UiA University of Agder

Master

Estimation of temperature development in secondary substation transformers

LEANDER BERG THORKILDSEN

SUPERVISOR
Ole-Christoffer Granmo

University of Agder, 2021
Faculty of Engineering and Science
Department of Engineering and Sciences

Acknowledgements

I would like to thank my supervisor Ole-Christoffer Granmo. His expertise has been highly appreciated, both before and during this thesis. Our discussions have enabled the quality of the process to be of a greater standard.

I would also like to thank Per-Oddvar Osland, Rebekka Olsson Omslandseter, and Solveig Fjellbakk from Agder Energi. Their domain knowledge and creative discussions have been tremendously helpful. Their guidance throughout the whole semester has made the project motivating to work on.

Additionally, I would like to thank my roommates and my girlfriend for supporting me and keeping me sane during these trying times.

Abstract

The temperature in a transformer element has a great impact on the aging process for a transformer. After a pilot project that aimed to make their power grid smarter, Agder Energi had temperature-data from sensors placed on some of their transformers. The purpose of this thesis was to establish a method for estimating transformer temperature based on available information such as load, capacity, and external weather conditions. To succeed, we had to deal with limited data, that is, few relevant samples combined with high levels of noise. To address the challenges of the estimation problem and the limitations of the data, we proceeded as follows. We used XGBoost on the dataset, which was resampled to put importance on the suitable samples. Feature engineering was done to improve the model. Optimization of the XGBoost parameters improved performance further. Training on 85% and testing on 15% of the data, we obtained an RMSE of 3.0. Additional results is presented, and the reliability of the results is discussed with consideration to the limitations. Agder Energi will take the project further, where it will be put into production to provide valuable insights.

Contents

Acknowledgements **i**

Abstract **ii**

List of Figures **v**

List of Tables **vi**

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Problem definition . . . 2
1.3 Limitations . . . 3
1.4 Outline . . . 4

2 Background **5**
2.1 Secondary Substation Transformers . . . 5
2.2 XGBoost . . . 6

3 Approach **7**
3.1 Data Exploration . . . 8
3.1.1 Examples . . . 9
3.1.2 Distribution . . . 11
3.2 Preprocessing . . . 14
3.3 Resampling . . . 16
3.3.1 Undersampling . . . 16
3.3.2 Oversampling . . . 17
3.3.3 Weighted-sampling . . . 17
3.4 Features . . . 18
3.5 XGBoost . . . 21
3.5.1 Parameters . . . 21
3.5.2 Optimization . . . 22
3.6 Evaluation . . . 23
3.6.1 RMSE . . . 23
3.6.2 Visual . . . 24

4 Results **25**
4.1 Resampling . . . 25
4.2 Feature Selection . . . 28
4.3 Optimization . . . 31
4.4 Comparison . . . 34

5 Discussion **37**
5.1 How reliable are the results? 37
5.2 Under what conditions does the method work? 38
5.3 How can the method be improved? 38

6 Conclusions **39**
6.1 Summary 39
6.2 Further work 39

Bibliography **40**

List of Figures

1.1 Showing three types of substations: Kiosk , Mast , and Building 2

2.1 Showing an example of a transformer. 5

3.1 Showing a visualization of the method outline. 7
3.2 Showing an example of load and capacity from an arbitrary transformer. . . 9
3.3 Showing the top and bottom temperature from sensors. 10
3.4 Showing weather-data from the same transformer in figure 3.2. 10
3.5 Showing distribution of temperatures in substation kiosk. 11
3.6 Showing distribution of temperatures in substation mast. 12
3.7 Showing distribution of temperatures in substation building. 12
3.8 Showing boxplot of temperatures in substation kiosk by month. 13
3.9 Showing the undersampled distribution of temperatures. 16
3.10 Showing the oversampled distribution of temperatures. 17
3.11 Showing the weighted-sampling distribution of temperatures. 18
3.12 Showing the importance of the starting features on the original dataset. . . . 19
3.13 Showing the feature-importance after weighted-sampling. 19
3.14 Showing the importance of lag-features up to t-9. 20
3.15 Showing the plot of temperatures for evaluation. 24
3.16 Showing the plot for visual evaluation. 24

4.1 Showing the evaluation of normal sampling. 25
4.2 Showing the evaluation of undersampling. 26
4.3 Showing the evaluation of oversampling. 26
4.4 Showing the evaluation of weighted sampling. 27
4.5 Showing the results of searching for optimal number of loadpercentage. . . . 28
4.6 Showing the results of searching for optimal number of load. 29
4.7 Showing the results of searching for optimal number of weather-features . . . 29
4.8 Showing the optimization of n_estimators and maxdepth. 31
4.9 Showing the search for the optimal learningrate with 4 different values. . . . 32
4.10 Showing the search for the optimal learningrate at 0.01 over 1000 estimators. 32
4.11 Showing the search of the optimal subsampling for all 3 parameters. 33
4.12 Showing the search of the optimal minimum weight for each child. 33
4.13 Showing estimated temperatures from the default method. 34
4.14 Showing estimated temperatures from method with resampling. 35
4.15 Showing estimated temperatures from method with added features. 35
4.16 Showing estimated temperatures from method with optimalization. 36
4.17 Showing visual evaluation of the final method. 36

List of Tables

1.1 Showing available data for (almost) all substations. 2
1.2 Showing the defined problems. 3

3.1 Showing an example of the raw dataset after renaming and removing columns. 8
3.2 Showing an example of time-shifted lag-features during preprocessing. 14
3.3 Showing available features after preprocessing. 15
3.4 Showing default, adviced, and grid-searched parameters 22

4.1 Showing the RMSE and bias of resampling. 27
4.2 Showing all features after selection. 30
4.3 Showing the final RMSE for each method added after each other. 36

Chapter 1

Introduction

1.1 Motivation

Maximum temperature in a transformer element (also known as "hotspot temperature" θh) is among the factors that have the greatest impact on a transformers aging process. Knowledge of transformer temperature is essential to assess the health status of a transformer and thus obtain decision basis for measures that optimize lifetime costs [13]. A typical measurement here can be to reduce the load on the transformer element. Alternatively, the transformer can be replaced with one of higher capacity.

Agder Energi is a power company that generates and distributes power to customers all over south-Norway[9]. In the case of a transformer breakdown, and Agder Energi fails to deliver power to their customers, an economic compensation is required to be paid to everyone affected. This requirement is enforced by the Norwegian Waterresources- and Energydirectorate (Norges vassdrags- og energidirektorat, NVE), and takes effect whenever a customer is without power for more than 12 hours. As of January 2021, the compensation will automatically be paid, and any additional downtime beyond 12 hours will be further added to the bill [28]. This does not include compensation for actual costs that customers experience as a result of power outages. The requirements is based on the laws established by NVE in 1999 and apply to all power companies in Norway [10].

A transformer is usually connected to many customers at once. A failing transformer can quickly become expensive in both compensational costs and repair costs. However, replacing the transformer too soon can become expensive as well. An optimal maintenance scheme is hard to perfect, but by knowing the estimated lifetime for a transformer, the relationship between the use of the transformer and the costs can be maximized. Until now, the lifetime of their transformers has been estimated based on the load it handles over time. Unfortunately, Agder Energi is still paying enormous amounts in compensational expenditure. The exact amount Agder Energi uses on maintenance and compensation is classified as internal information but is said to be in the tens of millions. A more accurate and reliable method of estimating a transformer's temperature would make customers more satisfied and save Agder Energi valuable time and unnecessary expenses.

1.2 Problem definition

This thesis aims to establish a method for estimating transformer temperature based on available information and suitable tools. Available information includes, among other things, load, capacity, external weather conditions, and type of transformer. Tools include python, pandas, and machine learning. The method will be tested on real cases at Agder Energi Nett.

The table below (1.1) shows the available data for (almost) all substation circuits:

Data	Description
`Load`	The amount of power a transformer handles, measured in kVAh/h. 1 hour time resolution. Data is collected directly from Agder Energi.
`Capacity`	The maximum amount of power a transformer should be handling for longer periods of time. Based on datasheet from the manufacturer. Measure in kVAh.
`Weather Data`	Various data from the respective area of the transformer. Includes precipitation, outside temperature, wind, and humidity. Delivered by the Norwegian Meteorological Institute (MET) [23].
`Type of Substation`	The type of structure surrounding the transformer. Simplified down to three types: Kiosk, mast-mounted, or inside buildings. See figure 1.1 for examples.

Table 1.1: Showing available data for (almost) all substations.

(a) Kiosk

(b) Mast

(c) Building

Figure 1.1: Showing three types of substations: Kiosk [20], Mast [21], and Building [19].

A small fraction (approximately 28) of these substations have been fitted with temperature sensors from Disruptive Technologies[32] in a pilot project that aims to make their power grid smarter[25]. The sensors are placed both below and above the transformer, as close to the core as possible. The measurements give an approximate value of the hotspot temperature and will be used as the target when training the model. Considering that setting up sensors can be costly and that Agder Energi has over 8000 substations in operation, the cheaper way of getting an approximate expected lifetime for all transformers is to use machine learning with the available data.

Some key problems have been defined with the help of Agder Energi, see table 1.2 below. These will be used to evaluate and discuss the results of the final method.

#	**Problem definition**
1	Will the method give an acceptable and reliable estimate?
2	Under what conditions does the method work, and when does it work less well?
3	What does it take to improve the method?
4	What does it take to use the method in production?

Table 1.2: Showing the defined problems.

1.3 Limitations

Number of samples
In general, a machine learning algorithm needs data that is both clean and sufficient. The data should not be mislabeled or with too much noise, and it should be of moderate-to-large amounts of samples needed for training the model. The dataset in this thesis is neither of these things. A significant limitation in the dataset is the number of samples that actually matters. As mentioned in the introduction, Agder Energi is interested in estimating the high temperatures which damage their transformers. Unfortunately, the dataset only contains about 1200 samples above 50 degrees, and even fewer samples in the range that actually matters. This amount corresponds to about 0.5% of the entire dataset.

XGboost
The main focus of this thesis will be to use domain knowledge on the available data. Considering that the dataset is lacking values that are important for Agder Energi, the method will focus mainly on the data and not the machine learning algorithm. There is no point in producing a perfect deep learning algorithm if the data is bad. XGBoost will therefore be used for its simplicity and its ability to almost always give good results.

Accuracy of data
Not only is the dataset lacking in the number of important samples, but the accuracy of the data is questionable. First, the sensors that are placed on top and bottom of the transformers are not suited for high temperatures, and they should not be in proximity of magnetic fields. All transformers and the sensors are in both of these conditions. Secondly, the capacity for all transformers may not be correct. The details of this will be further explored in the approach chapter. The inaccuracies, combined with the low amount of important data, make the task of creating a reliable method of estimation difficult. However, this is also what makes it interesting. The last chapters of the discussion and conclusion will hopefully reflect this.

1.4 Outline

This section will briefly describe how the problems and limitations were approached and form the structure of the thesis.

Background
First, some background on transformers and the hotspot temperature will be presented. Earlier tries on estimating temperatures in these transformers will be researched. XGboost will briefly be summarized, with a description of parameters and how they affect the model.

Approach
The method will analyze the dataset and describe how the dataset was cleaned and preprocessed. After this, the experiments will be described in detail, including resampling, feature selection, and optimization. The resampling will be done first to create a dataset that puts weight on the right temperatures. The feature selection will then find features that are important for estimating the right temperatures. The optimization will lastly improve the model on the selected dataset and features. How the experiments were evaluated with both a visual and a mathematical way is shown last.

Results
The results of each experiment will be shown and evaluated in separate sections. This will follow the same structure as the approach.

Discussion
The discussion will go over the problems defined in table 1.2 and the limitations from section 1.3, and discuss how well the method worked on solving the issues.

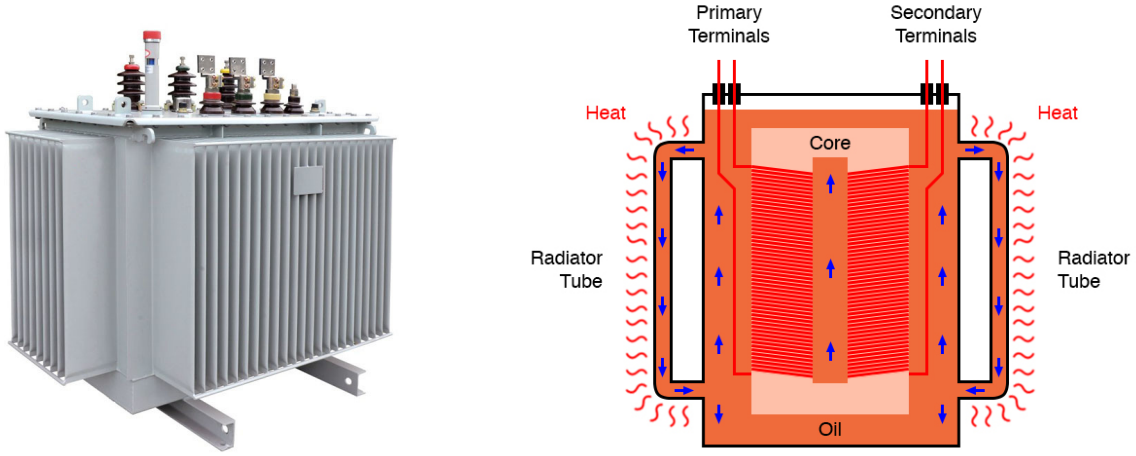
Conclusion
Lastly, the conclusion will summarize the results and discussion. The thesis finishes with a small section of advice on how to move forward with the method if it is desired to be used in production.

Chapter 2

Background

2.1 Secondary Substation Transformers

A transformer is a passive component that can convert a high voltage down to a lower voltage[35]. This conversion produces varying amounts of heat, depending on the load and capacity of the transformer, as well as the cooling system. The transformers in focus in this thesis, the ones placed in secondary substations, are oil-immersed transformers with a capacity between 50 and 1000 kvah. Figure 2.1 shows a picture of the outside, as well as a drawing of the inside of an oil-immersed transformer.



(a) Picture of a transformer[22]. (b) Drawing of a transformer[8].

Figure 2.1: Showing an example of a transformer.

The maximum temperature in a transformer element (also known as "hot-spot temperature" θh) is among the factors that have the greatest impact on a transformer aging process[29][14]. For each increase of 6°C degrees on average, the insulation in a transformer is degraded by a factor of 2[27]. An accurate estimation of the hot-spot temperature is therefore important. To minimize risk, guidelines are usually followed[18], and used as methods to crudely calculate the hot-spot. However, these estimations are inadequate[24].

A paper using a hybrid algorithm of Support Vector Regression and Information Granulation found that it consistently outperformed existing basic methods for modeling the hot-spot temperature[5]. Another paper that explored the possibility of using Artificial Neural Networks for predicting the top-oil temperature of transformers based on load found that Re-

current Neural Networks works better than static Neural Networks, getting a Mean Squared Error between 3.5 and 6, depending on the substation[17].

2.2 XGBoost

XGBoost stands for "eXtreme Gradient Boosting", and is based on Gradient Boosted Machines (GBM). GBM was first introduced by Friedman in 1999[11]. Later improved with scalability and distributed computing by Tianqi Chen[3]. It is now contributed by many developers and is being used in a wide range of applications. For example, XGBoost has been frequently used in winning Kaggle competitions in recent years. Since it will be used in this thesis, an understanding of the underlying structure is needed to optimally tune the parameters and get a good performance. This section will try to explain some of the theory behind XGBoost.

