UiA University of Agder

Master

Predicting Electrical Power Consumption on Yearly Events for Substations: Algorithm Design and Performance Evaluations

ADRIAN LANGEMYR

SUPERVISORS
Lei Jiao
Frank Reichert
Morten Goodwin

University of Agder, 2020
Faculty of Engineering and Science
Department of ICT

Abstract

Accurate prediction of electricity usage is critical for grid companies in order to ensure reliable power supply for their customers. Many factors influence usage patterns, but generally they consist of yearly-, weekly- and daily trends in addition to stochastic noise due to random user behaviour. Besides the above-mentioned cyclic trends, certain yearly events, i.e. events that take place once per year, can affect usage patterns significantly and thus may cause abnormally high or -low power consumption. Therefore, it is in the interest of grid companies to predict the consumption on such events so they can take measures in advance, if necessary. Much effort has been put into developing methods of improving forecasting accuracy through the use of time series clustering in conjunction with the actual prediction algorithm, but the methods' ability to specifically improve the prediction of power consumption on yearly events has not yet been evaluated. In this thesis, we are going to utilize machine learning algorithms to cluster electricity usage patterns and predict power consumption for yearly events based on real operational data at the substation level. More specifically, groups of similar usage profiles are formed by a clustering algorithm, and rather than training a prediction model on a single time series, a similar series from the same cluster is appended. In order to extend the prediction model's training set in this manner, the appended time series is transformed to fit the scale of the initial time series. Our experiments reveal that combining similar time series, thereby introducing additional yearly events to the prediction model's training set, can improve the accuracy of the load forecast on the event. This approach is also capable of compensating for missing events in the initial time series, when present in the appended-, similarly behaving time series.

Keywords: Short-term load forecasting; Smart-metering; Time series clustering; LSTM; XGBoost; K-Means; Hierarchical clustering

Acknowledgements

I wish to thank my supervisors from the University of Agder, Lei Jiao and Frank Reichert, for their invaluable feedback during the writing of this thesis. Their academical experience in both writing and supervision has helped me both present and formulate the problem of this thesis and my proposed solution in a structured and well organized way. I am also grateful for Morten Goodwin's contributions to improving technical aspects of the experiments.

I would also give thanks to Agder Energi Nett for providing the data necessary to conduct the experiments which form the basis of this thesis, and Per-Oddvar Osland and Tobias Haumann for feedback, as well as aiding in formulating the problem statement.

I also greatly appreciate Jahn Thomas Fidje and Nils Jakob Johannesen taking the time to discuss relevant research on the topic of prediction of time series and time series analysis in early stages of the thesis.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

1.1.1 Power grid and smart-metering

Electricity is delivered to consumers through a power grid which is structured into multiple layers. In Norway, these are: Transmission-, Regional- and Distribution grid. The voltage is higher in the two first layers, partially in order to minimize power loss as the electricity is transported across vast areas through the grid. The Distribution grid itself is divided into two parts: high voltage, which holds 11kV or 22kV, and low voltage, which holds 230V or 400V. Most households and small industries are connected to the low voltage Distribution grid. Substations transform the electricity from high- to low voltage in this final layer of the grid, and connect directly to end-users. These vital parts of the power grid are required to withstand the combined usage of all connected consumers at any given time.

A major challenge facing grid companies in this regard, is the fact that many users display a similar pattern of power consumption, or *usage profile*. When many end-users connected to the same substation share a similar usage profile, their peak demand in the course of a day may align, sometimes resulting in a short period where the substation is required to handle a very high demand.

In recent years, efforts have been made to aid in the reduction of this aligned peak load, made possible with the introduction of smart-meters. These digital power meters measure usage autonomously, and data is regularly sent to power grid companies. Previously, readings had to be done manually, resulting in time series that were sampled monthly, with sometimes irregular intervals. When the reporting of power consumption is performed digitally, the transmitted packets can contain data with a much higher resolution that is also more precise, as well as regularly spaced.

This detailed information about the consumers' usage profiles can also be made available in real time for customers through a Home Area Network (HAN) port. By accessing the usage data in real time, end-users are able to utilize third party solutions that will regulate various equipment like water heaters and heating cables automatically to even out the usage profile when demand is high. This is especially interesting to customers producing their own electricity with, for example solar panels, as they will also be able to

use such solutions to heat up their water additionally when production is high and usage low. By doing so, the heater can be utilized as a buffer that will allow the end-user to maximize the energy saving potential of the installed solar panels. Another way of using this data is through *learning thermostats*, which will tailor its behavior to align with the requirements of the user in an optimal way. This is done by allowing an algorithm to control a thermostat autonomously so that, for example heating cables, are warm in the morning, turned off until the afternoon in weekdays, and by the time the end-user is home from work, they have been brought back to the desired temperature. While this solution does not address the peak load phenomenon directly, it shows the usefulness of learning algorithms in problems related to the efficient reduction of electricity usage, given the data from smart-meters.

The introduction of smart-meters also leads to an increasing insight into the patterns of power consumption on higher levels of the power grid. Substations have their own power meters installed that measure the combined usage of all connected end-users, and transmit regularly to the grid companies. Those in possession of such data will then be able to better understand the needs of the consumers, and the load requirements for components in the power grid at different times during the day, week, month and year. This deeper understanding of the pattern of power consumption makes it possible to more precisely scale the components within the power grid to meet the demand. It is also important for making informed decisions when selling and purchasing electricity. This can be done in several ways, for example through statistical analysis.

1.1.2 Load forecasting

Forecasting is also a valuable tool for predicting future demand. Grid companies are quickly building up large databases containing history of usage values, and can greatly benefit from using this data to predict future demand for electricity. One reason why accurate forecasting is important relates to the strict requirements to reliably deliver high quality power. Load forecasts can allow the grid company to take measures in advance of peak loads by detecting them before they occur, and thus ensure reliable power supply for the connected end-users. If a substation goes through a period of excessive load, its expected lifetime is reduced. This could lead to more frequent maintenance work.

In order to select a suitable prediction algorithm, and analyze the problem in an efficient manner, it is important to define the scope of the forecast. Load forecasting is typically divided into three main categories, based on how far into the future one aims to predict: short-term, medium-term and long-term. These categories refer to load forecasts over a period of one hour to a week, from one week to a year, and more than one year respectively [1]. Sometimes very-short-term load forecasting is added to these categories, which refers to predictions made for the next hour or less than a day [2][3]. Each of the types of forecasts provide valuable information for different applications. A long-term forecast might be useful when predicting the load requirements of components in the power grid like substations. However, as it is not plausible to make such predictions with the accuracy required for dispatchers responsible for maintaining the grid's security and reliability, short-term forecasts are mostly used for this purpose [3]. One reason for this is that it is difficult to acquire accurate exogenous factors like weather data far into the future, and one would have to rely on historical climate data to estimate the future values. In the case of a grid company taking measures in advance of periods of excessive loads on substations, a good short-term forecast would provide the accuracy needed, as well as give them the time to do so.

For short-term load forecasting, statistical prediction models along with Machine Learning (ML) algorithms are widely used, and can be used to train a model to recognize patterns and correlations with factors that contribute to change in power consumption, or solely on the historical load values. The latter application would be the prediction of what is known as a univariate time series, meaning no exogenous factors are added to the training set. Temperature is one such factor which typically has a strong negative correlation with power consumption caused by the increase in usage from heating when the temperature is low. When using exogenous time series like temperature in order to make predictions, it will naturally restrict how many days in advance a predictions be made for any given day to the same as the weather forecast. This should be taken into consideration when training a prediction model, as it contributes to the trade off between prediction accuracy and how far ahead one can predict.

In addtion to weather, there are other factors that can be drawn out from the time stamp of each value in the series like hour of the day, day of week and month of year from which an ML model can learn much about customers' usage profiles. Learning daily and weekly patterns (given that such patterns

exist) requires relatively short history of values, as the model will see 52 examples of weekly patterns, and 365 daily patterns in the course of a single year. These factors, or ML features, will reveal the cyclic trends common in such time series like repeating daily and weekly patterns. Yearly events, on the other hand, require an entire year for the model to train on a single example of a given occurrence. If the training set contains one example where an event took place, and the temperature was particularly high or low causing unusually low power consumption, perhaps because of limited attendance. Then the predictions for next year is likely to be too low, as it has not learned the correlation between the event and weather conditions through observing that single example. More generally, the model might not learn the relation between the event and the effect it has on power consumption at all. This could cause the grid company to fail to predict a period of excessive load for a substation related to such an event.

1.1.3 Usage profile clustering

With the improved data quality from smart-meters, it is also possible to detect similarities in usage profiles. This can be used to improve prediction accuracy, as described later in the thesis. This classification, or *clustering*, is useful in order to for example better understand the composition of the customers under substations. Because it is not simply the number of residential-, industrial- or any other type of customer connected to a substation that determines whether the substation's usage profile as a whole resembles any of these profiles, it is of value to be able to categorize these aggregated time series. While it can be easy to distinguish usage profiles and detect similarities when manually observing the time series, it is a much more difficult process to automate which is necessary in order to do this on a large scale with thousands, or even hundreds of thousands of time series. This additional knowledge about the usage profile measured at the substation level could potentially be useful for prediction algorithms, for adjusting the algorithms according to the substation for which it makes predictions.

Depending on the goal of the classification, there are several ways one can go about clustering time series: whole time series-, subsequence- and time point clustering [4][5]. Whole time series clustering forms groups based on some metric calculated for each time series in its entirety. This includes a feature based approach, where for example statistical properties are gathered from each time series and put into a vector, and similarity is determined by

comparing this vector of features. Another variant of whole time series clustering is shape based. Like the feature based approach, it often utilizes conventional clustering algorithms, but the similarity is measured based on the actual shape of the time series. Non linear manipulation of the time axis is also applied to be able to detect similarities even when the patterns are not perfectly aligned. Subsequence clustering is when a single time series is split into equally sized sections, and a clustering algorithm is applied to this set of sequences. This could be used to detect patterns within a time series and categorize for example daily usage profiles. Time-point clustering creates groups of similar time points rather than sequences.

1.2 Problem statement

For a power grid company, it is necessary to accurately predict the electricity usage of its customers in order to provide high quality power reliably. The recent introduction of smart-meters provides modern algorithms with the information needed to perform more precise predictions of power consumption. The usage patterns are generally speaking largely made up of yearly-, weekly- and daily cyclic trends in addition to stochastic noise. However, some yearly events, like for example Christmas, affect consumers differently resulting in unusually high or -low usage. It is in the interest of power grid companies to predict these abnormalities as they might want to take measures in advance of the event. If such events are missing, or sparse in a time series, they will be more difficult to predict, and a prediction model will likely be less accurate. Power consumption is measured at multiple levels of the power grid, and the combined usage of end-users measured at the substation level is easier to predict than for each individual customer. It is also a critical point in the power grid where overloads can occur as a result of aligned peak loads related to yearly events.

1.3 Thesis definition

This thesis will attempt to compensate for sparsity in-, or lacking yearly events in the training sets of ML prediction models. This will be achieved through forming groups of similar time series in such a manner that an ML model can learn across them to better understand the impact of a yearly

event on power consumption, and thus improve its accuracy when predicting power consumption during the event.

1.3.1 Thesis goals

Goal 1:
Improve prediction of electricity consumption on yearly events at the substation level.

Goal 2:
Develop a method to create a single training set from multiple similar time series originating from different substations.

1.3.2 Hypotheses

Hypothesis 1:
A prediction of power consumption at the substation level around yearly events can be improved by training an ML model on additional data from other, similarly behaving time series.

Hypothesis 2:
Time series related to power consumption on the substation level can be clustered into similarly behaving groups.

Hypothesis 3:
Substations with generally similar power consumption patterns also share the same behavior in periods with a yearly event.

1.3.3 Summary

Electricity is delivered to end-users through a multi-layered power grid. Substations are where often multiple end-users are connected, meaning that these essential components of the grid have to endure the combined usage of all connected users. Certain events take place only once per year, that cause many end-users to alter their usage patterns in a similar way. This can cause aligned spikes in power consumption that result in short periods

where the substation must handle a much higher load than outside of such events. The recent introduction of smart-meters makes it possible to deploy sophisticated ML prediction models to make accurate load forecasts that allow the grid companies to take measures in advance of such spikes. However, these yearly events can be difficult to predict, as they are sparse in the time series. This thesis will address this problem through the combined use of clustering- and prediction algorithms.

1.4 Thesis structure

The remainder of the thesis is organized as follows: In Chapter 2, previous work related to load forecasting and clustering of usage profiles is presented. Chapter 3 describes the proposed solution of this thesis which will address the problem that has already been presented. Chapter 4 explains what- and how experiments were conducted, and discusses the results. Chapter 5 evaluates the results, and whether the thesis goals are achieved or not, before describing what can be done as further research on the topic of this thesis. Appendix contains graphs showing aggregated results from the different prediction experiments as well as examples of single predictions.

Chapter 2

Related work

This chapter will present work that has been done previously, leading up to the state-of-the-art for the topic of this thesis. The chapter is structured as follows: Section 2.1 will briefly describe the development from early statistical prediction models to more sophisticated ML algorithms. In Section 2.2, different applications for clustering algorithms will be described as they relate to time series clustering. Section 2.3 will present the state-of-the-art by evaluating various implementations of the combined use of clustering- and prediction algorithms.

2.1 Prediction algorithms

While accurate load forecasting is a difficult task, as power consumption to some degree is stochastic in nature, much effort has been put into tackling this issue. Statistical approaches, such as Auto-Regressive Integrated Moving Average (ARIMA) [6] have for a long time been very popular. While originally being used for univariate time series, it has been used in conjunction with other models that handle the effect of temperature on usage, as seen in [7].

Another approach is to introduce temperature as an explanatory variable, as seen in [8]. What is meant by this, is to introduce an additional correlated but independent time series to the model. This method can be seen in later iterations of such models, like Seasonal Auto-Regressive Integrated Moving Average with eXogenous factor (SARIMAX), which can utilize additional time series, like temperature, to improve predictions [9]. Another improvement is that SARIMAX also handles some seasonality in the time series, where as ARIMA requires the time series to be made stationary. This means removing any trend and cyclic variations in the time series before training the model, and then reintroducing it into the predicted values. A potential drawback of statistical models for predicting power consumption is the need for statistical analysis, and an in depth understanding of the time series at hand. This makes the process difficult to perform autonomously on a large scale, as the various time series may differ significantly in terms of statistical properties.

In more recent years, the use of ML has become quite popular, as it can learn the relationship between factors like weather data, holidays, day of week, hour of the day and power consumption at a given point in time.

All of these factors combined will, as a general rule, be good indicators on which predictions can be based. Among the most popular types of algorithms are neural networks like Recurrent Neural Networks (RNNs) [10], Convolutional Neural Networks (CNNs) [11] and Artificial Neural Networks (ANNs) [12]. Long Short-Term Memory (LSTM) belongs to the category of RNNs, and has become a popular algorithm for short-term load forecasting due to its ability to learn both short- and long term dependencies in time series, given enough data [10]. It has also been shown to outperform more traditional statistical prediction methods. ANN has been compared to statistical approaches when performing short-term load forecasting, like SARIMAX in [12], showing that for both one day ahead, and weekly predictions, SARIMAX outperformed the NN in the actual forecasting stage of the experiments. Another popular ML algorithm is eXtreme Gradient Boosting (XGBoost), which is a type of gradient boosted decision tree. It has had much success in ML competitions on sites such as Kaggle where one year 17 of the 29 winning solutions implemented XGBoost [13]. It can be used for both classification and regression, and is a more transparent algorithm than NNs. What is meant by this is that the trained models can be analyzed more easily, even down to the level of looking at the splits in each tree made during the training phase. It has also shown to perform better than more traditional algorithms like the aforementioned LSTM [14] for short-term load forecasting, both with the regular- and with the proposed implementation of the XGBoost regressor.

2.2 Clustering algorithms

While much has been done to improve the prediction accuracy of electrical power consumption, the process of forming groups of similar usage profiles has also been of interest for a long time. As previously stated, the introduction of smart-meters provides a greater insight into the end-users' consumption patterns. Many different clustering methods have been developed with the aim of capturing characteristics of parts of- or entire usage time series, and categorizing them. Like with time series forecasting, there are different kinds of time series clustering, and the optimal method, as well as algorithm, is dependent on the goal one wishes to achieve. This section will show some uses of popular clustering methods, and briefly explain algorithms used in relevant research articles.

Clustering customers based on their usage profiles has been shown in [15] to be a viable method of correctly classifying the various types of customers defined by power grid companies. Self-organizing maps were used in order to capture the most characteristic time points within each series, removing stochastic noise and reducing the computational complexity of the problem by limiting the size of the training set. Then the actual clustering methods were applied. Both K-Means and Hierarchical clustering were used for this purpose. While these two algorithms are widely used, they work quite differently. K-Means requires the number of clusters to be specified by the user, and while there are methods to determine the optimal number of clusters, it is still eventually determined by the user. Hierarchical clustering will initially form a cluster of the two most similar candidates. Then the cluster will be treated as a single candidate, and the process is repeated until there is only one large cluster. The user is required to select at which stage of this process the clusters will be decided. This is why it is said to operate from bottom to top.

