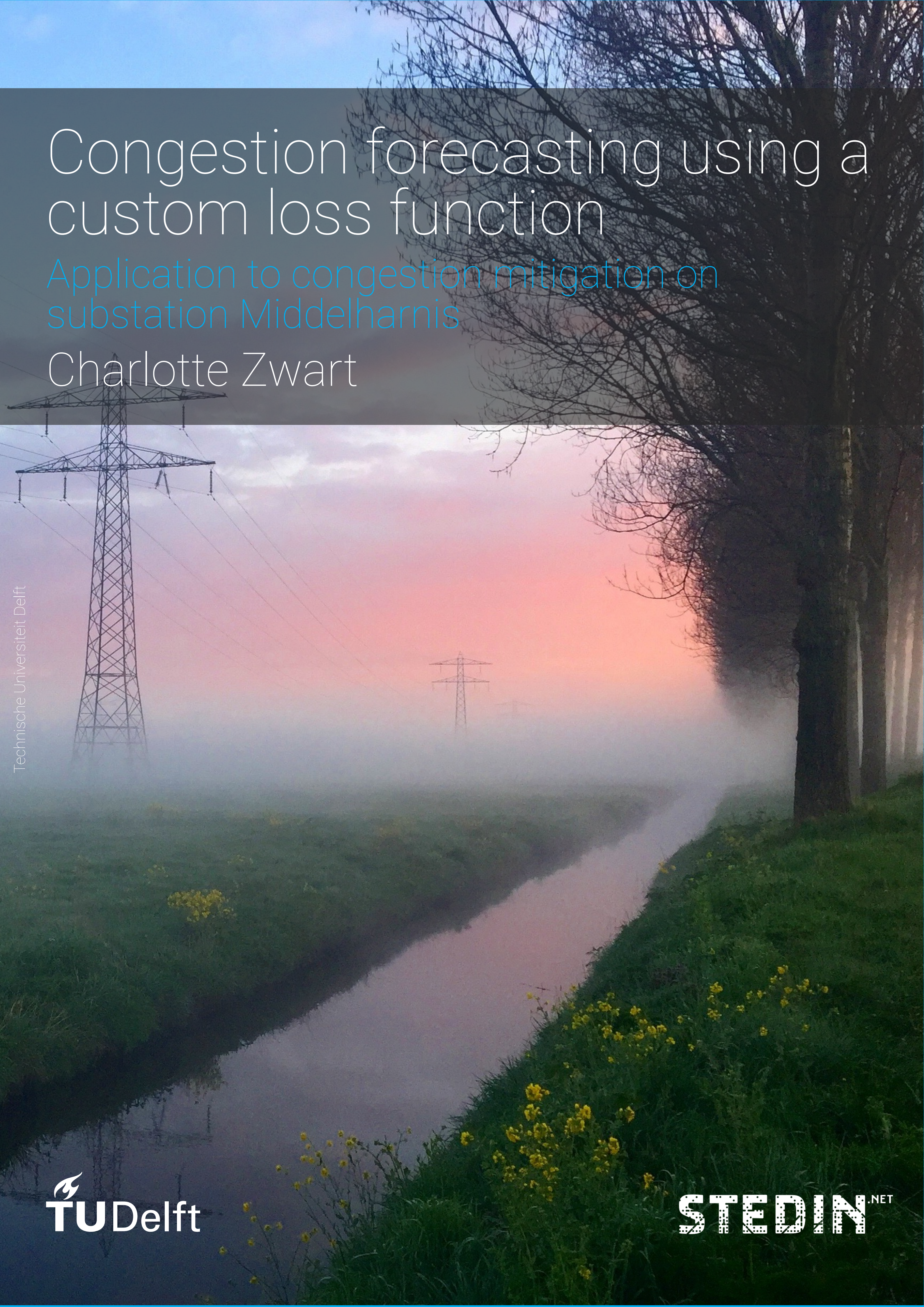
Congestion forecasting using a custom loss function

Application to congestion mitigation on substation Middelharnis

Charlotte Zwart



TU Delft

STEDIN.NET

Congestion forecasting using a custom loss function

Application to congestion mitigation on substation Middelharnis

by

Charlotte Zwart

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday November 2, 2022 at 9:00 AM.

Student number:	4430867	
Project duration:	November 8, 2021 – October 3, 2022	
Thesis committee:	Prof. Dr. P. Palensky,	TU Delft
	Dr. S. H. Tindemans,	TU Delft, supervisor
	Dr. P. Manganiello,	TU Delft
	R. Stojakovic, BICT,	Stedin, supervisor

Abstract

Whereas in the past, the Distribution System Operator (DSO) almost never encountered congestion in their grids, nowadays, with the increase of connected renewable energy sources, this will become more prevalent. To forecast congestion on transformer stations, with the goal of mitigating it, the Dutch DSO Stedin uses machine learning models with standard loss functions for regression. These fall short in predicting congestion peaks, as loss functions like MSE are unaware of the prediction goal, which is minimizing the cost associated with the congestion. Without knowledge of this goal, a loss function will not put any extra importance on finding congestion peaks and is more likely to miss them.

In this thesis, the use of a a custom loss function is explored, which incorporates the different costs associated with congestion mitigation. This function aims to improve upon existing congestion forecasts by having a loss function in line with the congestion forecasts goal. For this research, the case of substation Middelharnis is used. This substation encounters congestion due to connected wind parks and distributed photovoltaics. The congestion encountered will be solved by grid expansion in 2024, but until then will have to be mitigated using the redispatching market GOPACS.

A custom function was generated using two cost components: a congestion fee, which needs to be paid if wind parks are disconnected and GOPACS costs, which are the price of buying flexible power on the redispatch market. The function takes into account two-time horizons: costs associated with buying GOPACS power day-ahead acting on a prediction of the load, and the resulting cost on the day itself if any remaining congestion was resolved via the congestion fee. The contribution of this function is that it depends on the difference between the costs of the prediction and the cost of the realization, instead of the cost of the difference between prediction and realization.

Four models were trained, each with a different loss function: MSE, Pinball, and Cost. The cost model is trained twice and for each model, a different value for GOPACS cost is used. The results show that the cost models outperform the Pinball and MSE-trained models, when the performance is evaluated on the final cost metric. However, if the GOPACS price is high, the cost model becomes conservative in predicting congestion peaks. It is not expected that this will be an issue in the future, as with more renewable energy, the energy price and subsequently the GOPACS price will be lower.

Preface

First of all, I would like to thank my supervisor Simon Tindemans for his guidance and suggestion over the past couple of months. When I was stuck in thinking the problem was more complicated than it actually was, you cut right through that and showed me that the problem was in actuality hiding right under my nose. Next, I want to thank Ranko Stojakovic. Although the machine learning concepts might've sometimes eluded you, you were great in helping me in getting to the core and find the story I needed to tell. I also want to thank Santor Warmerdam. Although he couldn't be there until the end of the project, he was there the majority of the time. You answered any and all questions at all times of the day. You were my go-to for all machine learning-related questions and you were critical of my intermediate results as I often took unconventional approaches. Furthermore, I want to thank Leon Kuiper for guiding me through the last months and helping me polish my thesis. It goes without saying that I could not have completed this work without all of you. It was an immense learning experience and I'm thankful you were all there to guide me through.

Next, I also want to thank Stedin, and especially Arjan van Voorden for offering this project and all Data Scientists and Engineers from the department for helping me set up the project and offer advice. I'm also grateful that Anneroos Renaud took the time to explain the whole flexibility chain and kept me updated, and for Nuran Cihangir Martin for doing and sharing her flexibility pricing study.

Lastly, I'm thankful for all my friends and family for being enthusiastic with me, sometimes telling me that it's okay to take a break, and for keeping me motivated through the long hours in the university library or on EWI.

Charlotte Zwart
Delft, September 2022

Contents

1 Introduction **1**
1.1 Motivation . . . 2
1.2 Research goal . . . 3
1.3 Outline . . . 3

2 Background and Context **5**
2.1 Responsibilities of a Distribution System Operator . . . 5
2.1.1 Congestion Management . . . 6
2.2 Forecasting methods . . . 8
2.2.1 Long- and Medium-term forecasts . . . 8
2.2.2 (Ultra/Very) Short-term forecasts . . . 10
2.3 Loss functions . . . 10
2.3.1 Commonly used loss functions . . . 10
2.3.2 Evaluation metrics . . . 12
2.3.3 Impact of different loss functions on predictions. . . . 13
2.3.4 Previous research into customized loss functions . . . 14
2.4 XGBoost model . . . 19
2.4.1 Boosting . . . 19
2.4.2 Loss function per boosting round t. . . . 20
2.4.3 Learning per tree. . . . 21

3 Proposed Solution **23**
3.1 Case: Middelharnis . . . 23
3.1.1 Topology . . . 23
3.1.2 Operation . . . 27
3.2 Proposed loss function . . . 28
3.2.1 Definition of loss. . . . 28
3.2.2 Proposed custom loss function . . . 30
3.2.3 Additional evaluation metrics . . . 38

4 Methodology **39**
4.1 Data. . . . 39
4.1.1 Data sources . . . 39
4.1.2 Data cleaning . . . 40
4.1.3 Data transformation . . . 42
4.2 Features. . . . 44
4.2.1 Feature generation . . . 44
4.2.2 Feature selection. . . . 48
4.3 Training. . . . 48
4.3.1 Train/validate/test set . . . 49
4.3.2 Model Tuning . . . 50

5 Evaluation and Results **55**
5.1 Simple forecasting methods. . . . 55
5.2 Mean scenario . . . 56
5.3 Minimum scenario . . . 60
5.4 Comparison of models . . . 61
5.5 Interpretation of results . . . 62
5.5.1 Data availability . . . 62
5.5.2 Individual transformer overloads . . . 62
5.5.3 Sensitivity to hyper parameters . . . 62

5.5.4 Feature selection . . . 62

6 Conclusion and Recommendations 63
6.1 Research questions . . . 63
6.2 Discussion and Future work . . . 64
6.2.1 Definition of loss . . . 64
6.2.2 Computational capacity needed . . . 65

Bibliography 67

A Features 71

B Loss function 73
B.1 Additional loss function plots for example values . . . 73
B.1.1 GOPACS part of the loss function . . . 74
B.1.2 Congestion part of the loss function . . . 75
B.2 Loss function plots with $C_{gopacs} = 268$ €/MWh . . . 76
B.3 Loss function plots with $C_{gopacs} = 70$ €/MWh. . . . 77
B.4 Loss function implementation in Python . . . 77

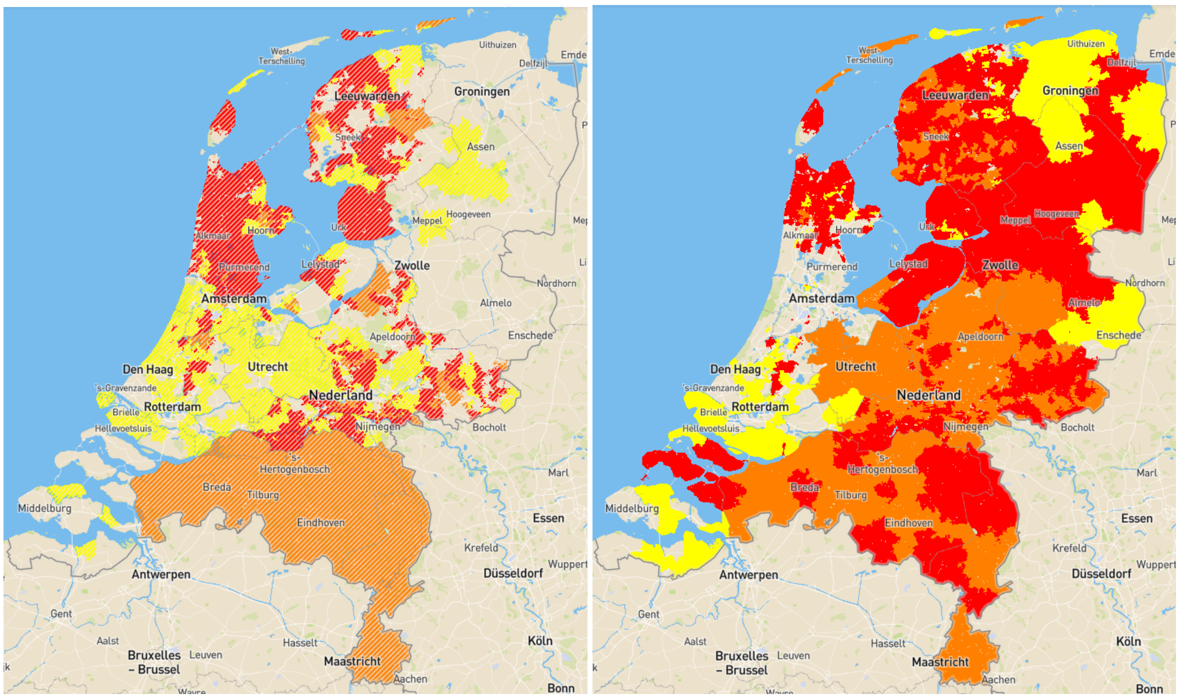
C Train/test set transformer data 79

D Results 83
D.1 Mean Scenario . . . 86
D.2 Minimum Scenario . . . 87

1 Introduction

Recently, the growth of renewable energy has been explosive, which has been a struggle for grid operators to prepare their grid for these new energy sources. This is because after the Paris Climate Agreement of 2015, Governments moved to create policies on a national level. For the Netherlands, the Paris Agreement was translated into the Climate Agreement ('Klimaat akkoord') in 2019, which aims to reduce greenhouse gases by at least 49%. In terms of renewable energy, the goal is to have 11.5 GW of wind power at sea, 3.7 GW of wind power on land, and 14.3 GW of solar power [1]. All this power needs to be transported and distributed. Unfortunately, the grid is not dimensioned to (distributed) renewable energy sources, as these sources are implemented at a faster pace than the grid can expand, and because these sources can create power peaks for small time frames for which the grid cannot be reasonably dimensioned [2]. As a consequence, parts of the grid now encounter congestion, meaning that power that needs to be transported or distributed exceeds the physical capabilities of the grid. Figure 1.1 shows the Netherlands's current transport/distribution capacity. Both maps are heavily colored, meaning that in those regions, there is no or limited transport/distribution capacity available for new customers or current customers looking to increase their existing capacity connection as these regions encounter (structural) congestion.

Distributed System Operators (DSOs) are required to provide a secure, reliable and efficient system to all system users. They do so by following a copper plate approach. This approach assumes that there is no physical constraint on the flow of energy. In this way, the DSO can facilitate any desired transfer of electricity between market parties [2]. This approach is now under pressure with the level of congestion seen in the Netherlands. DSOs like the Dutch DSO Stedin are heavily investing in their grids [4], but cannot catch up to the expansion of renewable energy. The aim is to mitigate congestion via flexibility measures, until the grid expansion is finished, and for new cases where the grid can't be expanded for only the few intermittent capacity peaks. The Authority Consumer & Market (ACM) has also signaled the need for congestion management methods and has updated the Netcode [5] to provide a legal framework for DSOs to perform congestion management. Congestion management is the act of increasing or reducing the capacity used by producing or generating parties. For this reduction of needed capacity, these parties utilize a market structure for redispatching, called GOPACS [6]. Since this is a form of an intraday market, the net power movement needs to be 0 MW, meaning that if a party in one part of the country needs to reduce load, another party in a different part of the country needs to increase its load. They bid on GOPACS where the difference between the bids is compensated by the DSO [7].



(a) Capacity map for load (b) Capacity map for generation

Figure 1.1: (Available) Capacity map of the Netherlands for load and generation, version 04-08-2022 9:11. Yellow: transport shortage imminent, altered transport tender regime. Orange: Advance notice of structural congestion at the Authority Consumer and Market (ACM). Red: Structural congestion, new requests for transport will not be granted [3].

1.1. Motivation

To mitigate congestion, first, it is necessary to know the extent of the congestion. To do this, the DSO can perform load forecasting. The goal of this forecast is to predict the amount of load that will be on a component, like a transformer or cables, for a time window in the future. With this prediction, the DSO has information on whether there is congestion, at which points in time it will occur, and what the extent of the capacity problem will be. With this information, it can attempt to perform congestion management to alleviate the probable congestion.

At Stedin, the current method to perform load forecasting is by using a Machine Learning method based on Gradient Boosting trees, which they use through the package XGBoost [8]. With these models, they can forecast the load on substations up to 64 hours ahead, although the 24-hour ahead forecast is primarily used. These models have proven very useful in predicting the load for residential areas or industrial areas with a sufficiently repeating load pattern, but fail to accurately forecast areas with a very high penetration of renewable sources. These models specifically fail to identify when a load peak with the potential to cause congestion will occur.

The reason why these models miss these congestion peaks has to do with the loss functions on which these models train. Different loss functions will give a different model result. For instance, the Mean Squared Error will try to optimize towards the prediction mean. Loss functions therefore bias a model in a prediction direction, which is not necessarily in line with the goal for which the prediction is made. For Stedin, the goal of these load predictions is to use them in making the decision to buy reserve capacity on GOPACS or not. For them, the objective is to accurately predict the height and duration of the peak, such that they can minimize the cost associated with the congestion. If there is congestion, and it is not mitigated via GOPACS, Stedin has to intervene on the day itself, which they pay a penalty for that is higher than the GOPACS costs.

In recent years, researchers have begun to identify this shortcoming of traditional loss functions in minimizing their forecast cost and developed methods to integrate this cost component in their customized loss functions. For instance, in both [9] and [10], a loss function is defined, which a model tries to minimize, that takes into account a cost component. These approaches demonstrated that by exposing a model to the asymmetrical risk/reward, the total cost of the prediction will be significantly reduced as compared to standard models.

However, even though [9] and [10] have a cost component, both still have their loss function depending on the difference between prediction $\hat{y}$ and realization y: $L = C(y - \hat{y})$. This in contrast to the proposed function where the loss is the cost of the prediction, with additionally added the cost when the realization deviated from the prediction: $L = C(\hat{y}) + C(y, \hat{y})$. The belief is that the concept of using a loss function based on the prediction and realization cost rather than load has the potential to yield even better forecast results in terms of minimizing cost.

1.2. Research goal

The goal of this research is to define a loss function, that integrates the different costs which are associated with congestion management in the DSO grid. With this loss function, a better congestion forecast can be made, which, when used in a mitigation strategy, will minimize the costs for the DSO as compared to traditional models. Its research contribution is that this function will define the cost both for prediction and realization instead of depending only on the difference between the two. This is summarized in the following main research question:

What is the value of using a custom loss function for load forecasting in reducing expected congestion costs in the distribution grid?

To aid in answering the main question, the following four sub questions are defined:

1. What is the cause of congestion and which responsibilities does the DSO have in regards to congestion management?
2. What is a suitable model of the different cost components of congestion and how can that be incorporated into a loss function?
3. How does a custom loss function perform in comparison to traditional load forecasting methods?
4. What are the advantages and disadvantages of using a custom loss function?

These questions outline this thesis and will be answered per chapter.

1.3. Outline

First, chapter 2 will start with the necessary background and context in describing the legal framework in which DSOs operate, introduce different forecasting methods, explain the relevance of loss functions, and ends with describing how a machine model learns. After this, chapter 3 will start by introducing the case which will be used to illustrate the impact of this loss function. Next, the proposed loss function will be constructed. Then, chapter 4 will go through the methodology, from data ingest to the model training. After this, chapter 5, will evaluate the results and show the differences in using standard forecast methods and the proposed loss function. Lastly, chapter 6 will conclude the report by discussing the limitations of the proposed methods and making recommendations for future work.

2 Background and Context

The first step in this research is defining the context in which the challenges lie. In recent years, the energy transition has accelerated, bringing with it new challenges that did not have to be addressed before. One such issue is accurate load forecasting and more recently also congestion forecasting. In this context, this chapter will aim to answer the following research question:

What is the cause of congestion and which responsibilities does the DSO have in regards to congestion management?

This question will be answered by giving the legal framework and by the projections about what the future of the DSO grids will look like by Netbeheer Nederland and Stedin.

Next to this, context is given on load forecasting methods in section 2.2, the concept of loss functions is introduced in section 2.3 and lastly the concept of machine learning through XGBoost is introduced in section 2.4.

2.1. Responsibilities of a Distribution System Operator

The responsibilities of a DSO are established in the Dutch Electriciteitswet 1998 [11]. In summary, a DSO is required: to operate and maintain the grid; to ensure the safety and reliability of the grids and the transmission of electricity; to construct, repair, renew or extend networks and make them future-proof in the context of the energy transition; to connect third parties to the grid and carry out the transmission on their behalf; to maintain sufficient reserve capacity for the transmission of electricity.

Netbeheer Nederland, the branch organization of all electricity and gas grid operators in the Netherlands, has done research into what the energy transition means for the system operators. They have concluded that: the current infrastructure needs extensive expansions, that these expansions will not be done rapid enough to keep up, and that a long-term perspective is necessary to come up with creative measures for flexibility to mitigate these problems in the meantime [12]. Reasons which hamper the expansion pace are currently the lack of personnel, supply of materials, and the financial burden for Transmission System Operators (TSO)/DSOs which needs to be shouldered in a limited amount of time.

This has prompted the Authority Consumers and Market (ACM) to redefine the Netcode [5]. The part which has now been updated includes the legal framework for congestion management. The ACM concluded that the necessary expansion of the grid is lagging the implementation of renewable energy, and won't be adequately completed within 5-10 years. To still be able to connect new customers and expand customers' current connections, it now obligates system operators to perform congestion management to utilize the grid more efficiently [7].

The new changes [13] to the Netcode [5] now add to the responsibilities of the DSOs. As of November 2022, the DSOs are obligated to perform congestion management. This means that DSOs are now obligated to do load flow forecasts followed by a mitigation measure for congested areas if the DSO signals a distribution problem in its daily operational planning. In case of a congestion area and a predicted congestion period by the DSO, the DSO needs to mitigate the congestion either through contracts for voluntary curtailement, and if that doesn't suffice, or in addition, then through a redispatching market. The difference between the day-ahead market and redispatching market, is that the redispatching market requires that all bought load or generation be compensated elsewhere, such that there is a net 0 MW injection in the grid.

In performing daily load flow forecast for congestion areas, DSOs are required to estimate the expected congestion, but also to provide and publish the power flow expectations to neighboring DSOs and the TSO. According to the Netcode [5], DSOs should do this on the basis of the distribution prognosis that customers in their areas are required to supply. Unfortunately, in reality, these distribution prognoses are often incorrect. According to the Netcode, connected customers only need to send an update when the change in production or generation is larger than 5% if their installation is within 60 - 200 MW or if their change in production or generation is larger than 10 MW, when their installation is larger than 200 MW. This means that if your installation is smaller than 60 MW, giving updates is not required. In fact, if the installation is smaller than 20 MW, it is not even required to give the DSO a prognosis. This poses an issue to the DSOs as they have a large number of smaller customers, which includes solar and wind parks. Without accurate distribution prognosis by their customers, they need to fall back on their own system of forecasting. At this point in time, no such forecasting is done at Stedin as it was never needed before. Section 2.2 will go more in-depth into different methods for performing load forecasting.

2.1.1. Congestion Management

Once congestion is detected, congestion management is done in two steps: First, it needs to be determined if an area is congested according to the definition given in the Netcode. An area can be congested in two ways: it either has become congested due to the natural growth in the area or because market conditions have changed; it is considered congested as no new distribution capacity is available to customers who want this. The Netcode states that an area can be considered congested if the available distribution capacity is no longer sufficient as a result of one of the above reasons. When this happens, contrary to its obligations as stated in the Electriciteitswet [11], the DSO is no longer obligated to reserve distribution capacity for new or existing customers. When this situation occurs, the DSO will notify the ACM of the situation and give off a prior notice to market parties. This prior notice contains the geographical area, the duration of the congestion period, the reason for the expected congestion, the total available and needed distribution capacities, a timeline and planning for the grid expansion, and an invitation for market parties to help come up with ways to alleviate the congestion. After this notice, it will start an investigation on whether if congestion management is required [5]. The reason this investigation is done is that it is not always feasible for the DSO to adhere to the procedures required by the Netcode to alleviate congestion. In the following cases, the DSO is relieved of its requirement to perform congestion management:

- The distribution capacity shortage will be shorter than one year and the congested area has not been a congestion area in the past three years;
- The price to perform congestion management in the time frame from the prior notice up until the finished grid expansion is larger than the financial limit. This financial limit is 1.02€per MWh available distribution capacity in the congested area;
- The demand for distribution capacity is larger than the technical limit. Here the technical limit is defined as 110% of the existing distribution capacity plus additional distribution capacity by means of controllable flexible capacity, with a maximum of 150% of the existing distribution capacity;
- Distribution will lead to the short-circuit power exceeding legal limits.

Once the investigation yields the result that the investigated area is in fact formally congested, the DSO will need to start performing active congestion management. This means the DSO is not required to install new customers or expand connections for current customers above 110% of the distribution capacity and that it acquires more system operation duties.

According to the Netcode [5] the DSO needs to perform the actions as visually represented in Figure 2.1. For a congestion area, the DSO can have bilateral contracts with load/generation parties. Once the DSO signals that congestion is imminent it can call on these contracts for voluntary curtailment, for which the market party receives a predetermined amount of compensation in €/MWh. After the distribution prognosis is made, the DSO can also buy either balancing reserve contracts or can put out a market call for redispatch capacity on a redispatch market. These fit the capacity reduction and redispatch products as laid out in the annex of the Netcode [5]. Intraday, the DSO doesn't have the option to buy reserve capacity contracts as a net 0 MW power balance is imposed. When the previously bought capacity contracts are activated, but insufficient, this results in only redispatching products being available to the DSO. If the DSO expects no intraday congestion, but underlying market parties could cause congestion after trading on the intraday market, the DSO can impose market restrictions for these parties to keep the power balance. Real-time system operation tasks involve monitoring and if needed, switching or calling into action an existing contract manually or autonomously.

GOPACS

For the Netherlands, a redispatching trading platform was introduced in December of 2018, called GOPACS. The aim of this platform is to solve intraday congestion, by compensating market parties for using their load or generation flexibly. This platform uses existing intraday markets ETPA and EPEX Spot (from 2023). Via these markets, a capacity request due to congestion from the TSOs/DSOs can reach market parties, who can bid their available flexible capacity. Since redispatching requires a net 0 MW movement, the platform couples buy and sell bids in pre-specified geographic areas together, and checks if the new situation solves the congestion and does not cause a new congestion situation. The difference between the bids will be paid by the TSO/DSO and the bids are executed, solving the congestion situation [6].

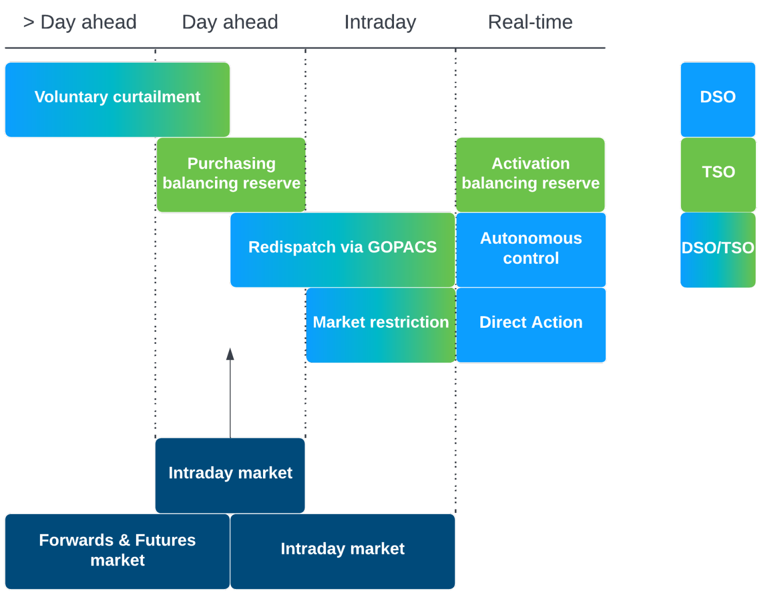


Figure 2.1: System operation tools, using markets for varying time scales. Adapted from [14].

2.2. Forecasting methods

Load forecasting is estimating the load for a specific time horizon in advance. Depending on the time horizon this has different uncertainties and different use cases. The time horizon for load forecasting is represented in Figure 2.2. Long-term forecasting is useful to be able to predict if, when, and where grid expansion will become necessary. Medium-term planning is used for seasonal planning. With it, you can plan when connections or transformers will be taken offline for maintenance. Also, with seasonal planning, an assessment is done to see if the grid can be safely operated in the coming year. Short-term forecasting has in recent years become more important. With short-term forecasts, the DSO is able to make preparations for switching actions, do grid safety analysis for the next hour to a day ahead and determine where congestion will happen, so it can be mitigated. Ultra/very short-term forecasts are used for continuous real-time monitoring and detecting disturbances quickly. One such application is fault classification, such that restoration can happen either automatically or the process is sped up by having the knowledge of the root cause [15].

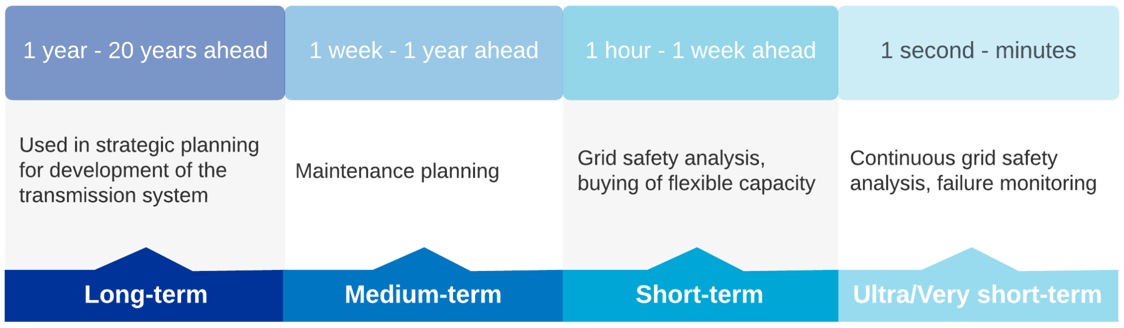

1 year - 20 years ahead
1 week - 1 year ahead
1 hour - 1 week ahead
1 second - minutes
Used in strategic planning for development of the transmission system
Maintenance planning
Grid safety analysis, buying of flexible capacity
Continuous grid safety analysis, failure monitoring
Long-term
Medium-term
Short-term
Ultra/Very short-term

Figure 2.2: Load forecast time horizons and their use cases. Adapted from [15].

After determining the time horizon of the forecast, the next step is to determine the suitable model for this case. Figure 2.3 gives a classification of the different forecasting methods available. This figure is by no means exhaustive but gives a summarized overview. The figure shows that the methods can be split into two categories: Qualitative and Quantitative. The qualitative category is often utilized if data is scarce, of bad quality, or nonexistent. The prediction is made on the subjective opinions of experts or with relatively rough/simplistic data modelling. The quantitative category is utilized when sufficient data is available and if it can be assumed that past trends will continue into the future. Quantitative methods include mathematical and statistical models based on the available data [16].

2.2.1. Long- and Medium-term forecasts

For forecasting load in the medium to long term, the methods to use lie in the qualitative direction. Here, there are two main methods: the Econometric approach and the End-use approach. "The econometric approach combines economic theory and statistical techniques for forecasting electricity demand. The approach estimates the relationships between energy consumption (dependent variables) and factors influencing consumption" [21, p. 1533]. The advantage of this method is that it gives a good explanation as to what factors contribute to the prediction and explains how the model came to the prediction [16]. A disadvantage is that past information is in some cases not always useful for predictions. The End-use approach uses a bottom-up approach. It looks at what the energy is used for, like appliances, mobility, industry, etc. The model will try and explain the demand for energy as a function of the number of appliances [21]. To forecast the next 20 years, often scenarios are made to see how these appliances/industries will grow to see what the impact is on the energy demand. This is often done through the Delphi method, which polls experts through various rounds until a consensus is reached.