To understand XGBoost and how to tune it, we first have to look at the algorithm it uses: A gradient boosting decision tree algorithm. This has lots of names, such as Gradient Boosting, Stochastic Gradient Boosting, or Gradient Boosting Machines[2]. It is essentially an ensemble of decision trees and refers to the technique of sequentially adding decision trees (weak learners) to correct its previous learners' mistakes. The name comes from the gradient descent algorithm used to minimize the loss at each new decision tree[6]. It is built on the principle that multiple trees give a better performance than a single tree, such as in Random Forest. However, there is a fundamental difference between Random Forest and XGBoost. Random Forest uses bagging, which uses all weak learners in parallel to calculate the output based on a collective average (in regression problems). XGBoost uses boosting, which uses the decision trees in a sequence where each addition of trees tries to correct the residual errors of the previous tree. The residual errors come from the loss function. As in any supervised learning problem, we have an objective function we want to optimize. The problem is regression in this thesis, so this can be to minimize a loss function such as Mean Squared Error(MSE). But how many trees are optimal to minimize the loss while still being generalizing? The number of trees can be static, but should ideally be grid-searched and reached by early stopping when the validation-loss converges during training. To avoid making similar trees that all uses the same features again and again, randomness can be introduced to the creation of trees. This comes in the form of subsampling. XGboost supports three types of random sampling: Subsample the rows before creating each tree, subsample columns before each tree, and subsample column before each split in the tree. Additionally, the maximum size of the trees can be set to avoid over-fitting. A large number of weak learners are better than few strong learners. Learningrate should also be tweaked, as it contributes to the same principle. Generally, models that learn slowly are better than the ones that learn fast. Each additional tree should only correct its previous tree by a small amount. All these parameters are dependent on the input of data. The size or complexity of the dataset determines how the model learns.

Chapter 3

Approach

This chapter will describe how the problems and limitations were approached. The process of solving the problems was iterative. However, a description of how the problems were also approached in a structured manner will be described. A finalized structure of the chapter can be seen in the diagram in figure 3.1.

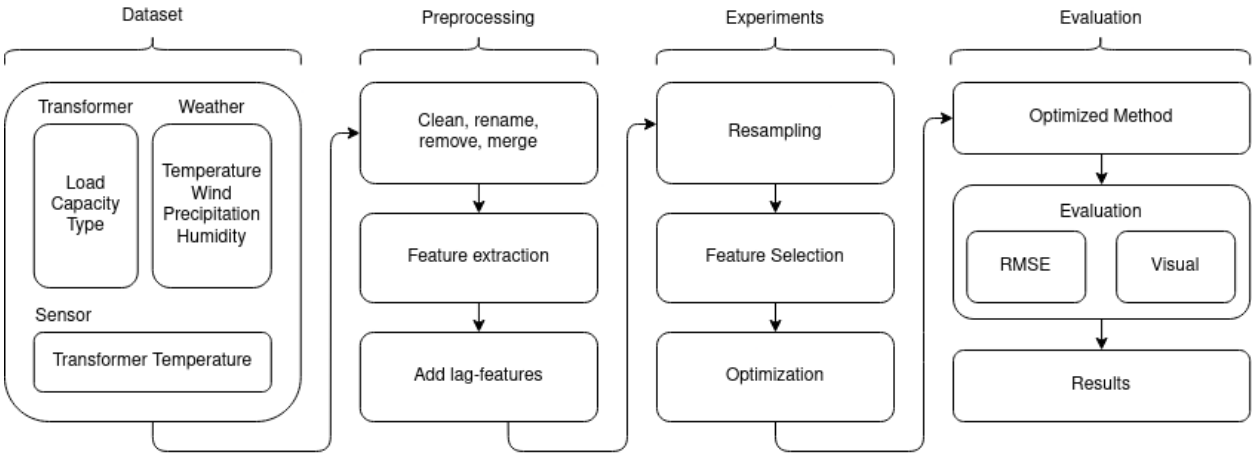


Figure 3.1: Showing a visualization of the method outline.

The dataset will first be described. This includes some plots to explore the dataset, as well as the challenges with it. The following section will be going over how the dataset was preprocessed. This will describe how some features were extracted and how lag-features were made. Next is the experiments, which were solved iteratively, but will be described sequentially. The same structure makes the results easier to present. First will the resampling be done to accommodate the lack of important samples. Then the feature selection will be done, which relies on exploration of feature-importance before and after the dataset was resampled. Only after features are selected will the optimization of XGBoost be done. The optimization section will first explain the parameters in XGBoost, then choose initial parameter-values based on advice from experts. Lastly, the evaluation method will be described, which will use both a mathematical and visual way of evaluating the results.

3.1 Data Exploration

The exploration of the data provided by Agder Energi will be described in this section. Some examples of data will be shown, including line plots to show the development of load and temperatures, histogram to show the distribution of temperatures between substations, and boxplots to show the distribution of temperature through time. Additionally, some inaccuracies and difficulties will be mentioned.

A dataset is gathered from their internal data-storage and delivered in a csv-file. As of this writing, the rows are just over 243'000. It contains several columns with data about the transformer and weather. This includes a timestamp with 1-hour resolution, and the id of the transformer. An example of rows can be seen in table 3.1. This shows the dataset after renaming some of the columns and removing unwanted ones. The Top and Bottom is the sensor-values at the top and bottom of the transformer. The value at the top is usually the highest (because of rising heat), but sometimes the bottom one is higher, so both are kept in the dataset.

Id	Timestamp	Top	Bottom	Load	Air	Prec	Humidity	Wind
17140	2020-03-27 14:00	28.5	22.8	50	8.51	0	0.75	8.43
17140	2020-03-27 15:00	28.6	22.8	50	8.89	0	0.78	8.45
17140	2020-03-27 16:00	28.6	22.8	35	9.35	0	0.75	6.90
17140	2020-03-27 17:00	28.6	22.9	25	9.63	0	0.74	5.72
17140	2020-03-27 18:00	28.6	22.9	25	8.34	0	0.79	4.33
...								
09.3830	2020-09-03 12:00	34.4	25.0	200	13.72	0.67	0.84	16.12
09.3830	2020-09-03 13:00	34.4	25.0	180	13.38	0	0.87	13.32
09.3830	2020-09-03 14:00	34.3	24.9	160	13.82	0	0.84	11.82
09.3830	2020-09-03 15:00	34.3	24.9	165	13.87	0	0.86	11.66
09.3830	2020-09-03 16:00	34.4	24.8	145	13.95	0	0.87	9.09

Table 3.1: Showing an example of the raw dataset after renaming and removing columns.

Unfortunately, the accuracy of the data is questionable. The accuracy of values from the sensors is 0.9C degrees [31], so the decimals in resolution are a bit much and will be adjusted down in the preprocessing. Also note that load is measured in intervals of 5. The sensors that are placed on top and bottom of the transformers are not suited for high temperatures. The datasheet from the manufacturer states that the maximum operating temperature is 85C degrees, and the recommended range is no higher than 50C degrees. The datasheet also states that "The sensor shall not be exposed to strong magnetic fields"[31]. Given that a transformer is basically a device for transporting energy *magnetically* between two circuits, this may be detrimental to the accuracy of the data. Additionally, the capacity for all transformers may not be correct. When a transformer is replaced, the data must also be updated to reflect this change. Given that the capacity is not a column that is automatically updated (but manually entered) a change in capacity in the middle of gathering the data may "corrupt" the dataset. Keep in mind that the task of creating a reliable method with data that is both minimal and inaccurate is difficult. The following chapters will describe what was done to try to remedy this.

3.1.1 Examples

To better understand the dataset, we will take a look at data from a transformer with high loads. There are three plots to show here: the load and capacity, the weather-features, and the transformer-temperature (top and bottom).

A section of the load and capacity from a single transformer can be seen in figure 3.2. This is a transformer placed in a ski-resort, where heavy machines such as snow-canons and ski-lifts are used sporadically. The blue line is the load, and the orange line is the recommended capacity. The ID and time-period is removed by request of Agder Energi to anonymize the data.

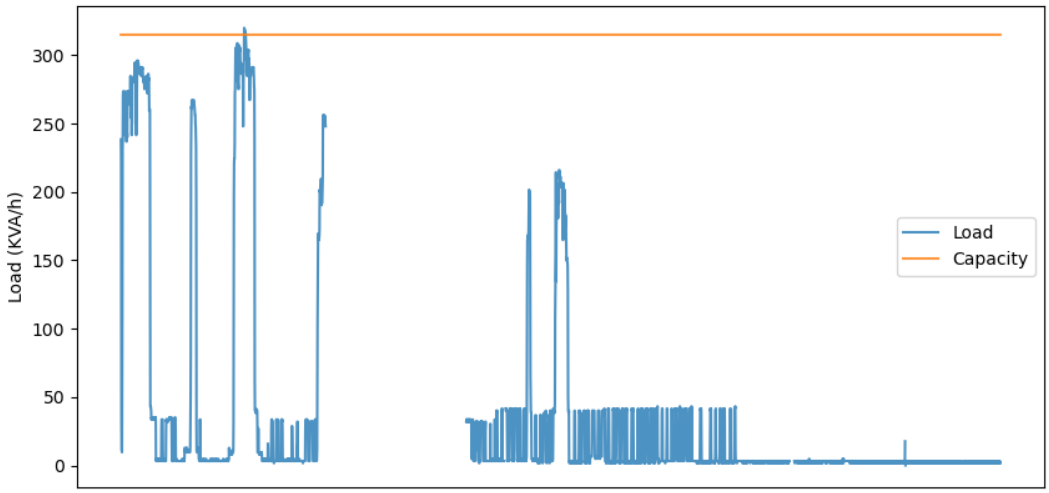


Figure 3.2: Showing an example of load and capacity from an arbitrary transformer.

As seen in the first section of the plot, it shows great variations and peaks. The same peaks can be seen correlated to the temperatures in figure 3.3. When the load in this transformer reaches its capacity, the temperature quickly rises. This is sustained as long as the load is kept up and will rapidly decrease as soon as the load stops. This fast decrease in temperature may be because of the weather outside. As seen in figure 3.4, the temperature outside is below zero, which will cool the transformer down depending on the insulation of the substation.

After a certain period, some values are missing. This may be because the sensors get broken. However, as seen in the plots with transformer-temperature and weather-data (figure 3.3 and 3.4), they are missing data as well. This makes it difficult to say why the dataset is missing values in the middle, but it is a common occurrence that will have to be handled in the preprocessing.

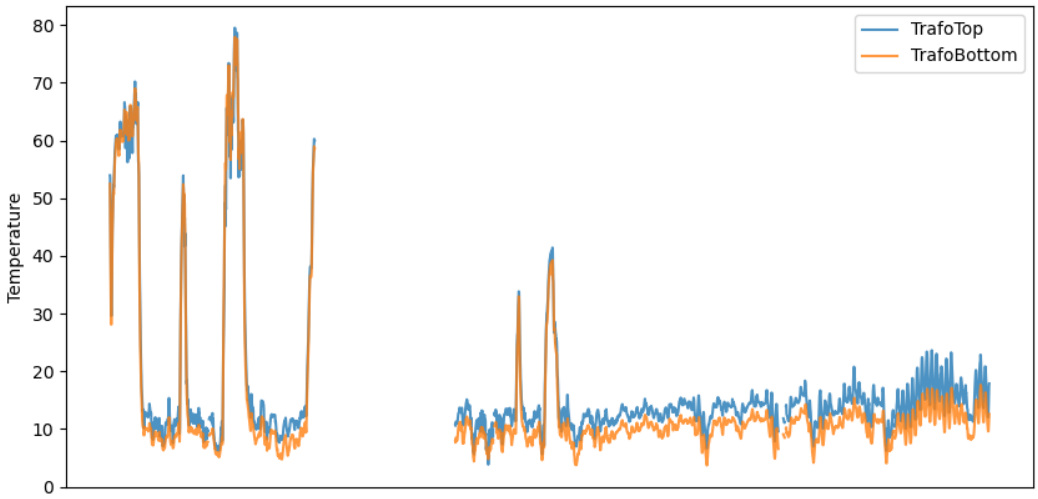


Figure 3.3: Showing the top and bottom temperature from sensors.

The weather-features can be seen in figure 3.4. Since the section is in the winter, the air temperature is mostly below zero. Precipitation is measured in millimeters, and humidity is in a percentage ratio from zero to one. The dimensions of values do not matter, as the decision trees in XGBoost do not care about normalized values when splitting nodes.

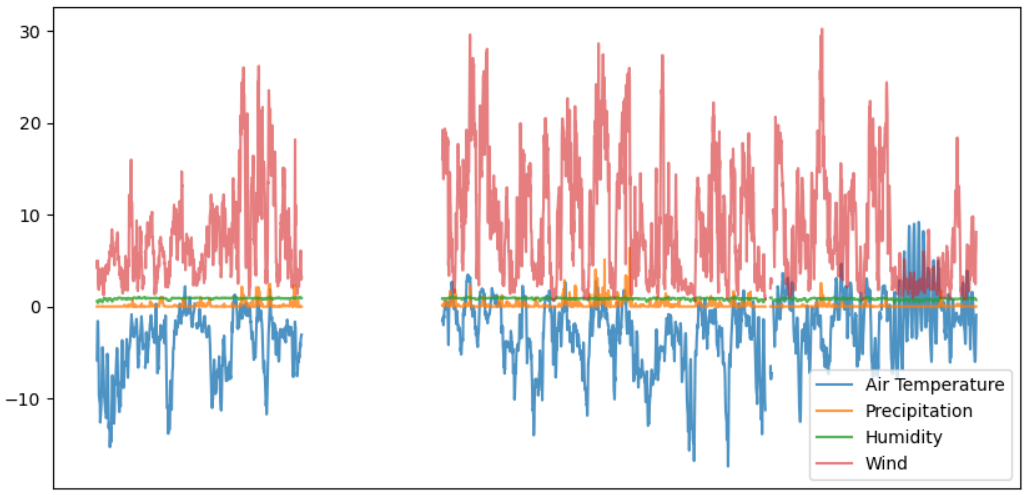


Figure 3.4: Showing weather-data from the same transformer in figure 3.2.

3.1.2 Distribution

This section will show the distribution of measured temperatures in the three different substations. Figure 3.5 shows the distribution in kiosk. The temperatures are on the x-axis, while the number of samples is on the y-axis. Kiosk has temperatures from 0 to 102 degrees, with most sampled concentrated around 20 to 30 degrees, and very few above 50 degrees. The number of samples up to 100 degrees is so low that it does not even show in the histogram. This is unfortunate, because Agder Energi is primarily interested in estimating the higher temperatures that damage the transformers. If the dataset was to be split into an upper range, say above 50 degrees, it would contain only 1250 rows. That is barely 0.5 percent of the dataset.

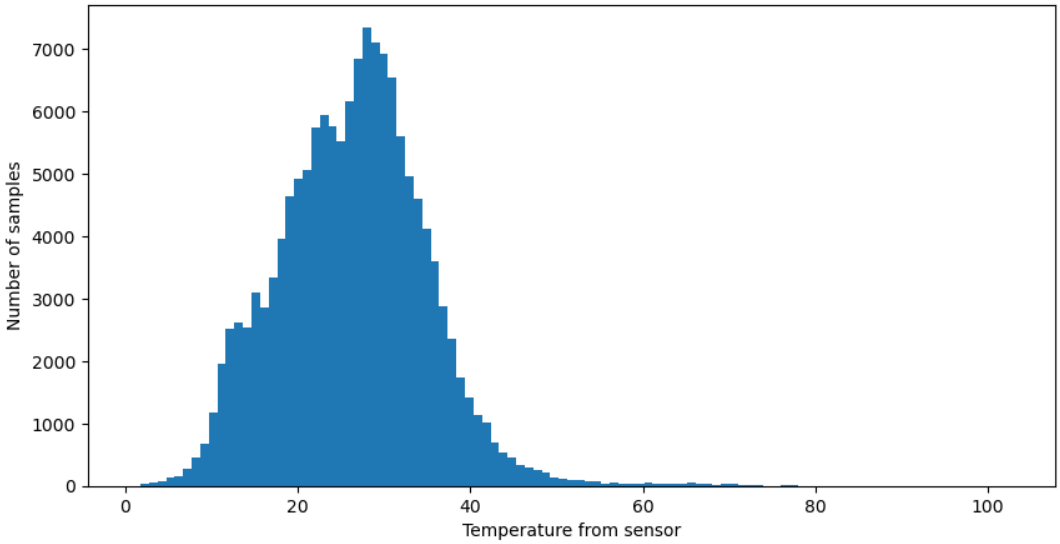


Figure 3.5: Showing distribution of temperatures in substation kiosk.