ARIMA has for a long time been a popular statistical prediction model. However, it has also been used in model based clustering applications as well [6]. Time series were modeled by the prediction algorithm, and clusters were then created based on these modeled time series. The model captured the characteristics of the time series in such a manner that clustering on these yielded better results than when applying the clustering algorithm to the original time series.

For shape based clustering, algorithms like Dynamic Time Warping are popular. As previously mentioned, a part of shape based clustering is applying some manipulation of the time axis. Although this is done to more accurately align time series displaying similar patterns, the current implementation does not preserve each time series unique characteristics sufficiently for certain clustering applications [16]. This would in turn possibly compromise the ability to create meaningful clusters of similarly behaving time series.

2.3 Improving prediction through clustering

Prediction models have been implemented in various ways in order to improve their accuracy. The combined use of clustering- along with prediction algorithms is one such implementation. This section will look at the use of

clustering to improve predictions directly, or improving data quality allowing for better prediction models to be trained. It will also reflect on how the solutions proposed in the presented articles relate to the problem of this thesis.

In the case of power consumption at the substation level, this can be done as seen in [17][18][19] by clustering the customers under a single substation based on similarity in consumption pattern. A forecast is then made for each cluster, and the summarized forecasts are shown to be more accurate than a single prediction made for the combined usage of all customers under the same substation.

The aforementioned application of clustering to improve forecasting accuracy may help predict yearly events by better predicting each cluster's behavior at that given point in time, but requires a large number of customers to be connected to the substation in question. This is because in order to find similarities in the different consumption patterns one would need an adequate sample size, and also so that the similarities in the patterns will stand out enough when the usage within each cluster is aggregated compared to the stochastic noise.

However, more recently clustering has been applied to independent time series in order to improve load forecast accuracy, as seen in [5]. The goal was to train an ML model across a cluster of similar time series in order to improve forecasts. Predictions were made by an LSTM NN, and the time series were clustered using several clustering algorithms with a feature vector consisting of statistical properties describing each time series. Expanding the training data in this manner is a way of addressing the challenge faced by more complex forecasting models, such as LSTM and other ML algorithms, of having enough data. Further more, it allows the model to better understand recent data, rather than being affected by data that is outdated [5]. This means that the pattern in power consumption is subject to change over time, as the composition of customers underneath a single substation can change. The dataset used to evaluate the model had a monthly resolution [20], and consisted of 72 time series, each with at most 108 datapoints (months of data), so limited historical values were a major challenge. The number of datapoints to forecast was up to 12 months.

While this approach to clustering in order to improve forecast accuracy does address the issue of having insufficient data to train an ML model

like LSTM, the resolution of the time series must be much higher than in these experiments in order to register a single day yearly event. However, it shows that a prediction model can be trained on similar time series to improve predictions for this application. If the method can be adjusted to achieve similar results with an hourly sampled time series, then the ability to predict yearly events can be evaluated.

Using similar sets of data to compensate for missing values has been done in [21]. This method was aimed at creating a regularly spaced time series from sets of irregularly spaced time series, as well as handling missing values with improved accuracy when compared to state-of-the-art imputation methods. The article focuses on what it refers to as Missing at Random (MAR) and Missing Completely at Random (MCAR), the distinction being whether the missingness of datapoints is dependent on variables in the dataset or not [22].

This research highlights the usefulness of information drawn from similar time series for handling the problem of missing values. This is important because for a yearly event to be missing entirely, only a relatively small number of datapoints have to be missing. However, this article does not consider clustering to compensate for missing values in the context of predicting yearly events.

2.4 Summary

The work reviewed in this chapter shows that much effort has been put into tailoring algorithms to both predict- and categorize time series for different purposes. It also highlights that more recently, clustering- and prediction algorithms have been combined to produce forecasts with improved accuracy. However, there still exists a demand for an implementation that targets load forecasting on yearly events. For such an implementation to be successful, it would have to be able to predict electricity consumption more accurately on these events, without compromising predictions outside of the events.

Chapter 3

Proposed Solutions

In this chapter the proposed solution will be described in detail, and it is structured as follows: Section 3.1 will go through the process of combining similar time series to create a single, extended dataset. It will describe the structure of the data, what data is used for the different parts of the solution, and the origin of the data. For certain experiments, the dataset will have to be transformed in order to induce stationarity. This process will also be described in the same section. Section 3.2 will explain how the similar usage profiles are categorized, what distance metric is used to determine similarity, and briefly describe two different clustering algorithms that will be compared in Chapter 4. It will also propose a multi-level approach to clustering. Section 3.3 will explain how the models are trained on the extended datasets to produce more accurate predictions for yearly events, using two different algorithms. The features used to train the model and make predictions will also be described. The parameters used to tune the prediction algorithms will be revealed in the last part of this section.

3.1 Data preparation

3.1.1 Dataset structure

Information about power consumption is gathered from smart-meters measuring usage at the substation level. The dataset has an hourly resolution, is measured in kilo Watts (kW), and consists of *300* time series. These have been selected in a way that ensures that substations with different compositions of connected end-users are represented. This is in order to better preserve the diversity of substation usage profiles, while limiting the computational complexity of both the clustering and prediction problem when performing the experiments. Temperature, measured in Celcius, is used as a feature for a set of experiments, and this data is calculated based on data from the nearest weather stations for each substation.

Clustering is performed on data from *2018*, and predictions are based on data from *2018-06-01* to *2019-11-30*, while predictions are made from *2019-12-01* to *2019-12-31*. This ensures that the clustering is not influenced by the time period that will be predicted, in other words: the prediction test set. It also ensures that when training the prediction models, the time series in the training sets contain a single occurrence of the event to predict each.

While the main focus of the experiments is the prediction accuracy on the yearly event, which spans from *2019-12-24* to *2019-12-26*, the predictions are made for the entire month when the event takes place. This is to better understand how the proposed solution performs outside of the event. If the accuracy decreases significantly outside of the event, the solution will be of no use. In addition to this, a cross-validation is performed for each experiment and aggregated statistics from the results are compared between the proposed solution and the baseline. This will aid in evaluating each model's performance outside of the test set.

3.1.2 Time series rescaling

When introducing other similar time series to the training set of a prediction model, they have to be rescaled. This is because the goal is to simulate having more data than exclusively the original time series, and if the other time series are not to scale, the model will misinterpret the added data. To achieve this, the time series are normalized around the mean for each month. This involves calculating the monthly mean and standard deviation of the original-, and the introduced time series, as shown in Algorithm 1 between line *4* and *9*. Line *9* describes the following steps: The means are first subtracted from the respective introduced time series, and then divided by the standard deviations. Then the standard deviations from the original time series are multiplied with the introduced time series, and the means are added.

Algorithm 1 Monthly rescaling of added time series

1: Initialize $\boldsymbol{timeSeriesCluster} = \{ts_0, ..., ts_N\}$
2: **for each** $tsOriginal \in \boldsymbol{timeSeriesCluster}$ **do**
3: $tsAdditional \leftarrow timeSeriesCluster.GetMostSimilar(tsOriginal)$
4: **for each** $month$ **do**
5: $\sigma_{orig} \leftarrow Std(tsOriginal[month])$
6: $\mu_{orig} \leftarrow Mean(tsOriginal[month])$
7: $\sigma_{add} \leftarrow Std(tsOriginal[month])$
8: $\mu_{add} \leftarrow Mean(tsOriginal[month])$
9: $tsAdditional[month] \leftarrow \frac{(tsAdditional[month] - \mu_{add}) \times \sigma_{orig}}{\sigma_{add}} + \mu_{orig}$
10: **end for**
11: **end for**

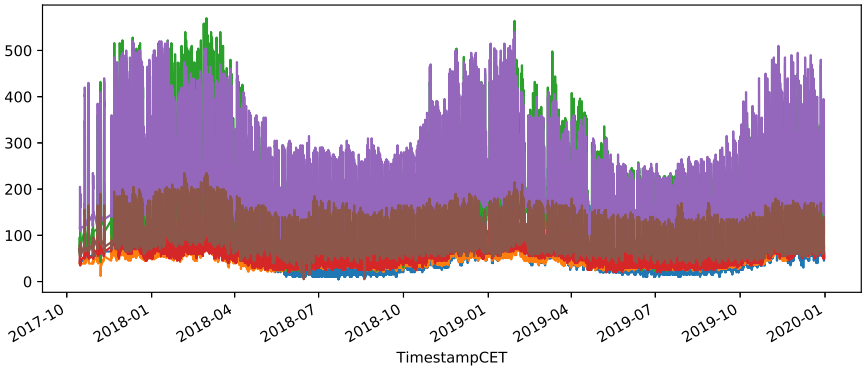

Figure 3.1: Original time series within a cluster.

Although the additional time series behave similarly to the original time series, they still have to be rescaled before they are introduced to the training sets of their respective prediction models, as seen in Figure 3.1 and 3.2.

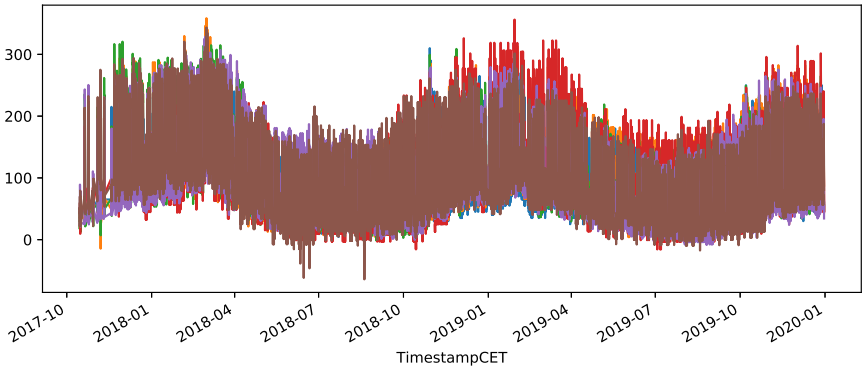

Figure 3.2: Rescaled time series within a cluster.

As a part of the preprocessing stage in the prediction, usage values are clipped at *0*. If a value is below *0* after the rescaling, it will be replaced by a *0*, as negative values are impossible, yet a value of *0* can occur, and must therefore be considered a real value.

3.1.3 Time series filtering

When calculating the cross-correlation between time series, a minimum requirement for number of overlapping data-points is set. If there are fewer than the equivalent of a week, the time series is removed from the dataset. After this filtering step, *272* of the *300* time series in the initial dataset remained. For the clustering experiments, the time series are then limited to the year prior to when the event that will later be predicted takes place. This means that when predicting a yearly event in *2019*, the time series are clustered based on data from *2018*. This is in order to test the validity of the method in a production setting, where the time series are clustered one year in order to improve predictions of the yearly event the following year.

3.1.4 Time series transformation

As stated in [10], LSTM can have difficulties learning low frequency cyclic trends, or the long-term dependencies, in a time series unless enough data is provided. Therefore, the aim is to ensure that the training data does not contain such cyclic components. This is done by analyzing the autocorrelation of the time series for different lag values, and subtracting a lagged value that has a high statistical significance for a given data-point. For the usage time series, a lag value of *168* was selected, corresponding to a lag of one week, and *24* for temperature. When the time series used to train a prediction model is differenced in this manner, the prediction model will not learn the low frequency cyclic trends, but the more stochastic residual. If the model is able to discover a pattern in the residual values, or a correlation between these and a feature, a more accurate prediction can be made by predicting the residual before reintroducing the trends that were removed from the training set. In order to validate the stationarity of the transformed time series, a windowed mean and standard deviation function is then applied to the time series. If these properties are near constant with no apparent trend, it suggests that the time series has achieved a level of stationarity.

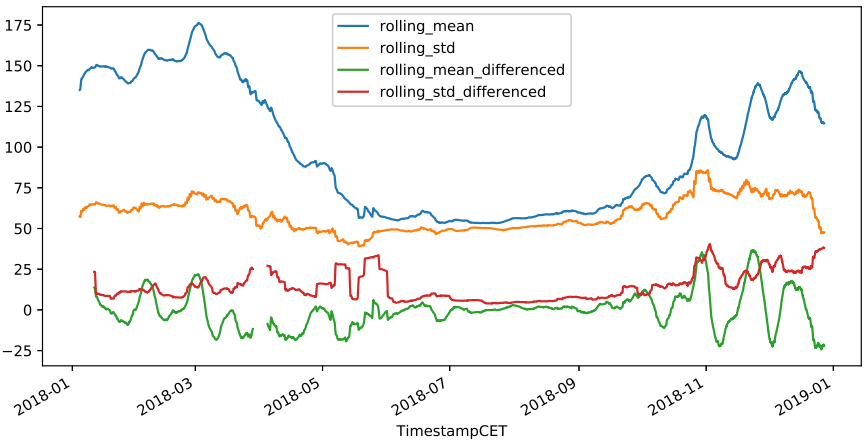

Figure 3.3: Rolling mean and standard deviation for the raw- and differenced usage time series.

An Augmenter Dickey Fuller test is used to further validate the stationarity of the time series after the transformation. It evaluates the null hypothesis which states that the time series has a *unit root*. This means that it displays a form of seasonality, or cyclic trend. The results of the test will reveal with what statistical significance the null hypothesis can be rejected.

Time series	**ADF Statistic**	**p-value**	**CV 1%**	**CV 5%**	**CV 10%**
Raw	-0.833	0.809	-3.431	-2.862	-2.567
Differenced	-8.249	0.000	-3.431	-2.862	-2.567

Table 3.1: Results of an Augmented Dickey Fuller test before and after differencing.

A p-value below *0.05* indicates that we can disprove the presence of a unit root in the time series. As the results in Table 3.1 show, there exists a unit root in the raw time series, but not after the differencing. We can also, by looking at the ADF statistic, with a significance of less than *1%* say that

the time series is stationary after the differencing, since the value is below the *1%* critical value. Figure 3.4 and 3.5 show the effect of the differencing on the correlation with lagged values in the time series, and that the lower frequencies have a significantly reduced influence on the time series. It also clearly shows the presence of a weekly cyclic trend in the raw time series, as every *168th* lagged value has a correlation of close to *1*.

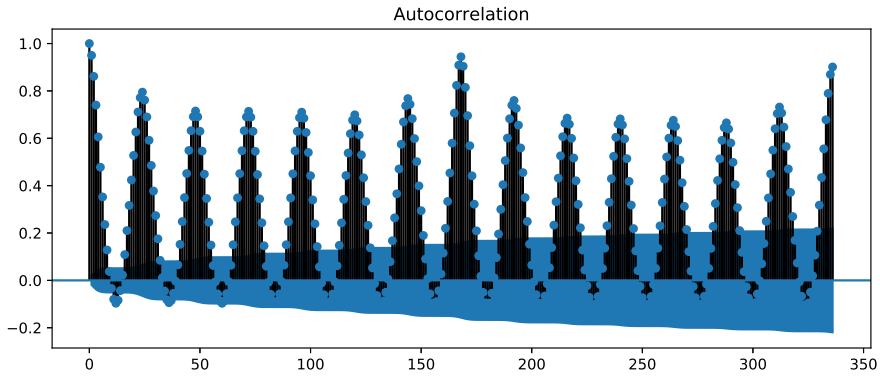

Figure 3.4: Autocorrelogram shows a strong correlation with the lagged value equivalent to last week's value in the raw time series.

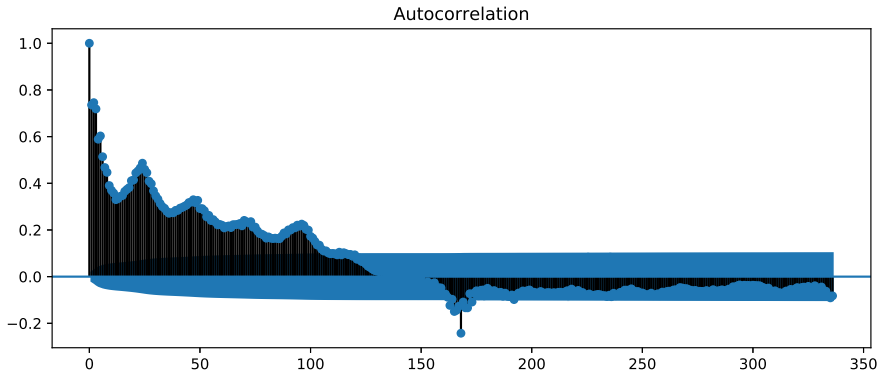

Figure 3.5: Autocorrelogram shows that the correlation with last week's value is greatly reduced after differencing.

Figure 3.5 reveals that there still exist relatively highly correlated lagged values for a given data-point in the time series. However, as the lag increases,

this correlation clearly drops off within a week. The *95%* confidence interval for whether a lagged value is significant to the given data-point or not is shown by the filled area of the plot. This means that statistically, we can expect to see *1* non-significant lagged value outside of this interval for every *20th* data-point, but the trend is still a decreasing correlation as the lag value increases. This suggests that the low frequency cycles have been removed.

Through statistical analysis, the time series have been made stationary through differencing. By removing the low frequency cyclic trends, LSTM will be able to discover patterns in the residual time series, even if it lacks the necessary data to learn the long-term dependencies. The removed trends will then be reintroduced to the predicted values.

3.2 Clustering

3.2.1 Usage profile clustering features

There are several categories of whole time series clustering: feature based, shape based and model based. As stated in [5], an advantage of feature based clustering is its robustness to missing data-points, noise and scale difference of the time series. For example, normalized features can be used, removing the need to rescale each time series when clustering, if it is beneficial to do so. Time series can also be of different lengths, and still be accurately represented by features extracted from them, making the approach more robust in this regard compared to other methods.

