Norwegian University
of Life Sciences

Master's Thesis 2019 30 ECTS
Faculty of Science and Technology

Machine learning for load prediction for a single transformer

Espen Sønneland
Mater of Science in Data Science

Abstract

Statnett is in a process to improve their algorithms for predicting the need for electrical power. In this thesis I tested the possibility of using machine learning to predict the active and reactive load for transformers up to 48 hours ahead in the future, and see how accurate predictions it creates. This was done by using Gradient Boosting Regression, where the model predicts one hour ahead at a time, and then uses the previously predictions to predict even further ahead.

The data used in the model consisted of weather, calendar, and electrical demand data from the time period 2013 – 2018.

The model outperformed the baselines for 1, 24, and 48 hours ahead for predicting active effect. It did not for reactive effect.

The model performs well when the load follows a recurring pattern, but struggles to predict irregularities and sudden magnitude shifts.

Acknowledgements

First of all, thanks to Simon Weizenegger for making this thesis possible, and for being a great guide to Statnett.

Thanks to Hans Ekkehard Plesser for giving me the structure I needed, and for being a great advisor.

And of course, thanks to Yngve and Linda, for listening to my ideas, and for giving me motivation when I needed it the most.

Contents

List of Figures

Chapter 1

Introduction

Statnett have the responsibility of making sure that everyone in Norway have constant access to electrical power at all time [1]. This is a complex job that demands a lot of planning. A big part of this is trying to predict how much power is going to be needed in the future. This prediction is called a forecast.

The forecast is used by Statnett to keep the electric frequency at 50 Hz. It is important to keep the electric frequency constant, because all electronics in Norway is based on a constant frequency of 50 Hz [2]. To keep the electric frequency stable at 50 Hz, the power producing companies are informed of how much they should produce short-term at a given time to match the expected consumption.

However, a long-term forecast is also necessary for the production planning and maintenance. With an accurate long-term forecast, an abrupt increase or decrease in power demand will already be predicted and facilitated towards. Additionally, check-ups and maintenance of electrical equipment can also organised to be executed at appropriate times.

Statnett use algorithms to predict the future power demand. These predictive algorithms are very simple and use few variables, such as very recent power demand data. In other words, the algorithms used today are oversimplified. Although the algorithms implemented by Statnett do a decent job of predicting the power demand, they want to explore whether new technologies could improve the prediction performance. A predictive method they wish to explore is by the use of machine learning.

SAS defines machine learning as [3] "(...) a method of data analysis that automates

analytical model building (...) based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention". In other words, machine learning is useful for discovering patterns in large data sets that is otherwise difficult for humans to find. Therefore, Statnett wants to research the potentiality of using machine learning as a tool for predicting power consumption.

The goal of this thesis is to explore the possibility of using machine learning and historical data to predict the average power consumption. The predictions made are done for each hour for the next 48 hours. In collaboration with Statnett, this hypothesis will be tested using a machine learning model for a single transformer, and compare the model's predictions with the true power consumption to calculate the accuracy of the machine learning model.

Chapter 2

Theory

2.1 Electric load

An electric load refers to any type of electric device where power is consumed [4, p. 127]. This can be anything from a glowing light bulb to a big electric motor. It can also refer to the entire power consumption of an area. As load is a measurement of energy, we have chosen to use MW (Mega Watts) as the unit for resistive loads and MVAr (Mega Volt Ampere reactive) as the unit for inductive/capacitive loads [4, p. 136].

Power is produced at a very high voltage, and is transported for very large distances to cities. The power is transformed to lower voltage when getting close to cities and households. This thesis focuses on the last step of the power before it is transformed to a low voltage.

2.1.1 Transformers

A transformer is an electrical device that is used for transforming the voltage up or down in an AC circuit [4, p. 168]. An ideal transformer does this while transmitting all the power, which means that the transformer lowers the current to raise the voltage and vice versa [see equation 2.1]. P_1 is the power coming into the transformer, and P_2 is the power coming out. I is the current, and V is the voltage.

$$\begin{aligned} P_1 &= P_2 \\ I_1 V_1 &= I_2 V_2 \end{aligned} \tag{2.1}$$

When transmitting power, the goal is to minimize energy loss. This is called resistive heating [4, p. 15] and is calculated as in equation 2.2. As resistance (R) is mostly constant, we want to minimize the current (I). To do this, while keeping the amount of power transmitted constant, we have to raise the voltage (V).

$$\begin{aligned} P &= IV \\ &= I^2 R \end{aligned} \tag{2.2}$$

2.2 Machine learning

Machine learning is a method of using algorithms to learn patterns in data sets. In this thesis the type of machine learning used is called supervised learning.

2.2.1 Supervised learning

The main goal of supervised learning is to make a model that can be used to predict the label/value of unseen data. [5, p. 3] Supervised learning use attributes of a data set, called features, to predict the target variable. The target variable is the attribute of a data set one wants to predict. An example could be to use the height and weight of a person (features) to predict the person's age (target variable).

Before learning an algorithm, one divides a data set into training, validation and test set [5, p. 191]. The training set is used for learning the model. The validation set is used for checking the performance. The test set is for estimating the fit of the model.

If there is a large difference in performance between the training and test sets, this might indicate that the model is overfitting [5, p. 73]. An overfitted model is able to accurately predict a very specific pattern, but does not accurately predict for a general trend.

In this thesis the goal is to predict a continuous value (load). That means a subcategory of supervised learning, called regression, is used.

Measuring regression performance

As in all categories of supervised learning, a measurement of performance is needed to. In this thesis the Root Mean Squared Error (RMSE, see Equation (2.3)) is used.

$$RMSE = \sqrt{\frac{\Sigma(\hat{y} - y)^2}{N}}, \tag{2.3}$$

where $\hat{y}$ is the estimated value, y is the true value, and N is the total number of values.

RMSE is a useful metric for estimating performance when large errors can lead to impractical and expensive consequences. This is because large errors have a greater impact on the score than smaller ones. RMSE also has the same unit as the value (y), making the RMSE estimation easy to interpret.

Additionally, Mean Absolute Percentile Error (MAPE, see Equation (2.4)) is used as a measurement in graphs. This makes it easier to compare the quality of predictions from different magnitudes.

$$MAPE = \frac{1}{N}\Sigma\frac{|\hat{y} - y|}{y} \tag{2.4}$$

2.2.2 Machine learning methods

There are many different methods available for training a model. They all have advantages and disadvantages, and one does rarely know what method is going to perform best on a given assignment. This is often called the "*No free lunch theorem*" [5, p. 12] Hence, I have compared the quality of these different methods:

- Gradient Boosting Regression uses multiple weak models, where the average output of the models becomes the final output of the model. [6]

- Random Forest Regression is an ensemble of decision trees. The output from the decision trees becomes variables in a linear function that produces the final output of the random forest. The weights of the trees are adjusted in the training of the model. [7]
- AdaBoost Regression uses multiple weak models, where a weighted average output of the models becomes the final output of the model. The weights are adjusted during the training of the model. [6]
- Lasso Regression is a form of linear regression, where the weights are optimized by minimizing the squared error of the current model. [8]
- Elastic Net Regression is a compromise between lasso and ridge regression. This means that it is form of a linear regression that is optimized by minimizing a weighted sum of the squared and the absolute error of the current model. [9]
- Ransac Regression starts each training iteration by selecting a random subset of the training data. Then it creates a linear model by using the least mean squared on the current subset. The method does this multiple times, and uses the best scoring model. [10]
- Linear Support Vector Regression maps the data into a higher dimensional features space before doing linear regression. [11]

Chapter 3

Method

3.1 The data set

The data set I was given from Statnett consists of a time series from the period 2013 - 2018 for two transformer stations, each with two transformers. The time series has a resolution of one hour. The data set contains both active and reactive effect for each transformer. I have not received any information of which kinds of areas the transformers are connected to (for example if it is connected to a rural, industry, or farming area).

I have merged the data set from Statnett with weather and calendar data. The resulting data set is what I have used in this thesis.

Each transformer has its own time series, where active/reactive effect are divided into separate time series. They have been given code names where A/B are two different transformer stations, T1/T2 are the two different transformers, and P/Q tells if the time series contains active- or reactive load. For example AT1P and AT2P are both at station 'A' and describes active load.