Unfortunately, further exploration shows that almost all samples with a high temperature (over 50 degrees) come from the same transformer. This is the one in the ski-resort shown as the example. Ideally, the defined problem should have been to estimate the temperature in this particular transformer. However, Agder Energi is interested in using this transformers temperatures to estimate all their other transformers. The low variety we get from only using samples from one transformer might become a problem when trying to generalize the model. Therefore, samples from other transformers in the lower range of temperatures should also be used, especially because we need them to not make the model only make high estimates.

The distribution of temperatures in the substations Mast and Building can be seen in figure 3.6 and 3.7, respectively. As we can see on the x-axis, these substations have only measured temperatures up to 40 and 50 degrees. Estimating these temperatures is probably very easy, as they have low load and most likely have a high correlation with the outside temperature. However, these temperatures are not important for Agder Energi to estimate. Therefore, the rest of the thesis will only focus on the substation Kiosk from here on. The number of sensors on these types of substations was low anyway, so the impact is not that big. Only using samples from the substation Kiosk leaves us with about 150'000 rows in the dataset.

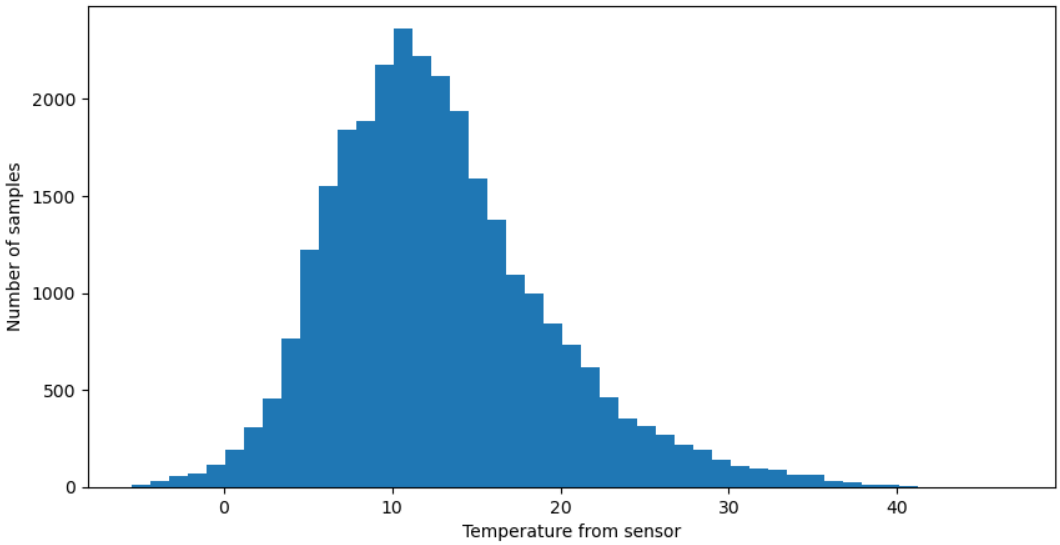


Figure 3.6: Showing distribution of temperatures in substation mast.

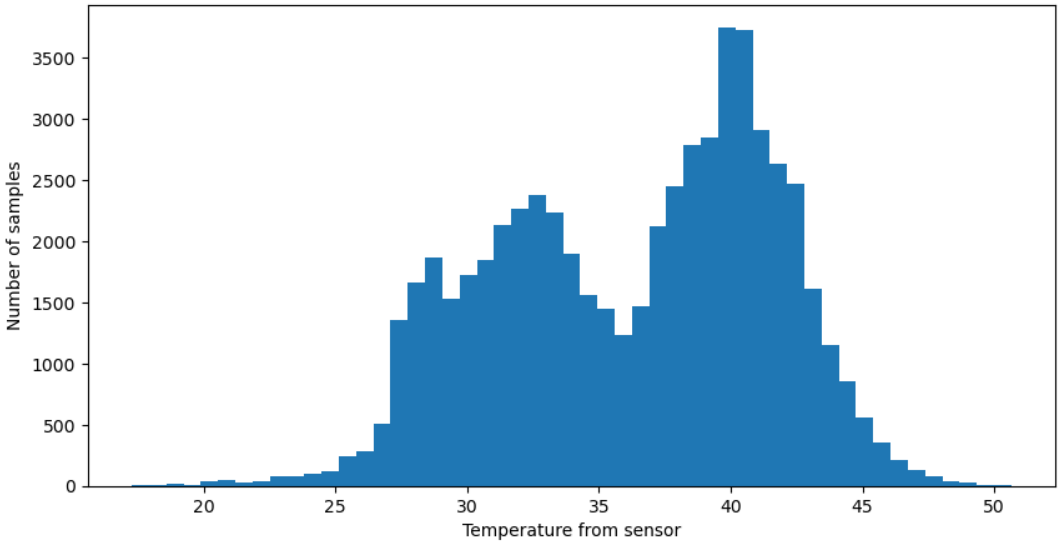


Figure 3.7: Showing distribution of temperatures in substation building.

Boxplot

Some transformers run at high load in winter, while others in summer, depending on where they are located. A transformer in a ski resort will have a high load in the winter, while a transformer in a camping site will most likely only have a high load in the summer. To better visualize the difference from month to month, the distribution of temperatures will be shown with a boxplot grouped by month. The plot in figure 3.8 shows the distribution in the kiosk substation.

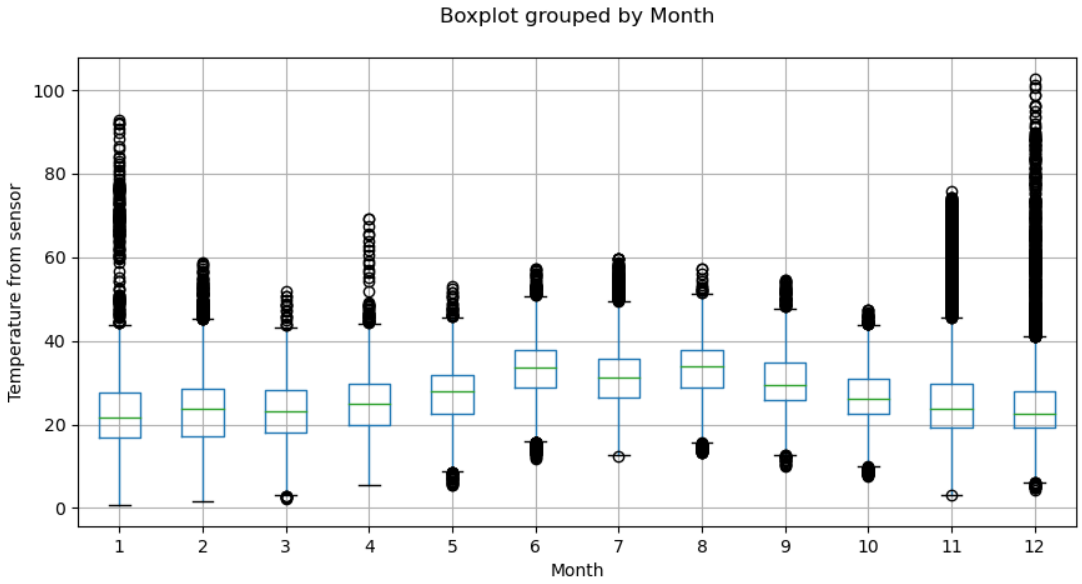


Figure 3.8: Showing boxplot of temperatures in substation kiosk by month.

The green line inside the box is the median (Q2) temperature of that month. The box itself is made up of a *lower quartile* (median of all samples below the Q2 median), and a *upper quartile* (median of all samples above the Q2 median)[34]. The whiskers from each side of the box are a set distance away from the lower and upper quartile, defined at 1.5 times the size of the box. Any values above or below the whisker are considered outliers and are drawn as a black circle.

Notice a large number of outliers in the winter months. Months are described on the x-axis as 1 (January) through 12 (December). Additionally, one can actually spot the easter-outliers in April. Since the kiosk substation is isolated inside a small cabin, the average temperature is above 20 degrees even in January. Furthermore, the average temperature increases steadily in the summer months. This indicates that most samples are based on the temperature outside. If the method were to train on estimating the average seen in the boxplot, it would probably get a great accuracy by simply guess a certain amount of degrees above the temperature outside. However, the important temperatures to estimate in this thesis is not the average, but the outliers.

3.2 Preprocessing

The dataset in this thesis is a time series. To fully utilize this type of dataset, the features should not only be of the current time-step, but several time-steps backward in time as well. The XGBoost algorithm demands a certain shape of the input (a flattened array), where each feature or column is an element in the array. This section will describe how the dataset was preprocessed to accommodate this. This includes an explanation of how lag-features were made, and why certain features were added.

For the processing, the csv-file is read with the python library pandas. Dataframes from pandas make the data easier to work with. In production, this first step will be different, but the data should still be in a dataframe. Next, the two columns Top and Bottom (temperature on top and bottom of the transformers) are merged together to create a single column; TrafoTemp. The temperature on top is not always the hottest. Therefore, the highest temperature of the two columns in each row is used. The resolution in the newly created column is then converted to integers. This makes it easier to work with later, and since the accuracy of the sensors is at 0.9C degrees, no information is essentially lost.

Normalization is a popular technique used to convert all values to a standard range. For example, the load could be resized to a value between 0 and 1. This is usually done when the algorithm relies on calculating the error with the distance between data-points. But in the case of XGBoost, this is not necessary. The split-points in the decision trees do not care about the distance. Hopefully, the algorithm will use the Load value to estimate how much heat the transformer is able to generate. Therefore, the Load feature, which can vary between zero and several hundred kvah/h, will not be normalized. However, a new feature consisting of the ratio between load and capacity will be added. This feature, which will be called the LoadPercentage, will be used to get a direct value of how much power the transformer handles in relation to its capacity. The feature is simply calculated by dividing the Load by the capacity of the respective transformer. XGBoost might be able to figure this out itself by combining Load and Capacity in its decision trees, but it's better to not leave this to chance.

Now that we have the main features, the temporal aspect can be added. By using lag-features from n hours back in time $(t - n)$, there will be n samples with Nan in 1 or more columns. An example of this can be seen in table 3.2. To avoid using samples from another sensor in the lag-features, or from skipping over time-periods when there are missing rows, every row with Nan will be removed. By making sure that the time-index is consistent, the lag-features will then be made correctly. However, this effectively reduces the size of the samples by the number of lags for each sensor. If there are 20 sensors, and each of them uses 10 time-steps as features, the total loss of samples will be 200 at minimum. Considering that there are several places in the dataset with missing values, the number of lag-features should be kept to a minimum while still using enough features to make a good estimation.

t-4	**t-3**	**t-2**	**t-1**	**t**
Nan	Nan	Nan	Nan	0.32
Nan	Nan	Nan	0.32	0.35
Nan	Nan	0.32	0.35	0.41
Nan	0.32	0.35	0.41	0.39

Table 3.2: Showing an example of time-shifted lag-features during preprocessing.

There is a couple of more features that can be extracted from the temporal component of the dataset. The timestamp can easily be converted into aggregated versions. As seen in the data exploration, there are some months that have a larger impact than others. This can be used as a feature. Month was therefore added, with values from 1 to 12 depicting January to December. Additionally, the season was extracted as another feature. This is similar to months, and it is using the months as another aggregated layer. This was mostly added for experimentation. The season of summer is from April to September, while the winter is from October to Mars. An example of the final features with 3-hour lags can be seen in table 3.3.

#	**Feature**
0	Load t-3
1	Load t-2
2	Load t-1
3	Load t
4	LoadPercentage t-3
5	LoadPercentage t-2
6	LoadPercentage t-1
7	LoadPercentage t
8	TempAir t-3
9	TempAir t-2
10	TempAir t-1
11	TempAir t
12	Precipitation t-3
13	Precipitation t-2
14	Precipitation t-1
15	Precipitation t
16	Humidity t-3
17	Humidity t-2
18	Humidity t-1
19	Humidity t
20	Wind t-3
21	Wind t-2
22	Wind t-1
23	Wind t
24	Month
25	Season
26	TempTrafo

Table 3.3: Showing available features after preprocessing.

3.3 Resampling

A common issue in machine learning is dealing with an imbalanced dataset. In this thesis, the dataset is imbalanced in a ratio of 1 to 200. One part represents the values that are important, while the other 199 parts are of values mainly in the range of 10 to 40 degrees. If the model were to train on this imbalanced dataset, the model would prioritize estimating around 30 degrees. The temperatures over 50 degrees would probably be disregarded since they are "outliers". However, the lower temperatures can not just be removed. The method must be able to estimate correctly when there actually are low temperatures as well, as to avoid false positives.

There are a few ways of resampling datasets, which will be described in the sections below. The two most common ways are removing samples from the part with most samples (undersampling), or adding samples in the underrepresented part (oversampling). There is also a third way that will be tested in this thesis. Instead of making a custom loss function that weights the higher temperatures more, the dataset can instead be resampled with more samples the higher the temperature. This will hopefully make the training naturally shift priority to the high temperatures.

3.3.1 Undersampling

Undersampling is the act of randomly removing samples from the majority to get a more balanced dataset. This type of resampling has been critiqued because it may remove important information. Moreover, it may make the performance lower simply by the fact that there is less data. Additionally, the approach of randomly removing samples without regard to heuristics is quite naive. More intelligent ways of undersampling have been proposed, such as only removing noisy or irrelevant samples from the majority[26], but this will not be implemented in this thesis. However, previous literature suggest that randomly removing samples from the majority can give good results[1], so it will be tested in the experiments. A plot of the dataset distribution after undersampling with a maximum of 1000 samples for each temperature is shown in figure 3.9.

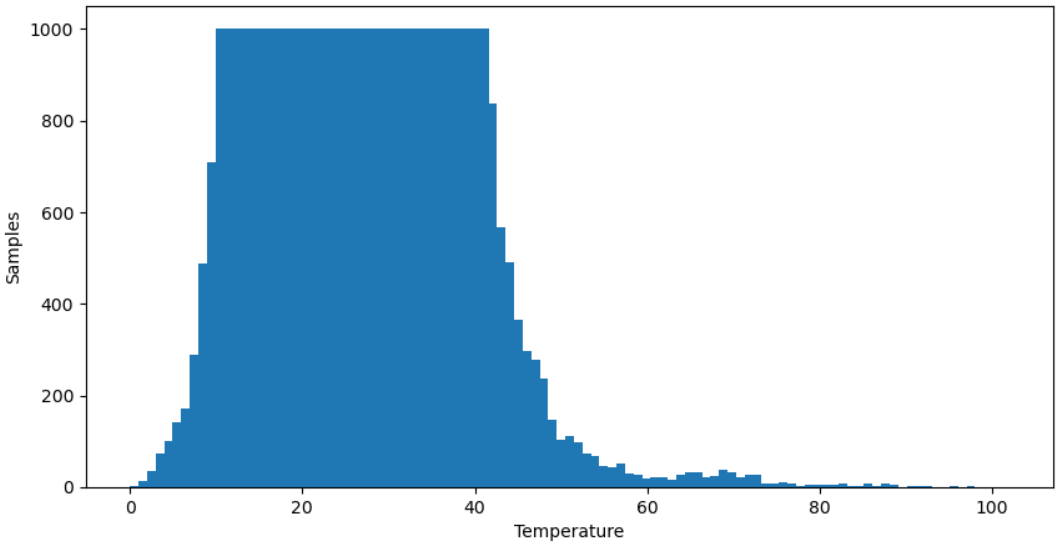


Figure 3.9: Showing the undersampled distribution of temperatures.

