 85%. All together, when both models counts equally we have the total results at 89.6%.

When looking at the states in Appendix D.1 we can see that the first two takes looks quite good. Then, at the third take, we can see that after the washing machine goes on, it does not return to the Bad model. It goes slightly into state 4 during the washing cycle, and then back to the Good model even though the washing machine is off. The same thing happens in take 4. Similar things happens in take 6 and 7. In take 5, it starts in state 2, and nothing happens before the washing machine gets turned off. After the wash, the state is correct and it predicts that those measurements belongs to the Bad model. The biggest problem in this take is that it starts in the wrong state, and since nothing special is happening the first 233 measurements it just stays in that state. Take 8,9 and 10 is quite good, with 100% score for 8 and 9. In take 4 there is one wrongly predicted value for the Good and the Bad model, giving a slightly worse score.

By decreasing the training data a bit at a time, we can look at the result and see how much training data is actually needed for a decent model. The next ratio we are testing are with 40 training days with no washing and 15 training days with washing in the washing machine. The results are shown in Table 2.

	Good model	Bad model	Total results
1	90%	100%	95%
2	100%	84.2%	92.1%
3	100%	84.2%	92.1%
4	90%	100%	95%
5	90%	84.2%	87.1%
6	90%	84.2%	87.1%
7	90%	84.2%	87.1%
8	90%	84.2%	87.1%
9	100%	99.6%	99.8%
10	100%	100%	100%

Table 2: Score for training with 40 days for the Bad model and 15 days for the Good model.

This amount of training data has an overall score of 92.2%, where the Good model scores 94%

and the Bad model scores 90.5%. This is better then the previous Table.

When looking at the states in Appendix D.2 we can see a lot of similar things as in the previous table. Like before, we can see that the model struggles to chose the correct state after the washing machine cycle. For take 2,3,5,6, 7 and 8 this is a problem. For take 1,4,5,6,7 and 8 the washing machine cycle gets predicted a bit too short, and the model predicts that it starts a bit late, or a value in the middle of the washing cycle gets predicted as the bad model. Take 1 is quite good, and is only one value off at the Good model. Take 9 and 10 is also good, with 9 having one value off at the bad model and 10 with a perfect 100%.

The next ratio for testing is 32 days of training data with the washing machine off and 12 days of training data when the washing machine is on. These results are shown in Table 3.

	Good model	Bad model	Total results
1	100%	99.6%	99.8%
2	100%	99.6%	99.8%
3	90%	100%	95%
4	90%	99.6%	94.8%
5	90%	99.6%	94.8%
6	90%	99.6%	94.8%
7	100%	100%	100%
8	100%	99.6%	99.8%
9	100%	15.8%	57.9%
10	90%	84.2%	87.1%

Table 3: Score for training with 32 days for the Bad model and 12 days for the Good model.

This test shows a result of the Good model at 95%, the Bad model got a score at 89.7% and the total score is 92.4%. That is the highest total score for now.

If we look at the predicted states in Appendix D.3 we can see that most of the runs are pretty good, with often 1 state off at both the Good and Bad model. One take is a bit different, and that is take 9. In take 9, state 2 is predicted at the beginning. It stays at state 2 until the washing machine is done. This gives a lot of wrong states when it should be in the Bad model, giving a quite bad result. The score for the Bad model in that run was only 15.8% meaning only 15.8% of the states were correct for the Bad model. Also, at the last run we can see the same thing that has happened earlier; it stays in the Good model after the washing machine is done, giving wrong predictions for the last 44 values.

Table 4 shows the result with 24 days of no washing and 9 days of washing.

	Good model	Bad model	Total results
1	100%	100%	100%
2	100%	15.8%	57.9%
3	100%	99.6%	99.8%
4	100%	15.8%	57.9%
5	100%	100%	100%
6	100%	0%	50%
7	100%	15.8%	57.9%
8	90%	84.2%	87.1%
9	100%	100%	100%
10	100%	99.6%	99.8%

Table 4: Score for training with 24 days for the Bad model and 9 days for the Good model.

The score of the Good model is 99% and the score of the Bad model is 63%. The overall score is 81%. These results are quite bad in comparison to the previous results. Especially when it comes to the bad model. It looks like the model is good at predicting when the washing machine is on, but if we take a closer look at the data in Appendix D.4, we can see that it also looks a bit random. For take 7, it predicted that both days were in state 1 the entire period; which is a wrong result. The Good model then got 'lucky' with correct predictions. Take 2,4,6 and 7 all stays in the Good model for the entire period before the washing machine is turned off, meaning they are in the wrong model for 233 values. This gives a lot of error, which results in bad scores. Even though it gives a lot of bad results, some of the takes are quite good as well. Like take 1,3,5, 5,9 and 10 are perfect or almost perfect! So this test gives kind of all or nothing in regards to results. We can see that it can predict quite good, but then all of a sudden we get really bad results.

Table 5 shows the result with 16 days of no washing and 6 days of washing.

	Good model	Bad model	Total results
1	100%	84.2%	92.1%
2	0%	100%	50%
3	100%	99.6%	99.8%
4	0%	100%	50%
5	100%	15.8%	57.9%
6	100%	84.2%	92.1%
7	80%	84.2%	82.1%
8	100%	84.2%	92.1%
9	90%	84.2%	87.1%
10	100%	84.2%	92.1%

Table 5: Score for training with 16 days for the Bad model and 6 days for the Good model.

The Good model has a score of 77% and the Bad model has a score of 82.1%. The overall score is 79.5%.

For this results it can be seen that no tests got a 100% score, and some of them has a 0% score for the Good model. That is not good, and not something reliable. By looking at the results

in the Appendix D.5 we can see that we have one run that it quite good, take 3, and for all the others we have quite a lot of wrongly places states. The test shows that the model is quite good at putting the first day into the correct state, but struggles after the washing machine has been on.

After looking at how good the models are, we also need to check how bad they are doing. How many false positive they give us.

The first confusion matrix in Figure 34 shows the results from Table 1, and highlights the true positive/negative values as well as the false positive/negative values. In this way it is easier to evaluate the model, and see what's good and bad.

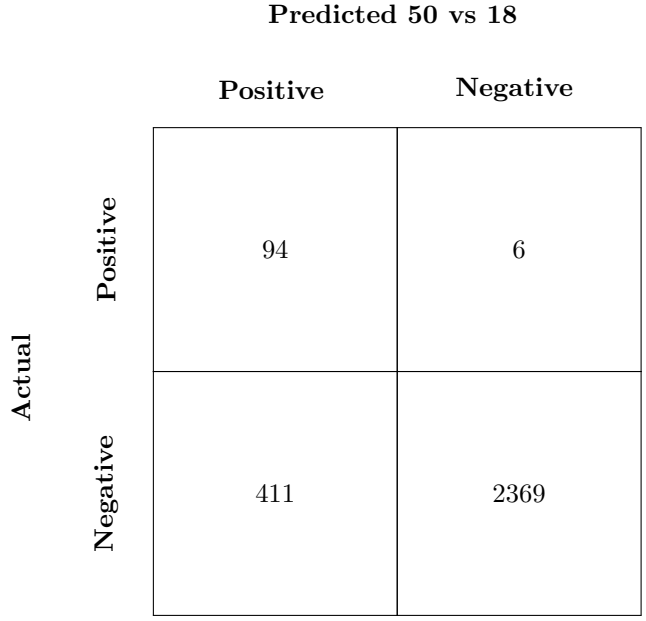


Figure 34: Confusion matrix showing the result of the data in Table 1.

This confusion matrix shows that we had 94 out of 100 correctly classified values for the good model. This gives 94 true positive values. We are then left with 4 false negative values, these are the values that should have been positive but are classified as negative. For the bad model we have 2369 true negative, and 411 false positive. We can see that the main problem of this model is the false positive values.
The next confusion matrix shows the results from Table 2, and is shown in Figure 35.