T1/T2 are connected in parallel. This means that they distribute power to a portion of an area each, for example T1 distributes 40% of an area while T2 distributes 60%. The portion can be changed manually.

I have split the data into training-, validation-, and test data as such:

Training data: January 2013 – November 2016

Validation data: December 2016 – November 2017

Test data: December 2017 – November 2018

These periods have been chosen such that the training data is the largest, and such that all data sets contain all months.

3.1.1 Data sources

The load data for each transformer was loaded from Statnetts internal communication platform; "Innsikt". It was given in the form of an xlsx-file, and then converted and cleaned into a csv-file by using my own Python script as described bellow.

The weather data was collected from *Meteorologisk institutts* weather data service *eKlima*. It was given as an xlsx-file, and then transformed and cleaned into a csv-file by using my own Python script.

3.2 Features

3.2.1 Calendar features

Seasons

The active power changes seasonally over the year (see Figure 3.6). To be able to store the seasonality factor to the machine learning model I made a function that creates a new feature containing the defined season for each time stamp.

As the meteorologic definition of the seasons is not strictly defined [12], I have chosen to use the astronomical definitions of the seasons:

Spring 20.March to 20.June

Summer 21.June to 21.September

Fall 22.September to 20.December

Winter 21.December to 19.March

[13]

Day off

day_off is a binary feature that tells if people have the day off, as people use power differently based on what day of the week it is. A typical week can be seen in Figure 3.1 and Figure 3.2, where the load is largest during the weekdays (Monday to Friday), and lower in the weekend.

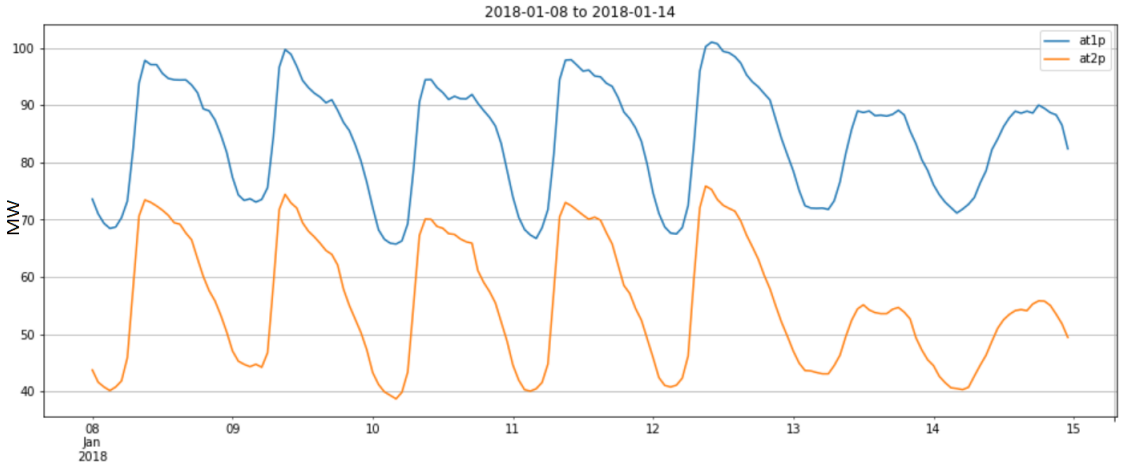


Figure 3.1: A week of active effect from transformer 'A'. The figure shows a weekly pattern (Monday to Sunday).

In Figure 3.1 it looks like workdays (Monday to Friday) have a similar pattern, while Saturday and Sunday have a similar pattern. This gave me the idea that the main factor separating the different types of day was whether people where going to work or not. To test this hypothesis I compared the weekdays to the official Norwegian holidays (collected from https://www.timeanddate.no/kalender/) .

Another idea was that people take a day off when there is a work day squeezed in between two off-days. I made a function that find work-days where both the day before and the day after are off-days. There were few days like this, but compared to work days and off-days it seemed like they where similar in magnitude to off-days.

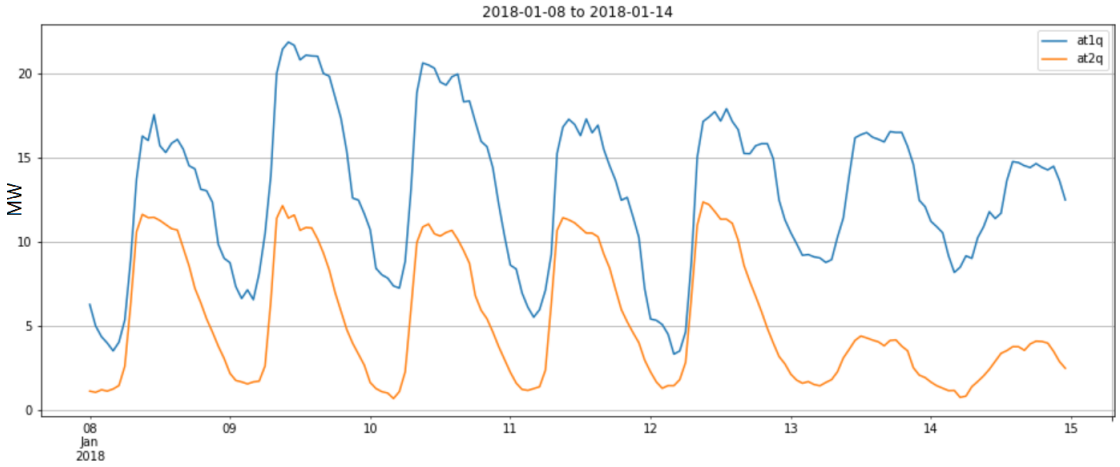


Figure 3.2: A week of reactive effect from transformer 'A'. The figure shows a weekly pattern (Monday to Sunday).

When trying to categorize the day types, I tried clustering by using k-means. I did by collecting data for each day (mean/max/min load, hour when max/min, month, weekday etc), and by having 24 columns containing the load of that hour where each day was a data sample. Both these methods resulted in clustering based on either if mean load was high/low, or what day of the week it was. This did not give any more information than I already had in the data set, and I therefore concluded only to categorize the dates by day_off.

Circular time features

Time cycles are important for describing the load patterns.For example you can see that each day has a similar pattern (see Figure 3.1). To get these patterns as a feature, I have chosen to use a method from [14], which describes circular time features.

Underneath you can see the pseudo-code for getting the month in year as a circular time feature.

$sin\_month = sin(\frac{2*pi*month}{12})$

$cos\_month = cos(\frac{2*pi*month}{12})$

I have chosen three cycles: daily, weekly, yearly. This means I assume that the load is dependent on what time of day it is, which weekday it is, and what month

it is.

Disabled

The transformers are sometimes disabled for different interval lengths. In these intervals the load drops to zero. This can be because of maintenance on the transformer, or other reasons that I don't have the data to explain.

Sometimes the load is close to, but not zero. At some of these timestamps I have gotten conformation from Statnett that the transformer was disconnected. Because of this I have chosen to mark all active loads where $P \leq 5MW$ as disabled.

Parallel disabled

Par_disabled is a bi-nominal feature that tells if the connected transformer is disabled. This is important information, as we are only predicting for one transformer at a time. Without this feature, a sudden disabling of the connected transformer would seem random.

The load on 'AT1P' suddenly jumps when 'AT2P' is disabled as can be seen in Figure 3.3. This happens because the total load on the transformer station remains the same, so the other transformer has to carry the load that previously was shared.