3.3.2 Oversampling

The dataset distribution with oversampling can be seen in figure 3.10. One thousand samples are added for each temperature over 40 degrees. With this, the minority temperatures are better represented. A study that compared several methods of balancing datasets found that random oversampling gives surprisingly competitive results[1]. However, the samples are not synthetically made or augmented; they are simply duplicated. Duplicating samples may lead to over-fitting. Because of the temporal component in the dataset, methods like Synthetic Minority Oversampling Technique (SMOTE) are challenging to implement. This will hopefully not be a problem, given that the distribution still favors the lower temperatures.

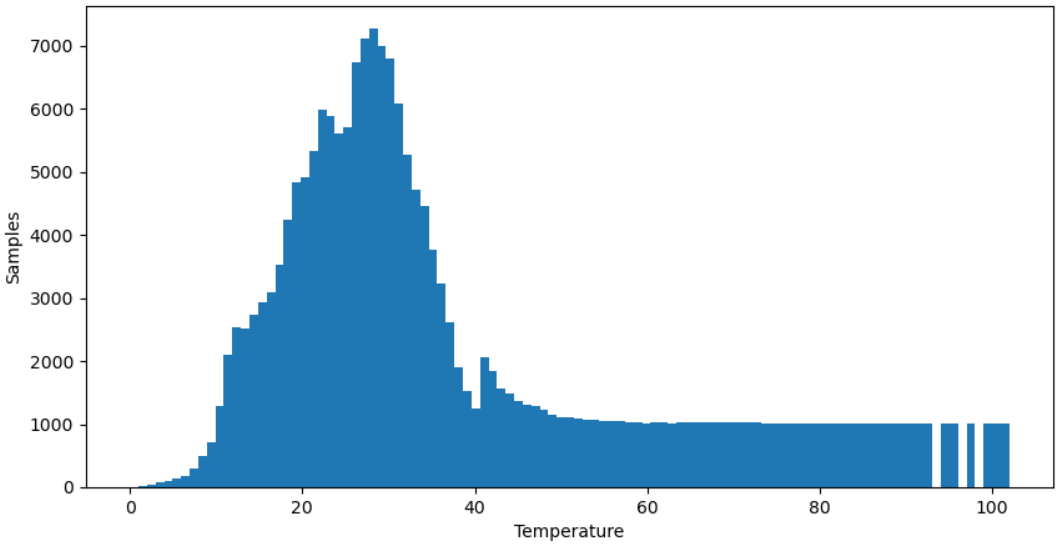


Figure 3.10: Showing the oversampled distribution of temperatures.

3.3.3 Weighted-sampling

Weighted-sampling, also called importance-sampling, is a proposed method of resampling this particular dataset based on the temperatures. The higher the temperature, the more samples the dataset gets. Because the importance of samples is known, the implementation of this type of resampling is easy. With this, there is no need to create a custom loss function. The high temperatures will naturally be favored by the model when training.

Figure 3.11 is showing the dataset distribution after weighted-sampling. Temperatures approaching zero degrees are more or less useless, so they will get the least amount of samples. Then for each new step in temperature, the number of samples increase by twenty. This continues for all 102 degrees, all the way up to 2040 samples. The three missing bars at the right in the plot are due to missing samples of temperatures for 93, 96, and 98 degrees after random resampling.

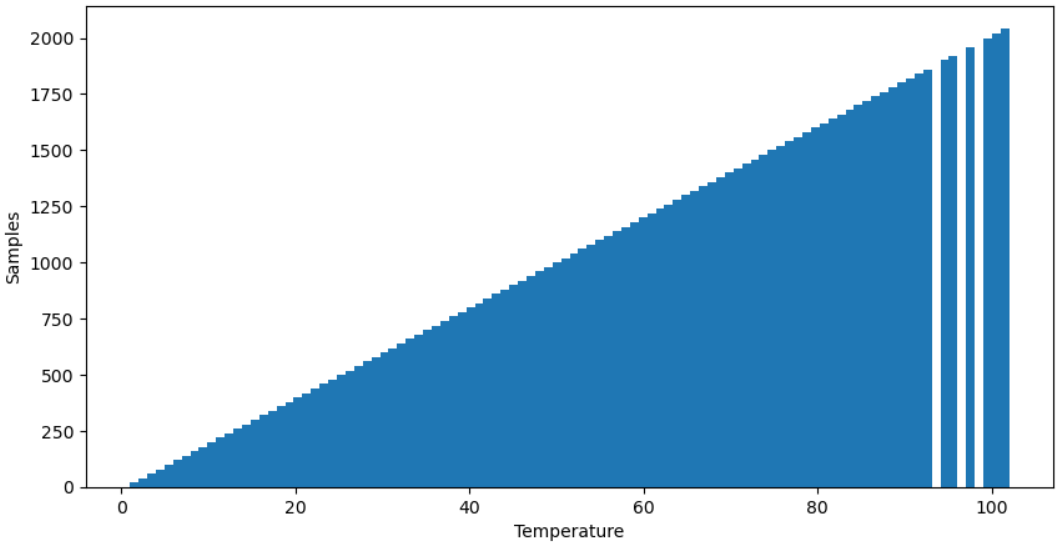


Figure 3.11: Showing the weighted-sampling distribution of temperatures.

3.4 Features

As mentioned in the introduction, an essential part of this thesis is to use domain knowledge from Agder-Energi to create a method of estimation. This section will describe the thinking behind the feature selection. It will also conduct some data exploration and discuss how the experiments were done.

Feature selection is the act of carefully selecting the relevant features for the model to use as input. In the preprocessing section, a handful of features was extracted from the dataset. Since the data is in a time series, relevant features were in this case the features as timesteps back in time. The question is: how many steps back in time is the best? Only a couple of hours back in time may not be enough to capture how the temperature has developed throughout the day. However, too many timesteps may add noise by including unnecesary features with useless information. To get an initial impression of the features and their necessities, the feature importance will be discussed. The feature importance is calculated by a built-in function in the XGBoost library. This is based on the number of times each feature is used to split a tree and improve the performance[15].

The feature importance of the original features with default sampling can be seen in figure 3.13. Here we can clearly see that the air temperature has the highest importance of all features. This is true even several steps back in time. This is not surprising given the fact that the load is mostly low in most of the dataset. Therefore, the temperature inside the substations is primarily based on the temperature outside. But this is not what we want to estimate. If we were to make a model out of this, the temperatures that are mostly dependent on the load would not be estimated correctly. To fix this, the dataset was resampled.

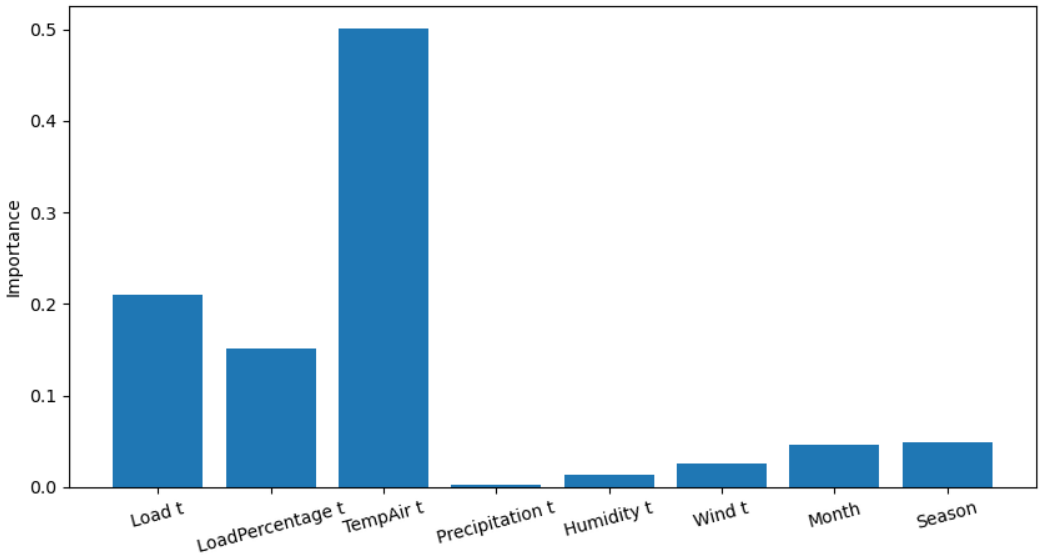


Figure 3.12: Showing the importance of the starting features on the original dataset.

The feature importance of the features after weighted sampling can be seen in figure 3.13. The fact that load is given high importance was not a surprise. LoadPercentage, the load divided by the capacity, is also given high importance. When only looking at the current time-step, both Month and Season show equal importance as load. However, the importance of just the current time-step does not show the whole picture. Since lag-features will be grid-searched, the importance of several features should also be taken into consideration.

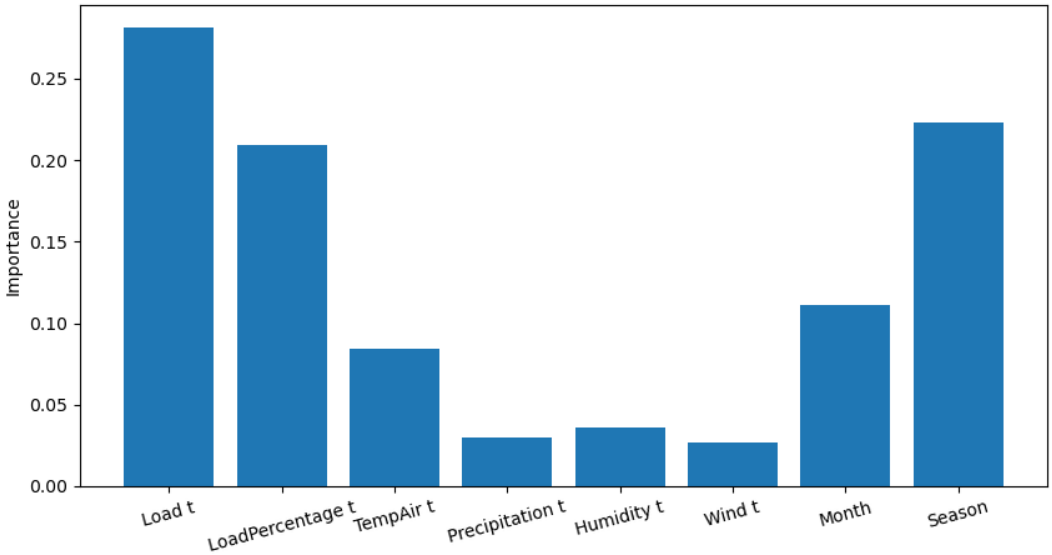


Figure 3.13: Showing the importance of the starting features after weighted-sampling.

When including more timesteps, the importance of nearly all features changes. An example can be seen in figure 3.14, where 9 lag-features are added for each feature. The LoadPercentage seems to be of most importance, especially longer back in time. The Load is no longer the dominant feature. Notice how the Month and Season seem to be insignificant in this graph. To accommodate the shift in importance, the LoadPercentage will be grid-searched first. The Load will be done after this, then the weather features. The Month and Season will be done last to see if they make a difference when all other features are added.

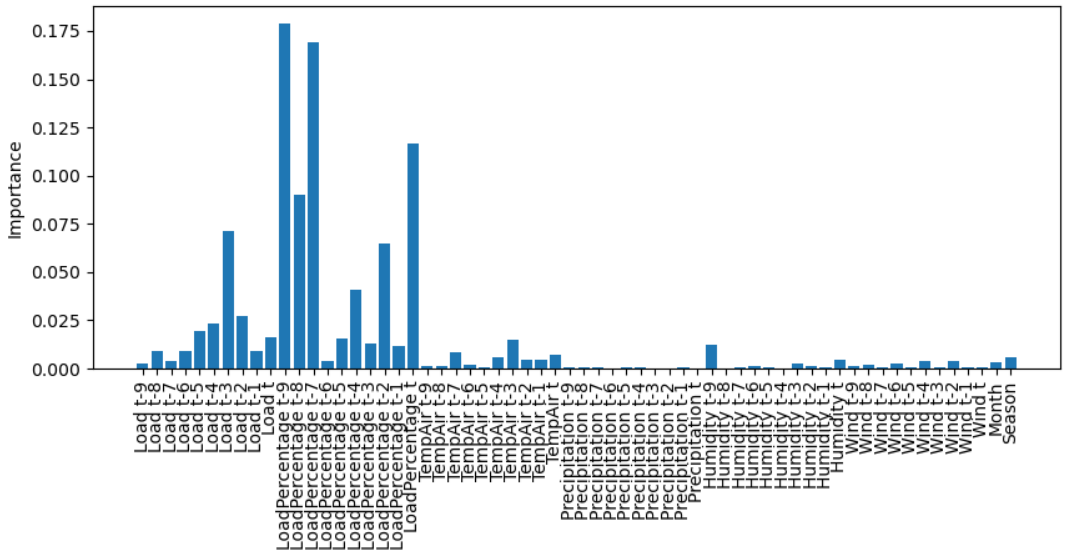


Figure 3.14: Showing the importance of lag-features up to t-9.

3.5 XGBoost

XGBoost is today used in a variety of winning solutions in Kaggle's machine learning challenges [4]. Even though it is versatile, it still has to be adapted and configured to each problem. This section will describe how the optimization of the XGBoost algorithm was done. It will start with an overview of the parameters that can be tweaked. The default parameters will be explained, and some advice from gradient boosting experts on the best parameters to start with will be presented. The rest of the section will go through each method used to search either a single parameter, or a group of parameters relative to each other.

3.5.1 Parameters

Number of trees
XGBoost is a gradient boosting algorithm consisting of decision trees. The default number of trees in the XGBoost library is 100. This is sufficient for most problems, and adding more trees does generally not increase performance beyond a certain limit. This is because of how XGBoost is constructed. As each tree added tries to correct the errors made by previous trees, the return of performance diminishes. However, the number of trees is difficult to say what is optimal on its own. There is a trade-off between the number of trees, the size of the trees, and the learningrate. Owen Zhang, a popular Kaggle competition winner, said in his talk *Winning Data Science Competitions* in NYC Data Science Academy to set the number of trees to a fixed value between 100 and 1000, depending on the size of the dataset [30]. This is in accordance with Friedmans *Greedy Function Approximation: A Gradient Boosting Machine* from 1999 [11], where he suggests the number of trees to be between 100 and 500.

Size of trees
The size of the decision trees, also called max depth or terminal nodes, determines how many nodes each tree can have at its maximum. Friedman suggests in his 1999 paper *Stochastic Gradient Boosting*[12] to set the size at 6. One might be tempted to set the size as large as possible, but if the trees become too large, it can lead to over-fitting and degrade performance. In the book *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* chapter 10 [16], it mentions that a good value of the number of nodes in the tree is 6, but can also be in the range of 4-to-8. Owen Zhang recommends to grid-search the values between 4 and 10.