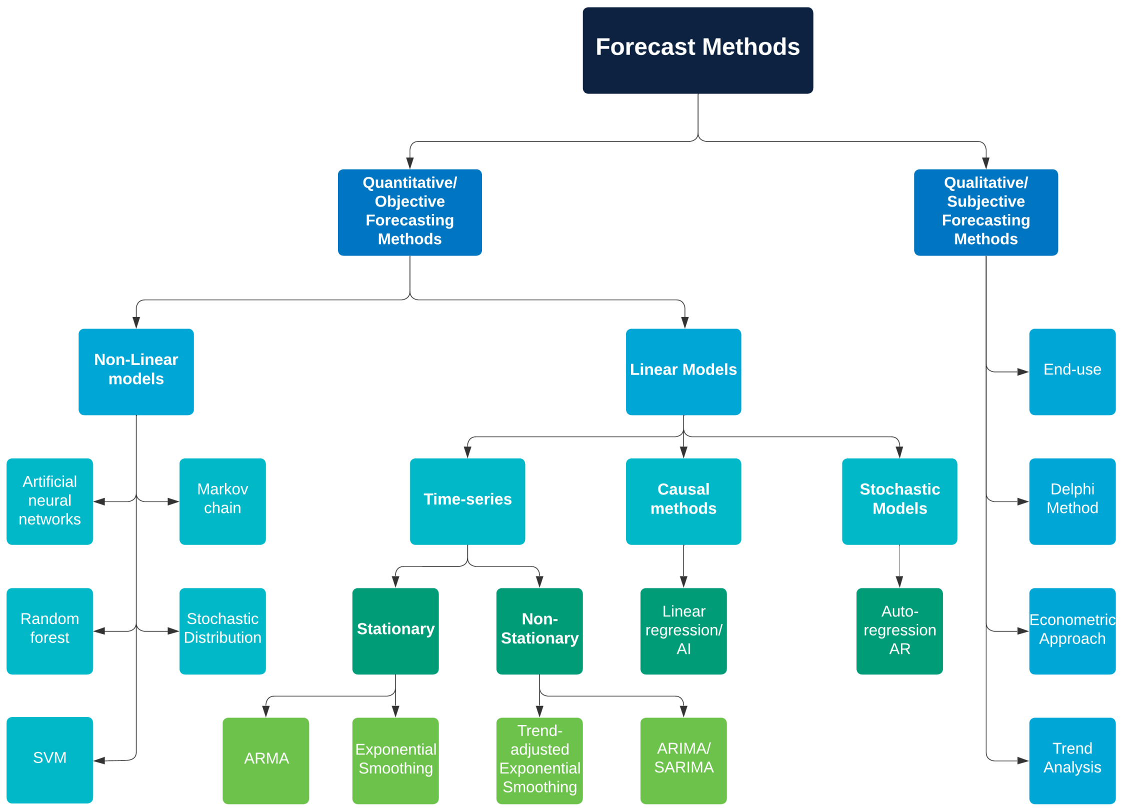


Figure 2.3: Classification of the different load forecasting methods. Adapted from [17–20].

Netbeheer Nederland [12] has done such a prediction for the entire Dutch energy system, to see how our current grid will hold up against these scenarios and where issues will arise with their root cause. They use four scenarios: Regional, National, European, and International. These four scenarios are based on the scale the Dutch Government will try and steer the energy transition. The goal of this report is to predict the energy transition development in the Netherlands from 2030 to 2050. It goes into detail about which infrastructure adjustments we need to make to the grid to facilitate the different scenarios, the (societal) cost associated with these changes, and gives insight into the viability of the transition on the entire energy system.

Stedin [4] has also made scenarios to determine the impact on their infrastructure. The three scenarios they work with are Climate Agreement, National Push, and International Ambition. The first scenario predicts a push for the energy transition, but without a lot of urgency, and the initiative is left to individuals and companies. In the second scenario, the government takes a very active role in making the energy transition possible and they want the Netherlands climate neutral by 2050. Lastly, the international scenario sees climate agreements on an international level with a strong international energy market. These markets ensure that there are enough energy carriers like electricity, hydrogen, and biofuels. In this scenario, there is less electrification and more use of (green) fuels. With these scenarios, Stedin determines if, where and on what timescale their grid needs maintenance, upgrades, or expansion.

Depending on the use case of the forecast, it might be that for the medium term these two approaches might not yield the desired results. In these cases, there is a small overlap with the Short-term forecast. For these purposes, it’s also possible to do Trend analysis, to extrapolate the future energy demand through historic data. Another option is Statistical models. These also utilize past load data, but also take into account other variables like the day of the week, the hour of the day, or weather data [16].

2.2.2. (Ultra/Very) Short-term forecasts

Short-term forecasting is almost always exclusively done through Quantitative methods. Within these methods, there is a distinction between linear and non-linear models. Linear models predict future values based on a linear combination of past values. The non-linear category features many machine learning methods, although Figure 2.3 only shows a small subset of all the possible solutions here. Linear models can be broken down further into Stochastic models, like autoregression models, Causal methods which seek a way to find a relation between all variables that can influence a prediction target in the future, and time series. In time series there is a further distinction between stationary and non-stationary series. Stationary in this context means that the series has a constant variance at all times, meaning there is no trend and no seasonality. Although some non-stationary data sets can be transformed to be stationary, this isn't always possible. For the stationary series, you can have the statistical method of Auto Regression and Moving Average or Exponential smoothing, which has a moving average, but it weighs samples exponentially, such that older samples have a smaller influence. The non-stationary analysis has the same basic methods but these are adapted, such that they can handle the non-stationarity of the set.

Whichever model is better depends on the goals of the short-term forecast. A distinction can be made if the approach is going to be a probabilistic or deterministic forecast. A probabilistic forecast provides "the realization of a random variable at some future time, which is an estimate of the probability distribution of the possible future values of that variable" [22]. A probabilistic forecast will therefore describe the uncertainty associated with a prediction. This stands in contrast to a point forecast, a deterministic forecast that is a single predicted value, which has no information about the associated uncertainty.

If it is important to have an interpretable model, statistical approaches might be best as machine learning models are often seen as a 'black box'. Training machine learning models requires a lot of computational resources. Nowadays, this is becoming less of an issue, but this could be a bottleneck in applying these techniques. The best method is also dependent on which and how much data is available [21]. Therefore, there is no 'one best' forecasting method. It is dependent on the application, data availability, and goals of the user. In general, there is a trend in which methods are most preferred by researchers. Hamma, Jereb, and Dragan [18] have reviewed different load forecasting methods and models. In it, they note that the topic of load forecasting is becoming increasingly more popular, measured by the volume of papers written over the years. Of these papers, most were written about Artificial Neural Networks, followed by regression models. They conclude that research into methods using time-series models like ARIMA/SARIMA/ARMA is declining in favor of Neural Networks, Regression, and Support Vector Machines. This is explained by a recent extensive review paper on forecasting [23], where this decline in interest for time-series in favor of Neural Networks (NN) is also described for short-term load forecasting. They conclude that in the literature for short-term forecasting time-series, Machine learning (ML), and hybrid approaches are the most accepted. Depending on the specific application, ML and hybrid approaches generally yield better results, explaining the decline in interest for time-series based models.

2.3. Loss functions

As with the type of model, there can also be a wide variety of loss functions with which a model can train. The type of loss function that can be chosen first depends on the task: Classification or Regression. For regression, the task this thesis focuses on, there are a number of loss functions to choose from, each yielding a different result. The next subsections will introduce common loss functions, show the impact of different loss functions on a prediction, and end with showing the use of a custom loss function through previous research.

2.3.1. Commonly used loss functions

The purpose of the loss function is for a machine learning model to minimize the loss as defined by this function. The loss function used, should be chosen depending on the prediction goal. Figure 2.4 shows three loss functions, which will be individually introduced in the paragraphs below, together in one graph.

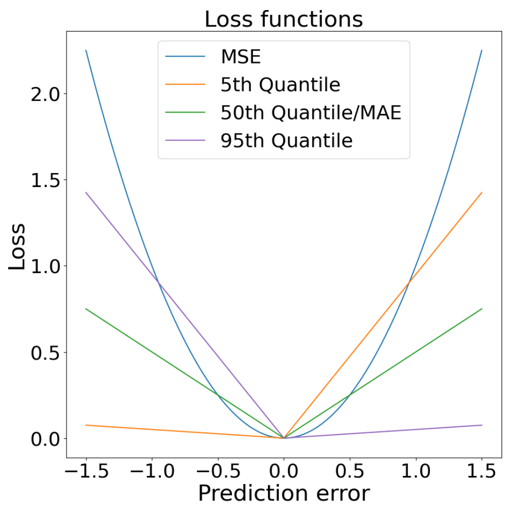


Figure 2.4: Collection of four different commonly used loss functions for regression. Their behavior is shown for both positive and negative prediction error.

MSE

The Mean Squared Error (MSE) is the most commonly used loss function for regression tasks. The MSE is defined in Equation 2.1, where N is the total amount of samples, y_i is the true value of your target, and $\hat{y}_i$ is the predicted value of the target.

$$MSE(y,\hat{y}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2.1}$$

The MSE incorporates both the variance and the bias of the error in its score, as it measures the spread of the predictions from one sample to another and how far off the prediction is from the true value. As the MSE is the squared distance between prediction and realization, it is symmetric, meaning that over and under forecasting by the same amount gives the same MSE. Due to the squaring, MSE is sensitive to outliers, as predictions that are far away, result in a squared distance as a loss. Another consequence of using the square is that the function is differentiable twice. This makes it useful in models which require the second derivative. Also, the first derivative is a function with a gradual gradient, which helps the model converge to the minimum efficiently [24].

MAE

The Mean Average Error (MAE) is one of the simplest loss functions. It is the average over the absolute error between prediction and realization as defined in Equation 2.2. The advantage of the MAE over MSE is that the relation between the error and the loss is linear, this makes it more robust to outliers than MSE or RMSE. This also makes the interpretability of the error easier as it is the average of the absolute error, as compared to the mean of the (square root of the) squared errors. As with the RMSE, the derivative of the MAE has constant gradients, making it more difficult than MSE to find the minimum [24].

$$MAE(y,\hat{y}) = \frac{1}{N}\sum_{i}|y_i - \hat{y}_i| \tag{2.2}$$

Pinball loss

Pinball loss provides a forecast with a τ probability of under forecasting and a $(\tau-1)$ probability of over forecasting [25]. Therefore, Pinball loss functions are useful if you want to bias your prediction more towards over or under forecasting or in case you want to learn more about the distribution by training multiple models. Pinball loss, sometimes also called Quantile loss, is the loss function in which MAE (τ=0.5) is a special case. It is defined in Equation 2.3, where $\tau \in [0,1]$ is the probability that target variable y occurs in the quantile q_τ [25]. For instance, if τ is 0.8, it can be said that a value of y is with 80% certainty smaller or equal to the quantile q_τ. The quantile is an evaluation of the distribution's Cumulative Distribution Function at τ: $q_\tau = F^{-1}(\tau)$. The loss is split up in two parts. The left part will multiply τ with the difference in prediction and realization if the prediction is smaller than the realization, thus being part of the quantile q_τ. The right side will multiply $(1-\tau)$ with the difference, when the prediction is larger than the realization, which is when the prediction falls outside the quantile.

$$\text{Pinball loss}\left(y,\hat{y}\right) = \frac{1}{N}\left[\tau \sum_{\forall y_i \geq \hat{y}_i} |y_i - \hat{y}_i| + (1-\tau) \sum_{\forall y_i < \hat{y}_i} |\hat{y}_i - y_i|\right] \tag{2.3}$$

2.3.2. Evaluation metrics

After training a model, scoring needs to be done to evaluate model performance. These scoring functions can be the same as the loss functions, such as MSE, MAE, or Pinball loss. Other times, different metrics are used to score the model.

RMSE

One such metric is the Root Mean Squared Error (RMSE), which is adopted from the MSE. The RMSE is defined in Equation 2.4. Instead of the variance, the RMSE captures the standard error. This has an advantage over MSE in evaluating model performance, that the RMSE score is in the same unit as the target variable, giving a more intuitive understanding of the result [24].

$$RMSE\left(y,\hat{y}\right) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(y_i - \hat{y}_i\right)^2} \tag{2.4}$$

R^2 score

Another function that is often used as an evaluation metric is the R^2 score, also known as the coefficient of determination, which is defined in Equation 2.5. This score is generally in the range [0,1], where 1 is the best possible score. The score becomes 0 when the model consistently predicts the average value of y. Also, the score can become negative, indicating a particularly poor prediction. This is different from loss functions, where one prefers the model with the lowest score.

$$R^2\left(y,\hat{y}\right) = 1 - \frac{\sum_{i=1}^{N}\left(y_i - \hat{y}_i\right)^2}{\sum_{i=1}^{N}\left(y_i - \overline{y_i}\right)^2} \tag{2.5}$$

"This score represents the proportion of variance (of y) that has been explained by the independent variables in the model. It provides an indication of goodness of fit and therefore a measure of how well unseen samples are likely to be predicted by the model, through the proportion of explained variance" [26]. In the case where the prediction error $(y_i - \hat{y}_i)$ has a zero mean, the R^2 score is the same as the Explained variance score. Therefore, the advantage of the R^2 score, is that it takes into account systemic offset (bias) in the prediction.

2.3.3. Impact of different loss functions on predictions

In this section we illustrate the impact is of choosing different loss functions. For this, we have two data sets, which can be seen in Figure 2.5. The first dataset, on the left, is data generated around an increasing mean, also with increasing variance based on normal distributions. The second dataset, on the right, has used the same mean in generating these samples, but this time the variance is added by adding to each sample a random value drawn from a Pareto distribution.

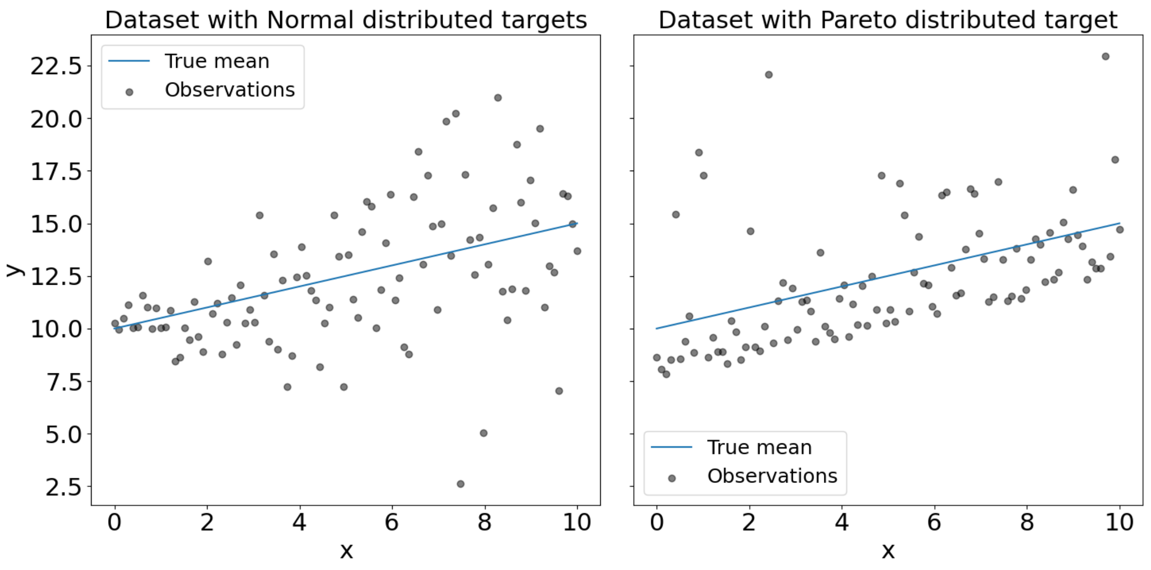
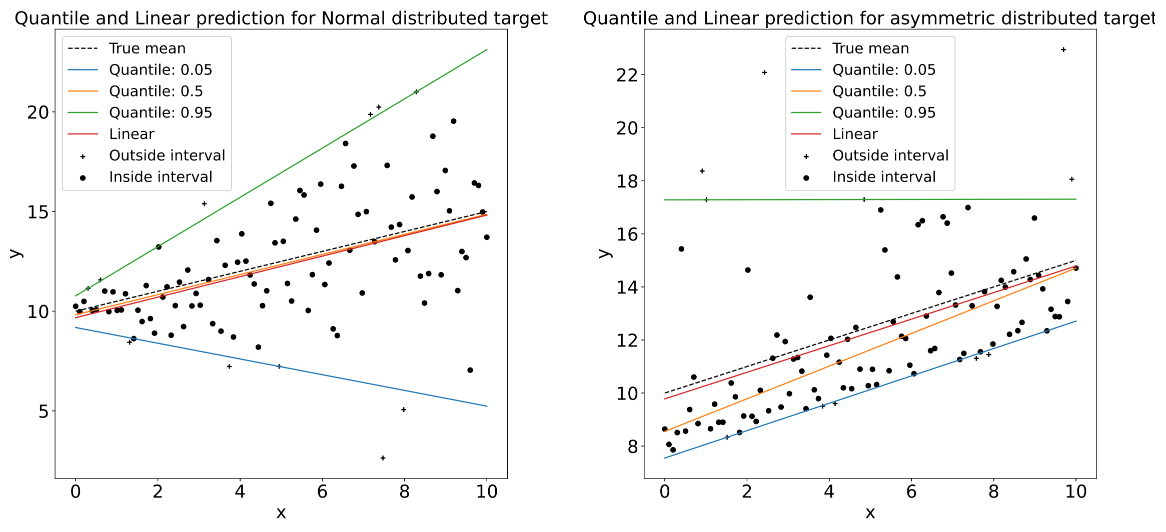


Figure 2.5: Generated data sets for mean with added noise from a Normal (left) and Pareto distribution (right). Adapted from [27].

To predict these two datasets, two different regression methods will be used to generate a linear prediction: LinearRegressor() and QuantileRegressor() from the sklearn toolbox. The Linear regressor trains on the MSE loss function (Equation 2.1) and the Quantile regressor trains 3 separate times for the 5$^{\text{th}}$, 50$^{\text{th}}$, and the 95$^{\text{th}}$ quantile according to Equation 2.3. In Figure 2.6a we see that the 50$^{\text{th}}$ quantile, or the MAE and the Linear model training on MSE coincide almost with the true mean of the distribution. This is expected as the data is normally distributed and the MSE estimates the mean, and MAE estimates the median, which in a normal distribution is the mean. The only difference from the mean is due to one side having higher outliers than the other. As MSE is more sensitive to outliers than MAE, we can see that the red line of the MSE is just below the MAE prediction. In Figure 2.6b the difference between the two is more noticeable. The 50th Quantile/MAE under-predicts the mean, as it weighs all sample differences equally, and therefore predicts closer to the median, which sits lower than the mean in this distribution. MSE is more sensitive to the higher outliers, as it squares large distances. Therefore, we can see that the MSE, in this case, approximates the true mean better, as the model training is biased upwards by these high outliers, which is expected as MSE approximates the mean.

For both cases, we can see that the upper and lower quantile prediction behave as expected, with respect to including the data. This dataset contains 100 samples and for both figures, we count 5 samples above and below the quantile predictions. Both data sets have the slope of the quantile prediction not matching the true mean. They widen to include/exclude 95% of the samples.

Which function is best depends on both the dataset and the noise distribution, as well as the goal for the prediction. If the goal of the prediction is to contain 95% of all the samples, for fear of missing the extreme cases a 5$^{\text{th}}$ quantile prediction might be best. If the dataset contains extreme outliers, MAE might be a better fit to predict closer to where the bulk of the data points resides. If the dataset contains normally distributed noise, MSE might be better as it converges better to the minimum, due to having a smooth gradient.



(a) (b)

Figure 2.6: Prediction results for different loss functions on both datasets. Left (a), the model with Normally distributed error, and right (b) with Pareto distributed error.

2.3.4. Previous research into customized loss functions

As seen in the previous section, the choice of loss function will bias a prediction in the direction of a specific outcome. For MSE this was the mean and for MAE the median. Next to the standard functions from the previous sections, custom-made loss functions for specific applications can also be constructed. The authors of [9, 10, 25, 28] argue and show that for load forecasting purposes a custom approach yields better results on the evaluation metrics they deemed important than the standard evaluation metrics.

Probabilistic short-term forecast

One approach to a customized approach is described by Gürses-Tran, Flamme, and Monti [25]. This approach breaks away from the main consensus that load forecasting can best be done by point forecast through time series models, Neural networks or hybrid combinations of the two [23]. Instead, the authors of [25] choose a probabilistic approach. They argue that this is necessary since congestion by Distributed Energy Resources (DERs), like solar or wind, is highly volatile, and residential energy consumption patterns can be highly variable as they depend non-linearly on weather and on specific special calendar days.

The authors use an RNN as a model. This model is trained three separate times; first with an Ignorance Score, second with a Pinball Loss, and lastly with a Continuous Ranked Probability Score (CRPS). For the Ignorance Core and CRPS trained models, the prediction returns the parameters of a normal distribution. The pinball loss model returns a predicted value $\hat{y}_t$ for each time step t, as well as the lower and upper quantiles.

Ignorance Score

The Ignorance Score, also referred to as the Gaussian Negative Log-Likelihood Loss when the Probability Density Function of the prediction is normally distributed, is defined as:

$$ign(f, y_t) = -log\, f(y_t) \tag{2.6}$$

Here f is referred to as the Probability Density Function of the prediction $\hat{y}_t$ with a mean μ and a standard deviation σ. The ignorance score aims to verify if the true value y_t, lies in the predicted distribution f, and it quantifies the information deficit or ignorance. However, this is a score for only one pair of a prediction distribution and realization. For the entire prediction set from $[y_0, ..., y_T]$ we have an averaged ignorance score, which in the case of this paper is defined for a normally distributed PDF of the prediction, which is defined in Equation 2.7.

$$IGN\left[N\left(\mu,\sigma^2\right),y\right] = \frac{1}{T}\sum_{t=1}^{T}\left(\frac{1}{2}ln\left(2\pi\sigma_t^2\right) + \frac{(y_t-\mu_t)^2}{2\sigma_t^2}\right) \tag{2.7}$$

Where $N\left(\mu,\sigma^2\right)$, is the normally distributed prediction distribution f, μ_t is the mean and σ_t^2 is the variance of this prediction distribution at time step t, y the realization and y_t the realization at every time step t.

Pinball Loss

Pinball Loss is also referred to as Quantile Score. Here a quantile q_τ indicates a probability $\tau \in [0,1]$ for which the target random variable y resides in it. For instance, if τ is 0.8, it can be said that a value of y is with 80% certainty smaller or equal to q_τ. The quantile is an evaluation of the distribution's CDF at τ: $q_\tau = F^{-1}(\tau)$. The pinball loss is a method optimized to provide a forecast with a τ probability of under forecasting and a $(\tau-1)$ probability of over forecasting:

$$L_\tau\left(y,\hat{y}\right) = \frac{1}{T}\left[\tau \sum_{\forall y_t \geq \hat{y}_t} |y_t-\hat{y}_t| + (1-\tau)\sum_{\forall y_t<\hat{y}_t} |y_t-\hat{y}_t|\right] \tag{2.8}$$

Continuous Ranked Probability Score

"The Continuous Ranked Probability Score is the mean squared error (MSE) of the predicted CDF $F(\hat{y})$ and of the Heavyside function $\mathbb{1}(\hat{y}-y)$, which is 0 if $\hat{y}_t$ is lower than the true value y_t and 1 otherwise" [25]. It measures the difference between the predicted cumulative prediction $F(\hat{y})$ and occurred distribution $\mathbb{1}(\hat{y}-y)$. It is defined as:

$$CRPS = \frac{1}{T}\sum_{t=1}^{T}\int_{-\infty}^{+\infty}\left[F\left(\hat{y}_t\right)-\mathbb{1}\left(\hat{y}_t-y_t\right)\right]^2 d\hat{y} \tag{2.9}$$

The score is the average of the scores over the individual time steps. The final output is a score between 0 and 1, where 0 is a perfect forecast and 1 is the worst obtainable score.

The authors of [25] conclude that for 40-h ahead predictions of load and potential congestion, the CRPS model performs the best, according to four evaluation metrics they set, although the performance of all three models is very close.

Linear type cost-oriented loss function

Next, Li and Chiang [9] show a method of making an optimal point forecast with the use of cost-oriented loss functions with the goal to forecast wind power generation. Forecasting this with an asymmetric loss function is highly profitable as the highest profit is achieved by minimizing the cost. Their cost is based on selling energy in the day-ahead market and then participating in the balancing market. For their research purpose, having to increase their wind energy is more expensive than downregulating. This is because it's not possible to increase wind energy output (unless you're operating below your maximum operating point), therefore you need to buy balancing capacity from another generator. It is possible to decrease your operating point and sell this space. Due to this asymmetry, it is beneficial to skew the forecast towards under-forecasting.

To achieve this, the authors use a custom-made loss function. They note that "ARIMA models, the backpropagation neural network, and more, assume that the quadratic loss function is built in. Unlike cost-oriented loss functions, the quadratic loss function is differentiable for the whole range of $-\infty$ to ∞. Furthermore, these methods are specially designed to introduce the quadratic loss function in the model building process and then carry over in the forecasting process. Hence, these methods are ineffective to handle cost-oriented functions" [9]. To solve this, the authors use, what they call, a cost-oriented boosted regression tree method (COBRT) in their approach. This allowed them to implement their custom loss function, defined in Equation 2.10. Here their loss function is split into discrete pieces, depending on the magnitude of the error between the prediction $\hat{y}$, and the realization y. Figure 2.7 gives a visual explanation of these discrete segments for their final implementation.

$$L_{co}(y,\hat{y}) = \begin{cases} l_1(y,\hat{y}), & \text{if } -\infty < y-\hat{y} < \delta_1 \\ \dots & \\ l_i(y,\hat{y}), & \text{if } \delta_{i-1} < y-\hat{y} < \delta_i \\ \dots & \\ l_n(y,\hat{y}), & \text{if } \delta_{n-1} < y-\hat{y} < \delta_\infty \end{cases} \tag{2.10}$$

Here, $\hat{y}$ is the prediction of the actual realization y, $l(y,\hat{y})$ a linear function that quantifies the loss in terms of cost on that segment, and δ is the breakpoint between different segments where the different costs apply. To this loss function, three conditions apply:

1. $L_{co}(y,\hat{y}) = 0$, when $y = \hat{y}$, meaning there is no cost if the forecast error is 0.
2. min $L_{co}(y,\hat{y}) = 0$, meaning that cost cannot be negative, it's always larger or equal to zero.
3. $L_{co}(y,\hat{y})$ is monotonically non-decreasing as it moves away from zero. This means that on both sides of the origin, a larger value of forecasting error leads to a larger cost.

For the implementation of Equation 2.10 with actual costs, the authors base themselves on papers they read on regulation costs related to the example they were working on. The result is Equation 2.11. The visual representation of this can be seen in Figure 2.7. The results they've shown is that their method performs better at reducing cost than a method based on quadratic error. Their method doesn't minimize RMSE, but it does minimize the total cost, which the quadratic models fail to do.

$$\begin{aligned} e &= y-\hat{y} \\ C(e) &= \begin{cases} -1.2e-0.4 & \text{if } -\infty < e < -0.1 \\ -0.8e & \text{if } -0.1 \le e < 0 \\ -0.2e & \text{if } 0 \le e < 0.1 \\ 0.4e-0.02 & \text{if } 0.1 \le e < +\infty \end{cases} \end{aligned} \tag{2.11}$$

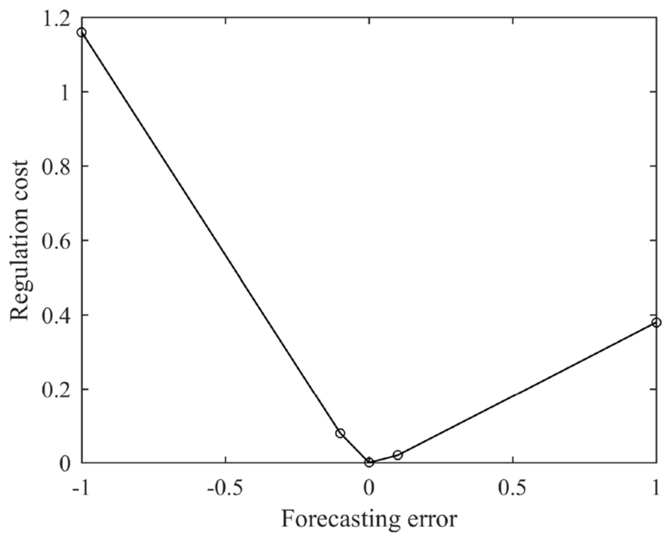


Figure 2.7: Implementation of Equation 2.11. Forecast error is defined as $e = y - \hat{y}$, where y and $\hat{y}$ are normalized between [0, 1]. Regulation cost is unitless [9].

A similar approach to this loss function was made by Kebriaei, Araabi and Rahimi-Kian [28]. They defined their loss function as $J_k = P_k \cdot e_k^2$. Here $e_k = \hat{y}_k - y_k$, which is the forecast error. P_k is visually represented in Figure 2.8. The full loss function is defined as:

$$J_k = \begin{cases} |e_k|^2 \cdot \left[\left(\frac{N_k + M_k}{2} \right) - \left(\left(\frac{N_k - M_k}{2} \right) \cdot sgn(e) \right) \right] & |e_k| \geq \varepsilon \\ 0 & |e_k| < \varepsilon \end{cases} \tag{2.12}$$

Where k is the hour index, N_k is the penalty for forecasting more than the actual load, M_k is the penalty for forecasting less than the actual load, here N_k is always larger than M_k and ϵ is a lenience window in which no penalties are necessary. In actuality these are two step functions at $-\varepsilon$ and ε, where the first is defined from $-\infty$ to $-\varepsilon$, and the second from ε to ∞ with a height of M_k and N_k respectively [29]. In the region between $-\varepsilon$ and ε the function is 0.

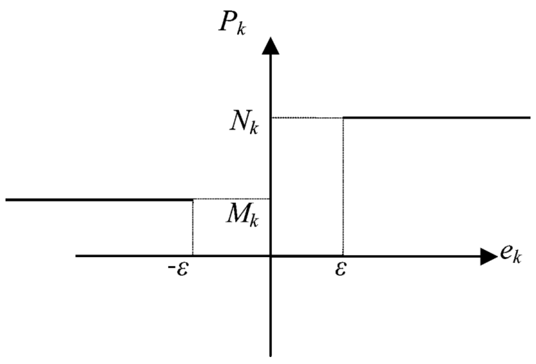


Figure 2.8: Visual definition of the cost coefficient P_k. N_k is the cost/MW for over forecasting the load and M_k is the penalty for under forecasting. The forecast error range $[-\varepsilon, \varepsilon]$ is a leniency window in which no penalty is given [28].

However, this approach has a shortcoming in assuming the costs to be static. In real-time, the cost for up/down-regulating depends on the time of day. At different times of the day, different units may be available or not, changing the pricing.

Sampled cost loss function

Zhang, Wang, and Hug [10] have taken a different approach in generating their own loss function. Their opinion is that approaches like the ones mentioned above don't reflect the actual cost. Even though they have a cost component to them, they are constructed with discrete sections and cost. To mitigate their concern, they made a loss function that is only based on sampled actual cost. Their goal was to successfully make a day-ahead forecast, where they define the ideal cost as $C(y_i^*)$ and the actual cost associated with the forecast as $C(\hat{y}_i)$. They define their loss as :

$$L(\hat{y}_i - y_i^*) \simeq C(\hat{y}_i) - C(y_i^*) \tag{2.13}$$

To generate their loss, they sample a total of 10,000 load forecasting scenarios uniformly between $[0.9y_i, 1.1y_i]$, where y_i is the actual load, for every hour out of the day using the Monte-Carlo method. A loss function is then made by calculating the cost for every sample, ordering them, and performing a linearization step, to ensure the function is continuous and differentiable everywhere. These steps are done three times, to generate three loss functions with different resolutions: hourly, daily and linear. In Figure 2.9 the hourly loss function for the hours 8 and 19 can be seen. The figure also shows the average daily loss function and a simple linear loss.