The cross-correlation coefficient between two time series is a normalized measure of similarity based on the change in difference between the data-points in each time series. This will be a useful measure of behavioral similarity, as it will be directly affected by the relative change in power consumption between substations. As it is a normalized feature, there is no need for rescaling or normalization of the time series when clustering.

However, as previously mentioned, there are several cyclic trends that will affect the correlation between time series: yearly, weekly and daily. For many substations, the yearly trend, which to a large degree is dependent on the temperature, will have the greatest influence on the cross-correlation. On top of these low frequency cycles, there will typically be a weekly pattern.

Lastly, as a general rule, there exists a daily trend. The importance of each of these trends will depend on how far one wishes to predict, but they should all be accounted for in the clustering process, and a cluster should show similarity also in the higher frequency components. Ideally the weekly trend should indicate how the time series responds to yearly events by revealing the behavior around more frequent, yet special events like weekends.

The experiments conducted in Chapter 4 will include a multi-level clustering, as described in Algorithm 2, one for each component in a predefined set aimed at targeting the cyclic trends. The results will be compared to a baseline where the clustering algorithm is implemented on the raw time series correlation matrix. The time series will be transformed as needed to calculate these different correlation coefficients. This will be achieved through the use of a windowed mean function of appropriate size to smooth out the raw time series to remove higher frequencies. For the yearly component, this will be enough, but in order to isolate the higher frequencies, the lower frequencies must also be removed. These lower frequencies will be removed by subtracting a smoothed version of the same time series.

Algorithm 2 shows pseudo code for the multi-level clustering method. The different components that will be extracted from each time series are: *yearly*, *yearly residual*, *monthly residual* and *weekly residual*. Each of these components are defined by an upper- and lower frequency cut-off value, as shown in line *1*. The values can be seen in Table 3.2. The low-pass values refer to the size of the windowed mean function applied to smooth the time series. High-pass filters also refer to the the same windowed mean function, but when this filter is applied, the resulting time series is subtracted from the original time series. For the yearly component, the high-pass filter is set to *NaN*, which means that this filter is not applied. The same is true for the low-pass filter in the weekly residual component.

Algorithm 2 Multi-level clustering

Initialize $\boldsymbol{components} =$
$\{\{yearly : (NaN, lowpass)\},$
$\{yearlyResidual : (highpass, lowpass)\},$
$\{monthlyResidual : (highpass, lowpass)\},$
$\{weeklyResidual : (highpass, NaN)\}\}$

Initialize $\boldsymbol{timeSeries} =$
$\{\{tsIdx_0 : ts_0\}, ..., \{tsIdx_N : ts_N\}\}$

Initialize $\boldsymbol{tsComponents} = \{\}$

for each $compName, limits \in \boldsymbol{components}$ **do**
    $tsComponents.update(\{compName : \{\}\})$
    for each $tsIdx, ts \in \boldsymbol{timeSeries}$ **do**
        $tsDecomposed \leftarrow FilterHighpass(ts, limits[0])$
        $tsDecomposed \leftarrow FilterLowpass(tsDecomposed, limits[1])$
        $tsComponents[compName].update(\{tsIdx : tsDecomposed\})$
    end for
end for

for each $compName, \_ \in \boldsymbol{components}$ **do**
    $tsComponent \leftarrow tsComponents[compName]$
    $corrMat \leftarrow CreateCorrelationMatrix(tsComponent)$
    $euclidMat \leftarrow CalculateEuclideanDistance(corrMat)$
    $tsCluster \leftarrow C[clusterIdx] \sim CreateClusters(euclidMat)$
    $tsComponents \leftarrow FilterTimeSeries(tsComponents, tsCluster)$
end for

Component	Low-pass	High-pass
Yearly	672	NaN
Yearly residuals	168	672
Monthly residuals	72	168
Weekly residuals	NaN	24

Table 3.2: Table shows filter cut-off values.

For reference, the low-pass filters in Table 3.2 starting from *Yearly* are the equivalent of four week-, one week- and three day intervals. The yearly, and yearly residual components are interpolated using a quadratic function as the large window sizes make them prone to missing values. The graphical representation of these components can be seen in Figure 5.41, 5.42, 5.43 and 4.1. Then, between line *4* and *11*, the individual time series will be decomposed and mapped to the id of the original time series, before being mapped again to their respective component names. The last section, from line *12* to *18*, describes the creation of the correlation matrices for each component, calculation of the pairwise Euclidean distance and clustering process. For each iteration of the loop, a cluster is selected in line *16*, and the clusters formed in the next iteration will be from these time series on the next component. Euclidean distance is a popular measure of distance used for clustering [16], and will allow the model to be able to determine how close each time series is to another in terms of the similarity measure, which in this case is cross-correlation.

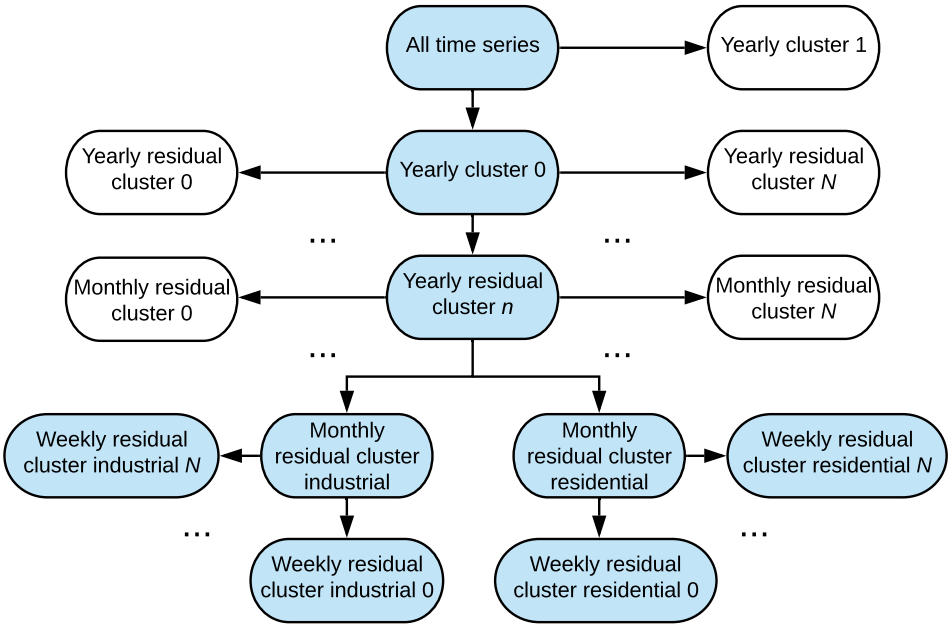

Figure 3.6: Figure shows the cluster selection process where time series are grouped based on cross-correlation on different frequency components within the time series.

3.2.2 Usage profile clustering model

There exists a wide range of clustering algorithms that are useful for different applications when clustering time series. Amongst the most popular are Hierarchical clustering and K-Means. Hierarchical clustering has the advantage of allowing the number of clusters to be determined indirectly by a parameter defining a clustering threshold, as explained earlier. It can be used to cluster similar time series based on a correlation matrix as described in [23]. The distance between each cluster candidate to each cluster can also be measured from for example the farthest-/nearest point or centroid of each cluster. The latter of which allows one to lessen the effect of outliers on the distance between clusters and new candidates. However, complexity increases faster per data-point than with models like K-Means, thus making it less ideal for large clustering operations. Since both K-Means and Hierarchical clustering are commonly used clustering algorithms, they will both be tested, and results will contribute to the model selection.

3.3 Prediction with clustering

3.3.1 Load forecasting features

In order to analyze the effects of introducing additional data from a cluster to the training set, features should be limited to time stamp meta data for a portion of the experiments. What is meant by this, is to limit the features to what can be deduced from the time stamp of each data-point within each time series: year, month of year, day of month, day of week, hour of day and information about various holidays. This is in order to isolate the effect of the additional data as much as possible. Importance of relevant features will be analyzed and compared between the models where additional events were introduced to the training set, and where they were not.

The time stamp meta data will not be represented categorically or numerically, i.e. the feature *hour* being 1, 2, 3, etc., but rather as a cyclic feature, represented by the sine and cosine values as seen in [24]. The cyclic features are created as shown in Equation 3.1 and 3.2: Each value in the feature column is first rescaled so they span from *0* to 2π. Then the sine and cosine function is applied, resulting in a column for the sine- and another for the cosine value.

$$cosFeature = \cos(2\pi \times featureValue/lengthOfPeriod), \tag{3.1}$$

$$sinFeature = \sin(2\pi \times featureValue/lengthOfPeriod). \tag{3.2}$$

The advantage of this approach over using categorical- or numerical features in this case is that it will more accurately represent the relationship between the values of the features: hour *23* is as close to hour *0* as hour *0* is to hour *1*. This will not be reflected when the feature is represented categorically or numerically. The aim is to allow the models to better understand the cyclic nature of these features, thus improving prediction accuracy. Then other features can be added, such as temperature, and the effect can be analyzed further. The impact of introducing additional events will then be analyzed in the same manner as described earlier.

The features used for the prediction models in the experiments conducted in Chapter 4 will be: a cyclic encoding of hour of day, -day of week and

-month of year, and a column for a selection of Norwegian holidays where *1* indicates that the observation takes place on the corresponding holiday, otherwise the column is *0*. The set of holiday features consists of: Boxing Day, first- and second day of Pentecost, Easter Sunday and -Monday, Workers' Day, New Year's Eve, Constitution Day, Christmas Eve, Ascension Day, Good Friday, Easter (whole week) and Maundy Thursday. For a set of experiments, temperature is also added.

3.3.2 Load forecasting model

XGBoost and LSTM are popular ML prediction models that are effective at making short-term load forecasts. A major advantage of using tree models like XGBoost over various forms of NNs is that they can more easily be analyzed. The forests created in the training phase can be displayed, and the training can then be evaluated. This is especially useful when experimenting with training models in less traditional ways, like introducing multiple time series into the training set.

XGBoost can also display feature importance by various metrics, both by number of times the feature made an impact on the prediction, as well as the improvement in prediction accuracy. This can be useful when evaluating the features related to the yearly events when additional events are added, and will be used to analyze the effect of adding time series to the training set. XGBoost has been used specifically for this purpose in earlier research [25], in conjunction with another ML model for performing the actual prediction. There are various metrics for estimating the importance of a feature, but for this purpose, it is most useful to look at *gain*. Rather than for example counting the number of times the feature occurs in each tree or branch in the trained model, gain is determined by the degree to which the feature improves the prediction accuracy during the training phase.

Predictions will also be made using LSTM to see if the method of introducing additional data from similar time series can be successfully implemented using a popular state-of-the-art prediction algorithm for short-term load forecasting. Like with the XGBoost experiments, a regular implementation of LSTM will be used as a baseline, and results will be analyzed and compared to the clustering approach where the model trains on additional time series. The necessary data preparation described earlier will be applied when utilizing LSTM, and predicted values will be processed accordingly.

Algorithm 3 Prediction based on similar time series

Initialize ***timeSeriesCluster*** $= \{ts_0, ..., ts_N\}$

for each $tsOriginal \in$ ***timeSeriesCluster*** **do**
    $tsAdditional \leftarrow timeSeriesCluster.GetMostSimilar(tsOriginal)$
    $tsAdditional \leftarrow MonthlyRescale(tsAdditional, \text{“}mean\text{”})$

    if $includeTemperature$ **then**
        $tsAdditional \leftarrow AddTemperature(tsAdditional)$
        $tsOriginal \leftarrow AddTemperature(tsOriginal)$
    end if

    if $predictionAlgorithm == \text{“}LSTM\text{”}$ **then**
        $tsAdditional \leftarrow Stationarize(tsAdditional)$
        $tsAdditional \leftarrow Normalize(tsAdditional, \text{“}minmax\text{”})$
        $tsOriginal \leftarrow Stationarize(tsOriginal)$
        $tsOriginal \leftarrow Normalize(tsOriginal, \text{“}minmax\text{”})$
    end if

    $tsCombined \leftarrow tsAdditional.Append(tsOriginal)$
    $tsCombined \leftarrow CreateTimeStampFeatures(tsCombined)$
    $model \leftarrow model.Train(tsCombined)$
    $prediction \leftarrow model.Predict()$

    if $predictionAlgorithm == \text{“}LSTM\text{”}$ **then**
        $prediction \leftarrow DeTransform(prediction)$
    end if
end for

Algorithm 3 shows the pseudocode for the prediction method. Firstly, a cluster of time series from Section 3.2 is selected. Then, for each of the time series, the following steps are applied: Line *3* is where the time series which is most similar to the original time series based on the weekly residuals is drawn out from the cluster. Then, in the following line, it is rescaled for each month as explained at the beginning of the chapter and in Algorithm 1. For the experiments including weather data, temperature is added to both time series. Between line *9* and *14* each time series is made stationary and normalized if the selected prediction algorithm for the experiment is LSTM. For XGBoost, this step is not applied. The original- and the additional time series are then combined to a single time series, which the prediction model is trained on. In line *16*, features from the time stamp of each value in the training set, like hour of day and holidays, are extracted. If the prediction algorithm used is LSTM, the predicted values must be detransformed as they will be normalized and stationary like the training set.

The LSTM used in this thesis is a Python implementation provided by Keras. The model consisted of one LSTM layer with *48* hidden neurons, and a Dense layer, also provided by Keras. Instead of adding an Input layer, the input data was reshaped to a 1D array, and its dimensions were passed directly to the first layer of the model. The batch size was set to *168*, updating weights for each time the model has seen a week's worth of hourly values, and epochs to *100*, passing the training set through the network *100* times. The selected number of epochs resulted in the loss function converging without increasing after reaching its minimum, which could have suggested overfitting. Mean squared error was used as the loss function, and the learning rate was set to *0.01*.

The XGBoost model used is also a Python implementation provided by the XGBoost API. The fraction of randomly sampled feature columns before creating each tree in the training phase was set to *0.4*, and the randomly sampled fraction of observations was set to *0.6*. A learning rate of *0.01* was used. Max depth of each tree in the trained model was set to *10*, minimum child weight to *1.5*, and regularization parameters gamma, reg_alpha and reg_lambda were set to *1*, *0.75* and *0.45* respectively. The number of estimators was set to *2000* and the objective function *reg:squarederror* was used to train the model.

Chapter 4

Experiments and Results

This chapter will present and analyze the results of the experiments conducted as part of evaluating the proposed solution, and is structured as follows: Section 4.1 will show and explain results that contributed to the clustering model selection. It will also describe how the models are implemented for the different experiments. Two clustering algorithms are evaluated, both with the proposed multi-level implementation, as well as a baseline approach. Section 4.2 will explain the choice of clustering algorithm. Section 4.3 will present the actual prediction results and how the experiments were conducted. The proposed solution where prediction models are trained on multiple time series is compared to a baseline where models are trained on a single time series each. Section 4.4 will analyze the results further as they relate to the problem statement, and comment on various aspects of the experiments and how they were conducted.

4.1 Clustering model evaluation

In this section, two clustering algorithms will be evaluated: Hierarchical clustering and K-Means. Each algorithm will be implemented using a multi-level approach, as explained in Chapter 3. The results will be compared with each other, as well as with a baseline implementation of the algorithms. The goal of these experiments is to determine which algorithm to use as part of the proposed prediction with clustering method.

In the following experiments, the unprocessed time series are referred to as the *raw* time series, meaning that no components like cyclic trends are extracted. This method of clustering will act as a baseline for each model, and will be compared to the multi-level clustering implementation. Results will be presented as tables with statistics describing the clusters formed by each implementation of the clustering algorithms. Because the statistics are from the final clusters, those that are formed by clustering on multiple time series components have one statistic for residential profiles, and another for industrial profiles. These two sets of profiles distinctly resemble the profiles of substations where all connected usage points are either residential or industrial, but have been detected at an earlier stage in the multi-level clustering implementation.

After filtering the time series as a part of the data preparation stage, the remaining dataset consists of 272 time series. In order to create the fea-

tures for the clustering algorithms, the correlation matrix is made, and the pairwise Euclidean distance is calculated. In order to evaluate the proposed clustering algorithms' ability to form groups based on the pairwise Euclidean distance of the cross-correlation between the time series, statistics are gathered from each cluster for each implementation, which are labeled as follows:

- Hierarchical clustering on the raw time series (HR)
- Hierarchical clustering performed on multiple components (HMI for industrial clusters, HMR for residential clusters)
- K-Means on the raw time series (KR)
- K-Means performed on multiple components (KMI for industrial clusters, KMR for residential clusters)

Statistics for each model implementation are calculated by looking at the correlation matrix of each cluster, denoted as *clusterCorrMats* in Algorithm 4. This algorithm is run for each model implementation in order to better evaluate the clustering performance. The number of rows, mean and standard deviation are calculated for each column of the matrix, as seen between line *5* and *9*. The result is that each cluster's correlation matrix is reduced to a set of aggregated statistical values for each column. The average of each statistic from the current matrix is stored from line *10* to *12*. At this point each matrix is reduced to a single average mean, -standard deviation and -row count value. Then an average for each statistic is calculated across all matrices, as shown from line *14* to *16*, giving a general overview of each cluster's similarity with a single average cluster size number, -mean and -standard deviation. In line *17*, the total number of time series in all clusters is added resulting in the following values: average cluster size (*avg cnt*), average mean correlation (*avg mean*), average standard deviation (*avg std*), and total number of members in all clusters (*tot cnt*). This is done for the original, unprocessed time series within each cluster, as well as each time series component.