Learningrate
The learningrate, also called the shrinkage parameter, controls how fast the model learns. The default value in the XGBoost library is 0.3. However, Friedman said that a learningrate of 0.1 or below usually leads to a better generalization error [12]. As mentioned earlier, there is a trade-off between the learningrate and the number of trees. Smaller values of learningrate lead to a larger number of trees. The book *Elements of statistical learning* [16] suggests to set learningrate at a small value (0.1 or less), then choose number of trees by early stopping.

Subsampling
When XGBoost adds new decision trees, it tries to correct the previous errors of the model. In this case, it tries to minimize the squared error, which is the objective function. The process, often called a greedy process, can sometimes make the selection of attributes the same. This can result in similar trees being chosen. Even the split points can be the same each time. To combat this, randomness can be introduced.

By adding randomness such as subsampling, we make sure that trees are varied and different. One of these techniques is called bagging, where a random subset of rows is used when creating trees. Another technique is used in Random Forest, where the columns (features) are randomly sampled at each split point. In the parameters of XGBoost, there are three ways of subsampling: 1. Subsampling of rows for each tree. 2. Subsampling of columns for each tree. 3. Subsampling of columns for each split in the tree. Abhishek Thakur, another one of the top Kaggle Competition winners, usually goes for a row-sampling in the range [0.5, 0.75, 1.0], or fixed at 1 [33]. Owen Zhang suggests sampling the columns in the range [0.3, 0.4, 0.5].

Minimum child weight
This parameter defines the minimum sum of weights of all observations required in a child [7]. Essentially, the larger the value, the more conservative the model will be. This can control over-fitting by preventing the model learning patterns that rarely occur. Owen Zhang uses a simplified ratio of $3/\alpha$ or $1/sqrt(\alpha)$, where α is the percentage of rare events. In our case, the rare events of high temperatures have been resampled. There is no longer just 0.5 percent of rare temperatures. A low value of min_child_weight would be preferable if the resampling was not done. However, high values may lead to under-fitting. The value will therefore be grid-searched in a large range as the last parameter to optimize.

To summarize the suggested parameters and their range, and to show the values that will be used when grid-searching other values, table 3.4 is presented:

Parameter	**Default**	**Adviced**	**Range**
Number of trees	100	100	[100..1000]
Size of trees	6	6	[4, 6, 8, 10]
Learningrate	0.3	0.1	[0.01, 0.1, 0.2, 0.3, 0.4]
Subsample	1	0.75	[0.5, 0.75, 1.0]
Columnsample by tree	1	0.4	[0.3, 0.4, 0.5]
Columnsample by level	1	0.4	[0.3, 0.4, 0.5]
Min Child Weight	1	1	[1..30]

Table 3.4: Showing default and adviced values of parameters, and the suggested range to grid-search.

3.5.2 Optimization

Number and Size of Decision Trees
As mentioned, there is a relation between the number of trees and the size of trees. A larger amount of trees would generally require a smaller depth in trees, and vice versa. This relationship can be experimented on by evaluating n_estimators against max_depth. When training the model, an evaluation dataset will score the model for each tree added. This will be done for each max_depth, and plotted in a graph. The search will be performed on up to 100 n_estimators, or up to as many necessary for the evaluation to converge.

Learningrate and Number of Decision Trees
There is also a relationship between learningrate and the number of trees. The lower the value of learningrate, the more trees are needed. This can, the same way as mentioned above, be searched by evaluating them against each other. This search will also start with up to 100 n_estimators, and add the number of trees as needed.

Sampling Subsets
The suggested values to grid-search was in the range of [0.5, 0.75, 1.0] for rows and [0.3, 0.4, 0.5] for columns. However, because of research purposes, the entire range between 0.1 and 1.0 will be searched for each subsampling method.

Minimum Child Weight
Because of the resampling, it is difficult to say have many rare events there are. The min_child_weight parameter will therefore be grid-searched in a large range.

3.6 Evaluation

Two different methods have been chosen to evaluate the experiments. The first is an error function that calculates the root-mean-squared-error (RMSE), which will be used when training and optimizing the model. This will be combined with K-fold cross-validation. The second method is a visual approach, which uses a section of high and low temperatures from a transformer with the highest loads.

Even though the dataset is a time series, the dataset is preprocessed in such a way that it can use K-fold cross-validation without samples being duplicated in other folds. This ensures that the score is not affected by how the dataset is split.

3.6.1 RMSE

RMSE is easily calculated by: $RMSE = \sqrt{(\hat{y} - y)^2}$ where $\hat{y}$ is the predicted value and y is the observed value. By adding all the errors over the whole evaluation-set, the equation becomes:

$$RMSE_{eval\_set} = \sqrt{\sum_{i=0}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

Where n is the number of observations in the evaluation-set. Since the values are rooted after being squared, the error will be the average deviation of the predicted value from the actual target-value in a relative value. This final value is the average of how many degrees it misses the target by. In other words, an RMSE of 5 would suggest that the average estimation is off by 5°C degrees.

A resampled dataset is used as a final plot to visualize what the model estimates on the whole range of temperatures. Figure 3.15 is an example of such a dataset. The values have more and more samples the higher the temperature gets. The step-up in the plot at the far right is due to missing values in the 90 degrees-range after randomly splitting the dataset.

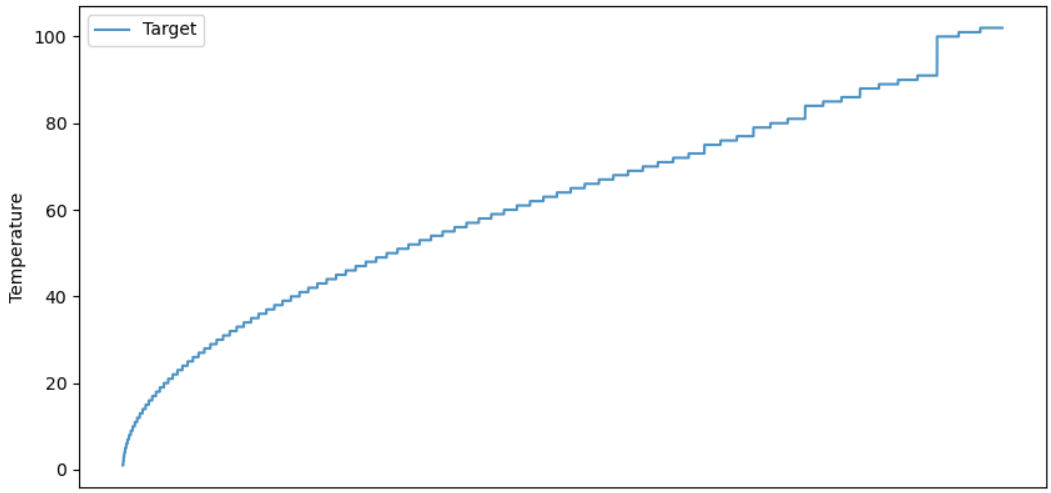


Figure 3.15: Showing the plot of temperatures for evaluation.

3.6.2 Visual

Because some temperatures are more important to estimate than others, it may be wrong to only base the evaluation on RMSE. To get a more varied evaluation, a visual approach is also used. A figure with a selection of temperatures that are thought to be a good representation of the peaks that damage the transformers can be seen in figure 3.16. This will be used when resampling and in the final comparison.

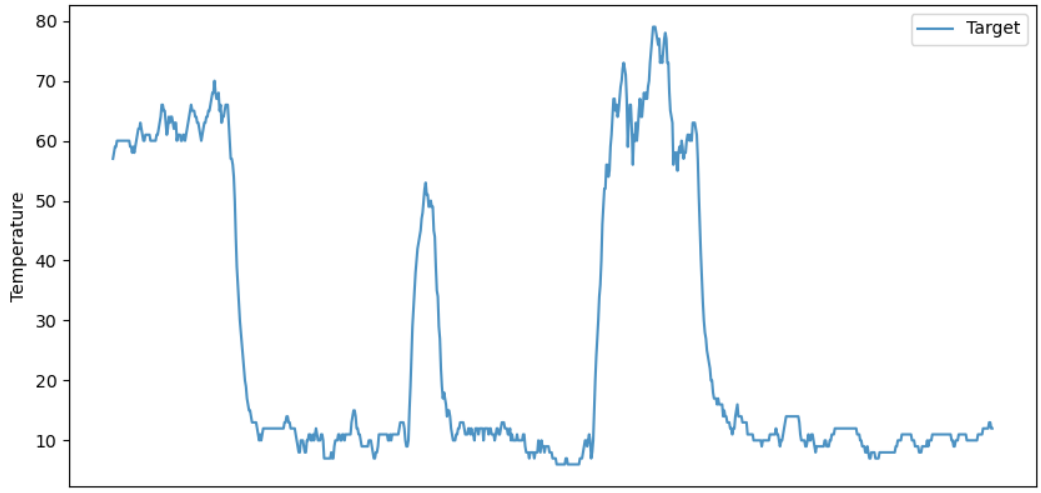


Figure 3.16: Showing the plot for visual evaluation.

Chapter 4

Results

This chapter will reveal the final results from the approach taken in the previous chapter. There are three sections that will be described separately. First, the resampling section will look at which of the types of resampling is best to move forward with. Then, with the new dataset, the results of the feature selection will be presented. Lastly, the optimization will show the optimal parameters for XGBoost. This includes a description of all plots of the parameters being grid-searched. All results are finally compared in the fourth section, where both RMSE and visual evaluation is used to show how the final method estimates over the range of all temperatures.

4.1 Resampling

Four different distributions are evaluated in this section. First, the normal dataset is used as a baseline. Figure 4.1 shows the visual evaluation on the normal distribution dataset. The blue line is the target temperature, while the orange is the estimated temperature. It seems to be struggling with estimating the first half of the left peak and high enough in the middle peak. Though, it certainly estimates well when the temperature drops and rises. The RMSE from this small part of the visual evaluation-dataset is calculated and sets a baseline of 5.959.

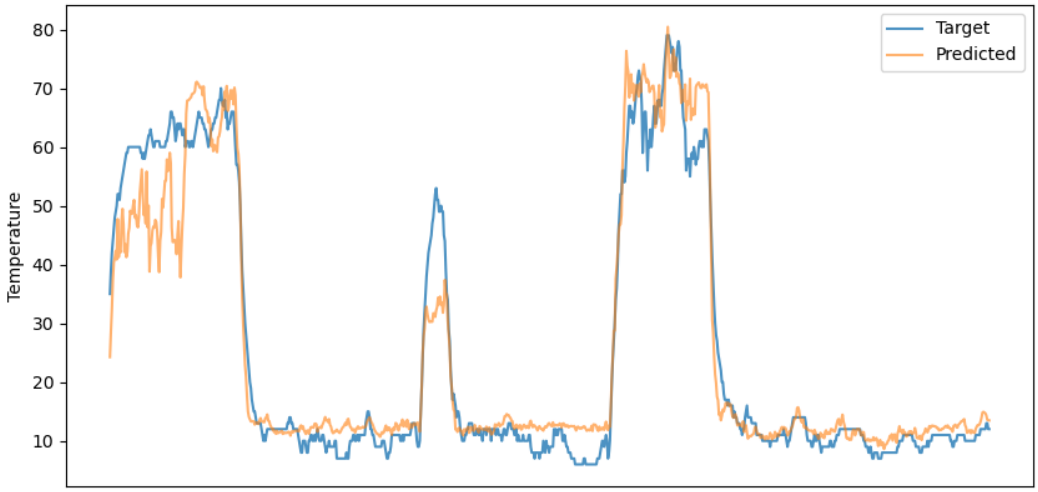


Figure 4.1: Showing the evaluation of normal sampling.

The undersampled evaluation can be seen in figure 4.2. Both visually and with RMSE, the evaluation is roughly the same as the normal dataset. The noticeable difference is how it over-estimates by 0.9 on average, where the normal distribution only over-estimated by 0.2.

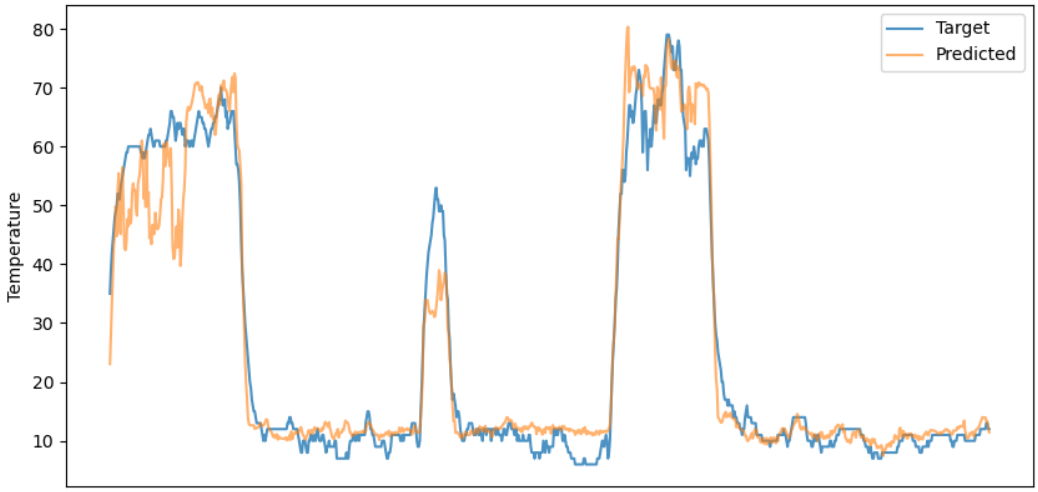


Figure 4.2: Showing the evaluation of undersampling.

The oversampled evaluation can be seen in figure 4.3. This is surprisingly not estimating any better in the middle peak. It is, however, over-estimating in the first and second peak with a calculated bias at 0.7. The RMSE is still better than undersampling at 4.812.

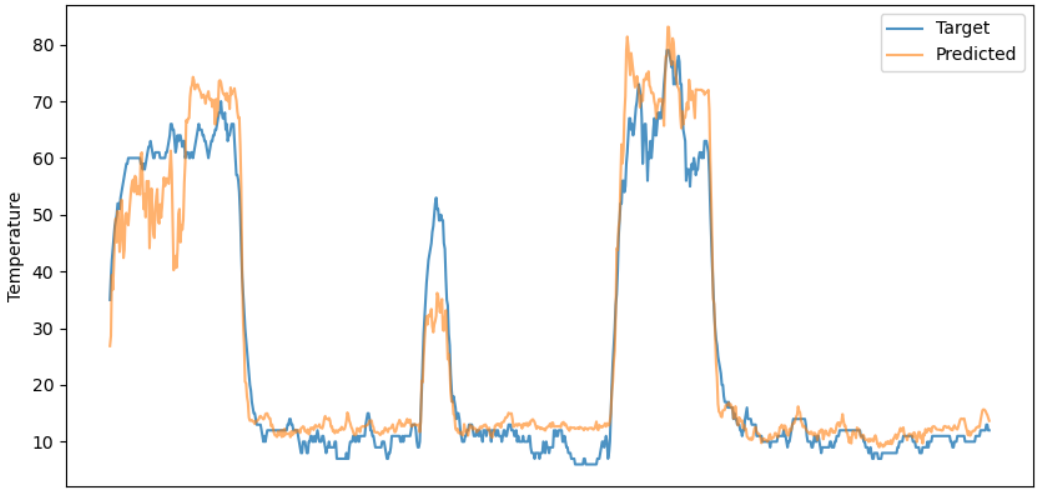


Figure 4.3: Showing the evaluation of oversampling.