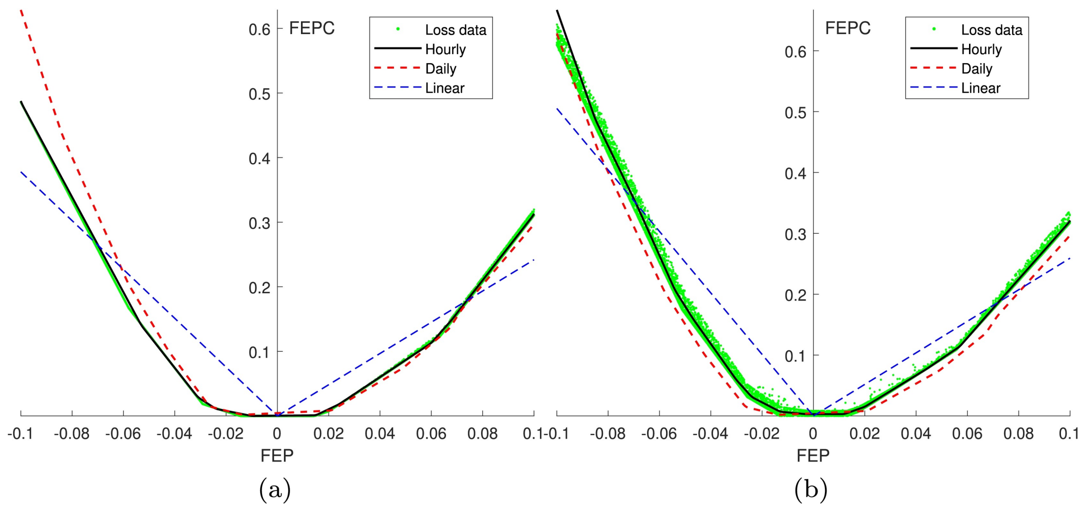


Figure 2.9: Hourly cost-oriented loss function at (a) hour 8 and (b) hour 19. The x-axis is normalized prediction error $\frac{\hat{y}_i - y_i}{y_i}$ and the y-axis normalized cost loss $\frac{C(\hat{y}_i) - C(y_i)}{C(y_i)}$ [10].

To evaluate the models four evaluation metrics are used, namely the Mean Absolute Percentage Error (MAPE), the Forecasting Error Percentage Cost (FEPC), the Mean Forecasting Error Percentage Cost (MFEPC), and the Under,- and Over Forecasting Percentage (UFP, OFP).

The authors conclude that their model outperforms their benchmark MSE error model by a little over 13% in terms of Mean Forecasting Error Percentage Cost for the hourly model. They also conclude that hourly is far superior to daily and linearly, due to the cost distribution changing over the day due to time and weather effects.

2.4. XGBoost model

The next step in understanding machine learning methods is to examine how a Gradient Boosted Tree learns and how the loss function fits into this. In particular, we will look at how XGBoost [8] learns, as this will be the model of choice in this thesis. The first step is understanding the terminology. Figure 2.10 shows an example regression tree with the component names highlighted. How the tree will look will depend on the constraints put on the tree building by hyperparameters. If this tree would learn on a feature set **x** with 5 rows of training data, each leaf would contain one distinct sample of x. Figure 2.11 shows two examples of decision trees. The first tree shows two nodes. Each node corresponds to a feature in the dataset, in this case age, male and 'uses computer daily'. Based on these features the data can be sorted. How individual trees are learned and on which criteria a split is made will be explained at the end of this section.

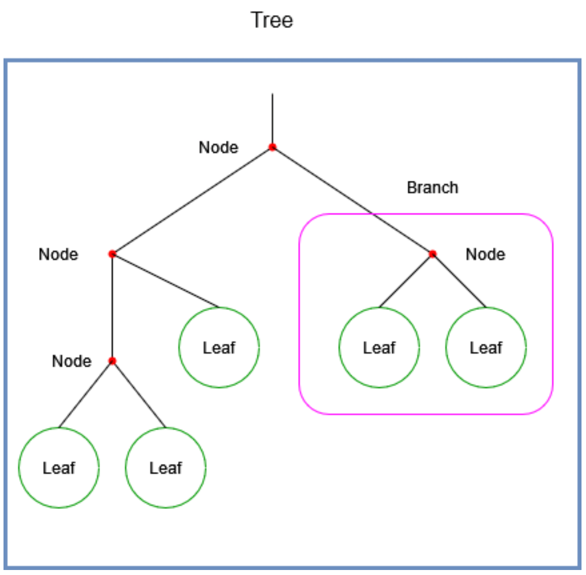


Figure 2.10: Example regression tree showing the component names.

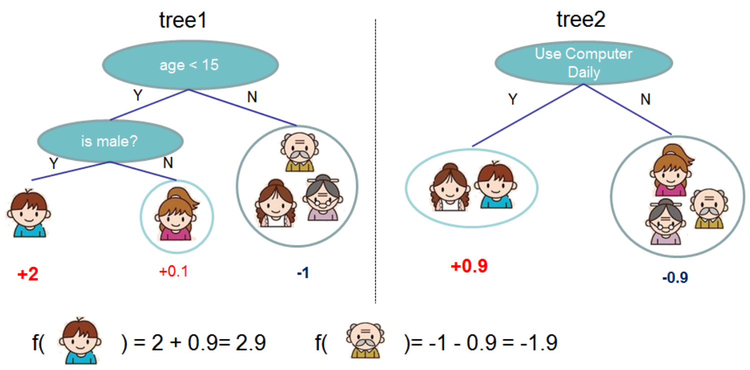


Figure 2.11: Tree ensemble with two trees for 5 distinct y [8].

2.4.1. Boosting

XGBoost is a method that uses gradient boosting trees. This means that multiple decision trees are built, where each new tree learns with the residual error of the previous trees. How the final prediction is realized is shown in Figure 2.11. In classifying the boy, his score is the score of the first decision tree '2' plus the result of the second tree '0.9'. The final score of this method is the addition of all these trees.

The first step in learning a tree is defining how we predict a value for y, the realization [8]:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i),\ f_k \in \mathcal{F} \tag{2.14}$$

In this formula, x is the feature set, where i corresponds to the index (row), here $i \in [0, n]$, where n is the total amount of samples. This formula states that each prediction instance $\hat{y}_i$ comprises of the sum of all created tree mapping structures f from boosting round 1 until the boosting round K, applied to the feature vector x_i, where K is the total amount of boosting rounds. The tree mapping function f is a function that maps the weight w of each leaf to the fixed structure of the tree q (in Figure 2.10 q would be the structure in the blue box). f therefore gives all the information of the tree to make a prediction, whereas q only describes its shape. All tree structures f_k are in the function space $\mathcal{F}$, which is the set of all possible regression trees. For a prediction of $\hat{y}_i$ at prediction step t, where t < K, it can be seen how Equation 2.14 extends:

$$\begin{aligned}
\hat{y}_i^{(0)} &= 0\\
\hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i)\\
\hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i)\\
&\dots\\
\hat{y}_i^{(t)} &= \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)
\end{aligned} \tag{2.15}$$

Here we see that the model learns by starting with predicting 0 for its target and learns by adding a regression tree at each timestep. The prediction at time t, $\hat{y}_i{}^t$, where t < k, therefore, comprises of the sum of all previous predictors $\hat{y}_i{}^{(t-1)}$ together with the next tree.

2.4.2. Loss function per boosting round t

In the previous step, we've seen how the model is set up and what processes are necessary to come to a prediction. What wasn't included is how the best tree structure at every boosting step t is decided, and how the model finds the best value for $\hat{y}_i$. In order to do this, an objective function $\mathscr{L}$ is necessary for training, which needs to be minimized. Next to this, XGBoost also has (hyper) parameters that try and reduce complexity. These are γ, which control the level of L1 regularization, and λ, controlling the L2 regularization. The goal of XGBoost's general objective function is to find a balance between a good predicting model and reducing the complexity of the model. This general objective function is defined as [8]:

$$\begin{gathered}
\mathscr{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{k=1}^{K} \Omega(f_k)\\
\text{where } \Omega(f) = \gamma T + \frac{1}{2}\lambda ||w||^2
\end{gathered} \tag{2.16}$$

Here l is a differentiable convex loss function of y_i, the target, and $\hat{y}_i$, the prediction. For this summation, i corresponds to the current row in the feature set x. $\Omega(f)$ is a term that adds constraints and regularization to the tree building process to reduce its complexity, where f corresponds to an independent tree structure with leaf weights w. For this summation, k refers to the instance in the amount of boosting rounds in the range [1, K], where K is the total amount of boosting rounds. In this case, Ω consists of the L1 and L2 regularization terms. T is the number of leaves in the current tree structure f_k, γ is a regularization parameter that can be set, λ is also a regularization parameter that can be set and w are the leaf weights.

Due to the nature of the above function, which includes tree structures as parameters, it cannot be optimized with traditional optimization methods in Euclidean space [8], contrary to a traditional optimization problem which is only dependent on a gradient. Therefore, the training of the model happens additively with t iterations, which is a parameter that can be set, which we've seen in Equation 2.15. This means that an initial tree is built, and new trees are added a total of t times towards what optimizes the loss function. Therefore Equation 2.16 can be specified as the following function, which is the general objective defined per boosting step t:

$$\mathscr{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i{}^{(t-1)} + f_t(\mathbf{x}_i)) + \Omega(f_t)$$
$$\text{where } \Omega(f_k) = \frac{1}{2}\lambda \sum_{j=1}^{L} w_j^2 + \gamma T \tag{2.17}$$

This means that every $\mathscr{L}^{(t)}$ is created by adding a tree $f_t(\mathbf{x}_i)$ which minimizes the loss function the most. The authors of [8] are of the opinion that a second-order approximation of a Taylor series is sufficient to approximate the loss function. This means that $l(y_i, \hat{y}_i{}^{(t-1)} + f_t(\mathbf{x}_i))$ can be defined as:

$$l(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)) = l(y_i, \hat{y}_i^{(t-1)}) + \frac{\partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})}{1!} f_t(\mathbf{x}_i) + \frac{\partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})}{2!} f_t^2(\mathbf{x}_i)$$

Where $f_t(\mathbf{x}_i)$ fulfills the function of $(x - a)$ in the traditional expression of the Taylor series. The second and third terms are also known as the gradient and hessian, henceforth they will be defined as g_i and h_i. The first term can be omitted as this term is constant. Substituting the second order Taylor approximation of $l(y_i, \hat{y}_i{}^{(t-1)} + f_t(\mathbf{x}_i))$ from above into Equation 2.17, and simplifying the second term as g_i and the third term as h_i leaves us with the following simplified objective at iteration t:

$$\tilde{\mathscr{L}}^{(t)} = \sum_{i=1}^{n} [g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i)] + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 + \gamma T \tag{2.18}$$

The last two terms are the regularization constants. Here, T is the total number of leafs and w is the leaf weight, where w_j is the weight on leaf j, where j is the leaf index $\in$ [1, T].

With the XGBoost method, Regularization happens per leaf, as it is the leaf, which produces a leaf score, not the individual samples within. The benefit of this is that regularization in this way poses restrictions on the weights of the leafs. This ensures that a seemingly decent prediction won't go into a terrible prediction from one iteration into the next. Also, it ensures that a tree doesn't overfit in some local conditions. In the equation just above we can recognize that the regularization terms on the end are summations over the leafs, as the summation index is the leaf index j. Therefore, to continue, the above equation needs to be rewritten as a summation over the leaves and then samples, instead of as over all the samples. Furthermore, the role which $f_t(\mathbf{x}_i)$ fulfills, is of an array with a mapping of all the different weights w_j on the different leaves for a tree structure q, which optimizes the loss function the most for the coming iteration. Therefore, when summing over all the leafs, the term $f_t(x_i)$ can be substituted by w_j. Next, S_j is defined as the set of indices of data points mapped to the j^{th} leaf sample collection in leaf j. Considering the above, $\mathscr{L}$ can be rewritten as:

$$\tilde{\mathscr{L}}^{(t)} = \sum_{j=1}^{T} [(\sum_{i \in S_j} g_i) w_j + \frac{1}{2}(\sum_{i \in S_j} h_i + \lambda) w_j^2] + \gamma T \tag{2.19}$$

To then find optimum leaf nodes, such that the objective $\mathscr{L}$ is minimized, we can differentiate $\mathscr{L}$ with respect to w and set it to zero to find the optimum weight w_j^*. After derivation and setting to 0, this leaves us the optimal leaf weight:

$$w_j^* = -\frac{\sum_{i \in S_j} g_i}{\sum_{i \in S_j} h_i + \lambda} \tag{2.20}$$

In case a loss function has no hessian, like the MAE, the hessian needs to be manually set to 1, which makes this equation the standard Gradient Boosting Regressor.

2.4.3. Learning per tree

After the general loss function per boosting round, the next step is determining how for every step t a tree will be learned. In Equation 2.14 we've seen that for every boosting round there exist multiple possible decision trees q, where f was the function that maps the weights of the leaves to the tree structure q. It is not possible to create every possible tree and iterate over all these trees to find the best tree structure q. Instead, the loss function will be used to grow a tree.

Substituting this optimal leaf weight w_j^* from Equation 2.20 back into Equation 2.19 we're left with the final form of the loss function for the fixed tree structure q:

$$\tilde{\mathscr{L}}^{(t)}(q) = -\frac{1}{2}\sum_{j=1}^{T}\frac{(\sum_{i\in S_j} g_i)^2}{\sum_{i\in S_j} h_i + \lambda} + \gamma T \tag{2.21}$$

Here q is $q(\mathbf{x})$, which is one fixed tree structure. This function will determine how good a tree structure $q(x)$ is. To approximate the best tree structure, a greedy tree growing algorithm is used, to build the individual tree for iteration t. It starts from a single leaf, which is split into two, dividing the data into two new leaves. In the next iteration, this step is repeated for both leaves. Doing this at every level, builds the tree layer by layer from the top. S_j is again the set of indices of data points assigned to the j^{th} leaf. This leaf will be split into two. Assume that S_L and S_R are the indices sets of the left and right leafs after the split of the original leaf where $S = S_L \cup S_R$. The loss reduction after the split is then given as:

$$\mathscr{L}_{split} = -\frac{1}{2}\left[\frac{(\sum_{i\in S} g_i)^2}{\sum_{i\in S} h_i + \lambda} - \left(\frac{(\sum_{i\in S_L} g_i)^2}{\sum_{i\in S_L} h_i + \lambda} + \frac{(\sum_{i\in S_R} g_i)^2}{\sum_{i\in S_R} h_i + \lambda}\right)\right] + \gamma \tag{2.22}$$

This can be interpreted as the original leaf score minus the score on the left and right. If the original leaf score minus the two new leafs is positive, that means the split is successful as the combined score of the two new leafs is reduced. This positive value is then multiplied by -0.5, which reduces $\mathscr{L}$, our objective. The last term here is the regularization term gamma (γ), which is the minimum loss reduction required to make a split. In case the split does increase the score, but only by a little amount smaller than γ, the split won't be executed as $\mathscr{L}_{split}$ will be positive, signaling that adding this tree would increase the loss, instead of reducing it. If the split were to be executed, the loss would of course be reduced. The purpose of this term is to prune the tree, reducing depth and complexity.

The last step in building the trees is giving them constraints to work in. Previous sections already saw λ, α, and γ, which are the regularization parameters that act on leaf weights to prevent overfitting. These two constraints are implicitly present in the objective function, but next to these, there exist a host of other hyperparameters in the XGBoost package [8] which help in constraining the tree growing process:

- the first is eta, also known as the learning rate. This value in the range [0,1] is multiplied by the leaf weights. This shrinks the weights, making the prediction improvement more gradual.
- max_depth is the maximum depth the tree may go in levels. High max depth levels often lead to overfitting on local minima.
- min_child_weight is the sum of all the hessian values in a leaf: $\sum_{i\in S} h_i$. There is a relation to the number of samples in a leaf and the sum of the hessian values, therefore this metric prevents leaves with only a small amount of samples, as this indicates possible overfitting.
- max_delta_step is a constraint on how large the output of a leaf is allowed to be. This makes the update step more conservative.
- subsample [0,1] is the ratio of the training instances which will be randomly sampled and used. The advantage of this is reducing overfitting as it reduces training instances randomly. At the same time, this can also be a disadvantage when training on a set with limited instances which you're trying to predict.
- colsample_bytree and colsample_bylevel are ratios [0,1] for how many columns (features) will be sampled. Colsample_bytree resamples every time a new tree is built and colsample_level resamples for every level. The colsample parameters work cumulatively, meaning that if colsample_bytree is 0.5 with 20 features, there are 10 features for colsample_bylevel to resample, of which it uses 5 features. [30]

3 Proposed Solution

The previous chapter gave context and introduced the responsibilities which the DSO faces in terms of congestion. This chapter will introduce a specific case in the Stedin grid, where congestion has developed over the past years. This case will be used as an illustration of the difference between using different forecast methods. The research question which will be answered in this chapter is:

What is a suitable model of the different cost components of congestion and how can that be incorporated into a loss function?

These questions will be answered by identifying the different cost components in congestion mitigation and showing an application of how this can be fit in a loss function.

3.1. Case: Middelharnis

One case where grid capacity has recently become an issue in the Stedin grid is substation Middelharnis, located in the Province of South Holland, municipality Goeree-Overflakkee. This region is historically sparsely populated, which means that the energy infrastructure up until recently was only equipped for minimal distribution capacity. In 2013 the Government together with the provinces made an agreement to install 6000 MW wind capacity before 2020 [31]. In 2020 they did a revision and now plan on installing it before 2023 [32]. For the province of South Holland, this came down to 725.5 MW of capacity of which 167.6 MW will be installed on top of the existing capacity on Goeree-Overflakee [32]. This new capacity needs to be connected to the current grid of Stedin. Stedin has done a congestion investigation [33], which showed significant shortages of available capacity on the 13 kV - 50 kV transformers. In this light, wind parks are now mostly connected to the 50 kV voltage level. However, the 50 kV - 150 kV transformers also have limited capacity available with the trend in growth. The expected capacity shortage on the 13 kV side by 2031 is expected to be anywhere from 31 - 73 MW and on the 50 kV side 30 - 112 MW. Stedin has realized these shortages and is investing in adding an additional 13 kV - 50 kV transformer, to be realized in 2024, and an additional 50 kV- 150 kV transformer that will be built after 2024 [4]. Furthermore, the cable between Middelharnis and Ooltgensplaat will be reinforced to be able to handle the addition of wind capacity in the eastern area of Goeree-Overflakkee.

3.1.1. Topology

To understand how this situation came to be and how it can be solved, it is important to look at the topology. Figure 3.1 shows the grid layout in Goeree-Overflakkee. From this image, it can be seen that wind parks connected to the 13 kV side only have the option to transport their energy via the 13-50 kV transformer on Middelharnis. On the 50 kV level, there are two possibilities: distribution via Ooltgensplaat to the east or transformation to 150 kV into the TenneT transportation grid. However, at the moment, there is no capacity to distribute eastwards, meaning all power on the island will arrive and leave via the 150 kV line. As of right now, the 50/150 kV transformer capacity is sufficient and no congestion is expected. However, with the expected realization of another 167.7 MW of wind [32], Stedin realized the current capacity is insufficient and will realize more transformer capacity starting from 2024 [4].

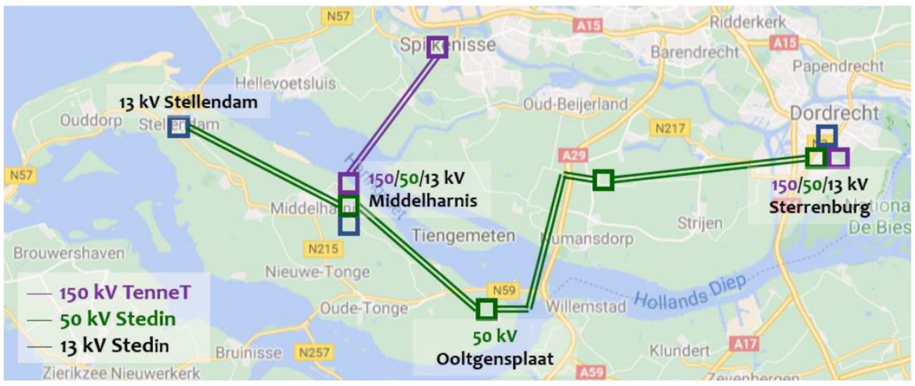


Figure 3.1: 50 kV grid lay-out with the 50/13 kV transformers of Stedin and 50/150 kV transformers of TenneT on Goeree-Overflakkee [33].

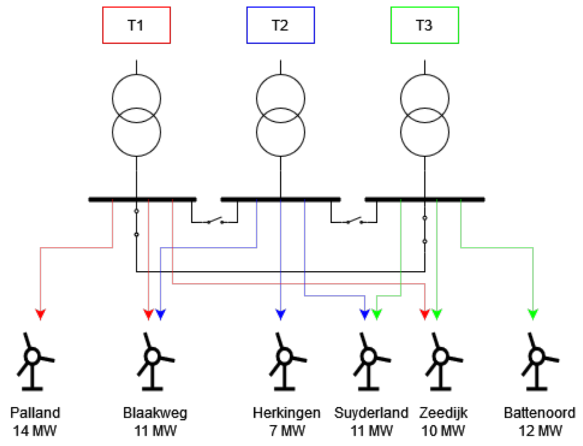
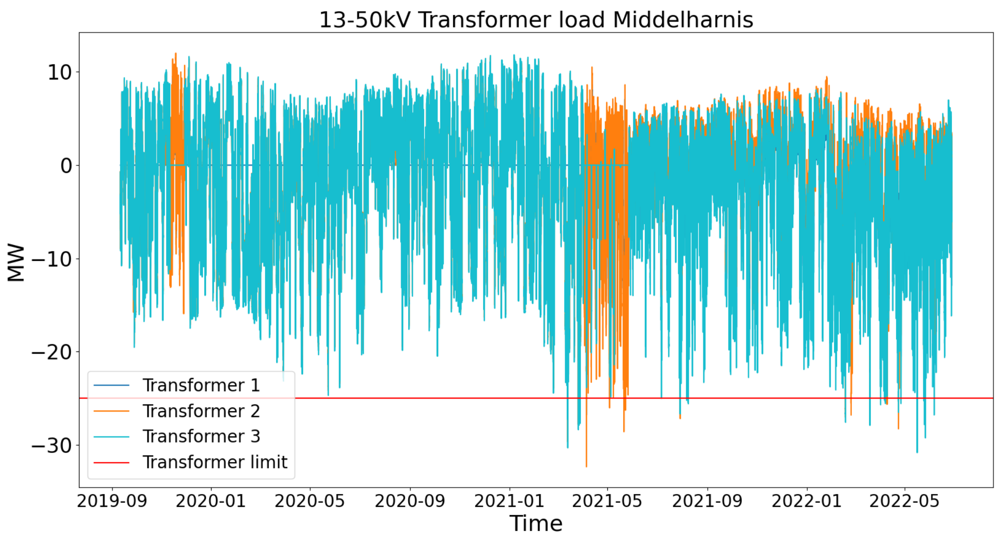


Figure 3.2: Schematic representation of the transformer connections on the 13 kV side of substation Middelharnis. Transformers 1 and 3 are connected in parallel, and transformer 2 is islanded. Three wind parks have two separate cables connected to different transformers, to be able to do load balancing. These cables may never be used at the same time.

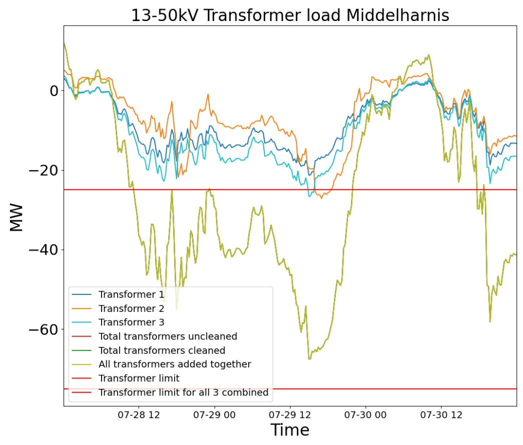
Figure 3.2 shows the connected wind generation on the three 13-50 kV transformers of Middelharnis. If we sum up their power, it reaches 65 MW, which is not yet surpassing the safe operating range of the transformers. However, the reason it surpasses the safe operating region of 75 MW is due to the decentrally connected photovoltaics by households and farmers. The solar generation was 31.5 MW in April of 2021 [33, 34]. Resulting in a maximum of 96.5 MW under extreme circumstances. In reality, this extreme wont be seen, as weather conditions resulting in maximal production of solar and wind power are not likely to co-exist. Also, inhabitants on the island also contribute to the load. For 2021 and the first part of 2022, transformer loading up to 80 MW due to generation has happened a handful of times. The expectation is that these occurrences will be more frequent, and the shortage above the transformer limit will increases to 31 - 73 MW by 2031.

The three transformers are not connected in parallel. Although all three transformers are compatible with each other, connecting them all together in parallel gives a short circuit power that is higher than the protection can mitigate in case of a fault. Instead transformers 1 and 3 are connected in parallel, while transformer 2 is operated islanded from the rest. With the need to expand from 2 to 3 transformers and the protection problem in foresight, the newer wind parks have been connected by two independent lines to two transformers. This allows grid operators to balance the generation over the three transformers, as one transformer is islanded from the rest.

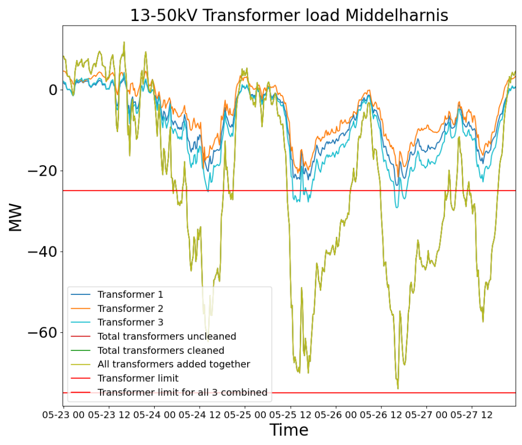
However, even with this tool at Stedin's disposal, there are still instances when transformers operate at or above their rated limit of 25 MVA, due to excessive generation. An example can be found in figure Figure 3.3. In Figure 3.3a it can be seen that transformers 2 and 3 are operating above their rated limits in a number of instances, even after an additional transformer was added in May of 2021. Figure 3.3b & 3.3c, show these instances in more detail.



(a) Transformer loading limits are 25 MW as indicated by the red line. Prior to May of 2021, only two transformers were in use, explaining the overloading the months prior as new wind parks were connected. Subsequent overloads are caused by congestion due to generation.



(b) Excerpt of July 2021 showing in more detail that an individual transformer can be overloaded, without there being an overload on substation level. It's possible Stedin switched a wind park from transformers 1 & 3 to 2, to relieve pressure, although unsuccessfully as transformer 2 is now congested.



(c) Excerpt of May 2022 showing in more detail that an individual transformer can be overloaded, without there being an overload on substation level.

Figure 3.3: Individual transformer loading of the three 13-50 kV transformers of substation Middelharnis, where positive loading is consumption and negative loading is generation.

Transformer overloading effects

Since it is expected that the transformers in this substation, but also in other areas of the Stedin grid, will be periodically overloaded, it is meaningful to take stock of the risks associated with overloading. Transformer overloading will lead first and foremost to a temperature rise inside the transformer. It is this temperature increase that is responsible for most failure modes. The following risks are associated with temperature increase due to loads in excess of transformers rated loads [35]:

- Reduced dielectric strength and integrity due to gas or fluid produced by the windings. This could lead to dielectric breakdown as a conducting path can occur through the oil. This eventually leads to a destructive breakdown.
- Loss of life through deterioration of insulating material and mechanical parts.
- Thermal expansion of transformer parts due to high temperature could lead to eventual permanent deformation, which is a contributing factor to mechanical or dielectric failures.
- If the temperature of the top oil exceeds 105 °C, the oil inside the transformer expands. There exists a probability that the expansion is severe enough that it triggers the pressure relief device, expelling the oil. Loss of oil brings further risks upon cooling.

This list is not exhaustive, as it reflects only the risks of minor overloading, not the risks due to extreme overloading as in fault conditions. As said above, these risks are a direct result of temperature. Temperature increase only happens gradually. To reach harmful temperatures, a transformer needs to be overloaded by an extreme amount in a short time, which happens during fault conditions, or structurally overloaded for longer periods of time, in the order of hours-days-weeks (depending on the percentage of overloading) [35–37].

According to Montsinger [36], transformers can be overloaded with 3% for each 10% that the system load is below 100% prior to overloading. System load is defined as the average load over the past 24 hours. Per one degree below 30 °C ambient temperature, an additional 1% per 1°C can be added to the baseline 3% per 10% below 100% system load. This would mean a transformer operating at 50% load at 30 °C can be overloaded to $100 + 5 \times 3 = 115\%$. At 20 °C, the same transformer can be overloaded to $100 + 5 \times 3 + 30 - 20 = 125\%$. This is only valid when assuming a similar daily load pattern, where the oil has the same temperature at the start of every day.

Nichols [37], has, prior to Montsinger, done research into the loss of life due to transformer overloading. He concludes that a transformer at 20 °C ambient temperature at 50% loading can be overloaded at 125% for 1 hour before the loss of life occurs. At prior 100% loading, this reduces to 15 minutes. When these times for a 125% overload are exceeded, loss of life is between 0.002-0.6% of total life. The current active IEEE standard C57.91-2011 [35], comes to a similar conclusion, albeit with higher resolution equations.

In conclusion, the overloads which are currently seen on transformers in substation Middelharnis don't pose a significant short-term risk, as they are minor and within acceptable limits. However, if the transformers encounter the projected overloads, they are at a significant higher risk of accelerated aging, resulting in failures if not addressed by maintenance and/or premature replacement.

3.1.2. Operation

As was seen in the previous section, there are currently transformers that are being overloaded, making the area serviced by substation Middelharnis a congestion area. As laid out in subsection 2.1.1, there is a procedure with the ACM that needs to be followed when encountering (possible) congestion as a system operator [5]. Stedin followed this procedure for Middelharnis and let a 3rd party perform an independent investigation [33]. D-Cision B.V. [33] reviewed the following four criteria as set forth by the Netcode article 9.5 5th paragraph [5]:

1. the grid operator considers congestion management feasible with the available infrastructure;
2. the grid operator considers performing the act of congestion management possible in its grid operation;
3. the period of structural congestion is longer than one year but shorter than 4 years, and;
4. the affected congested area has enough potential participants for performing congestion management.

D-Cision B.V. found points 1 and 3 feasible, but points 2 and 4 impossible. For point 2, they argue that to perform congestion management it is necessary to be able to remotely gauge and control the 13 kV switching gear, which isn't possible in the current distribution grid. Furthermore, connections of generation (especially solar on the roof) need to be able to be remotely disconnected, which is not allowed, as this would also impact the consumer's ability to use electric load. For point 4 they argue multiple reasons. Firstly, the congestion size should not be higher than 120% of the distribution capacity. Middelharnis is estimated to be at 131%, making this an infeasible solution. Secondly, they argue that not enough flexible capacity is currently available. As renewable energy is excluded from needing to participate in congestion management, this only leaves one flexible load of 4 MW, which is not sufficient. Lastly, they investigate if compulsory congestion management is an option. Renewable energy is again excluded from congestion management. The one 4 MW load is insufficient to aid in congestion management. Therefore, on basis of points 2 and 4, congestion management as defined by the Netcode version in their report is not possible.

Unfortunately, there is no straightforward solution to the existing congestion. Those causing the congestion cannot be forced to participate in solutions to reduce the problems. Therefore, solutions need to be found on a voluntary basis. Currently, Stedin asks wind park operators on a very short-term notice (15-1 minutes ahead) to voluntarily reduce their generation in exchange for a disturbance of service compensation. This is not a long-term or sustainable solution, as it is expensive for Stedin, and it isn't balance neutral. Therefore, Stedin has the intention to consult (future) market parties to see if a more structural solution can be found.

However, a solution through market parties hinges on the ability of Stedin to be able to predict when congestion will happen. Stedin has as of yet no accurate methods to perform a day-ahead forecast for this location, making market-based solutions difficult to implement. Stedin has signaled this [15] and is actively seeking solutions [38].

3.2. Proposed loss function

In the previous chapter, a number of loss functions were introduced as well as previous research trying to find a loss function that improves over the standard methods. In forecasting the load on Middelharnis, the standard functions which will be used will be MSE and Pinball. As mentioned before, MSE is the most popular loss function in regression problems, and will therefore be serving as a benchmark. Pinball loss is used to bias the model into predicting the generation peaks, which are a type of outlier. The last loss function will be a custom-made function, to see if this can improve the forecasting over the standard functions.

3.2.1. Definition of loss

The first step is defining a custom loss function, where we need to define what constitutes 'loss'. If we look at Stedin's goal for this prediction, it isn't to minimize statistical metrics, but rather the operational costs.

Normally there are no costs associated with load forecasting for DSOs, as they are not responsible for processes, like economic dispatch or optimal battery storage processes, that deal with energy markets. This means that if the loading on a transformer station stays within its limits there are no costs for the DSO. The costs only happen outside this region, when congestion happens.