Algorithm 4 Correlation matrix statistics

Initialize $\boldsymbol{clusterCorrMats} = \{corrMat_0, ..., corrMat_N\}$
Initialize $\boldsymbol{avgStats} = \{\{std : \{\}\}, \{mean : \{\}\}, \{cnt : \{\}\}\}$
for each $corrMat \in \boldsymbol{clusters}$ **do**
    Initialize $\boldsymbol{stats} = \{\{std : \{\}\}, \{mean : \{\}\}, \{cnt : \{\}\}\}$
    for each $col \in \boldsymbol{corrMat}$ **do**
        $stats[std].append(Std(col))$
        $stats[mean].append(Mean(col))$
        $stats[cnt].append(Len(col))$
    end for
    $avgStats[std].append(Mean(stats[std]))$
    $avgStats[mean].append(Mean(stats[mean]))$
    $avgStats[cnt].append(Mean(stats[cnt]))$
end for
Initialize $avgCnt \leftarrow Mean(avgStats[cnt])$
Initialize $avgMean \leftarrow Mean(avgStats[mean])$
Initialize $avgStd \leftarrow Mean(avgStats[std])$
Initialize $totCnt \leftarrow Sum(avgStats[cnt])$

When determining the parameters for the different clustering algorithms, it is important to ensure that the resulting clusters contain distinct groups like residential- and industrial profiles. However, if many time series with one of these usage profiles appear as lone members in their own clusters, the clustering will not be of any use, as there will be no other time series to append to the original time series to create the extended prediction model training sets. The number of centroids for the KR clustering experiments was set to a value of *100*, at which distinct clusters of usage profiles emerged, without removing the residential or industrial profiles entirely. With the number set too high, the remaining clusters would mostly be various residential profiles. In the HR clustering experiments, a similar goal was achieved by selecting a threshold value of *0.05*. This value is relative to the maximum value of the matrix used to evaluate the distance between cluster candidates. For the multilevel experiments, the threshold and number of clusters was set to an appropriate value for each step in order to first filter out outliers for the yearly component. The process was then repeated for the yearly-, monthly- and weekly residuals. For the K-Means experiments, the number of yearly clusters was set to *2*, yearly residuals *2*, monthly residuals *4*, and weekly residuals for residential- and industrial clusters *10*. For the Hierarchical clustering experiments, the threshold value relative to the maximum value of the distance matrix was set to *0.2* for the yearly component, *0.5* for the yearly residual, *0.175* for the monthly residual, and *0.15* for the weekly residual for the industrial cluster, and *0.05* for the residential.

stat	**HMI**	**HMR**	**HR**	**KMI**	**KMR**	**KR**
avg cnt	4.5	8	9	5.333	9.167	3.804
avg mean	0.902	0.962	0.893	0.882	0.915	0.848
avg std	0.077	0.028	0.085	0.077	0.046	0.156
tot cnt	18	16	171	16	55	213

Table 4.1: Statistics for original component of time series.

Table 4.1 shows statistics from the final clusters on the unprocessed time series for each model and implementation. There is a high average correlation for all model implementations, but the total number of time series within each cluster varies greatly. For HMI, HMR, KMI and KMR, this is

mostly due to the filtering that happens when the residential and industrial clusters are formed at the monthly residual level. For HR and KR, clusters that consist of only a single member will be filtered out. Because Hierarchical clustering only forms groups of the most similar member candidates for each step of the algorithm, outliers might be filtered out more easily compared to when K-Means is implemented and clusters comprised of only a single member are removed. This could explain why the total count is lower when Hierarchical clustering is implemented.

stat	**HMI**	**HMR**	**HR**	**KMI**	**KMR**	**KR**
avg cnt	4.5	8	9	5.333	9.167	3.804
avg mean	0.921	0.998	0.977	0.964	0.991	0.912
avg std	0.078	0.002	0.023	0.032	0.008	0.106
tot cnt	18	16	171	16	55	213

Table 4.2: Statistics for yearly component of time series.

When analyzing the yearly component in Table 4.2 of the same clusters, the cross-correlation is, as hypothesized, high for both HR and KR as it will have had a large effect on the correlation between the raw time series.

stat	**HMI**	**HMR**	**HR**	**KMI**	**KMR**	**KR**
avg cnt	4.5	8	9	5.333	9.167	3.804
avg mean	0.789	0.958	0.812	0.84	0.885	0.792
avg std	0.167	0.029	0.172	0.102	0.062	0.22
tot cnt	18	16	171	16	55	213

Table 4.3: Statistics for yearly residual component of time series.

For the yearly residual, there is a distinct drop in correlation for most profiles in Table 4.3. The standard deviation is also higher for these clusters. HMI has the lowest average cross-correlation, while HMR has the highest. The

residential profiles clustered using K-Means are also more highly correlated for this component than the industrial profiles on average.

stat	HMI	HMR	HR	KMI	KMR	KR
avg cnt	4.5	8	9	5.333	9.167	3.804
avg mean	0.95	0.883	0.804	0.912	0.741	0.772
avg std	0.038	0.081	0.158	0.059	0.124	0.221
tot cnt	18	16	171	16	55	213

Table 4.4: Statistics for monthly residual component of time series.

At the monthly residuals, shown in Table 4.4, the model implementations start to deviate in terms of both consistency as seen in *avg std*, and *avg mean*. The main reason why Hierarchical clustering is able to outperform K-Means at this point might be due to the bottom to top approach. This could lead to Hierarchical clustering being able to better distinguish the residential profiles without filtering out the industrial profiles in later iterations of the clustering operation. Note that the filtering is not part of the algorithms themselves, but clusters of single time series are filtered out as part of the clustering script.

stat	HMI	HMR	HR	KMI	KMR	KR
avg cnt	4.5	8	9	5.333	9.167	3.804
avg mean	0.902	0.872	0.677	0.863	0.691	0.661
avg std	0.079	0.091	0.236	0.083	0.134	0.315
tot cnt	18	16	171	16	55	213

Table 4.5: Statistics for weekly residual component of time series.

Finally, Table 4.5 shows the statistics for the correlation between the weekly residuals. At this level, both models outperform their baseline implementation, in terms of mean values and standard deviation. Figure 4.1 shows the

weekly residual component of a sample of time series. It becomes clear that this component can reveal certain characteristics of a substation's usage profile on events related to time stamp meta data, like day of week. Therefore, it is hypothesized that a high cross-correlation between time series in this component is a good indication of similar behavior on yearly events.

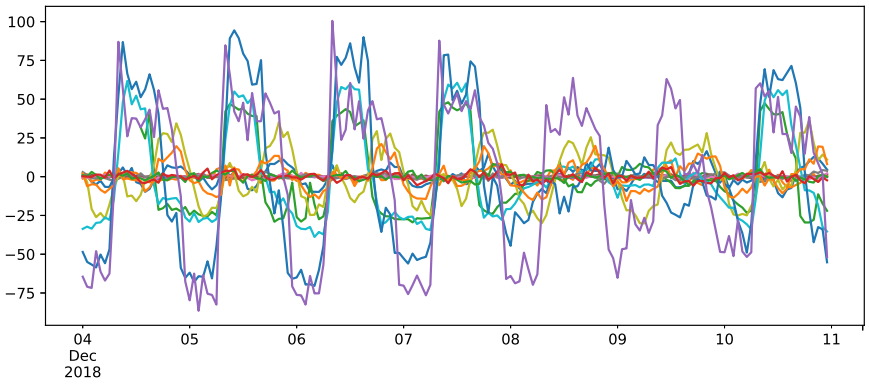

Figure 4.1: Weekly residual component.

4.2 Clustering model selection

From the results of the experiments analyzed in this chapter, it is concluded that Hierarchical clustering should be utilized as a part of the proposed prediction method. This is because it is difficult to determine the optimal number of clusters in this case, as is required when using K-Means, and it is more logical for this application to let the number of clusters be determined by a similarity parameter. It would also seem that outliers will not be assigned a cluster until later iterations of the clustering process, which allows them to be filtered out through the removal of clusters with only a single member. The algorithm will be implemented using the multi-level clustering approach. This will ensure a high cross-correlation for all cyclic trends described earlier, and also make use of the distinction between residential- and industrial profiles found in the clustering based on monthly residuals. In Section 4.3, a single residential cluster, along with a single industrial cluster will be used to evaluate the prediction models, as well as the proposed solution where the models are trained on additional time series.

4.3 Prediction with clustering

The prediction experiments were conducted using two different algorithms: XGBoost and LSTM. A cluster of industrial profiles, and a cluster of residential profiles were used, and a prediction was made by each algorithm for every time series within the clusters for one month containing a yearly event. The average errors were calculated using overall Mean Absolute Percentage Error (MAPE), MAPE on the yearly event, MAPE on non events, as well as a daily MAPE. In Equation 4.1, t_0 denotes the first hour, T denotes the last hour, and N is the number of values in the error estimate.

$$MAPE = \frac{1}{N} \sum_{t=t_0}^{T} \left| \frac{real[t] - predicted[t]}{real[t]} \right|. \tag{4.1}$$

In order to detect overfitting, a 5-fold cross-validation step was also added to the experiments. Temperature was added to the training sets as a feature for one set of experiments, while another consisted of only usage data, as well as time stamp meta data, as explained earlier.

A case was also added where the time series that the model was predicting had the event removed before training. This was done in order to evaluate the method's ability to make up for missing data by supplying the training set with events from a similar time series. The yearly event selected for the experiments was Christmas Eve, Christmas Day and Boxing Day, and time series training sets started at *2018-06-01*, and ended at *2019-11-30*, leaving each time series with a single occurrence of the yearly event.

Initially, experiments were run with the inclusion of all time series within the selected cluster when training the prediction model, but as this seemed to decrease accuracy for predictions outside of the time period of the event, only one additional time series was introduced. When selecting which time series to introduce to the training set, the one with the highest correlation on the weekly residuals was selected.

Error values shown in Table 4.6, 4.7, 4.10 and 4.11 are categorized as follows:

- MAPE using additional data from cluster on event (MCE)
- MAPE not using data from cluster on event (MNCE)

- MAPE using additional data from cluster not on event (MCNE)
- MAPE not using additional data from cluster not on event (MNCNE)

Experiments shown in Table 4.6, 4.7, 4.10 and 4.11 are labeled as follows:

- Industrial cluster (I)
- Residential cluster (R)
- Univariate training data (U)
- Temperature as feature (T)
- Missing event in predicted time series (M)

4.3.1 XGBoost

As shown in Table 4.6 in column MCE and MNCE, the model trained on an additional time series from its cluster has improved accuracy on average when predicting the yearly event, with one exception. This suggests that when the clustering method is applied, the model is able to predict the event more accurately when compared to the baseline, as well as in the case of the actual event being absent in the original time series, as seen in Figure 4.2 and 4.3. The MAPE is calculated per day to better understand how the prediction models perform throughout the predicted time period. The average of these values are then calculated for all experiments conducted for the industrial and the residential cluster respectively. *clustered_mape* denotes the results when prediction models were trained on additional time series, while *non_clustered_mape* represents when they were trained on a single time series.

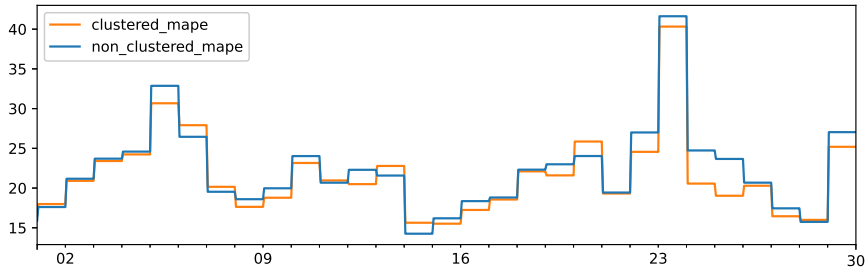

Figure 4.2: Figure shows average daily MAPE for the time series in the industrial cluster performed by XGBoost.

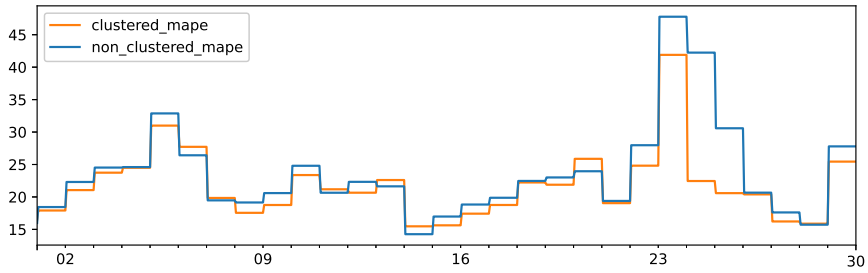

Figure 4.3: Figure shows average daily MAPE for the time series in the industrial cluster performed by XGBoost when the yearly event was missing in the time series to predict.

As for experiment RUM, there are two important findings to point out: the baseline performed better than when additional data was introduced, and the baseline performed better when the event was not present in the time series at all. When observing the results in Figure 4.4 and 4.5 it becomes apparent that the decrease in accuracy happens on Christmas Eve, while Christmas Day saw improved predictions, and there was little difference on Boxing Day.

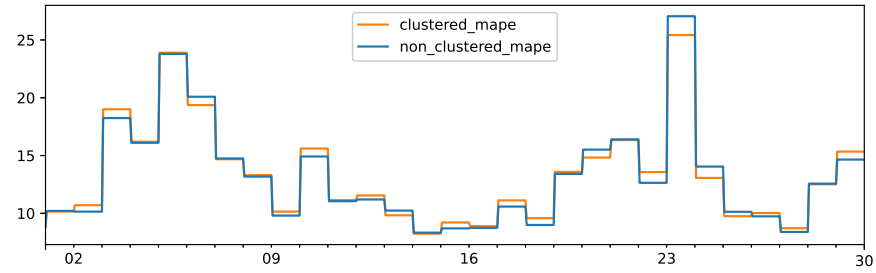

Figure 4.4: Figure shows average daily MAPE for the time series in the residential cluster performed by XGBoost.

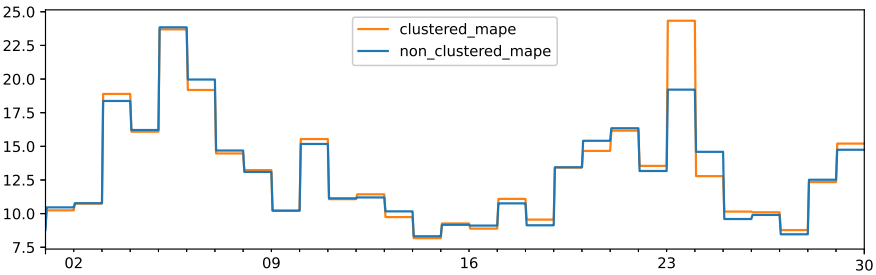

Figure 4.5: Figure shows average daily MAPE for the time series in the residential cluster performed by XGBoost when the yearly event was missing in the time series to predict.

This might suggest that for this cluster, the model is not able to learn the influence of the feature related to Christmas Eve with either implementation. This is also true when temperature is added as a feature. However, when temperature is introduced to the training set, the model trained on an additional time series from the same cluster, MAPE is lower in both cases, and the difference is much smaller between the experiment where the event is present in the original time series and where it is not (see Figure 4.6 and 4.7).

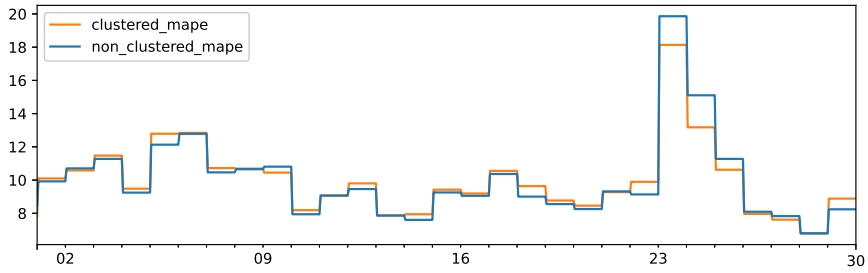

Figure 4.6: Figure shows average daily MAPE for the time series in the residential cluster performed by XGBoost.

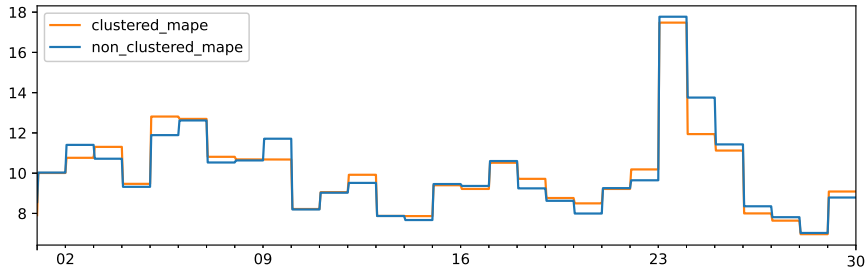

Figure 4.7: Figure shows average daily MAPE for the time series in the residential cluster performed by XGBoost when the yearly event was missing in the time series to predict.