The last distribution, weighted-sampling, can be seen in figure 4.4. Here can the middle peak finally be seen estimated better. The first peak is also better estimated, while the last peak slightly over-estimated. The RMSE is calculated to be 4.920, which is slightly above the oversampling. However, the estimation seems to be less biased, with 0.165 degrees over on average. Additionally, the visual evaluation seems to be that the estimation fits better on the important temperatures and peaks than the other methods of resampling. With this, the chosen method to move forward with will be the weighted-sampling. A summary of all the variants of resampling can be seen in table 4.1.

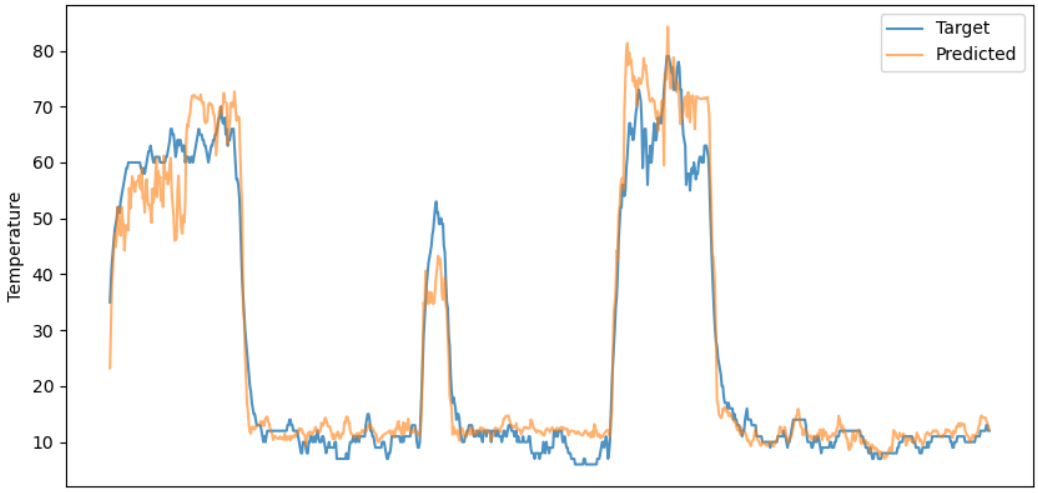


Figure 4.4: Showing the evaluation of weighted sampling.

Resampling	**RMSE**	**Bias**
Normal	5.959	0.258
Undersampling	5.335	0.916
Oversampling	4.812	0.730
Weighted-sampling	4.920	0.165

Table 4.1: Showing the RMSE and bias of resampling.

4.2 Feature Selection

This section will describe the results of the feature selection. As mentioned in the approach, the LoadPercentage will be grid-searched first. This is believed to be the main feature that is most correlated to the temperature. Next, the Load will be searched in case it is needed as a reference to the size of the transformer. After the primary features are selected, the weather features will be grid-searched. Lastly, the additional features Month and Season will be tested to see if they are needed or not.

LoadPercentage

The grid-search of LoadPercentage can be seen in figure 4.5. The y-axis is the Error in RMSE, while the x-axis is the number of features with LoadPercentage. Zero on the x-axis means no features are used. One means just the current time-step is used. Two means the current *and* 1 lag-feature is used - and so on. Not using LoadPercentage (0 features) gives a high RMSE, while up to 35 features converge with a relative low RMSE. Since a high number of features requires more samples to be disgarded in the preprocessing, and that too many features may introduce noise, 25 features of LoadPercentage was selected. This will conveniently use the current time-step and a full day (24-hours) back in time.

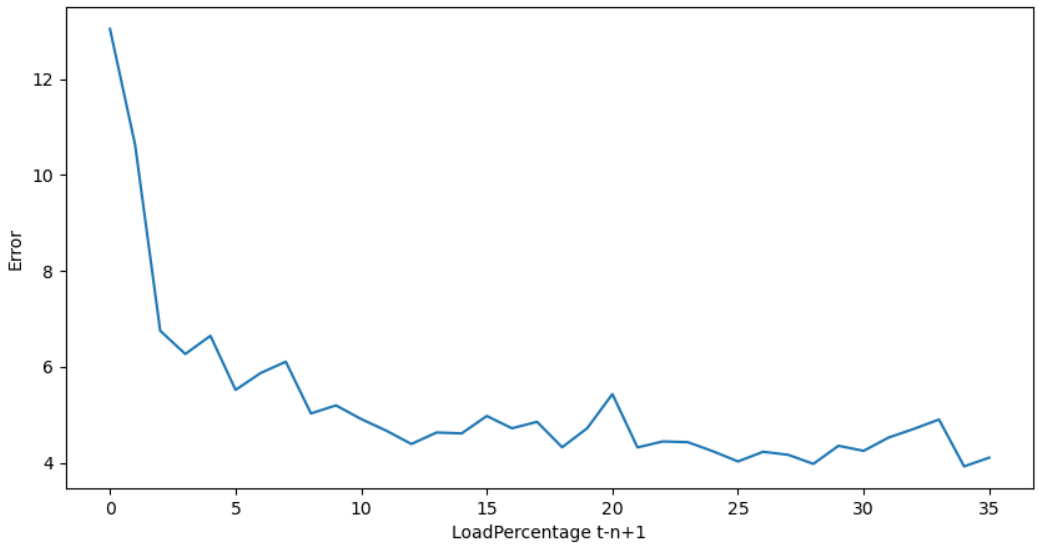


Figure 4.5: Showing the results of searching for optimal number of loadpercentage.

Load

Next, the Load feature was calculated the same way as LoadPercentage. Figure 4.6 shows the plot of grid-searching from 0 to 14 features. As suspected, only a few are needed from Load. In this case, only the current time-step gives the best results. This is a prime example of how more features can increase the error.

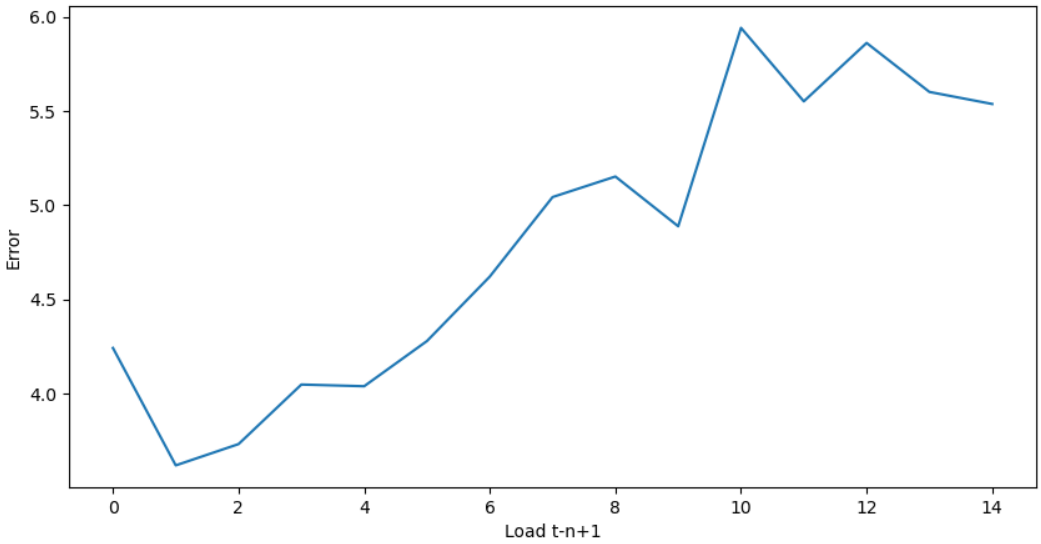


Figure 4.6: Showing the results of searching for optimal number of load.

Weather

After the primary features with the most importance is selected, the weather-features was grid-searched. The RMSE of each feature was calculated separately one by one. All four can be seen in the same plot in figure 4.7. Notice how all four seem to favor two features. Humidity is the only feature that may have a slightly smaller error when increased to 6 or 8. However, these results should be taken with a pinch of salt as they are calculated separately. Two features were therefore chosen for all four.

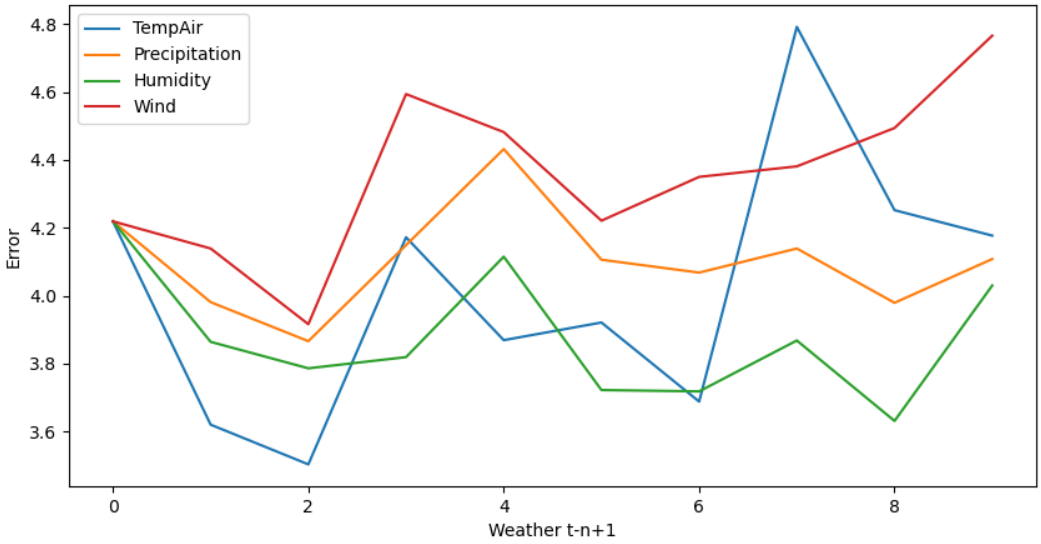


Figure 4.7: Showing the results of searching for optimal number of features from air-temperature, precipitation, humidity and wind.

Lastly, the two last features Month and Season was calculated. The RMSE shows a decrease of 0.3 when included, so they will be kept in the final selection of features. A summary of all features can be seen in table 4.2.

#	**Feature**
0	Load t
1	LoadPercentage t-24
2	LoadPercentage t-23
3	LoadPercentage t-22
4	LoadPercentage t-21
5	LoadPercentage t-20
6	LoadPercentage t-19
7	LoadPercentage t-18
8	LoadPercentage t-17
9	LoadPercentage t-16
10	LoadPercentage t-15
11	LoadPercentage t-14
12	LoadPercentage t-13
13	LoadPercentage t-12
14	LoadPercentage t-11
15	LoadPercentage t-10
16	LoadPercentage t-9
17	LoadPercentage t-8
18	LoadPercentage t-7
19	LoadPercentage t-6
20	LoadPercentage t-5
21	LoadPercentage t-4
22	LoadPercentage t-3
23	LoadPercentage t-2
24	LoadPercentage t-1
25	LoadPercentage t
26	TempAir t-1
27	TempAir t
28	Precipitation t-1
29	Precipitation t
30	Humidity t-1
31	Humidity t
32	Wind t-1
33	Wind t
34	Month
35	Season
36	TempTrafo

Table 4.2: Showing all features after selection.

4.3 Optimization

After shifting the weight to higher temperatures in resampling, and creating an input of samples to the model that is customized to estimate temperature based on features from the feature selection, the optimization of XGBoost can be done. This section will describe the results of grid-searching for the optimal parameter values in the XGBoost algorithm. This includes the number of trees, maximum tree depth, learningrate, sampling (subsampling, column sampling by tree, column sampling by level), and minimum weight of children. The starting point was not the default parameters from XGBoost, but a set of parameters by advice from several sources.

Number and Size of Decision Trees

The grid-search for the size of the trees can be seen in figure 4.8. Separate models with different maximum depth for trees are plotted after running up to 100 trees. Evidently, the model with a depth of 6 (orange) shows the lowest RMSE. This seems to be the case even after the model converges.

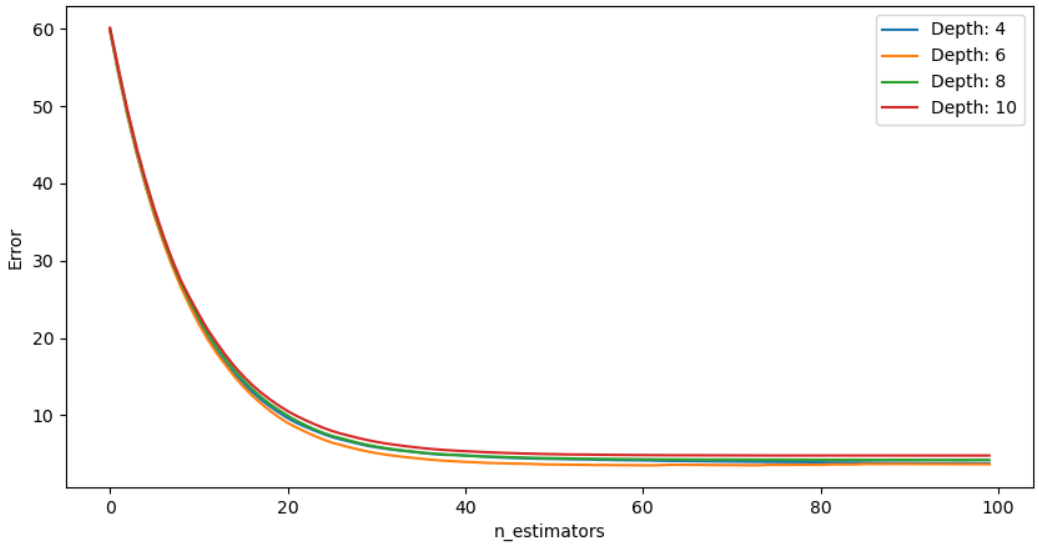


Figure 4.8: Showing the optimization of n_estimators and maxdepth.

Learning Rate and Number of Decision Trees

The search for learningrate can be seen in both figure 4.9 and 4.10. The first plot shows the three biggest learningrates are converging within 100 trees, with a learningrate of 0.2 and 0.3 are over-fitting with increasing RMSE after 50 trees. The smallest learningrate (0.01) needs more than 100 trees and is therefore shown in its own plot. The only learnigrate which did not over-fit and with the lowest RMSE overall was 0.1.

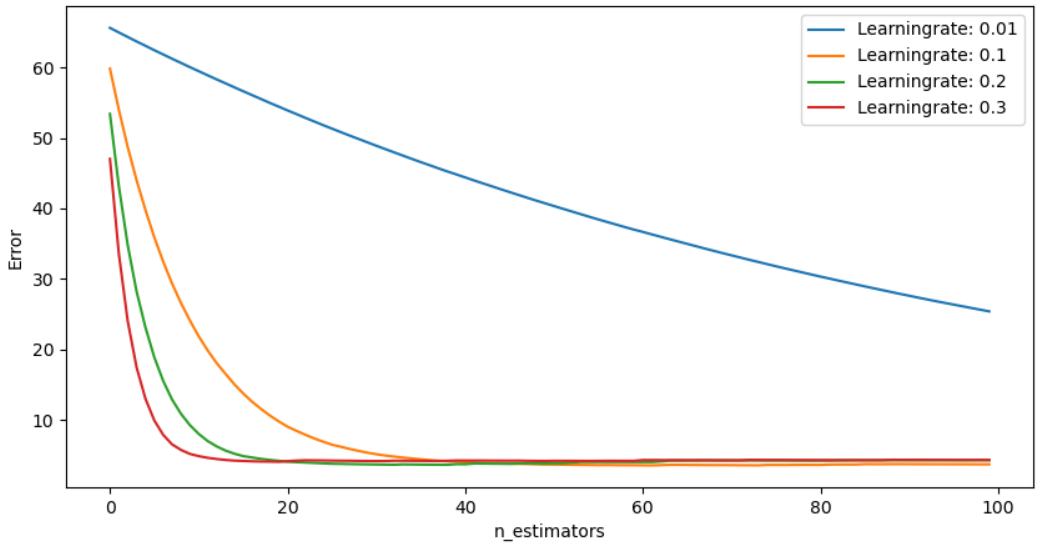


Figure 4.9: Showing the search for the optimal learningrate with 4 different values.