As described in Figure 2.1, the DSO has some tools to resolve its congestion. The tools, which are relevant to the Middelharnis case, are voluntary curtailment, direct action, and redispatch. Here, voluntary curtailment and direct action have the same result. The only difference between the two is that for voluntary curtailment the generating party can shut off their generation themselves and in direct action, this is done for them by the DSO. The cost associated with both is known as the disturbance of service fee [39]. For the 13 kV level, there are two compensations: one for wind parks larger than 10 MW and one for below. Above 10 MW, the compensation is €0.35 per contracted kW, meaning that for a 10 MW wind park this comes down to €3500 per incident, lasting up to 8 hours. Below 10 MW, there is a one-time fixed compensation of €910. As shown in Figure 3.2, the connected wind parks are all around this size, except for the windpark in Herkingen. If the disturbance of service is longer than 8 hours, an additional compensation is given. These won't be quantified, as the peaks causing congestion last around 5h, which is below 8 hours.

The redispatch method is fulfilled by the market GOPACS and contrary to the disturbance of service compensation, GOPACS prices are not a fixed number. GOPACS costs are also highly location dependent as it takes bids only from a pre-specified location. If in that location there are only a few market participants, this could drive up the price. Ideally, the historic GOPACS prices for the forecasting location are known, to use them as metrics in a loss function. Unfortunately, Stedin has not had congestion areas in the past and has therefore no historic GOPACS pricing [40]. Since the price is location-dependent, historic data from other DSOs don't transfer over to the Middelharnis location.

Therefore, Stedin has done a study on what the expected costs will be for a wind park on the 13-kV side of Middelharnis. This study was done in preparation, as Stedin is expecting to start using GOPACS in the coming months. Cihangir Martin [41] has simulated wind park bidding behavior given constraints, like location, weather data, market conditions, etc. From this, she infers what the lost opportunity cost is if Stedin announces congestion and asks market participants for redispatch bids. The wind parks will, at minimum, bid their lost opportunity cost in the redispatch market. As GOPAS is an intraday market structure, down regulating wind on Middelharnis, means that another area needs to either increase generation or decrease load. A different market party there will also bid a price to buy the opportunity to generate more or reduce load. These two bids will not match and this difference will be paid by the DSO. Determining the exact GOPACS price from this is difficult, as it depends on the number of market parties in the region to resolve the congestion. To pin down a cost for GOPACS, the assumption is made that the wind park will exactly bid its lost opportunity cost, and the other side of the bid will bid almost nothing. This situation creates the maximum costs for Stedin, as it needs to compensate for the price difference between the two.

In Figure 3.4, and Table 3.1, we see a summary of the results for lost opportunity cost for wind parks of [41]. From the figure, we can infer that the lost opportunity cost is relatively condensed between 180 €/MWh and 280 €/MWh. The fact that the mean is skewed closer to the 75th quantile is due to the numerous higher outliers.

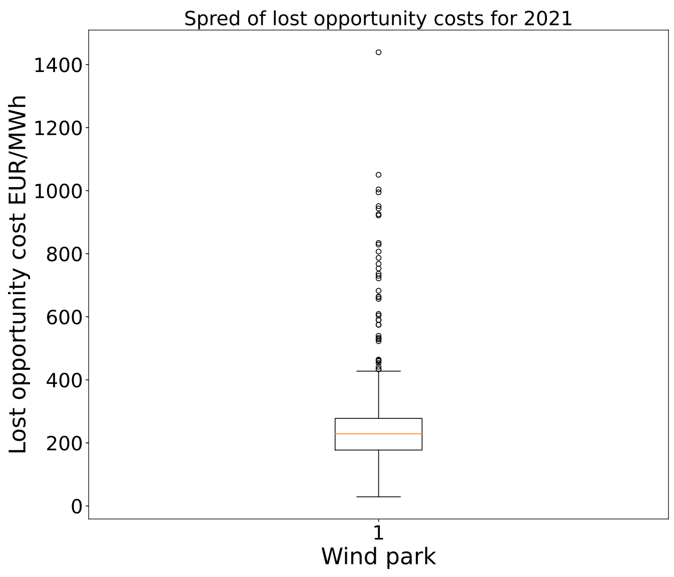


Figure 3.4: Summary of simulation results, where the spread of lost opportunity cost for wind park Palland, connected to the 13-kV of Middelharnis, is simulated for the year 2021. [41]

Table 3.1: Summary of the simulation statistics, belonging to Figure 3.4. [41]

Statistic	Value
count	415
mean	267.18 €/MWh
std	164.73 €/MWh
min	28.98 €/MWh
25% quantile	177.17 €/MWh
50% quantile	228.83 €/MWh
75% quantile	277.61 €/MWh
max	1438.92 €/MWh

The costs in Figure 3.4 are associated with a set of very specific circumstances. Since the scope of this thesis is limited to forecasting load, there will be no attempt to predict future GOPACS prices to use as a metric for the loss function. Instead, we draw up a set of scenarios, based on different GOPACS costs, to predict on:

- Mean scenario: using the mean cost value of 267.18 €/MWh.

- Minimum scenario: using a minimum cost value of 70 €/MWh.

Next to the scenario with the mean cost, a minimum scenario will give another perspective. The assumption is that when wind farms produce the most, the energy price will be lower as higher costing units are pushed out of the merit order. As these instances coincide with congestion, since higher wind park production causes more loading on transformers, the energy price, therefore the lost opportunity cost for these wind parks will be lower. This will result in a lower GOPACS price. The value of 70 is chosen as it lies below the mean and is 10 times smaller than the defined congestion fee, which is introduced in Table 3.2.

Table 3.2: Summary of the different types of cost, occurring for location Middelharnis. As we'll see in chapter 4, the data resolution is in 15 min, meaning all costs here are implemented by dividing them by 4.

Type of Cost	When this cost occurs	Value of the cost [€/MWh]
None	If the load or generation do not exceed the transformer limits	0 €
GOPACS	If the load and/or generation exceeds the transformer limits	· 70 €/MWh · 267.18 €/MWh
Congestion	If the load and/or generation exceed the transformer limits and if insufficient amounts of flexible power are bought	3500 €/incident or 700 €/MWh

Table 3.2 summarizes the costs that are encountered for the 13-50 kV transformers of Middelharnis. The takeaway from this table is that the costs are dependent on each other. If we predict a situation where the loading on the transformer will exceed its limit by an x amount of MW, there is a range for the final cost. The cheapest resolve is buying the x amount of MW on the GOPACS market. The most expensive resolve is not buying any power on the GOPACS market and paying the congestion fee of 3500 €/incident. A compromise is where power is bought on the GOPACS market, but it its not sufficient. The congestion fee of 3500 €/incident is transformed into 700 €/MWh, as the average peak lasts about 5 hours.

3.2.2. Proposed custom loss function

The cost is a function of the combination of the three different types of cost, but also a function of time. In an ideal world, we have perfect knowledge of the transformer loading for the next hours and if it exceeds a limit, we will buy this amount on the GOPACS market so we never have to pay a disturbance of service fee. In reality, we need to work with a prediction of the loading and base our GOPACS buying off of this. If we then arrive at the congested hour, we either underpredicted the amount of load and pay that amount in disturbance fees, or we've overshot and bought too much on GOPACS. This process is visually shown in Figure 3.5.

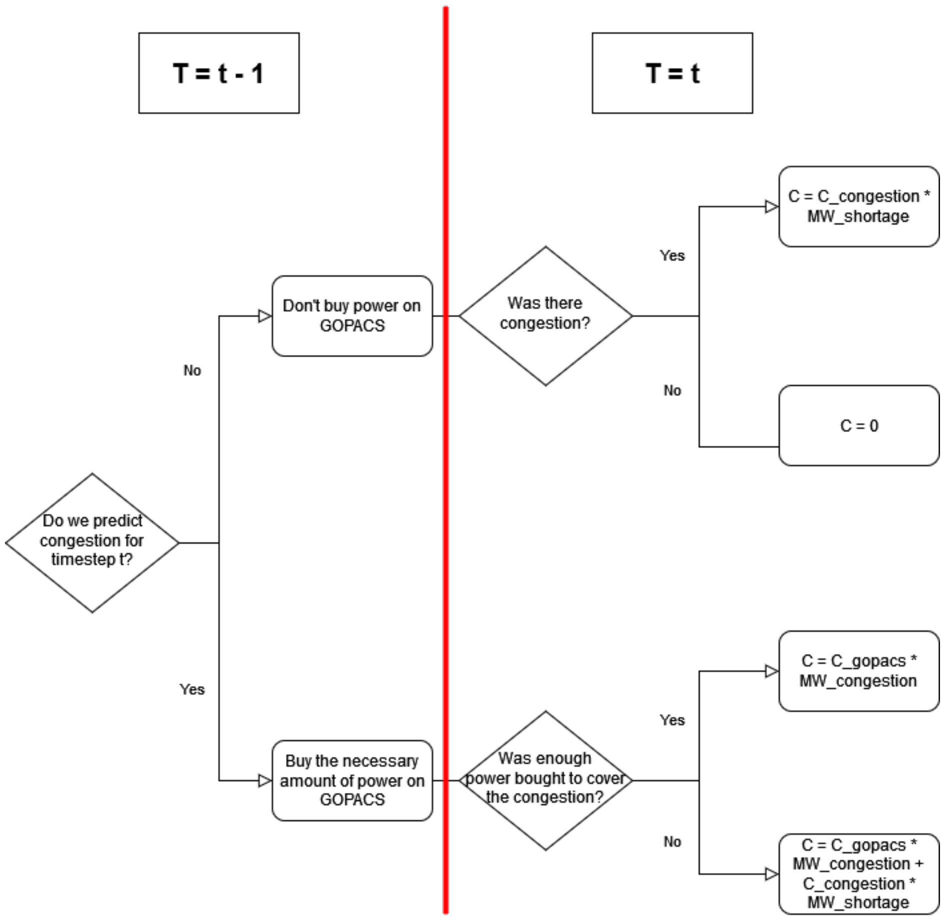

T = t - 1
T = t
Do we predict congestion for timestep t?
No
Yes
Don't buy power on GOPACS
Buy the necessary amount of power on GOPACS
Was there congestion?
Was enough power bought to cover the congestion?
Yes
No
C = C_congestion * MW_shortage
C = 0
C = C_gopacs * MW_congestion
C = C_gopacs * MW_congestion + C_congestion * MW_shortage

Figure 3.5: Flowchart showing the conditions to end at the final determination of cost for an hour t.

As Figure 3.5 shows, if we are correct in our prediction of the presence of congestion, the cost is what we expected. If we're wrong, for the case where we predicted no congestion, but there was congestion, we pay the disturbance of service fee. In the case where we did predict congestion, but we didn't predict high enough, we pay the disturbance of service fee for the part we didn't already solve via the GOPACS market.

As described in chapter 2, the standard loss functions all depend on the prediction error $\hat{y} - y$. If we were to add costs in the mix, this would be the costs of the prediction error, as we've seen Li and Chiang [9], Kebriaei, Araabi, and Rahimi-Kian [28], and Zhang, Wang, and Hug [10] do in subsection 2.3.4. This has worked for them, as the cost was dependent on the prediction error. However, as we've seen from Figure 3.5, the cost at a timestep is not only dependent on the forecast error, but also on what actions we took the previous timestep. This means that the custom loss function is not dependent on $\hat{y} - y$, but on $\hat{y}$ and y independent of each other.

In formulating the loss function, the following variables are introduced:

- y, the target variable, which is the true transformer loading
- $\hat{y}$, the prediction of the target variable
- k, the transformer limit. For Middelharnis this is 75 MW, but for the purpose of this research, it is set at 50 MW.
- C_{gopacs}, the GOPACS costs in €/MWh, for illustration purposes set at 5, instead of any of the values from Table 3.2.
- $C_{congestion}$, the disturbance of service fee in €/MWh, for illustration purposes set at 15, instead of the value from Table 3.2.
- ε, a constant, for illustration purposes set at 2.5 €/MWh here.
- $L(\hat{y}, y)$, the total cost minimization objective in €/MWh

In formulating the proposed loss function, the loss function, gradient, and hessian are given. This is because, as seen in section 2.4, the implementation is done solely through the gradient and hessian.

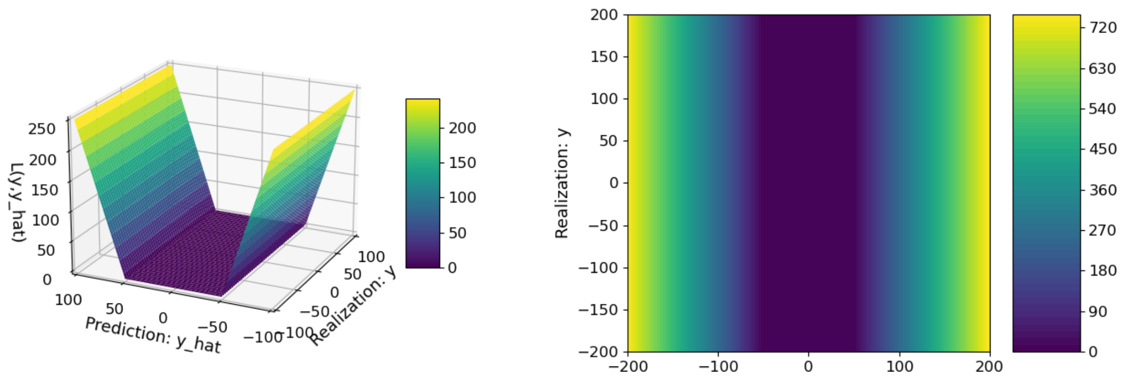
Also, for the cost components C_{gopacs} and $C_{congestion}$ their values for costs are the same for congestion due to load and generation. Adjusting the function to be able to distinguish between congestion due to oad or generation is easily done by multiplying the load and generation-related terms in the formulas each by a different cost component for C_{gopacs} and/or $C_{congestion}$. This is not necessary for the researched case of Middelharnis as only generation-caused congestion occurs here.

Gopacs component of the loss function

The first step in defining the loss function starts with defining the cost at time step $t-1$.:

$$\begin{aligned} L\left(y,\hat{y}\right)_{gopacs} &= C_{gopacs}\cdot\left(\max(\hat{y}-k,0)+\max(-\hat{y}-k,0)\right)\\ \frac{\partial L\left(y,\hat{y}\right)_{gopacs}}{\partial\hat{y}} &= C_{gopacs}\cdot\left(\mathbb{1}(\hat{y}-k)-\mathbb{1}(-\hat{y}-k)\right)\\ \frac{\partial^2 L\left(y,\hat{y}\right)_{gopacs}}{\partial\hat{y}^2} &= C_{gopacs}\cdot\left(\delta(\hat{y}-k)-\delta(-\hat{y}-k)\right)\end{aligned} \tag{3.1}$$

This part of the loss function has two components to it. Both do the same: determine the distance to the transformer limit k, but one is for the generation region and the other for the load region. The max function ensures that the function only returns an amount to be bought on gopacs if $\hat{y}$ is above k and that the formula works for both generation and load predictions of $\hat{y}$. This can be seen in Figure 3.6. Figure 3.6a is the representation of Equation 3.1 above. Figure 3.6b shows the same function, but then as contour plot in 2D.



(a) (b)

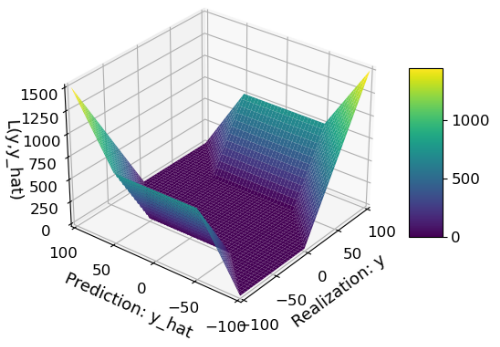
Figure 3.6: Loss function for only the GOPACS cost part. Left the function in 3D and right the contour plot in 2D.

Congestion part of the loss function

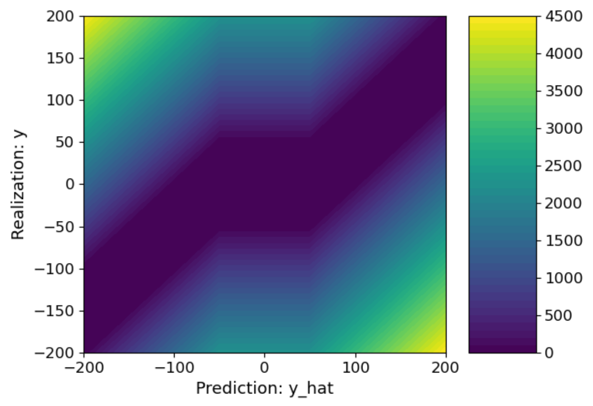
The next step is when we arrive at time step t, where we did or didn't buy flexible power on GOPACS. If we bought power on GOPACS the amount was the sum of buying load and/or generation capacity: $max(\hat{y}-k,0)+max(-\hat{y}-k,0)$. The cost for not predicting $\hat{y}$ (enough) is:

$$
\begin{aligned}
L(y,\hat{y})_{congestion} =& C_{congestion}\cdot \quad [\max(y-k-\max(\hat{y}-k,0)+\max(-\hat{y}-k),0),0)+\\
& \max(-y-k-\max(-\hat{y}-k,0)+\max(\hat{y}-k,0),0)]\\
\frac{\partial L(y,\hat{y})_{congestion}}{\partial \hat{y}} =& C_{congestion}\cdot \quad [\left(\mathbb{1}(\hat{y}-k)+\mathbb{1}(-\hat{y}-k)\right)\cdot\\
& (\mathbb{1}\left(-y-k-\max(-\hat{y}-k,0)+\max(\hat{y}-k,0)\right)-\\
& \mathbb{1}\left(y-k-\max(\hat{y}-k,0)+\max(-\hat{y}-k,0)\right))]\\
\frac{\partial^2 L(y,\hat{y})_{congestion}}{\partial \hat{y}^2} =& C_{congestion}\cdot \quad [\left(\delta(\hat{y}-k)+\delta(-\hat{y}-k)\right)\cdot\\
& (\mathbb{1}\left(-y-k-\max(-\hat{y}-k,0)+\max(\hat{y}-k,0)\right)-\\
& \mathbb{1}\left(y-k-\max(\hat{y}-k,0)+\max(-\hat{y}-k,0)\right))+\\
& \left(\mathbb{1}(\hat{y}-k)+\mathbb{1}(-\hat{y}-k)\right)\cdot\\
& (\delta\left(-y-k-\max(-\hat{y}-k,0)+\max(\hat{y}-k,0)\right)-\\
& \delta\left(y-k-\max(\hat{y}-k,0)+\max(-\hat{y}-k,0)\right))]
\end{aligned}
\tag{3.2}
$$

This function is again in two parts for the generation and for the load region. This function determines the disturbance of service fee height. It subtracts from the realization y, the transformer limit k, and the power bought on GOPACS. Figure 3.7b helps with interpretability the best for this equation. We can see that inside the transformer region, the costs are still 0. Above this region, in y direction, we see that these costs are purely defined by the congestion cost, as this cost increase is parallel to the $\hat{y}$ axis. In the upper right corner, we see that if we are below the line $\hat{y}=y$ where $\hat{y}>y$, y was over forecasted. This is the consequence of buying enough power on GOPACS. Above this line, where $\hat{y}<y$ we've under-forecasted, and the further we move from the line $\hat{y}=y$, the higher the cost will be. On the left side of the transformer limits in the upper left corner, there was a severe underforecast of $\hat{y}$. $\hat{y}$ here is assumed negative, while the realization y, was positive. Due to this under forecast, power was bought on GOPACS in the wrong direction. For example, if the realization y is 100 MW, and the prediction $\hat{y}$ was -100, this means that 50 MW was bought on GOPACS. This means that at time = t, the load on the transformers is 100 + the extra 50 is 150 MW. This then needs to be mitigated with the congestion fee. This is why that region is extremely expensive.



(a)



(b)

Figure 3.7: Loss function for only the congestion costs part. Left the function in 3D and right the contour plot in 2D.

Gopacs and Congestion parts together

Lastly, we bring the two functions $L(\hat{y},y)_{gopacs}$ and $L(\hat{y},y)_{congestion}$ together:

$$
\begin{aligned}
L(\hat{y},y) =& C_{gopacs}\cdot && [\max(\hat{y}-k,0)+\max(-\hat{y}-k,0)]+\\
& C_{congestion}\cdot && [\max(y-k-\max(\hat{y}-k,0)+\max(-\hat{y}-k),0),0)+\\
& && \max(-y-k-\max(-\hat{y}-k,0)+\max(\hat{y}-k,0),0)]\\
\frac{\partial L(y,\hat{y})}{\partial \hat{y}} =& C_{gopacs}\cdot && [\mathbb{1}(\hat{y}-k)-\mathbb{1}(-\hat{y}-k)]+\\
& C_{congestion}\cdot && [(\mathbb{1}(\hat{y}-k)+\mathbb{1}(-\hat{y}-k))\cdot\\
& && (\mathbb{1}(-y-k-\max(-\hat{y}-k,0)+\max(\hat{y}-k,0))-\\
& && \mathbb{1}(y-k-\max(\hat{y}-k,0)+\max(-\hat{y}-k,0)))]\\
\frac{\partial^2 L(y,\hat{y})}{\partial \hat{y}^2} =& C_{gopacs}\cdot && [\delta(\hat{y}-k)-\delta(-\hat{y}-k)]+\\
& C_{congestion}\cdot && [(\delta(\hat{y}-k)+\delta(-\hat{y}-k))\cdot\\
& && (\mathbb{1}(-y-k-\max(-\hat{y}-k,0)+\max(\hat{y}-k,0))-\\
& && \mathbb{1}(y-k-\max(\hat{y}-k,0)+\max(-\hat{y}-k,0)))+\\
& && (\mathbb{1}(\hat{y}-k)+\mathbb{1}(-\hat{y}-k))\cdot\\
& && (\delta(-y-k-\max(-\hat{y}-k,0)+\max(\hat{y}-k,0))-\\
& && \delta(y-k-\max(\hat{y}-k,0)+\max(-\hat{y}-k,0)))]
\end{aligned}
\tag{3.3}
$$

Which are represented in Figure 3.8. Figure 3.8b shows that if there is congestion, which is outside the region where y is between [-50,50], it is comparatively cheap to over forecast $\hat{y}$. The upper right corner shows this, as above $\hat{y} = y$, where $\hat{y} < y$, costs are higher, due to underforecasting than below this line. This might lead one to believe that it is beneficial to always overforecast, as even when excess power is bought on GOPACS it is still cheaper than paying the congestion fee. Caution needs to be taken against this sentiment, since, as seen in Figure 3.8b, if we overforecast, we risk forecasting in the wrong direction, which is the most expensive cost. In reality, the machine learning model wants to find the point which minimizes the costs most, which will be at $\hat{y} = y$. However, it will bias its prediction to prefer forecasting more towards over forecasting by a small amount as this will lead to the least amount of costs.

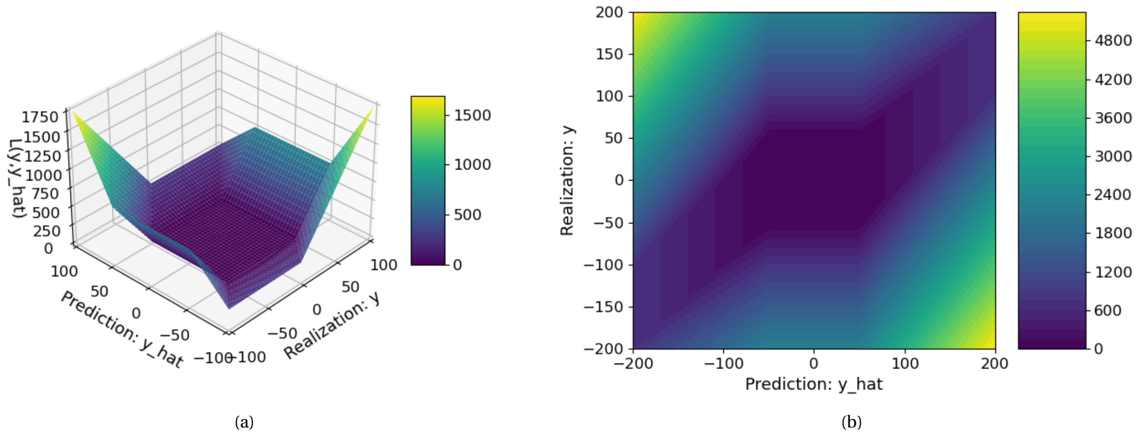
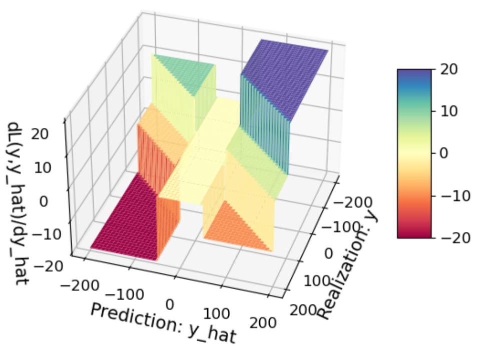


Figure 3.8: Loss function for both gopacs and congestion components. Left the function in 3D and right the contour plot in 2D.

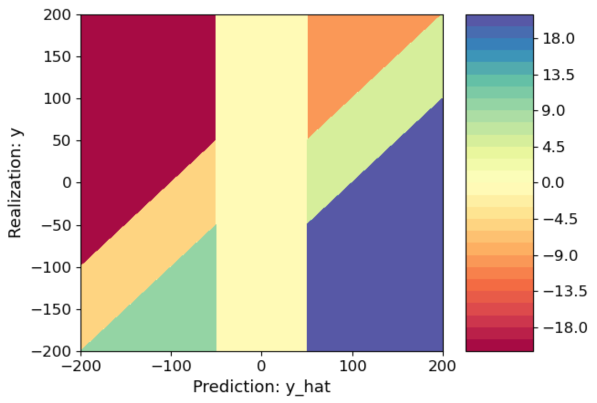
At this point, the function is finished from a model standpoint. However, from a training perspective, this function brings with it an issue. The loss function in section 2.4 is expressed in terms of the gradient and hessian. Where the gradient is taken in the $\hat{y}$ direction, as can be seen in Equation 3.3. In Figure 3.8b, we can see that inside the transformer region for $\hat{y}$, the value of the loss function only depends on y, meaning the gradient in the $\hat{y}$ direction in this region is 0. The gradient plots in Figure 3.9 show this.

The goal of the loss function is to be minimized to zero, which is achieved through making the gradient 0. Usually, this is only when $\hat{y} = y$. Figure 3.9 shows a wide band where the gradient is 0. The issue with this is twofold: firstly, the algorithm will never optimize to where $\hat{y} = y$, as it will stop optimizing once it hits the area where the gradient is 0. This is also not necessary, as inside the transformer limit region it isn't required to make an accurate prediction. Secondly, the initial prediction is $\hat{y}^0 = 0$, as seen in section 2.4. As we can see in Figure 3.9, the gradient for $\hat{y} = 0$ is 0. The result of this, is that it will never optimize to $\hat{y} = y$, as the algorithm believes it has finished, as the gradient is 0. To remedy this, one could set the first prediction $\hat{y}$ to 100. Our starting point in Figure 3.9b would then be the dark blue region. If the realization y would be negative, the model tries to optimize to the negative side at the left, as this minimizes the loss function. However, once it reaches $\hat{y} = 50$, the gradient is 0. At that point, it will think that it has optimized the objective function and found the minimum, whereas, in reality, the optimum lies on the other side of this band, meaning it will never be reached.

Furthermore, this function has a hessian that is largely 0. As seen in Equation 2.21, the loss function depends on both the gradient and hessian, meaning the hessian can't be 0. To mitigate this, the hessian for all instances can be manually set to 1. If this is done, the XGBoost algorithm of finding the optimum will revert back to normal Gradient Boosting Trees [8].



(a)



(b)

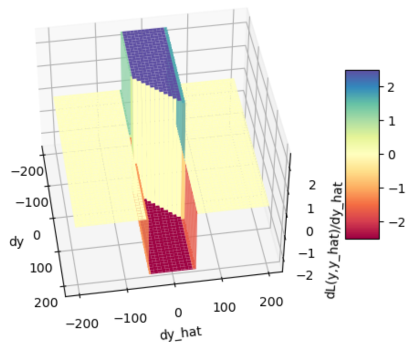
Figure 3.9: Gradients corresponding to Figure 3.8 from Equation 3.3 in $\hat{y}$ direction.

Epsilon function

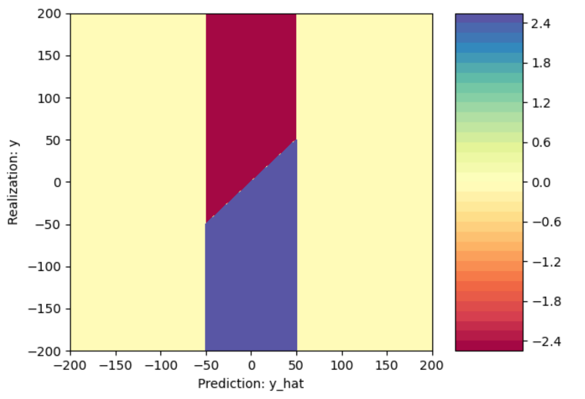
To remedy the issue with a wide band where the gradient is 0, we can add an overlay function. This function will only affect the region inside the transformer limit, therefore not influencing the prediction. This function is defined as follows:

$$\begin{aligned} L(\hat{y}, y)_\varepsilon &= \varepsilon \cdot \int_{-\infty}^{\hat{y}} \mathbb{1}(k-\hat{y}') \cdot \mathbb{1}(k-\hat{y}') \cdot \left[\mathbb{1}(\hat{y}'-y) - \mathbb{1}(y-\hat{y}')\right] d\hat{y}' \\ \frac{\partial L(\hat{y}, y)_\varepsilon}{\partial \hat{y}} &= \varepsilon \cdot \mathbb{1}(k-\hat{y}) \cdot \mathbb{1}(k-\hat{y}) \cdot \left[\mathbb{1}(\hat{y}-y) - \mathbb{1}(y-\hat{y})\right] \end{aligned} \tag{3.4}$$

This term is again defined for two regions: the region above $\hat{y} = y$ and below. The first part of the formula selects the boundaries. This function is only defined between $\hat{y} = -50$ and $\hat{y} = 50$. The second part selects the region: below or above the line $y = \hat{y}$. The result of this function can be seen in Figure 3.10.



(a)



(b)

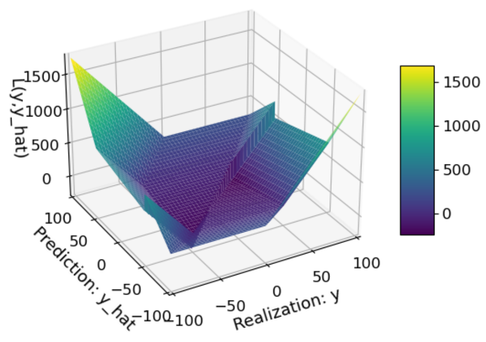
Figure 3.10: Gradient plots for the ε part of the loss function.

Final loss function

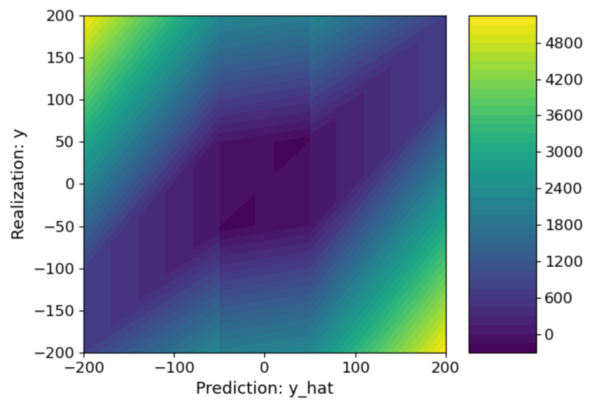
If we combine the loss function from Equation 3.3 with Equation 3.4 we are left with the following complete equation to define the loss in terms of the present costs:

$$
\begin{aligned}
L(\hat{y}, y) =& C_{gopacs} \cdot & \left[\max(\hat{y}-k,0)+\max(-\hat{y}-k,0)\right]+ \\
& C_{congestion} \cdot & [\max(y-k-\max(\hat{y}-k,0)+\max(-\hat{y}-k),0),0)+ \\
& & \max(-y-k-\max(-\hat{y}-k,0)+\max(\hat{y}-k,0),0)]+ \\
& \varepsilon \cdot & \int_{-\infty}^{\hat{y}} \mathbb{1}(k-\hat{y}')\cdot\mathbb{1}(k-\hat{y}')\cdot\left[\mathbb{1}(\hat{y}'-y)-\mathbb{1}(y-\hat{y}')\right]d\hat{y}' \\
\frac{\partial L(y,\hat{y})}{\partial \hat{y}} =& C_{gopacs} \cdot & \left[\mathbb{1}(\hat{y}-k)-\mathbb{1}(-\hat{y}-k)\right]+ \\
& C_{congestion} \cdot & [\left(\mathbb{1}(\hat{y}-k)+\mathbb{1}(-\hat{y}-k)\right)\cdot \\
& & (\mathbb{1}\left(-y-k-\max(-\hat{y}-k,0)+\max(\hat{y}-k,0)\right)- \\
& & \mathbb{1}\left(y-k-\max(\hat{y}-k,0)+\max(-\hat{y}-k,0)\right))]+ \\
& \varepsilon \cdot & \mathbb{1}(k-\hat{y})\cdot\mathbb{1}(k-\hat{y})\cdot\left[\mathbb{1}(\hat{y}-y)-\mathbb{1}(y-\hat{y})\right]
\end{aligned}
\tag{3.5}
$$

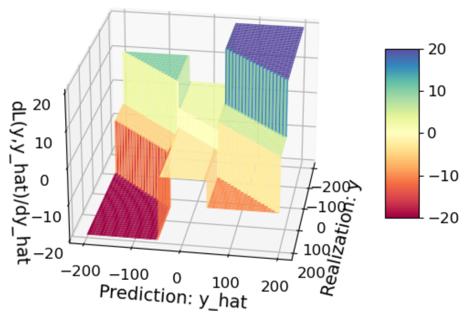
This loss function now has a gradient in all regions, which can be used by the model to optimize to $\hat{y} = y$. This can be seen in Figure 3.11c and Figure 3.11d. This means that the loss function is not as smooth anymore, as can be seen in Figure 3.11a, but now there is a defined gradient everywhere, which is unique and at a minimum at $\hat{y} = y$.