When we look at the predictions made for the rest of the month, excluding the yearly event (MCNE and MNCNE), the predictions for the time series within the industrial cluster are improved for all experiments when the model is trained on multiple time series. As expected, the improvement is more subtle than for the event, but performance is improved. The difference in prediction accuracy is much smaller outside of the event, suggesting that this approach achieves the goal of specifically improving the prediction of yearly events by bettering the model's understanding of the corresponding feature.

experiment	MCE	MNCE	MCNE	MNCNE
IU	**26.639**	30.008	**20.719**	21.011
IUM	**28.311**	40.198	**20.768**	21.329
RUM	15.757	**14.467**	**12.718**	12.725
RU	**16.08**	17.076	12.777	**12.594**
IT	**31.184**	32.411	**18.155**	18.243
ITM	**32.257**	40.558	**18.084**	18.447
RTM	**13.515**	14.319	9.56	**9.511**
RT	**13.978**	15.413	9.527	**9.394**

Table 4.6: Average error values on various subsections of the predicted time series measured in MAPE predicted by XGBoost.

The decrease in accuracy for certain predictions of power consumption on the yearly event when temperature is introduced as a feature might be caused by the event's effect not being related to temperature. This can be seen in IT and ITM in Table 4.6. As the prediction model to a larger degree relies on temperature in order to make a forecast, the importance of features indicating that an event takes place might not influence the model to the same degree. When we analyze the feature importance calculated by the XGBoost model, as shown in 4.8, it shows that when a similar time series from a cluster is introduced to the model's training data, the importance of features related to the yearly event is increased more in the experiments where temperature is introduced.

In addition to observing the average errors in predictions and comparing the proposed solution to the baseline approach, it is necessary to analyze the standard deviation in accuracy. Table 4.7 shows the average standard deviation in prediction accuracy for the prediction made for each time series within each cluster. Once more, statistics are calculated both for the event, as well as for the rest of the month, excluding the event. The proposed use of clustering to improve prediction accuracy for yearly events is of no use if the prediction accuracy is much less consistent than the baseline approach. However, there is generally little difference between the two approaches in this regard, and for most experiments the proposed solution has a smaller standard deviation in prediction accuracy.

experiment	MCE	MNCE	MCNE	MNCNE
IU	**5.292**	5.976	**7.675**	7.738
IUM	**4.272**	6.147	**7.594**	7.842
RUM	**2.534**	2.77	1.886	**1.869**
RU	**1.693**	1.843	**1.878**	1.894
IT	**8.483**	8.934	**5.779**	6.234
ITM	**6.371**	6.703	**6.044**	6.533
RTM	**2.779**	2.908	**1.105**	1.111
RT	2.661	**2.059**	1.144	**1.199**

Table 4.7: Standard deviation of error values on various subsections of the predicted time series measured in MAPE predicted by XGBoost.

As previously explained, XGBoost evaluates the importance of each feature in the training set that results in splits in the various trees when the model is trained. Table 4.8 shows the percentage change in importance of the features related to the yearly event in the prediction experiments in terms of gain when comparing the proposed solution to the baseline. The results show a consistently higher increase in feature importance for the experiments where temperature was used as a feature. The events are denoted as follows: Christmas Eve (CE), Christmas Day (CD) and Boxing Day (BD).

experiment	CE	CD	BD
IU	-3.769	10.819	15.421
RU	1.625	-4.775	25.696
IT	30.52	31.207	28.318
RT	6.515	18.145	27.062

Table 4.8: Table shows percentage increase in feature importance when comparing the proposed solution to the baseline.

The cross-validation results from each experiment are also evaluated. If the proposed solution where prediction models are trained on multiple time series has increased accuracy, but the cross-validation score is significantly lower, it suggests overfitting. This means that the model has been trained on noise, rather than learning characteristic patterns within the time series. The average cross-validation results for each experiment are labeled

experiment	MC	SC	MNC	SNC
IU	0.842	0.03	0.674	0.082
IUM	0.84	0.031	0.678	0.084
RUM	0.827	0.074	0.352	0.298
RU	0.827	0.076	0.359	0.281
IT	0.875	0.024	0.738	0.081
ITM	0.874	0.024	0.741	0.085
RTM	0.892	0.038	0.623	0.143
RT	0.892	0.04	0.623	0.139

Table 4.9: Table shows the average cross-validation results from the XGBoost experiments.

as follows: mean cross-validation clustered (MC), standard deviation cross-validation clustered (SC), mean cross-validation non clustered (MNC), and standard deviation cross-validation non clustered (SNC).

The results reveal that the models show less signs of overfitting in all the experiments where the proposed solution is utilized. When inspecting the standard deviation in the cross-validation score, it also shows that these results are more consistent than for the baseline. This underlines the validity of the actual prediction results.

4.3.2 LSTM

The LSTM experiments were conducted in the same way as those for XGBoost, except that the time series for temperature and usage were made stationary as explained earlier. When analyzing the average statistics for predictions made for the event, most experiments show an improved accuracy when the additional time series is introduced to the training set. However, when temperature is added as a feature, the predictions have a decreased accuracy when including an additional time series for the residential cluster (RT and RTM), as seen in Figure 4.8 and 4.9.

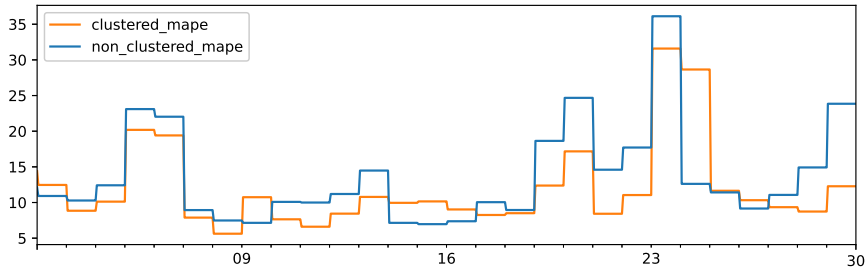

Figure 4.8: Figure shows average daily MAPE for the time series in the residential cluster performed by LSTM.

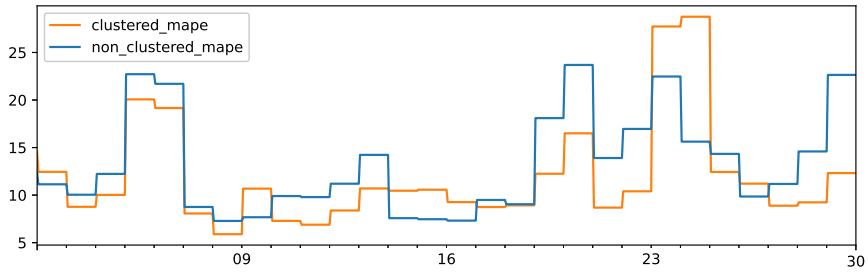

Figure 4.9: Figure shows average daily MAPE for the time series in the residential cluster performed by LSTM when the yearly event was missing in the time series to predict.

For these two experiments, the accuracy for the rest of the month is still improved. In fact, when the temperature is used as a feature, the accuracy outside of the event is improved for all experiments, while it is decreased for *3* out of *4* experiments where the dataset was univariate.

experiment	MCE	MNCE	MCNE	MNCNE
IU	**31.928**	34.401	20.099	**20.035**
IT	**27.655**	29.261	**17.267**	19.25
ITM	**27.115**	39.349	**17.277**	17.663
RTM	22.543	**17.024**	**11.0**	12.625
RT	23.269	**18.59**	**10.932**	12.751
IUM	**35.104**	50.42	22.6	**19.917**
RUM	**26.81**	27.645	19.998	**18.059**
RU	**26.082**	26.093	**18.001**	18.026

Table 4.10: Average error values on various subsections of the predicted time series measured in MAPE predicted by LSTM.

Table 4.11 shows much inconsistency in the prediction accuracy when comparing the baseline implementation to the proposed solution. However, for the experiments where the yearly event was removed from the original time series, the standard deviation in accuracy for the event is lower when an additional time series is introduced, except for RTM. The industrial clusters show the greatest improvement.

experiment	MCE	MNCE	MCNE	MNCNE
IU	5.416	**4.492**	7.893	**7.397**
IT	6.806	**3.227**	6.895	**6.403**
ITM	**7.624**	10.825	**6.269**	7.72
RTM	4.468	**3.714**	**1.694**	2.741
RT	**3.558**	3.784	**1.413**	2.68
IUM	**9.785**	13.549	10.837	**7.833**
RUM	**4.926**	6.382	**3.437**	3.888
RU	5.33	**5.169**	**2.345**	4.007

Table 4.11: Standard deviation of error values on various subsections of the predicted time series measured in MAPE predicted by LSTM.

When examining the cross-validation results for the LSTM experiments in 4.12, it becomes apparent that the models score very low. While the results are more stable when the proposed solution is utilized, they are either equal to- or in certain cases lower than the baseline. The low scores

over all might suggest that the models are not able to learn the cyclic trends in the time series. This could indicate that the differencing used for removing the lower frequency patterns in the time series left a time series with no significant pattern for the model to learn. It is important to keep in mind that the scores reflect the model's ability to predict the residuals after the time series has been differenced. The scores can be expected to be lower, as the seasonality has been removed for the most part, meaning that the residuals will be less predictable.

experiment	**MC**	**SC**	**MNC**	**SNC**
IU	0.005	0.001	0.005	0.002
IT	0.005	0.001	0.005	0.002
ITM	0.005	0.001	0.005	0.002
RTM	0.003	0.001	0.007	0.003
RT	0.003	0.001	0.007	0.003
IUM	0.005	0.001	0.005	0.002
RUM	0.003	0.001	0.007	0.003
RU	0.003	0.001	0.006	0.003

Table 4.12: Table shows the average cross-validation results from the LSTM experiments.

4.4 Discussion

When conducting the experiments described earlier in this chapter, observations were made that will be analyzed further in this section. Comments will also be made on results and how certain experiments were conducted.

Both K-Means and Hierarchical clustering were able to create clusters containing distinct industrial- and residential profiles. However, the optimal number of clusters for K-Means was difficult to determine through the elbow method, and was increased until the distinct profiles emerged, like with the Hierarchical clustering results. The fact that it was possible to distinguish time series with a residential- and industrial profile is interesting, and might prove useful as it is not always easy to determine this without manually labelling each time series through observing it. Also the resulting

profile is, as stated earlier, not necessarily determined purely by the number of industrial or residential customers connected to a substation, but in combination with their total usage for each hour.

While both LSTM and XGBoost could improve predictions for the industrial profiles, LSTM with the proposed solution for expanding the model's training set performed worse at the yearly event for residential profiles when temperature was added as a feature when compared to the regular implementation. This could be due to the added complexity caused by the differencing, which is necessary to make the time series stationary. Perhaps through further analysis of the residential time series one could solve this issue, but then the process would be more difficult to automate, and thus harder to implement on a large scale. This might suggest that it is beneficial to use models like XGBoost that are able to handle non-stationary data with the provided amount in these experiments when implementing the method described in this thesis, as it can learn these cyclic characteristics of the time series on its own. In order to compare the two prediction models fairly, one would have to run experiments where both models are trained on stationary time series. The goal of the experiments that have been conducted was to evaluate the proposed solution for improving the prediction of time series around yearly events through clustering of similarly behaving time series with two different popular algorithms.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

It is necessary for grid companies to make accurate predictions of its customers' power consumption in order to provide high quality electricity reliably. Certain events that correlate with abnormally high- or low usage take place once per year. It can be difficult for prediction models to make accurate load forecasts for these events, as the models will require an entire year of additional data in order to acquire another example of the event for their training sets. If the event is missing in the training set, the model cannot predict its influence on power consumption until another year of data is gathered.

In this thesis, a method is proposed that seeks to improve load forecasting accuracy on yearly events by appending similar time series to create extended training sets for prediction models. By introducing additional time series, we can also add more yearly events from said time series to the training set. A clustering algorithm creates groups of similarly behaving time series, and each extended training set is formed by transforming the appended time series to fit the scale of the initial time series.

Through experiments conducted on different usage profiles, the proposed solution is shown to increase accuracy when predicting electricity usage on the event, without significantly affecting the forecast outside of the event. The method is also shown to be effective at compensating for missing events in the original time series. The method developed for transforming the appended time series successfully simulates having a larger training set for prediction models to be trained on. However, when the time series and its features are differenced and normalized, results indicate that the method might have to be adjusted.

The state-of-the-art, presented as part of the related work in this thesis, addresses the issue of inadequate prediction accuracy through the use of clustering. However, this thesis contributes with a solution that can improve the prediction of power consumption on yearly events for time series with an hourly resolution. It also does not require a large number of end-users to be connected to a single substation. These two advantages are this thesis's main contributions to the state-of-the-art.

5.2 Future Work

Future research on the topic of improving prediction of usage on yearly events should consider implementing the solution proposed in this thesis on a larger scale, on other profiles of similarly behaving time series for usage on the substation level. Also, performing experiments with another yearly event, possibly one that occurs on different dates each year, like Easter.

Experiments could be conducted in order to further explore the possibility of adding additional time series to the training set. For the experiments described in this thesis the number of added time series was limited to one, but it is possible that a process could be developed through which the optimal number of time series could be determined. It would also be interesting to compare prediction accuracy when a single time series has two occurrences of an event to a prediction made when the model was trained on two similar time series, each with only one occurrence of the same event.

Another possibility would be to develop a method of training a single model for all time series within a cluster. The purpose of this would not only be to decrease the number of models that would have to be trained, but to improve the accuracy of the predictions for each time series performed by a single model. This would require the time series to be transformed in a way that solves the scale issue.

References

[1] Eugene A Feinberg and Dora Genethliou. Load forecasting. In *Applied mathematics for restructured electric power systems*, pages 269–285. Springer, 2005.

[2] Jorge Valenzuela, Mainak Mazumdar, and Anoop Kapoor. Influence of temperature and load forecast uncertainty on estimates of power generation production costs. *IEEE Transactions on Power Systems*, 15(2):668–674, 2000.

[3] George Gross and Francisco D Galiana. Short-term load forecasting. *Proceedings of the IEEE*, 75(12):1558–1573, 1987.

[4] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. Time-series clustering–a decade review. *Information Systems*, 53:16–38, 2015.

[5] Kasun Bandara, Christoph Bergmeir, and Slawek Smyl. Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach. *Expert Systems with Applications*, 140:112896, 2020.

[6] Konstantinos Kalpakis, Dhiral Gada, and Vasundhara Puttagunta. Distance measures for effective clustering of arima time-series. In *Proceedings 2001 IEEE international conference on data mining*, pages 273–280. IEEE, 2001.

[7] MY Cho, JC Hwang, and CS Chen. Customer short term load forecasting by using arima transfer function model. In *Proceedings 1995 International Conference on Energy Management and Power Delivery EMPD'95*, volume 1, pages 317–322. IEEE, 1995.

[8] G Juberias, R Yunta, J Garcia Moreno, and C Mendivil. A new arima model for hourly load forecasting. In *1999 IEEE Transmission and Distribution Conference (Cat. No. 99CH36333)*, volume 1, pages 314–319. IEEE, 1999.

[9] Stylianos I Vagropoulos, GI Chouliaras, Evaggelos G Kardakos, Christos K Simoglou, and Anastasios G Bakirtzis. Comparison of sarimax, sarima, modified sarima and ann-based models for short-term pv generation forecasting. In *2016 IEEE International Energy Conference (ENERGYCON)*, pages 1–6. IEEE, 2016.

[10] Shahzad Muzaffar and Afshin Afshari. Short-term load forecasts using lstm networks. *Energy Procedia*, 158:2922–2927, 2019.

[11] Chujie Tian, Jian Ma, Chunhong Zhang, and Panpan Zhan. A deep neural network model for short-term load forecast based on long short-term memory network and convolutional neural network. *Energies*, 11(12):3493, 2018.

[12] Nengbao Liu, Vahan Babushkin, and Afshin Afshari. Short-term forecasting of temperature driven electricity load using time series and neural network model. *Journal of Clean Energy Technologies*, 2(4):327–331, 2014.

[13] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[14] Xiaoqun Liao, Nanlan Cao, Ma Li, and Xiaofan Kang. Research on short-term load forecasting using xgboost based on similar days. In *2019 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS)*, pages 675–678. IEEE, 2019.

[15] Teemu Räsänen, Dimitrios Voukantsis, Harri Niska, Kostas Karatzas, and Mikko Kolehmainen. Data-based method for creating electricity use load profiles using large amount of customer-specific hourly measured electricity use data. *Applied Energy*, 87(11):3538–3545, 2010.

[16] Warissara Meesrikamolkul, Vit Niennattrakul, and Chotirat Ann Ratanamahatana. Shape-based clustering for time series data. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 530–541. Springer, 2012.

[17] Fateme Fahiman, Sarah M Erfani, Sutharshan Rajasegarar, Marimuthu Palaniswami, and Christopher Leckie. Improving load forecasting based on deep learning and k-shape clustering. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 4134–4141. IEEE, 2017.

[18] Mahesh Kumar and Nitin R Patel. Using clustering to improve sales forecasts in retail merchandising. *Annals of Operations Research*, 174(1):33–46, 2010.

[19] Kasun Amarasinghe, Daniel L Marino, and Milos Manic. Deep neural networks for energy load forecasting. In *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*, pages 1483–1488. IEEE, 2017.