The own plot for the learningrate of 0.01 can be seen in figure 4.10. It takes over 900 trees until it converges. The difference between 0.1 and 0.01 was small, at 3.5 and 3.45, respectively. The smaller learningrate shows a slight improvement, but at the cost of being ten times slower. This also comes with the higher possibility of over-fitting. Early stopping was instead used with learningrate of 0.1, where it stopped at approximately 100 trees.

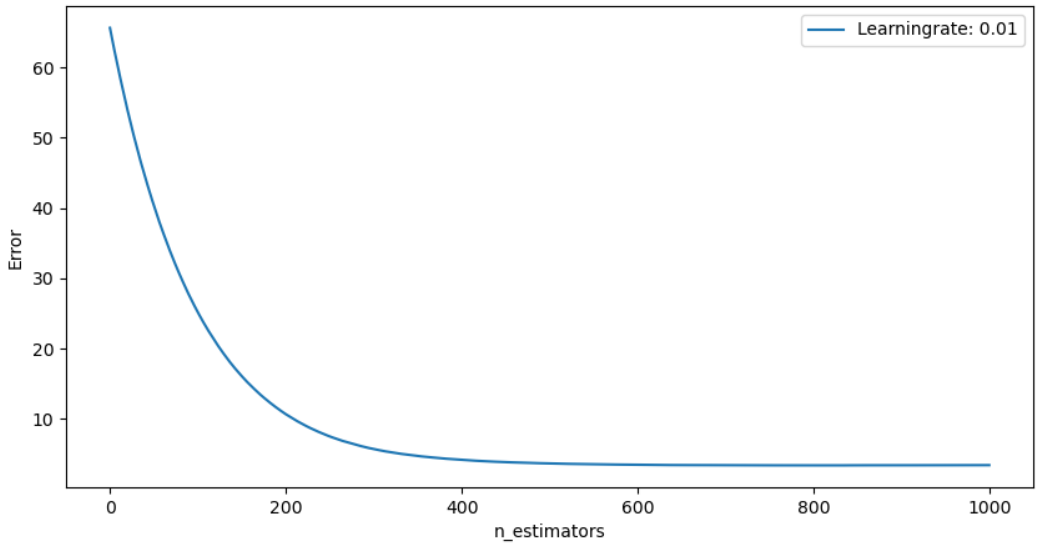


Figure 4.10: Showing the search for the optimal learningrate at 0.01 over 1000 estimators.

Subsampling Subset

The plots of the three methods of subsampling can be seen in figure 4.11. The row subsampling (blue) shows a preference for 0.9, while both parameters of column-sampling are showing the best score at 0.4.

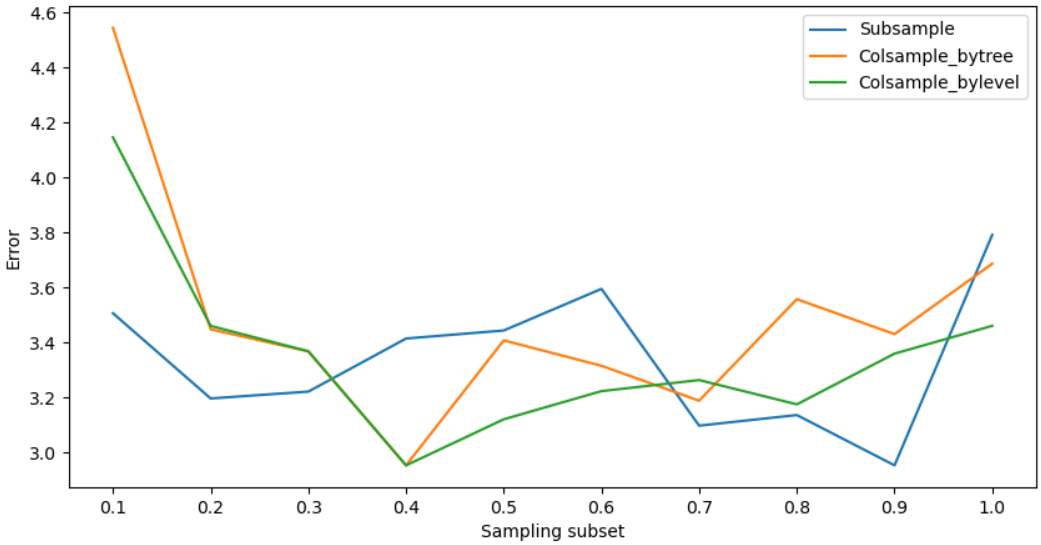


Figure 4.11: Showing the search of the optimal subsampling for all 3 parameters.

Minimum Child Weight

The search for min_child_weight can be seen in figure 4.12. The lowest error was found at 16, but 19 and 26 show almost equal RMSE. The lowest value of 16 was chosen to reduce the chance of under-fitting.

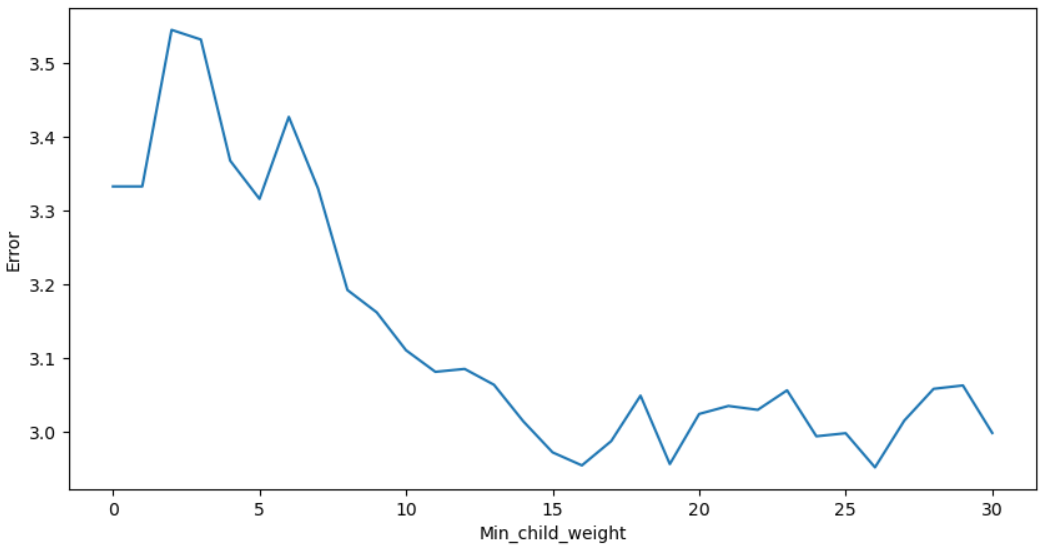


Figure 4.12: Showing the search of the optimal minimum weight for each child.

4.4 Comparison

This final comparison will look at how the different methods estimates the full range of temperatures. Figure 4.13 shows the estimation from the default method, which is without resampling, feature selection, or optimization. The y-axis shows both the target temperature (blue) and the range of estimated temperatures (orange). The x-axis is the range of temperatures from 0 to 102 described in the evaluation section. Because of the lack of samples in the upper range of the temperatures, the default method struggles to estimate over 60 degrees. Interestingly, the range with the most amount of samples (30 to 40 degrees) is also the range where it is least certain about the estimation. This might suggest that more samples are not necessarily the way to improve if the data is too varied or inaccurate.

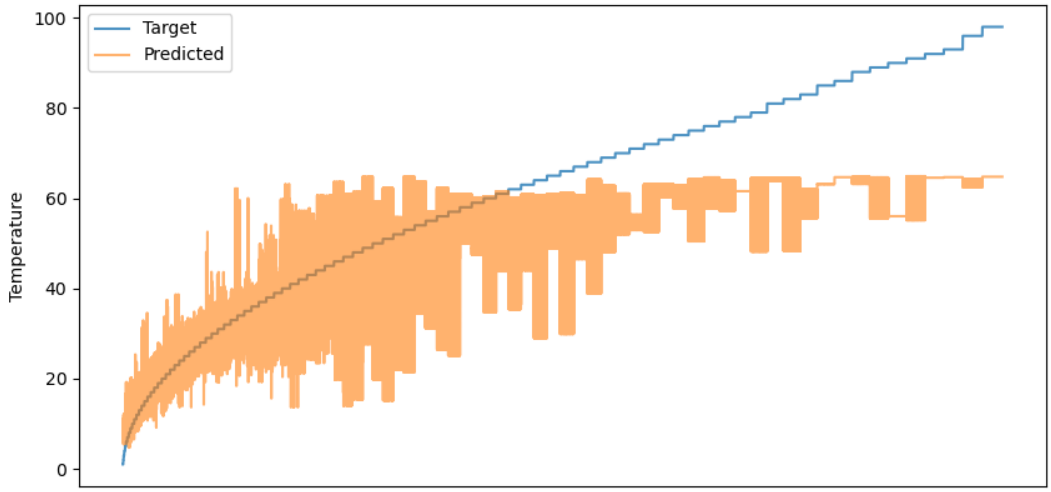


Figure 4.13: Showing estimated temperatures from the default method.

The next approach is with the weighted-resample. This can be seen in figure 4.14. After resampling, the model gets better at estimating the higher temperatures. It still estimates slightly below average. More importantly, it estimates higher in the lower range of temperatures. For example, it estimates almost 100 degrees when the target temperature is 50. It is difficult to say why it learns this by simply having more samples to learn from in the upper range.

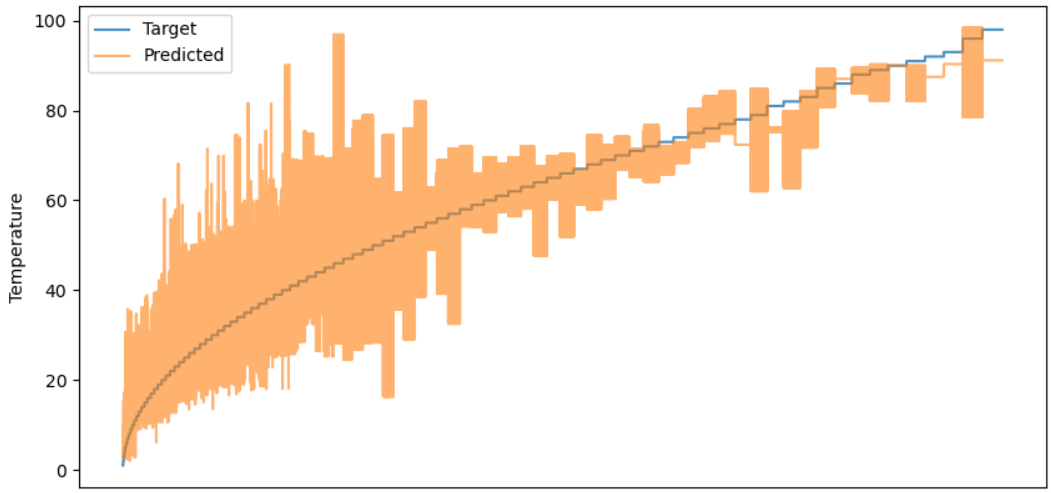


Figure 4.14: Showing estimated temperatures from method with resampling.

The estimated temperatures after feature selection can be seen in figure 4.15. Some of the high estimation in the lower range after resampling were corrected when more features was selected. Additionally, some of the uncertainty in the upper range was removed.

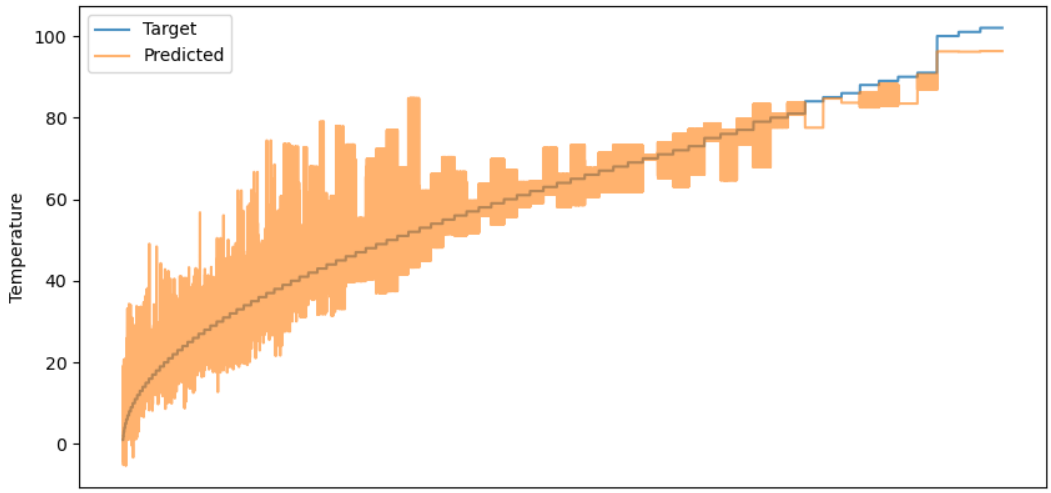


Figure 4.15: Showing estimated temperatures from method with added features.

Finally, the last method with optimization can be seen in figure 4.16. The optimization primarily lowers the uncertainty of estimations. The lower end of the temperature range is no longer estimated too much above the target.

To wrap things up, the visual evaluation is shown in figure 4.17. Here we can see that the peaks are finally being estimated (mostly) correctly. A table of errors over the different methods are summarized in table 4.3.

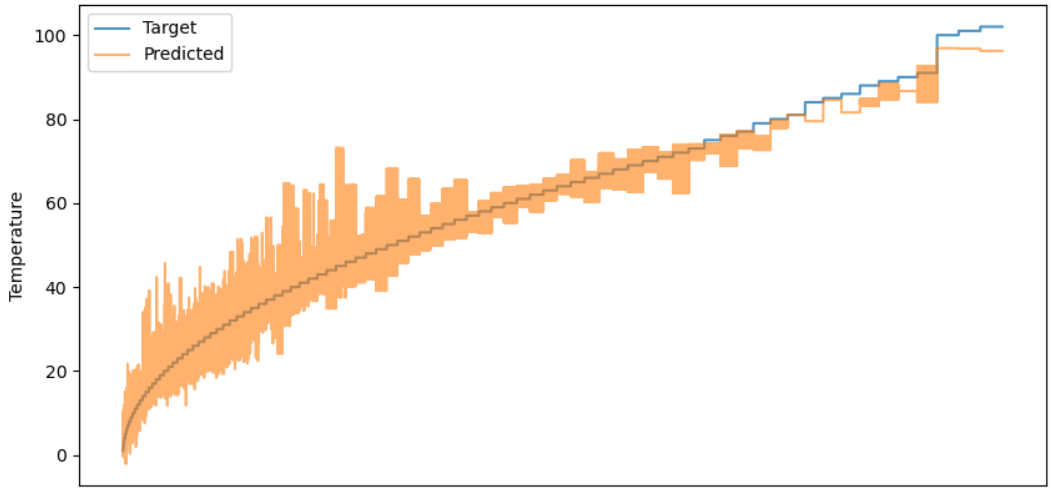


Figure 4.16: Showing estimated temperatures from method with optimalization.