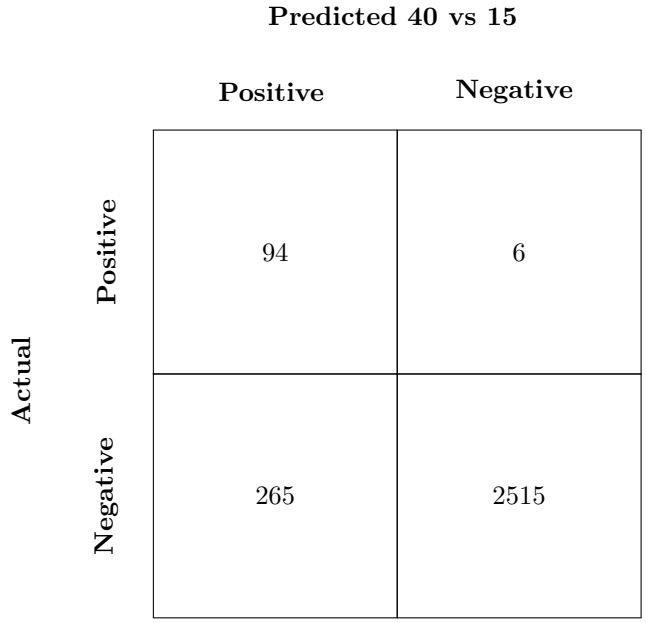


Figure 35: Confusion matrix showing the result of the data in Table 2.

This confusion matrix also shows the true positive/negative and false positive/negative values for the model. We can see that the true positive and false negative are the same as in the previous model. For the false positive and true negative it is a bit different. We get more true negative we get 2515, compared to 2369 in the previous model. This is quite a bit better, meaning that we also get fewer false positive in this case. That means that this is a better result than the previous one.

The next confusion matrix in Figure 36 shows the results from Table 3.

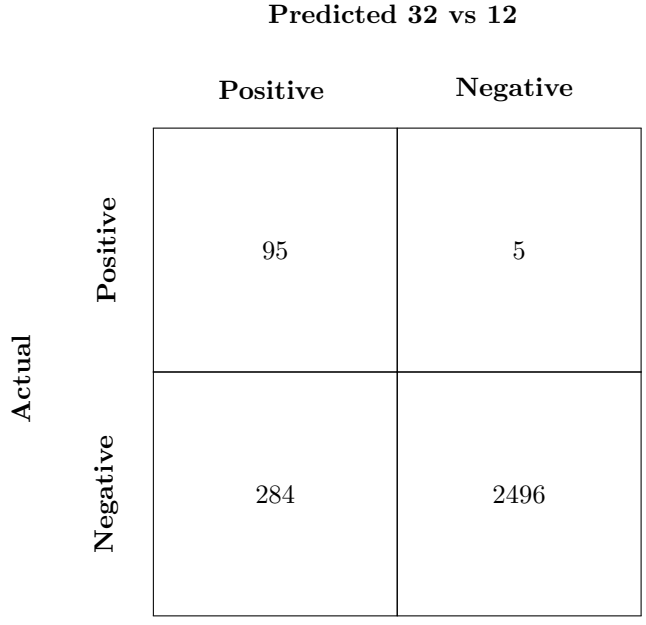


Figure 36: Confusion matrix showing the result of the data in Table 3.

In this confusion matrix we can see that the true positive is a little bit higher than previously, and we only have 5 false negative. When looking at the false positive and true negative we see that we have 2496 true negative and 284 false positive. This is a bit worse than the previous. That means it is both better and worse than the previous in different areas, but if look at the values i Table 3 we see that the overall score is a little bit better.

The next confusion matrix in Figure 37 shows the result from Table 4.

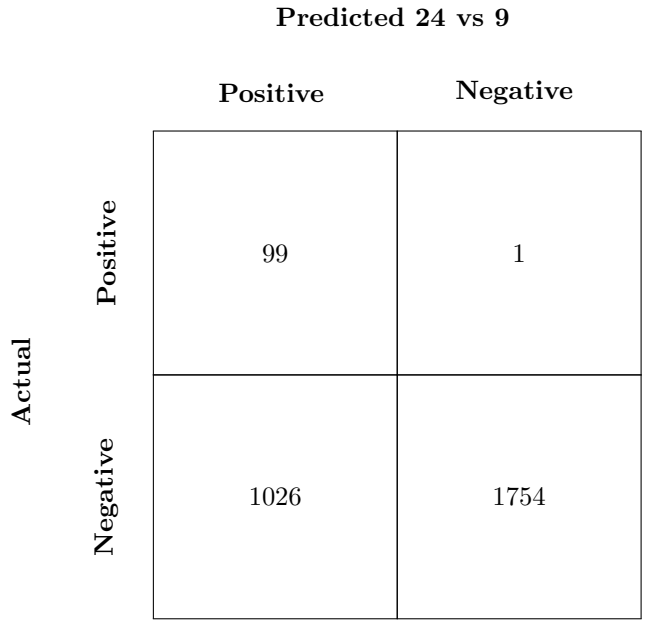


Figure 37: Confusion matrix showing the result of the data in Table 4.

In this confusion matrix we can see how the true positive is quite high, and we only have 1 false negative. That is better than previous! On the other hand we have 1754 true negative, meaning we have 1026 false positive. That is a lot false positive, and makes the overall model score worse than the others.

In the last confusion matrix in Figure 38 we can see the results from Table 5.

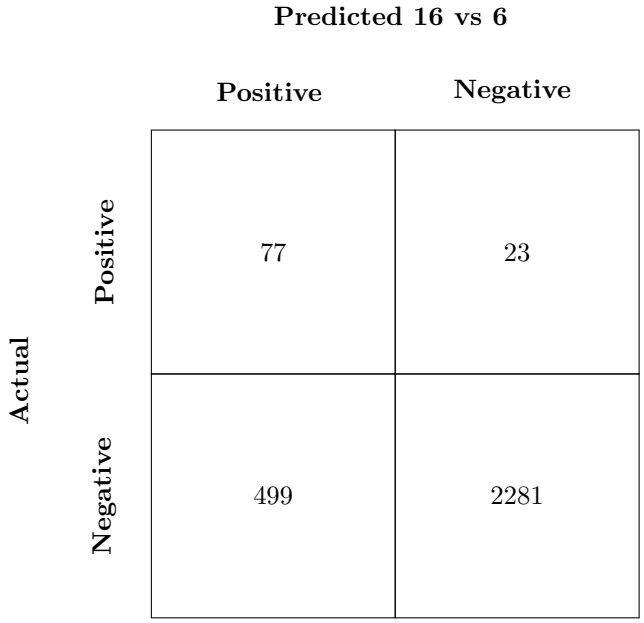


Figure 38: Confusion matrix showing the result of the data in Table 5.

In this confusion matrix we can see that the number of true positive has decreased a lot, and we only have 77 true positive. We then have 23 false negative, which is much higher than in the other confusion matrices. For the true negative we have 2281, which is not the best but also not the worst. We do have 499 false positive, which is way too much and overall the score of this model is not that good.

By looking at the results in the confusion matrix above, it is possible to see that the first three looks quite good and are quite equal. When we move to the last two we see that the accuracy decreases a lot. Table 6 shows the results from Table 1, 2, 3, 4 and 5 together in one table.

TD Good model	TD Bad model	Good model	Bad model	Total results
18	50	95%	85%	89.6%
15	40	94%	90.5%	92.2%
12	32	95%	89.7%	92.4%
9	24	99%	63%	81%
6	16	77%	82.1%	79.15%

Table 6: Score for different amount of training data.

TD Good model stand for amount of training days for the Good model and TD Bad model stands for amount of training days for the Bad model.

4.3.2 Amount of states

In the following chapter we will test how the model reacts to different amount of states in the model. This is tested on 20 different days in May, including both days when the washing machine has been off all day and when it has been on that day. Figures showing the washing machine cycles

for these days can be found in Appendix E.1. The model will get a score based on how well the Good and Bad model are doing, and they are all represented in different Tables for each test. In Table 7 we can see the results for testing with 3 states. This model is based on 32 days of training data for the Bad model and 12 days for the Good model.