[20] Computational Intelligence in Forecasting. Computational intelligence in forecasting international time series competition 2016, 2016.

[21] Agung Andiojaya and Haydar Demirhan. A bagging algorithm for the imputation of missing values in time series. *Expert Systems with Applications*, 129:10–26, 2019.

[22] Roderick JA Little. A test of missing completely at random for multivariate data with missing values. *Journal of the American statistical Association*, 83(404):1198–1202, 1988.

[23] Gautier Marti, Frank Nielsen, Philippe Donnat, and Sébastien Andler. On clustering financial time series: a need for distances between dependent random variables. In *Computational Information Geometry*, pages 149–174. Springer, 2017.

[24] Kaweh Mansouri, John HK Liu, Robert N Weinreb, Ali Tafreshi, and Felipe A Medeiros. Analysis of continuous 24-hour intraocular pressure patterns in glaucoma. *Investigative ophthalmology & visual science*, 53(13):8050–8056, 2012.

[25] Huiting Zheng, Jiabin Yuan, and Long Chen. Short-term load forecasting using emd-lstm neural networks with a xgboost algorithm for feature importance evaluation. *Energies*, 10(8):1168, 2017.

Appendix

XGBoost results

Industrial univariate

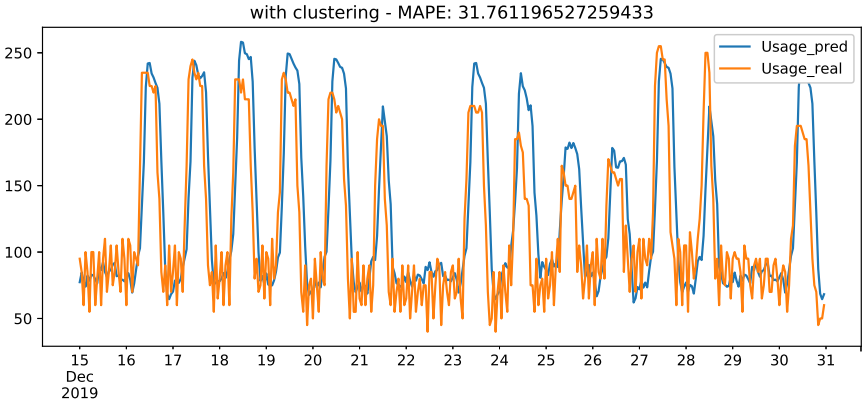

Figure 5.1: Figure shows a prediction for an industrial cluster when XGBoost is trained on an additional time series.

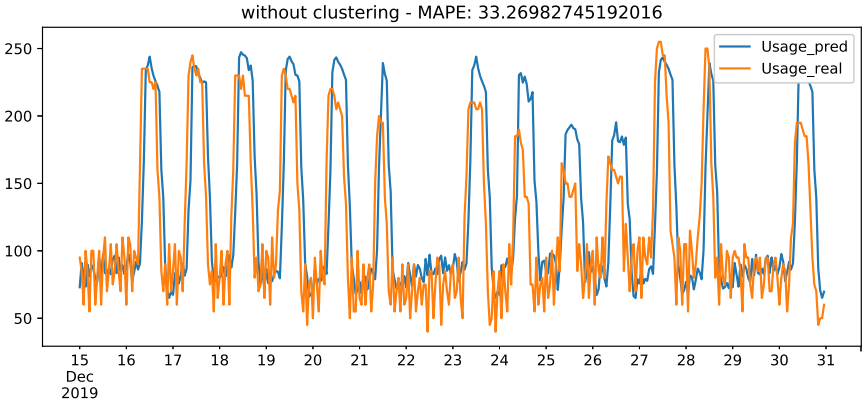

Figure 5.2: Figure shows a prediction for an industrial cluster when XGBoost is trained on a single time series.

Industrial univariate missing event

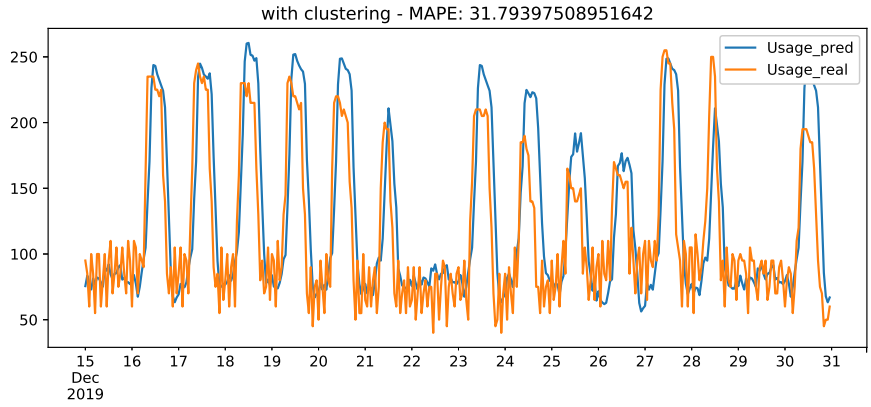

Figure 5.3: Figure shows a prediction for an industrial cluster when XGBoost is trained on an additional time series when the event is missing in the original time series.

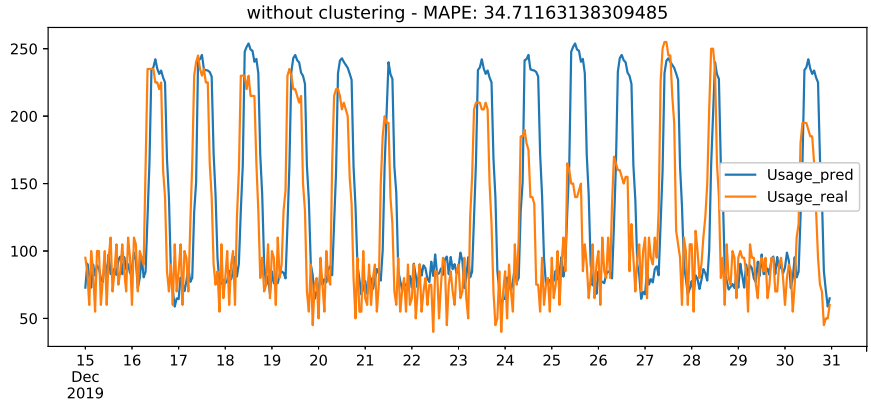

Figure 5.4: Figure shows a prediction for an industrial cluster when XGBoost is trained on a single time series when the event is missing.

Residential univariate

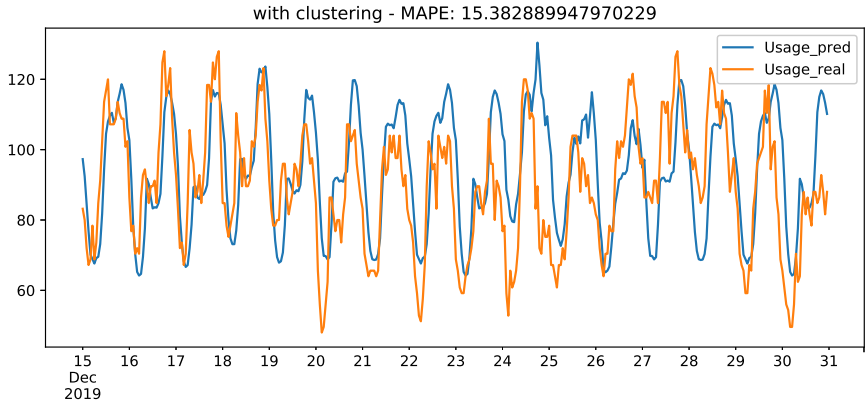

Figure 5.5: Figure shows a prediction for a residential cluster when XGBoost is trained on an additional time series.

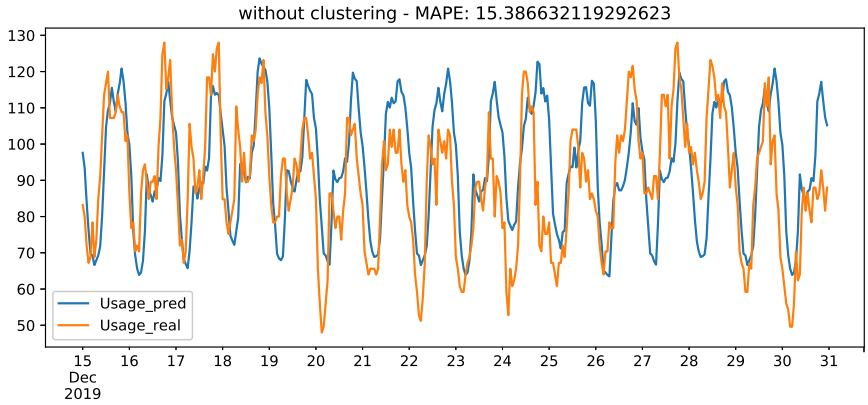

Figure 5.6: Figure shows a prediction for a residential cluster when XGBoost is trained on a single time series.

Residential univariate missing event

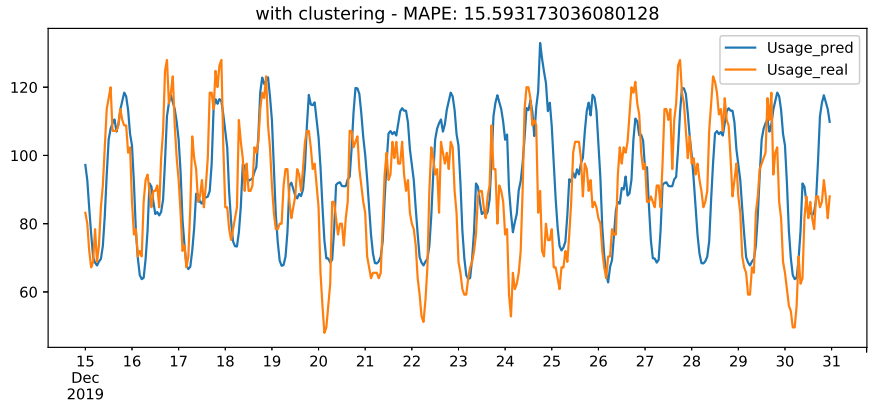

Figure 5.7: Figure shows a prediction for a residential cluster when XGBoost is trained on an additional time series when the event is missing in the original time series.

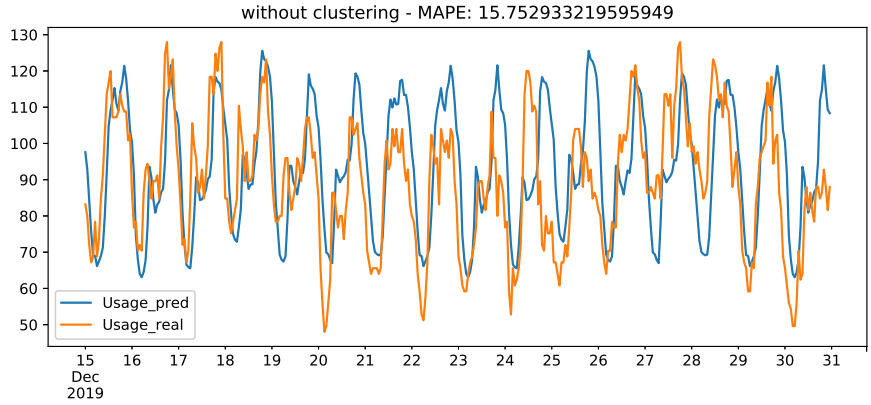

Figure 5.8: Figure shows a prediction for a residential cluster when XGBoost is trained on a single time series when the event is missing.

Industrial with temperature

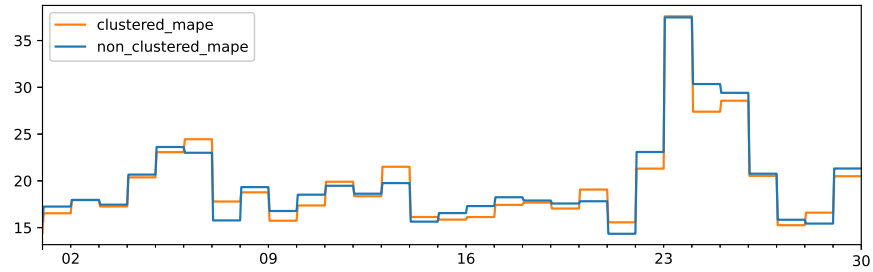

Figure 5.9: Figure shows average daily MAPE for the time series in the industrial cluster performed by XGBoost.

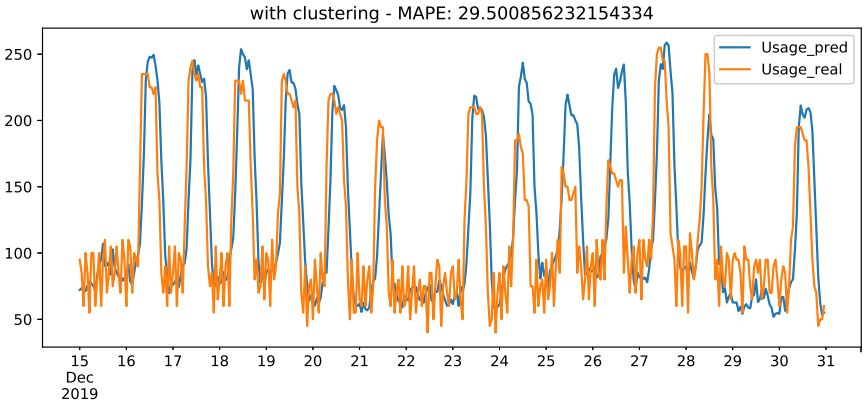

Figure 5.10: Figure shows a prediction for an industrial cluster when XGBoost is trained on an additional time series.

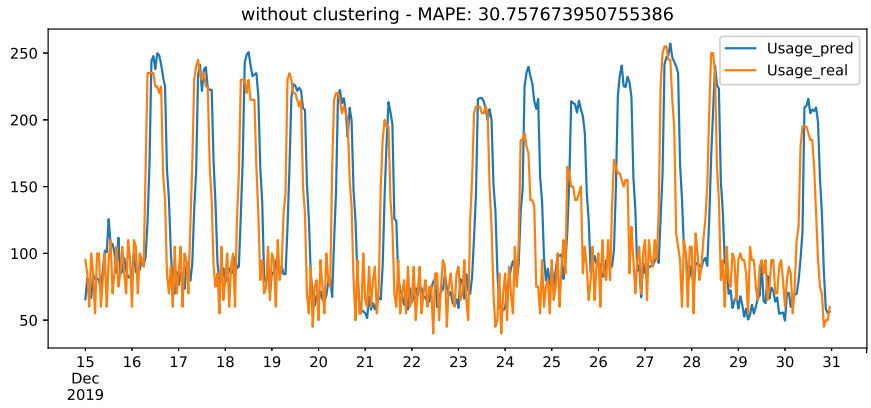

Figure 5.11: Figure shows a prediction for an industrial cluster when XGBoost is trained on a single time series.

Industrial with temperature missing event

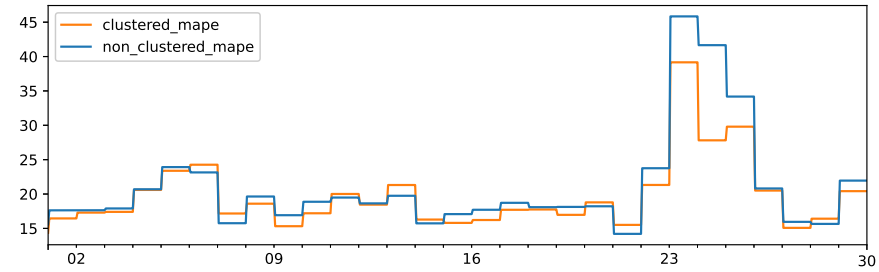

Figure 5.12: Figure shows average daily MAPE for the time series in the industrial cluster performed by XGBoost when the yearly event was missing in the time series to predict.

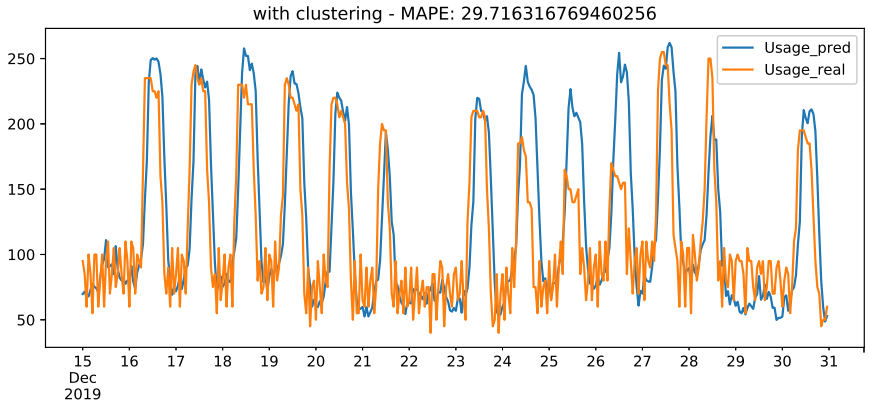

Figure 5.13: Figure shows a prediction for an industrial cluster when XGBoost is trained on an additional time series when the event is missing in the original time series.