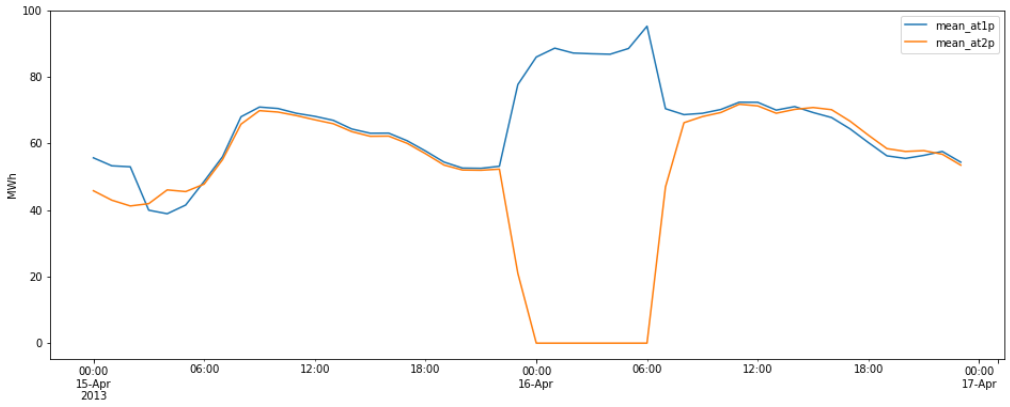


Figure 3.3: Two connected transformers when one gets disconnected

Weather features

The weather can affect the load (low temperature makes people use more heating). I chose to try many weather features, to see if there could be some information hidden in them. Unfortunately some of the features I was looking for did not have data far back. These are the features I ended up using in the data set:

DD wind direction (degrees)

DX_1 wind direction of the most powerful wind the last hour (degrees)

FF wind speed 10 m above ground level (m/s)

FX_1 most powerful wind the last hour (m/s)

PO air pressure (hPa)

TA air temperature ($^{\circ}C$)

All weather data is the recorded weather, as opposed to the predicted weather. One would have to rely on weather predictions when predicting load in the future, therefore using the predicted weather would be preferable for this thesis. Unfortunately, historical weather prediction data was difficult to find, so I have chosen to use the recorded weather.

Lagging features

Lagging features tell what the load was h-hours ago. This means that 'h-1' means the load for 1 hour ago, 'h-12' means the load 12 hours ago, and 'd-1' means 24 hours ago.

These features are created by simply shifting the load by h-hours, such that the timestamp matches the current lag.

In this thesis I chose to use h-1, h-2, h-3, and d-1, and later systematically see if adding more lagging features gives better predictions.

3.3 Cleaning the data

When I converted the raw xlsx-files into csv-files I did some adjustments to the load data:

The first thing I did with the load data was converting the time features (hour, date) into a single time feature(datetime). After that I was now able to fill in missing timestamps with a 'NaN' value, so that it would be easier to interpolate afterwards. Additionally, I chose to change the sign of the load of transformer station 'B', so that both 'A' and 'B' had a positive direction. This is done because active effect is positive per definition.

3.3.1 Missing data

To get a complete time series, I wanted to check for missing data in the data sets. I looked for both missing values, and missing time stamps. In the weather data I found no missing data, but I found some in the load data.

I counted and plotted the missing data. In transformer station 'A' I found that 246 time stamps where not recorded in the load data (See Figure 3.4). There is supposed to be 52 584 time stamps in this period, so the amount of missing data is relatively low (0.47%). For transformer station 'B' there where 212 missing time stamps (See Figure 3.5), which is 0.40% of the expected amount of time stamps.

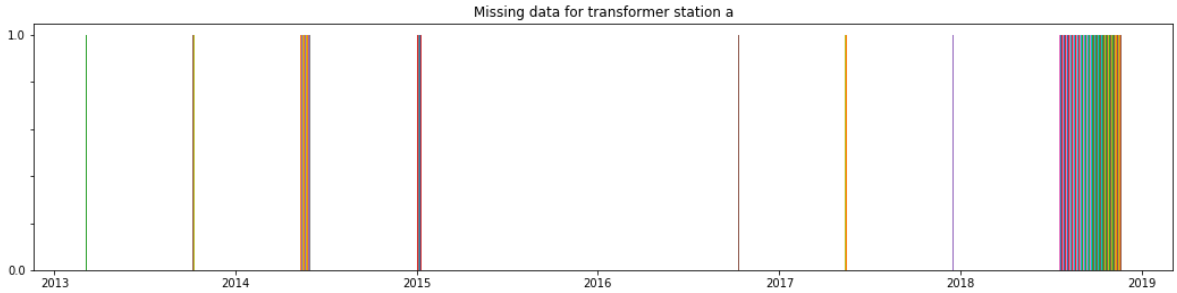


Figure 3.4: All missing time stamps between 2013 and 2018. Each line is a missing time stamp. The colour of each line is to make it easier to separate each timestamp.

By plotting the missing data for transformer station 'A', I saw that most of the missing data (180 / 246) was in December 2018 (See Figure 3.4). December is

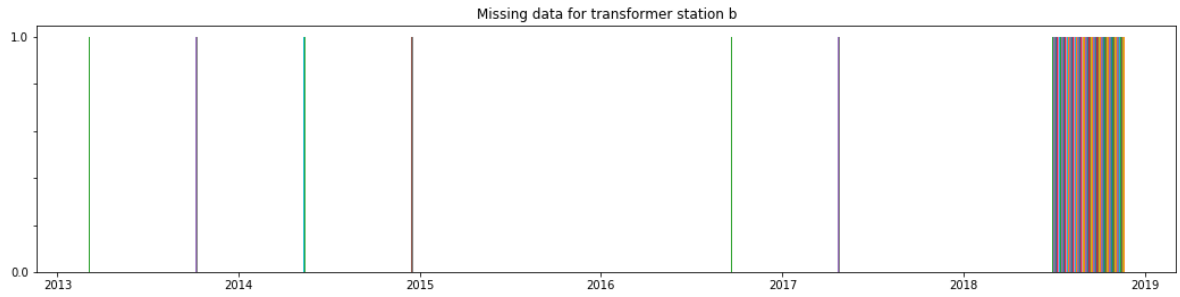


Figure 3.5: All missing time stamps between 2013 and 2018. Each line is a missing time stamp. The colour of each line is to make it easier to separate each timestamp.

supposed to contain 744 time stamps, so this means that December 2018 is missing 24.19% of the time stamps. As December 2018 is the last month of the data set, excluding it does not make any holes in the data, only shortens it. Therefore I chose to exclude December 2018 from the data set for both transformer stations.

There is also a batch of missing data in June 2014 (33) for transformer station 'A'. A lot of these time stamps have a value close to or equal to 0 MW. According to *Statnett* this is because the transformer was disabled during these time periods. Therefore I assumed that all points where $0MW \leq P \leq 5MW$, where P is the mean load, the transformer is disabled. Hence, all these missing values were set to 0, and the time stamp is market by the feature "Disabled" as 1.

Outliers

The load data is periodical, therefore what classifies as extreme value varies annually. The algorithm I created therefore works as a sliding window that uses the standard deviation of the current window to define the limit of what is considered an outlier.

The window is defined for each time stamp as ± 350 hours (approximately a month), and I have chosen to define an outlier within this window as:

Not disabled

Not parallel disabled

Outside the limit: mean load ± 4*Standard Deviation

Disabled and par_disabled time stamps are kept in the data set, as they can be explained. 4*standarad deviation was chosen as I wanted to keep as much data as possible, but still removing the clearly extreme values.

The result of this can be seen in Figure 3.6

When the outliers have been defined, they where removed, and the now missing time stamp is interpolated by using Pandas 'time' interpolation tool.

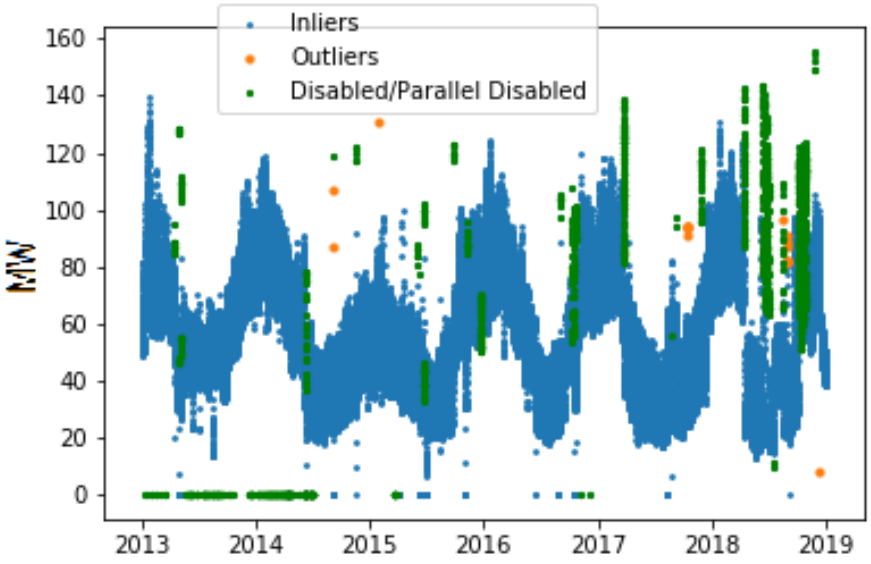


Figure 3.6: Defined inliers, outliers, and disabled/parallel disabled loads for AT1P

Interpolate

Interpolation is a useful tool for filling out missing data. In this case it also is used for replacing outliers, as if the time stamp was missing. This was important as there where some loads that was far higher or lower than the other loads around the time stamp. These values could not just have been removed, as I used lagging features that depended on continuous data.