Date	Washing Machine on/off	Good model	Bad model	Total results
08.05.2023	On	100%	99.3%	99.7%
09.05.2023	Off	100%	100%	100%
10.05.2023	Off	100%	100%	100%
11.05.2023	Off	100%	100%	100%
12.05.2023	Off	100%	100%	100%
13.05.2023	On	90.0%	61.2%	75.6%
14.05.2023	On (x2)	46.2%	93.8%	70.0%
15.05.2023	Off	100%	100%	100%
16.05.2023	On	100%	99.3%	99.7%
17.05.2023	Off	100%	100%	100%
18.05.2023	On	100%	99.3%	99.7%
19.05.2023	Off	100%	100%	100%
20.05.2023	Off	100%	99.3%	99.7%
21.05.2023	On	100%	100%	100%
22.05.2023	Off	100%	100%	100%
23.05.2023	Off	100%	100%	100%
24.05.2023	Off	100%	100%	100%
25.05.2023	Off	100%	100%	100%
26.05.2023	Off	100%	100%	100%
27.05.2023	On	30%	99.3%	64.7%

Table 7: Testing with 3 states per model

In this table we can see a mix of different results. In Appendix E.3 the predicted states can be seen. The washing machine has been off for 13 of these days, and 12 of these days are predicted perfect. The last one, 20th of May, have one wrong value. This measurements had a bad start when the model predicted that the washing machine were on for 10 minutes before it went off for the rest of the day.

For the days when the washing machine were on we have different kinds of results. We can see the same in some of these, were the Bad model were 1 value off. This is when the Bad model scores 99.3%. For the 13th of May, the model predicted correct from the start but after the washing machine was done the model kept predicting that the washing machine were on. For the 14th of May, the washing machine were on two times. One time for a 30 minute wash, and then a 100min wash an hour later. This seemed to have confused the model. The model predicted that the washing machine went on at the correct time, but predicted that the first 30minute wash lasted for 120 minutes, and this prediction stopped in the middle of the seconds run. This is an understandable mistake since the model is mostly trained on longer washes, and not that many short washes but it would still be nice if it made the correct decision. Otherwise most of the results looks quite good, except the last one. For this wash the model failed to predict when the washing machine were on and only got 3 out of 10 values correct.

The overall score of this model is 95.5%.

Table 8 shows the results from testing with 2 states per model. The training data still consists of 32 training days for the Bad model and 12 days for the Good model.

Date	Washing Machine on/off	Good model	Bad model	Total results
08.05.2023	On	90%	100%	95%
09.05.2023	Off	100%	100%	100%
10.05.2023	Off	100%	100%	100%
11.05.2023	Off	100%	100%	100%
12.05.2023	Off	100%	100%	100%
13.05.2023	On	90%	100%	95%
14.05.2023	On (x2)	100%	100%	100%
15.05.2023	Off	100%	100%	100%
16.05.2023	On	70%	100%	85%
17.05.2023	Off	100%	100%	100%
18.05.2023	On	73.0%	100%	86.5%
19.05.2023	Off	100%	100%	100%
20.05.2023	Off	100%	100%	100%
21.05.2023	On	90%	100%	95%
22.05.2023	Off	100%	100%	100%
23.05.2023	Off	100%	100%	100%
24.05.2023	Off	100%	100%	100%
25.05.2023	Off	100%	100%	100%
26.05.2023	Off	100%	100%	100%
27.05.2023	On	20%	99.7%	59.9%

Table 8: Testing with 2 states per model

In this table we can also see different results, and by looking at the days when the washing machine is off we see that these days are all perfectly predicted. There are no incorrectly predicted values for these cases, which is good. The predicted states can be found in Appendic E.2. For the 14th of May, which was a problem for the previous model, the results are also 100%. The thing we can recognize in this model is that the score for the Good model when the washing machine is on is a bit worse. Only one of them is 100%, and the rest is a bit incorrectly predicted. We can see that the most incorrect prediction is the last one on the 27th of May. This was also a problem in the previous test, so both models had problems predicting this one.

The overall score of this model is 95.8%.

Table 9 shows the results when testing with 4 states for each model. This model needed more training data to work, and therefore this model is trained with 40 days of training data for the Bad model and 15 days for the Good model.

Date	Washing Machine on/off	Good model	Bad model	Total results
08.05.2023	On	30%	32.8%	31.4%
09.05.2023	Off	100%	100%	100%
10.05.2023	Off	100%	100%	100%
11.05.2023	Off	100%	100%	100%
12.05.2023	Off	100%	100%	100%
13.05.2023	On	100%	100%	100%
14.05.2023	On (x2)	100%	35.1%	67.55%
15.05.2023	Off	100%	100%	100%
16.05.2023	On	80%	100%	90%
17.05.2023	Off	100%	100%	100%
18.05.2023	On	100%	45.9%	72.95%
19.05.2023	Off	100%	100%	100%
20.05.2023	Off	100%	100%	100%
21.05.2023	On	80%	98.5%	89.3%
22.05.2023	Off	100%	100%	100%
23.05.2023	Off	100%	100%	100%
24.05.2023	Off	100%	100%	100%
25.05.2023	Off	100%	100%	100%
26.05.2023	Off	100%	0%	50%
27.05.2023	On	20%	56.7%	38.4%

Table 9: Testing with 4 states per model.

For these results we can see that the score of the Good model got a bit better, but the score of the Bad model got a lot worse. The predicted states can be found in Appendix E.4. In one of the runs, 26th of May, the score of the Bad model were actually 0. By looking at the predictions we can see that a lot of the scores of the Good model looks good because the prediction sometimes only consists of the Good model. If the entire day is predicted as the Good model, meaning the washing machine is on; it's not good.

The overall score of this model is 86.9% making it a lot more incorrect than the previous ones.

Table 10 summarizes the results from this chapter:

States	Good model	Bad model	Total results
2	91.6%	99.9%	95.8%
3	93.3%	97.6%	95.5%
4	90.5%	83.45%	86.9%

Table 10: Score for testing with different amount of states.

It can be seen that the model with 2 states is the best overall, while the model with 3 states are second best. The model with 2 states is the best model in regard of the Bad model, while the model with 3 states are better at predicting the Good model.

4.3.3 Testing with Tibber Pulse Data

To see how a model would work on Tibber pulse data, a test were made to check this. For this test a model with 3 states were used, and the model were trained with 32 days for the Bad model and 12 days for the Good model. The consumption data from the day we tested are shown in Figure 39.

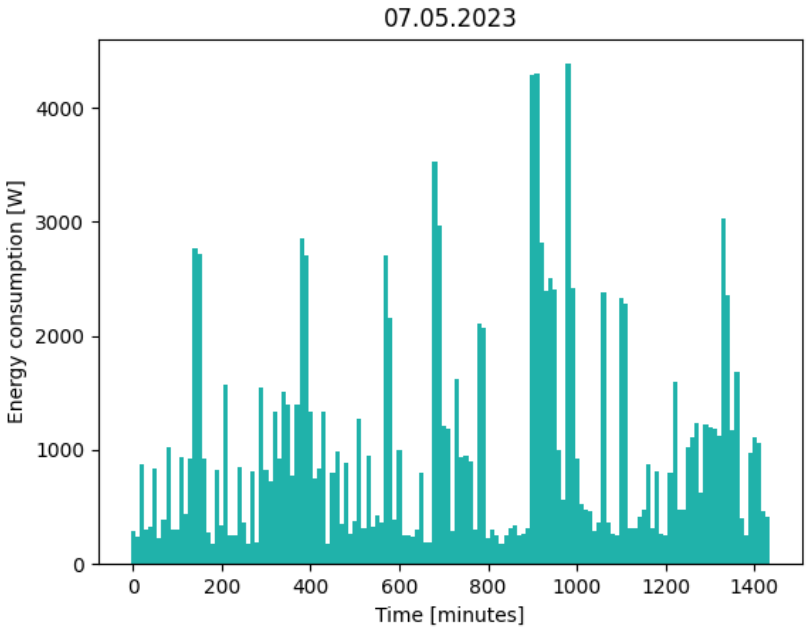


Figure 39: Pulse data for every 10 minutes 07.05.2023

During the first two runs of the model, the model predicted the state 4 for the entire day. When testing the model the third time, some more interesting results appeared. The model detected that the washing machine were turned on 6 times during that day. This makes sense, since the pulse data has 6 big peaks that day. During some of the peaks, that are not washing machine peaks, the model predicted a quite short 'washing machine run'. This means between 3-4 states of the washing machine 'being on', which equals 30-40 minutes. When the washing machine is actually on, between approximately minute 900-1000, the model predicts that the washing machine to be on for 12 states. This is 120 minutes, which is correct. By looking at this, we might be able to separate the correct predictions with the 'fake' predictions if we know the washing machine length. Unfortunately this can vary from 30-150 minutes, so with no labeling/knowledge this would not help much. Figure 40 shows the transition matrix for this run.