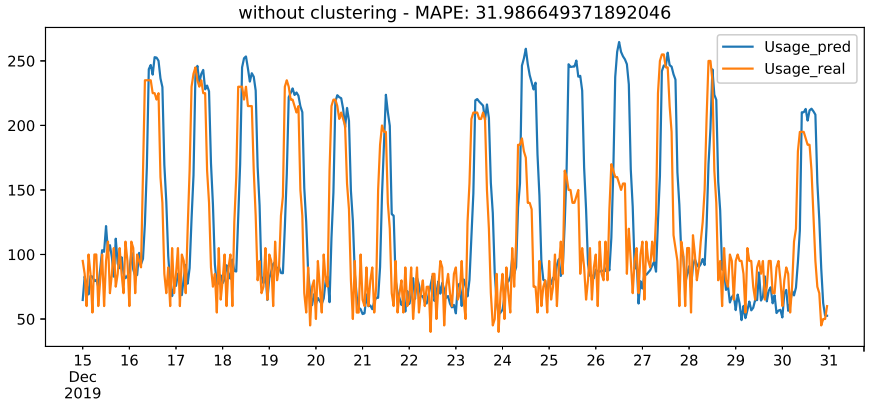

Figure 5.14: Figure shows a prediction for an industrial cluster when XGBoost is trained on a single time series when the event is missing.

Residential with temperature

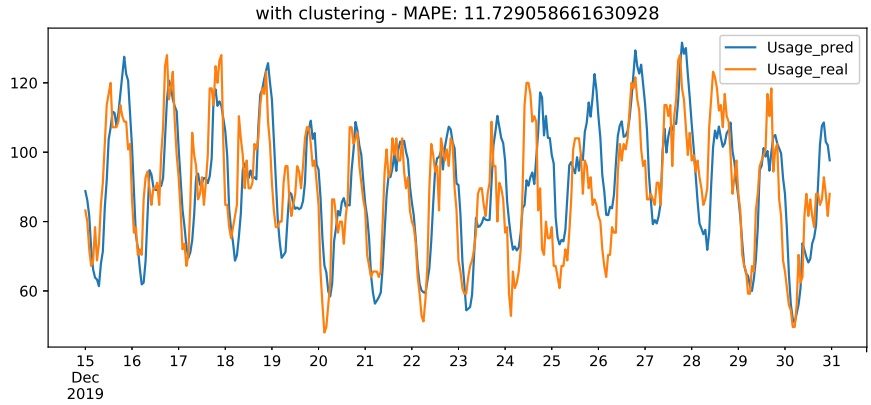

Figure 5.15: Figure shows a prediction for a residential cluster when XGBoost is trained on an additional time series.

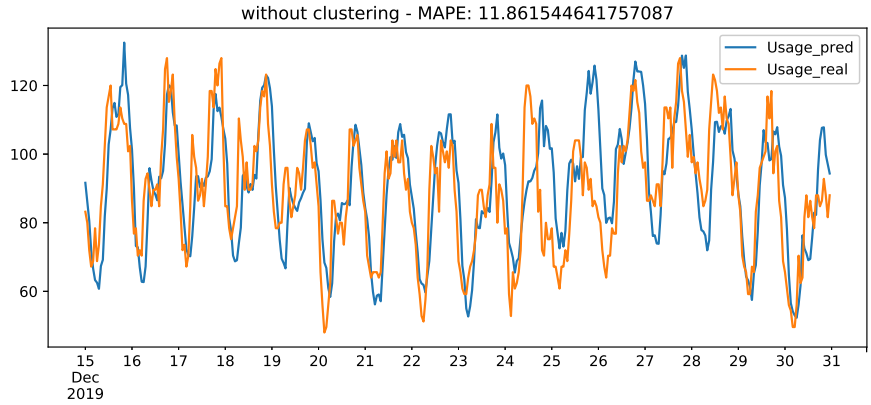

Figure 5.16: Figure shows a prediction for a residential cluster when XGBoost is trained on a single time series.

Residential with temperature missing event

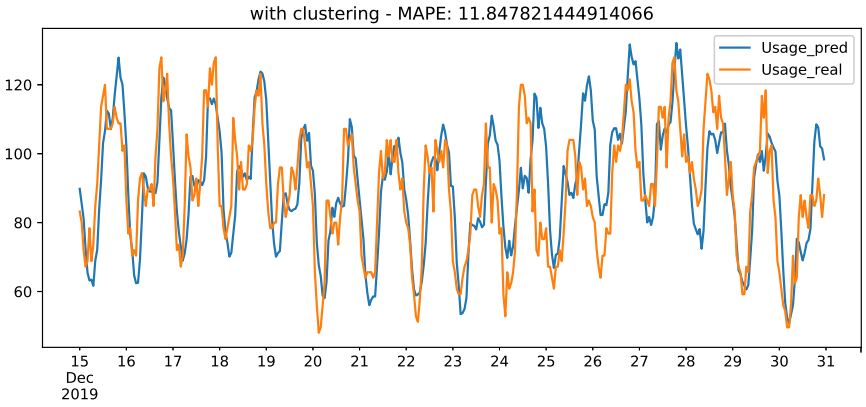

Figure 5.17: Figure shows a prediction for a residential cluster when XGBoost is trained on an additional time series when the event is missing in the original time series.

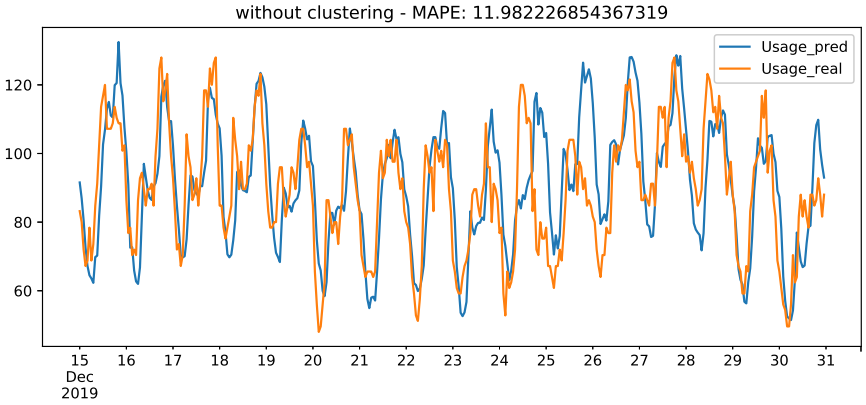

Figure 5.18: Figure shows a prediction for a residential cluster when XGBoost is trained on a single time series when the event is missing.

LSTM results

Industrial univariate

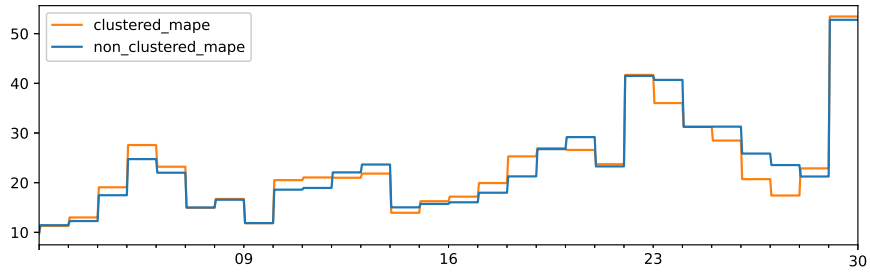

Figure 5.19: Figure shows average daily MAPE for the time series in the industrial cluster performed by LSTM.

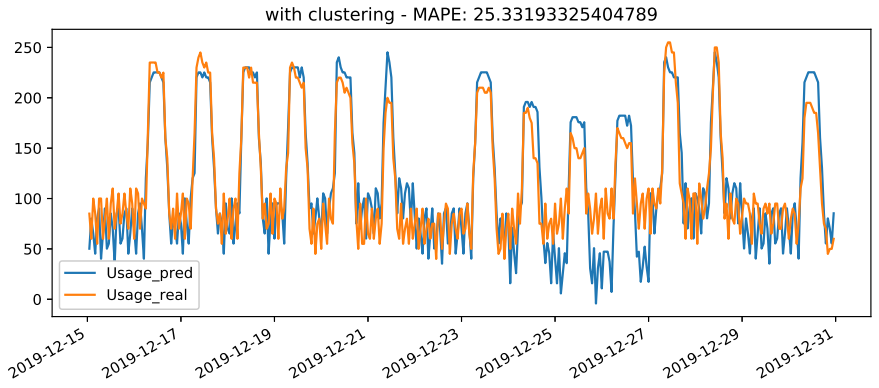

Figure 5.20: Figure shows a prediction for an industrial cluster when LSTM is trained on an additional time series.

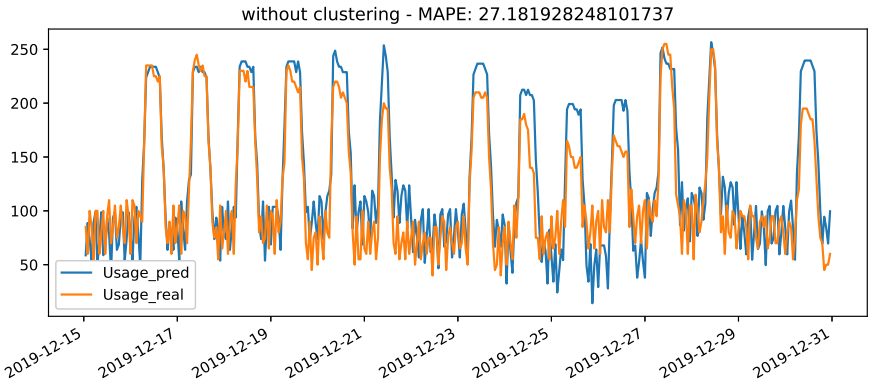

Figure 5.21: Figure shows a prediction for an industrial cluster when LSTM is trained on a single time series.

Industrial univariate missing event

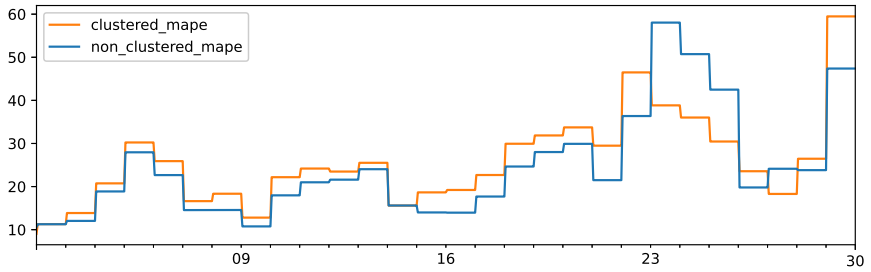

Figure 5.22: Figure shows average daily MAPE for the time series in the industrial cluster performed by LSTM when the yearly event was missing in the time series to predict.

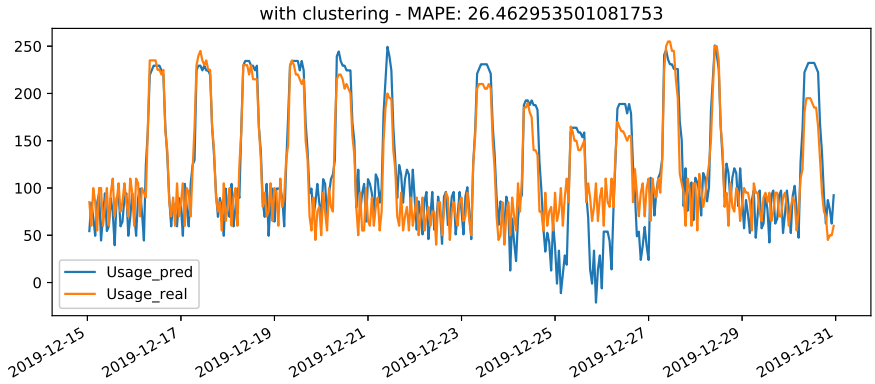

Figure 5.23: Figure shows a prediction for an industrial cluster when LSTM is trained on an additional time series when the event is missing in the original time series.

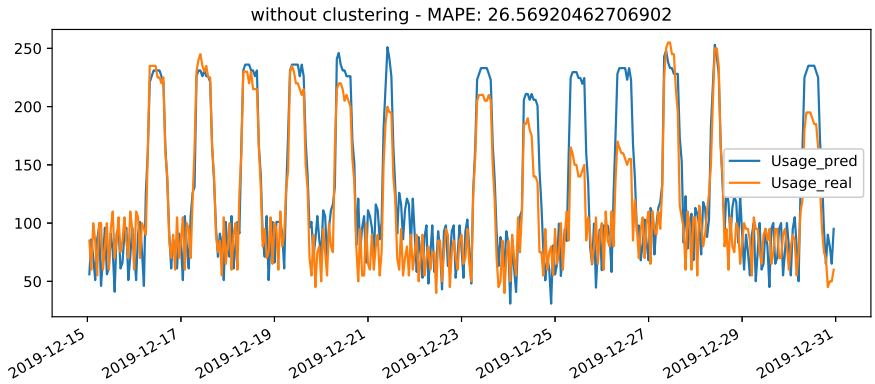

Figure 5.24: Figure shows a prediction for an industrial cluster when LSTM is trained on a single time series when the event is missing.

Residential univariate

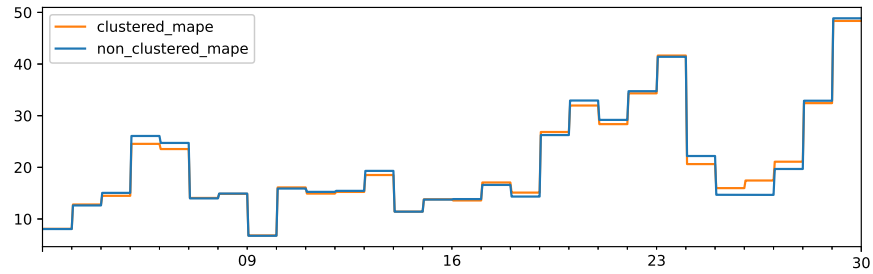

Figure 5.25: Figure shows average daily MAPE for the time series in the residential cluster performed by LSTM.

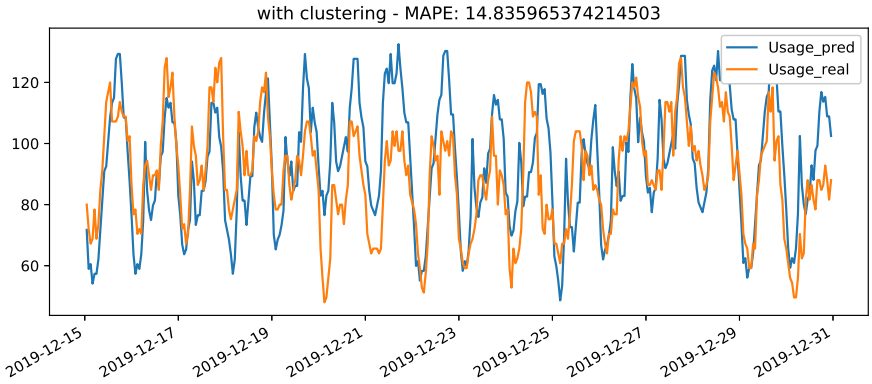

Figure 5.26: Figure shows a prediction for a residential cluster when LSTM is trained on an additional time series.

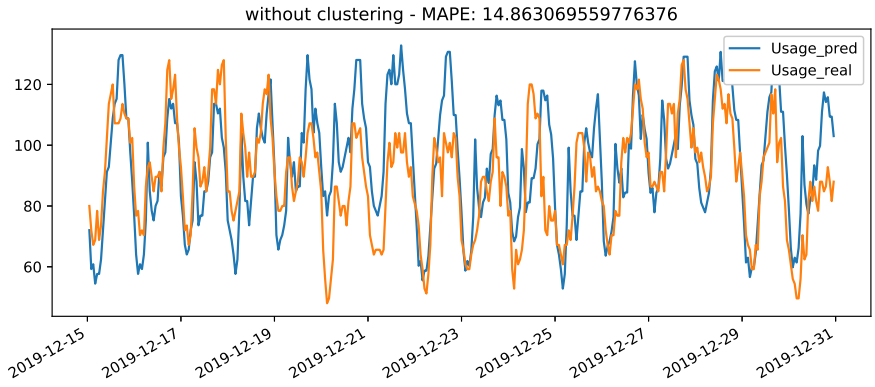

Figure 5.27: Figure shows a prediction for a residential cluster when LSTM is trained on a single time series.

Residential univariate missing event

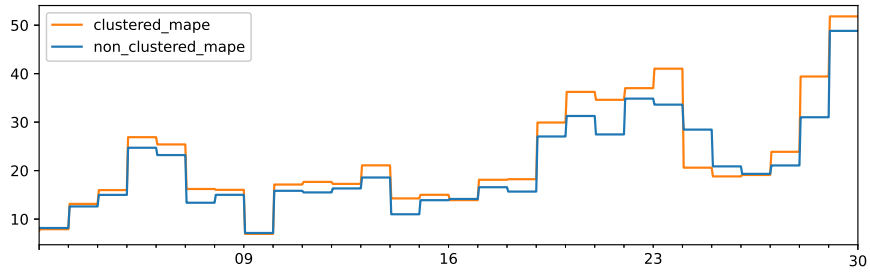

Figure 5.28: Figure shows average daily MAPE for the time series in the residential cluster performed by LSTM when the yearly event was missing in the time series to predict.