I created a function for filling missing data as an alternative to interpolation, so that I could have something to compare the quality of the interpolation function

from the Pandas package. The alternative function replaces the missing time stamp with the values from 1 week back. This was done to see if a simple algorithm like this would be better than the Pandas method.

To measure the quality of the interpolation and my own function I used a measurement defined bellow. I tested the methods by selecting a week with no missing values or outliers (6 June – 12 June 2016, transformer AT1P), and then removing one time stamp at a time. For each time stamp removed, we interpolate (with both methods) and record the error compared to the real value. When all time stamps have been interpolated, the RMRSE is calculated. In this example the results where:

Pandas interpolation = 0.025

Replacement function = 0.072

From this we conclude that Pandas interpolation is better than the replacement function.

Root Mean Relative Squared Error = RMRSE

$$\text{RMRSE} = \sqrt{\Sigma(\frac{(\hat{y}-y)}{y})^2} \tag{3.1}$$

3.4 Finding the best model

I have chosen to find an optimal model for transformer AT1P, and assume that these hyper parameters and features will be sufficient for training models for the other transformers.

3.4.1 Predicting 48 hours ahead

The trained model uses recent loads as features (lagging features) for prediction. As we don't have the loads after the current time, we chose to assume that the predicted loads are correct. This means that we predict one hour ahead at a time, and then update the lagging features with the predicted values.

3.4.2 Choosing the best machine learning method

The first step is to find which machine learning methods are most suited for this assignment. To do this I have chosen a handful of regression methods, with their own set of hyper parameters.

To make the process faster, I only test for predicting 1 hour ahead, and assume that this correlates to being able to predict up to 48 hours ahead.

The selection is done one model type at a time, where each model type performs a grid search for the optimal hyper parameters. The model score is based on the RMSE of the validation set. The results from this selection can be seen in results. The best scoring models are used in the next steps.

3.4.3 Feature selection

Choosing the best features is hard, and can be a very computing heavy problem if you want to find the absolute optimal features. To make this process easier I have assumed that some of the features are essential (see list below), and divided the selection process in 2 parts: Lagging features, and weather features.

Calendar features (day_off etc)

Circular time features

Lagging features (h-1, h-2, h-3, d-1)

For feature selection I use the mean RMSE for all hours ahead predicted as a quality measure.

Lagging features

For selecting the number of lagging features I add one feature at a time (one more hour further back (h-4, h-5, ...)), and test the quality of the 48 hour prediction for the validation set for each added feature. Once the extra added features stops improving the quality, I stop adding features, and continue with the currently added features.

Weather features

For selecting the optimal weather features I try all possible combinations of weather, with the essential features and the optimal number of lagging features.

3.4.4 Testing the model

When we have chosen model and features, we predict for all timestamps in the test set. If the quality of the prediction of the validation set is not too different from the quality of the predictions done on the test set, we can say that the model is not over fitted.

3.5 List of used software

- Python 3.7.1
- Numpy 1.15.1
- Pandas 0.23.4
- Matplotlib 2.2.3
- Seaborn 0.9.0
- Scikit-Learn 0.19.2

Chapter 4

Results

4.1 Finding the best model

In this section you will find the results from the different stages for finding the best model.

4.1.1 Choosing machine learning model

RMSE for each model predicting 1 hour ahead	
RandomForestRegression	2.388
GradientBoostRegression	2.301
AdaBoostRegression	6.054
LassoRegression	6.031
ElasticRegression	5.672
RansacRegression	4.205
LinearSVR	5.251

GradientBoostingRegressor and RandomForestRegression gave the lowest/best scores. They were so similar in quality, that I chose to try both methods for predicting 48 hours ahead. In this step, GradientBoostingRegressor outperformed RandomForestRegression, therefore I have only included the results of GradientBoostin-

gRegressor.

Optimal features

Lagging features: h-1, h-2, h-3, h-4, d-1

Weather features: PO, TA, FF

Core features: disabled, par_disabled, day_off, sin_month, cos_month, sin_week, cos_week, sin_hour, cos_hour, weekend

The features mentioned above together with the core features where used in the final 48h prediction model.

The best hyper parameters for the best performing model is listed underneath, as well as the parameter grid I used for finding the optimal hyper parameters.

Optimal hyper parameters for GradientBoostingRegressor

learning_rate: 0.1

loss: 'huber'

max_features: 'auto'

min_samples_split: 4

n_estimators: 500

Hyper parameter grid search for GradientBoostingRegressor

learning_rate: 1, 0.1, 0.01

loss: 'ls', 'lad', 'huber', 'quantile'

max_features: 'auto', 'sqrt', 'log2'

min_samples_split: 2, 4, 8

n_estimators: 100, 200, 500

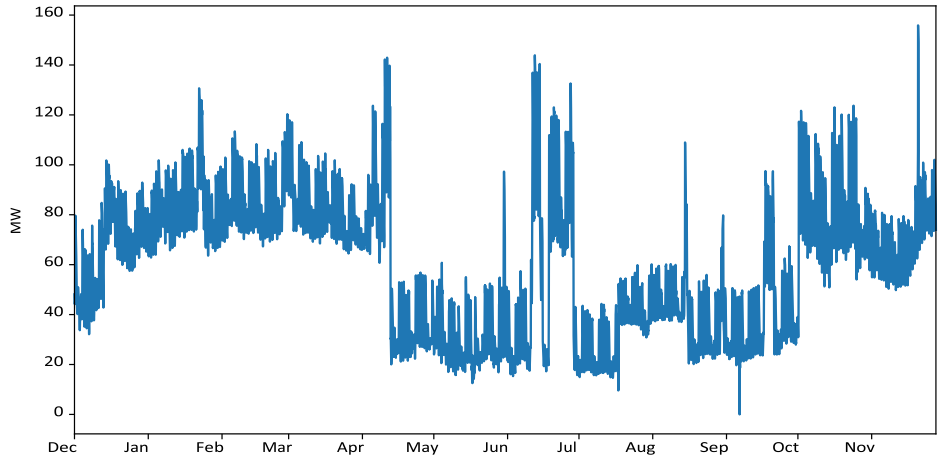


Figure 4.1: Overview of the test data set (December 2017 – Novemeber 2018 for AT1P)

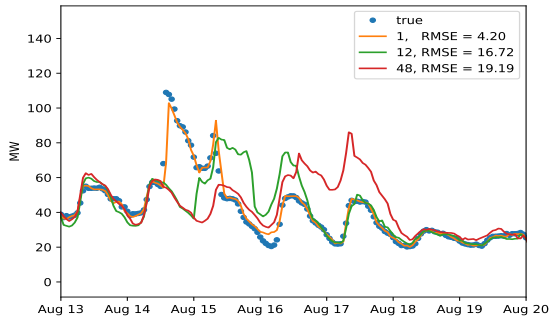
4.1.2 Results for 48 hours predictions for each transformer

The results are based on predicting 48 hours ahead on the test set (see Figure 4.1). The test set has some sudden magnitude shifts because the parallel transformer gets disabled/abled (see the the sudden drop between April and May), while some shifts have other unexplained origins (see the small shift between July and August). There are also some magnitude spikes for which I do not have an explanation (see the two spikes in early September)

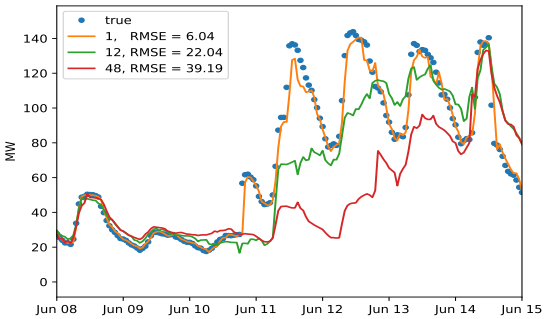
The weeks in Figure 4.2 show examples of predictions for 1, 12, and 48 hours ahead. The example week in Figure 4.2a, Figure 4.2b, Figure 4.2c, are examples of bad predictions, and Figure 4.2d is an example of a good prediction. Looking at the true values, it shows that sudden changes makes it hard to predict far ahead.