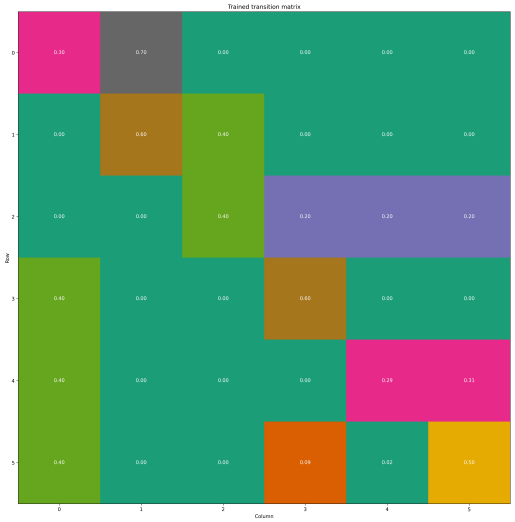


Figure 40: Transition matrix for training before testing on pulse 07.05.2023

The predictions from this run is shown below:

states = [4 5 4 4 4 4 5 4 4 5 5 4 0 1 1 1 2 2 2 4 4 4 5 5 4 4 5 4 5 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 4 4 4 4 5 4 4 4 4 4 4 0 1 1 2 4 5 5 5 5 4 5 0 1 1 2 4 4 4 4 4 4 0 1 1 2 2 2 2 2 2 2 2 5 0 1 1 1 1 1 1 1 1 1 2 4 4 4 4 4 4 4 5 0 1 1 2 2 4 4 4 4 4 5 5 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 4 4 4 4 4]

The states above shows the predicted state for each 10-minute for the 7th of May. The states 0,1 and 2 describes the washing machine and the Good model, while the numbers 3,4 and 5 describes the Bad model which should be everything that's not the washing machine.

5 Evaluations

It is now time to evaluate the different parts of the paper, and I will evaluate them one part at a time starting with the System.

5.1 Measurement system

The system is the Z-Wave system explained in Chapter 3.1 and executed in Chapter 4.1. It starts with the plugs that successfully sends data through Z-Wave to the Raspberry PI and then Home Assistant. We know this because we can see the measured data in various plots in Chapter 4.1. Figure 12 shows the measurements from the Z-Wave plug attached in the electrical outlet for the washing machine. The plot itself makes sense given that a washing machine first uses some time to warm up water, and then it starts running. We know that water heating takes a lot of energy, and we also have a peak at the end showing the centrifuging. Figure 14 shows washing machine measurements from four different days, and we can see that the measures are quite similar but at the same time has their own characteristics. Like on the bottom left of the figure we can see a day with two washes, one 30 degree wash of 30 minutes and then a normal 40 degree wash which lasts for a lot longer.

Figure 15 shows the Tibber Pulse measurements from the same day as in Figure 12. This figure shows the entire consumption for the household for that day. The washing machine should be in one of the peaks in the figure. It can be seen that the Tibber Pulse measurements have quite a lot of peaks, and it might be hard to distinguish where the washing machine 'peak' is among the other peaks.

Since we have the measurements the data is successfully send into Home Assistant. To be able to see and fetch the measurements from outside of the home port forwarding is set up. This is also tested and it was then possible to automate the process of getting the data and put it into Timestream, the database. The automation was set up, and I was able to get the data from Timestream. That means that the port forwarding and automation works well. Given that we are able to get the data from Timestream it can also be concluded that the automated process showed in Figure 16 works as it should.

5.2 Data processing

From Chapter 4.2.1 it is possible to see how the data has been extracted from Timestream using the TimestreamQueryClient. We can see that using the two different queries it is possible to get the values from Tibber pulse, and other appliances like the washing machine. These values are put into an object and then into a file containing all the information from that day. This gives an clear overview over the different data we have gathered, and makes it easy to use later.

In the next chapter it can be seen that the raw data from the different plugs are quite different, with a different interval on the measures. Tibber pulse takes measurements every 2. second while the Z-Wave plugs are more random in the time period for measurements.To solve this, linear interpolation is used. The process is explained in chapter 4.2.2 and the results are shown in the figures below. Figure 17 shows one day of measurements from when the washing machine has been on, where the first one is before processing and the second figure is after using linear interpolation. We can see quite a difference, and this is important for comparing and working with the measurements later.

In Figure 18 we can see how the measurements from the washing machine and Tibber Pulse is presented on top of each other after interpolation. The seagreen values are Tibber Pulse and the orange values are the washing machine. Based on this plot we can see that it looks like the washing machine is places the correct place. The peak is within a peak on the Tibber Pulse measurements, so it looks correct. The values would most certainly not match if we had made mistakes with the interpolation.

Further down in the chapter we are checking other time periods for linear interpolation, with 5, 10 and 30 minutes if Figure 19, 20 and 21. We can see that 5 and 10 minutes looks okay, and that the washing machine measurements still has its characteristics. When moving over to 30 minute-interpolation we see that the washing machine measurements are quite low, and we lost the peak in the measurements. This is because the linear interpolation step is bigger than the peak itself, making it disappear. This tells us that we need to make sure the interpolation time is smaller than the biggest characteristics we need to bring with us.

Figure 22 shows Tibber Pulse and washing machine data for four different days in May. This figure shows what we assumed after Figure 18; the linear interpolation is accurate. We have 5 different washing machine peaks in this figure, and they all fits into a Tibber Pulkse measurement peak. Especially in the Figure 22c, where there is two peaks quite close. They both go correctly into the two peaks from Tibber Pulse, meaning the peak matches quite perfectly.

In chapter 4.2.3 we add more appliances to the measurements. Figure 23 shows measurements of Tibber Pulse, washing machine and two ovens at the same day. In is interesting to see appliances together on a plot like this to see how much each of the appliances consumes in form of energy. The pattern on the appliances is also useful to see how they work. For example for ovens you can look at the pattern, and see if it is on the entire day. Then you can turn it down 1-2 degrees and see the change. Figure 24 shows the same for four different days during the spring. They look quite different, but still consists of the same appliances. My guess is that the outside temperature has been quite high in the first and last plot, meaning the ovens did not use that much energy these days. It is hard to see how the oven measurements fits into the Tibber Pulse measurements and if they are perfectly aligned since they doesn't have the peaks and characteristics like the washing machine.

In the end of the processing chapter we take a look at how it is possible to break down the Tibber Pulse measurements one appliance at the time. Figure 25 shows first of all the Tibber Pulse data for one day, and then how the signal looks when one and one appliance is removed. In Figure 25b we can see how the signal now looks like in orange when the washing machine measurements are gone. Furter we can see how the measurements looks like when the two ovens are removed. One can see that the signal is still quite messy, with a lot of peaks and a lot of consumption we don't know what is. But by repeating this process with more appliances one should be able to get to the bottom of what appliances are consuming how much.

Figure 26 shows a kind of summary of the Tibber Pulse data before removing measurements, then the Tibber Pulse data after removing and lastly the removed measurements that consists of washing machine and two ovens. Even though the Tibber Pulse data looks quite like before, messy and hard to understand, we still see that we have been able to remove quite a lot of the measurements. Continuing the work would make the measurements easier to understand. When it comes to recognizing appliances we might have a hard time recognizing the ovens as they don't have the characteristics that the washing machine has with it's peak.

In chapter 4.2.4 it is briefly looked into using Short Time Fourier transform for finding characteristics from the different appliances, to check if this could be something to use in the machine learning. The results are not great, but at least in the end with the ovens it is possible to see some-

what clear lines at certain frequencies meaning it might be able to use it for recognizing ovens. However, the focus further on has been on the washing machine using the measured values but this chapter shows this also might be a decent approach for some appliances.