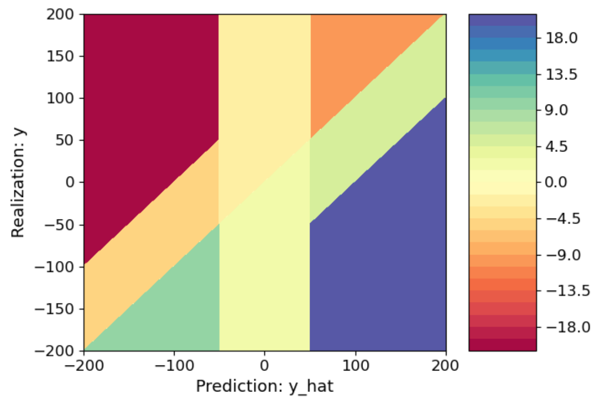
(a)



(b)



(c)



(d)

Figure 3.11: Final custom loss function (above) and the corresponding gradients (below).

3.2.3. Additional evaluation metrics

As mentioned in subsection 2.3.2, to evaluate a model, the same function can be used as on which the model was trained. In the case of the proposed loss function, this would be the total cost. However, the total cost will, with the same prediction, be different as there will be two different costs used for C_{gopacs}. To be able to compare the two together, a normalized cost metric is proposed from [10]. This normalized cost is the Forcasting Error Percentage Cost (FEPC) and is defined as:

$$FEPC(\hat{y}_i, y_i) = \frac{C(\hat{y}_i) - C * (y_i)}{C(y_i)} \cdot 100\% \tag{3.6}$$

This function quantifies the difference in cost between prediction and the ideal cost scenario and normalizes this. The ideal cost is the cost if we would have a perfect forecast, equal to the realization y. In this situation, no congestion fee would ever have to be paid en we will have bought the exact amount of power necessary on GOPACS.

4 Methodology

This chapter will provide the methodology followed to perform to provide a day-ahead load prediction on substation level on the 13 kV side of the 13-50 kV substation Middelharnis. Figure Figure 4.1 provides an overview of the workflow from data collection up to the final result. The subsections in this chapter will follow the colors of the workflow, where the purple corresponds to section 4.1, which will detail the data collection, cleaning, and transformation, the blue corresponds to section 4.2, which will explain the transformation from data to features, the green corresponds to section 4.3, which explains the model tuning and generation, and the orange is the final prediction an results which are shown in chapter 5.

4.1. Data

The first step in any Machine Learning model is data collection. This section details the data available from different sources. Next, the cleaning steps to deal with any missing or incorrect data, and lastly the necessary transforms to effectively use the data as features.

4.1.1. Data sources

At Stedin, all data is stored centrally in a SQL database, called 'Datalake 2.0'. This includes measurements from the Stedin grid, as well as other data sources necessary to make day-ahead load forecasts.

The day-ahead load prediction will be done on substation level. Therefore the measurement data of the individual transformers are summed together to the substation level, instead of taking them individually.

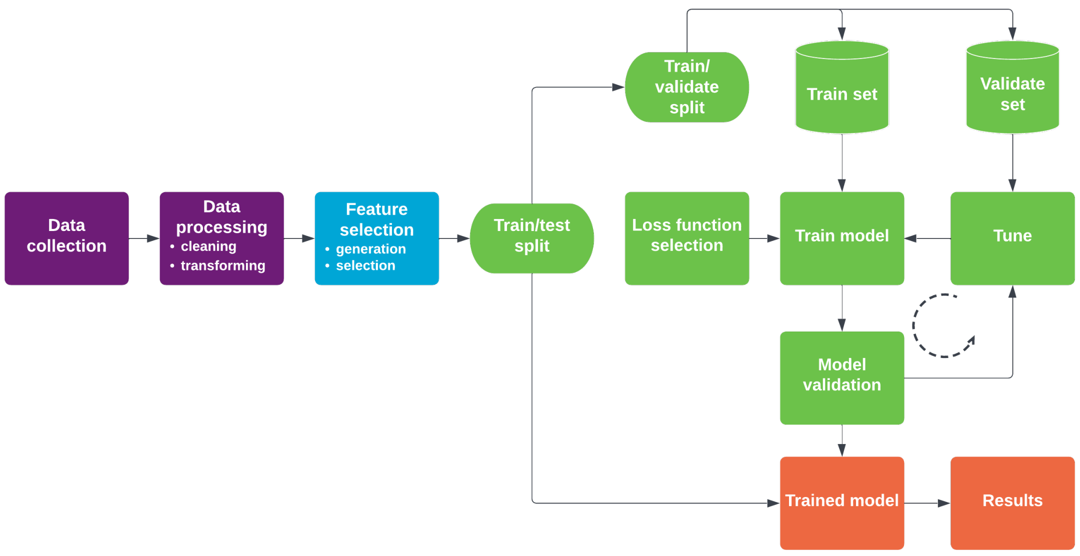


Figure 4.1: Workflow for building, tuning, evaluating and predicting the machine learning models based on different loss functions.

Table 4.1: Description of the different data used for day-ahead prediction of the load on the three combined 13kV transformers of substation Middlharnis (MIDH) with their source, resolution, and availability in real-time. KNMI breakdown per weather station in Table A.1, consumer profile breakdown in Table A.2. The stopdate for all data was 28-06-2022.

Variable	Source	Resolution	Start date	Real-time availability
MIDH 13 kV transformer 1, 2, 3 min	Stedin SCADA measurements	15 min	04-09-2019 13:15h	No, 3-6 hour delay
MIDH 13 kV transformer 1, 2, 3 mean	-	-	-	-
MIDH 13 kV transformer 1, 2, 3 max	-	-	-	-
Wind direction	KNMI [42]	1 hour	Depending on station: 1951, 1981, 1991	No
Wind speed hour mean	-	-	-	-
Wind speed last 10 min	-	-	-	-
Highest gust	-	-	-	-
Temperature	-	-	-	-
Dewpoint temperature	-	-	-	-
Sun duration	-	-	-	-
Radiation	-	-	-	-
Cloud cover	-	-	-	-
Precipitation duration	-	-	-	-
Precipitation sum	-	-	-	-
Air pressure	-	-	-	-
Visibility	-	-	-	-
Humidity	-	-	-	-
10 different consumer load profiles	NEDU [43]	15 min	2016	Yes

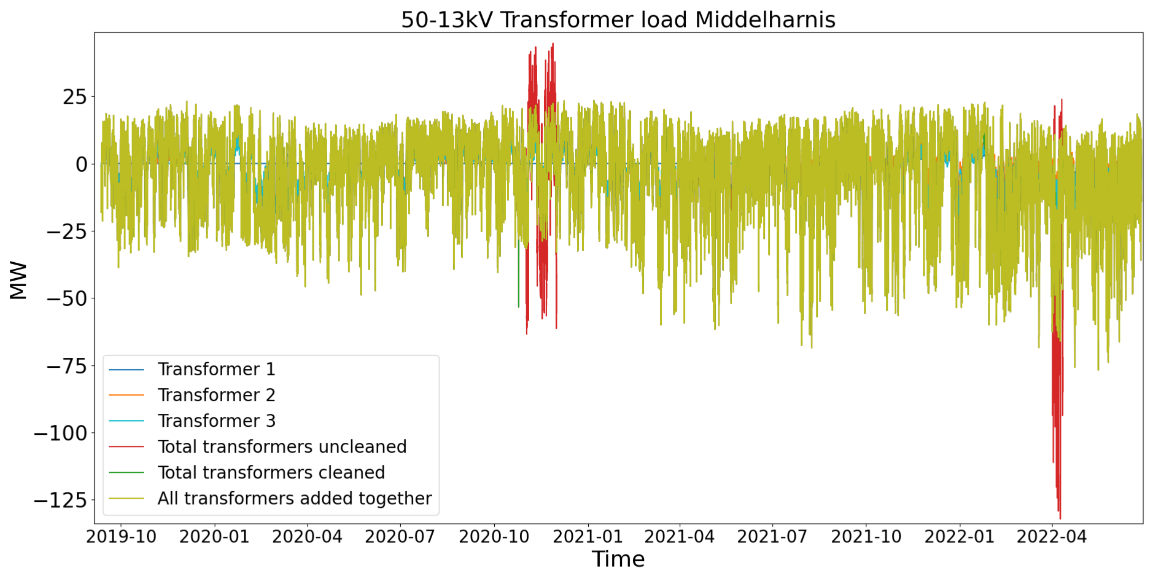
4.1.2. Data cleaning

After data ingest, the next step is checking the data for errors. Both the KNMI and NEDU sources had complete data, with no apparent outliers. However, upon inspection of the transformer data, there were some irregularities. In an earlier ingest of this data, there were approximately 2 months of data missing. However in a later ingest, this issue was corrected externally. The errors which are left include three types of erroneous data: incorrect data due to double data, abnormal patterns due to maintenance and abnormal patterns due to congestion management.

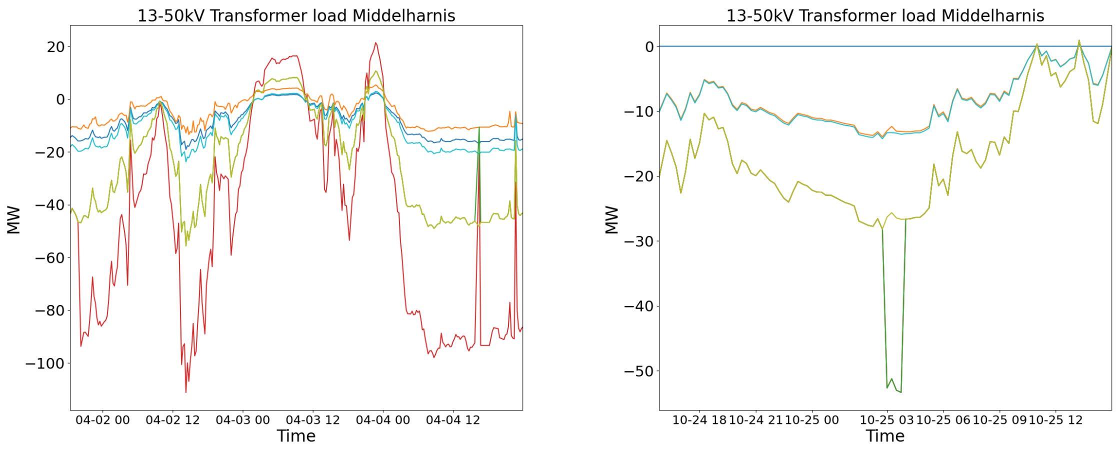
Erroneous data

The measurement data from Middelharnis had two distinct data errors: double data with the same timestamp and identical values, and double data with the same timestamp, but unique data values. The first data error is easily cleaned by calling Pandas drop_duplicates() as the double data rows are identical. The second error is unaffected by this command because the indices are equal, but the data is not. These instances had their values lie close together, but the data varied enough from row to row that picking the 'correct' instance proved difficult. As the amount of data points affected by this error was very minor, only 17 instances, which is in total 4h 15min in total, on 3 occasions randomly throughout the dataset, the decision was made to leave this errors as the expectation is that these errors will not majorly impact the model.

The effect of these errors can be seen in Figure 4.2. Here, the yellow line is the three transformers added together manually, which is computationally heavy, compared to Pandas group command, but is error-free. It can be seen that the uncleaned data ingest, which is the red line, is following the same pattern as the correct yellow line, but magnified. The green is the data ingest cleaned from the double identical data, but with the remaining instances of double unique data.



(a) Overview plot of the entire dataset. The yellow is the ideal situation without errors. The red spikes are instances with double data, which are grouped together in the data ingest and green spikes are residual errors after cleaning due to more complicated errors. Figure 4.2b shows an excerpt of uncleaned data and Figure 4.2c an excerpt of residual errors after cleaning.



(b) Excerpt of April 2022 showing the start of the faulty data on the left side due to double identical data. On the right, after 12h on 04-04 a small green spike is seen, which is due to a more complicated data error.

(c) Three combined transformers load spike due to double, non-identical data, which isn't filtered out.

Figure 4.2: Plots showing the different data errors in the transformer data of Middelharnis. The 3 different transformers are plotted separately, as Total Transformers, which is then added together through Pandas before and after data cleaning, and as All transformers added together, which is manually summing the individual transformers.

Maintenance

The next source of deviation is maintenance. Unfortunately, there is no database detailing when maintenance takes place and if customers are asked to reduce their grid usage. Maintenance after May 2021 can be detected by looking at the individual transformer loading and seeing if one out of the three transformers is at zero load. In talks with the engineer responsible for Middelharnis, he explained that after May 2021, the 'reserve' transformer was added in permanently to help with the growing wind power generation. Before May 2021, maintenance can be detected by checking if either of the standard transformers 1 & 3 is at 0.

However, this method comes with a significant caveat. As this substation is already operating near its limit [33], to be able to perform maintenance, wind parks connected to the 13 kV side were asked to reduce generation (regardless of loading or weather conditions) to make sure that maintenance could be carried out safely. Unfortunately, these arrangements have not been recorded in any database. Also, these arrangements are for a longer period, whereas maintenance is carried out in working hours and not necessarily for the full planned maintenance period. Maintenance can wrap up quicker than planning, or run over.

For these reasons, no good detection mechanism or assumptions can be made about when maintenance occurs and what the load pattern would have been if no maintenance occurred. Therefore, this data deviation is not corrected.

Congestion management

The last source of deviation which needs to be cleaned is congestion management. As of 2022, substation Middelharnis had, according to the operators at Stedin, the potential to be in congestion. To prevent this, a connected wind park was asked, on several occasions, to scale down production. This can be seen in Table 4.2. The actual transformer loading would have been higher if not for this curtailment. Therefore, the curtailed amount needs to be added back to load data.

Unfortunately, there are no profiles in MW known, only the duration and the curtailed production in kWh. The 19th of March peak was not corrected as the start/stop and duration were not known. For the other three peaks, the curtailed production was converted to MWh and divided by the duration to provide an average MW rate. This was added back to the data between the start and stop timestamps, rounding to the closest timestamp, and respecting the duration which is closest to the actual duration.

Table 4.2: Congestion instances where a wind park connected to the 13 kV side of Middelharnis was asked to reduce production. Data was provided by the wind park itself.

Timestamp start	Timestamp stop	Duration	Curtailed production [MWh]
19-03-2022	19-03-2022		47.1
23-04-2022 12:51	24-04-2022 6:43	17h:51min:53s	21.314
16-05-2022 16:48	16-05-2022 18:10	1h:22min:19s	1.620
26-05-2022 13:50	26-05-2022 19:23	5h:33min	23.577

4.1.3. Data transformation

After the cleaning stage, the last step in the data preparation is transforming the data into a coherent dataset that can be interpreted by a model. To achieve this, three more steps are necessary: data resampling, scaling, and detrending.

Resampling

Part of a coherent dataset is ensuring that the data has the same resolution and that the data lines up. This is already the case for the measurement data and the consumer usage profiles, but not yet for the weather data. Since KNMI data is the historic measurement, as compared to future predictions, the value is given at the end of the hour, instead of as a 'prediction' at the start. This means the KNMI data doesn't start at 0:00, but at 01:00. To mitigate this, the entire dataset is shifted by an hour. Next, the resolution of the data is in hours instead of 15 minutes. Therefore, the data is resampled to 15, and data is filled through linear interpolation.

Scaling

No scaling was applied, because decision tree-based models are not sensitive to the scaling of data. Also, they are immune to outliers, which is often one of the arguments for performing a scaling step [44].

Detrending

The last step in data transformation is removing the trend from the prediction target. This is a step that less critical when using methods like Support Vector Machines or Neural Networks as it doesn't provide an advantage in terms of accuracy or computational time and might in some cases be harmful [45]. The reason these models don't need detrending is that they have the ability to extract linear combinations of features from the data, thereby identifying a trend. This is something a decision tree-based method cannot do [44].

Real-world energy data is almost never stationary as the population changes organically and recently a lot of new renewable generation is added [31]. In Figure 4.3 this effect can be seen. The top two figures capture the maximum load and generation per week in the dataset of 2.5 years. As can be seen from the linear fit line and the same y-axis scaling for both, the load does increase, although not as quickly as generation increases throughout these 2.5 years. The load growth can be attributed to population increase and the population using more electricity, whereas the generation increase is due to the addition of multiple wind parks over these 2.5 years.

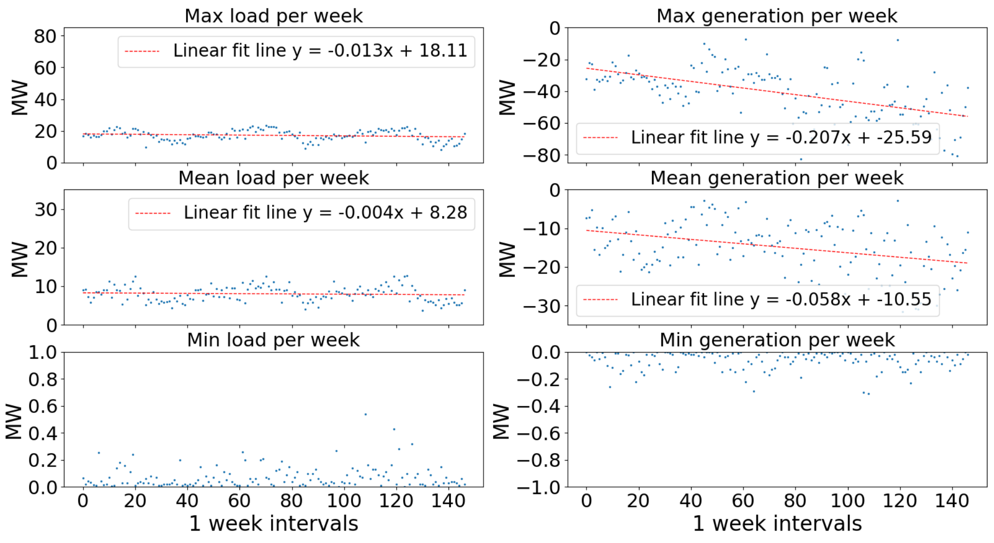


Figure 4.3: Trend lines of the three 13-50 kV transformers of Middelharnis together from September 2019 (week 1) to June 2022 (week 145). The data was split for load and generation, where left is the load, and right is the generation. The weekly value is plotted by taking the minimal, average, and maximum value of that week for either load or generation.

To correct a trend in the prediction target, time series decomposition can be used [45]. This time series is an example of an Additive Model, which can be defined as:

$$Y_t = \bar{Y}_t + S_t + T_t + \epsilon_t \tag{4.1}$$

Here the time series Y_t is made up of its average component $\bar{Y}_t$, a cyclical seasonal component S_t, a linear trend T_t, and lastly noise ϵ_t which has an average of 0.

However, this method cannot be used for the current model due to the choice of the loss function. The proposed loss function in Equation 3.5 is dependent on the transformer limit. When removing the trend, the load will never reach a set transformer limit. To mitigate that, the transformer limit would no longer be a single number, but an array starting at the limit and decreasing with time. This would be extremely computationally expensive since for every time step in the train set, there would be a different gradient map. Therefore, this standard method of detrending is abandoned.

Since the model will capture the seasonal effect with the time features (which will be introduced in Table 4.3) the only strictly necessary detrending step is to compensate for the linear trend component.

Instead of taking away the trend, an approach is used to make the model aware of the trend. This is done by fitting a linear line to two years of the training set. This line is not fitted to the entire train set, as the seasonality component influences the trend if not all 4 seasons are used. The line is fit between September 2019 and September 2021 on the mean generation per week. The reason that only this set is chosen is that the drift in load is already captured in the consumer usage profiles.

Next, the trend can be defined as $y = ax + b$. Where a is the slope of the trend and x is the number of weeks. The set $a \cdot x$ is then resampled from week resolution to 15-minute resolution by dividing a by 672. Then, an array is made by multiplying a with an array [0, ..., T], containing the quarter index from quarter 0 to T, the total amount of quarters in the train set. This array is called 'coefficient' and is used to multiply a subset of features with it, and as a direct feature, as shown in Equation 4.2.

$$\text{Feature set} = \mathbf{x} + \text{coefficient} \cdot \mathbf{x}_j + \text{coefficient} \tag{4.2}$$

Here **x** is the final feature set after all the steps detailed in section 4.2, and $\mathbf{x}_j$ is a subset of the total feature set. This subset excludes the time features, as well as the consumer usage profiles, as these profiles already capture a trend in the load. The remaining features are the weather data and any features corresponding to the transformer loading.

For the test set, the coefficient array is extended to time indices: [0, ..., T, T+1, ... , H]. Where H is the total amount of quarters from the train and test sets added together. It is then truncated to time indices [T+1, ..., H], which are the time indices for the test set. The feature set for the test set is then constructed similarly to the train set.

4.2. Features

The next step after the data gathering and transformation is splitting it into the target to be forecasted, y, and the feature matrix **X**. Some data can be used directly as features, like the consumer load profiles, while other data types are used to generate new features, or are transformed before being used as features. The reason for doing this is to gather more information with the same data by helping the model find relations, that are otherwise too deeply hidden. After feature generation, feature selection is done, as not all features are beneficial for the prediction. This is because not all features will have a sufficient relation to the target y, and will therefore hamper the model in finding a relation between features and the target.

4.2.1. Feature generation

With the data available, four types of features can be generated:

- derived features: new features generated by applying a transform to an existing feature.
- lagged features: features delayed in time.
- windowed features: features which are a maximum/minimum/average over a specified time period.
- combined features: combinations of the above transforms.

Derived features

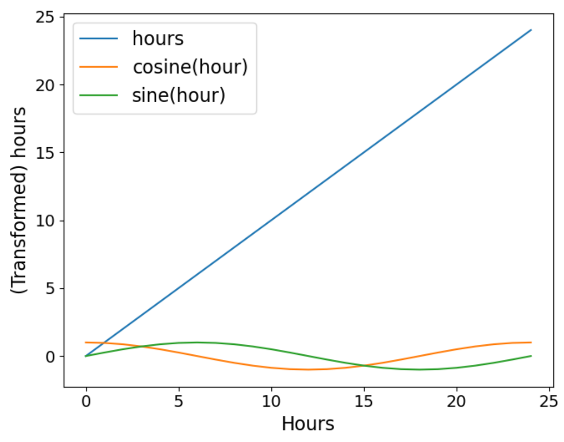
This first category consists of time features. The index of the data is a timestamp, but this is not yet a feature. Therefore, we derive time features from the index. Some time features are easily extracted, for instance: 'year', and 'is it daylight savings time, yes/no', as they can be directly interpreted by the model without needing a transformation. For the other time features the difficulty lies in letting the model understand that January comes after December and that after 23:00h comes 0:00h. In the eyes of an ML model, these are the furthest apart they can be, whereas, in reality, they come after one another. A transformation will take care of this cyclical relation.

To make this transformation, there are two commonly used options: categorical encoding and cyclic representation. The issue with categorical encoding methods like one-hot encoding is that they add a lot of sparsity to your feature matrix. For instance, a feature 'month', will instead of being one column, be 12 columns with all zeros, except for when it is that column's month, then it will be one. If we were to only do this for the number of months, this would still be a manageable situation, but extrapolating this to the day of the month/year or second of the day, one could see how there are a lot of empty columns. Apart from adding sparsity, the ML model will also not recognize that there is a relation between those 12 columns. It will see the different months, but it cannot infer that data in June and July have a relation to each other. XGBoost was partly developed to be able to handle sparsity, which set it apart from other tree-based methods [8], but it is still not the best practice to do this if there are other alternatives.

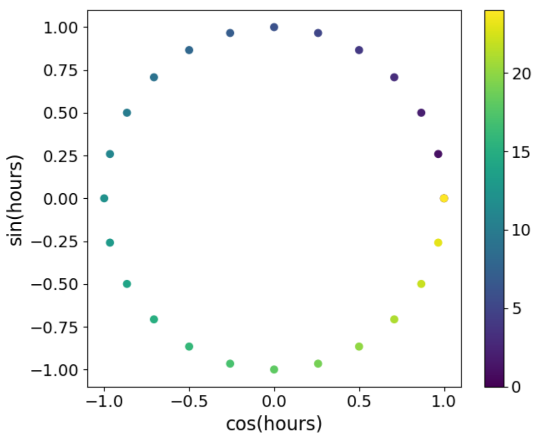
The better alternative is to use cyclic representation. This creates far fewer features and leaves intact the cyclical relation. The way to create these features is to transform them using trigonometric functions. The transform for the hour transform using cosine and sine is defined as follows :

$$\begin{aligned}\text{cosine feature} &= \cos\left(2\pi \times \frac{\text{hour data}}{24}\right)\\ \text{sine feature} &= \sin\left(2\pi \times \frac{\text{hour data}}{24}\right)\end{aligned} \tag{4.3}$$

Unfortunately, we still need two features to fully map one time feature. This is because if we were to only use cosine, then there would be no distinction between, for instance, 05:00 and 17:00. This can be seen in Figure 4.4a. Therefore, it is necessary to add a sine function as well, to make this distinction. What we're left with is a complete cyclical mapping of time, as seen in Figure 4.4b.



(a) Normal hours and cosine, sine functions on the $0-2\pi$ cycle



(b) The created cosine, sine coordinates system

Figure 4.4: Hour representation using trigonometric encoding.

The features derived from the timestamp can be found in Table 4.3. The reason this list is not more extensive is that adding features like 'month', 'season', 'hour of day', or 'minute of day' does not provide more information than already provided by the listed features. Month and season can also be covered by day of the year, which provides the same information and is higher in resolution than a month, and hour and minute of day are covered by the same reason through the second of the day.

Table 4.3: Derived features from existing variables.

Variable	Derived Feature
Timestamp	Is daylight savings time
	Year
	Second of day sine
	Second of day cosine
	Day of the week sine
	Day of the week cosine
	Day of the year sine
	Day of the year cosine

Lagged features

Next are the lagged features. We can't provide the model of the transformer load that we wish to predict directly, as we don't have this data. Instead, what we can give the model is the transformer loading of the last hour, the same time yesterday, or the same time last week. These time frames are usually a good predictor of what will happen from now to tomorrow [46]. Long time frames are usually useful as there is often a strong weekly or daily repeating pattern. The shorter time frames, for instance from yesterday to 1 hour ago will improve prediction accuracy, as they capture the varying conditions with shorter time horizons, like the influence of weather conditions. Table 4.4 provides an overview of the lagged variables.

Table 4.4: Lagged features from existing variables. For a day-ahead prediction, only the bold features are available.

Variable	Lagged Feature
Sum of Transformer 1, 2, 3 (mean)	Transformers 1h offset
	Transformers 12h offset
	Transformers 1 day offset
	Transformers 2 day offset
	Transformers 7 day offset

Windowed features

Next are the windowed features. These are generated by taking a window, for example from 12:00h to 15:00h, and applying a transform there. One example is taking the average value of that window. So for timestamp 15:00h, the value would be the average over the last 3 hours, and for timestamp 15:15h it would be the average value over the window 12:15-15:15h. Next to the average, this was also done for the minimum and maximum values for select variables.

The reason to include windowed features with averages, minimum or maximum values is that it provides insight into patterns. Weather features might follow the same global pattern on a daily or 3 hourly scale but might have different instantaneous patterns. This makes it more difficult for a machine learning model to find a relation [46]. Windowing these events prevents the model from overfitting on the event by reducing the resolution and thereby reducing variance, which helps identify similarly, but not necessarily identical events. Table 4.5 provides an overview of all windowed features. During previous research, Stedin has found that only the sum of the precipitation and temperature benefit from more windowed options, as not all weather features carry the same predictive values. Extending only these two features gives the same information as doing it for all the features. By not doing it for all the features, it reduces the complexity of the model.

Table 4.5: Windowed features from existing variables.

Variable	Windowed Feature
Wind speed hour mean	Wind speed 3h mean Wind speed 1 day mean
Wind speed last 10 min of hour	Wind speed last 10 of hour 3h mean Wind speed last 10 of hour 1 day mean
Highest gust	Highest gust 3h mean Highest gust 1 day mean
Temperature	Temperature 3h min Temperature 3h mean Temperature 3h max Temperature 1day min Temperature 1 day mean Temperature 1 day max
Dewpoint temperature	Dewpoint temperature 3h mean Dewpoint temperature 1 day mean
Sun duration	Sun duration 3h mean Sun duration 1 day mean
Radiation	Radiation 3h mean Radiation 1 day mean
Cloud cover	Cloud cover 3h mean Cloud cover 1 day mean
Air pressure	Air pressure 3h mean Air pressure 1 day mean
Precipitation duration	Precipitation duration 3h mean Precipitation duration 1 day mean
Precipitation sum	Precipitation sum 3h min Precipitation sum 3h mean Precipitation sum 3h max Precipitation sum 1 day min Precipitation sum 1 day mean Precipitation sum 1 day max
Visibility	Visibility 3h mean Visibility 1 day mean
Humidity	Humidity 3h mean Humidity 1 day mean

Combined features

Lastly, there are the combined features. These features consist of combinations of lagging and windowing. These are useful for finding patterns in recent data. An example is applying it to the target. By windowing the target, for instance, a 3h mean, and then lagging it a day, this provides the model with more opportunity to see the relation between past transformer load and the load it will predict [46]. The reason for this is that by windowing the target, the resolution of the data goes down to, in the example 3h, thus reducing variance. It provides the model with broader ranges where a current developing trend can fall in when a trend is not easily captured by the instantaneous values. Next, the window is lagged. For this transformation to the target, it is because we use the last day/week/ or other time windows into the future to predict the target. This transformation is also applied to some weather data as past weather events have a relation to future weather patterns. Table 4.6 provides an overview of all windowed and lagged features.

Table 4.6: Windowed and lagged features from existing variables. To a day-ahead prediction, only the bold features are available.

Variable	Windowed & Lagged Features
Sum of Transformer 1, 2, 3 (mean)	Load 4h min, 2h offset
	Load 4h mean, 2h offset
	Load 4h max, 2h offset
	Load 12h min, 12h offset
	Load 12h mean, 12h offset
	Load 12h max, 12h offset
	Load 1 day min, 1 day offset
	Load 1 day mean, 1 day offset
	Load 1 day max, 1 day offset
	Load 5 day min, 2 day offset
	Load 5 day mean, 2 day offset
	Load 5 day max, 2 day offset
Temperature	**Temperature 1 day mean, 1 day offset**
Precipitation sum	**Precipitation sum 1 day mean, 1 day offset**

4.2.2. Feature selection

After generating all the features, a selection is made to reduce the number of features. The selection process is done manually, as the training process of the model using the proposed cost function takes too long to do a search over features and hyperparameters with the currently available hardware and settings.

The features which were excluded from the final feature list included the min and max values of the transformers, as the target will be the mean value (of a 15-minute interval) and the min/max value do not provide significantly more information in lagged/windowed/combined form than only the mean value. Furthermore, all weather features from all weather stations except for Rotterdam were excluded. Since these stations are in the same 50 km range, feature values are very similar. The reasoning behind including multiple weather stations was that wind fronts above land move and provide different wind speeds at different times in different areas. By utilizing wind data to the north, east, south, and west of Middelharnis the expectation was that a better prediction of the generation due to wind could be made as these wind farms are also distributed over the island of Goeree-Overflakkee. However, it turned out that the addition of these sources only degraded performance, hence the removal of these features.