From Figure 4.3 we can see that the predictions have similar error distributions. As we try to predict further ahead, the predictions are gradually getting worse. Everyone except for d) is distributed around 0. This means that d) has a bias towards predicting a higher value than the actual value. This bias is increasing the further ahead we try to predict.

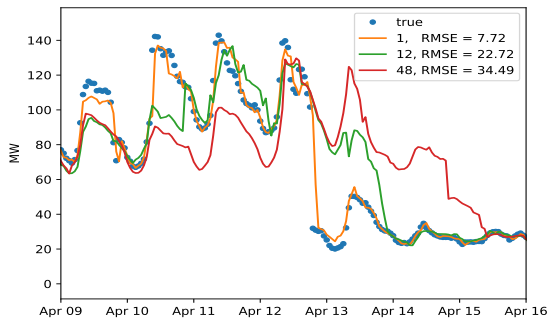
Looking at Figure 4.5 we can see that the quality decreases similarly for all transformers with the number of hours ahead.



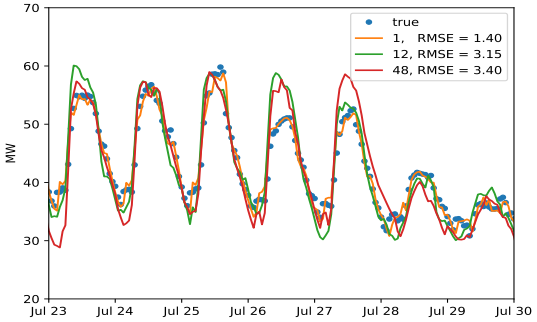
(a) Bad week. Monday - Sunday



(b) Bad week. Friday - Thursday

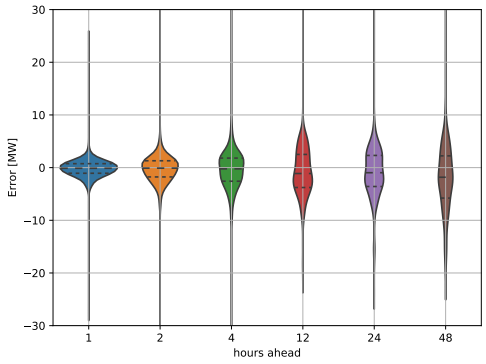


(c) Bad week. Monday - Sunday

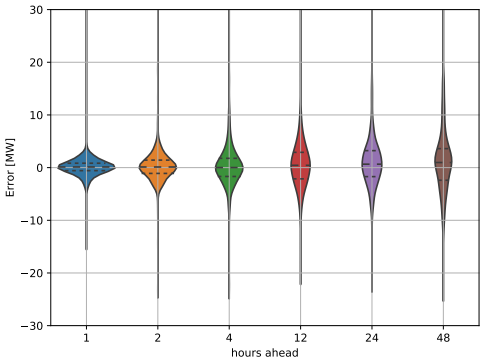


(d) Good week. Monday - Sunday

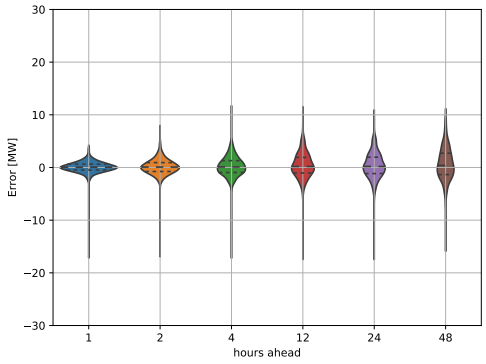
Figure 4.2: Example weeks showing good and bad predictions for AT1P



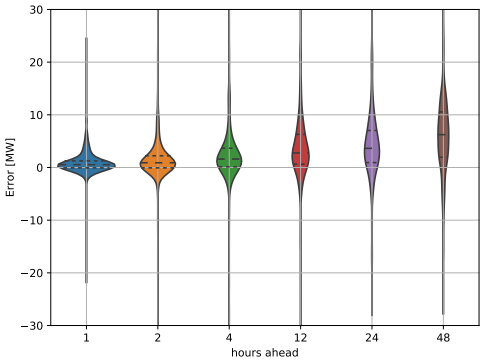
(a) Error distribution for at1p



(b) Error distribution for at2p

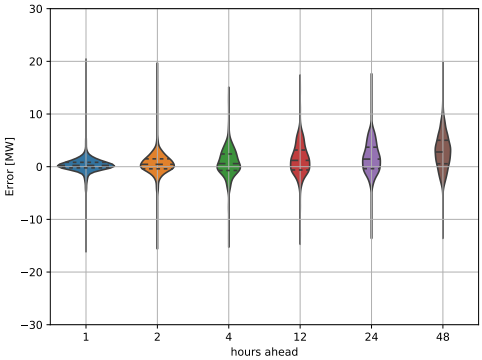


(c) Error distribution for bt1p

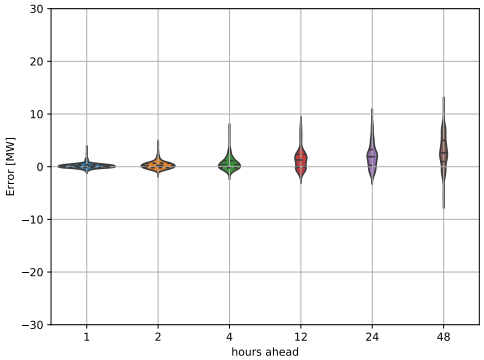


(d) Error distribution for bt2p

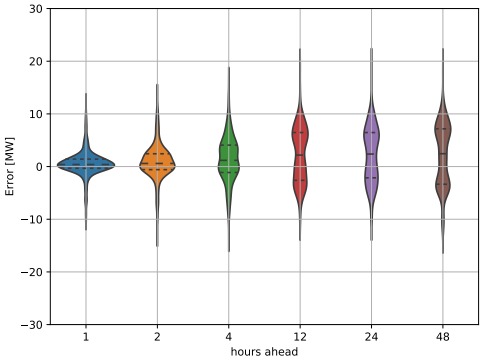
Figure 4.3: These plots (violin plots) show the distribution of errors for active effect for all transformers. A narrow distribution means that the error size is consistent and vice versa. If the distribution is centered around 0, then it indicates that the model has no bias. If it centers above 0, then it has a bias towards predicting too high, and vice versa.