5.3 Hidden Markov Model

In Chapter 4.3 it is explained how to make the HMM in two parts; the Good model and the Bad model. In the beginning a model with 5 states is shown for both the Good and Bad model. They are each trained and put together to the big model. Figure 33 shows the finished model and represents what we saw earlier in Figure 7. During testing this model did not work well. We tested the model with 1 min interpolation data, and 5 states for each model. This model was wrong most of the time, and really bad at recognizing the washing machine. Because of this it was decided to move on with 3 states per model and 10 min interpolation data. This was based on the fact that it is hard for the model to recognize a too long sequence of data. With the 1 minute interpolation we have 1440 measurements for one day, while with the 10 minute interpolation we only have 144 measurements per day. This makes it easier for the model, and as we saw in Figure 20 we still have the correct characteristics from the washing machine.

In Chapter 4.3.1 it is tested how much training data that is needed for the model to work well. During this test the models are trained 10 times, and the first test is with 50 days of training data for the Bad model and 18 days of training data for the Good model. Each model has 3 states, and Table 1, and we can see that the overall results are 89.6% correct predictions, with a 85% score for the Bad model and 95% for the Good model. The most common wrong predictions were that the model failed to switch back into the bad part after a washing machine run and sometimes it started in the wrong model and were stuck until the washing machine had ran.

Table 2 shows the results for 40 days of training data for the Bad model and 15 days for the Good model. The results shows that the overall score was 92.2%, with a Good model score of 94% and a Bad model score of 90.5%. The most common errors were the same as before. Some of the runs also predicts a bit shorter washing machine run. Table 3 shows the results of the tests for 32 days of training data for the Bad model and 12 days for the Good model. This gives an overall score of 92.4%, with a Good model score of 95% and a Bad model score of 89.7%. This is the best results for now.

The testing continues, and for the tests with 24 days of training for the Bad model and 9 days of training for the Good model we can see the results in Table 4. The overall score is 81%, with a Good model score of 99% and a Bad model score of 63%. This is a lot worse than the previous runs. Table 5 shows the last test for this purpose with 16 days of training for the Bad model and 6 days of training for the Good model. This gave a bad result of 79.5%, with a Good model score of 77% and a Bad model score of 82.1%. For these last two tests the results are a lot worse and even though they sometimes might look good for the Good model they are really bad for the Bad model.

By looking at these results we can see that the three first tests are quite good and the third one, with 32 training days for the Bad model and 12 training days for the Good model are the best. I would predict that the model with the most training data would be the best, but apparently not. I think the three first tests are quite similar, and coincidences that makes one of them better than the previous one. Since the results gets a lot worse when decreasing the training data more we can conclude with the fact that 32 training days for the Bad model and 12 training days for the Good model is sufficient. Figure 34, 35, 36, 37 and 38 shows confusion matrices displaying the results from the Tables discussed above. In these tables we can see how many values are incorrectly

predicted, and it gives the results in a different way showing how good and how bad the models are doing. The overall results are showing the same, and our conclusion still stands as it is. Table 6 shows the results from the different amount of training days in the same table. This way it is easier to compare the values, and we can still see that the three first tests are quite equal with their own results.

In the next chapter we are testing different amount of states, to see if we can find the best fit for our model. These tests are done with data from 20 different days in May, and the Tables shows whether the washing machine has been on or off that day and the score of the different models. First we are testing with 3 states for each part of the model as before, still with 10 min interpolation values. The training data is 32/12 as tested earlier. Table 7 shows the results for these kind of test for 20 different days in May. The overall score of this model is 95.5%. We can see from the table that the model is quite good, but makes some mistakes every now and then.

One of the problems that happens is that at the beginning of the predicted states the model predicts one value from the Good model before it moves onto the Bad model. This is what happens when the score of the Bad model is 99.3%, that represents 1 incorrectly predicted value. One other case that happened were on the 14.05.2023, where the washing machine were on two times during the day. This seemed to confuse the model. The washes consisted of one 30 minute wash, and then about one hour later we had a 100 minute wash. The model predicted the start of the 30 minute wash correctly, but also predicted that it lasted for about 100 minutes as most of the washes do. It then also incorrectly predicted the washing machine to be off when the seconds wash were running. This is understandable, but still unwanted.

In Table 8 we can see how the model predicted the different days with 2 states for each model. If we look at the 14.05.2023 for the 2-state model we can see that the model predicted that day perfect. That means that this model did better for that case. Overall we can see that the Bad model does a lot better with 2 states, while the Good model does a bit worse at some days even though it was better the 14th of May. The score of this test is 95.7% which makes in slightly better that the test with 3 states per model. In Table 9 we can see the score for testing with 4 states. This model needed a bit more training data to work as it has more states. This is trained with 40 days for the Bad model and 15 days for the Good model. This model has an overall score of 86.9%, which is a lot worse than the previous ones. We can see that at certain days this model scores quite bad. As an example we can see that the 8th of May is quite bad, 26th of May is quite bad as well as 27th of May. By looking at 14th of May we can see that the Good model scores good, but the Bad model is quite bad meaning the model scores quite bad at that day.

By looking at these tests we can conclude that the model with 2 states for each model is the best we have tested. The worst day for this model was the last day, May 27th. This day has been quite bad for all the models to predict. Otherwise I would say that it is quite good, but could have been better. The bad model for this model scores 100% almost every day, which is quite good. The recurring error is that the Good model is predicted a bit too short; meaning the model think the washing machine is on for a shorter period than it actually is. This means that the model does well at predicting if the washing machine has been on during the day, that was the original goal. The timing is not perfect, since it thinks the washing machine run is shorter; but as mentioned earlier we would rather have a mistake like that then a mistake were the model predicts that the washing machine is one a day when it is off.

At the end of the Hidden Markov model problem solving there was made an attempt of predicting if the washing machine were on during a day with Tibber pulse measurements. Figure 39 shows the energy consumption during that day, with a lot of peaks that might look like a washing machine. As assumed, the predicted values shows that the washing machine is predicted to be on

6 times during that day. This is as assumed, since the model is not trained on Tibber pulse data. This was mostly just a test to see how the model would react. For training the model with Tibber pulse data, it might also be smart to send in other parameters like the derived measurements to look at the changes in measurements and not only the measurement itself.

6 Discussion

The system that has been developed in this paper is a good start of an energy saving system for households. The overall system is divided up into three parts during the paper, which is the measuring system, data processing and Hidden Markov model. The paper goes through an introduction, description of the system, problem solving with results and then evaluation of the system.

The measurement system is successfully set up with Z-Wave plugs, Z-Wave hub, Home Assistant and storage on Amazon Web Services. The port forwarding made it possible to automate the fetching process from Home Assistant to make sure the data is stored in Timestream and S3 every day. The plugs can communicate with each other and the hub without any problems, and then send the data to Home Assistant. The data is then successfully sent to storage in AWS as seen in various figures. By comparing measurements from Timestream with knowledge of the usage of appliances, we can conclude that the measuring system works as planned. It might need some minor changes if you were to add more measurements or change something, put the system should handle that well and the process should be easy.

The data processing part is where we process the data to make sure all the data has the same time interval, to make sure it would be possible to compare them to each other. In the beginning we can see how to successfully get the measurements from Timestream, and then we move into the linear interpolation. From plots we can see how we can interpolate with different steps, and we can see what kinds of limits we get. By looking at plots from different days seeing how the washing machine fits into the Tibber pulse data we can conclude that the interpolation looks quite successful. In some of the plots we see that we get negative values when we subtract all the measured appliances from the Tibber pulse data. This can possibly be an affect from the interpolation, otherwise it can be from the measurements itself. The washing machine measurements have a tendency to measure a small value even though it is off. This seems a bit wrong, and this might cause the negative values since the washing machine is not actually using any energy when it is off. This part of the paper shows us the interpolation, and how it is possible to subtract certain appliances from the total consumption. This gives us an idea of how we can continue the process to decompose the total consumption measurements, and the further work would be easy to continue into.

In the part where we make and test the Hidden Markov model we focus on first making the model, and then testing the model by tweaking on different things. First of all the amount of needed training data is tested with different amounts. By looking at the results we see that one of the tests were better than the others; so the conclusion is that this amount of training data were the best fit for our model. This might be a bit strange since in general more data means more accuracy. The reason might be that the first washes in the training data fit the test data better, and by adding more training data of less similar washes the results could have gotten worse. Another reason could also just be that it is coincidences that makes that model better. The three models with the most training data gave quite equal scores, and all gave good results.