If computational limitations were not a restraint, a better alternative would be using the shap-hypetune package [47], which became (functionally) available in January 2022. This package interfaces with both the XGBoost and Scikit learn environments and allows for different methods of feature selection and hyperparameter optimization.

4.3. Training

After preprocessing the data, and doing feature selection, the next step is training the model. First, the dataset is split into a training set, containing 85% of the data, and a test set containing 15% of the data. The test set is set aside and the train set is again split up, this time in 80% training and 20% validation.

4.3.1. Train/validate/test set

The simplest way of training a machine learning model will be tuning and training the model on a train and validation set. When data is scarce, often cross-validation is performed. The most common method is k-fold cross validation with 5 folds, which is visually represented in Figure 4.5. Here, the entire training set is split into 5 equal parts and the model is fit 5 times, with each time a different hold-out set. The cross-validation function will give an estimate of the test error curve and with this function, we can find the hyperparameters which minimize the error [44].

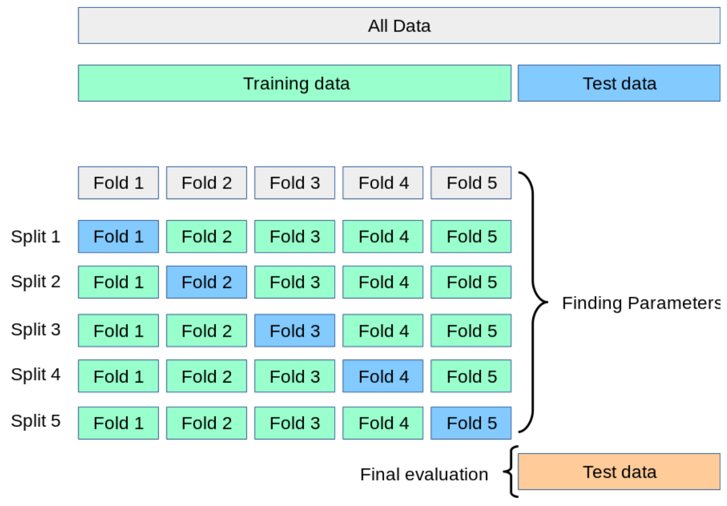


Figure 4.5: K-fold cross-validation, in this case with the standard of 5 folds. [48]

However, with time series, there is a serial dependence in the data, which means it can't be shuffled or split in this way, as it is nonsensible to train on future data to predict the past when the test set lies even further in the future. To mitigate this, we can perform cross-validation with an expanding window as seen in Figure 4.6. This does require enough data, such that the smallest window has sufficient training data. It also requires the data to be of the same distribution throughout the entire train set, otherwise, it will choose parameters that fit, for instance, the beginning part of the training set perfectly, but are less optimal when taking the final fold into account. If the trend continues into the test set, this will be especially detrimental.

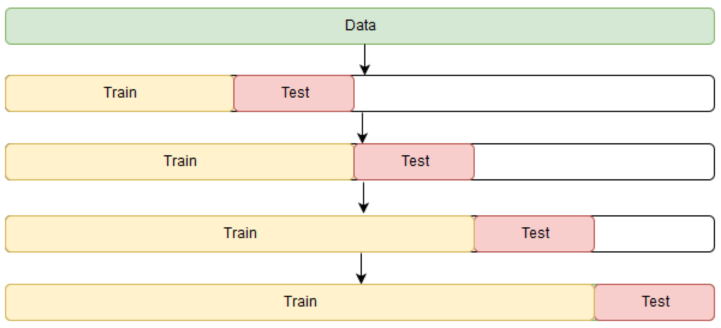


Figure 4.6: Time series cross-validation of a training set. [49]

Unfortunately, although this method is robust, it can't be used on the Middelharnis training data, because the peaks that violate the transformer limit, only start in the last part of the training set, thereby making the distribution of the folds far from equal to another. Next to this, the training set is only approximately 2 years, meaning that the smaller splits will not be large enough to effectively train a model on. Therefore, tuning the model will only be done on a train and validation set, coming from the entire train set.

4.3.2. Model Tuning

The final step is model tuning on the train/validate set. The tuning is done using GridSearchCV() from the Scikit learn package [50]. Three different models are tuned, all using a different loss function from the set: MSE, Pinball or the proposed loss function, which are scored by GridSearchCV() on the same metric as their loss function. Tuning them starts the same for all three:

1. Set global parameters in model XGBRegressor() or LGBMRegressor() for pinball loss. These include: Loss function (and alpha for pinball loss), random seed, tree method [hist], and importance type [total gain]. The importance type is not relevant for training, but how feature importance is plotted. In GridSearchCV(), set the model, the parameter grid, the train/validate indices, and the scoring function.
2. Set regularization parameters alpha and lambda to 1. For lambda this is the default parameter, setting alpha to 1 trains the model more conservative. This is beneficial, since there is not a lot of data to train on.

After setting these parameters, tuning the model with MSE and Pinball loss functions happens in the same manner, but tuning the model with the proposed loss function requires a different approach, as due to the epsilon part of the loss function, tuning is more difficult. Below is the tuning phase for models with MSE and Pinball loss:

3. Set a parameter grid for with a wide range for n_estimators to find a starting point to work from.
4. Once a suitable value for n_estimators is found, set this parameter and define ranges for gamma, subsample, colsample_by_tree, and node, to increase robustness to noise.
5. Next, fix the tuned parameters from the last step. Define a grid with the top 5 values for estimators from step 3, and tune parameters that influence the complexity of the model: min_child_weight, learning rate, and max_depth.
6. To prevent optimizing to a local minimum, vary the parameters from step 4 together with max_depth one by one, for values around the optimal values for n_estimators, min_child_weight, and learning rate. If a better set of parameters is found, restart tuning from step 4 or 5, depending on which parameters yielded the better results, in the same direction as which yielded the better results.
7. After the final search, manually train three models with parameters from the top three performing models of the final search. Pick the model which reduces the scoring metric the most when fit on the entire train set. Also, be careful of increasing n_estimator or reducing min_child_weight. This will improve the score on the train set, but only because the model will overfit it, which degrades an eventual test set score. Use the prediction time series plot and a scatter plot of the train set and compare for different values of estimators and min child weight. Choose values, which based on the plots seem the best. One point to note is a peak in the train set, which was due to uncleaned double data. If this peak is predicted, the model certainly overfit.

Tuning the model for MSE and Pinball loss as above yielded the hyperparameters as shown in Table 4.7, and Table 4.8 respectively. Both have a different value chosen for estimators, then what the hyper parameter search yielded. This is because for higher values of n estimators, the model overfits.

Table 4.7: Hyper parameter search space for MSE loss function trained model, with optimal hyperparameter underlined and chosen hyperparameter **bold**.

Hyper parameter	Values
Alpha	{**1**}
Lambda	{**1**}
Gamma	{0, 0.2, 0.4, **0.6**, 0.8, 1}
Subsample	{0.2, **0.4**, 0.6, 0.8, 1}
Colsample by tree	{**0.4**, 0.7}
Colsample by node	{**0.4**, 0.7}
Max depth	{None, 3, 4, **6**, 8, 10}
Learning rate	{**0.01**, 0.04, 0.1, 0.4}
N estimators	{200, 300, 400, 500, **600**, 700, 800 900, 1000, 1100, 1200, 1300, 1400, 1500}
Min child weight	{200, 300, 400, 500, 600, **700**, 800, 900, 1000, 1100, 1200, 1300 }

Table 4.8: Hyper parameter search space for Pinball loss function trained model, with optimal hyperparameter underlined and chosen hyperparameter **bold**.

Hyperparameter	Values
Alpha	{**1**}
Lambda	{**1**}
Gamma	{0, 0.2, 0.4, **0.6**, 0.8, 1}
Subsample	{0.2, **0.4**, 0.6, 0.8, 1}
Colsample by tree	{**0.4**, 0.7}
Colsample by node	{**0.4**, 0.7}
Max depth	{**None**, 3, 4, 6, 8, 10}
Learning rate	{0.01, **0.04**, 0.1, 0.4}
N estimators	{200, 300, 400, 500, 600, 700, 800 900, 1000, **1100**, 1200, 1300, 1400, 1500, 1600, 1700, 1800, 1900}
Min child weight	{200, 300, 400, 500, 600, 700, 800, 900, **1000**, 1100, 1200}

Next, the model is tuned on the proposed loss function. Tuning this model is different from the MSE and Pinball trained models, as here the ϵ function needs to be chosen. This variable also influences model convergence and hyper parameters need to be used both for model convergence as well as model tuning. As explained in chapter 3, for this loss function two models are trained based on the two different costs for the parameter C_{gopacs}. The first two steps were described above, as they are the same as for the MSE/Pinball models.

3. Set initial values for the hyperparameters gamma, subsample, colsample by tree/node, and depth. If an MSE/pinball model was trained first, use these parameters. Otherwise, set all to 0.5 and max depth to 5.
4. Set the value of epsilon larger than 2, but smaller than 10 or smaller/equal to the lowest cost/10.
5. Manually attempt multiple runs with different values for epsilon, min child weight, estimators, and keep delta = 0 and learning rate = 0.1. For each attempt, check the total cost, normalized cost, as well as the time series graph of the train set, scatter plot of the train set, and violin plots of the train sets. These indicate if the model converges on the train set. A combination of time series and violin plots will indicate if the model has not yet converged, in case time series peaks are not met and violin plots for the 50+ MW range have density high up, but no density lower down. If peaks are overshot, this is seen in the time series plot and on the violin plots by having an even density from bottom to top, but with a lower mean, which indicates over forecasting. This could be because the model converged long ago and is now overshooting. This is remedied by choosing higher epsilon, lower learning rates, fewer estimators, or a combination of the above. The goal is to find an epsilon and an approximate estimator range for which the model slightly over forecasts, as this can be tuned by a combination of max delta step, number of estimators, and learning rate.
6. After a suitable epsilon and approximate start point is found for the number of estimators, do a broad sweep over the number of estimators.
7. Once a suitable estimator is found, sweep over a narrower region, which includes the top 4 estimators from the previous sweep, but this time include max delta step as a parameter as well.
8. After fixing the estimators and delta step, determine min child weight. This includes three different runs with the estimator, delta combination that came first, second, and third in the previous sweep. Determine the best estimator, delta, and min child combination.
9. Next, try if a lower learning rate improves the result. Do this by doing separate searches, and fixing the learning rate per search. When going to a lower learning rate, increase the number of estimators and use a wider range for min child weight. The approach per learning rate is repeating steps 6-8. Choose the most suitable learning rate, estimator, delta step, and min child weight combination.
10. Lastly, to prevent a local minimum, do a broader search. With parameters dependent on how stable they were in previous runs. If delta shows consistent best results for one specific value, keep the value fixed. Run over multiple estimators and use the best 3 values for searching broader ranges min child weight. If a better option is found, repeat while changing the search regions in the direction of the previous best result.
11. Once a stable solution is found, fix all parameters, apart from estimators. Vary gamma, subsample, colsample by tree/node, and max depth one by one for various estimators. If a better value is found, repeat the process from step 6.

Table 4.9: Hyper parameter search space for proposed loss function with C_{gopacs}=267 €/MWh trained model, with optimal hyperparameter underlined and chosen hyperparameter bold.

Hyperparameter	Values
ε	{2, **3**, 5, 7, 10 }
Alpha	{**1**}
Lambda	{**1**}
Gamma	{0, 0.2, 0.4, **0.6**, 0.8, 1}
Subsample	{0.2, **0.4**, 0.6, 0.8, 1}
Colsample by tree	{**0.4**, 0.7}
Colsample by node	{**0.4**, 0.7}
Max depth	{None, 3, 4, **6**, 8, 10}
Learning rate	{0.01, **0.04**, 0.1, 0.4}
N estimators	{2000, 3000, 4000, 5000, 6000, 7000, 8000 9000, **10000**, 11000, 12000}
Min child weight	{200, 300, 400, **500**, 600, 700, 800, 900, 1000, 1100, 1200}
Max delta step	{**0**, 5, 10, 20, 25, 100}

Table 4.10: Hyper parameter search space for proposed loss function with C_{gopacs}=70 €/MWh trained model, with optimal hyperparameter underlined and chosen hyperparameter bold.

Hyperparameter	Values
ε	{2, **3**, 5, 7, 10 }
Alpha	{**1**}
Lambda	{**1**}
Gamma	{0, 0.2, 0.4, **0.6**, 0.8, 1}
Subsample	{0.2, **0.4**, 0.6, 0.8, 1}
Colsample by tree	{**0.4**, 0.7}
Colsample by node	{**0.4**, 0.7}
Max depth	{None, 3, 4, **6**, 8, 10}
Learning rate	{0.01, **0.04**, 0.1, 0.4}
N estimators	{2000, 3000, 4000, 5000, 6000, 7000, **8000** 9000, 10000, 11000, 12000}
Min child weight	{200, 300, 400, **500**, 600, 700, 800, 900, 1000, 1100, 1200}
Max delta step	{**0**, 5, 10, 20, 25, 100}

5 Evaluation and Results

After training and tuning the three models, this chapter will show the results. Next to the machine learning models, two simple methods of forecasting are used to compare the ability to minimize cost. The research questions which will be answered in this chapter are:

How does a custom loss function perform in comparison to traditional methods?

What are the advantages and disadvantages of using a custom loss function?

These questions will be answered by first showing the simple forecasting methods of 'previous week value' and 'average day of the week', after which the results of the custom loss function are shown per scenario. These scenario's were set up in chapter 3 and has a different cost for C_{gopacs}. For the mean cost scenario, this was 268 €/MWh and for the minimum cost scenario 70 €/MWh.

5.1. Simple forecasting methods

Often an average day of the week or previous week as a model is used as a benchmark or as a simple prediction. An average day of the week model is for weekday the average of that day in the data set. A previous week's model uses the value corresponding to last week's value as prediction. Both methods tend to work well if there is a clear periodic daily pattern, which is often the case in the distribution grid. Figure 5.1 shows the results of these baselines for a month of the test set and Table 5.1 shows the corresponding evaluation results.

Table 5.1: Results for baseline methods forecasting load of the three combined 13-50 kV transformers on Middelharnis in the test set.

	Scenario independent		Mean Scenario		Minimum Scenario	
	R^2 [-]	RMSE [MW]	Total cost [€]	FEPC [%]	Total cost [€]	FEPC [%]
Previous week method	-0.67	23.88	577,964	214	**454,974**	**847**
Average day of week method	**-0.18**	**20.12**	**480,611**	**161**	480,611	900

The figure and table together show that for location Middelharnis both the previous week and the average day of the week model are not good predictors. The previous week's model has a strong negative R^2 value, indicating it performs worse than just taking the average value of the dataset as the prediction. The average day of the week model performs better at this metric, as its metric is closer to 0. This is not surprising as the average day of the week model is close to the average value, which would yield an R2 score of 0. It performs worse here in this test set because this set contains more outliers in the form of peaks. The R^2 score contains an MSE component, which punishes outliers heavier. Therefore, resulting in a worse R^2 score in the test set.

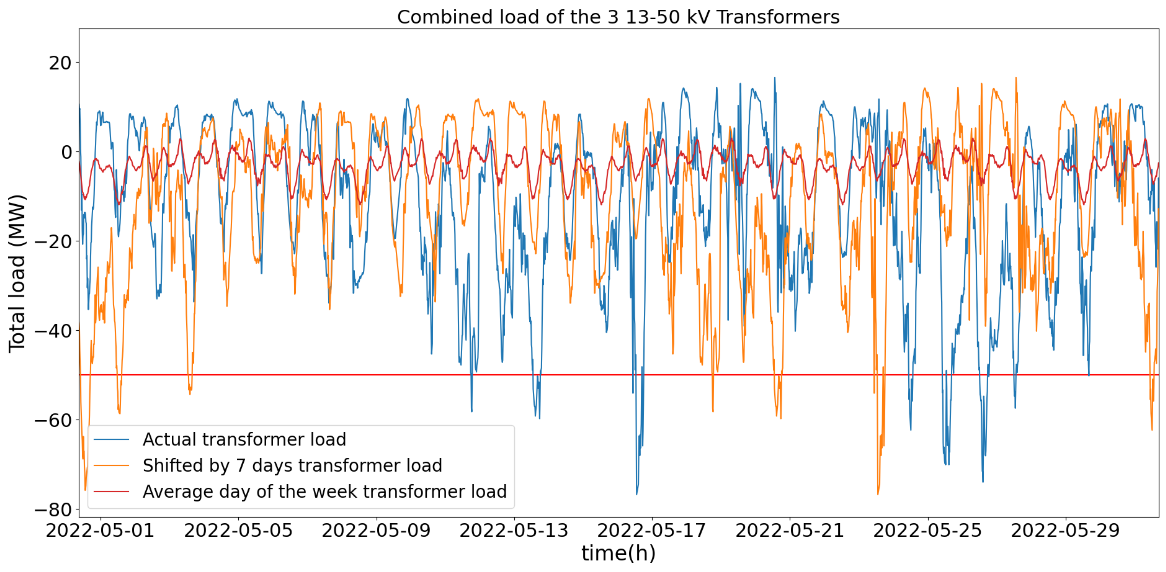


Figure 5.1: Load of the three combined 13-50 kV transformers of Middelharnis compared against a previous week, and average day of the week method shown for a month in the test set.

Although the MSE and R^2 metrics of the previous week model are far worse than the average day of the week model, one area where it has the potential to outperform the average day model is the costs. This can be explained by the cause of these peaks. The main component of the peak is wind, which doesn't have a strong significant periodicity, but a large part of the peak is solar, which on many days pushes the peak caused by wind over the set limit of 50 MW. Solar generation does have a strong daily pattern. Although weather conditions differ from week to week, if there are multiple weeks of similar weather conditions, this makes the previous week's value a moderate predictor of peaks. If GOPACS costs are low, as happens in the minimum scenario the total costs of the previous week model are smaller than the average day of the week model, since some of the peaks are correctly forecasted. However, when GOPACS costs are higher, as in the mean scenario, this advantage disappears as costs are made buying GOPACS for peaks that do not in actuality occur.

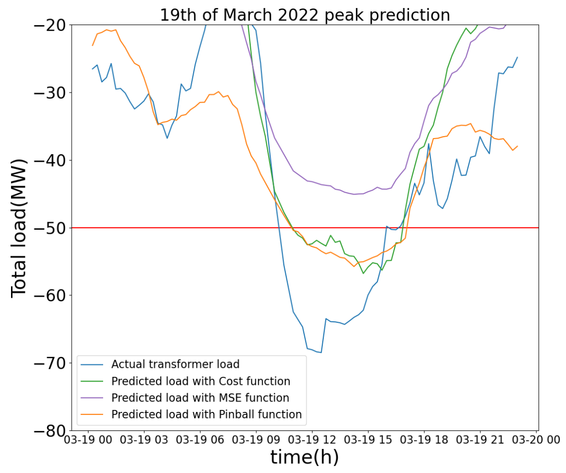
5.2. Mean scenario

Next, we will look at the results of the mean GOPACS costs scenario, where C_{gopacs} is 268 €/MWh. To do this, four congestion peaks due to generation are highlighted in a time series plot in Figure 5.2. These peaks are significant, as Stedin has had to actively perform congestion management in these instances. To see if this algorithm has merit in aiding Stedin, these are a good illustration of the proposed loss function's potential. These four congestion peaks can be seen in Figure 5.2. What can be observed in this image, is that an MSE loss function trained model is not a good predictor of peaks. Given the nature of an MSE loss function of predicting towards the mean, this is as expected. Next, with this C_{gopacs} cost, the model trained on the proposed cost loss function performs very similarly to the pinball loss trained model in these four higlighted peaks.

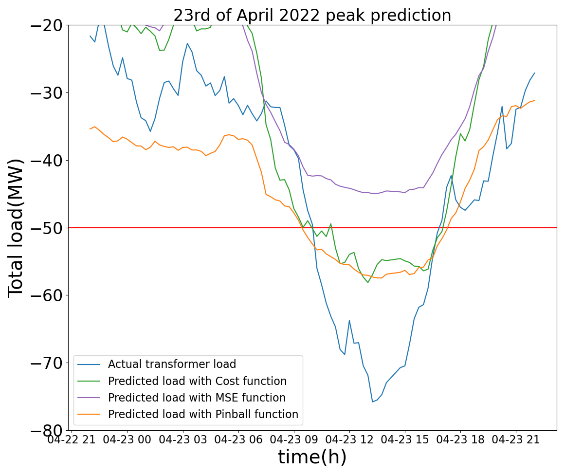
Table 5.2: Results on the test set for the mean scenario for the three trained models.

	R^2 [-]	RMSE [MW]	Pinball ($\alpha = 0.05$) [MW]	Total cost [€]	FEPC [%]
MSE model	0.799	8.32	3.68	480,611	161
Pinball model	0.510	12.99	**1.05**	407,032	121.21
Cost model	**0.800**	**8.29**	3.24	**405,958**	**120.62**

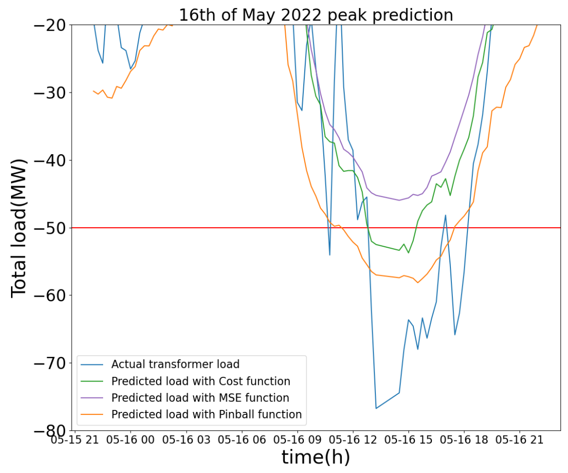
Table 5.2 shows the results on the full test set. Here we can also see, that in terms of cost and normalized cost the MSE-trained model performs significantly worse than the other two models. Furthermore, we can also see that the pinball and proposed cost loss function models perform very similarly to each other, with the proposed model only slightly outperforming the pinball model.



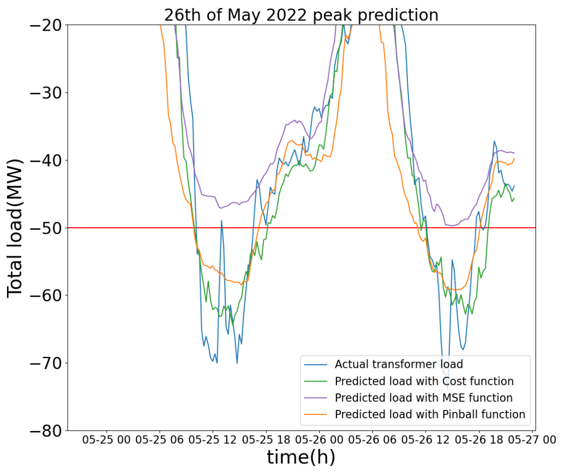
(a) Peak of 19th of March 2022.



(b) Peak of 23rd of April 2022.



(c) Peak of 16th of May 2022.



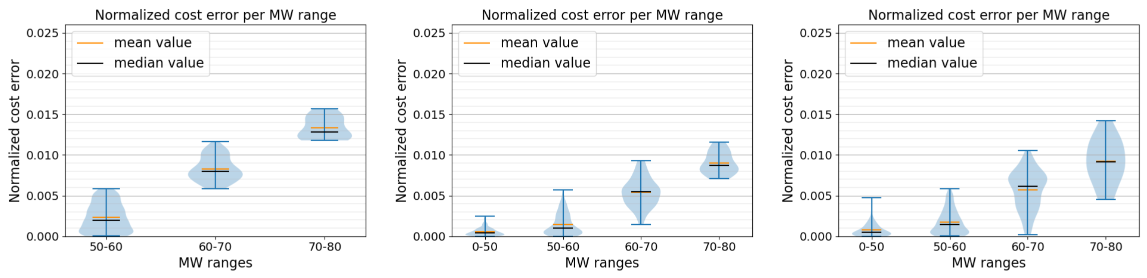
(d) Peak of 26th of May 2022.

Figure 5.2: Time series results of the mean scenario in predicting the load on the three 13-50 kV transformers of Middelharnis. The results are for four congestion peaks that happened in the test set.

Figure 5.3, sheds more light on where the pinball and proposed loss function model differ from each other. We can see that for the 0-50 MW and 50-60 MW ranges, the distributions are fairly similar, with only the 0-50 MW range showing a small difference in that the pinball trained model overpredicts less in this range. The reason for this overprediction, is because this range has a realization between 0-50 MW, meaning there is no cost related to a missed prediction and the cost is solely explained by overpredicting and buying unnecessary power on GOPACS. Therefore, having costs in this region suggests overpredicting. The distributions start to significantly differ for the 60-70, and 70-80 MW ranges. For the 60-70 MW range, the distribution for the Pinball loss model begins higher up, meaning that for these peaks it more often misses them and more congestion power needs to be bought. The proposed loss function does start closer to 0, meaning that comparatively more peaks are found, just not the full peak. The same can be seen for the 70-80 MW range. Here the means are again very similar but the distribution of the proposed function starts lower down, meaning it has more success in partially finding peaks, as compared to the pinball function. However, the distribution of the proposed function also ends higher, meaning it also misses peaks more often than the pinball function. For all distributions, the means of the cost function model are higher than the pinball model, even though Table 5.2 shows that the models perform similar, with the cost function model performing slightly better in terms of cost.

Figure 5.4 gives a different angle to the difference between these two functions. Figure 5.4b shows in the area below $y = 50$, that most points lay above the line $y = \hat{y}$, meaning a structural underprediction. In comparison Figure 5.4c shows that for the same area the cost function is more symmetrical in under and over predicting. It is not fully symmetrical as most points are also above the line. Comparing these two figures, we see that for peaks larger than approximately -60 MW the pinball model predicts around the same values for increasing realizations of generation of y. In comparison, the cost model will try to predict these values, it is just not very accurate in doing so.

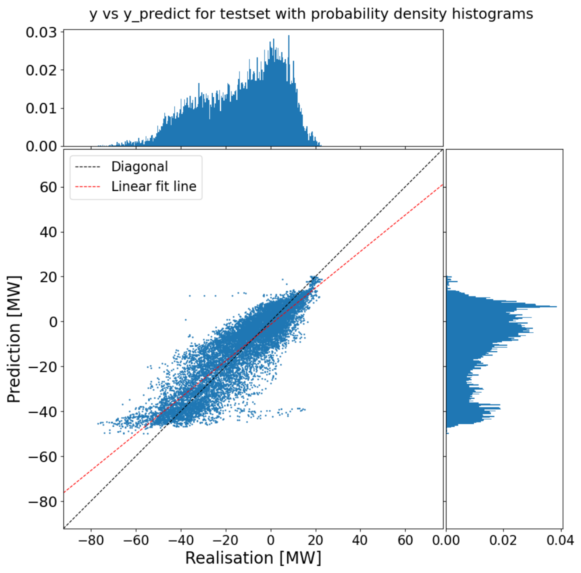
The results of the previous Figure 5.3 and Table 5.2 are also reflected in this plot for the MSE-trained model. Here, the 0-50 MW range is missing as the MSE trained model never predicts a peak that isn't there. From the other ranges it can also be seen that the MSE trained model has little success in finding generation peaks.



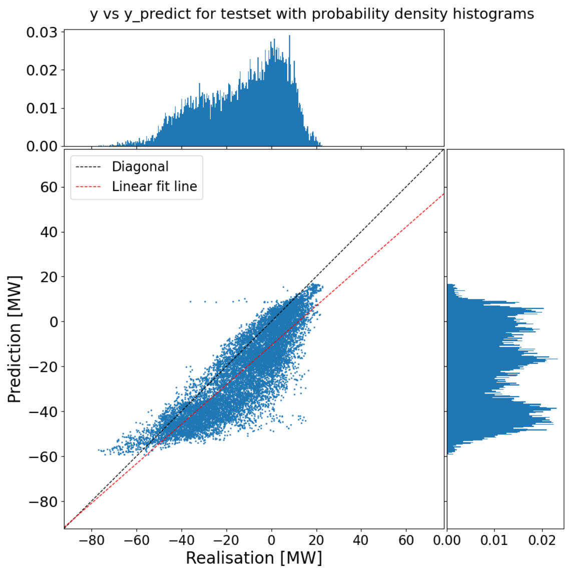
(a) MSE (b) Pinball (c) Cost model for mean scenario

Figure 5.3: Violin plots of the mean scenario for the three different models, showing the normalized cost error broken down per MW range of the test set.

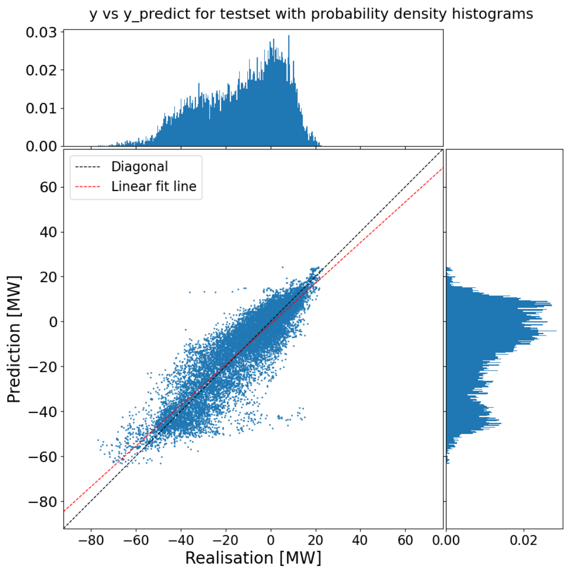
Figure 5.4 gives more information about how each loss function influences the prediction. Table 5.2 showed that the MSE and proposed loss function trained models have an almost identical R^2 and RMSE score. Looking at Figure 5.4, we can see that their shapes are also similar. For the MSE function, we see that it completely misses all peaks, but is slightly denser around the diagonal, meaning the prediction there is closer to the realization. Comparatively, the proposed loss function in the mean cost scenario is slightly wider around the diagonal, but it approximates the peaks significantly better, which contributes to a better R^2 and RMSE score. This can be seen from the points in the area below -50 MW being closer to the diagonal, compared to the MSE function where these are very far of. Figure 5.4b shows the scatterplot of the pinball function. We can clearly see here what the effect is of using a quantile, in that most of the data points fall below the diagonal, which is expected for a 5^{th} quantile prediction. Also, we can see that this distribution shows a small tail where the generation peaks occur. Similar to the MSE model, there is a plateau here where. It differes from the MSE that this plateau isn't flat, but moves down with lower realiations of y, signalling that it finds peaks, but underpredicts them.



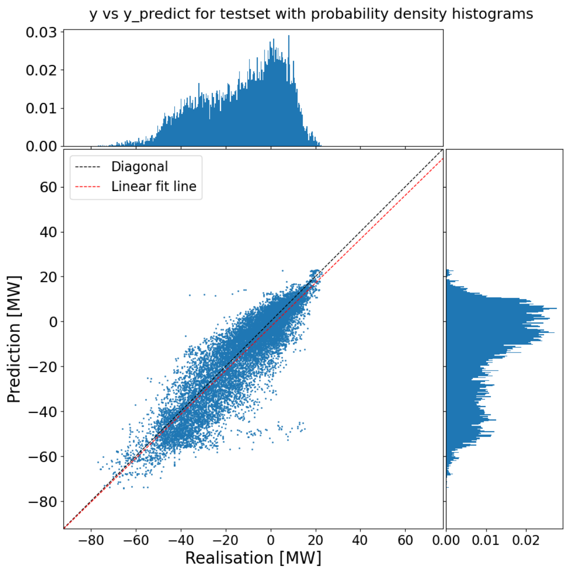
(a) MSE trained model



(b) Pinball trained model



(c) Cost function trained model for $C_{gopacs} = 268$ €/MWh.

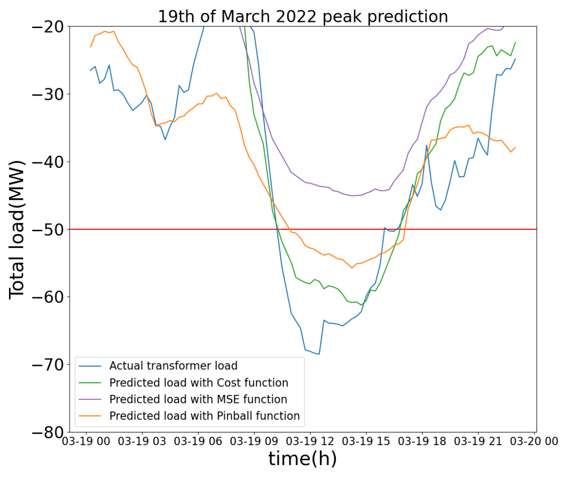


(d) Cost function trained model for $C_{gopacs} = 70$ €/MWh.