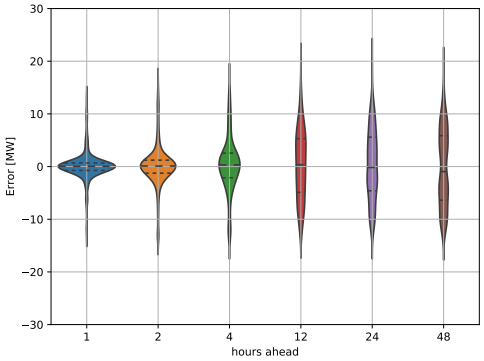
(a) Error distribution for at1q



(b) Error distribution for at2q

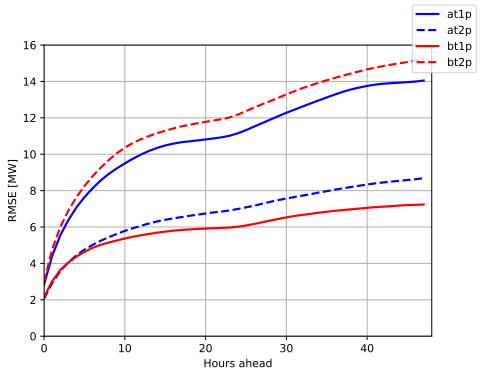


(c) Error distribution for bt1q

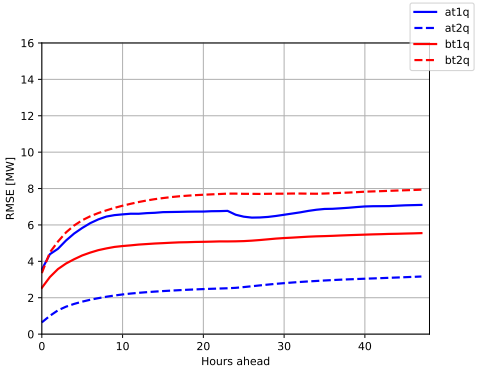


(d) Error distribution for bt2q

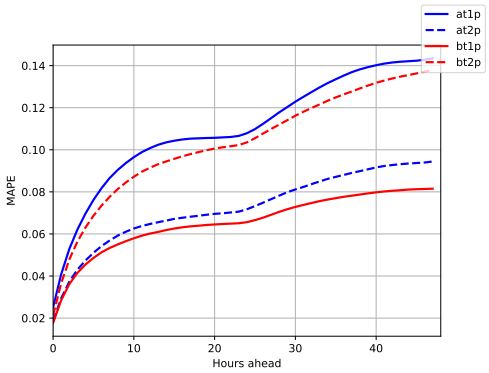
Figure 4.4: Error distribution for reactive effect for all transformers



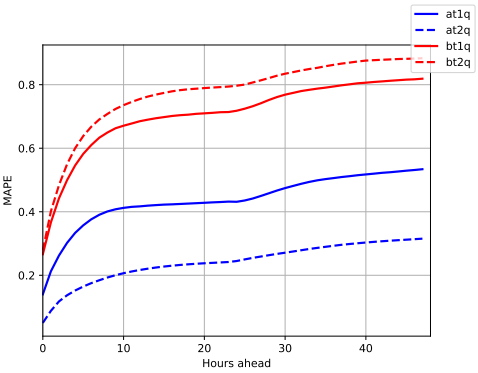
(a) RMSE for all active effect transformers.



(b) RMSE for all reactive effect transformers.



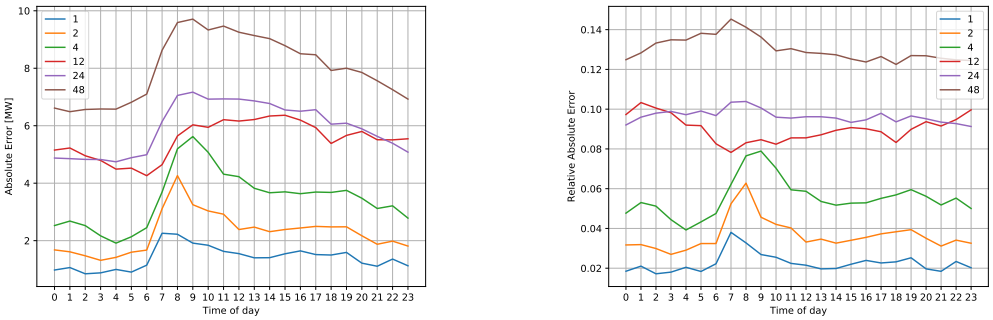
(c) MAPE for all active effect transformers.



(d) MAPE for all reactive effect transformers.

RMSE-baseline: h+1: 4.18, h+24: 13.60, h+48: 19.32
MAPE-baseline: h+1: 0.039, h+24: 0.118, h+48: 0.187

Figure 4.5: Scores for all trafos for all hours ahead predicted.



(a) Absolute error (b) Relative error

Figure 4.6: Error per time of day, for different hours ahead

In Figure 4.5c we can see that the transformers for active effect have similar quality. The transformers for reactive effects in Figure 4.5d are worse. This might be because they have a much lower magnitude.

4.1.3 Analysis of the prediction model for 'AT1P'

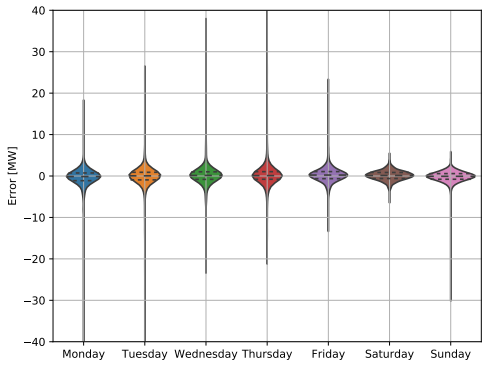
Mean absolute error per time of day

For all hours ahead in Figure 4.6 we can see a peak starting at 06:00 – 08:00. This indicates that the model is at its worst in the morning for all predicted intervals.

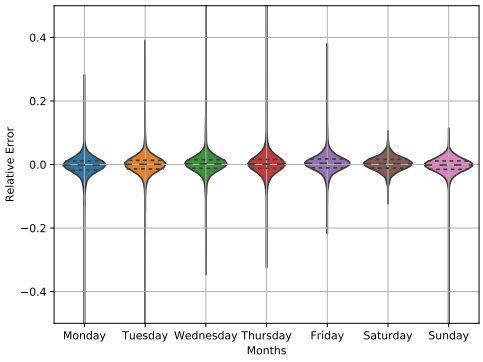
In Figure 4.7 we see that the error distribution gets wider for each hour ahead predicted. For all hours ahead the smallest distribution is in the weekend.

In Figure 4.8 there doesn't seem to be any seasonal pattern for the error distribution. The most noticeable effects seem to be that June, October, and April are the months with the largest distribution.