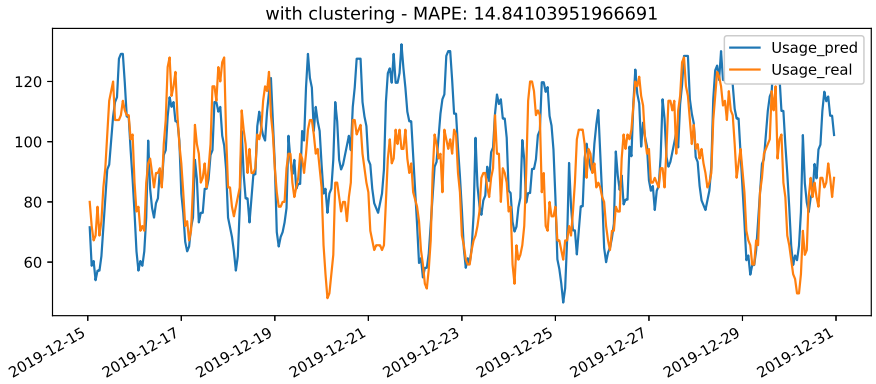

Figure 5.29: Figure shows a prediction for a residential cluster when LSTM is trained on an additional time series when the event is missing in the original time series.

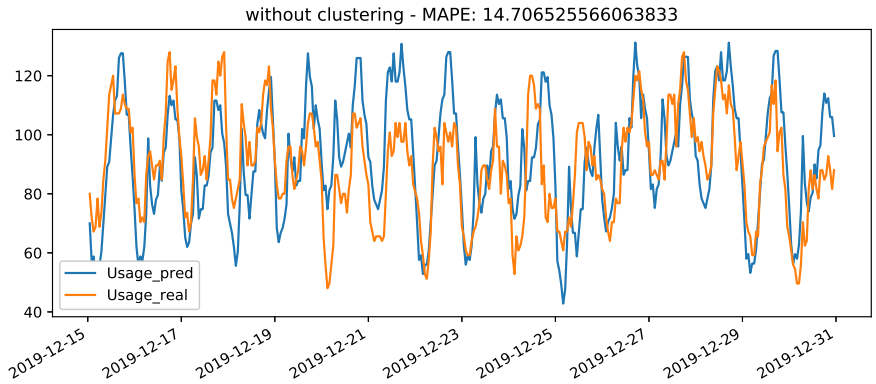

Figure 5.30: Figure shows a prediction for a residential cluster when LSTM is trained on a single time series when the event is missing.

Industrial with temperature

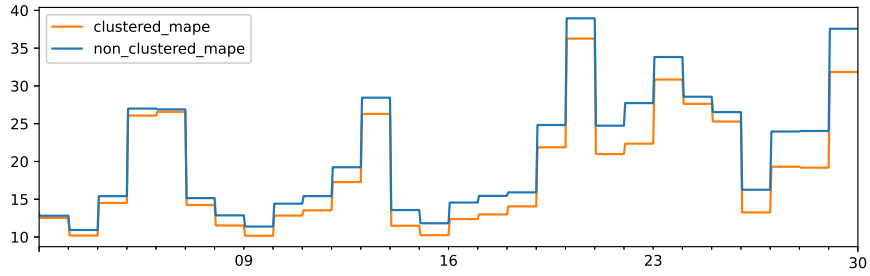

Figure 5.31: Figure shows average daily MAPE for the time series in the industrial cluster performed by LSTM.

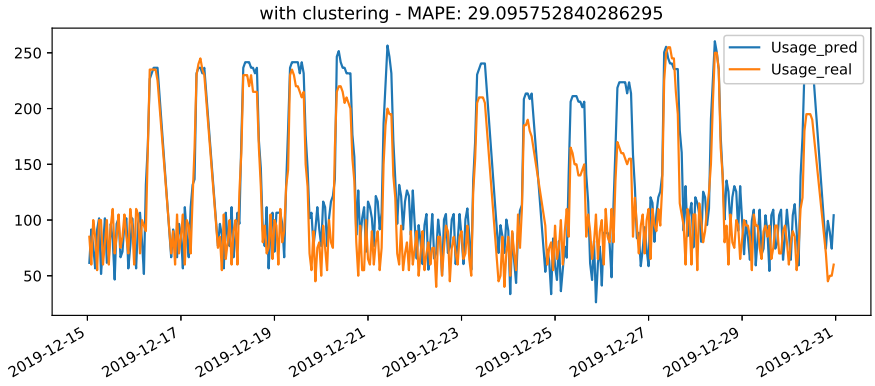

Figure 5.32: Figure shows a prediction for an industrial cluster when LSTM is trained on an additional time series.

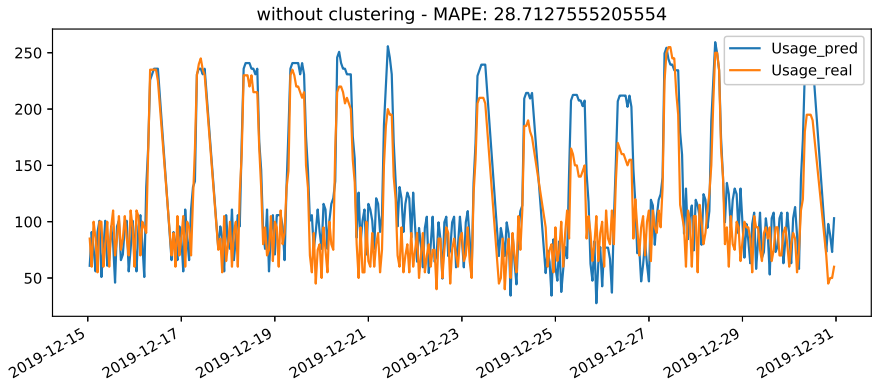

Figure 5.33: Figure shows a prediction for an industrial cluster when LSTM is trained on a single time series.

Industrial with temperature missing event

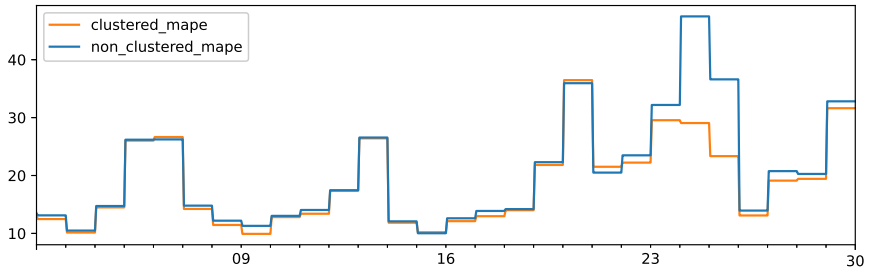

Figure 5.34: Figure shows average daily MAPE for the time series in the industrial cluster performed by LSTM when the yearly event was missing in the time series to predict.

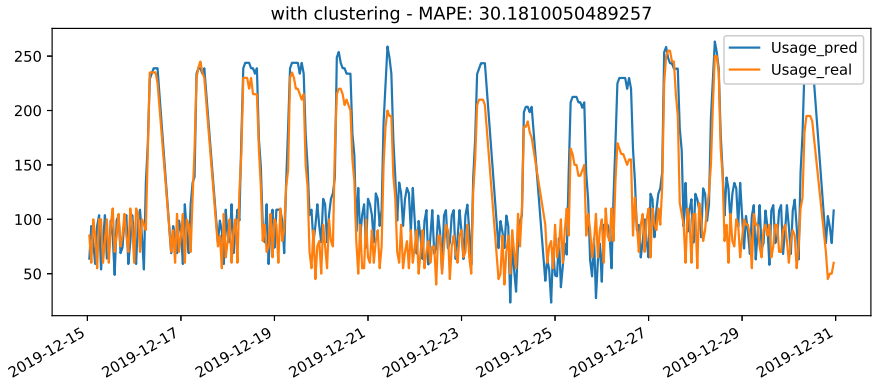

Figure 5.35: Figure shows a prediction for an industrial cluster when LSTM is trained on an additional time series when the event is missing in the original time series.

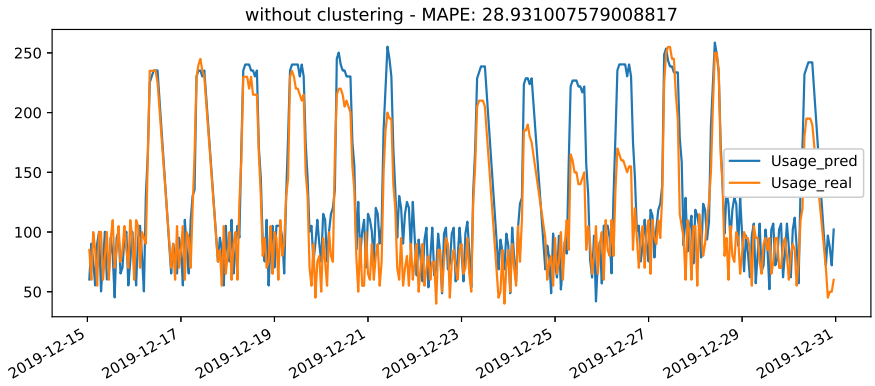

Figure 5.36: Figure shows a prediction for an industrial cluster when LSTM is trained on a single time series when the event is missing.

Residential with temperature

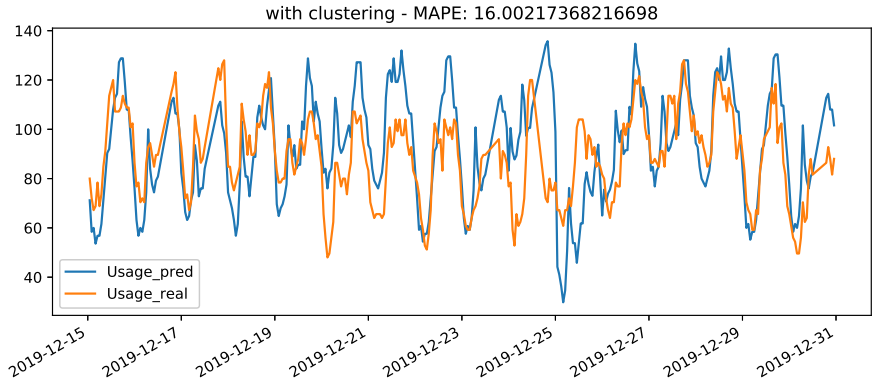

Figure 5.37: Figure shows a prediction for a residential cluster when LSTM is trained on an additional time series.

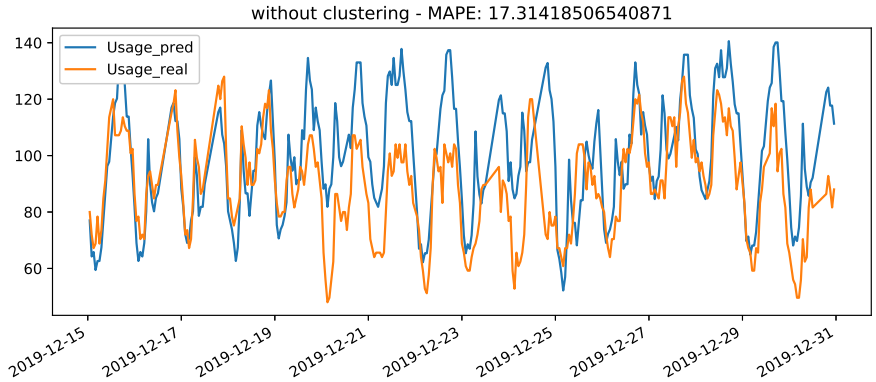

Figure 5.38: Figure shows a prediction for a residential cluster when LSTM is trained on a single time series.

Residential with temperature missing event

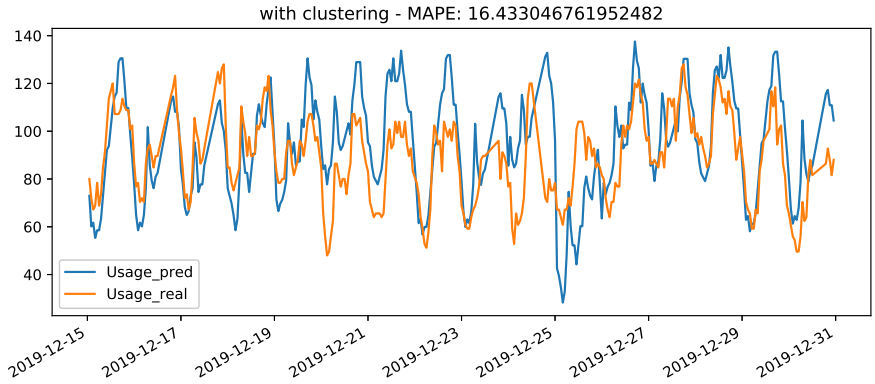

Figure 5.39: Figure shows a prediction for a residential cluster when LSTM is trained on an additional time series when the event is missing in the original time series.

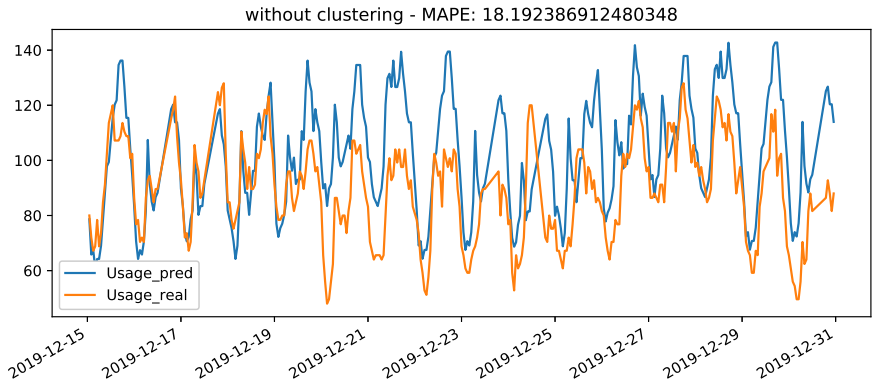

Figure 5.40: Figure shows a prediction for a residential cluster when LSTM is trained on a single time series when the event is missing.

Time series components

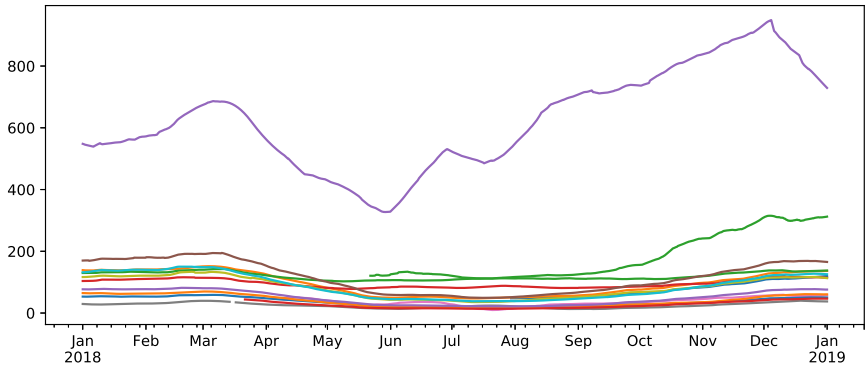

Figure 5.41: Yearly component.

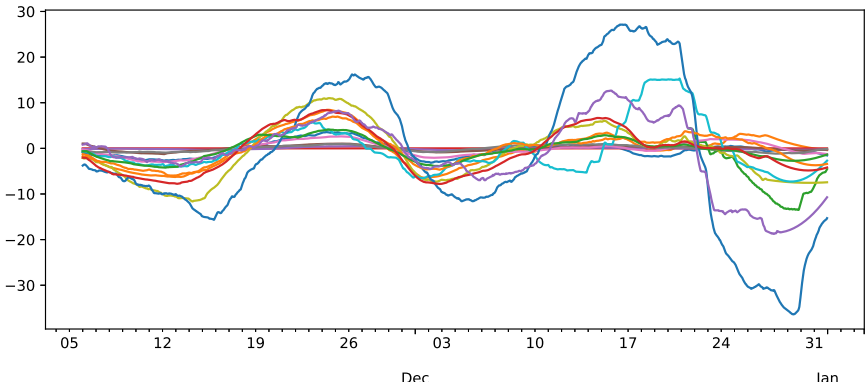

Figure 5.42: Yearly residual component.

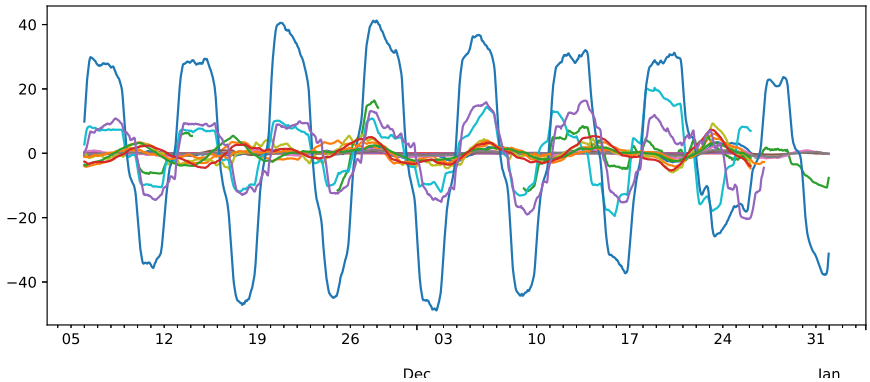

Figure 5.43: Monthly residual component.

UiA University of Agder
Master's thesis
Faculty of Engineering and Science
Department of ICT