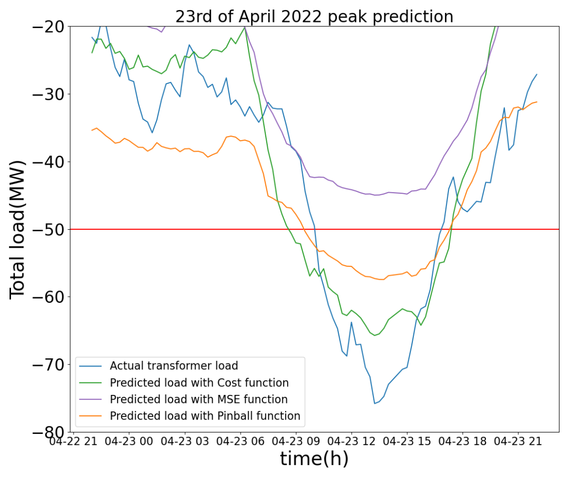
Figure 5.4: Scatter plots of realization y versus predictions $\hat{y}$ in the test set, including probability density distributions for both on all three loss functions.

5.3. Minimum scenario

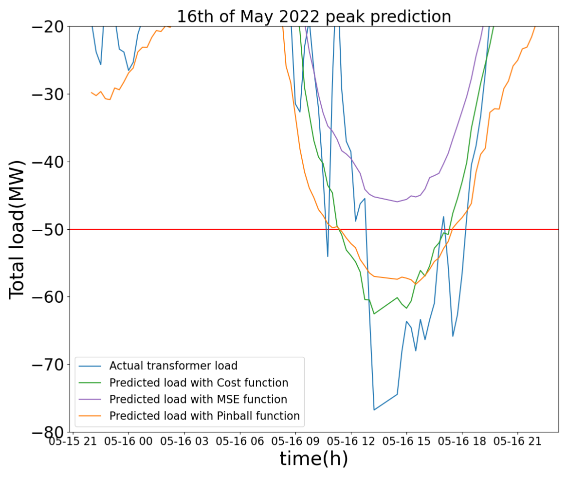
Lastly, we will look at the results of the minimum GOPACS cost scenario, where $C_{gopacs} = 70$ €/MWh. Again, there are four congestion peaks to highlight the merit of this function for Stedin. These can be found in Figure 5.5. Compared to the same plot in the previous scenario, we can now see that the model trained on the proposed function significantly outperforms the pinball loss trained model for a lower GOPACS cost.



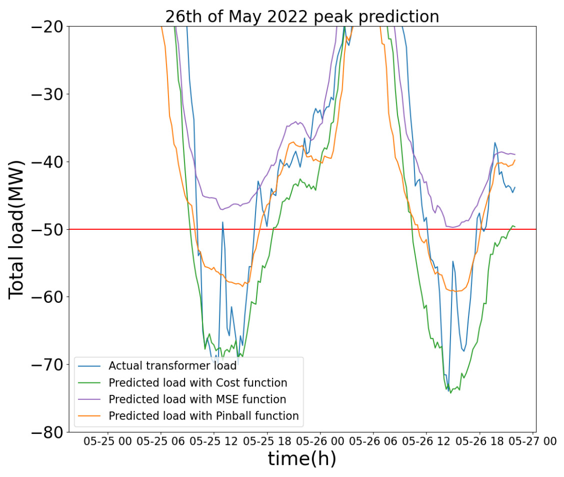
(a) Peak of 19th of March 2022.



(b) Peak of 23rd of April 2022.



(c) Peak of 16th of May 2022.



(d) Peak of 26th of May 2022.

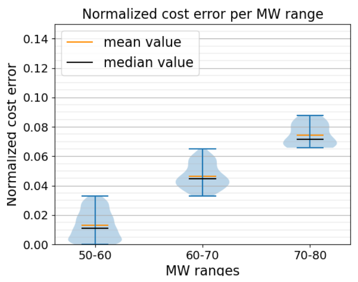
Figure 5.5: Time series results from the minimum scenario of predicting the load on the three 13-50 kV transformers of Middelharnis. The results are for four congestion peaks that happened in the test set.

This result is also visible in Table 5.3. It can be seen that in terms of total and normalized cost, the proposed loss function outperforms the pinball-trained model by a little over 1.5 times.

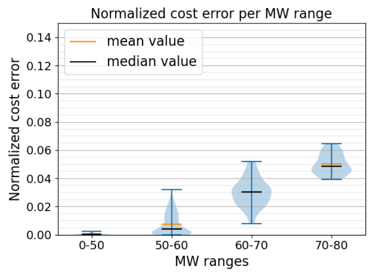
Table 5.3: Results on the test set for the minimum scenario for the three models.

	R^2 [-]	RMSE [MW]	Pinball ($\alpha = 0.05$) [MW]	Total cost [€]	FEPC [%]
MSE model	**0.799**	**8.32**	3.68	480,611	900
Pinball model	0.510	12.99	**1.05**	328,552	584
Cost model	0.778	8.75	2.43	**221,673**	**361**

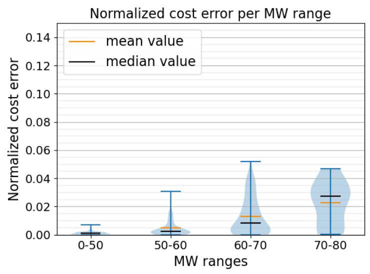
Figure 5.6 shows where this gain in performance is coming from. Again comparing the pinball and proposed loss function model with each other we see that the proposed model (partially) finds a peak more often. This can be seen from the distributions of the cost function beginning closer to 0, and with means that are lower. Closer to zero means that it at least has partially found the peak. Seeing as the means and medians of the proposed function are also significantly lower, this means it has found peaks more often than the Pinball function. Although for the 60-70 MW range the distributions end at the same height, for the 70-80 MW range, which are the most extreme peaks, the proposed loss function's distribution ends lower, meaning when it predicts a peak, it predicts it higher than the pinball model, resulting in fewer costs.



(a) MSE



(b) Pinball



(c) Cost scenario 2

Figure 5.6: Violin plots of the minimum scenario for the three different models, showing the normalized cost error broken down per MW range of the test set.

5.4. Comparison of models

As became apparent from the very first section, the simple methods of the average day of the week and the previous week's value are unable to give a day-ahead load prediction, indicating the challenge of forecasting for this location.

For this location a MSE trained model is also surprisingly disappointing. This loss function is often used as main or benchmark loss function in many different point forecasting models. In part it could have been expected, as an MSE-trained model will optimize towards the mean, meaning that peaks, which are outliers are not predicted.

To determine which model between the pinball and proposed loss function, is more suitable, it is important to know what the (average) GOPACS prices will be. In the mean scenario, it could be seen that both loss functions behaved relatively similar to each other. The real value of the proposed loss function is when the difference between the congestion costs and GOPACS costs are high. The difference between the two scenarios for the proposed function can be compared in the scatter plots of Figure 5.4. Here, we can see that for generation peaks the data points of the minimum scenario are more closely grouped around the diagonal $\hat{y} = y$, indicating a better prediction. The reason that the cost function in the mean scenario performs worse is that with a higher GOPACS cost, it is more costly to overforecast in hopes of not missing a congestion peak. If peaks are forecasted that aren't (fully) there, the total cost increases more rapidly than if GOPACS costs were lower. With lower GOPACS cost the model is motivated to find all peaks, as the costs associated with these peaks are the main source of cost, whereas with higher GOPACS cost the model is also motivated to reduce instances of overforecasting, as this adds up quickly with the higher GOPACS cost.

The advantage of using the proposed loss function is that it not only provides a load prediction but also tries to find a buying strategy that minimizes the cost. Although a pinball loss function can be used to show the cost of a prediction, it misses this step. If GOPACS cost would be higher than in these scenarios the loss function would be very conservative in buying power on GOPACS, as this could be more costly than doing nothing.

In the future, it is likely that GOPACS prices might decrease from the prices shown in section 3.2 with the increase of renewable energy. As GOPACS prices are linked to energy markets via lost opportunity costs, when prices drop there, GOPACS prices will also reduce. The expectation is that energy prices will become lower when enough renewable energy is available, as renewable energy will push more expensive generating units like gas generators out of the merit order. Therefore, the results that can be achieved with this proposed loss function will get better if this future trend will occur. On the contrary, if energy prices keep being high, as they are at the current time of writing due to the Russian invasion into Ukraine, the proposed loss function will become even more conservative in its prediction and underforecast peaks. This scenario might see the pinball trained model outperform the cost model in terms of finding congestion peaks. However, this does not also mean that the pinball model provides the cheapest solution.

5.5. Interpretation of results

5.5.1. Data availability

In training the model, the available data had a large impact on the results. For weather features, the historic values were used, as no historic forecast database was available. This means that for the actual predictions, a level of uncertainty is added and performances on historic predictions (when available) will not yield results with the same accuracy as presented in this thesis.

Furthermore, the decision was made to only use one weather station, as this gave better model performance than using all weather stations. Although the difference should be minor, a performance increase could be gained by using interpolation between the different weather stations.

Lastly, the dataset was rather short: 2.5 years. And saw few congestion peaks due to generation in the train set. Only at the very end of the train set were these encountered. Splitting the train set further into a separate train and validating set, reduced the number of peaks on which could be learned even further.

5.5.2. Individual transformer overloads

As we've seen in Figure 3.3, individual transformers can overload, while the aggregation of the three transformers on the substation is not yet in overload. These instances are rare and the overloads are minor, but predicting on substation level does not guarantee that all cases of congestion are considered for predictions.

5.5.3. Sensitivity to hyper parameters

The proposed loss function is very sensitive to the choice of hyperparameters. Where for a loss function like MSE, in a certain hyperparameter range search, the model accuracy might reach an optimum and degrade from there, for the proposed function there were multiple local minima. This has to do with model convergence in the sense that the gradients on which the model trains have multiple gradient areas with a fixed gradient and large differences between the different gradients. Next to this, the epsilon value needs to be chosen suitably for model convergence. This meant that certain hyperparameters, like the number of estimators, learning rate, and max delta step were not only used to tune the model, but also to let the model converge. This dual role of these parameters is sub-optimal for tuning.

5.5.4. Feature selection

As mentioned in subsection 4.2.2, no feature selection was done as this was not computationally feasible. This could have degraded model performance and a better result might be attainable with the application of feature selection. Feature selection might prove difficult, as the model with the proposed loss function is very sensitive to tuning.

6 Conclusion and Recommendations

In this work, a custom loss function on the basis of costs has been developed. The goal of this function was to improve the finding of congestion peaks in short-term load forecasting, such that the costs of the congestion can be minimized. The proposed function was applied to the case of substation Middelharnis, where congestion due to renewables is ongoing. Two models were trained on the proposed loss function, each for a different assumed cost for buying flexible power on GOPACS. The following sections will describe the main findings and discuss the proposed function's limitations with recommendations for future work.

6.1. Research questions

What is the value of using a custom loss function for load forecasting in reducing expected congestion costs in the distribution grid? The main benefit of using a custom loss function based on cost is that the prediction is done in line with the prediction goal of minimizing congestion cost. The proposed loss functions is both useful for load predictions, as well as functioning as a buying strategy for flexible power on GOPACS. In the results, we saw that for a higher price of power on GOPACS, the function predicts the load more conservatively to minimize the costs. This shows that this function is not only aiming to predict all congestion peaks but also tries to minimize the overall cost. When GOPACS costs are low, peaks are better forecasted, as the costs associated with these peaks are the main contributors to the overall cost.

Furthermore, the cost function outperforms traditional benchmarks models and models trained on MSE and Pinball functions in terms of cost reduction.

What is the cause of congestion and which responsibilities does the DSO have in regards to congestion management?
The cause of congestion is the global ambition and realization of increasing the use of renewable energy, which the Dutch government has implemented through their Wind on Land plan, as well as through Renewable Energy Strategies. On Middelharnis this meant an increase in the number of wind parks, which in the beginning were connected on the 13 kV side, instead of the 50 kV side of the TenneT Middelharnis substation. This happened parallel to the growth of distributed rooftop photovoltaics. Together the renewable energy available surpassed the transformer capabilities. Stedin signaled this, but extending a substation takes a considerable amount of time.

In the meantime DSOs are required to perform congestion management for congested areas. As per the report from a 3rd party seen in subsection 3.1.2, Stedin is not obligated to perform congestion management on Middelharnis, as the report stated that this isn't feasible according to the Netcode version at that time. Although Stedin is now actively seeking solutions [38]. With the issue of congestion growing in other parts of the Stedin grid, Stedin will encounter situations in which congestion management will be required. Once that happens, Stedin needs to mitigate this congestion. To do this it has multiple tools available, like bilateral contracts with customers for voluntary curtailment, although the obligation is to handle congestion primarily on the redispatching market GOPACS. To be able to use these tools, Stedin will need to be able to forecast congestion day-ahead, to be able to activate these tools.

What is a suitable model of the different cost components of congestion and how can that be incorporated into a loss function?
To capture the costs associated with congestion, two different cost types were used. The first is a disturbance of service fee, which needs to be paid when a wind park is disconnected by Stedin, due to overloading of the transformers. The second is a term that captures the cost, if flexible power was bought on a redispatching market. The implemented costs of this market are based on simulated results done by Stedin on what these costs could entail.

With these cost components, two scenarios were made: one using the mean costs from this research, and the other a smaller amount, more in line with costs that can be expected when congestion due to excessive renewable energy occurs. A relation between these costs was made in Equation 3.5. Here the cost is implemented by taking into account two time horizons. The total loss in terms of cost is dependent on how much reserve power we bought the day ahead, and how much penalty we need to pay on the day itself, due to a (partially) missed prediction. Both models are a valid representation of costs. The scenario with the mean GOPACS costs shows what the prediction will be for the average costs that 2021 saw. The scenario with the minimum GOPACS costs is less representative for that year, but more in line with the expectation that GOPACS costs will reduce in the future due to more renewable energy integration.

How does a custom loss function perform in comparison to traditional load forecasting methods?
As seen in the previous chapter, the average day of the week and the previous week's value methods are not suitable for a location with a lot of renewable integration like on Middelharnis. A model trained on MSE is also not suitable, as the nature of the MSE loss function is to predict towards the mean. As congestion peaks are outliers in this set, it will not (accurately) predict them. Compared to these methods and models, the custom loss function is a massive improvement.

Compared to a model trained on pinball loss, the improvements really depend on the choice of C_{gopacs}. If this term is sufficiently small as compared to the penalty term, the custom function will outperform the pinball loss model. If the difference is smaller, the two models perform comparably. When the difference between C_{gopacs} and the penalty term becomes smaller, the pinball loss model might start outperforming the custom loss function trained model.

What are the advantages and disadvantages of using a custom loss function?
The main advantage of using a custom loss function is that it trains in line with the goal of the prediction: minimizing congestion cost. This prediction is therefore at the same time also a buying strategy for buying power on GOPACS.

However, tuning this model is computationally expensive and also very sensitive to tuning. It could provide comparable or worse results than a pinball loss trained model, which trains significantly quicker.

6.2. Discussion and Future work

6.2.1. Definition of loss

One large assumption done in this thesis is the definition of the cost components. Only two components were defined, whereas in reality there can be more sources of cost, like from bilateral contracts. Also, the cost of power bought on GOPACS will fluctuate through the year, whereas in this model, this cost is fixed. Lastly, the research into potential GOPACS costs focused on the year 2021, which due to corona might not be the most representative year.

At the time of writing, the energy prices are extremely high due to the Russian invasion of Ukraine. For the foreseeable future, this could cause prices for GOPACS to rise, as GOPACS prices are linked to the energy markets via lost opportunity costs. As we've seen from the two scenarios, a higher price for power bought on GOPACS leads to a degradation in finding peaks, due to the function becoming more conservative.

The recommendation is to extend the current loss function with more cost components that we have seen in Figure 2.1.

Furthermore, although the goal of Stedin is to operate in the most cost-effective way, they also value the reliability of their grids. This means that shutting wind parks off, because the predictive function failed to predict a congestion event has a higher 'loss' to them, then is captured by a congestion penalty. It is recommended to research a term that can capture this extrinsic loss.

6.2.2. Computational capacity needed

One limitation in creating the model is the computational capacity needed. The model requires a low learning rate and a high amount of estimators to converge, leading to long training times. This limits model tuning, as narrower hyperparameter ranges are needed due to computation times. This also leads to the decision not to optimize feature selection, which might have negatively impacted the performance of the model.

It is recommended to alter the loss function to remedy this. Altering the function by squaring it makes it twice differentiable. This allows XGBoost to more efficiently find the optimum since it now has a hessian, which helps model convergence and speed. Although the advantage of having the output directly in the wanted quantity € would be lost.

Bibliography

[1] M. of Economic and C. affairs, “Klimaatakkoord,” June 2019, accessed 06-09-2022. [Online]. Available: https://www.rijksoverheid.nl/onderwerpen/klimaatverandering/documenten/rapporten/2019/06/28/klimaatakkoord

[2] D. Kuiken and H. F. Más, “Integrating demand side management into eu electricity distribution system operation: A dutch example,” *Energy Policy*, vol. 129, pp. 153–160, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0301421519300813

[3] N. Nederland, “Capaciteitskaart afname/invoeding electriciteitsnet,” accessed 06-09-2022. [Online]. Available: https://capaciteitskaart.netbeheernederland.nl/

[4] Stedin, “Investeringsplan,” 2022, accessed 8-11-2021. [Online]. Available: https://www.stedin.net/over-stedin/jaarverslagen-en-publicaties/investeringsplan

[5] Autoriteit Consument en Markt van 2016, “Netcode elektriciteit,” accessed: 27-06-2022. [Online]. Available: https://wetten.overheid.nl/BWBR0037940/2022-05-18

[6] GOPACS, “Hoe werkt gopacs?” accessed on 30-08-2022. [Online]. Available: https://www.gopacs.eu/hoe-werkt-gopacs/

[7] Autoriteit Consument en Markt, “ACM stimuleert netbeheerders slimmer gebruik te maken van bestaande netten,” May 2022, accessed: 22-06-2022. [Online]. Available: https://www.acm.nl/nl/publicaties/acm-stimuleert-netbeheerders-slimmer-gebruik-te-maken-van-bestaande-netten

[8] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’16. New York, NY, USA: Association for Computing Machinery, 2016, p. 785–794. [Online]. Available: https://doi-org.tudelft.idm.oclc.org/10.1145/2939672.2939785

[9] G. Li and H.-D. Chiang, “Toward cost-oriented forecasting of wind power generation,” *IEEE Transactions on Smart Grid*, vol. 9, no. 4, pp. 2508–2517, 2018.

[10] J. Zhang, Y. Wang, and G. Hug, “Cost-oriented load forecasting,” *Electric Power Systems Research*, vol. 205, p. 107723, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378779621007045

[11] Ministry of Economic Affairs and Climate Policy, Netherlands, “Electriciteitswet 1998,” accessed: 27-06-2022. [Online]. Available: https://wetten.overheid.nl/BWBR0009755/2021-07-01/

[12] Afman, M et. al., “Samenvatting Het Energiesysteem van de Toekomst,” Netbeheer Nederland, Tech. Rep., April 2021, accessed: 12-11-2021. [Online]. Available: https://www.netbeheernederland.nl/_upload/files/NetbeheerNL_Rapport-Energiesysteem_A4_FC.pdf

[13] Autoriteit Consument en Markt, “Besluit van de Autoriteit Consument en Markt van 24 mei 2022 kenmerk ACM/UIT/577139 tot wijziging van de voorwaarden als bedoeld in artikel 31 van de Elektriciteitswet 1998 betreffende regels rondom transportschaarste en congestiemanagement ,” *Staatscourant van het Koninkrijk der Nederlanden*, no. 14201, May 2022, accessed: 27-06-2022. [Online]. Available: https://zoek.officielebekendmakingen.nl/stcrt-2022-14201.html

[14] Netbeheer Nederland, “Instrumentenpakket flexibiliteit,” not publicly available.

[15] A. R. K. Renaud, “Ontwikkeling AM Bedrijsvoering elektra, op weg naar System Operator,” November 2021, internal Stedin communication.

[16] G. Papaioannou, C. Dikaiakos, A. Dramountanis, and P. Papaioannou, "Analysis and Modeling for Short- to Medium-Term Load Forecasting Using a Hybrid Manifold Learning Principal Component Model and Comparison with Classical Statistical Models (SARIMAX, Exponential Smoothing) and Artificial Intelligence Models (ANN, SVM): The Case of Greek Electricity Market," *Energies*, vol. 9, p. 635, 08 2016. [Online]. Available: https://www.researchgate.net/publication/306243928_Analysis_and_Modeling_for_Short-_to_Medium-Term_Load_Forecasting_Using_a_Hybrid_Manifold_Learning_Principal_Component_Model_and_Comparison_with_Classical_Statistical_Models_SARIMAX_Exponential_Smoothi

[17] Boundless.com, "Forecasting," accessed: 06-07-2022. [Online]. Available: http://www.library.snls.org.sz/boundless/boundless/management/textbooks/boundless-management-textbook/strategic-management-12/internal-analysis-inputs-to-strategy-88/forecasting-428-883/index.html

[18] M. A. Hammad, B. Jereb, B. Rosi, and D. Dragan, "Methods and models for electric load forecasting: A comprehensive review," *Logistics & Sustainable Transport*, vol. 11, pp. 51–76, 02 2020. [Online]. Available: https://www.researchgate.net/publication/339403486_Methods_and_Models_for_Electric_Load_Forecasting_A_Comprehensive_Review

[19] A. Ahmad, N. Javaid, A. Mateen, M. Awais, and Z. A. Khan, "Short-term load forecasting in smart grids: An intelligent modular approach," *Energies*, vol. 12, no. 1, 2019. [Online]. Available: https://www.mdpi.com/1996-1073/12/1/164

[20] S. Ungureanu, T. Vasile, and A. Cziker, "Deep learning for short-term load forecasting—industrial consumer case study," *Applied Sciences*, vol. 11, p. 10126, 10 2021. [Online]. Available: https://doi.org/10.3390/app112110126

[21] T. Anwar, B. Sharma, K. Chakraborty and H. Sirohia, "Introduction to load forecasting," *International Journal of Pure and Applied mathematics*, vol. 119, no. 15, pp. 1527–1538, 2018. [Online]. Available: https://acadpubl.eu/jsi/2018-118-14-15/issue15.html

[22] A. S. Tay and K. F. Wallis, "Density forecasting: a survey," *Journal of Forecasting*, vol. 19, no. 4, pp. 235–254, 2000. [Online]. Available: https://onlinelibrary-wiley-com.tudelft.idm.oclc.org/doi/abs/10.1002/1099-131X%28200007%2919%3A4%3C235%3A%3AAID-FOR772%3E3.0.CO%3B2-L

[23] Petropoulos, Fotios et al., "Forecasting: theory and practice," *International Journal of Forecasting*, vol. 38, no. 3, pp. 705–871, 2022. [Online]. Available: https://doi.org/10.1016/j.ijforecast.2021.11.001

[24] R. Hirekerur, "A comprehensive guide to loss functions — part 1 : Regression," *Medium*, July 2020, accessed on 1-08-2022. [Online]. Available: https://medium.com/analytics-vidhya/a-comprehensive-guide-to-loss-functions-part-1-regression-ff8b847675d6

[25] G. Gürses-Tran, H. Flamme, and A. Monti, "Probabilistic load forecasting for day-ahead congestion mitigation," in *2020 International Conference on Probabilistic Methods Applied to Power Systems (PMAPS)*, 2020, pp. 1–6.

[26] Scikit learn, "3.3.4.1. r^2 score, the coefficient of determination," accessed on 18-09-2022. [Online]. Available: https://scikit-learn.org/stable/modules/model_evaluation.html#r2-score

[27] ——, "Quantile regression," accessed on 1-08-2022. [Online]. Available: https://scikit-learn.org/stable/auto_examples/linear_model/plot_quantile_regression.html

[28] H. Kebriaei, B. N. Araabi, and A. Rahimi-Kian, "Short-term load forecasting with a new nonsymmetric penalty function," *IEEE Transactions on Power Systems*, vol. 26, no. 4, pp. 1817–1825, 2011.

[29] Y. Wang and L. Wu, "Improving economic values of day-ahead load forecasts to real-time power system operations," *IET Generation, Transmission & Distribution*, vol. 11, no. 17, pp. 4238–4247, 2017. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-gtd.2017.0517

[30] XGBoost, "Xgboost parameters," 2021, accessed 22-03-2022. [Online]. Available: https://xgboost.readthedocs.io/en/stable/parameter.html

[31] Ministerie van Infrastructuur en Milieu en Ministerie van Economische Zaken, “Structuurvisie windenergie op land,” March 2014, accessed: 30-06-2022. [Online]. Available: https://www.eerstekamer.nl/overig/20140331/structuurvisie_windenergie_op_land/meta

[32] Rijksdienst voor Ondernemend Nederland, “Monitor wind op land over 2020, 8th edition,” June 2021, accessed: 30-06-2022. [Online]. Available: https://www.eerstekamer.nl/behandeling/20210714/brief_regering_monitor_wind_op/info

[33] D-Cision B.V., “Onderzoek naar de toepassing van congestiemanagment in het station Middelharnis 50/13 kV,” July 2021, accessed 8-11-2021. [Online]. Available: https://www.stedin.net/-/media/project/online/files/zakelijk/congestiemanagement/stedin-onderzoek-toepassing-congestiemanagement-middelharnis-5013-kv-versie-10.pdf

[34] Stedin, “’Groeipijn’ op het elektriciteitsnet van Goeree-Overflakkee,” April 2021, accessed 26-07-2022. [Online]. Available: https://www.stedin.net/over-stedin/pers-en-media/persberichten/groeipijn-op-het-elektriciteitsnet-van-goeree-overflakkee

[35] “Ieee guide for loading mineral-oil-immersed transformers and step-voltage regulators,” *IEEE Std C57.91-2011 (Revision of IEEE Std C57.91-1995)*, pp. 1–123, 2012. [Online]. Available: https://doi.org/10.1109/IEEESTD.2012.6166928

[36] V. M. Montsinger, “Effect of load factor on operation of power transformers by temperature,” *Transactions of the American Institute of Electrical Engineers*, vol. 59, no. 11, pp. 632–636, 1940. [Online]. Available: https://doi.org/10.1109/T-AIEE.1940.5058023

[37] L. C. Nichols, “Effect of overloads on transformer life,” *Electrical Engineering*, vol. 53, no. 12, pp. 1616–1621, 1934. [Online]. Available: https://doi.org/10.1109/EE.1934.6540128

[38] Stedin, “Congestie in middelharnis,” accessed on 26-09-2022. [Online]. Available: https://www.stedin.net/zakelijk/energietransitie/flexibel-energiesysteem/beschikbare-netcapaciteit/congestie-en-congestiemanagement/middelharnis

[39] ——, “Compensatieregeling elektriciteit en gas,” accessed 3-06-2022. [Online]. Available: https://www.stedin.net/storing-en-onderhoud/vergoeding-berekenen

[40] GOPACS, “Costs for using idcons for redispatch,” accessed on 03-08-2022. [Online]. Available: https://idcons.nl/publicexpenses#/expenses

[41] N. C. Martin, “Gopacs cost estimation through lost opportunity cost by simulating wind park trading,” simulation data from windpark Palland over the year 2021.

[42] Koninklijk Nederlands Meteorologisch Instituut, “Uurgegevens van het weer in nederland.” [Online]. Available: https://www.knmi.nl/nederland-nu/klimatologie/uurgegevens

[43] NEDU, “Verbruiksprofielen,” 2021, accessed: 31-05-2022. [Online]. Available: https://www.nedu.nl/documenten/verbruiksprofielen/

[44] T. Hastie, R. Tibishirani, and J. Friedman, *The Elements of Statistical Learning*, 2nd ed., ser. Springer Series in Statistics. Springer New York, NY, 2009. [Online]. Available: https://doi.org/10.1007/978-0-387-84858-7

[45] F. Montesino Pouzols and A. Lendasse, “Effect of different detrending approaches on computational intelligence models of time series,” in *The 2010 International Joint Conference on Neural Networks (IJCNN)*, 07 2010, pp. 1–8. [Online]. Available: https://doi.org/10.1109/IJCNN.2010.5596314

[46] F. Lazzeri, *Time Series Data Preparation.* John Wiley & Sons, Ltd, 2020, ch. 3, pp. 61–99. [Online]. Available: https://onlinelibrary-wiley-com.tudelft.idm.oclc.org/doi/abs/10.1002/9781119682394.ch3

[47] M. Cerliani, “Shap-hypetune,” January 2022, version v0.2.4 or above.

[48] Sci-kit learn, “3.1. cross-validation: evaluating estimator performance.” [Online]. Available: https://scikit-learn.org/stable/modules/cross_validation.html

[49] J. Garg, “Modelling - using k-fold cross-validation for time-series model selection.” [Online]. Available: https://stats.stackexchange.com/questions/14099/using-k-fold-cross-validation-for-time-series-model-selection

[50] Pedregosa, F. et al., “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

A
Features

	Vlakte v.d. Raan	Schaar	Stavenisse	R'dam-Geulhaven	Rotterdam
Wind direction	✓	✓	✓	✓	✓
Average wind speed of the last 10 min of the hour	✓	✓	✓	✓	✓
Highest wind gust	✓	✓	✓	✓	✓
Mean hourly wind speed		✓	✓	✓	✓
Temperature					✓
Dew point temperature					✓
Radiation					✓
Sun duration					✓
Precipitation duration per hour					✓
Sum of the precipitation of the last hour					✓
Airpressure					✓
Visibility					✓
Humidity					✓
Cloud cover					✓

Table A.1: KNMI features per weather station [42].

Profile	Minimal connection	Maximal connection	Comments
E1A	-	≤ 3 x 25 A	small retail meter that can't be gauged remotely non-measured connection
E1B	-	≤ 3 x 25 A	small retail meter that is remotely gaugeable night tariff regime
E1C	-	≤ 3 x 25 A	small retail meter that is remotely gaugeable evening tariff regime
E2A	> 3 x 25 A	≤ 3 x 80 A	non-remotely gaugeable small retail meter
E2B	> 3 x 25 A	≤ 3 x 80 A	remotely gaugeable small retail meter
E3A	> 3 x 80 A	≤ 100 kW	operation time ≤ 2000 hours
E3B	> 3 x 80 A	≤ 100 kW	operation time > 2000 hours, ≤ 3000 hours
E3C	> 3 x 80 A	≤ 100 kW	operation time > 3000 hours, ≤ 5000 hours
E3D	> 3 x 80 A	≤ 100 kW	operation time > 5000 hours
E4A	all measured connections that are controlled by the driving signal of public lighting with a connected power of less than 100 kW		

Table A.2: Nedu profiles with their respective characteristics [43].

B

Loss function

B.1. Additional loss function plots for example values

Table B.1: Example values for the costs and ε.