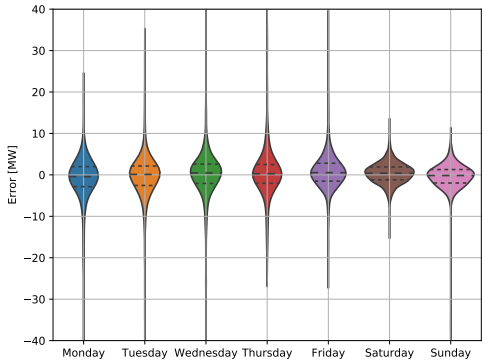
Error distribution per weekday



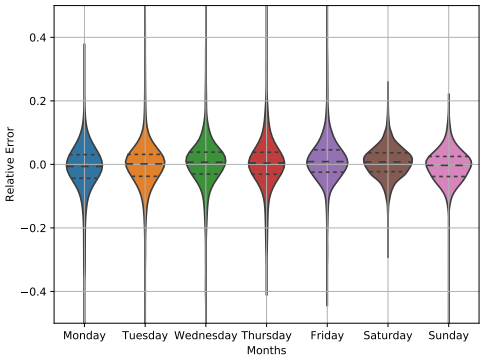
(a) 1 hour ahead



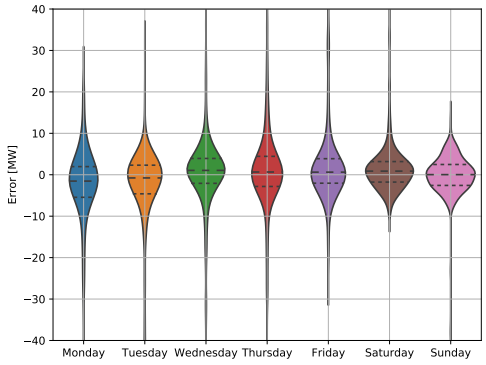
(b) 1 hour ahead, relative



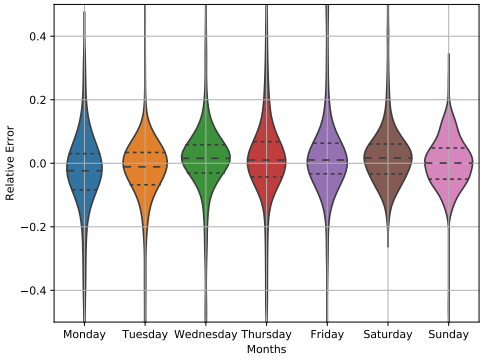
(c) 4 hours ahead



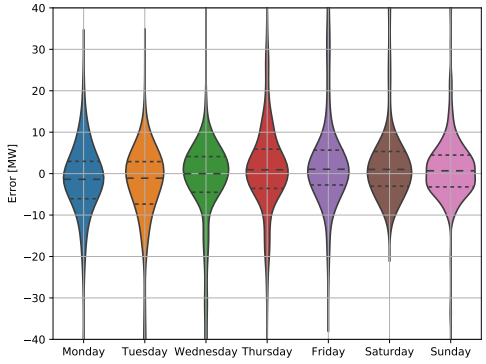
(d) 4 hours ahead, relative



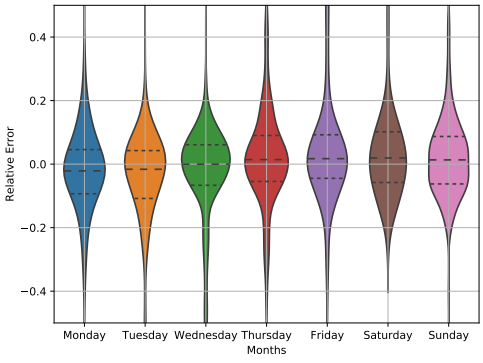
(e) 24 hours ahead



(f) 24 hours ahead, relative



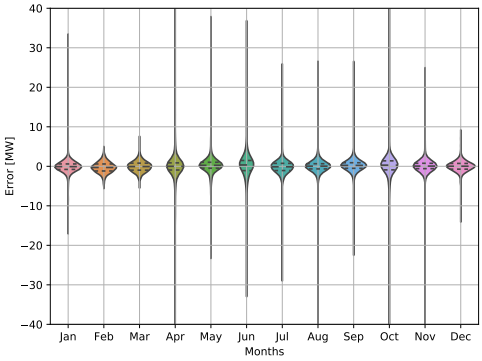
(g) 48 hours ahead



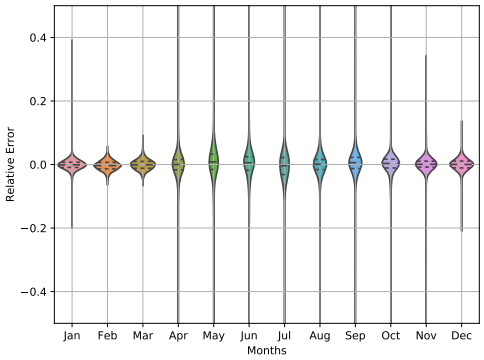
(h) 48 hours ahead, relative

Figure 4.7: Error distribution per weekday per hours ahead

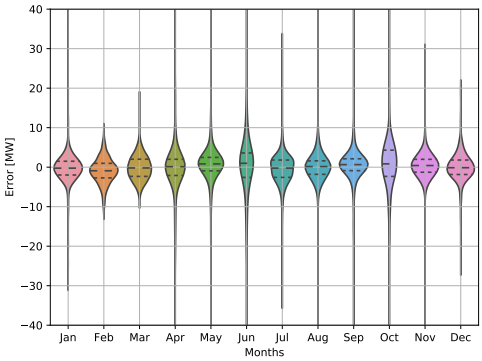
Error distribution per month



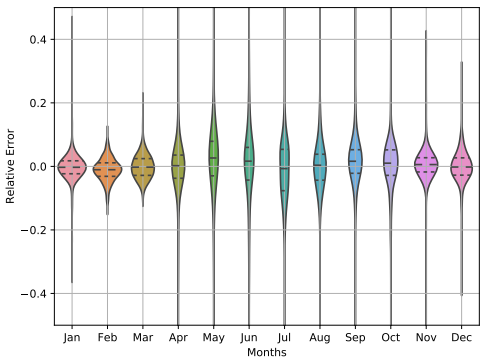
(a) 1 hour ahead



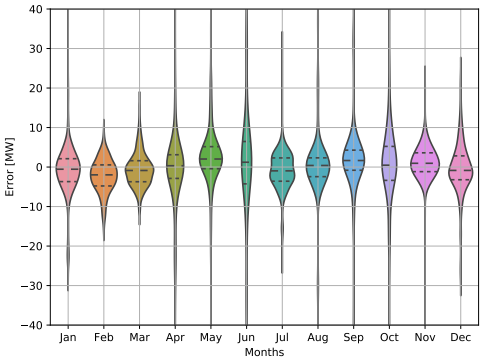
(b) 1 hour ahead, relative



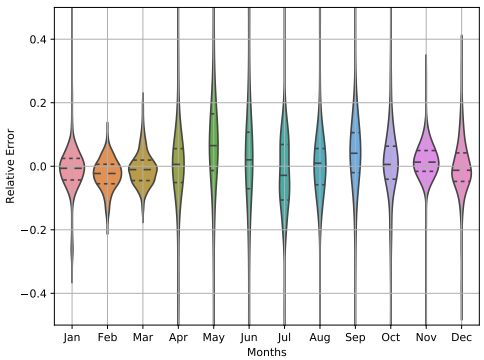
(c) 4 hours ahead



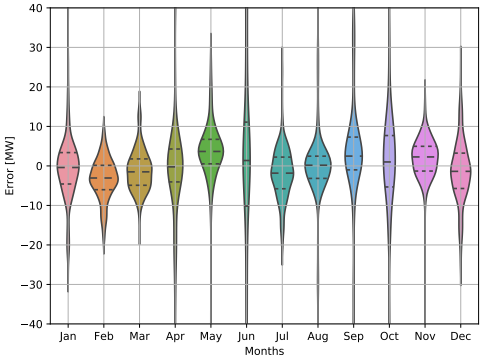
(d) 4 hours ahead, relative



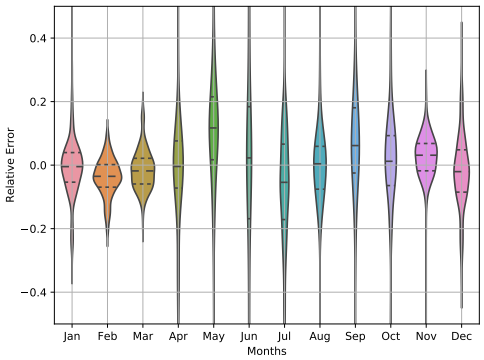
(e) 24 hours ahead



(f) 24 hours ahead, relative



(g) 48 hours ahead



(h) 48 hours ahead, relative

Figure 4.8: Error distribution per month per hours ahead

Chapter 5

Discussion

As seen in Figure 4.5, we can see that all the models for active effect outperformed the baseline for both RMSE and MAPE.

The models for reactive effect outperformed the RMSE baseline, but not the MAPE. As the reactive effect has a much smaller magnitude than the active effect, this shows that the performance for predicting reactive effect was quite bad.

5.1 Discussing the method

The load data was given with a measuring frequency of an hour. This means that there is a whole hour where a change in the load can go unregistered. If the frequency was higher, a lot of these changes could have been picked up on. This is especially the case where there are sudden jumps up or down.

If the historical records of the load is not recorded in a higher frequency, then there is little to be done. But if it is possible to get records with a higher frequency, this could help getting better predictions for areas with sudden changes.

The chosen weather station was a central station from the same area as all the transformers. It was assumed that the one weather station would give sufficient accuracy for all transformers, but it might seem that selecting weather stations that were closer to the areas the transformers provided for would be more accurate. For example could the precise temperature of the areas possibly explain small shifts in the load.

There were weather features that I tried to collect, that were not available. One of these were the amount of direct sunlight. This feature, among others, could possibly help predicting the load. I would therefore assume that more weather features could make the forecast more accurate.

As seen in Figure 4.1 there are sudden shifts, were the magnitude of the load changes unannounced. As seen in the bad weeks in Figure 4.2 these shifts often result in a bad prediction. This is especially true for long term predictions.

If these shifts are explainable, it could be a great help. Then it would be possible to make a feature that told if a shift was happening. This would be similar to the *par_disabled* and *disabled* features, which have proven to work.

A good way of selecting the best features would be to complete a Sequential Backwards Selector (SBS). As I had to use a function for predicting 48 hours after the model was done training for predicting one hour ahead, this would had to be done manually. This could have been done, but then I would have to change the function for updating lagging features, in such a way that it always knew what lagging features where in the model at the time. This is absolutely possible, but I did not have time to create it.

Even though I did not complete an SBS, I am still confident that all the necessary features available were present in the model. The model would probably still benefit from testing removing some features, as they could be more noise than beneficial.