For the next part of the Hidden Markov model the amount of states are tested; with 2,3 and 4 states for each model. The one with 4 states scored a lot worse than the ones with 2 and 3. For the ones with 2 and 3 states they were quite equal when it came to score, were one of them were better at predicting the Good model and the other were best at predicting the Bad model. Here one would have to pick the best amount of states for the task. For my task of recognizing if the washing machine has been on during the day; 2 states gives the best results. The worst thing about this model is that it predicts that the washing machine has been on for 10-30 minutes less then the actual run. While for the model with 3 states, the model did predict that the washing machine were on during some days when it was off.

In the end of the Hidden Markov model chapter a small test were done to see how the model acted on Tibber pulse data, mostly just to see how it reacted and see what should be done further. The results was that the model predicted that the washing machine were turned on 6 times during the day, one for each peak in the Tibber pulse measurements. This makes totally sense since the model is not trained on washing machine data. This means that we need to train a model for detecting the washing machine on the Tibber pulse data, as assumed, but still the model were able to find the washing machine in the Tibber pulse data which is kind of nice even though it is a test with a lot of incorrectly predicted values. One big challenge with testing with Tibber pulse for this system, is that the total consumption data consists of both my household and one apartment that is rented out. Based on this I will not know exactly when or if a washing machine has been on during a day, and it is hard to correctly verify the predictions of a model for certain appliances.

All in all we have a working system, with the different parts working all together; but to reach the goal of an energy saving system there is a lot of work left. Especially from the Hidden Markov model and further on, to actually make the system.

7 Future work

From previous discussion and results it can be seen that we have a good measurement system that does what it is supposed to do, and we are able to add more Z-wave product if wanted. Because of this I would say that we don't necessarily need to improve the measuring system in further work. The same counts for the processing part of this report. It looks like it work quite well, and there is no immediate things that needs fixing. By looking at other appliances you might have to change the interpolation interval, but the method in itself works okay. With that said, there might be other better ways of processing the data before giving it to the Hidden Markov model; and that's something that might be interesting to work on further.

Then we are at the Hidden Markov Model step. This is where there will be a lot of things for further work. For now we see that the model is able to recognize if the washing machine has been on or off during a time period based on measurements from only the washing machine. This gives a proof of concept, but is only a small part of an overall energy saving model. For further work one should look into the cases where the washing machine is predicted to be on the first 10 minutes of a day, since that might be because of something wrong when setting up the model. Secondly a model should be trained on the Tibber Pulse data, so that we would be able to recognize whether an appliance has been on during a day only based on the Tibber Pulse measurements. The method for this could be to use the trained model we have right now together with a new model trained on Tibber Pulse data. The model trained on plug-data could be used to label the Tibber Pulse data, and could be used when testing the model. If we trust the plug model, we can compare the results from the plug model with the results from the Tibber Pulse model to get the results.

After this is done, the focus should be on adding models for more appliances, so that we are able to recognize more appliances. This would happen one at a time, with most likely a time with plug measurements and then Tibber Pulse measurements. This could be appliances like dishwasher and cooking oven, as they are big energy consumers. After being able to recognize more appliances, the work with the report should start. The report should have measuring data for a period, probably a week or a month, with energy consumption data. The data should include total consumption, daily consumption, consumption data from similar households and most importantly the consumption data for the specific appliances measured in this system. A lot of this data is already given from Tibber, and can easily be extracted from the S3 bucket on AWS and put into the report. The specific appliances that are measured comes from the Hidden Markov models, and needs to be put in the report. As well as basic measurements, the system should compare the consumption pattern to the energy prices at that moment to see whether it was smart or really stupid to use an appliance at a given time. If a user puts on the washing machine every time the energy price is really high, the user should get some feedback on this. As well as the feedback the user should get tips on when to use the appliance, and data on what the user could save by changing the time.

The information from the system could be sent to the user via email, but it would be nice with a website/application in addition to that. The weekly or monthly information and updates could be send on email, and then all the data should be stored on a web site. This would make it easy to see previous consumption, and see the changes over a bigger time period. The website should come with daily tips to the user based on power prices, weather and consumption pattern. If the system can learn that a user will use the washing machine every Saturday, it could come with a warning on Saturdays if the energy prices are especially high that day.

References

[1] Bharathi. Latest guide on confusion matrix for multi-class classification. https://www.analyticsvidhya.com/blog/2021/06/confusion-matrix-for-multi-class-classification/. Retrieved 2023-05-27.

[2] Tobias Gausemel Backe. Bli mer miljøvennlig og spar energi hjemme. https://blogg.fortum.no/bli-mer-miljovennlig-og-spar-energi-hjemme. Retrieved 2023-04-20.

[3] N. Iivari S. Tuomela, M. de Castro Tome´. Impacts of home energy management systems on electricity consumption. 2021.

[4] M. Ito, R. Uda, S. Ichimura, K. Tago, T. Hoshi, and Y. Matsushita. A method of appliance detection based on features of power waveform. 2004.

[5] Tibber. Fibaro wall plug. https://tibber.com/no/store/produkt/fibaro-wall-plug? Retrieved 2023-06-08.

[6] Fibaro. Manual. https://manuals.fibaro.com/content/manuals/en/FGWPEF-102/FGWPEF-102-EN-A-v2.1.pdf. Retrieved 2023-06-08.

[7] Popp. Smart outdoor plug. https://manual.zwave.eu/backend/make.php?lang=ensku=POPE700397. Retrieved 2023-06-08.

[8] AWS. What is amazon timestream? https://docs.aws.amazon.com/timestream/latest/developerguide/what-is-timestream.html. Retrieved 2023-06-08.

[9] AWS. What is amazon eventbridge? https://docs.aws.amazon.com/eventbridge/latest/userguide/eb-what-is.html. Retrieved 2023-06-08.

[10] AWS. What is aws lambda? https://docs.aws.amazon.com/lambda/latest/dg/welcome.html. Retrieved 2023-06-08.

[11] AWS. What is amazon s3? https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html. Retrieved 2023-06-08.

[12] Home Assistant. Install homeassistant operationg system. https://www.home-assistant.io/installation/raspberrypi. Retrieved 2023-06-08.

[13] Home Assistant. Onboarding home assistant. https://www.home-assistant.io/getting-started/onboarding/. Retrieved 2023-06-08.

[14] Home Assistant. Automating home assistant. https://www.home-assistant.io/getting-started/automation/. Retrieved 2023-06-08.

[15] Home Assistant. Setting up presence detection. https://www.home-assistant.io/getting-started/presence-detection/. Retrieved 2023-06-08.

[16] Sharon Shea. Z-wave. https://www.techtarget.com/iotagenda/definition/Z-Wave. Retrieved 2023-05-03.

[17] https://www.cloudflare.com/. What is https? https://www.cloudflare.com/learning/ssl/what-is-https/. Retrieved 2023-05-28.

[18] AWS. Timestream. https://docs.aws.amazon.com/iot/latest/developerguide/timestream-rule-action.html. Retrieved 2023-05-28.

[19] AWS. What is mqtt? https://aws.amazon.com/what-is/mqtt/. Retrieved 2023-05-28.

[20] Robert Zak. How to open ports and set up port forwarding on your router. https://www.maketecheasier.com/port-forwarding-router/. Retrieved 2023-06-05.

[21] https://help.konnected.io/. Set up home assistant with secure remote access using duckdns and nginx proxy. https://help.konnected.io/support/solutions/articles/32000023964-set-up-home-assistant-with-secure-remote-access-using-duckdns-and-nginx-proxy. Retrieved 2023-06-05.

[22] Cuemath. Linear interpolation formula. https://www.cuemath.com/linear-interpolation-formula/. Retrieved 2023-06-05.

[23] Raymond Kwok. Baum-welch algorithm for training a hidden markov model — part 2 of the hmm series. https://medium.com/analytics-vidhya/baum-welch-algorithm-for-training-a-hidden-markov-model-part-2-of-the-hmm-series-d0e393b4fb86. Retrieved 2023-06-08.

[24] Raymond Kwok. Viterbi algorithm for prediction with hmm — part 3 of the hmm series. https://medium.com/analytics-vidhya/viterbi-algorithm-for-prediction-with-hmm-part-3-of-the-hmm-series-6466ce2f5dc6. Retrieved 2023-06-08.

[25] AWS. @aws-sdk/client-timestream-write. https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/clients/client-timestream-write/. Retrieved 2023-06-12.

[26] AWS. Writes. https://docs.aws.amazon.com/timestream/latest/developerguide/writes.html. Retrieved 2023-06-12.

[27] hmmlearn. Api reference. https://hmmlearn.readthedocs.io/en/latest/api.htmlhmmlearn-hmm. Retrieved 2023-06-12.

Appendix

A Data storing - Lambda function

```
const https = require('https')
const { TimestreamWriteClient, CreateDatabaseCommand, WriteRecordsCommand }
= require("@aws-sdk/client-timestream-write");

const handler = async(event) => {
  const headers = {
    "Authorization": "Bearer ***",
    "content-type": "application/json",
}
const today = new Date()
const yesterday = new Date(today)
yesterday.setDate(yesterday.getDate() - 1)

const dateForFetching = yesterday.toISOString().split('T')[0]
  const date = `${dateForFetching}T00:00:00.000000+00:00`
  const allData = [];
  const entities = ['sensor.popp_smart_outdoor_plug_ip44_rated_electric_consumption_w',
  'sensor.power_hans_osnes_veg_16',
  'sensor.metered_wall_plug_switch_power',
  'sensor.metered_wall_plug_switch_power_2']
  for (const entity of entities) {
    const filter_entity_id = `?filter_entity_id=${entity}`;
    const data = await new Promise((resolve) => {
        https.request(
          {
          hostname: `lenakh.duckdns.org`,
          path: `/api/history/period/${date}${filter_entity_id}`,
          port: 8123,
          headers: headers,
          verify: false
          },
          (res) => {

            const responseData = []

            res.on("data", (data) => {
              responseData.push(data.toString())
            });

            res.on('end', () => {
              resolve(JSON.parse(responseData.join('')));
            })
          }
        ).end();
      });
      allData.push(data[0]);
```

```
  }

     const dimension = [
    {'Name': 'SensorData', 'Value': 'HomeAssistant',
 'dimensionValueType': 'VARCHAR'}
]
    const record = []
    for (const entity of allData){
      for (const value of entity){
        const date = new Date(value.last_updated)
        const timeForMeasurement = date.getTime()
        record.push({'Time': String(timeForMeasurement),
        'Dimensions': dimension,
        'MeasureName': value.entity_id,
        'MeasureValue': String(value.state) ,
        MeasureValueType: 'VARCHAR'})
      }

    }
     const records = []
  if (record.length>100){
    let j=0
      for (let i = 99; j < record.length; i+=100) {
        records.push(record.slice(j,i))
        j=i
      }
   }
   else  {
    records.push(record)
  }
  const client = new TimestreamWriteClient({ region: "eu-west-1" });
  for (const rec of records){
    const params = {
    DatabaseName: process.env.DATABASE_NAME,
    TableName: process.env.TABLE_NAME,
    Records: rec
  }

  const command = new WriteRecordsCommand(params);

  try {
    const data2 = await client.send(command);
    console.log('success')
  }
  catch (error) {
    console.log('Error:', error)
  }
  }
};
module.exports = { handler }
```

B Data processing - linear interpolation

```
def procesData(data, step):
    newDict = {}
    for key in data:
        newDict[key[11:19:1]] = data[key]
    widthArray = []
    value = 0
    for key in newDict:
        if (value == 0):
            value = key
            continue
        t1 = datetime.strptime(value, "%H:%M:%S")
        t2 = datetime.strptime(key, "%H:%M:%S")
        wattLength = t2-t1
        value = key
        wattLengthFloat = float(wattLength.seconds/60)
        widthArray.append(wattLengthFloat)

    widthArray.append(0.1)
    x_axis_minutes = []
    startTime = list(newDict.keys())[0]
    for key in newDict:
        t12 = datetime.strptime(startTime, "%H:%M:%S")
        t22 = datetime.strptime(key, "%H:%M:%S")
        wattLength2 = t22-t12
        wattLengthFloat2 = float(wattLength2.seconds/60)
        x_axis_minutes.append(wattLengthFloat2)

    i = 0
    x_arr = []
    for item in x_axis_minutes:
        val = item + (widthArray[i]/2)
        x_arr.append(val)
        i = i+1
    values = list(newDict.values())
    LI_XArr = []
    LI_Values = []
    for i in range(0, int(x_arr[-1]), step):
        newXArr.append(i)
        y_interp = interp1d(x_arr, values, fill_value="extrapolate")
        value = y_interp(i)
        newArr.append(value)

    return LI_XArr, LI_Values
```

C Hidden Markov Model

```
#Setting the start probability of the model
startprob = np.array([0, 0, 0, (1/3),(1/3),(1/3)])

#Initializing the Bad model transition matrices
badModel = np.zeros((3, 6))
for i in range(3):
    badModel[i, 0] = 0.5

badModelLeft = np.zeros((3, 3))
for i in range(3):
    badModelLeft[i, 0] = 0.5

badModelRight = np.zeros((3, 3))
for i in range(3):
    badModelRight[i, 0] = 1
startProbBadModel = np.zeros(3)

#Initializing the actual Bad Model
badMatrixModel = hmm.GaussianHMM(
    n_components=3, covariance_type="full", n_iter=100)
badMatrixModel.startprob_ = startProbBadModel
badMatrixModel.transmat_ = badModelRight

#train model to convergence
trainIterations(badMatrixModel, noWashTrainingData)

#Initializing the Good model transition matrix
#The values in the matrix is set after training and testing
goodModel = [[0.3, 0.7, 0, 0, 0, 0],
             [0, 0.6, 0.4, 0, 0, 0],
             [0, 0, 0.4, 0.2, 0.2, 0.2]]

goodModelLeft = np.zeros((3, 3))
for i in range(3):
    goodModelLeft[i, 0] = 1

startProbGoodModel = [1, 0, 0]

#Initializing the actual Good Model
goodMatrixModel = hmm.GaussianHMM(
    n_components=3, covariance_type="full", n_iter=100)

goodMatrixModel.startprob_ = startProbGoodModel
goodMatrixModel.transmat_ = goodModelLeft

#train model to convergence
trainIterationsGoodModel(goodMatrixModel, trainingArrayWM)
```

```
#Add the to matrices together, both the Good model matrix and the Bad model matrix
transMat = np.concatenate((goodModel, badModel))

# Initialize the entire model
model = hmm.GaussianHMM(n_components=6,
                        covariance_type="full")
# set mean and var values from previous training
mean, var = badMatrixModel.means_, badMatrixModel.covars_
mean2, var2 = goodMatrixModel.means_, goodMatrixModel.covars_

#Set startprob and transmat
model.startprob_ = startprob
model.transmat_ = transMat
#Train the model
model.fit(trainingArr.reshape(-1, 1))

#add values together, and then add them to the model
mean3 = np.concatenate((mean2, mean), axis=0)
var3 = np.concatenate((var2, var), axis=0)
model.means_ = mean3
model.covars_ = var3
transmat_trained = model.transmat_

#Overwrite training and put the correct goodModel matrix in the model to make sure
#the pattern is correct for the Good model
transmat_trained[: 3, :] = goodModel
#Set the transition values from the Bad model to the Good model.
transmat_trained[3: 6, 0: 3] = [[0.4,0,0],[0.4,0,0],[0.4,0,0]]
badMatr = badMatrixModel.transmat_
#Normalize the values in the Bad model so they are 0.6, to make sure the sum of the row is 1.
for i in range(3):
    badMatr[i, :] = badMatr[i, :]/np.sum(badMatr[i, :])*0.6
transmat_trained[3: 6, 3: 6] = badMatr

#Get the predicted states by using viterbi
logprob, states = model.decode(testArr.reshape(-1, 1), algorithm='viterbi')
```

D Results - amount of training data testing

D.1 50 - 18 training data

https://docs.google.com/document/d/1ORzUHURGehmEol-qQThIfpEBRkuLYMYBNn9WphfGzXs/edit?usp=sharing

D.2 40 VS 15 training data:

https://docs.google.com/document/d/1Wgoq2llp1DzPbz_ZTxVYa_PEEaaOHNJz5Qpegn5oxP8/edit?usp=sharing

D.3 32 VS 12:

https://docs.google.com/document/d/1wdOhzDLSHS973eY6Efmsm2EuoTiQBNYp7aihPqciQN4/edit?usp=sharing

D.4 24 VS 9

https://docs.google.com/document/d/1WisJqysu3srzSn6qUTbmLapMbl2YXllLt-tFPgdxTH8/edit?usp=sharing

D.5 16 VS 6:

https://docs.google.com/document/d/1ETjOKvi3_asUK8sDtPWBucLFTT_WINrlVhq9wBzwsJs/edit?usp=sharing

E Results - amount of states

E.1 Washing machine plots

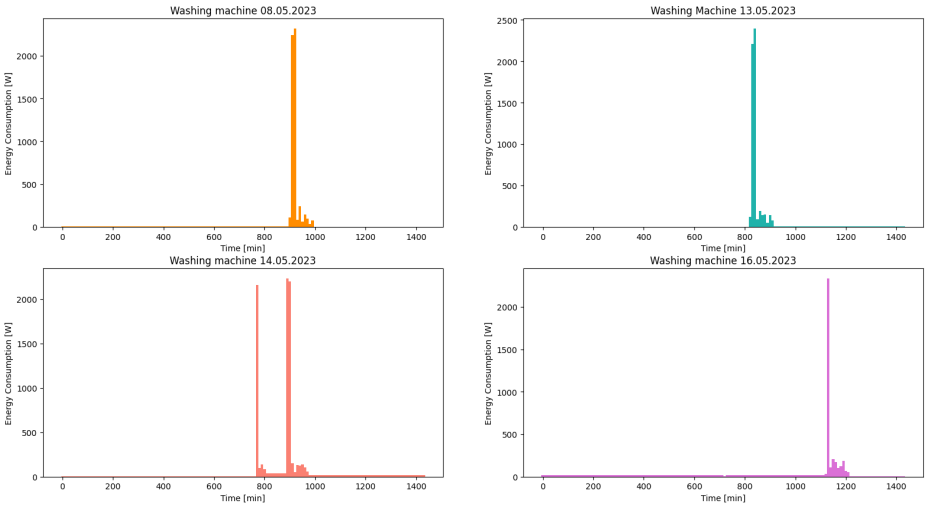


Figure 41: Washing machine runs for testing.

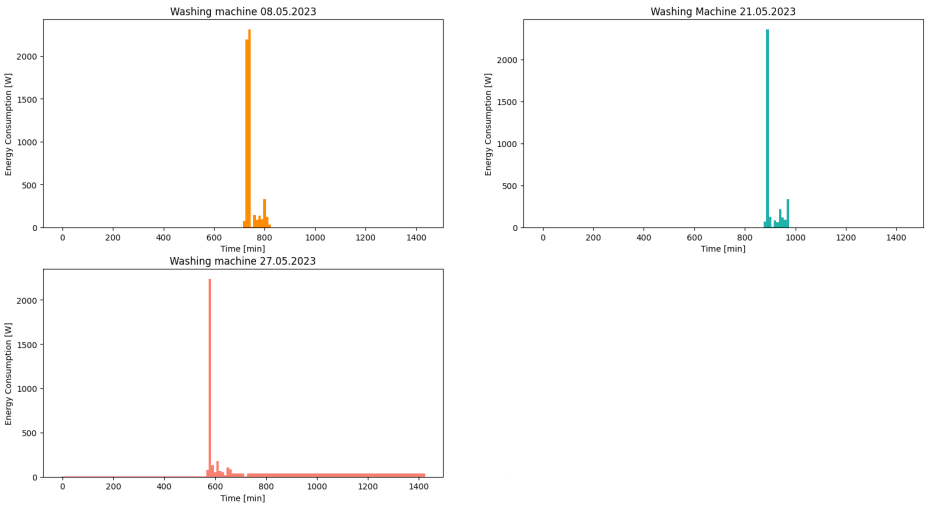


Figure 42: Washing machine runs for testing.

E.2 2 states

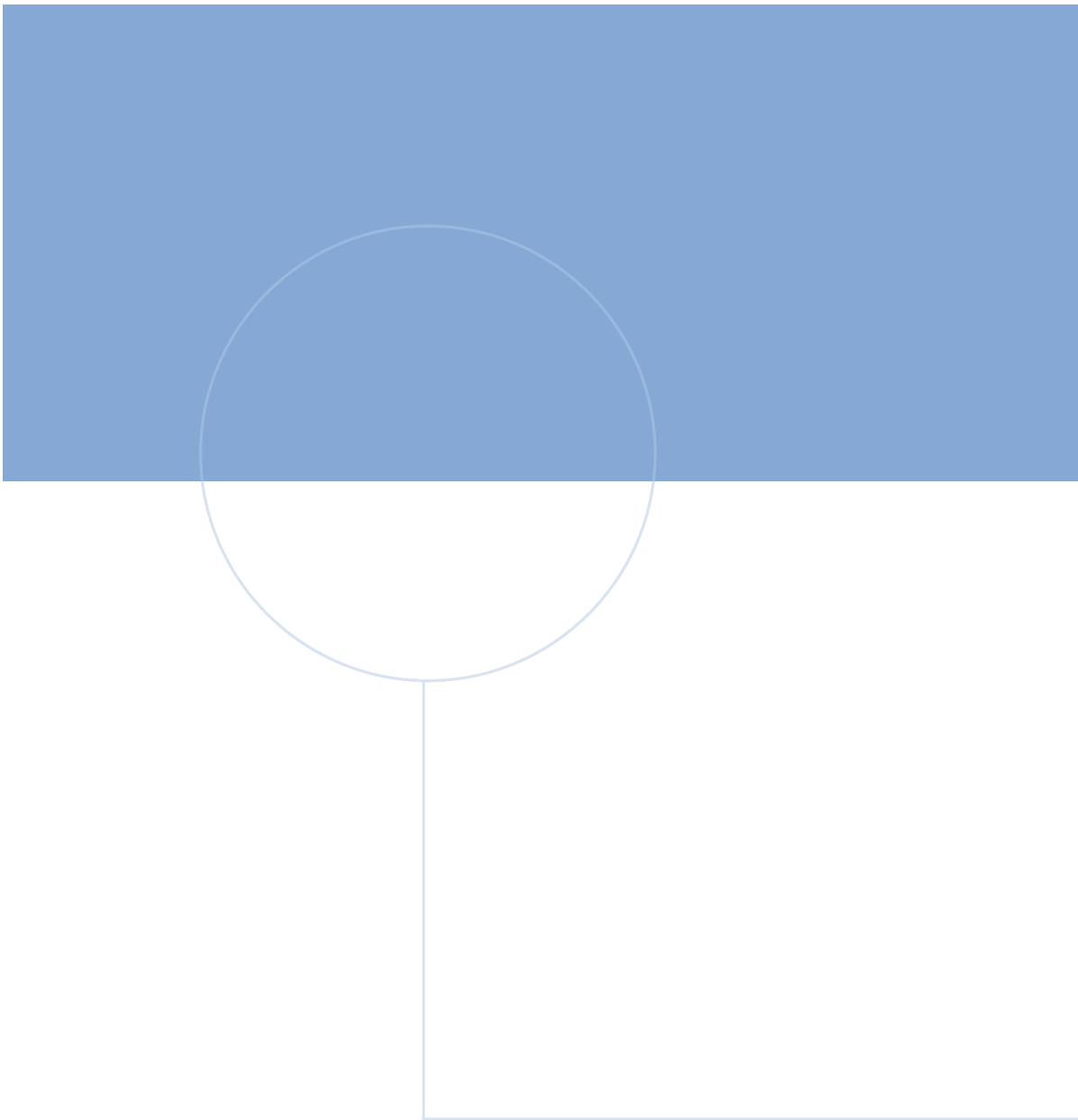
https://docs.google.com/document/d/173yYCN1dElev65LkMO5oyjk6rxM76lJNja3e-0U_vro/edit?usp=sharing

E.3 3 states

https://docs.google.com/document/d/19AdfhMm_WGIh-fz7aFh-wRPIJp75t6TzA5lUsyZ2zA4/edit?usp=sharing

E.4 4 states

https://docs.google.com/document/d/1RmvfxmnpTl9X6hdrjG4CZri74fPck9ydm7ihSVbC3cw/edit?usp=sharing



NTNU

Norwegian University of
Science and Technology