Variable	Value
C_{gopacs}	5
$C_{congestion}$	15
ε	2.5

B.1.1. GOPACS part of the loss function

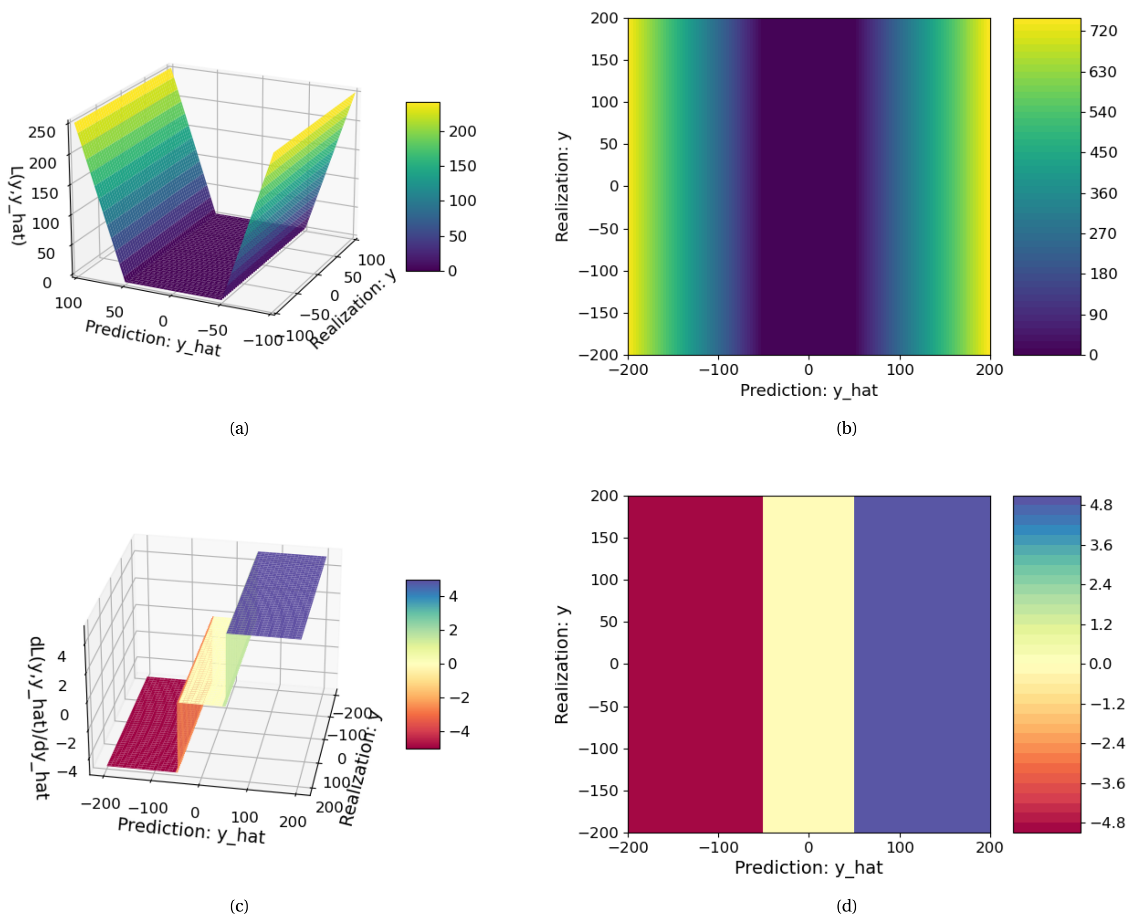
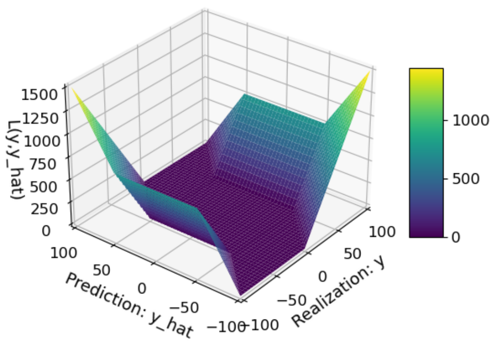
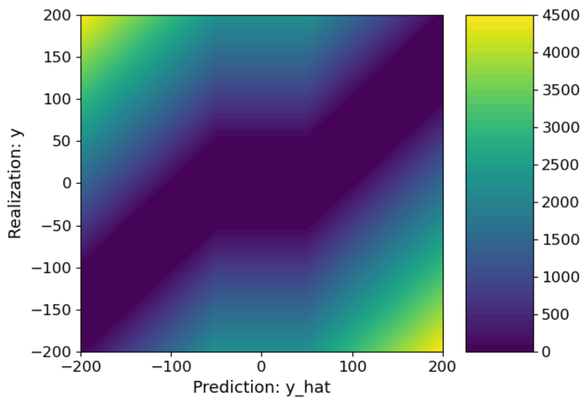


Figure B.1: GOPACS part of loss function (above) and the corresponding gradients (below).

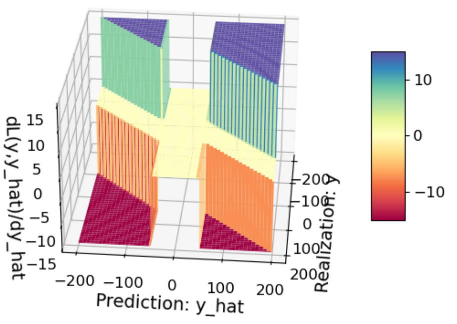
B.1.2. Congestion part of the loss function



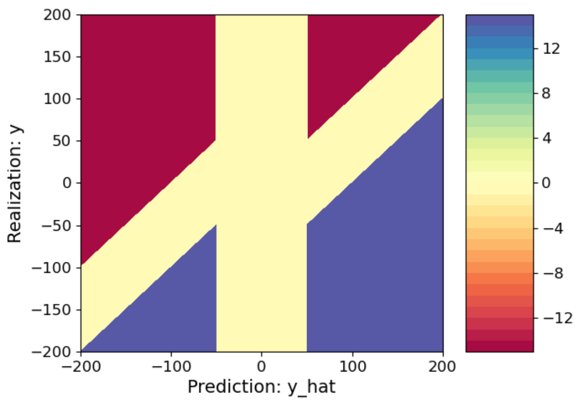
(a)



(b)



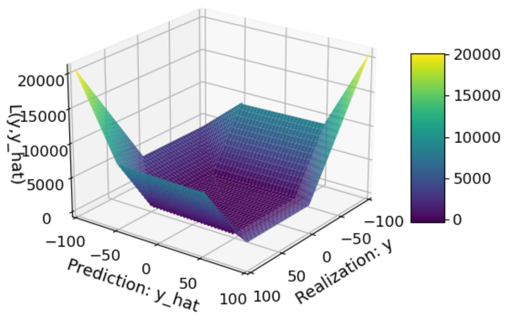
(c)



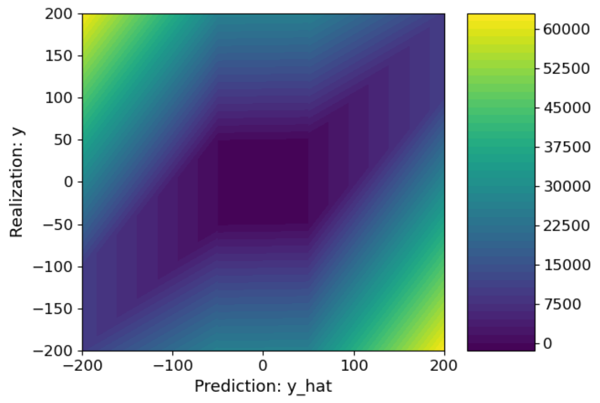
(d)

Figure B.2: Congestion part of loss function (above) and the corresponding gradients (below).

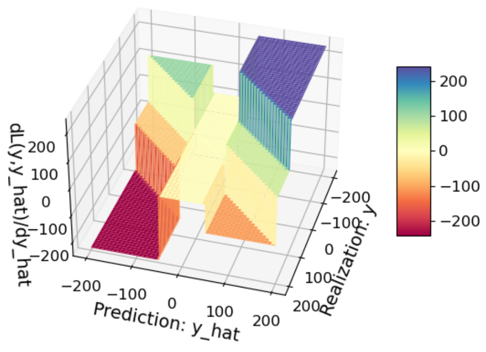
B.2. Loss function plots with $C_{gopacs} = 268$ €/MWh



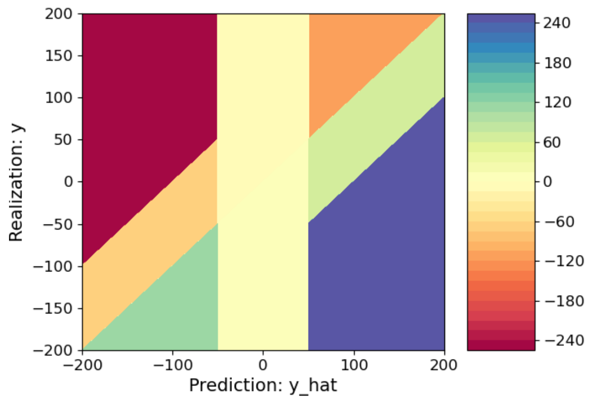
(a)



(b)



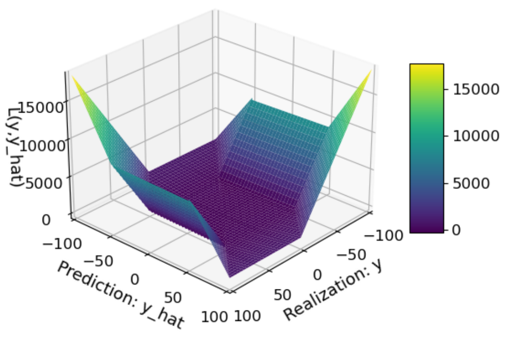
(c)



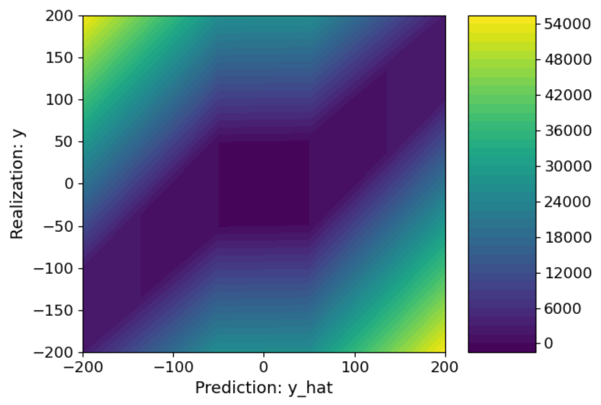
(d)

Figure B.3: Complete loss function (above) and the corresponding gradients (below) for $C_{gopacs} = 268$ €/MWh.

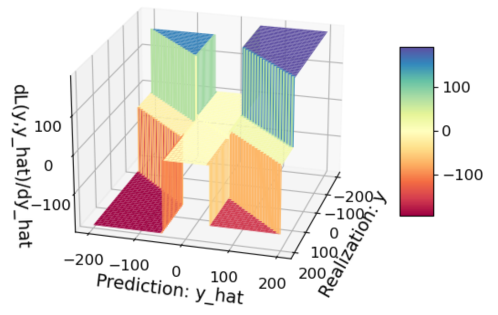
B.3. Loss function plots with $C_{gopacs} = 70$ €/MWh



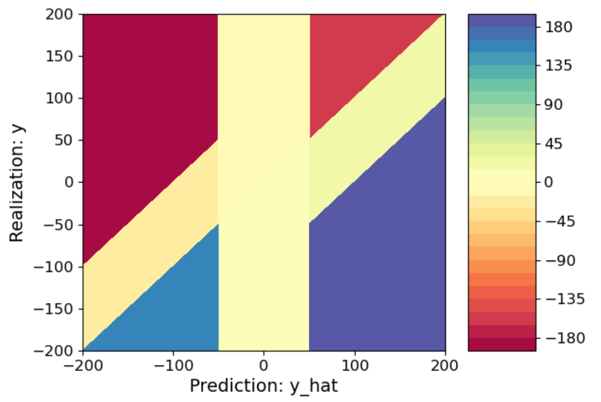
(a)



(b)



(c)



(d)

Figure B.4: Complete loss function (above) and the corresponding gradients (below) for $C_{gopacs} = 70$€/MWh.

B.4. Loss function implementation in Python

```python
import numpy as np


# training function called by XGBRegressor
def cost_train(y_true, y_predict):
    tlim = 50
    Cgop = 17.5 #67
    Ccon = 175
    e = 3

    MW_gop = (np.heaviside((y_predict - tlim), 0) + np.heaviside((-y_predict - tlim), 0))

    grad = Cgop * (np.heaviside((y_predict - tlim), 0) - np.heaviside((-y_predict - tlim), 0)) + \
           Ccon * (-1*np.heaviside((y_true - tlim - np.maximum(y_predict - tlim, 0) + \
                   np.maximum(-y_predict - tlim, 0)), 0) * MW_gop + \
                   np.heaviside((-y_true - tlim - np.maximum(-y_predict - tlim, 0) + \
                   np.maximum(y_predict - tlim, 0)), 0) * MW_gop) + \
           -e * (np.heaviside((y_predict + tlim), 1) * np.heaviside((y_true-y_predict),1) * \
                np.heaviside((-y_predict+tlim),1) -
                np.heaviside((-y_predict + tlim), 1) * np.heaviside((y_true+y_predict),1) * \
                np.heaviside((y_predict+tlim),1))

    # addition of 0 * y_true to generate array of size(y_true) instead of int(1)
    hess = 1 + 0 * y_true

    return grad, hess

# scoring function called by GridSearchCV
def cost_score(y_true, y_pred):
    k = 50
    Cgop = 17.5 #67
    Ccon = 175

    L = Cgop * (np.maximum(-y_pred - k, 0) + np.maximum(y_pred - k, 0)) + \
        Ccon * (np.maximum(-y_true - k - \
                np.maximum(-y_pred - k, 0) + np.maximum(y_pred - k, 0), 0) +
                np.maximum(y_true - k -
                np.maximum(y_pred - k, 0) + np.maximum(-y_pred - k, 0), 0))

    return np.sum(L)
```

C
Train/test set transformer data

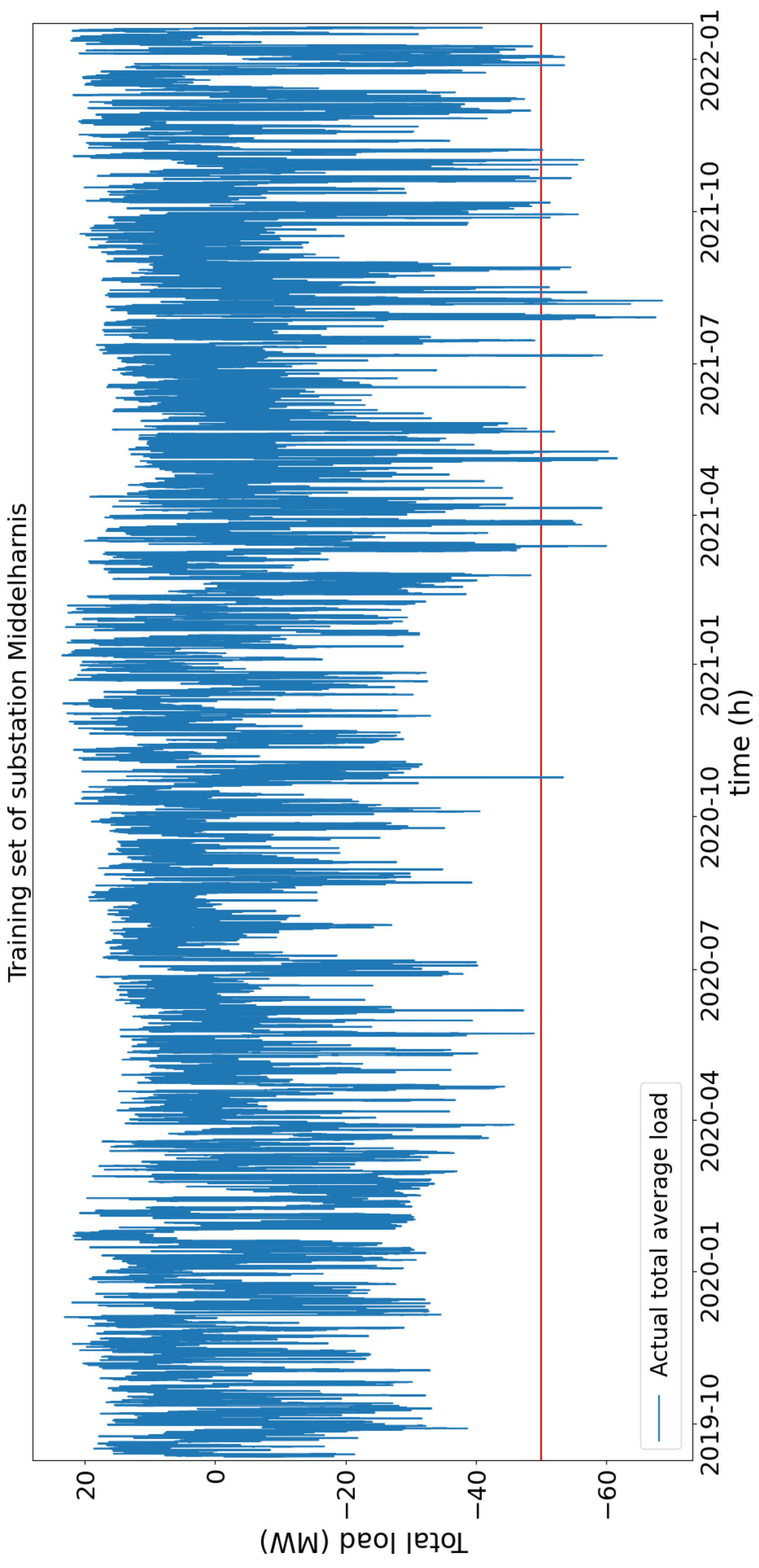


Figure C.1: Transformer data in the training set.

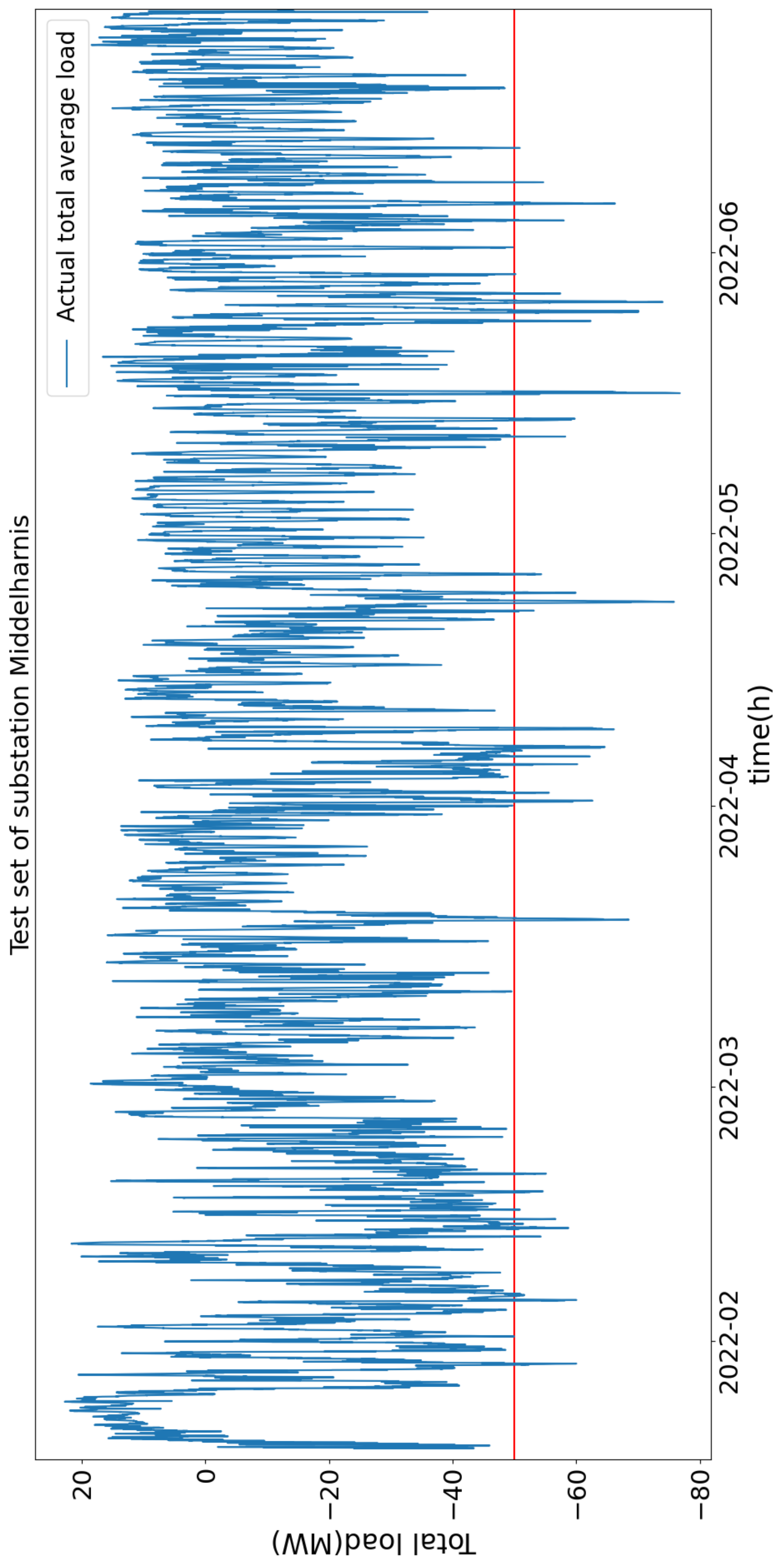
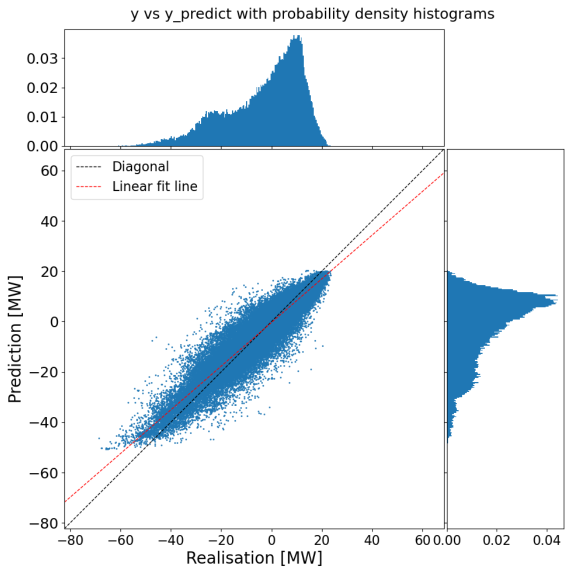


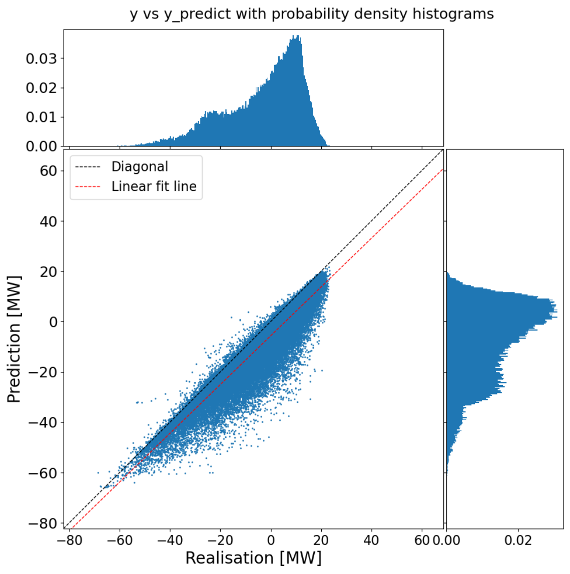
Figure C.2: Transformer data in the test set.

D

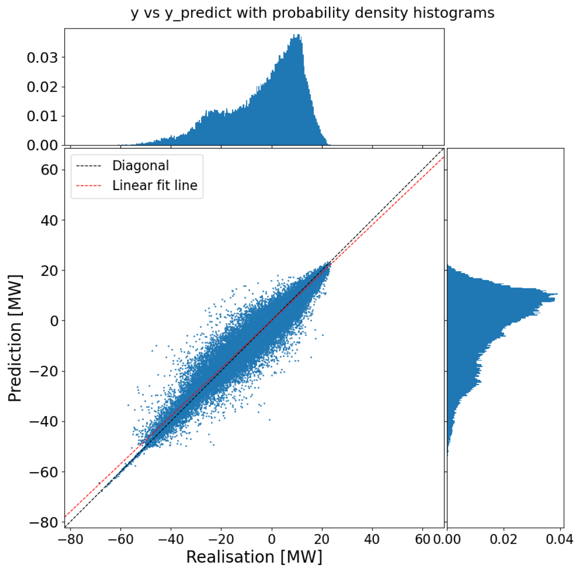
Results



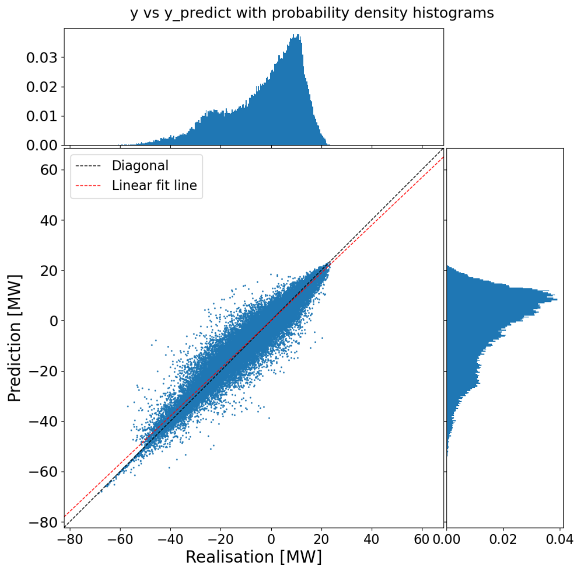
(a) MSE trained model



(b) Pinball trained model

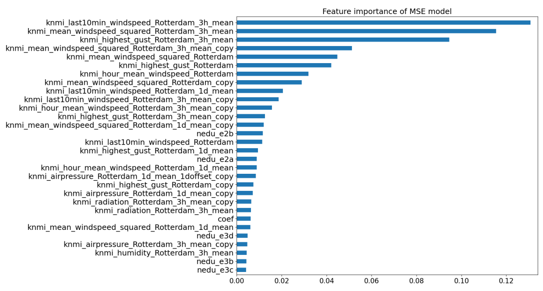


(c) Cost function trained model for $C_{gopacs} = 268$ €/MWh.

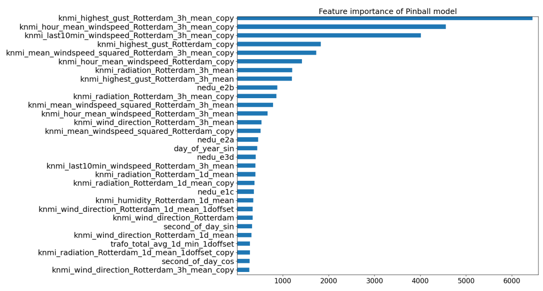


(d) Cost function trained model for $C_{gopacs} = 70$ €/MWh.

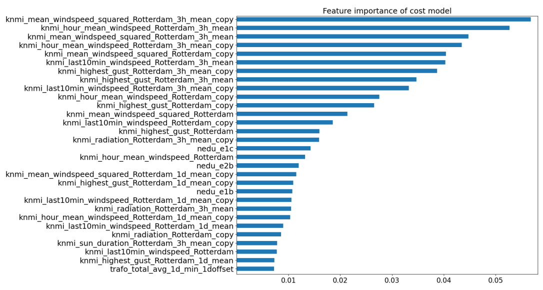
Figure D.1: Scatter plots of realization y versus predictions $\hat{y}$ in the training set, including probability density distributions for both on all three loss functions.



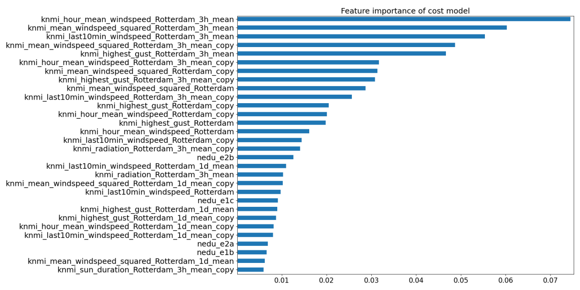
(a) MSE trained model



(b) Pinball trained model



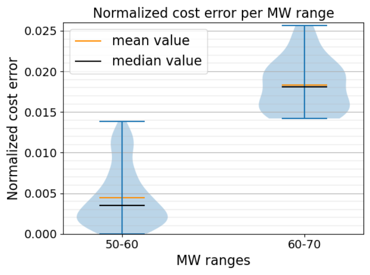
(c) Cost function trained model for $C_{gopacs} = 268$ €/MWh.



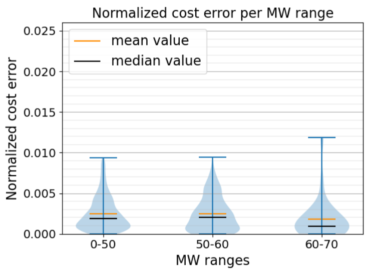
(d) Cost function trained model for $C_{gopacs} = 70$ €/MWh.

Figure D.2: Feature importance for the top 30 features per model. When a feature has the suffix '_copy', this signals that this a scaled feature, as detailed in the detrending step in chapter 4.The feature importance of the quantile function is not normalized, as it was trained with the LGBM package, which has a known bug with not being able to display features normalized.

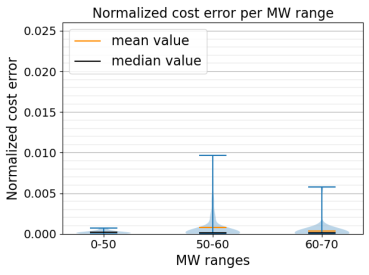
D.1. Mean Scenario



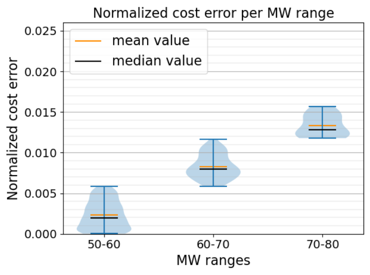
(a) MSE train set



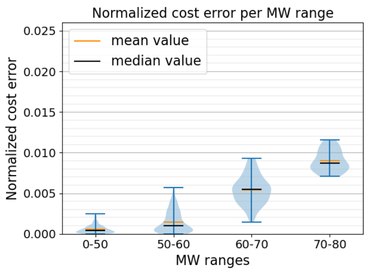
(b) Pinball train set



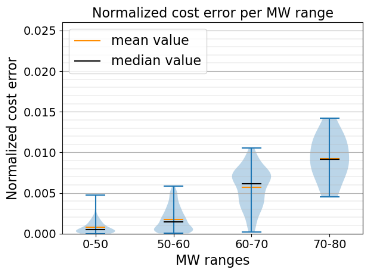
(c) Cost mean scenario train set



(d) MSE test set



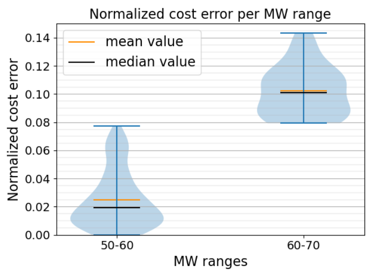
(e) Pinball test set



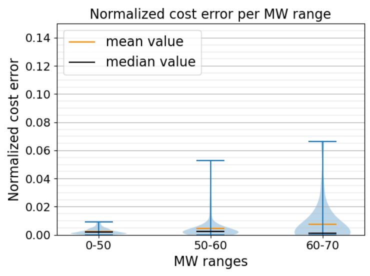
(f) Cost mean scenario test set

Figure D.3: Violin plots of the mean scenario for the three different models, showing the normalized cost error broken down per MW range. The top row are done on the training set, the bottom ones are from the test set.

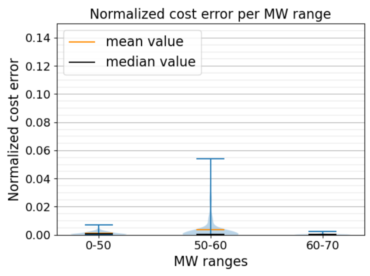
D.2. Minimum Scenario



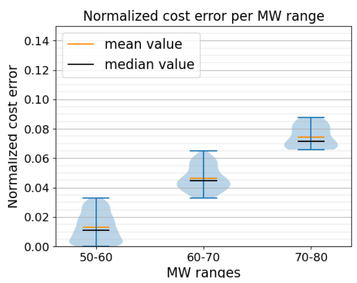
(a) MSE train set



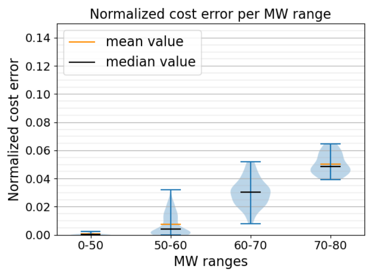
(b) Pinball train set



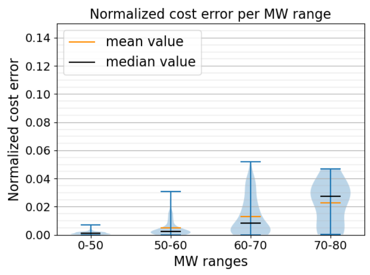
(c) Cost mean scenario train set



(d) MSE test set



(e) Pinball test set



(f) Cost mean scenario test set

Figure D.4: Violin plots of the mean scenario for the three different models, showing the normalized cost error broken down per MW range. The top row are done on the training set, the bottom ones are from the test set.