Method	**RMSE**
Normal	17.207
Resampling	6.768
Feature selection	4.605
Optimization	3.405

Table 4.3: Showing the final RMSE for each method added after each other.

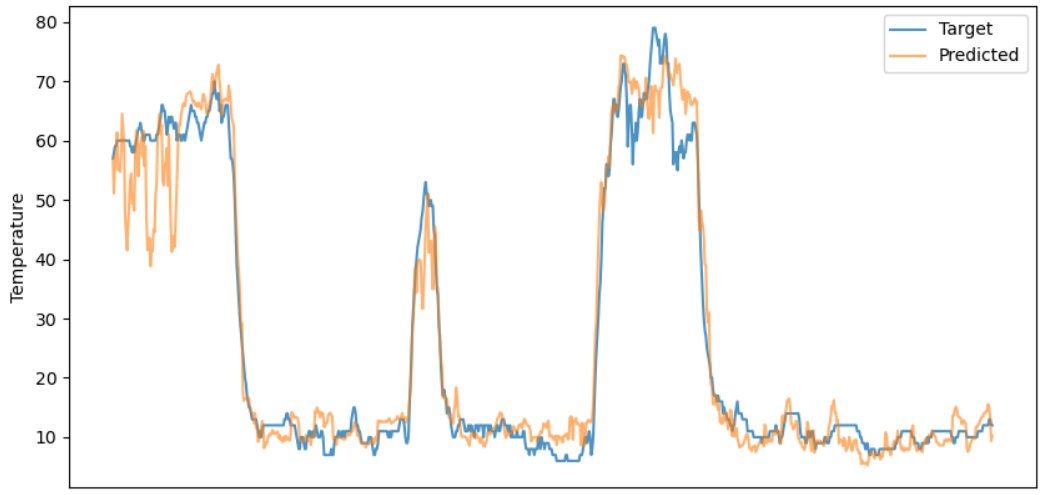


Figure 4.17: Showing visual evaluation of the final method.

Chapter 5

Discussion

This chapter will go over the problems and limitations from the introduction and discuss how well the method worked on solving the issues.

5.1 How reliable are the results?

The dataset in this thesis had a couple of problems: few important samples and some inaccuracies. Both of these problems make the results difficult to evaluate. The problem of dataset size became evident when we looked at the distribution of samples. It was eventually realized that more data with high temperatures was needed. In the results, the RMSE is calculated on 25% of the data. When increasing the training data from just 75% to 85%, the error decreases from 3.4 down to 3.0. This is a clear indication that more data will give better and more reliable results.

One of the methods chosen in this thesis was resampling. However, over-fitting on all the duplicate values may be an issue. In the results, the highest values (100 and over) seem to not be varied at all. The method still seems to be biased towards estimating lower values than the target.

The uncertainty about the accuracy of the data makes it difficult to say exactly how reliable the proposed method is. As the majority of samples tested was chosen to be from the highest temperatures, which also happens to be from mostly a single transformer, the reliability of the results is dependent on this transformer's data. A fix could be to place more sensors on critical transformers. However, the sensors may be unreliable themselves. As mentioned, the sensors are sensitive to heat and magnetic fields, both of which are in abundance when the samples of the transformer that the results rely on were made. Given how frequently the dataset is missing data, it may not be surprising if the sensors are damaged by this. Two improvements are proposed: Experiment on how reliable the sensors are when subjected to heat and magnetic fields, or use industrial sensors that can tolerate extreme conditions. The position of the sensors is not known either, other than "as close to the core as possible". The highest value between the top and bottom sensors was chosen as the target when preprocessing. In the case when the top values were missing (Nan in the pandas column), the bottom value was automatically chosen. This may have contributed to the model under-estimating since the bottom value was usually 5-10 degrees lower than the top value.

Considering the limitations, the chosen method gave acceptable results, that is, an RMSE of 3.0 under adverse conditions. However, to be more confident in the results, more diverse and a larger amount of training data is highly suggested.

5.2 Under what conditions does the method work?

Considering how most of the samples with the higher temperatures are from a single sensor, there is not much variety in the samples of this upper range. This can be problematic if one attempts to build a generic model. However, our results show that the particular sensor targetted can be modeled, providing a specialized model. Most of the method in this thesis has focused on the higher temperatures in the dataset due to their importance. And if those higher temperatures are considered a good representation of how other transformers operate, then the results can be considered good. But what if this transformer is an outlier? What if other transformers that produce high temperatures are doing so on other conditions? This brings us back to the reliability of the data, but it is also a good question to ask: Is the method conditional, that is, do we have to retrain the model for each transformer? On the one hand, the method is good by the fact that the results show certainty and low RMSE in the estimations in the upper range. On the other hand, this can be considered over-specialization for the particular sensor. This means that the method should be evaluated on more sensors as further work to uncover its ability to generalize across sensors.

5.3 How can the method be improved?

There are several things that could have been done to improve the method. The dataset is already discussed, but the three main approaches - resampling, feature engineering, and optimization - could also be improved upon. First, the resampling could have been experimented on more. For example: How many samples are necessary for a reliable result, and how are the results affected by varying the number of samples? Additionally, the effects of duplicating samples are unknown. How much duplication is too much, and when does over-fitting become a problem?

When it comes to features, additional work on feature engineering could potentially improve the method. The fact that the extracted features Month and Season made a difference at all was surprising. The expectation was that the temperature outside was enough to know what season it was. This may suggest that extracting additional features based on already known data could improve performance further. What type of features this could be, or how they would affect the results, are not known. Additionally, there are possibilities of getting more features from Agder Energi. For example, the type of transformer (and not just the surroundings of the substation) or the type of cooling/insulation in each transformer could become impactful features. Furthermore, the resolution can be higher. The sensors are sampling in intervals per hour, but it should be possible to get a higher interval such as per minute. This would make the dataset bigger and give us more samples in the upper range.

Lastly, the model itself could be improved. This could be by either optimizing XGboost further or by replacing the model with a more advanced deep learning model. For example, the learningrate could be grid-searched in the smaller range below 0.1. The number of trees will then have to be increased, which will significantly increase training time, but there may be some extra optimization to squeeze out. Deeper and more complex models could also be experimented with. However, making several complex and specialized models to compare is time-consuming, and the lack of data would still be a problem that a different model may not solve.

Chapter 6

Conclusions

6.1 Summary

The temperature in a transformer element has a great impact on the aging process for a transformer. After a pilot project that aimed to make their power grid smarter, Agder Energi had temperature data from sensors placed on some of their transformers. The purpose of this thesis was to establish a method for estimating transformer temperature based on available information such as load, capacity, and external weather conditions. Considering the limitations, a method of three parts was accomplished: Resampling was done to accommodate the lack of important samples, feature engineering was explored to find lag-features and extractions that best correlated to the highest temperatures, and optimization of parameters in XGBoost was done. The results show that the final method makes good estimations in the whole range of temperatures. However, this may be due to the low variety in the dataset. Reliability can be increased with more data.

6.2 Further work

Before using the final method in production, there are a couple of things that can be worked on. As mentioned in the discussion, the dataset should be improved. This can be done in several ways: More sensors at critical transformers, better sensors that can handle extreme conditions, higher resolution when gathering sensor-data, and more features about the conditions surrounding the transformers.

If the dataset gets improved, the method can also be altered. All parts of the experiments in this thesis (resampling, feature selection, and optimization) can be done again, as they are highly customized to the dataset. Suggestions were mentioned in the discussion. As this thesis only focused on one substation (kiosk), the other types of substations will also have to be taken into consideration. This can be done as a feature. However, separate methods should be considered since the substations have different characteristics that influence the temperature.

Additionally, the method needs some further work before it is put into production. As the measured temperature is not the same as the temperature inside the transformer element, the hotspot temperature can be calculated after an estimate is made. After delivery, a presentation will be held for the employees and developers in Agder Energi to help them put the method into operation.

Bibliography

[1] Gustavo E. A. P. A. Batista, Ronaldo C. Prati, and Maria Carolina Monard. "A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data." In: *SIGKDD Explor. Newsl.* 6.1 (June 2004), pp. 20–29. ISSN: 1931-0145. DOI: 10.1145/1007730.1007735. URL: https://doi.org/10.1145/1007730.1007735.

[2] Jason Brownlee. *A gentle introduction to XGBoost for applied machine learning*. URL: https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/. (accessed: 31.05.2021).

[3] 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*. KDD '16. San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 785–794. ISBN: 9781450342322. DOI: 10.1145/2939672.2939785. URL: https://doi.org/10.1145/2939672.2939785.

[4] Distributed (Deep) Machine Learning Community. *Awesome XGBoost, Machine Learning Challenge Winning Solutions*. URL: https://github.com/dmlc/xgboost/blob/master/demo/README.md#machine-learning-challenge-winning-solutions. (accessed: 22.05.2021).

[5] Yi Cui, Hui Ma, and Tapan Saha. "Transformer hot spot temperature prediction using a hybrid algorithm of support vector regression and information granulation." In: *2015 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC)*. 2015, pp. 1–5. DOI: 10.1109/APPEEC.2015.7381066.

[6] XGBoost Developers. *Introduction to Boosted Trees*. URL: https://xgboost.readthedocs.io/en/latest/tutorials/model.html. (accessed: 31.05.2021).

[7] XGBoost Developers. *XGBoost Parameters*. URL: https://xgboost.readthedocs.io/en/latest/parameter.html. (accessed: 22.05.2021).

[8] *Drawing of oil-immersed transformer, technocrazed.com*. URL: https://www.technocrazed.com/9-8-practical-considerations-transformers.

[9] Agder Energi. *Agder Energi Website*. URL: https://www.ae.no/. (accessed: 11.05.2021).

[10] Norges vassdrags- og energidirektorat. *Forskrift om økonomisk og teknisk rapportering, inntektsramme for nettvirksomheten og tariffer*. URL: https://lovdata.no/dokument/SF/forskrift/1999-03-11-302/KAPITTEL_4-4%20-%20KAPITTEL_4-4#KAPITTEL_4-4%20-%20KAPITTEL_4-4. (accessed: 11.05.2021).

[11] Jerome Friedman. "Greedy Function Approximation: A Gradient Boosting Machine." In: *The Annals of Statistics* 29 (Nov. 2000). DOI: 10.1214/aos/1013203451.

[12] JH Friedman. *Stochastic gradient boosting. Department of Statistics*. Tech. rep. Stanford University, Technical Report, San Francisco, CA, 1999.

[13] Radu Godina et al. "Effect of Loads and Other Key Factors on Oil-Transformer Ageing: Sustainability Benefits and Challenges." In: *Energies* 8 (Oct. 2015), pp. 12147–12186. DOI: 10.3390/en81012147.

[14] Radu Godina et al. "Effect of Loads and Other Key Factors on Oil-Transformer Ageing: Sustainability Benefits and Challenges." In: *Energies* 8.10 (Oct. 2015), pp. 12147–12186. ISSN: 1996-1073. DOI: 10.3390/en81012147. URL: http://dx.doi.org/10.3390/en81012147.

[15] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. "The Elements of Statistical Learning: Data Mining, Inference, and Prediction." In: Springer. Chap. Relative Importance of Predictor Variables.

[16] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. "The Elements of Statistical Learning: Data Mining, Inference, and Prediction." In: Springer. Chap. 10 Boosting and Additive Trees.

[17] Q. He, J. Si, and D.J. Tylavsky. "Prediction of top-oil temperature for transformers using neural networks." In: *IEEE Transactions on Power Delivery* 15.4 (2000), pp. 1205–1211. DOI: 10.1109/61.891504.

[18] "IEEE Guide for Loading Mineral-Oil-Immersed Transformers and Step-Voltage Regulators." In: *IEEE Std C57.91-2011 (Revision of IEEE Std C57.91-1995)* (2012), pp. 1–123. DOI: 10.1109/IEEESTD.2012.6166928.

[19] *Image of bygg, dagsavisen.* URL: https://www.dagsavisen.no/rogalandsavis/nyheter/stavanger/2015/09/16/lyse-elnett-fikk-15-millioner-fra-enova/.

[20] *Image of kiosk, ABB nettstasjon.* URL: https://nnsn.geo.uib.no/eworkshop/uploads/Main/ABB-nettstasjon-a.jpg.

[21] *Image of mast, digitalmuseumet.* URL: https://digitaltmuseum.no/011012596169/hoyspentmaster-i-akershus-hoyspentmast-ved-parkeringsplass-ved-aros-bru/media?slide=0.

[22] *Image of oil-immersed transformer, electrical-power-transformer.com.* URL: http://www.electrical-power-transformer.com/photo/ps30565974-fully_sealed_oil_immersed_transformer_10kv_core_type_laminated_energy_saving.jpg.

[23] Norwegian Meteorological Institute. *Free meteorological data.* URL: https://www.met.no/en/free-meteorological-data.

[24] Mohd Taufiq Isha and Zhongdong Wang. "Transformer hotspot temperature calculation using IEEE loading guide." In: *2008 International Conference on Condition Monitoring and Diagnosis*. 2008, pp. 1017–1020. DOI: 10.1109/CMD.2008.4580455.

[25] Einar Karlsen. *Inn i energisektoren.* URL: https://elektronikknett.no/Artikkelarkiv/2020/Mars/Inn-i-energisektoren. (accessed: 02.06.2021).

[26] Miroslav Kubat and Stan Matwin. "Addressing the curse of imbalanced training sets: one-sided selection." In: *Proc. 14th International Conference on Machine Learning*. Morgan Kaufmann, 1997, pp. 179–186. URL: http://citeseer.ist.psu.edu/297143.html.

[27] W.J. McNutt. "Insulation thermal life considerations for transformer loading guides." In: *IEEE Transactions on Power Delivery* 7.1 (1992), pp. 392–401. DOI: 10.1109/61.108933.

[28] Reguleringsmyndigheten for Energi Mona Heien. *Utbetaling ved svært langvarige avbrudd (USLA).* URL: https://www.nve.no/reguleringsmyndigheten/okonomisk-regulering-av-nettselskap/om-den-okonomiske-reguleringen/utbetaling-ved-svaert-langvarige-avbrudd-usla/. (accessed: 11.05.2021).

[29] K. Najdenkoski, G. Rafajlovski, and V. Dimcev. "Thermal Aging of Distribution Transformers According to IEEE and IEC Standards." In: *2007 IEEE Power Engineering Society General Meeting*. 2007, pp. 1–5. DOI: 10.1109/PES.2007.385642.

[30] NYC Data Science Academy Owen Zhang. *Learn Kaggle techniques from Kaggle #1, Owen Zhang*. URL: https://www.youtube.com/watch?v=LgLcfZjNF44. (accessed: 12.05.2021).

[31] Disruptive Technologies. *Disruptive Technologies Temperature Sensor Datasheet*. URL: https://support.disruptive-technologies.com/hc/en-us/articles/360019263639-Wireless-Industrial-Temperature-Sensor-. (accessed: 12.05.2021).

[32] Disruptive Technologies. *Disruptive Technologies Website*. URL: https://www.disruptive-technologies.com/. (accessed: 11.05.2021).

[33] Abhishek Thakur. "Approaching (Almost) Any Machine Learning Problem." In: 2020. Chap. Hyperparameter Optimization.

[34] Wikipedia. *Box plot*. URL: https://en.wikipedia.org/wiki/Box_plot. (accessed: 13.05.2021).

[35] Wikipedia. *Transformer*. URL: https://en.wikipedia.org/wiki/Transformer. (accessed: 03.06.2021).