- Optimize for each transformer When cross validating the model to find the optimal hyper parameters, I chose to do this for only one transformer (*at1p*). I assumed that this would be sufficient for at least the other transformers with active effect. As seen in Figure 4.5, the hyper parameters for *at1p* seem to be good for the other active effect models.

As I was asked to focus on predicting active effect, I prioritized in such a way that I did not have time to do the same optimization for reactive effect. This would be done with the same method as in this thesis, but for a transformer for reactive effect.

5.2 Discussing the results

A common factor for the bad predictions is sudden changes, and irregular loads. As seen in the bad weeks in Figure 4.2 the predictions have great errors after sudden shifts, and struggle to predict small irregularities.

The models perform well in time periods without sudden changes (see Figure 4.2d), and where the load follows a smooth curve. The long term predictions for these time periods are also not far off from the short term predictions.

Looking at Figure 4.8, we can see that the distributions becomes wider and more biased the further ahead it is predicted.

In Figure 4.8h some of the months are are much worse than the others, specially May and June. This might be explained be comparing it to Figure 4.1. This time period has a lot of sudden magnitude shifts, and as previously stated, the models generally have a large error around these shifts. It should also be mentioned that May has a lot of holidays, which might cause some greater errors.

The opposite can be said for the months with the most narrow distribution (January, February, March, November, and December); there is few magnitude shifts in these time periods.

It is important to notice that there is only one sample for each month in the test set. This means that any conclusions drawn about a specific month is only relevant for that instance of that month. If given more data, containing more instances of each month, it could be possible to tell if any trends were relevant for all instances of a month.

There does not seem to be any difference in quality for predicting different weekdays. Looking at Figure 4.7, the relative error distribution seem to be equal for all weekdays for all hours ahead predicted. The error distribution is more narrow in the weekend, but this is most likely because the weekends have a lower magnitude. Hence, the conclusion from looking at the relative error distribution is most relevant.

In the absolute error graph in Figure 4.6 the model seem to start having larger errors in the morning (06:00 – 09:00). This is the case for almost all hours ahead predicted.

The relative error graph in Figure 4.6 have the same spike (06:00 – 09:00) for the

short term predictions (1, 2, and 4 hours ahead) as the absolute error graph. The long term predictions seem to be quite equal in quality throughout the day.

The morning spike is most likely caused by the sudden raise in load, as this is the period when most people wake up. This is a reoccurring problem, as the model has difficulties predicting sudden changes.

The model has input that tells the model what time of day it is (*sin/cos hour*). This means that the model should be able to know what is a typical load for that time of day. This could indicate that there is a lot of variance in the morning, or that the model does not weigh these features enough to pick up on the trend.

Code appendix

May 5, 2019

1 Cyclical time features

In [ ]:
```
def add_cyclical_month_column(df):
    df_copy = df.copy()
    months = 12
    df_copy['sin_month'] = np.sin(2*np.pi*df_copy.index.month / months)
    df_copy['cos_month'] = np.cos(2*np.pi*df_copy.index.month / months)

    df_copy = df_copy.drop('season', axis=1)

    return df_copy

def add_cyclical_week_column(df):
    df_copy = df.copy()
    days = 7
    df_copy['sin_week'] = np.sin(2*np.pi*df_copy.index.weekday / days)
    df_copy['cos_week'] = np.cos(2*np.pi*df_copy.index.weekday / days)

    return df_copy

def add_cyclical_hour_columns(df):
    df_copy = df.copy()
    hours = 24
    df_copy['sin_hour'] = np.sin(2*np.pi*df_copy.index.hour / hours)
    df_copy['cos_hour'] = np.cos(2*np.pi*df_copy.index.hour / hours)

    return df_copy

def add_weekend_column(df):
    df_copy = df.copy()
    weekday = pd.Series(df_copy.index.weekday.values)
    df_copy['weekend'] = np.where(weekday < 4, 0, 1)

    return df_copy
```

2 Predict 48 hours ahead

```
In [ ]: def _update_lagged_features(x, y, h, n_lagg):
            """
            Return the updated lagging features for the current hour(h)
            """
            for n in range(1,n_lagg):
                x['h-'+str(n)][h] = y[h - datetime.timedelta(hours=n)]

            x['d-1'][h] = y[h - datetime.timedelta(days=1)]
            return x.loc[h]

        def pred_48h(model, x_whole, y_true, start, n_lagg):
            """
            Make a 48h prediction starting with start-date.
            The function predicts one hour a head at a time,
            where the lagging functions are updated for each prediction,
            such that one can asume the prediction has no information
            after the start timestamp.
            """
            x = x_whole.copy()
            y = y_true.copy()

            end = start + datetime.timedelta(hours=48)

            preds = []
            dates = pd.date_range(start, end, freq='H')
            for h in dates:
                # update lagging functions

                x.loc[h] = _update_lagged_features(x, y, h, n_lagg)
                data = np.array(x.loc[h])
                data = np.expand_dims(data, 0)
                pred = model.predict(data)
                y[h] = pred
                preds.append(pred)

            preds = pd.Series(preds, index=dates).astype(float)

            return preds
```

2.1 Get quality of 48 hour predictions

```
In [ ]: def get_48h_quality(model, x_whole, y_true, start, end, n_lagg=4):
            """
            Runs 48h predictions from start to end
            Returns array with mean score for each hour ahead predicted
            """
```

```
scores = []
dates = pd.date_range(start, end, freq='H')

for h in dates:
    forecast = pred_48h(model, x_whole, y_true, h, n_lagg)

    y_period = y_true[h: h+datetime.timedelta(hours=48)]

    score = y_period - forecast
    scores.append(score)

scores = np.array(scores)

return scores
```

Bibliography

[1] Statnett, *Fremtiden er elektrisk*, Accessed: 2019-05-11. [Online]. Available: https://www.statnett.no/.

[2] Store Norske Leksikon, *Vekselstrøm*, Accessed: 2019-07-09. [Online]. Available: https://snl.no/vekselstr%C3%B8m.

[3] SAS, *Machine learning*, Accessed: 2019-07-09. [Online]. Available: https://www.sas.com/en_us/insights/analytics/machine-learning.html.

[4] A. von Meier, *Eletric Power Systems*. John Wiley & Sons, 2006, ISBN: 978-0-471-17859-0.

[5] V. M. Sebastian Raschka, *Python Machine Learning*. Packt, 2017, ISBN: 978-1-78712-593-3.

[6] T. Hastie, R. Tibshirani and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction*, ser. Springer Series in Statistics. New York: Springer, 2009.

[7] V. Svetnik, A. Liaw, C. Tong, J. C. Culberson, R. P. Sheridan and B. P. Feuston, 'Random forest: A classification and regression tool for compound classification and qsar modeling', *Journal of chemical information and computer sciences*, vol. 43, no. 6, pp. 1947–1958, 2003.

[8] T. Hesterberg, N. H. Choi, L. Meier, C. Fraley *et al.*, 'Least angle and 1 penalized regression: A review', *Statistics Surveys*, vol. 2, pp. 61–93, 2008.

[9] P. Waldmann, G. Mészáros, B. Gredler, C. Fuerst and J. Sölkner, 'Evaluation of the lasso and the elastic net in genome-wide association studies', *Frontiers in genetics*, vol. 4, p. 270, 2013.

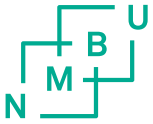
[10] M. T. El-Melegy, 'Model-wise and point-wise random sample consensus for robust regression and outlier detection', *Neural Networks*, vol. 59, pp. 23–35, 2014.

[11] K.-R. Müller, A. J. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen and V. Vapnik, 'Predicting time series with support vector machines', in *International Conference on Artificial Neural Networks*, Springer, 1997, pp. 999–1004.

[12] Petter Dannevig, *Årstider - klima*, Accessed: 2019-01-14. [Online]. Available: https://snl.no/%5C%C3%5C%A5rstider_-_klima.

[13] Kaare Aksnes, *Årstider*, Accessed: 2019-01-14. [Online]. Available: https://snl.no/%C3%A5rstider.

[14] Ian London, *Encoding cyclical continuous features - 24-hour time*, Accessed: 2019-04-24. [Online]. Available: https://ianlondon.github.io/blog/encoding-cyclical-features-24hour-time/.



Norges miljø- og biovitenskapelige universitet
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway