UNIVERSITY OF TWENTE

MASTER THESIS

Forecasting the controllability of electricity consumers using sparse multi-granularity time series data

Author:
Jeroen Peter BOS

Academic Supervisor:
Dr. Doina BUCUR

Daily Supervisor:
Dr. Rahul SHETTY

Examiner:
Dr. Alexander SKOPALIK

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Faculty of Electrical Engineering, Mathematics and Computer Science

September 3, 2021

Abstract

The growing effects of climate change and the introduction of cheap renewable energy bring transformational changes to electricity grids worldwide. Many grids resort to reducing the demand instead of increasing production to cover peak loads as this is often cheaper and reduces environmental impact. Parties providing the reduction of demand are called demand response providers. The amount of demand that such a provider can reduce is their potential performance. Leap is a demand response provider, and part of its operations require accurate forecasting of its future performance. Currently, Leap has a manual process to achieve this; however, this process is labour intensive and inaccurate. This study investigates the possibilities and potential for data-driven performance forecasting.

Leap provides a dataset consisting of many sparse time series that vary massively in scale, which is part of the nature of this problem. We developed two machine learning models for data-driven performance forecasting that circumvent the complications found in the data. The first model was developed to provide a solution to Leap at the earliest convenience, such that they could use it in their upcoming operational season. At the core of this model is a traditional machine learning technique for robust and fast development. The second model proposed an experimental approach to leveraging time series data consisting of a mix of monthly and hourly granularities. At the core of this model is a state-of-the-art deep learning technique for high-potential but complex development.

For both models, we achieve fully data-driven performance forecasts, and we can demonstrate very competitive accuracy. The base model demonstrates an improvement of about 10-20% on specific subsets of the data over the manual process. While the advanced model demonstrates an improvement of up to 70-80% on specific subsets of the data over the manual process. Additionally, we empirically show that our experimental approach to multi-granularity time series forecasting can make an improvement over comparable standard approaches.

We conclude that fully automated performance forecasts are feasible and can show attractive accuracies. Leap could benefit much from utilizing either of the prototypes developed in this study. The experimental approach to multi-granularity time series forecasting shows potential but has only been evaluated to a minimal extent in this study.

Contents

Abstract i

List of Figures iv

List of Tables v

Acronyms vi

Glossary viii

Symbols x

1 Introduction 1
1.1 Report Organization . . . 2

2 Background 3
2.1 Electricity Markets . . . 3
2.1.1 Deregulation . . . 3
2.1.2 Markets . . . 4
2.1.3 Demand Flexibility . . . 4
2.2 Leap . . . 5
2.2.1 Participation . . . 5
2.2.2 Goal and Requirements . . . 7
2.3 Research Questions . . . 7
2.4 Forecasting in the energy domain . . . 8
2.4.1 Distinctions . . . 8
2.4.2 Exogenous variables . . . 9
2.4.3 Performance metrics . . . 9
2.4.4 Model types . . . 10
2.4.5 Selected models . . . 11
2.5 Deep learning . . . 12
2.5.1 Loss . . . 12
2.5.2 Dropout . . . 13
2.5.3 LSTM & GRU . . . 13
2.5.4 Sequence to Sequence . . . 14
2.5.5 Attention . . . 14

3 Related Work 16
3.1 DeepAR . . . 17
3.2 MQ-RNN . . . 18
3.3 Temporal Fusion Transformer . . . 18
3.4 Selected architecture . . . 19

4 Data 20
4.1 Data sources . . . 20
4.2 Labelling . . . 21
4.2.1 Baselines . . . 21
4.3 Historical Nominations . . . 23
4.4 Data analysis . . . 23
4.4.1 Overview . . . 23
4.4.2 Meters by Metadata . . . 24

4.4.3 Relation with statistical characteristics . . . 24
4.5 Features . . . 25
4.6 Insights . . . 26

5 Base Model **29**
5.1 Data . . . 29
5.1.1 Processing . . . 29
5.1.2 Validation . . . 30
5.2 Model Type . . . 31
5.3 Optimization . . . 32
5.3.1 Feature Selection . . . 32
5.4 Results . . . 36
5.4.1 Error Analysis . . . 37
5.5 Discussion . . . 37

6 Advanced Model **40**
6.1 Data & Model . . . 40
6.1.1 Processing . . . 40
6.1.2 Features . . . 40
6.1.3 Time Series Dataset . . . 41
6.2 Optimization . . . 42
6.3 Results . . . 44
6.3.1 Error Analysis . . . 45
6.4 Discussion . . . 49

7 Discussion & Conclusion **50**
7.1 Insights . . . 50
7.1.1 Usage . . . 51
7.1.2 Limitations . . . 51
7.1.3 Reflection . . . 52
7.2 Future Work . . . 52
7.3 Conclusion . . . 53

References **55**

List of Figures

2.1 Revenue flows before deregulation and after deregulation of the grid. 4
2.2 The so-called “duck curve” as a result of an increase in roof-top solar. 5
2.3 A typical example of demand response on Leap’s platform. 6
2.4 Schematic overview of Leaps participation in the energy markets. 7
2.5 Schematic overview of the two most popular RNN specializations. (Olah, 2015) 13
2.6 Sequence to Sequence architecture 14

4.1 Distribution of the correlation between load and baseline. 22
4.2 Number of dispatched meters calendar. 23
4.3 Four typical meter time series. 24
4.4 Average historical load compared to the performance for different meter types. 25
4.5 Comparison of the performance with the average generation. 26
4.6 Comparison of the performance with the standard deviation of the historical generation. 27
4.7 Correlation of characteristics with the target label. 27

5.1 A schematic overview, detailing how the forecast horizon is determined. 30
5.2 The tabular dataset creation, to deal with the sparse time series and generate training samples. 30
5.3 A schematic overview to illustrate the process of expanding window walk-forward validation. 31
5.4 Schematic overview of the hierarchical base model. 32
5.5 A schematic overview of the base model’s optimization process. 33
5.6 The development of the negative log-likelihood per generation in the binary genetic algorithm for feature selection. 35
5.7 The feature presence across generations during feature selection using the binary genetic algorithm. 35
5.8 Comparison of the manual process with the base model on several key metrics. 36
5.9 A scatterplot showing how the manual nominations and the model nominations relate to the observed performance. 37
5.10 Comparison of the manual process with the base model on several key metrics for several subsets of meters, determined by the size of the performance. 38
5.11 Comparison of the mean bias and the root mean squared error for predictions by the base model for different partners. 38

6.1 A schematic overview of the extraction of training samples from multiple time series in a single-granularity time series forecasting problem. 42
6.2 A schematic overview of the extraction of training samples from multiple time series in a multi-granularity time series forecasting problem. 43
6.3 The advanced model’s optimization process. 44
6.4 A comparison of all the approaches using our key performance metrics across each month. 45
6.5 A comparison of all the approaches using our key performance metrics across each month and separated by load scale. 47
6.6 A comparison of all the approaches using our key performance metrics across each month for hot-start and cold-start meters. 48
6.7 A comparison of the ρ-risk for several prediction quantiles between the base and advanced models. 48

List of Tables

2.1 Overview of related work in relation to this research. 9

4.1 The most relevant fields in the meter baselines data source. 20
4.2 The most relevant fields in the meter metadata source. 21
4.3 The most relevant fields in the meter authorizations source. 21
4.4 Performance metrics on the historical nominations. 22

5.1 The permutations of data range, selected intervals and metrics of the initial feature exploration. 33
5.2 The selected subset of features that showed the highest correlation with the performance, to be used in the base model optimization process. 34
5.3 A description of the hyperparameter search space of the advanced model. 34
5.4 The best found configuration of features and hyperparameters for the base model. . . 36

6.1 The best found configuration of features and hyperparameters for the advanced model. 44
6.2 The number of dispatched meters and load quantiles by month and load type for the error analysis. 46
6.3 The overprediction rate by quantile. 47

Acronyms

ANN Artificial Neural Network. 11–13, 16, 19, 45

API Application Programming Interface. 6

AR Autoregressive. 10, 18

ARIMA Autoregressive Integrated Moving Average. 10, 41

ARMA Autoregressive Moving Average. 10

BGA Binary Genetic Algorithm. 34, 52

BP Back Propagation. 11

CAISO California ISO. 5, 6, 21

CSP Curtailment Service Provider. 3, 5, 6

DBN Deep Belief Network. 11

DEX Distributed Energy Exchange. 6, 21

DL Deep Learning. 11–13, 15, 17, 40, 41

DR Demand Response. 3–8, 23, *see*

ERCOT Electric Reliability Council of Texas. 4

GA Genetic Algorithm. 34

GRU Gated Recurrent Unit. 14, 16

IOU Investor Owned Utility. 3, 5, 6, 29

IPP Independent Power Producer. 3–5

ISO Independent System Operator. 3, 6, 8

LMP Local Marginal Price. 4

LSTM Long Short-Term Memory. 13, 14, 18

LTF Long Term Forecasting. 9, 11

MA Moving Average. 10, 23, 24

MAE Mean Absolute Error. 10, 44, 46, 47, 50, 52, 53

MAPE Mean Absolute Percentage Error. 10

MBE Mean Bias Error. 36, 46

MGTSD Multi-Granularity Time Series Dataset. 40–42, 44, 45, 49, 53, *see Glossary* multi-granularity time series dataset

ML Machine Learning. 10, 11, 13, 19, 41, 44

MLP Multilayer Perceptron. 18

MTF Mid Term Forecasting. 9

NLP Natural Language Processing. 14, 15, 18, 19

QL Quantile Loss. 10

RMSE Root Mean Squared Error. 10, 36, 44, 46, 49, 52, 53

RNN Recurrent Neural Network. 13, 18

RTO Regional Transmission Organization. 3–5

SARIMA Seasonal ARIMA. 10, *see Acronyms* ARIMA

Seq2Seq Sequence to Sequence. 14, 17, 18

STF Short Term Forecasting. 9

SVM Support Vector Machine. 11

SVR Support Vector Regression. 11

TFT Temporal Fusion Transformer. 18, 19, 40–43

TSD Time Series Dataset. 40, *see Glossary* time series dataset

VSTF Very Short Term Forecasting. 9

Glossary

***k*-fold cross-validation** Validation strategy in which the data available for training is split up into k equal sized folds. In each of the k iterations, one fold is used as validation and the remaining folds are used for training.. 30

ablation Ablation is the removal of a component from a machine learning system. This is often used to quantify the effect of said component.. 45

baseline A metric to compare against to determine the amount of curtailment of DR, calculated using a methodology defined by the RTO. 5

bias Systematic prejudice in an algorithm's results.. 10

curtailment The reduction of electricity consumption in DR as measured by the difference between the baseline and the actual load.. 4, 5, *see Acronyms* DR &

data augmentation The process of creating new data by altering existing data. Often used to create more realistic training samples from a single limited dataset.. 49

dispatch Command to activate energy production / demand response as a result of a bid into one of the energy markets. see. 4, 5, 21

DR Purposeful changes in electricity usage by end-use customers from their normal consumption patterns in response high load on the grid. *see Acronyms* DR

energy market A digital market that is run by ISOs in order to balance supply and demand.. 4, 5, 8, 9, *see Acronyms* ISO

ensembling The process of combining the predictions of multiple models to create a (hopefully better) unified prediction.. 49

expanding window The expanding window is a variation of the walk forward validation in which the validation sets of different slices are allowed to overlap.. 31

generation In terms of DR, synonymous to curtailment.. 5, 21, *see Glossary* curtailment

heteroscedastic Refers to the variable variance of the error terms in the predictions of a regression model. The opposite of homoscedastic.. 32

homoscedastic Refers to the constant variance of the error terms in the predictions of a regression model. The opposite of heteroscedastic.. 32

imputation The process of replacing missing values. There are several strategies, but the most common ones are replacing with a fixed value or replacing with the mean of the present values.. 29

information leakage Information leakage is when certain information is accessible to a machine learning model that should not be accessible. It often occurs very subtle and indirect, for example trough hidden conditional variables. Information leakage could occur when training and validation sets are not properly separated in time series forecasting.. 30

MarketOps Leap's market operations department. 6, 21

meter A device at the consumer endpoint which measures electricity usage and sends that data to the distribution utility. 6–8, 10

multi-granularity time series dataset A time series dataset that is able handle multiple granularities. The training examples extracted from this dataset will have a different granularity for the encoder and decoder.. *see Glossary* time series dataset

nomination A suggestion for the expected performance of a meter during a specific month. 6, 7, *see Glossary* performance

partner A partner of Leap that has integrated with the DEX which enables their clients / customers to participate in DR. 6, *see Acronyms* DEX & DR

performance The maximum two hour consecutive generation during an entire month.. 6, 21, *see Glossary* generation

probabilistic forecast Forecasting a distribution or quantiles instead of a single value.. 31

resource An entity which is able to participate in energy markets by generating electricity as a generator or curtailing demand as a set of meters. 5, 6

resource adequacy A program ran by CAISO to ensure that enough energy is available to meet demand in the long term future. 5

supply plan Indication of power production in a certain month to be submitted to CAISO and IOUs for resource adequacy. 6, *see Acronyms* CAISO & IOU

walk-forward validation Common k-fold cross-validation strategy for time series forecasting in which information leakage from the validation set to the train set trough time is prevented.. 30

Symbols

Symbol	Unit	Description
z	kW (kilowatt)	Target performance
$\hat{z}$	kW (kilowatt)	Predicted performance
x	kW (kilowatt)	Load value
$\hat{x}$	kW (kilowatt)	Baseline value
i		A meter / time series
N		The total number of meters / time series
m		A month
M		The last month
ω		A 15-minute interval
Ω		The set of 15-minute intervals in a month
d		Dispatch indicator
ρ		Quantile used in prediction or validation
$\hat{z}^{\rho}$	kW (kilowatt)	The ρ-quantile performance prediction

Chapter 1

Introduction

Electricity grids around the world are undergoing transformational changes. In the past 50 years, the United States (US) grid underwent deregulation and experienced a rapid increase in demand. The past 20 years brought cheap renewable energy and distribution-of-production through roof-top solar. These developments have led to significant changes in the operation and management of the grid. These developments have simultaneously spawned novel technological and economic opportunities.

In the electricity market, unlike other commodities, power supply must always exactly meet the demand. A surplus in electricity production would lead to an unstable grid, potentially breaking connected devices and machines. A surplus in demand for electricity leads to blackouts for certain areas of the grid. Accurate forecasts of the electricity demand or generation lead to reduced costs in electricity production and maintenance of the grid. Therefore, data and algorithms have played a major role in balancing the grid for a long time. Due to cheap renewable energy and roof-top solar, deep valleys and steep peaks in demand have become a daily occurrence. These steep ramp-ups make it increasingly difficult for traditional electricity producers to respond adequately and timely.

Over the last few years, an area that gained increasing interest in the field is demand response. It is the practice of purposefully reducing electricity consumption to "flatten the curve" of the peak demand. Traditionally, demand response was only accessible to large electricity consumers like factories. Leap[1] is a young company enabling third party small electricity consumers, from shopping malls to homeowners, to participate in demand response. As an active participant in balancing the grid, Leap is obligated to forecast its demand reduction capacity. Manual forecasts are feasible and common practice in the field. The availability of large amounts of data presents major opportunities for algorithms to play a role in the processes at Leap.

Objective

Leap has an obligation to the grid operators to forecast its available capacity. Currently, Leap has a manual process to forecast this capacity. For each device connected to Leap's platform, a forecast is made in collaboration with the respective partner. These predictions are aggregated and used to inform the grid operator of the capacity. Leap currently believes that the manual process of creating these forecasts is inaccurate and too time-consuming. Automated, data-driven capacity forecasts could therefore provide significant benefit to Leap's operations. This thesis aims to explore the possibility of developing accurate capacity forecasts fully based on historical meter data.

Electricity consumption is highly dependent on the weather. As a result, the need for demand response is very seasonal, just like the weather. The main operational season for demand response providers constitutes the summer months. Leap wants to improve its demand forecasting process before the coming operational season. As the capacity forecasts need to be submitted to the grid operators three months ahead, we need to deliver a solution within a few months.

One of the datasets that Leap provides has a monthly granularity and contains the historically demonstrated capacity. Due to the nature of demand response, this dataset is very sparse. Another dataset that Leap provides is not sparse and very related to the capacity; however, that data is at a 15-minute granularity. This means that a solution would have to deal with sparse time series or handle multi-granularity data.

As stated before, data and algorithms can make a tremendous impact on the operational efficiency of the grid. This explains the large body of research available on forecasting in the energy

[1] https://leap.energy/

domain. Unfortunately, most of this research is focused on demand forecasting of highly aggregated loads of large electricity consumers for grid operators. The current research is different, as we are forecasting demand reduction potential as a function of a signal to reduce demand. Additionally, we are dealing with either sparse or multi-granularity data. The body of research covering the overlap between this type of data and this target variable is minimal.

This research

Leap wants to improve the accuracy of its predictions and reduce the amount of time spent on manual capacity forecasts. The number of meters connected to Leap's platform is growing exponentially, as is the amount of data. However, the largest challenge to develop a data-driven approach to solve this problem is not regarding the size but the shape of the data. The nature of the problem and the company's age results in a dataset that contains very sparse time series on a short time span. Any solution to the problem needs to be able to deal with the shape of the available data. To consider Leap's needs regarding their operational season, we approach the problem using two separate methodologies.

In the first methodology, hereinafter referred to as the base model, we focus on delivering a solution in a short time frame, such that Leap can use it in the upcoming operational season. At the core of the base model is a more traditional machine learning technique. Traditional machine learning models are usually more interpretable, cheaper to train and less likely to produce a null result. To deal with the shape of the data, we develop a process to extract features from the time series and transform it into a tabular format.

In the second methodology, hereinafter referred to as the advanced model, we develop a solution in the time remaining for this research. At the core of the advanced model is a state of the art deep learning model. Deep learning models provide a lot of flexibility to shape the model to match the problem perfectly. However, they are hard to interpret, expensive to train and can provide null results if applied incorrectly. To deal with the shape of the data, we develop an experimental process to extract training samples from a combination of the monthly and 15-minute granularity time series.

Contribution

This research contributes to the efforts of Leap and the academic world in the following ways:

- The potential of forecasting demand response capacity for small-scale electricity consumers is explored.
- A robust model that creates accurate capacity forecasts is provided to Leap before the start of its operational season.
- A general framework for multi-granularity time series forecasting is proposed and tested on a small scale.
- A complex model that creates even more accurate forecasts using multi-granularity forecasting is provided to Leap.

1.1 Report Organization

In Chapter 2, detailed background information on electricity markets and Leaps relevant internal processes is provided. Using the domain-specific concepts and vocabulary, we then define the research questions. The chapter will conclude with an overview of traditional forecasting techniques in the energy domain and background information on state-of-the-art deep learning models. In Chapter 3, a review of related work, most relevant to state-of-the-art time series forecasting, is provided. Chapter 4 provides details about the available data, an extensive data analysis and correlations with the target values of extracted features.

In Chapter 5, we discuss the base model. In Chapter 6, we discuss the advanced model. In these chapters, we discuss the data preparation, model optimisation, results, and a review of the methodology for both of these models. And finally, we tie the discussions from both approaches together and provide a conclusion to this research in Chapter 7.

Chapter 2

Background

This research focuses on demand response, which is an integral part of deregulated electricity markets. Hence, we aim to provide you, the reader, with a very simplified introduction to the inner workings of the current state of the power grid (Blumsack, 2017). Additionally, we will shed light on Leap's role in the electricity market and Leap's requirements for the algorithm developed in this research. Then, we will continue with general information about forecasting in the energy domain and follow up with some building blocks for the related work on deep learning architectures.

2.1 Electricity Markets

Electricity is very different from other commodities; supply needs to match demand at any second of any day. Undersupply of electricity leads to decreased power supply frequency, while oversupply leads to increased frequency. As the devices we connect to the grid are designed to work on a specific frequency with a tight tolerance for deviations, supply and demand must always be balanced.

The electricity supply chain consists of several parts. First is the generation of electricity, second is the transmission of high voltage[1] electricity over large distances, and third is the local distribution of electricity at a low voltage. Historically, the complete supply chain was managed by a highly regulated for-profit electric utility in the US. However, this changed when many US states voted to deregulate the power grid at the end of the 20th century.

2.1.1 Deregulation

Instead of having a single organization manage all aspects of the electricity supply chain, deregulation split the electric utility into smaller organizations responsible for either generation, transmission or distribution. In addition, the electricity reform and new regulations introduced many new types of companies to the grid.

One of the new organizations is central to the power grid, the ***Regional Transmission Organizations*** (RTOs). RTOs are non-profit organizations that do not own any part of the generation, transmission or distribution process; they are the "balancing authority" of a regional part of the grid. They are responsible for running electricity markets, operating the transmission grid and system planning to ensure sufficient generation. The generation is currently in the hands of ***Independent Power Producers*** (IPPs), some of which were part of the former electric utility, while others are new companies formed after the deregulation. The IPPs are for-profit organizations that own and operate the power plants and sell the electricity into the markets that are run by the RTOs. The ***Curtailment Service Providers*** (CSPs) are for-profit companies that are practising ***Demand Response*** (DR). By providing reduced demand, the CSPs are direct competitors of IPPs.

The RTOs balance the grid by running an electricity market in which IPPs and CSPs offer capacity. They re-sell the electricity with a markup for the transmission fees to the ***Investor Owned Utilitys*** (IOUs). The IOUs are responsible for the local distribution; they then charge their customers for this electricity. An overview of the revenue flows is provided in Figure 2.1. ***Independent System Operators*** (ISOs) are similar to RTOs but usually cover a smaller geographic area, often limited to a single state.

[1] Transport of electricity is more efficient when using higher voltages.

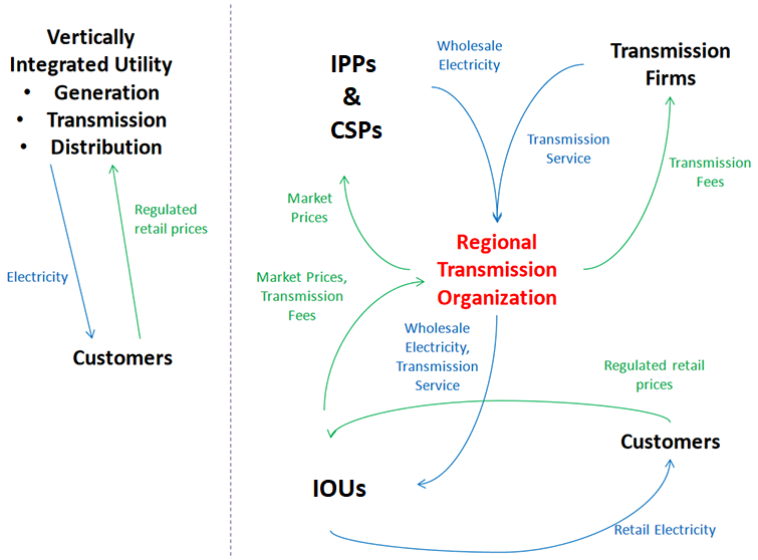


FIGURE 2.1: Flows of money before deregulation (left) and after deregulation (right) (Blumsack, 2017)

2.1.2 Markets

There are many markets that RTOs run to keep the grid in balance and the costs of electricity low. They appear in three different flavours: the capacity market, the energy market and the ancillary services market. The RTO forecasts the demand or desired capacity for all markets and tries to satisfy this by running auctions. The RTO accepts the bids with the lowest costs until it can satisfy the forecasted demand of that particular market. All bids will be paid based on the highest accepted bid, which incentivizes participants to place bids as low as they can afford. As energy is lost during transmission and as the shape of the grids can lead to bottlenecks, the energy price is not uniform across the grid. The local price in a certain geographic area is called the ***Local Marginal Price*** (LMP).

In this research, we deal mostly with capacity markets and energy markets. Most RTOs run two ***energy markets***. A day-ahead market that accepts bids 24 hours in advance and a real-time market that accepts bids one hour in advance. Participants bid production of energy or curtail of demand into the market for a time slot at a certain price. The bids with the lowest price per unit of energy are accepted until the expected demand matches the supply. The bids that got accepted are expected to produce energy or curtail demand; they are "dispatched" for this time slot. The maximum price in the energy markets is government-regulated. As a result, it can be difficult for RTOs to find investors to build IPPs for infrequent peak demand production. [2] Capacity markets aim to alleviate this problem by running capacity auctions in which participants get paid to be stand-by generation.

2.1.3 Demand Flexibility

Historically, the electric power industry went through much change; these changes were slow and steady. The current state of the electricity markets run by RTOs is one in which there are many generators with different efficiencies and other varying characteristics competing against each other. The recent plummeting of wind/solar power costs and the introduction of "smart" electricity grids have disrupted these electricity markets.

These cheap renewable energy sources can only produce electricity during specific times of the day, and their performance is driven by weather, making them highly variable. Roof-top solar puts electricity producers in the distribution segment of the energy supply chain, which complicates local stability. Furthermore, the "smart" grid enables individual consumers to participate in the electricity market by getting paid for DR, the practice of curtailing energy usage to help balance

[2] This became painfully apparent around the time of writing (February 2021) in Texas during extreme colds in territory operated by the ***Electric Reliability Council of Texas*** (ERCOT).

the grid. Examples of DR include retail stores that temporarily decrease the power of their cooling installation or electric vehicles owners pausing their battery charging for a limited time.

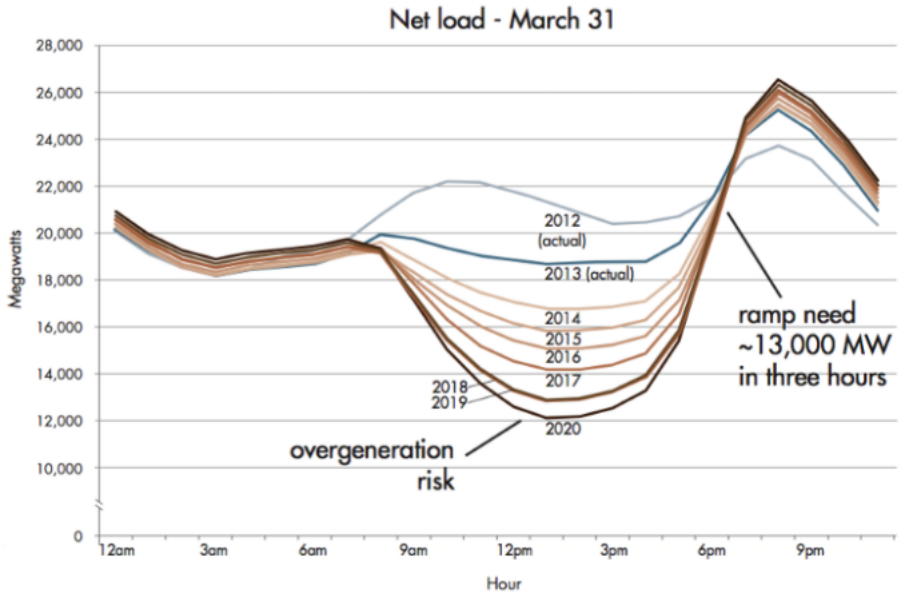


FIGURE 2.2: Introduction of roof-top solar to the grid has created the so-called "duck curve", which is characterized by a drop in energy demand around noon and a fast ramp-up towards 6 pm. (Dean Levi[3], NREL)

Roof-top solar changed the typical demand curve dramatically. Instead of a peak, there is a large drop in demand in the middle of the day, resulting in the duck curve in Figure 2.2. Around 6 pm, the power of solar panels drops dramatically, and at the same time, people return to their homes and start using more electricity. This can result in an increase in demand of up to 100% within only three hours. Fortunately, DR can make a big difference by flattening the peak in the demand curve.

In the electricity markets run by RTOs there is direct competition between IPPs and CSPs. However, DR is often cheaper and more environmentally friendly than power plants designed to handle peaks in demand. These demand peaks are becoming increasingly more common due to cheap renewable energy sources. The largest source of new bids in capacity markets has recently been by CSPs (Blumsack, 2021), which is where DR finds most of its revenue. The DR resources get dispatched after bidding into the energy markets. Afterwards, the ***curtailment*** of the resources is measured by comparing the consumption pattern with a ***baseline*** as defined by the RTO. CSPs provide (virtual) ***generation*** in the form of demand response. In Figure 2.3, typical consumption patterns for a DR participant are displayed. As CSPs are competing against peaking power plants that are idle most of the time, it makes sense that the introduction of DR has dramatically lowered prices in the capacity markets.

2.2 Leap

Leap is operating in multiple regions. This research only focuses on Leap's largest operation region, the California market, where the CAISO runs a state-wide energy market. CAISO operates an energy-only market; this means there is no public auction-based capacity market. However, the state does have a ***resource adequacy*** program to ensure sufficient capacity. This program requires IOUs to have a certain amount of capacity contracted with IPPs and CSPs.

2.2.1 Participation

Now the question arises, where does Leap stand in this ecosystem that makes up the grid? The answer to this question is schematically displayed in Figure 2.4. Leap participates in the energy

[3] http://www.caiso.com/Documents/Flexibleresourceshelprenewables_FastFacts.pdf

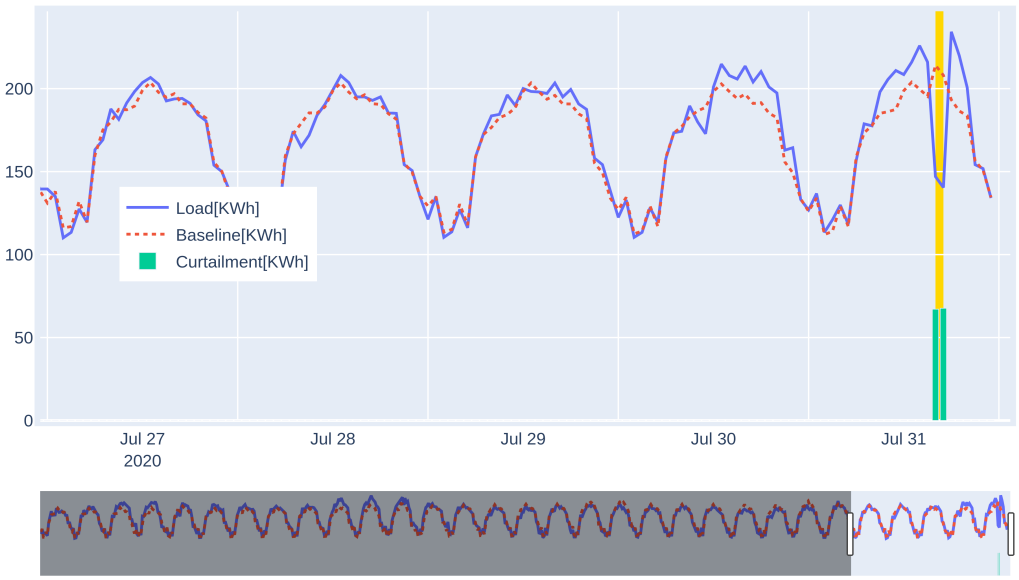


FIGURE 2.3: A typical example of demand response on Leap's platform. The blue line shows the observed load consumption. The red dotted line shows the baseline as defined by ***California ISO*** (CAISO); further details are provided in Section 4.2.1. The yellow area denotes a market dispatch with the green bars showing the curtailment this DR participant is providing.

markets as a CSP; however, it does not provide the DR itself. Instead, Leap builds a ***Distributed Energy Exchange*** (DEX), in which many small electricity consumers collectively participate in DR. ***Partners*** of Leap usually have a product or service for which they build digital integrations with the DEX using an ***Application Programming Interface*** (API). Clients and customers of the partners can then easily opt into DR and participate collectively. The electricity usage of these clients is measured by the IOU using ***meters***. As part of participating in DR, the electricity consumers allow the IOU to share their usage data with Leap. As a part of the resource adequacy program, Leap is required to create ***supply plans*** for each month a few months ahead of time. The supply plans are then submitted to the IOUs and CAISO. Leap aggregates the meters into ***resources*** and takes care of bidding in the market, processing dispatches, creating supply plans and settling for revenue with CAISO and the IOUs. As a result, participating in DR becomes very accessible for many small electricity consumers.

Monthly ***performance*** for meters and resources is defined as the maximum two-hour consecutive generation during dispatches over the entire month. As stated before, Leap is required to provide a supply plan, indicating the expected performance, to the ISO and IOUs for each resource three months in advance. A ***nomination*** is an estimate of the performance of a meter for a given month. Currently, Leap's partner-success department makes nominations for the meters in consultation with their respective partner. ***MarketOps*** creates and frequently changes the resources. For each resource, they aggregate the meter nominations into supply plans. As under-performance on the supply plans leads to penalties and over-performance on the supply plans lead to missed revenue, it is important to create accurate supply plans. Historically, Leap has seen a lot of under-performance, which can be explained by poorly optimized resource configurations and inaccurate meter nominations. As business rules largely dominate the process of resource configuration, the focus of this research is only on improving the meter nominations.

Many meters on Leap's platform have never been dispatched before but still need a performance forecast. Forecasting in the absence of historical datapoints is referred to as a **cold-start problem**. Most meters get dispatched very rarely, and none of the meters gets dispatched every month. As a result, the time series of monthly performance is often **very sparse**. A dataset with a monthly granularity containing the capacity and a dataset with the 15-minute granularity load patterns are available. The higher granularity load patterns dataset might be useful in forecasting the lower granularity monthly performance dataset.

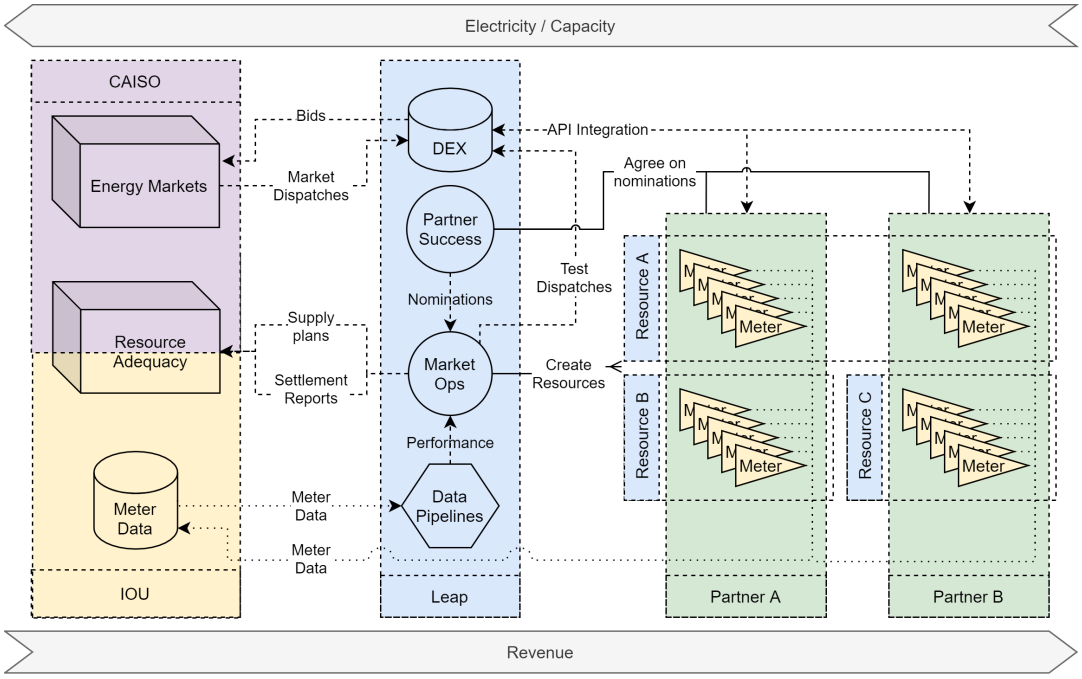


FIGURE 2.4: Schematic overview of Leaps participation in the energy markets and its interaction with several parties.

2.2.2 Goal and Requirements

The algorithms developed in this research should provide meter level nominations. The goal is to provide more accurate nominations compared to the manual process that is currently in use and to do so in a scalable automated way. It is important to be mindful of the sparsity of the time series and the cold-start scenario. In both cases, the model should still be able to make accurate nominations. The sparsity of the time series only exists on the monthly performance level. However, the monthly performance is calculated from the meter load and baseline time series at the 15-minute interval granularity. Therefore, an attempt is made at using multi-granularity forecasting to create the nominations.

In July, August and September, the total demand on the grid is the highest. As a result, DR is often dispatched during these months. This is Leap's main operational season. Leap has requested that an initial base model be provided at the start of the research. They want to use the results from this base model in the operational season of 2021. The rest of the time in this research is spent on developing the advanced model. The advanced model uses more complex state-of-the-art technologies. These technologies come with a higher risk of a null result.

2.3 Research Questions

Using the previously described vocabulary, we define the following research question:

> ***Research Question:*** *How can we forecast the performance of electricity consumers using data such as the historical load, historical performance and meter-specific characteristics to create accurate meter nominations?*

Additionally, as a support to the main question and to gain further knowledge, the research strives to answer the following subquestions:

> ***Sub Question 1:*** *To what extent can we forecast the performance in cold-start scenarios?*
>
> ***Sub Question 2:*** *To what extent can we forecast the performance using sparse performance time series?*
>
> ***Sub Question 3:*** *How can we combine data with different granularities into a single forecasting algorithm?*

Sub Question 4: *By what margin does the proposed algorithm outperform human nominations made using domain knowledge?*

2.4 Forecasting in the energy domain

Much research exists on forecasting time series in the energy domain. However, barely any of these works focus on forecasting DR capabilities. Therefore, this section contains broad background information on forecasting in the energy domain while trying to align as close as possible with the objective of this research. We can make a few important distinctions across the papers presented in this field and their objectives.

2.4.1 Distinctions

In general, there are four ways to make distinctions in the existing works in this field that are relevant to this research. Table 2.1 shows how these papers differ or relate to this research.

Direction

The first distinction is the forecasting of consumption patterns versus the forecasting of production patterns. Forecasting consumption patterns is often applied by ISOs to run the energy markets. The other type of forecasting that is prominent in the field is forecasting the production patterns of renewable energy sources. In this research, we are aiming for a combination of these. By definition, the virtual production of Leap is based on the consumption of its connected meters.

Garulli, Paoletti, and Vicino (2015) research a DR situation that is closely related to our research in terms of direction. They model the demand using an exogenous input for the equivalent of a dispatch signal. The model can forecast curtailment by forecasting both the load with and without a dispatch signal.

Sato and Azuma (2016) research the controllability of electricity consumers. They forecast the electricity demand as a function of the real-time electricity price. This form of controllability is different but relatively similar compared to this research.

Aggregation level

The second distinction is the forecasting of highly aggregated patterns versus forecasting for small consumers or producers. Highly aggregated patterns display relatively lower variance and higher predictability (Hernández, Baladrón, Aguiar, Calavia, et al., 2014). This is explained using a term borrowed from supply chain management, risk pooling.

> This effect can be explained using the following example. Our goal is to predict the percentage of electricity consumers that turn on their laundry machine at a certain time. Let's say that we know the probability that each electricity consumer turns on its laundry machine at that time. A small group of such consumers has a highly varying percentage of turned on laundry machines as each consumer has a large impact on that value. A large group of such consumers has a low varying percentage of turned on laundry machines as each consumer has a small impact. As a result, the group's percentage of turned on laundry machines is closer to the mean of the probabilities of the individual consumers. The small group has a higher relative variance.

Kuster, Rezgui, and Mourshed (2017) provide a good overview of different aggregation levels and which models are commonly used in which situation. In this research, the aggregation level is at the scale of single houses or stores, similar to Hsiao (2015) and Mathieu, Price, Kiliccote, and Piette (2011).

Related time series

The third distinction is modelling to forecast a single time series or modelling to forecast many related time series. In the case of demand forecasting by the ISOs, there is a single time series with a long history. However, when demand is forecasted for individual buildings, we deal with many

related time series. In this research, we have many related time series with many missing gaps. Therefore, a model might be able to benefit from looking at the related time series.

Horizon and resolution

When forecasting a time series, it is important to make two choices depending on the problem. First, choosing the size of the horizon, how far into the future do we want to forecast. Second, choosing the resolution, at which granularity do we want to forecast.

The horizon is typically classified into four categories. ***Very Short Term Forecasting*** (VSTF) has a horizon of less than an hour and is often used for frequency regulation. ***Short Term Forecasting*** (STF) has a horizon of up to a few days and is often used in the energy markets and other kinds of system planning. ***Mid Term Forecasting*** (MTF) has a horizon up to a few months; there exists very little research here (Kuster et al., 2017). ***Long Term Forecasting*** (LTF) has a horizon starting at a year, which is often used for system and capacity planning. The resolution can be classified as sub-hourly, hourly, daily, monthly, seasonal or annual, and is often related to the horizon.

This research deals with MTF with a monthly resolution. However, there is minimal research available for this horizon. Therefore most related research comes either from STF or LTF.

2.4.2 Exogenous variables

As noted by Kuster et al. (2017), Mamun et al. (2020) and Das et al. (2018) we can divide the exogenous variables used for energy forecasting into at least the following four categories. **Socio-economic** features, like the average wealth in a certain zone, are often used for high aggregate forecasts. **Meteorological** features, like the temperature and humidity, are used in MTF and shorter horizons due to their impact on both consumption and production. Meteorological features are not used for LTF due to unpredictability in the long term. **Building** features, like the number of machines in a factory, are traditionally used for forecasting demand of single buildings with a large demand like in the research of Mena et al. (2014). **Time index** features, like the day of the year or hour in the day, are often used, most often in combination with models that were not designed to handle time series but can be used in this way using the time index features.

This research mainly focuses on the historical patterns with time index features, as we know from domain expertise that these are most influential for meter performance. We can consider meteorological features; however, weather forecasts usually do not extend past two weeks. Therefore these features are ignored as the increased likelihood of higher temperatures in the summer can be expressed using time index features. Some building features are included as we know what type of device Leap dispatches, such as an electric vehicle or a smart thermostat. Finally, there is the opportunity to include some socio-economic features as it is possible to retrieve a meter's historical electricity pricing.

2.4.3 Performance metrics

To evaluate the performance of forecasting methods, there exist several metrics. There is a clear distinction in performance metrics for point forecasts and probabilistic forecasts. Each of which has its advantages and disadvantages (Mamun et al., 2020).

	This Research	Paper I	Paper II	Paper III	Paper V	Paper IV
Direction	**Curtailment**	Consumption	Consumption	Curtailment	Consumption	Generation
Aggregation	**Building**	Building	District	District	Building	Building
Time series	**Multiple**	Single	Multiple	Single	Single	Single
Horizon	**MTF**	STF	STF	STF	VSTF	Includes MTF
Resolution	**Monthly**	Sub-hourly	Hourly	Hourly	Sub-hourly	Includes Monthly

TABLE 2.1: Overview table displaying how the goal of this research compares to several related works in the field of electricity forecasting. The aliases are Paper I: Mena et al. (2014), Paper II: Hernández, Baladrón, Aguiar, Calavia, et al. (2014), Paper III: Garulli et al. (2015), Paper IV: Das et al. (2018) & Paper V: Hsiao (2015).

Point forecast metrics

There are several metrics for point forecasts that are commonly used. Most of these are described by Das et al. (2018). The ones that are most often used in related work are ***Root Mean Squared Error*** (RMSE), ***Mean Absolute Error*** (MAE), ***Mean Absolute Percentage Error*** (MAPE) and ***bias***. In the industry, the MAPE metric is prevalent; however, we avoid it due to its peculiar properties around values close to zero (which we are dealing with often in this research). Denoting the performance of meter i in month m by $z_{i,m}$, denoting m_0 as the first month in the future, denoting N as the total number of meters and denoting M as the forecast horizon. The goal is to forecast $\left[z_{i,m_0}, z_{i,m_1}, \ldots, z_{i,M}\right]$ using performance values and exogenous variables for which $m \in [1, \ldots, m_0 - 1]$. The forecasted performance values are denoted by $\hat{z}_{i,m}$. The point forecast metrics are defined as:

$$\text{bias} = \frac{\sum_{i,m}\left(\hat{z}_{i,m} - z_{i,m}\right)}{N(M - m_0)}$$

$$\text{MAE} = \frac{\sum_{i,m}\left|\hat{z}_{i,m} - z_{i,m}\right|}{N(M - m_0)}$$

$$\text{RMSE} = \sqrt{\frac{\sum_{i,m}\left(\hat{z}_{i,m} - z_{i,m}\right)^2}{N(M - m_0)}}$$

The RMSE metric is closely related to the variance of the error; it penalizes big mistakes relatively more than small mistakes. The MAE metric is less widely used but is useful when dealing with outliers; it does not penalize big mistakes relatively more than small mistakes.

Probabilistic Forecast Metric

We compare the probabilistic forecasts of the base model and advanced model using probabilistic forecast metrics. Just like in related work from Chapter 3, we compare the models using the ρ-risk. With $\rho \in (0,1)$ we have $\hat{z}^{\rho}_{i,m}$ as the ρ-quantile prediction. The original ρ-risk uses a sliding window over the forecasts. As our targets and thus forecasts available for analysis have many gaps, it is not feasible to use the sliding window approach. Instead, use a simplified version of the ρ-risk that looks at each prediction month individually before aggregating them.[4] The ρ-risk is defined using the ***Quantile Loss*** (QL)[5], it can be interpreted as the expected relative QL.

$$\rho\text{-risk} = \frac{2\sum_{i,m} QL(z_{i,m}, \hat{z}^{\rho}_{i,m}, \rho)}{\sum_{i,m} z_{i,m}}$$

2.4.4 Model types

Forecasting of time series can be achieved using ***Machine Learning*** (ML) models. There are roughly five types of models that are popular for forecasting in the energy domain. Four of these are shortly covered in the following sections; they are viable candidates to solve our problem. The fifth model type is the bottom-up approach, which involves modelling time series for *all* the individual devices present behind a meter (Fan, MacGill, & Sproul, 2015). However, since we do not have this information, this option is ruled out.

Time series analysis

The most widely used time series analysis models used for forecasting time series are the ***Autoregressive Moving Average*** (ARMA), ***Autoregressive Integrated Moving Average*** (ARIMA) and ***Seasonal ARIMA*** (SARIMA) models, introduced by Box, Jenkins, and Reinsel (2008), of which the first two as early as 1970. These models all consist of a ***Moving Average*** (MA) part, which models the time series as a linear regression of white noise, and an ***Autoregressive*** (AR) part which models the

[4] In related work this is referred to as the $ALL(k)$ ρ-risk

[5] The quantile loss is explained and discussed in more detail in Section 2.5.1

time series as a linear regression of the past values. These models are prevalent among the traditional tools for time series analysis because of their ability to extract the statistical properties of the time series (Das et al., 2018). Unfortunately, the models are not well suited for long-term prediction, and optimization is computationally expensive (Zhai, 2009). We have many time series, for which we need relatively long-term forecasts, so these models are not suitable for our problem.

Regression

Linear regression method is a statistical method used to find the best fitting linear line to describe the relationship between independent and dependent variables. One variant is the multiple linear regression model, which enables a linear regression on multiple independent variables. This method is prevalent in LTF due to its simplicity and accuracy in this time frame (Kuster et al., 2017). Each addition of an independent variables increases the risk of overfitting. Therefore feature selection is important. On the other hand, these models fit linear functions, limiting their overfitting potential but also limiting learning capacity.

Support vector machines

One of the most popular ML models are the ***Support Vector Machines*** (SVMs) developed by Cortes and Vapnik (1995). ***Support Vector Regression*** (SVR) uses SVMs to fit a function based on independent variables and can thus be used for time series forecasting. A line is estimated that minimizes a regularized loss function, and using the "kernel trick", these models can fit non-linear functions. The methodology optimizes a convex problem, which ensures that a unique solution is delivered (Mamun et al., 2020). A disadvantage of SVMs is the training complexity, which is on average $O(n^2)$ Bottou and Lin (2007).

Artificial neural networks

Kuster et al. (2017) describe ***Artificial Neural Network*** (ANN) as one of the more popular techniques for time series forecasting. Although ANNs have been around for some time, recently their popularity has spiked (Hsiao, 2015) due to advances which made the training of deep networks, those with at least two hidden layers, effective. In the last few years, ANN and ***Deep Learning*** (DL) have also seen a heavily increased interest in energy forecasting. More recent work by Mamun et al. (2020) show this increased interest of ANN next to the popularity of SVM.

ANNs are models in which one or more layers of linear transformations from the input to the output space are applied sequentially. These models are trained using ***Back Propagation*** (BP) in which the errors of the output are propagated through the layers, back to front, and used to update the weights that are part of the linear transformations. The universal approximation theorem (Z. Lu, Pu, Wang, Hu, & Wang, 2017) for ANN tells us that given sufficient neurons, an ANN with at least one hidden layer can approximate any continuous function.

Advantages are the reduced dependency on the practitioner to model complicated linear relationships and the great flexibility (Hernández, Baladrón, Aguiar, Carro, et al., 2014), which enables researchers to come up with very creative and effective network architectures, an overview of the possibilities is given by Wang, Lei, Zhang, Zhou, and Peng (2019). An example of a recently popular architecture is the ***Deep Belief Network*** (DBN), which works well in datasets with sparse labels, which is a problem we encounter. A disadvantage is the large potential for overfitting as the models can have an arbitrary capacity. Another disadvantage is that the optimization of the architectures takes a long time as there are virtually infinite ways to create an architecture.

2.4.5 Selected models

As explained in Section 2.2.2, this research is conducted using two separate approaches. The first approach produces the base model. The second approach produces the advanced model.

Base Model

As requested, the initial base model was created in a short time frame. To train the model, several features are extracted from the data, as explained in Section 4.5. For the base model, we use

the NGBoost model as developed by (Duan et al., 2020). NGBoost is a regression model that can provide predictions with advanced confidence intervals.

We cover this decision in more detail in Section 5.2. Gradient boosted decision trees are prone to overfitting, and the amount of features that can be extracted is virtually unlimited. Therefore, we apply feature selection in the form of a binary genetic algorithm. Binary genetic algorithms show good results in feature selection (Babatunde, Armstrong, Leng, & Diepeveen, 2014; Eseye & Lehtonen, 2020) and are distributive in nature, enabling quick model training.

Advanced Model

ANNs are very popular in the energy forecasting domain. The flexibility of designing the architecture is promising to handle the sparse and related time series. This flexibility is used to design an algorithm tailored to the problems and complications encountered in this research.

> "It is generally recognized that deep learning-based forecasting models exhibit attractive performance in terms of accuracy, stability and effectiveness, which are beneficial to energy system planning, scheduling and management." — Wang et al. (2019)

The next section covers some foundations for the state-of-the-art time series forecasting models. In Chapter 3, we cover architectures that aim to solve problems similar to those in this research. The methodology for the design and testing of the model is described in Chapter 6.

2.5 Deep learning

DL has gained increased interest in many domains over the last few years. These days there are many DL models for the general and energy-specific time series forecasting domain. This section provides the foundation for some of the state of the art DL models for forecasting.

2.5.1 Loss

When optimizing a DL architecture, one of the steps is selecting an appropriate loss function. There are two standard loss functions for regression problems: the mean squared error loss and the mean absolute error loss. With targets $\mathbf{y}$ and predictions $\hat{\mathbf{y}}$ the mean squared error loss is defined as:

$$loss(\mathbf{y}, \hat{\mathbf{y}}) = (\mathbf{y} - \hat{\mathbf{y}})^2$$

The mean squared error loss is the most commonly used loss for regression problems. It punishes large errors more heavily than smaller errors.

The mean absolute error loss is the next most used loss function for regression. With targets $\mathbf{y}$ and predictions $\hat{\mathbf{y}}$ the mean absolute error loss is defined as:

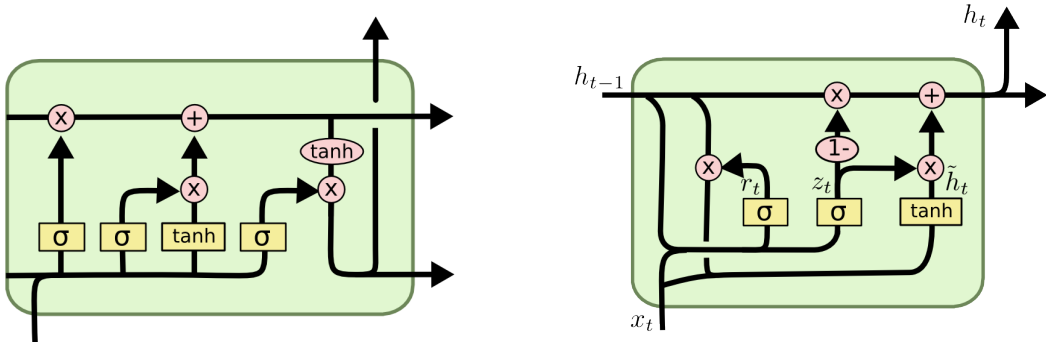
$$loss(\mathbf{y}, \hat{\mathbf{y}}) = |\mathbf{y} - \hat{\mathbf{y}}|$$

The mean absolute error loss does not punish larger errors more than smaller errors. As a result, this loss is often used in cases where there the distribution of the target variable has outliers or big values to prevent overfitting on those values.

Many real-world applications require more than just a single point prediction. Often it is beneficial to have information on the expected distribution of the target variable. In those cases, we can choose to predict multiple quantiles and are dealing with quantile regression. For quantile regression, we use the quantile loss function. The quantile loss function, as defined by Lim, Arik, Loeff, and Pfister (2020), is given using observed value $\mathbf{y}$, predictions $\hat{\mathbf{y}}^\rho$ for quantile ρ, and $(\cdot)_+ = \max(0, \cdot)$ as:

$$loss(\mathbf{y}, \hat{\mathbf{y}}^\rho, \rho) = QL(\mathbf{y}, \hat{\mathbf{y}}^\rho, \rho) = \rho(\mathbf{y} - \hat{\mathbf{y}}^\rho)_+ + (1 - \rho)(\hat{\mathbf{y}}^\rho - \mathbf{y})_+$$

The quantile loss penalises under and over prediction depending on the value of ρ. There is a strong relationship between the quantile loss and the mean absolute error loss. If we want to optimize for the value $\rho = 0.5$, we optimise for the median. Substitution of that value into the quantile



(A) A schematic overview of an LSTM cell.

(B) A schematic overview of a GRU cell.

FIGURE 2.5: Schematic overview of the two most popular RNN specializations. (Olah, 2015)

loss results in the mean absolute error loss. The intuition behind the quantile loss, with the 0.75 quantile as an example, is the following: For any set of elements A and the 0.75 quantile of this set as b, there are three times more elements in A smaller than b than there are elements in A larger than b.

$$|\{a \leq b | a \in A\}| \approx 3 \cdot |\{a > b | a \in A\}|$$

In the quantile loss, using the 0.75 quantile, a quantile prediction that is smaller than the observed value is penalized three times as hard as a quantile prediction that is larger than the observed value. Thus, the quantile loss tries to force three times more values below the quantile prediction as above the quantile prediction.

We can design an architecture with multiple outputs in quantile regression, each output optimizing for a different quantile. The quantile loss for each output can then be combined through addition to get a multi-quantile loss. Now we can optimize the architecture to provide quantile predictions.

2.5.2 Dropout

When developing ML models, it is important to be wary of overfitting. Overfitting is the phenomenon in which a model might perform very well on the training data but fails to generalize properly to real-world scenarios. To combat overfitting in DL, many regularization techniques limit the amount of overfitting. One such technique is called dropout.

Dropout is parameterized by a value $p \in (0,1)$. During training, each weight in the DL architecture is excluded with a probability p. After training is done, all weights are scaled by multiplying with the dropout probability. This technique has a regularizing effect on the model and therefore limits overfitting. Dropout can be interpreted as training many models that are all slightly different and, in the end, taking the average of them all.

2.5.3 LSTM & GRU

Traditional ANN models receive a static length input and a static length output. There are many examples for which this constraint is undesirable; one such example is time series forecasting. Some time series might have years worth of historical data, while others only have a few months available. Additionally, the order of the datapoints is essential in forecasting.

A specialized version of ANNs are the ***Recurrent Neural Networks*** (RNNs). RNNs receive ordered datapoints one by one and maintain a hidden state when processing the datapoints. The problem is that RNNs suffer from vanishing gradients and exploding gradients. For each new data point, the hidden state is multiplied by a weight. Thus, at the end of the sequence, the influence of the first data point has decayed exponentially. Therefore, RNNs have limited memory and are unsuitable for longer sequences.

Long Short-Term Memory (LSTM) are specializations of RNNs designed by Hochreiter and Schmidhuber (1997) to tackle the vanishing gradient problem. This specialization adds a cell state for long term memory. The default RNN state is referred to as the output state in an LSTM. The

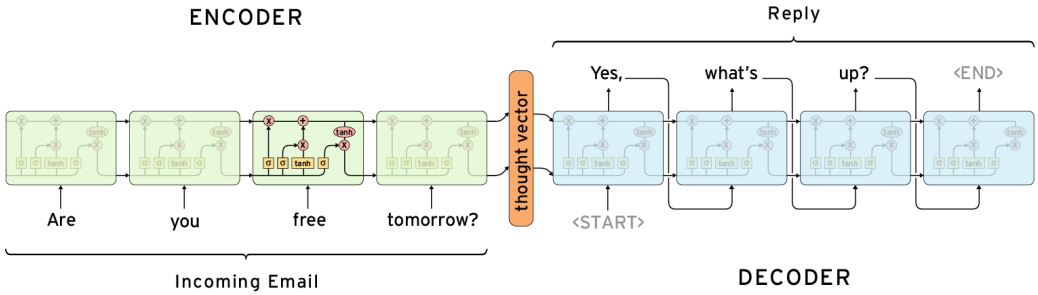


FIGURE 2.6: A schematic overview of the sequence to sequence architecture using LSTM models for the encoder and decoders. (Abeywardana, 2017)

flow of information through these states can be seen as the top horizontal arrow and the bottom horizontal arrow in Figure 2.5a for the cell state and output state. The internals of an LSTM cell comprises a forget gate, an input gate and an output gate. The forget gate decides what information to retain and what information to forget from the cell state; this is shown on the schematic's left side. The input gate decides what new information to absorb into the cell state; this is shown in the schematic's middle. The output gate decides what information to send as the output; this is shown on the schematic's right side. The new input, previous output state, and cell state are combined using these gates to create the new output state and cell state at each step. The separate cell state acts as a long term memory. This long-term memory prevents the explosion of gradients and therefore stabilizes training/backpropagation.

The ***Gated Recurrent Unit*** (GRU) is an adaptation of the LSTM (Cho et al., 2014). A schematic overview of the GRU cell design is provided in Figure 2.5b. Instead of keeping a separate cell state, the GRU only has one state, the output state. The update and forget gate are combined into a single update gate which decides which information to keep, shown on the right side of the schematic. Additionally, the cell has a reset gate, deciding which information and how much information to forget, shown on the left side of the schematic.

Due to the simpler implementation of GRUs, they are a bit faster. (S. Yang, Yu, & Zhou, 2020) Theoretically, LSTMs should be able to handle longer sequences. In practice, the best choice is often problem-dependent. Therefore practitioners usually try both and then pick the best one.

2.5.4 Sequence to Sequence

Many tasks in the ***Natural Language Processing*** (NLP) domain require differently sized input and output sequences. These issues gave rise to the ***Sequence to Sequence*** (Seq2Seq) models, which provided good results and allowed input and output sequences of different sizes. The use of these architectures has since then spread to other domains, like time series forecasting. The two papers most relevant to this research make use of Seq2Seq models.

Sutskever, Vinyals, and Le (2014) presented an encoder-decoder framework for Seq2Seq learning in the NLP domain. The input sequence is fed into an encoder network which compresses the input sequence into a single large embedding vector. The decoder network is then initialized with the embeddings in the final state of the encoder network. To generate the output sequence, the decoder network produces the first output, then that output is fed back as an input into the decoder network. This process is repeated until a special end of sequence token is returned. A schematic overview of this process is provided in Figure 2.6.

The field of NLP is moving very fast due to the tremendous amount of research being put into this by academia and industry alike. As a result, Seq2Seq models have become outdated in the NLP field already. The current state of the art models in the NLP space are based on attention and so-called transformers. Although the Seq2Seq models still play a prominent role in the time series forecasting field, more influences from the NLP domain are flowing into this field.

2.5.5 Attention

Seq2Seq had one major drawback in the NLP domain, information bottleneck. There are large distances between the encoder and decoder values, and all information has to be passed through a

single embedding vector. The (self-)attention mechanisms were introduced to relieve these problems. In an attention layer, the input is multiplied with three separate weight matrices to obtain three matrices: keys $\mathbb{K}$, queries $\mathbb{Q}$ and values $\mathbb{V}$. (Lim et al., 2020, p.9) Attention, using d_{attn} as an architecture dependent scaling factor, is defined using these matrices as:

$$\text{Attention}(\mathbb{Q},\mathbb{K},\mathbb{V}) = \text{Softmax}(\frac{\mathbb{Q}\mathbb{K}^T}{\sqrt{d_{attn}}})\mathbb{V}$$

The softmax activation forces all outputs to sum up to 1. This is where the name attention comes from. Using the keys $\mathbb{K}$ and queries $\mathbb{Q}$, we determine on which values $\mathbb{V}$ we should focus our attention. Attention also improves the interpretability of a model, as we can see for each sample what the model pays attention to. To allow a model to focus on multiple different things simultaneously, the so-called multi-head attention was introduced. While the attention mechanism is at the core of the recent developments in the NLP domain, the architectures dominating the space are called the transformers. Transformers use attention layers combined with more traditional DL techniques to create a separate encoder and decoder architecture.(Vaswani et al., 2017)

Chapter 3

Related Work

As described in Chapter 2, there exists much literature on time series forecasting in the energy domain. Table 2.1 shows how the research goals of some of these studies vary from this research. There are also major differences in the disposable approaches when considering the requirements and available resources. Any suitable approach to designing a forecasting algorithm in this research takes the following considerations and data characteristics into account:

- A large number of related time series
- The scale of the values varies massively between time series
- Sparsity of the performance data in the time series
- Cold-starts due to the absence of the performance data
- Incorporation of datasets with different resolutions

Related work that tries to tackle every one of these considerations could not be found in the energy domain. As a result, the search area for related work was expanded from the energy forecasting domain to the general forecasting domain.

General Forecasting

The "many related time series" consideration and the cold-start problem often go hand in hand. Xie, Tank, Greaves-Tunnell, and Fox (2018) present a matrix factorization approach to handling related time series. In their approach, they also tackle the cold-start consideration. The disadvantage of their approach is that they make assumptions on the profile of the time series. Preferable, the model learns this autonomously instead of relying on a manual effort. Jha, Ray, Seaman, and Dhillon (2015) focus their efforts on handling the cold-start considerations. In doing so, they rely on information from related time series.

(Che, Purushotham, Cho, Sontag, & Liu, 2018) try to tackle the problem of sparse time series. A key contribution is "informative missingness", which states that missing values and their missing patterns are often correlated with the target variable. This effect is present in our data due to the influence of the weather on both the dispatch probability and the performance of some meters. They leverage the informative missingness by creating a modified version of a GRU that accepts a mask denoting data availability. Such a mask can potentially be retrieved from our data for the hourly resolution or calculated for the monthly resolution.

(Shen, Zhang, Xia, Liu, & Li, 2019) create a multi-resolution prediction model in the energy domain. They predict the electricity consumption in the next second and the average over the next 30 seconds for each second. The differences with our research are in focus on multi-resolution inputs instead of output.

Deep learning has gained increased interest in general forecasting (Lim & Zohren, 2021) due to notable achievements in many domains like in the energy domain. We decided to design an ANN architecture by combining knowledge from the energy forecasting domain with related work from the general forecasting domain. The remaining sections of this chapter cover the papers that, according to the author, show the most potential for this research. The following papers form the basis of the model that will be developed in this research.

3.1 DeepAR

Salinas, Flunkert, and Gasthaus (2019) present a methodology for producing accurate probabilistic forecasts. The methodology, named DeepAR, involves training an auto-regressive recurrent network on a large number of related time series. This approach overcomes many of the challenges encountered in the classical forecasting approaches. A 15% improvement compared to state-of-the-art methods on real-world datasets is observed; One of which involves the forecasting of electricity consumption of individual households.

Model

In DeepAR, a Seq2Seq model is trained in which the encoder and decoder network are identical. The model is autoregressive as well as recurrent; This means that both the previous output of the model and the previous observation is fed back into the model. $\mathbf{h}_{i,t}$ denotes the output of the model at time step t, h denotes the function that represents the model with parameters Θ and $\mathbf{x}_{i,t}$ denotes the covariates. Then the network is given by:

$$\mathbf{h}_{i,t} = h(\mathbf{h}_{i,t-1}, z_{i,t-1}, \mathbf{x}_{i,t}, \Theta)$$

The likelihood that the model produces is a chosen distribution that is parameterized by a function $\theta(\mathbf{h}_{i,t}, \Theta)$. During encoding, the observed values $z_{i,t}$ are passed into the network. During decoding these values will be sampled such that $\hat{z}_{i,t} \sim \theta(\hat{\mathbf{h}}_{i,t}, \Theta)$, where $\hat{\mathbf{h}}_{i,t} = h(\hat{\mathbf{h}}_{i,t-1}, \hat{z}_{i,t-1}, \mathbf{x}_{i,t}, \Theta)$.

The result is that the encoder creates an embedding vector of the time series based on the observations; The decoder generates sample paths of the expected future observations based on this embedding.

Related time series & cold-starts

DeepAR was designed in a setting of forecasting weekly sales of many products. Each of those product sales represents a time series of historical sales. The authors realized that within this dataset, there must be many products that have similar time series. Training a single model on all the time series instead of a separate model for each time series enables fitting more complex models without overfitting. By training on the related time series, the model can learn cross-correlations and deal with cold-start time series.

Scale of the time series

A common practice in DL is to constraint the size of the weights; This reduces the overfitting potential of neural networks. This limitation makes it difficult to learn from related time series of vastly different scales properly. DeepAR scales each time series with a time series dependent scaling factor v_i at the input and reverses this in the output. This enables the model to learn from related time series while simultaneously handling large scale time series.

Missing values

In the supplementary materials Salinas et al. (2019) mention their approach to dealing with missing values. Similar to the decoder process, the missing values $z_{i,t}$ are replaced by a sample $\hat{z}_{i,t} \sim l(\cdot|\theta(\mathbf{h}_{i,t}))$. Although this approach worked well in their research, they did not conduct a proper evaluation. Additionally, the data in this research has relatively massive gaps in almost all time series instead of a few missing values.

Relation to this research

DeepAR is designed to learn from many related time series; With the right dataset preparation, it can handle cold-starts; Using scaling it can handle the large differences in scale of different time series; Finally, it can handle missing values, although this has not been tested thoroughly. This means that DeepAR focuses on solving at least three out of the five main considerations in this research. Additionally, it provides a potential solution for a fourth one. The main difference with

this research is that DeepAR has an identical encoder and decoder architecture. This research will use an encoder trained on a high resolution, while the decoder will forecast on a low resolution.

3.2 MQ-RNN

Wen, Torkkola, Narayanaswamy, and Madeka (2018) propose a framework, named MQ-RNN, for general probabilistic multi-step time series regression. The proposed algorithm trains a Seq2Seq model with an encoder network that is different from the decoder network. Encoding involves an RNN that creates a vector embedding of the time series. Decoding involves a ***Multilayer Perceptron*** (MLP) which extracts a context for each step in the forecast from the vector embedding; Followed by another MLP, which is applied to each step in the forecast to generate forecasted quantiles. This approach prevents error accumulation which AR models like DeepAR suffer from.

Model

The practitioner defines a set of required quantiles that the model will produce for each forecast time step. The encoding network in MQ-RNN is a vanilla LSTM network. However, instead of using an LSTM decoder, two branches of MLP are used. The first (global) MLP extracts a horizon-agnostic context c_a and a horizon-specific context c_{t+k} from the encoder output. The second (local) MLP combines the contexts and covariates into the requested quantile outputs for each forecast time step. The model is optimized by minimizing the quantile loss as described in Section 2.4.3.

Related time series & cold-starts

Just like DeepAR, MQ-RNN is designed to forecast in the presence of many related time series. A single model is trained on all the time series, enabling the learning of related patterns. Wen et al. (2018) demonstrate how this also leads to proper forecasting of cold-start time series.

Scale of the time series

The paper accompanying MQ-RNN does not explicitly mention that the model can handle time series of vastly different scales. However, the model is evaluated on a dataset in which this effect is present. As they report good performance on this dataset, likely, the model can properly handle this type of data.

Relation to this research

MQ-RNN is designed to learn from many related time series and performs well in cold-start situations. The paper does not mention anything about handling missing values, which is important for our research. This research shows the potential of using different encoder and decoder architectures in the forecasting domain. The separation between encoder and decoder eases the multi-granularity application proposed in the current research.

3.3 Temporal Fusion Transformer

Lim et al. (2020) propose an architecture for interpretable multi-horizon time series forecasting called the ***Temporal Fusion Transformer*** (TFT). The TFT borrows state-of-the-art concepts from the NLP domain and insightfully utilizes them to provide an interpretable deep learning architecture. This architecture achieves state-of-the-art performance on forecasting problems, outperforming DeepAR by 30%.

Autoregressive versus multi-horizon forecasting

Like MQ-RNN, the authors of the TFT point out the problems of autoregressive forecasting, namely the accumulation of errors. Therefore, just like the MQ-RNN, the TFT utilizes an encoder-decoder architecture in which the encoder has a different architecture than the decoder and a fixed forecast

horizon. Additionally, it is easier to use different granularities for encoding and decoding in a multi-horizon forecasting architecture.

Missing values

DeepAR has an elegant way of dealing with missing values. TFT cannot deal with missing values in such a way and therefore relies on imputation instead in a similar scenario. The low granularity data does not contain any missing values. Therefore, this issue is mostly eliminated if we can create a multi-granularity approach.

Features

Most architectures for forecasting accept three kinds of inputs: static covariates, target variable (values available in the history, but unknown in the future) and exogenous variables such as time encodings (values available in the past and the future). TFT also allows for multiple exogenous variables for which the values are only available in the history. This is excellent for the problem in the current research as the historical load, historical baselines and historical dispatches are available. All of these are very related to the performance of a meter but are unknown in the future. TFT might be able to extract the relevant relations and patterns between these variables to predict future performance.

Attention

Attention is a complicated but powerful concept used in most state-of-the-art architectures in the NLP domain. As explained in Section 2.5.5, the attention mechanism forces the model to "pay attention" to some inputs while partially ignoring other inputs. As a result, attention is also praised for improving the explainability of deep learning architectures. For each prediction, it will be possible to extract what the model paid attention to in an interpretable way. This explainability could be very useful in convincing colleagues at Leap to trust and then use this model.

Data availability

The most significant problem for this project, which existed for DeepAR and still exists for TFT, is the data availability. Table 1 in Lim et al. (2020) lists the datasets and their sizes. In our dataset, the number of entities is about 10k, and the number of intervals is about 300 million. However, due to the multi-granularity and rare dispatches of meters, we can only generate about 50k samples from this data, almost half of the smallest dataset in the TFT paper. Additionally, Leap has more limited supporting infrastructure to run these experiments.

3.4 Selected architecture

To decide on which architecture to base the advanced model of this research on, we look at the five identified considerations from the beginning of this chapter. The best scoring recently published ML models in the forecasting domain are frequently ANNs focusing on modelling many related time series. Making the architecture choice agnostic to the first consideration. Salinas et al. (2019) show that the model can handle the scale of the time series by scaling the target variable with a scalar characteristic of the time series. Although the other papers do not explicitly mention it, this same technique can be applied to all the other architectures. In our scenario, we can solve the last three considerations if we implement a proper multi-granularity approach. The TFT architecture is likely the best candidate to use in a multi-granularity setting; additionally, it has shown to perform at the current state-of-the-art level. Therefore, the TFT will form the basis for the advanced model in this research.

Chapter 4

Data

In order to improve the meter-level nominations, there are several data sources that we have access to. An overview is provided in Section 4.1. Depending on the source, the data can be very messy. Therefore, quite some extensive data preprocessing, data analysis and feature engineering are necessary. An exact definition of accurate nominations is described in Section 4.2. In Section 4.4 the initial data exploration and analysis is described, followed by the selection of features for the base model described in Section 4.5.

4.1 Data sources

Leap is a young company that is at an early stage of designing and building its data infrastructure. Therefore the source, the format and the quality of the data vary widely across the board. In order to use the data, we created several small scale data pipelines to extract, transform and load the data. In this research, there are mainly three data sources that result in several datasets. [1]

Meter Baseline Pipeline

As part of settling revenue with the partners, Leap calculates so-called baselines (more details on these in Section 4.2.1). The output of the pipeline that calculates these baselines is the main data source for this research. This pipeline must comply with many data quality requirements; this is our most reliable and complete data source. For each meter, this dataset contains time series of 15-minute interval data. The contents of the records in this dataset are described in Table 4.1. An example of one of these time series is given in Figure 2.3. The dataset contains about **10.000**[2] **meters and about 300 million records**. The oldest records originate from the start of 2019. However, the amount of meters has grown significantly, so not all meters have records that old. We extract the meter load, the meter baseline, the historical dispatches, and the historical performance from this data source. The definition of performance will be clarified in Section 4.2.

Meter Metadata

The second data source is the meter metadata file. This file contains static data that describes meters and their origin. All relevant types of metadata are listed in Table 4.2. This file is created and appended to by manual processes. As a result, there are no completeness guarantees, and

[1] The overviews of data sources given in this section are oversimplified and often do not originate from single sources but rather are aggregations of multiple sources.

[2] Leap's platform is growing fast, during the data analysis, there are about 10.000 meters on the platform. By the end of this research, that number has more than doubled.

Column	Data Type	Description
Meter ID	uuid	A unique identifier denoting a meter.
Interval Start	datetime	The start time of this 15-minute interval.
Dispatch Interval	boolean	Whether the meter got dispatched in this interval.
Load (Wh)	decimal	The energy consumption during this interval.
Baseline (Wh)	decimal	The baseline energy consumption for this interval.

TABLE 4.1: The most relevant fields in the meter baselines data source.

Column	Data Type	Description
Meter ID	uuid	A unique identifier denoting a meter.
Meter Type	categorical	Residential or commercial meter type.
Partner	categorical	The partner through which the meter joined the DEX.
Load Type	categorical	The type of controllable device that is behind the meter.

TABLE 4.2: The most relevant fields in the meter metadata source.

Column	Data Type	Description
Meter ID	uuid	A unique identifier denoting a meter.
Year Month	integer	The year and month combination associated with the nomination.
Nomination (kW)	decimal	The available capacity in kW expected to be delivered by this meter.

TABLE 4.3: The most relevant fields in the meter authorizations source.

the data requires cleaning. Still, it is the most reliable source for this information, which is very valuable, as shown in the data analysis section.

Meter Authorizations

The MarketOps and Partner Success departments collaborate on a so-called "meter authorization" spreadsheet in a manual effort. This spreadsheet contains a wide range of information, but the most important information for this research is the meter nominations made in 2020. A short description of the most valuable information is given in Table 4.3. These historical nominations are used as a baseline method to compare against.

4.2 Labelling

The data source "meter baseline pipeline" contains lots of data useful for this research. The first step is to extract the labels from this data source. ***Generation*** is the baseline minus the load during dispatches. The ***performance*** is the maximum two-hour consecutive generation in an entire month. The performance of meter i in month m is denoted by $z_{i,m}$. ω denotes a 15-minute long interval in m. $x_{i,\omega}$ denotes the load in kilowatt and $\hat{x}_{i,\omega}$ denotes the baseline in kilowatt for meter i during interval ω in month m. Ω denotes the set of all intervals in month m. $d_{i,\omega}$ denotes a dispatch such that $d_{i,\omega} = 1$ if the meter i was dispatched during interval ω and $d_{i,\omega} = 0$ otherwise. Using this notation the monthly performance is defined as:

$$\psi_i(\omega) = \begin{cases} \text{undefined} & \text{for } \sum_{\gamma=0}^{7} d_{i,\omega-\gamma} < 8) \\ \frac{\sum_{\gamma=0}^{7}(\hat{x}_{i,\omega-\gamma} - x_{i,\omega-\gamma})}{8} & \text{otherwise} \end{cases}$$

$$z_{i,m} = \max_{\omega \in \Omega} \psi_i(\omega)$$

Leap uses a minimal value of 0 for the monthly performance. We do not cap the maximum performance from below at zero in our labelling process and the above definition. Leaving the performance values uncapped prevents the introduction of an unwanted non-linearity in the labels. Any stakeholders are informed of this behaviour and are free to apply the cap at zero on the forecasted nominations where they see fit.

4.2.1 Baselines

The generation in demand response is equal to the amount of reduced load. Although the load (kW) of a meter can be measured, we cannot measure what the load would have been had the meter not been dispatched. Therefore, the CAISO developed and approved several 'baseline' methodologies to calculate baseline values (kW) to approximate these values. These values are then used to determine generation or performance as described above. In the following paragraphs, we sketch a rough summary of the two baselines mainly used by Leap.

Commercial Baseline

The calculation of the commercial baseline makes a clear distinction between business days and non-business days. To determine the baseline value for an interval on a business day, we calculate the average of the load over the **past 10 business days** at the same time in the day. The same is done for non-business days. However, we only look back to the **past 4 non-business days**. Days on which the meter was dispatched are excluded from these calculations. Instead, we look back until we find the right number of days.

Residential Baseline

The calculation of the residential baseline is similar. Instead of using the average of the selected historical days, we take a weighted average applied to the ordered loads. The weighted average puts more weight on the highest loads and less weight on the lower loads.

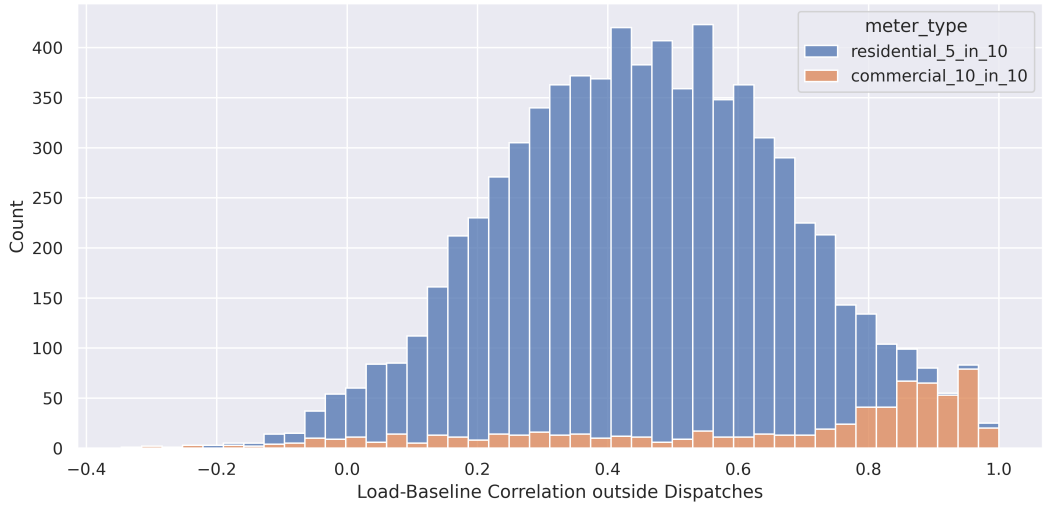


FIGURE 4.1: The stacked distribution of the correlation between the load and the baseline for intervals that were not part of a dispatch. For most commercial meters, the moving average model works well. However, there is a long tail; some meters even have a negative correlation. For residential meters, the baseline seems to correlate not as good.

Meter Type	Year-Month	# Dispatched Meters	MBE [kW]	MAE [kW]	RMSE [kW]
Commercial	2020-07	671	30.77	63.82	146.56
	2020-08	682	2.34	47.04	128.00
	2020-09	549	-3.51	40.22	216.83
	2020-10	695	26.12	46.28	102.86
	2020-11	154	39.16	89.78	167.50
	2020-12	187	19.00	79.81	151.60
Residential	2020-07	7066	-0.54	1.10	1.53
	2020-08	6764	-1.00	1.35	1.87
	2020-09	6670	-0.54	1.13	1.54
	2020-10	7068	-0.35	0.91	1.26
	2020-12	588	0.04	0.53	0.78

TABLE 4.4: The performance metrics of the nominations made in 2020 using the manual process.

Complications

This baseline calculation introduces several complications. The days on which DR is activated are usually different from the normal days; it is extra warm or extra cold. Many of the meters on Leap's platform are influenced by these same variables, such as air-conditioning devices. As a result, the baseline potentially underestimates the load of these devices. Additionally, **dispatching** of DR resources is **likely to happen on consecutive days**. In practice, this can lead to weeks of daily but short dispatches. All of these days have to be discarded for the baseline calculations. As a result, baseline calculations are sometimes based on days more than a month in the past.

These baseline methodologies can work well for meters that are well described using a ***Moving Average*** (MA) process. However, as can be seen in Figure 4.1, this is not always the case. We have no control over this part of the process. Therefore, the baseline will be treated as an exogenous variable. Still, we are aware of these data characteristics during the design and optimization of the models.

4.3 Historical Nominations

The historical, manual meter nominations are stored and are an important data source for benchmarking the forecasts. These nominations are made in several ways, either by manually specifying them in collaboration with the partners or using simple decision rules. As described in Section 2.4.3, several accuracy metrics are calculated for these nominations to create insight into the current process. We define the error as $\hat{z}_{i,m} - z_{i,m}$, where $\hat{z}$ denotes the nomination, z denotes the observed performance of meter i in month m. The performance metrics are displayed in Table 4.4.

The first observation we make is regarding the scale of the errors, which is unsurprising as the meters can be very unpredictable, as seen later in Section 4.4. The second observation is the bias in the residential and commercial meter nominations. The manual process has a negative bias for residential meter nominations, but has a postive bias for commercial meter nominations.

4.4 Data analysis

In order to get a better understanding of the data, an extensive data analysis was performed. The analysis was used as a guide in defining the project's scope and creating a prospective methodology.

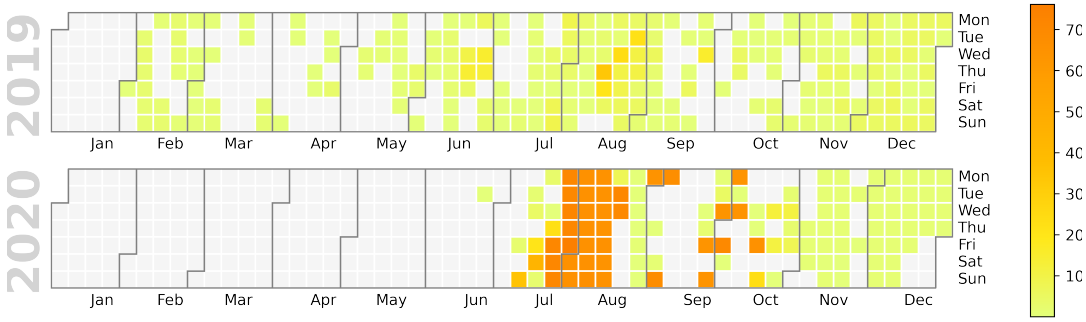
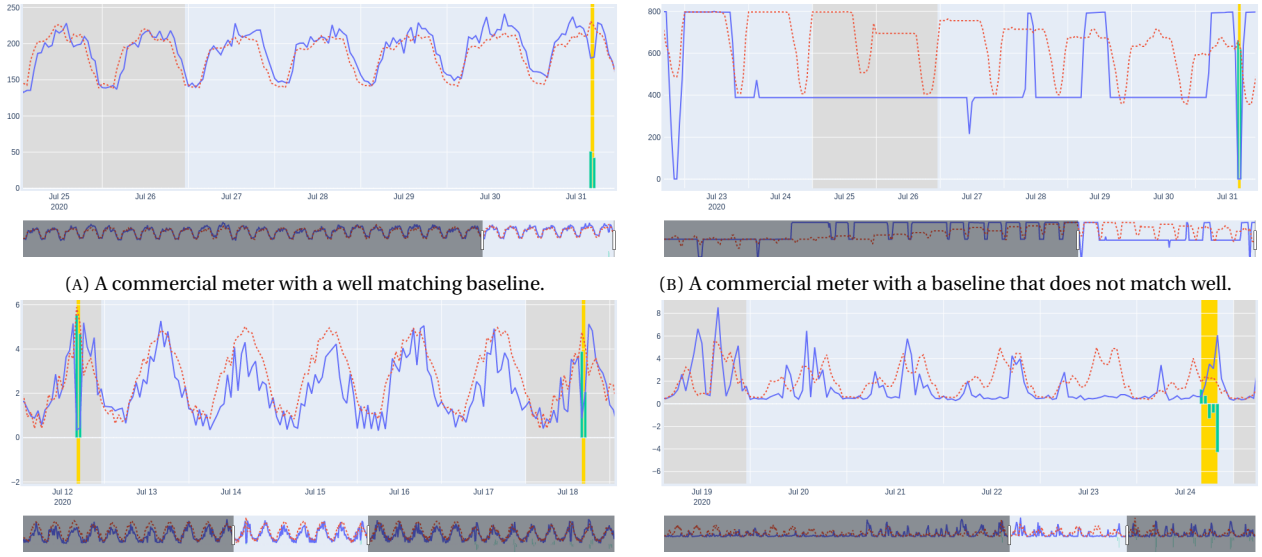


FIGURE 4.2: The number of meters that got dispatched on a certain day. The number of meters has grown over time, and due to changes in the business processes, there were no dispatches for the first half of 2020.

4.4.1 Overview

Some of the meters in the meter baselines dataset have not been assigned to resources or have not been dispatched. There are about 9.000 meters that have ever been dispatched and have therefore demonstrated performance. On 9.66% of the days, a meter receives at least one dispatch and

is expected to curtail its energy consumption. The months July, August and September constitute **Leap's operational season**. Therefore, most of the data analysis has been focused on these months. The importance of these months is highlighted in Figure 4.2. In 2019 there were fewer meters on the platform, which explains why the seasonality is harder to see in the figure. Something that stands out is the complete lack of dispatches between January and June 2020. This lack of dispatches was confirmed in conversations with stakeholders. However, no formal motivation for this gap was provided. The key takeaway once again is that the time series of our performance data is very sparse. Additionally, due to the rapid growth, there are often meters on the platform that have never been dispatched, which would result in "cold-start" forecasting.



(A) A commercial meter with a well matching baseline.

(B) A commercial meter with a baseline that does not match well.

(C) A residential meter with a well matching baseline.

(D) A residential meter with a baseline that does not match well.

FIGURE 4.3: A few time series with an hourly resolution of individual meters with different characteristics. The blue line represents the load, and the dashed red line shows the baseline. Non-business days are shaded in grey, and the dispatch hours have a yellow background. Using the green bars, the generation is displayed.

4.4.2 Meters by Metadata

In order to get a better feeling of the wide range of meters in the dataset, we created a dashboard for interactive visualization. The meter load, baseline and characteristics vary widely across the board. Figure 4.3 shows four handpicked meter patterns, each of which shows different load patterns that are regularly encountered in our dataset. As stated before, some meters are well described using a ***Moving Average*** (MA) process. An example is the meter in Figure 4.3a. For other meters like in Figure 4.3b, it does not work at all. The responsiveness to dispatch signals also varies widely across the board. Some meters clearly respond to dispatch signals, like the meter in Figure 4.3c. For other meters, it is difficult to say if there is even any response to the dispatch signal, like in Figure 4.3d.

Statistical meter characteristics were calculated for each meter for two main reasons. Firstly, to quickly explore the data. Secondly, to potentially use them as features in the initial model. More details on how these characteristics are calculated are provided in Section 4.5. The most important takeaway from the data analysis is the **vast differences that exist between residential meters and commercial meters**. These groups of meters differ in the load scale, the scale of the performance and the number of meters. Figure 4.4 gives a graphical representation of the vast differences. Based on this information, we decided a separate model for each meter type was required in the next two chapters.

4.4.3 Relation with statistical characteristics

We analyse the monthly performance (the target label) with the average generation of the meters. The results can be found in Figure 4.5, which shows that, on average, the meter has a positive

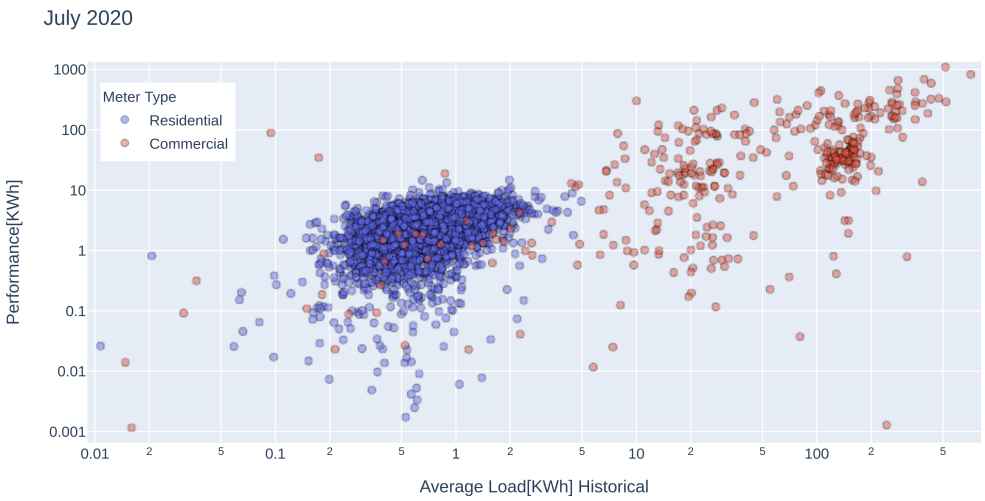


FIGURE 4.4: The historical average load in KWh compared to the performance in KWh in July 2020. The datapoints are coloured according to meter type. The types of meters vary massively.

average generation. However, this is by no means always the case. Many meters have an average negative generation. Especially the residential meters are often contributing negatively in terms of generation, which tells us something about the high stochasticity of the target variable.

A better potential feature is shown in Figure 4.6. It shows the relation between the generation and standard deviation of the historical performance. This visualization displays that there is some linear relationship between the two variables. Therefore, this variable is considered a potential feature for the base model.

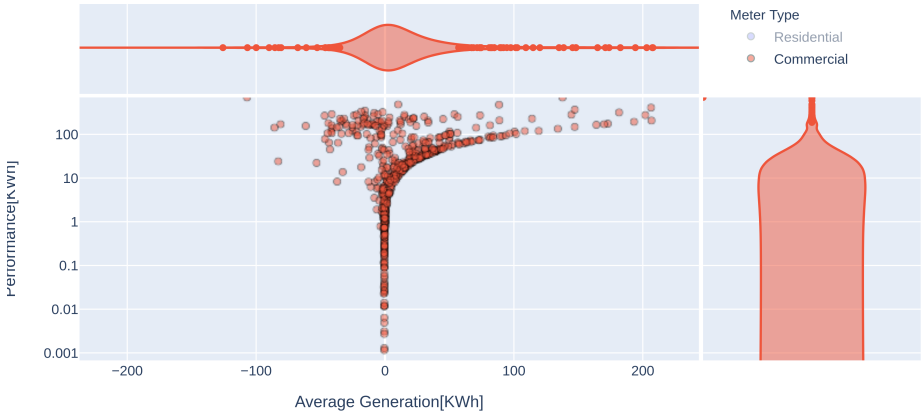
4.5 Features

Several characteristics have been calculated for each meter over different slices of the meter baselines and other data sources. The following three slices of the datasets have been used to calculate statistical characteristics for each meter and month combination.

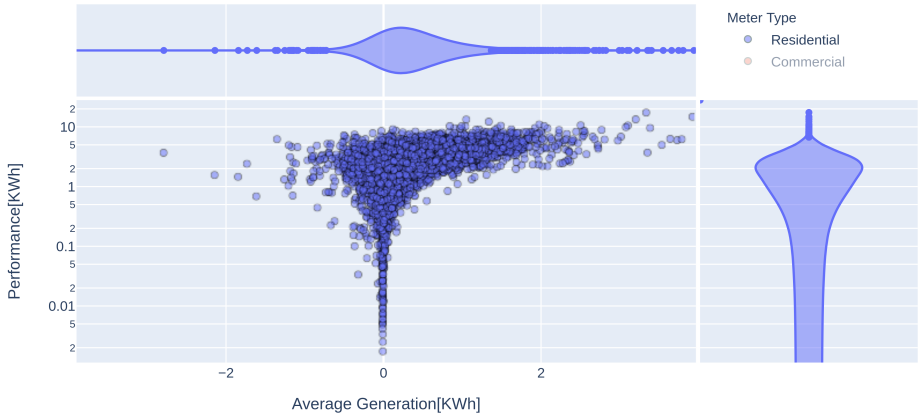
- **Entire history** - As the goal is to forecast three months into the future, the entire history of a meter with a given month as the target is capped three months before the start of the target month.
- **Most recent month** - Also, in this case, the three months before the target month are considered to be unavailable. Therefore this slice of the dataset only looks at the month three months separated from the target year.
- **Last year** - The last slice considers the same month, but one year ago.

Additionally, some cyclical features are included. We encode the month in terms of sine and cosine using a certain offset. Each characteristic is calculated for dispatch hours, non-dispatch hours, during the afternoon or just in general. Then, for each of the before mentioned dataset slices, the characteristics that are calculated are:

- Average of the load
- Standard deviation of the load
- Correlation between the load and the baseline
- Average of the generation



(A) The performance compared to the average performance of the commercial meters.



(B) The performance compared to the average performance of the residential meters.

FIGURE 4.5: The maximum performance (the target label) compared to the average performance of that same month (July 2020). The performance is displayed on a log scale; there were barely any negative performance values in the dataset.

- Standard deviation of the generation
- Maximum generation

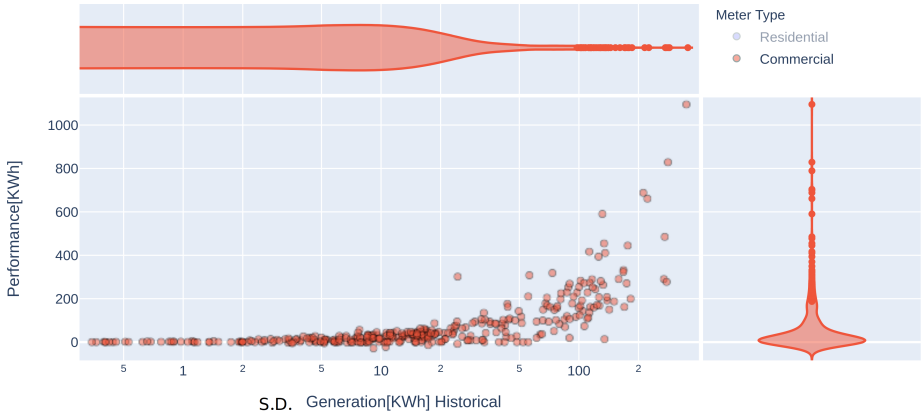
As the final step of the data analysis, we had a look at the correlations between these characteristics and the target label. This analysis was done for the entire datasets (all of the months) and individual months. We applied a translated logarithmic transformation to the target labels to look for logarithmic correlations. The analysis showed the most interesting results when running on all months and without the logarithmic scale. The results of that analysis can be seen in Figure 4.7.

Many of these meter characteristics show a high correlation with the monthly meter performance. However, from this analysis, it is not clear how much overlap there is between the correlations of different characteristics with the target label. Interestingly, the correlations with the target label do not seem to decrease rapidly with time, which suggests that we do not necessarily need fresh data to get good results.

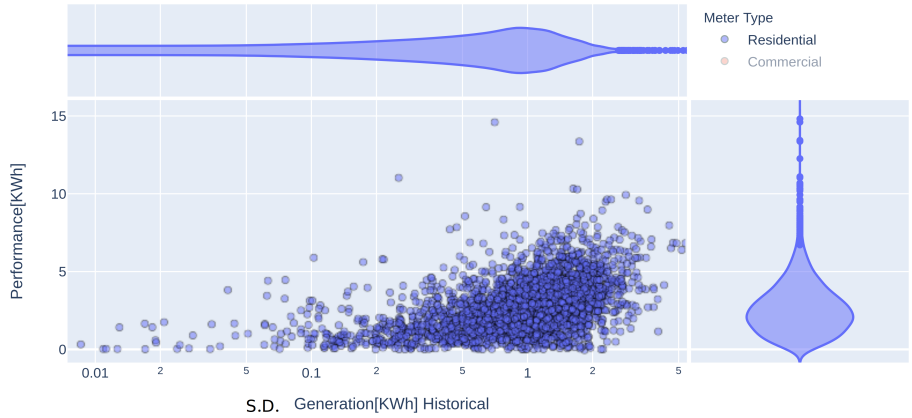
4.6 Insights

From this data analysis, there are a few key insights. These key insights largely determined the possibilities and limitations of the approaches we took to forecast meter capacity.

- The performance data is very sparse, and there even is a gap of 6 months without any performance data.
- Due to Leap's constant growth, many meters usually need a forecast from a cold-start scenario.



(A) The label versus the standard deviation of the historical performance for commercial meters.



(B) The label versus the standard deviation of the historical performance for residential meters.

FIGURE 4.6: The maximum performance (the target label) compared to the standard deviation of the historical performance. The x-axis is plotted on a log scale; we see some linear relationship between the standard deviation of the historical performance and the target label.

source	statistic	correlation
Historical	Afternoon Dispatched[%]	0.045848
	Average Afternoon Load[KWh]	0.490366
	Average Load[KWh]	0.483423
	Event Average Generation[%]	-0.088759
	Event Average Generation[KWh]	0.108993
	Event Load-Baseline Correlation	0.060992
	Event MSE Load-Baseline	0.557803
	Event Max Generation[%]	-0.125256
	Event Max Generation[KWh]	0.785409
	Event Std Generation[KWh]	0.786254
	Event Total Generation[KWh]	-0.000276
	Load-Baseline Correlation	0.111120
	MSE Load-Baseline	0.639690
	Non-event Load-Baseline Correlation	0.111039
	Non-event MSE Load-Baseline	0.637777
	Std Afternoon Load[KWh]	0.800925
	Std Load[KWh]	0.802930

source	statistic	correlation
RecentMonth	Afternoon Dispatched[%]	0.005376
	Average Afternoon Load[KWh]	0.485503
	Average Load[KWh]	0.481408
	Event Average Generation[%]	0.000428
	Event Average Generation[KWh]	0.265647
	Event Load-Baseline Correlation	0.032085
	Event MSE Load-Baseline	0.616179
	Event Max Generation[%]	0.000471
	Event Max Generation[KWh]	0.735543
	Event Std Generation[KWh]	0.692092
	Event Total Generation[KWh]	-0.084409
	Load-Baseline Correlation	0.065946
	MSE Load-Baseline	0.547802
	Non-event Load-Baseline Correlation	0.064999
	Non-event MSE Load-Baseline	0.543933
	Std Afternoon Load[KWh]	0.769135
	Std Load[KWh]	0.769298

source	statistic	correlation
LastYear	Afternoon Dispatched[%]	0.085660
	Average Afternoon Load[KWh]	0.782424
	Average Load[KWh]	0.756758
	Event Average Generation[%]	0.072329
	Event Average Generation[KWh]	0.041228
	Event Load-Baseline Correlation	0.026065
	Event MSE Load-Baseline	0.606309
	Event Max Generation[%]	0.083584
	Event Max Generation[KWh]	0.716107
	Event Std Generation[KWh]	0.786110
	Event Total Generation[KWh]	0.024617
	Load-Baseline Correlation	0.136810
	MSE Load-Baseline	0.569787
	Non-event Load-Baseline Correlation	0.136211
	Non-event MSE Load-Baseline	0.565467
	Std Afternoon Load[KWh]	0.803968
	Std Load[KWh]	0.780591

FIGURE 4.7: The correlations of the characteristics with the target label (monthly performance). These values are calculated for three dataset slices: historical, recent month and last year. This visualisation shows that the standard deviation and the average load, together with the standard deviation and the maximum performance, have the highest correlation with the target variables.

- There is a huge difference between the commercial and the residential meters.
- The demonstrated performance of (especially residential) meters is highly stochastic.
- Many meter characteristics show a high correlation with the target label.
- The correlation of these characteristics decreases slowly with time, suggesting that we do not necessarily need fresh data.

Chapter 5

Base Model

This research has been split into two phases; the first phase concerns the base model, the second phase concerns the advanced model. Leap's business needs require an improvement in the accuracy of the performance forecasts before the start of their operational season. Leap can use this base model to create forecasts for the coming summer months. This chapter covers the base model, with sections on the pre-processing, the model selection, the optimization, the results and the discussion of the model.

Currently, at Leap, the nominations are made in a manual process by the Partner Success team with the goal of forecasting meter performance. They make these forecasts in agreement with Leaps partners. This process is labour-intensive and difficult, leading to costly and inaccurate forecasts. To assist this team in their efforts, this project aims to develop a data-driven approach. The hypothesis was that a data-driven approach could improve the accuracy of the forecasts while saving time for the Partner Success team.

5.1 Data

The first phase started with a careful analysis of the data flows at Leap. This included considerations about data quality, data ingestion and validation strategy.

5.1.1 Processing

As per business requirements, forecasts need to be made a few days before the end of the month. The load data required for calculating the performance is provided by the IOUs. Usually, the delay for load data to be ingested is about two weeks. However, the IOU regularly update the values until several months after the load date [1]. There is no pattern in the way data gets updated over time (no positive or negative trend). Therefore, it has been decided that using data with a delay of 14 days is reasonable. [2] When all of these factors are combined, the model should be forecasting with a horizon of four months to be usable in a practical scenario.

Figure 5.1 shows the relations between the current time, the available data and the forecast horizon. As the Partner Success team has the ability to tweak the nominations slightly, the base model forecasts performance for each month leading up to the forecast horizon.

As explained in Chapter 4, there is plenty of data; however, the time series are very sparse and on a short time span. Most machine learning approaches to time series forecasting do not deal well with this many missing datapoints and short time series. Therefore, it was decided not to approach the problem as a time series forecasting problem, but instead approach it as a **regression problem**. The dataset is transformed to a tabular format by extracting summary statistics from the time series. Meters only have performance values when they are dispatched, which happens very infrequently. A single meter is likely only to be dispatched in **5 months of the year**. Therefore, the number of time series with several consecutive months of performance values is minimal. We can handle missing values in the features through ***imputation***. However, missing values in the target labels can not be handled in many optimization algorithms for traditional models. Instead of forecasting several months into the future given a single set of features, we add each target label

[1] There is no apparent reason for this behaviour by the IOUs. The IOUs are obliged by regulations to provide the load data. However, they have no (financial) incentive to do this properly and make it a streamlined process. Unfortunately, these processes are out of Leap's control.

[2] In the last week of developing the base model, it was decided to change the delay to 45 days. This aligns better with some of the other processes at Leap using the load data. The advanced model is developed using the longer delay.

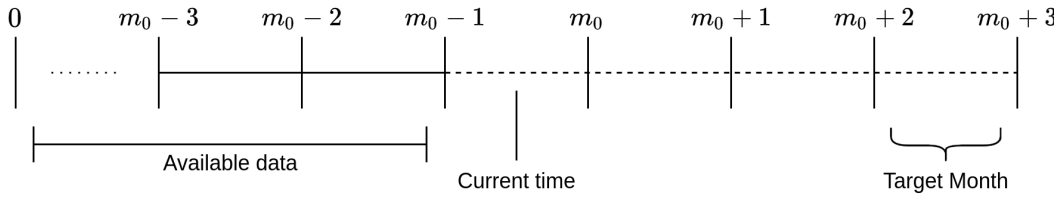


FIGURE 5.1: Schematic overview of the relation between the current time, the available data and the forecast horizon. m_0 denotes the end of the current month. Given that data has a delay of at least two weeks, the performance for the current month can be forecasted as soon as the data of the previous month is complete. As the goal of the project is to forecast three months into the future, the target month will be $m_0 + 3$, resulting in a forecast horizon of four months.

at a different horizon individually to the dataset, including an encoding of the month number. A schematic overview representing this process can be found in Figure 5.2.

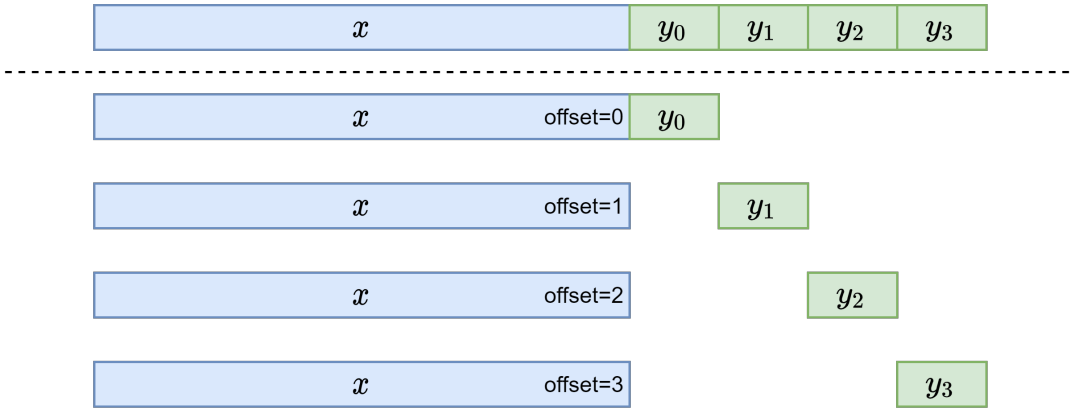


FIGURE 5.2: The process of transforming time series with many missing values into a tabular format. For each known performance value y_i, a sample with a set of features will be present in the dataset. If a meter did not get dispatched in a given month, then the target label is not known, and therefore the sample will not be present in the dataset.

5.1.2 Validation

To validate the quality of machine learning models, it is essential to measure the accuracy of a model that is trained using training data on validation data. When training machine learning models for time series forecasting, it is important to prevent ***information leakage*** from the validation data to the training data. In time series forecasting, information leakage can occur when the training and validation sets have an overlapping time span. Therefore, instead of using ***k-fold cross-validation***, it is common practice to apply so-called ***walk-forward validation*** in time series forecasting. In walk-forward validation, the model is trained on several slices of the dataset. A model is trained and evaluated for each slice by splitting the slice up into a training set and a validation set. This is done so that both the features and the labels of the samples in the training set are based on a time span before the time span on which the labels in the validation set are based. As a result, there is no information leakage in the training process of these models.

In standard walk-forward validation, the validation sets of all slices are non-overlapping with each other. Given that we have a forecast horizon of four months, this would give us only three slices per year. Leap has load and performance data only for a two-year time span. To use the data as effectively as possible, we decided to use a slightly modified version of the walk-forward validation. In this modified version, the validation sets in consecutive slices are allowed to overlap. For example, the first slice will have the months $[0,1,2,3,4]$ for training and the months $[5,6,7,8]$ for

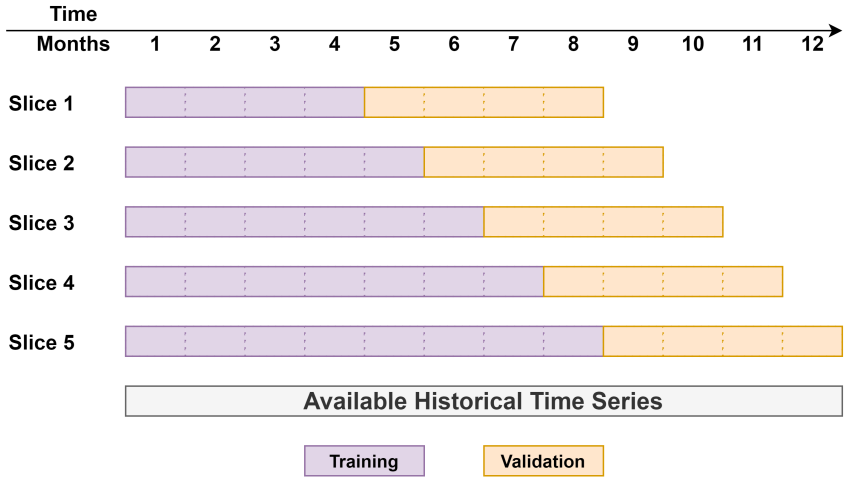


FIGURE 5.3: In the expanding window validation method, the validation sets from different slices are allowed to overlap. In the first slice, a certain time frame is used for training, and a consecutive time frame is used for validation. In each subsequent slice, we will move the training time span forward by one month, just like the validation time frame. There is no overlap between the training and validation sets within a single slice.

validation, the second slice will have the months [1,2,3,4,5] for training and the months [6,7,8,9] for validation, et cetera. As a result, we can extract more slices from the limited time span of data. We later found this same approach to have been developed at Uber by R. Yang (2019) as well under the name of the ***expanding window***. A schematic is shown in Figure 5.3.

This method of cross-validation limits information leakage through the model and its hyperparameters significantly. In this research, the expanding window was used such that the first slice contained the entire year of **2019 for training data** and the first four months of **2020 as validation data**. In each subsequent slice, the training data is expanded by one month, and the validation data is moved into the future by one month. In some of the months in 2020, there were no dispatches at all. After considering all of the above, we end up with about 7 slices per model. To measure the real-world accuracy of the model, we create a **test set out of the new data from 2021**. A downside of this approach is that it takes a year to gather a test set representing the accuracy throughout the entire year. The upside of this approach is that we measure the true accuracy of the model, as information leakage is excluded.

5.2 Model Type

The meters present on Leap's platform are considered low aggregate loads. This means that only a small number of devices or actors are present behind the meter. As a result, any stochasticity and change in behaviour of the actors behind the meter have a large impact on the load profile and, therefore, the curtailment capability. **Quantifying this stochasticity** is therefore valuable and motivates creating probabilistic performance forecasts as opposed to point forecasts, to gain insights into the uncertainty of the meter performance potential.

As residential and commercial meters are so different, we create two separate models and combine them into a hierarchical model. In this hierarchical model, we will use the same model type for all the meter types. A schematic overview of the hierarchical model is provided in Figure 5.4. As this is a base model, the model should be interpretable and quick to train. Therefore, traditional machine learning models are suitable.

Decision trees are good at handling missing values (Ding & Simonoff, 2010) through imputation and presence indicators. Gradient boosted decision trees have been shown to work well in tabular forecasting and were part of the winning algorithm in many machine learning competitions (Chen & Guestrin, 2016). Duan et al. (2020) present a method called NGBoost, for probabilistic regression using gradient boosted decision trees at its core. As pointed out by Duan et al. (2020), many traditional machine learning models cannot provide probabilistic regression. Even

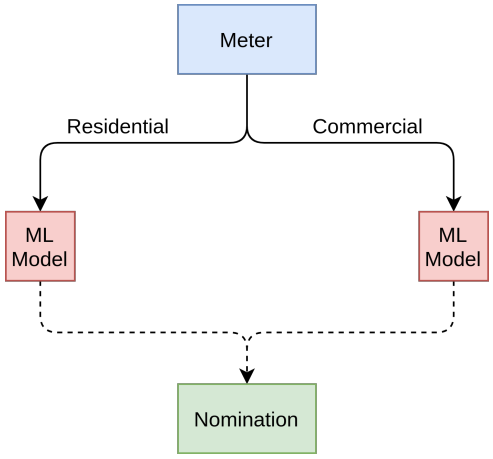


FIGURE 5.4: Schematic overview of the hierarchical base model. In the future, we can extend this model even further by separating the dataset into smaller subgroups.

if they can, they only provide ***homoscedastic*** regression, which means each forecast has the same variance. It is beneficial to produce ***heteroscedastic*** regression as the meters in the dataset are all of very different scales and have different characteristics. The NGBoost model has been designed to provide heteroscedastic regression while maintaining comparable accuracy to the best point forecast machine learning models.

To ensure the NGBoost model will be able to ingest the features and produce forecasts properly, it is part of a **pre-processing and post-processing pipeline**. In the optimization process, we try to find the optimal configuration of this entire pipeline. The preprocessing steps include feature selection and potentially include (logarithmically) scaling the inputs and outputs, transforming the categorical features using one-hot encoding, imputation of missing values and adding indicators for missing values.[3] The hyperparameter tuning step potentially tweaks the learning rate, the number of estimators/iterations, the predicted distribution and early stopping.

5.3 Optimization

Before training the model, the meter data is transformed to a tabular format. Training algorithms are at the risk of overfitting the training data and failing to generalize patterns to out-of-sample data. Although boosting algorithms have a lower tendency to overfit (Breiman, 2001) compared to other traditional machine learning techniques, it is still a relevant concern. Therefore, an important step in the optimization process is selecting **the right (number of) features**. The other step in the optimization process concerns finding good hyperparameters, such as the learning rate or imputation strategy.

The time that we could spend on developing the base model was restricted as per Leap's requirements. We struck a balance to try to find the most optimal algorithm within the constraints. A schematic overview of the entire optimization process of the base model is provided in Figure 5.5. As this optimization process is not an exhaustive search but rather an iterative search, there is no guarantee to find the optimal combination. However, given the constraints and the fact that domain knowledge plays a big role, we do not consider this a major problem for finding a good base model.

5.3.1 Feature Selection

There are several types of features presented to the model. There is metadata, such as the partner a meter belongs to or the type of device behind the meter, like electric vehicle or air-conditioning. For each meter, there is up to two years of 15-minute interval time series data. We can extract features from the historical load by calculating summary statistics. If a meter has been on Leap's

[3] Just like proposed by Salinas et al. (2019), we tried scaling the timeseries by a meter specific constant. However, this did not result in an improvement early on; therefore, we abandoned this idea for the base model.

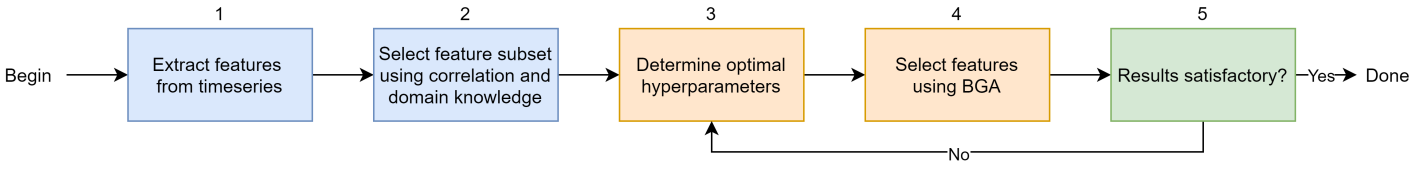


FIGURE 5.5: A schematic overview of the optimization process. The process starts with step 1 and 2 which closely aligns with the data exploration described in Chapter 4. Then steps 3, 4 and 5 form an iterative process of consecutively tuning the hyperparameters and selecting features. At the end of each iteration, we evaluated whether more improvement was expected and used that to decided to continue optimization or to stop.

platform for some time, it might have been dispatched, and we can calculate summary statistics on the historical performance. There are **virtually unlimited potential features** that can be calculated from the time series.

We focused on the features that we presented in Section 4.5. Using domain knowledge, data analysis and the correlation with the target variable, a subset of features was selected to be used in the next steps. This analysis found that using different data ranges than the entire history almost always resulted in a (slightly) lower correlation with the target label. Therefore, we decided only to use features that have been calculated over the **entire available historical data range**. The selected features are shown in Table 5.2.

Data Range	Selected Intervals	Statistical Metric
Entire history	All intervals	Average
Most recent available month	Afternoon intervals (4pm-9pm)	Standard deviation
Month one year before the target month[4]	Non-event intervals	Maximum
	Event intervals	Correlation with baseline

TABLE 5.1: We extracted a large set of features from the time series of several variables, such as the load or the baseline. For these features, we first select a certain date range to use, then within this date range, a certain set of intervals is selected, and then we calculate a summary statistic over the data in our selection. This resulted in a large set of potential features as analysed in Section 4.5.

Hyperparameter Optimization

As described in Figure 5.5, we employ an iterative process to find a good feature subset and hyperparameter configuration. There are hyperparameters in the pre-processing of the selected features and the training process of the model. Potential hyperparameters in the pre-processing included the scaling of in- and output labels, imputation strategies and inclusion of missing value indicators. Potential hyperparameters in the model included the learning rate, the number of estimators/iterations, the predicted distribution and early stopping. After initial experimentation, we fixed a certain subset of hyperparameters, and we gave another subset of hyperparameters a set of potential values to be used in a gridsearch. This **fixed-configuration** includes the disabled early stopping, the subtraction of the mean and the scaling of the variance of the input features to 1[6] and the usage of the normal distribution. The potential hyperparameter values in the search space are displayed in Table 5.3.

[4] Clarification: At any point in time, the goal is to forecast four months into the future. This data range calculates features based on the same month one year ago. Relative to the most recent month of available data, that is eight months ago. (See Figure 5.1)

[5] Extreme weather mainly dictates the potential for dispatches. Therefore, sinus encodings are shifted such that the value of 1 aligns with the hottest month and -1 aligns with the coldest months.

[6] A serious attempt was made to improve the predictions using power scaling either or both the input and output values. However, this gave no meaningful improvement.

Description	Data Type
Entire History - Afternoon - Average - Load	Float
Entire History - Afternoon - Standard deviation - Load	Float
Entire History - Afternoon - Maximum - Load	Float
Entire History - Afternoon - Average - Generation	Float
Entire History - Afternoon - Standard deviation - Generation	Float
Entire History - Afternoon - Maximum - Generation	Float
Entire History - Afternoon - Average - Performance	Float
Entire History - Afternoon - Standard deviation - Performance	Float
Entire History - Afternoon - Maximum - Performance	Float
'Last month'	Integer
Sinus of the 'last month'[5]	Float
'Target month'	Integer
Sinus of the 'target month'[5]	Float
Partner	Categorical
Load Type	Categorical

TABLE 5.2: The subset of potential features to be used in the further steps of optimization. 'Last month' denotes the number of the last month of available data(1-12). 'Target month' denotes the number of the month at the forecast horizon(1-12).

Parameter	Commercial model	Residential model
Missing indicator	True, False	True, False
Imputation strategy	Mean, Constant outside distribution	Mean, Constant outside distribution
Learning rate	0.1, 0.01, 0.001	0.1, 0.01, 0.001
Number of estimators	700 - 1300 with steps of 50	200 - 340 with steps of 20

TABLE 5.3: The hyperparameter search space. After fixing a certain set of hyperparameters given the initial exploration, we found a certain set of hyperparameters to be impactful, but we were unsure of the optimal value. Therefore they are part of the hyperparameter gridsearch.

Binary Genetic Algorithm

Using too many features with too much redundant data can lead to overfitting and reduced accuracy. Finding a good set of features can be classified as a combinatorial optimization problem. In related work ***Genetic Algorithms*** (GAs), have been applied to these kinds of problems, as well as specifically for feature selection (Babatunde et al., 2014; J. Lu, Zhao, & Zhang, 2008).

In hindsight, we can conclude that the ***Binary Genetic Algorithm*** (BGA) likely did not do a better job at feature selection as other simpler feature selection techniques could have done. Although there are cases of using BGAs for feature selection, it is by no means the standard. We mainly selected this approach out of curiosity. As the results were not very promising or impactful, this section will be kept short.

Using the hyperparameters for the pipeline as selected in the previous step, the BGA was run to find the optimal feature subset. The BGA used a total of **500 individuals** per generation, a total of **25 generations** and a **mutation rate of** 0.05. The selection step is performed using **tournaments with three individuals**. Each individual has DNA with 15 binary values, indicating the inclusion or exclusion of a certain feature. In Figure 5.6 , we see that the distribution of the negative log-likelihood scores decreases over time. However, the best feature subset found does not improve significantly over the course of the generations. We made this observation across several runs of the BGA. In Figure 5.7, we see how the presence of a certain feature changes over several generations until we land in a stable configuration. Some features are instantly discarded or tossed, while others take some more time to explore their usefulness. This process is quite stable, although additional analysis pointed out that we can swap out some of the features for other comparable features without much impact.

The author concludes that simpler and more common feature selection practices would have probably been more suitable for this problem.

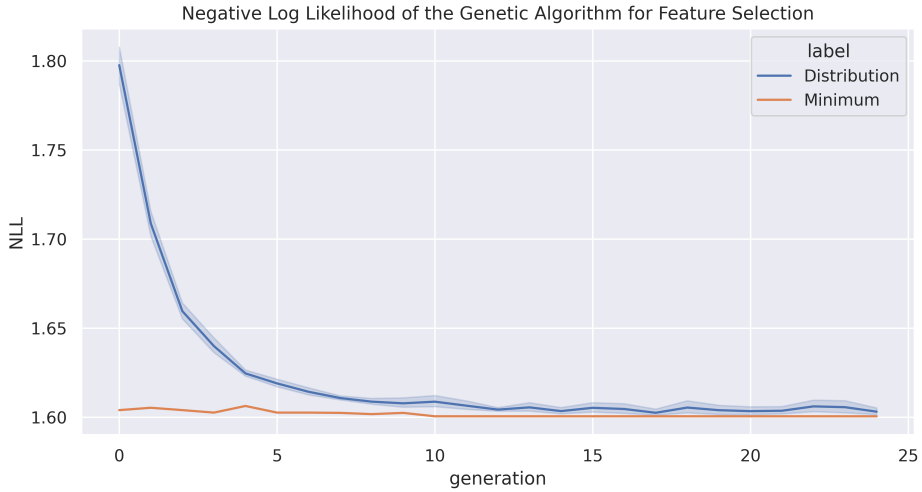


FIGURE 5.6: The distribution and the minimum of the negative log-likelihood in the binary genetic algorithm. The distribution decreases neatly, however from the initial iteration, there is not much improvement in the minimum.

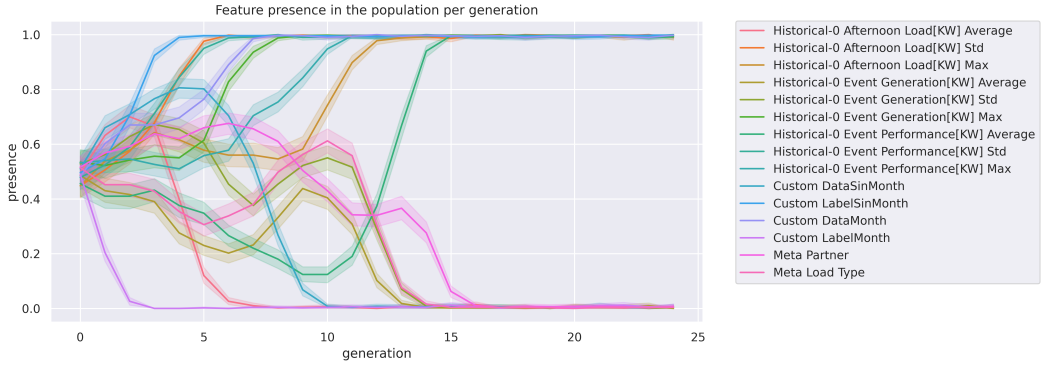


FIGURE 5.7: The inclusion rate of a certain feature per generation. It is clear to see that some features are quickly selected or discarded, while for other features the algorithm does a bit more exploration before making a decision.

	Commercial	Residential
Features	Afternoon - Standard deviation - Load Afternoon - Maximum - Load Afternoon - Average - Performance Afternoon - Maximum - Performance Last month of available data (1-12) Partner	Afternoon - Standard deviation - Load Afternoon - Maximum - Load Afternoon - Maximum - Generation Afternoon - Average - Performance Afternoon - Standard deviation - Performance Afternoon - Maximum - Performance Last month of available data (1-12) Sinus of the target month
Missing indicator	False	True
Imputation strategy	Mean	Mean
Learning rate	0.01	0.01
Number of estimators	950	240

TABLE 5.4: The optimized configuration of selected features and hyperparameters per model type. There are some apparent and significant distinctions in the type and number of features and training iterations.

5.4 Results

We executed the optimization procedure for three consecutive iterations. From this point on, the improvements in negative log-likelihood were minimal. The selected configuration of the machine learning pipeline is displayed in Table 5.4.

Using this configuration, the pipeline is trained for both the residential and commercial meters on data up and until November 2020. As previously mentioned we can now use **February 2021 for evaluation purposes**. In February 2021 only commercial meters were dispatched; this limits the evaluation in this chapter to the commercial model only. In Section 6.3, we will dive more deeply into the performance of the base model for commercial and residential meters. This section shows the insights and evaluation that was conducted right at the end of developing the base model. We compare the predictions with the nominations made in the manual process by the Partner Success team. As manual predictions are point predictions, we can only use point forecast metrics in this analysis. The comparison of the processes on the ***Mean Bias Error*** (MBE) and the ***Root Mean Squared Error*** (RMSE) can be seen in Figure 5.8. This figure shows that the base model outperforms the manual process both in terms of bias and RMSE.

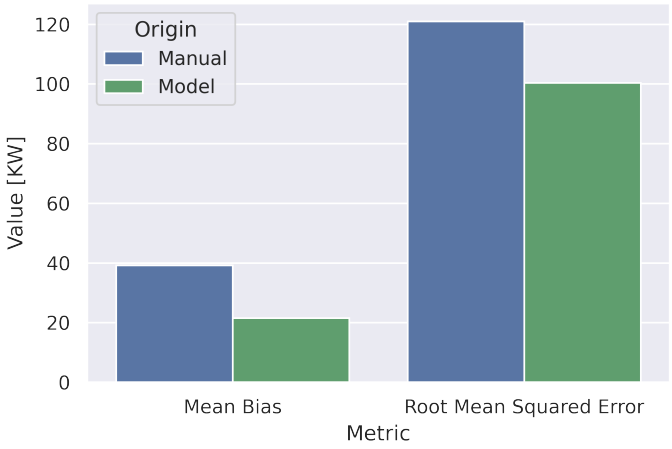
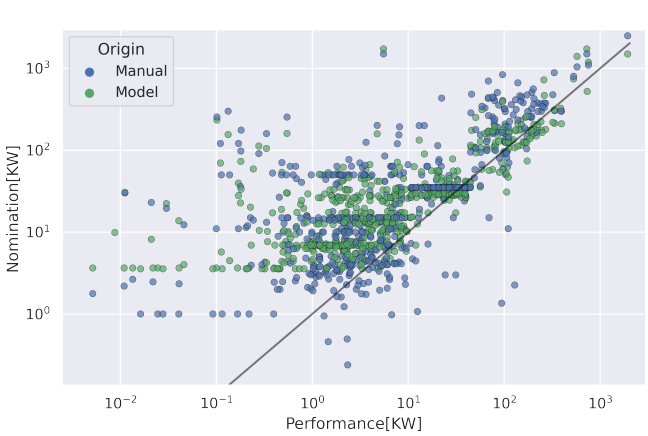


FIGURE 5.8: Comparison of the accuracy of the manual process versus the model process in terms of mean bias and root mean squared error.



(A) In general, both processes provide very similar nominations. Also, the observed positive bias is clearly seen as most points lay above the grey line.

(B) A focus of Figure 5.9a on the larger meters. In this area, there is a clearer difference between the manual and the model process. Clearly, the model's predictions are better.

FIGURE 5.9: Scatterplot of the observed performance compared to the nominated values on a log-log scale. The grey line denotes equality and is, therefore, the optimum. The blue points represent the manual process, and the green points represent the model's process.

5.4.1 Error Analysis

To simplify the error analysis, we only consider the forecasts made at the forecast horizon as this is the project's most important objective. In Figure 5.9, we scatter both the observed performance and the nominations made by the manual process and the model. The most important observation is that the predictions made by both processes are **relatively comparable**. The 'lines' of horizontal datapoints in the manual observations correspond with the practice of assigning meters to groups and then assigning a single nomination to all of the meters in the group. The model's nominations do not go below a certain point. This is likely the result of optimizing for the negative log-likelihood, in which case the high-performance meters are considered more important. Therefore less optimization work is done at the lower part of the predictions. It seems the manual process is slightly more accurate on the lower end of the performance. However, Figure 5.9b shows the model is more accurate on the high-performance meters. This observation is confirmed in Figure 5.10. In this figure, the same metrics are calculated over several quantiles of the observed performance. Separating by observed performance could potentially skew the figures and give an tainted view on the data. That is why in Chapter 6.3 we group the meters by historical average load instead. There is a significant difference between the two processes in the upper quantile.

Further analysis taught us that the model's improvement over the manual process could largely be explained using the nominations for a single partner. The accuracy metrics are calculated for each of the three partners with the most dispatched meters. Figure 5.11 compares the manual with the model process per partner. Here it is clear that for the partner GridPoint, the manual process massively outperforms the model predictions; there are mainly small meters. It is also clear that for the partner Tesla, the model process massively outperforms the manual predictions. This partner mainly has large meters.

5.5 Discussion

The results turn out to be very promising for Leap. A drawback is that, in this Chapter at the time of writing, we have only been able to evaluate the commercial model, only on the month of February. Currently, the data science team at leap is actively developing the required infrastructure to run continued evaluations of the base model. Once this infrastructure is in place, the continued evaluation will start building the trust of the model within Leap. Then we will try to get the model

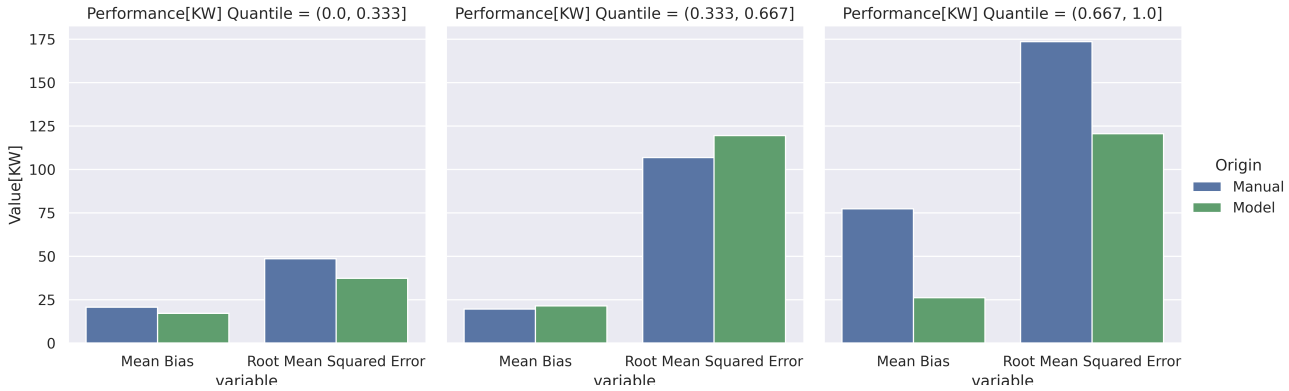


FIGURE 5.10: Calculation of the mean bias and the root mean squared error where we separate the performance into three equal quantiles. In the largest quantile, we observe a very significant difference.

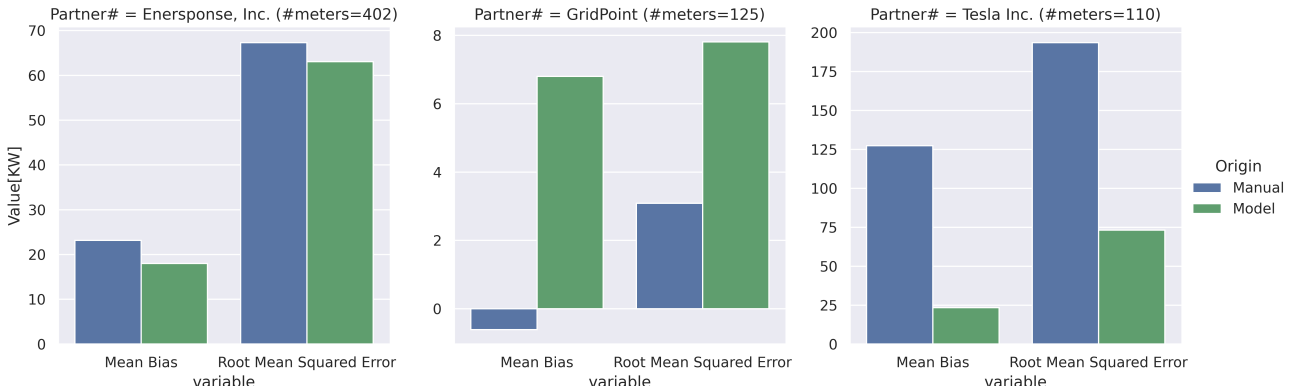


FIGURE 5.11: Calculation of the mean bias and the root mean squared error where we separate the datapoints by the largest three partners that got dispatched this month. A clear difference can be seen between two of the partners. An important observation is the difference in scale of each partner, which is explained since different partners have meters of very different types.

to be used as soon as possible. Chapter 6 already extends the analysis provided in this section with the data that was available at that time.

Currently, datapoints with a large performance are considered more important in the training process due to optimising the negative log-likelihood. Additionally, meters that have been dispatched more frequently occur more frequently in the dataset and months in which there were more dispatches occur more frequently in the dataset. Balancing the dataset might improve the generalization capability of the model. What also might improve is expanding the hierarchical model based on the scale of a meter, as currently, the smaller meters are undervalued in the base model. These learnings will be used by the data science team to improve the model and have been used in this project to develop the advanced model.

Chapter 6

Advanced Model

As stated at the beginning of Chapter 5, this research is split into two distinct phases. The second phase consists of developing an advanced model which will apply state-of-the-art techniques and apply it to provide a solution for the nomination forecasting problem. This chapter will completely cover the data ingestion, the preprocessing, the optimization, the results and the discussion of the advanced model.

The base model provided Leap with fully automated and more accurate nominations than the current manual process. To make the results usable at Leap, the rest of the data science team started developing supporting infrastructure. I assisted in the development of this infrastructure and leveraged this new infrastructure to develop the advanced model. As mentioned in Chapter 3, most of the challenges posed by the nomination forecasting problem are solved if a suitable multi-granularity model can be developed.

Training DL models on time series is usually done using ***Time Series Datasets*** (TSDs), in which multiple training samples are extracted from the multiple time series present in the dataset. The main contribution of the advanced model is the development of the ***Multi-Granularity Time Series Dataset*** (MGTSD) and applying it to the nomination forecasting problem using the TFT. The TFT architecture features a separate encoder and decoder architecture. Therefore, the challenges of a multi-granularity model are limited to the pre-processing of multi-granular data and are solved by the MGTSD.

6.1 Data & Model

Many of the challenges encountered during the development of the base model are directly transferable to the advanced model. The same data pipelines are feeding in the data for both calculating the performance and providing meter data. Additionally, we are dealing with the same irregularities of dispatches and thus of performance data on the short time span. Therefore, the same validation process will be used for the advanced model as proposed in Section 5.1.2.

6.1.1 Processing

Throughout this research, the definition of performance has changed and led to much confusion between many colleagues and me. We implemented a feature store in collaboration with my colleagues to standardize the definitions and streamline the development of these kinds of models. This feature store provides us with the meter data to be used in the advanced model. There are thousands of meters in this dataset. As each meter has data for every 15-minute interval, the dataset is huge in terms of datapoints and, thus, bytes. The data is aggregated to the hourly level to deal with this size using our limited computational resources. It is known that there is a large difference in meter characteristics between commercial and residential meters. Therefore, also for the advanced model, separate versions will be trained for both meter types. This results in a dataset of a few gigabytes instead of several tens of gigabytes of data.

6.1.2 Features

One of the major advantages of DL approaches is that it limits the need for extensive feature engineering. As we saw in Section 4.5, there are virtually unlimited features one can extract from a TSD. In DL, instead of engineering relevant features, the practitioner tries to engineer the most fitting architecture to the problem. As a result, the DL model will learn to extract relevant features from

the input by itself. For example, traditionally, a practitioner trying to train a model to recognize dogs in images would engineer a feature that recognizes eyes. However, in DL, there are architectures specialized for processing image data; these architectures potentially learn to detect eyes by themselves, without knowing or understanding what eyes are.

This flexibility and the automated feature extraction also increase the complexity of such models and reduce the interpretability. For example, it is difficult to determine what the architecture in the above example is looking for. If the model turns out to have low accuracy or a high bias, then it is not easy to figure out why for DL models. On top of that, DL architectures need a huge amount of data to train. Even though the datasets are several gigabytes large, we cannot extract that many training samples from them due to the sparseness at the monthly granularity.

We hypothesize that a suitable DL architecture might extract the relevant features from the high granularity meter data. Therefore, we use simple features like the hourly load and hourly baseline for the encoder. To best support the architecture in finding relevant patterns, we also create several encodings of existing features. These include:

- **Performance measures**: Next to the raw load and baseline, we also include the generation value, average past two hour generation and the average past two-hour performance (generation during events).
- **Missing indicators**: Some variables like the average past two-hour performance do not always have a value, like when a meter is not dispatched. Because the TFT cannot deal with missing values, those values are imputed. To inform the TFT about these missing values, we add a feature that indicates whether the value was imputed.
- **Time encoding**: Variables like the hour in the day, day of the week, day of the year or month of the year. These values can then be encoded simply as an integer or using more advanced cyclic encoding by taking the sinus or cosine of the variable in such a way that all the possible values span exactly one full period.

6.1.3 Time Series Dataset

The major challenge of the advanced model is dealing with multi-granularity data.

Single-Granularity Time Series Dataset

Most forecasting problems work on a single granularity. A trivial single granularity forecasting problem would be predicting the future hourly load using the historical hourly load. Several ML models are restricted to these single-granularity problems, such as most traditional time series models like ARIMA and modern DL architectures like DeepAR. In these scenarios, we can create several training samples from each time series in our dataset. In this research, we used a library called 'pytorch_forecasting' that provides the single-granularity time series dataset functionality. A schematic for that process is shown in Figure 6.1.

Multi-Granularity Time Series Dataset

In this research, we are trying to forecast meter performance. The trivial approach would be to forecast the meter performance by analyzing the performance time series. However, the performance time series is very sparse. Fortunately, the performance is calculated from several high granularity time series that are not sparse. This shows our need for a multi-granularity approach to forecasting the meter performance.

Due to the separation between the encoder and decoder of the TFT architecture, it is possible to use different granularities at both ends of the model. Therefore, the major challenge is creating a ***Multi-Granularity Time Series Dataset*** (MGTSD) that will create samples to feed into the model. Instead of creating an index from the low granularity dataset, the process is slightly changed. This is done by customizing the single-granularity time series dataset provided by 'pytorch_forecasting'. We aggregate the high granularity encoder level dataset into the lower granularity decoder level dataset. This aggregation can be customized depending on the problem. The number of ways in which this aggregation can be conducted is virtually infinite and highly problem dependent. Therefore, it is up to the practitioner to decide what would make the most sense to solve the task at hand. The decoder level dataset keeps a reference to the encoder level data used to create it.

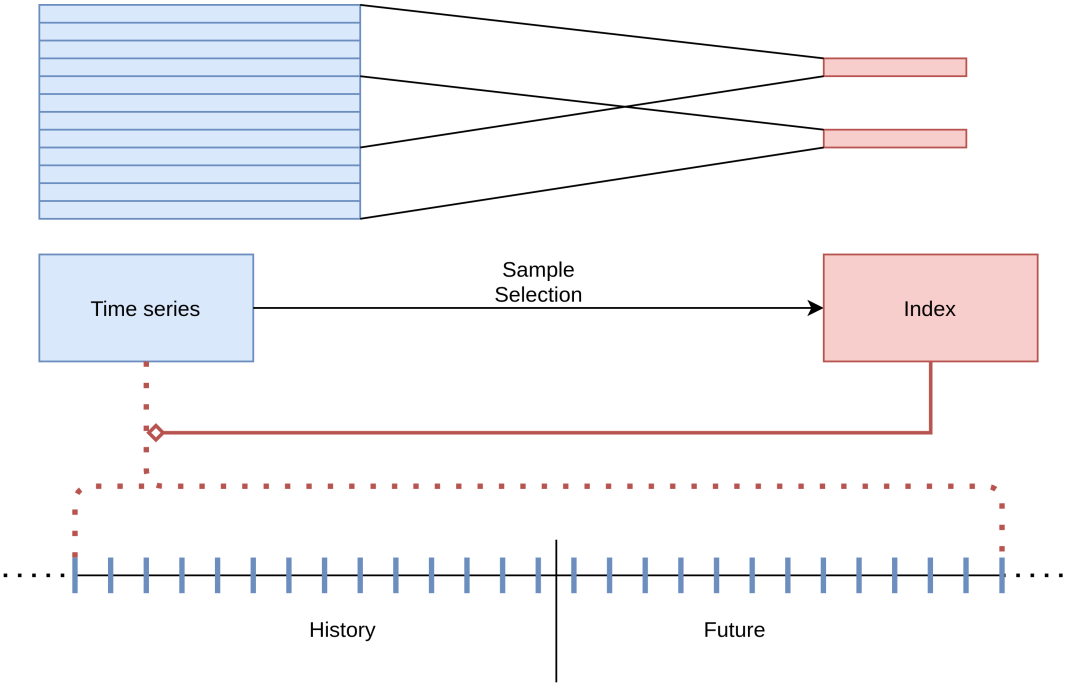


FIGURE 6.1: The process of extracting training samples from a single-granularity time series dataset. In blue, on the top left, is a time series with several measurements. A sample is selected by choosing a certain time range from the time series; this range will be saved in the index (red). From that same time series, more samples can be extracted. Each of these samples will have a record in the index pointing to them. These samples are allowed to overlap within a time series. The time series dataset potentially contains many different time series. During training, the index will point to the time series to create multiple unique samples. Part of the retrieved sample will be used as the history and fed into the encoder, and a part of it will be used as the future or, in other words, as the target for the decoder. As seen at the bottom of the figure, both the encoder and the decoder have the same granularity.

Then an index is extracted from the decoder level dataset, just like in the single-granularity time series dataset. The difference is that this index now points to part of the time series at the decoder granularity and part of the time series at the encoder granularity. The process is schematically described in Figure 6.2.

The final implementation of the MGTSD integrates seamlessly with the forecasting library. As a result, we can use its training process, the implementation of TFT and optimization tools without any major issues.

6.2 Optimization

The MGTSD will generate training samples from the hourly meter data. The training samples generated by the MGTSD are used to train a ***Temporal Fusion Transformer*** (TFT) architecture as designed by (Lim et al., 2020). The separation between the encoder and the decoder in the TFT architecture enables us to use a different granularity on both ends of the model.

Just like the base model, the advanced model is trained using a certain set of features. As the advanced model directly ingests the high granularity time series, there are much fewer potential features. Nevertheless, it still requires a practitioner with domain knowledge to optimize the set of features. We dedicated much time to developing the advanced model. Therefore, there was not much time spent on optimizing the feature set. Instead, we used our domain knowledge to try several sensible configurations. All configurations consisted of the hourly load, baseline, generation and two-hour rolling performance features. We explored about ten different combinations of time encoding and missing indicator features. We selected these features to align with the known seasonalities and known inter-variable relations as well as possible.

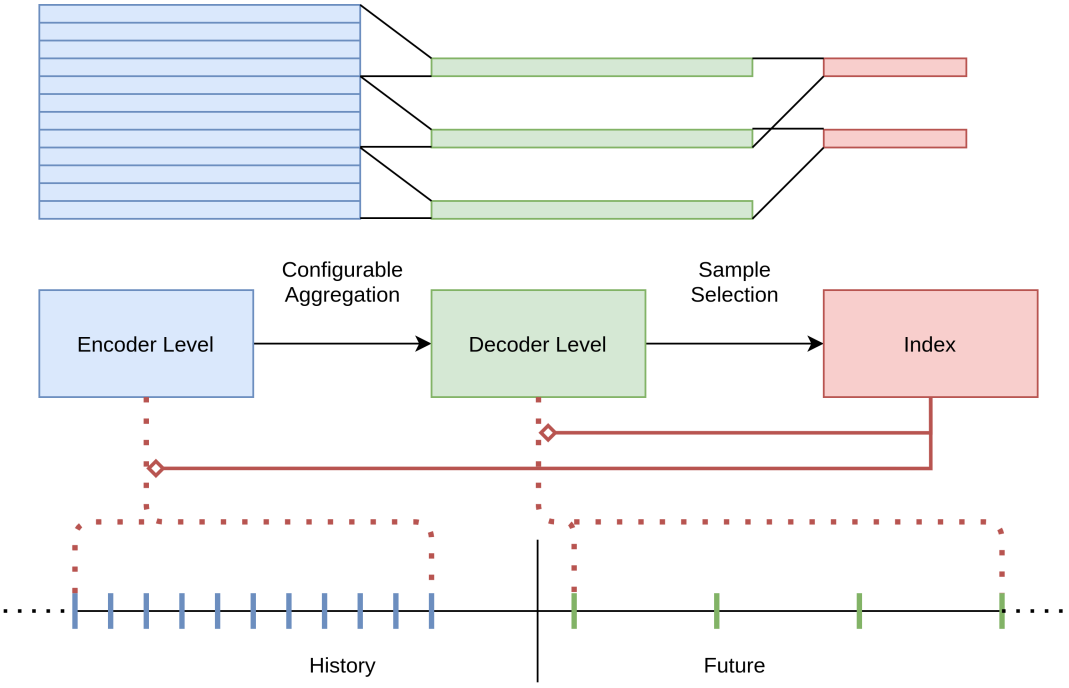


FIGURE 6.2: The process of extracting training samples from a multi-granularity time series dataset. Like in the single-granularity case, we have the time series dataset in blue on the top left with the encoder level granularity. This encoder level data is aggregated to the decoder granularity; this dataset is shown in green. An index will be constructed from this decoder level data, like in the single-granularity time series dataset. The index now also deduces which encoder level records occur right before the encoder level records used to create the decoder level records the index already points to. (these lines are not drawn to improve the clarity of the figure) The bottom of the figure shows how the index points to both datasets with different granularities. Additionally, we see how the granularities are different in the history and the future or, in other words, at the encoder side and the decoder side.

Just like the base model, the advanced model is configured using a certain set of hyperparameters. The most important parameters that we optimize, with the explored intervals specified in the end, are:

- **Hidden layer size**: The number of neurons in the hidden layers throughout the architecture. This determines the width of the network and indirectly the number of parameters or capacity of the model. [16, 265]
- **Attention heads**: The number of attention heads, as explained in Section 2.5.5. [1, 4]
- **Dropout range**: The value of the dropout parameter p, as explained in Section 2.5.2. [0.1, 0.3]

This search space is the recommended search space for the TFT architecture. These hyperparameters are optimized using a random search in which 50 trials are conducted and evaluated on a single training and validation split. Then using the best configuration, the walk-forward validation is executed.

We train the model by optimizing the multi-quantile loss where ρ has values $[0.02, 0.1, 0.25, 0.5, 0.75, 0.9, 0.98]$, as described in Section 2.5.1. The model is optimized using the Ranger (Wright, 2019) optimization algorithm, the default for TFT in the 'pytorch_forecasting' library. Based on the walk-forward validation results, the best set of features and hyperparameters is selected. Previous to this, we found a learning rate of 0.1 to work the best. The optimization process of the advanced model is schematically displayed in Figure 6.3. We executed the cycle in this process about three times. Using this configuration, we train a model for inference on all the available data (until the start of 2021) and use it to forecast the meter performances for January up to and until June.

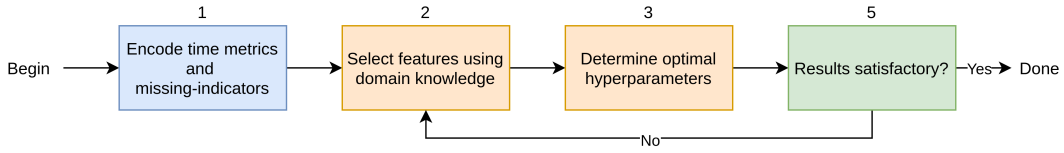


FIGURE 6.3: The advanced model optimization process. First, we define several extra features and then start an iterative process of feature set optimization.

6.3 Results

As Leap is a very young company, infrastructure around developing, deploying and evaluating ML models is very minimal if not absent. Therefore, we spent much time on developing and debugging not only the MGTSD but also data pipelines and computational resources. As a result, the time spent optimizing the feature set and thus the model's performance has been minimal, especially compared to the base model. In this results section, we compare the predictions made using the manual process and the advanced model. As a result of the previously described optimization process, we find the following configurations for the commercial and residential advanced models as displayed in Table 6.1. The difference between the commercial and the residential model is minimal. This is partially explained due to the minimal optimization we were able to conduct on the feature set. We consistently see during the optimization that the residential model has a lower optimal dropout rate but a larger hidden layer size.

	Commercial	Residential
Static Features	Partner name	Average load
	Average load	
Known Features	Hours since UNIX epoch (mean)	Hours since UNIX epoch (mean)
	The month of the year	The month of the year
	Cosine of the day of the year (mean)[1]	Cosine of the day of the year (mean)
Unknown Features	Is business day	Is business day
	Is event hour	Is event hour
	Has two-hour rolling performance [2]	Has two-hour rolling performance
	The hour in day	The hour in day
	Load	Load
	Baseline	Baseline
	Generation	Generation
	Rolling two-hour performance	Rolling two-hour performance
Dropout	0.3	0.15
Attention heads	3	2
Hidden layer size	78	142

TABLE 6.1: The best configurations that we found in the limited optimization process. Known features can be determined for the history and the future and are thus fed into the encoder and the decoder. Unknown features are measured in the history and are thus only fed into the encoder. If applicable, the text in parentheses describes the aggregation function to go from the encoder granularity to the decoder granularity.

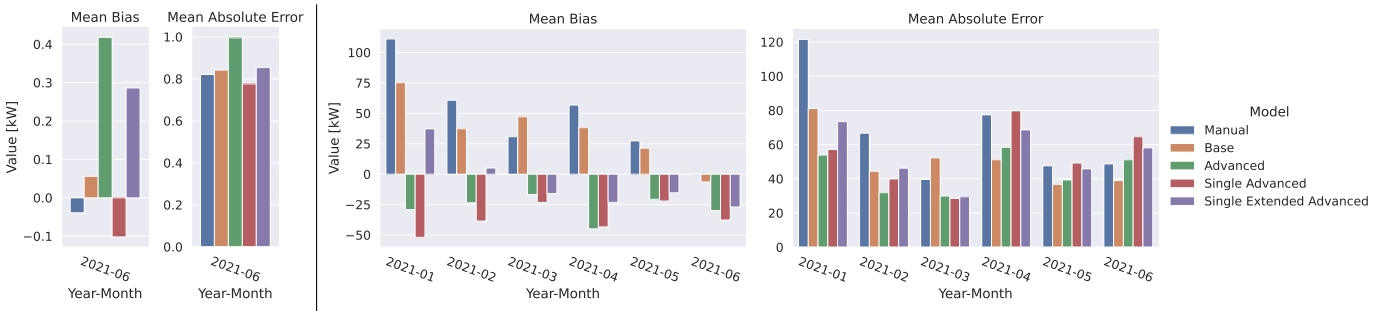
There is a large difference between the base model and the advanced model in the way they are optimized. The base model assumes a Gaussian distribution of the target label and **optimizes the maximum likelihood**, which is equivalent to minimizing the root mean squared error. The advanced model does not assume any distribution and **optimizes the quantile loss**, which is closely related to the mean absolute error. It is very problem dependent which metric provides the best perspective on the quality of a model. One of the most important uses of the nominations at Leap is creating supply plans by aggregating meter nominations. In this use case, having one nomination that overestimates with 90 kW and one meter that overestimates with 10 kW is not worse than having two nominations that overestimate with 50 kW. Therefore, we focus on the MAE as it is more important than the RMSE for this problem.

6.3.1 Error Analysis

The manual nominations are made three months before the start of a month at the latest. The historical performance data is deemed trustworthy after 45 days. As a result, we have to train our model to predict with a forecast horizon of six months. [3] In this analysis, we will be comparing the nominations from January 2021 up to and until June 2021.

Ideally, we would train a separate model and create a forecast right at the end of the forecast horizon for each of these six months. However, the limited time span of the available data would lead to a minimal number of training samples for most of the months. Therefore, we will have to settle for a single six-month horizon forecast and compare that with the manual nominations. We know from domain knowledge that the meter performance is **largely determined by two factors**: the weather and consumer behaviour. Weather-related forecasts are **not** used in our models. Consumer behaviour is unlikely to change drastically in a short amount of time. Therefore, the model will not access much more information due to the smaller time difference between the present and the forecasted month. Hence, we expect that the impact on the performance metrics using a single forecast will be limited.

In the following subsections, we describe our error analysis. We discuss several models, meters of different sizes, cold-start meters and uncertainty predictions. In the analysis, we considered many more views on these results, for example, by analysing the results by partner. However, we omit these analyses as there were either no patterns or there was insufficient data to draw conclusions from them.



(A) The residential meters. About 5500 got dispatched in June.

(B) The commercial meters got dispatched in each month of 2021, with varying numbers of dispatched meters per month. In January, March and April, about 150 meters got dispatched. In the remaining months, about 600 meters got dispatched.

FIGURE 6.4: A comparison of several metrics for the first six months of 2021 between the manual process, the base model and the advanced models.

Models

We compare the accuracy of the manual process with the base model and the advanced model. In developing the advanced model, we propose an experimental approach to preprocessing the data for the ANN, which we call MGTSD. To determine the effect of the MGTSD, we conduct an ***ablation*** analysis. Ablation is the process of removing a component from an AI system to quantify the effect of said component. Next to the manual approach and the base and advanced models, we train and evaluate two more models:

- **Single Advanced** is the name we give to the model that is trained on one month of historical data, just like the advanced model. However, it is trained using single-granularity data. The encoder level granularity is monthly, just like the decoder level granularity.
- **Single Extended Advanced** is the name we give to the single-granularity version of the advanced model that is trained on six months of monthly historical data.

The single advanced model shows us what the added benefit is of using the multi-granularity dataset. The advanced model is only trained using a single month of historical data as even more

[3] This is slightly different from what is described for the base model. These requirements changed during the development of the advanced model.

data would lead to substantial computational requirements. For the single advanced model, the size of the data is not a problem. Therefore we also use the single extended advanced model, which is allowed to use more data without leading to substantial computational requirements. This is a more realistic scenario imitating how a practitioner would design such a model. We use the median of the advanced model predictions ($\rho = 0.5$) for comparison with the base model and manual process using the point forecast metrics. In Figure 6.4, we show the ***Mean Bias Error*** (MBE) and the MAE between the predictions and the observed performance for the residential and commercial models. In Table 6.2 we display the number of meters that were dispatched each month.

First, we look at the commercial model, for which the results are shown in Figure 6.4b. While the manual process and the base model usually have a positive bias, the advanced models have the opposite and show negative biases. When we look at the MAE, we see that some of the time, the advanced model is better than the base model, but some of the time, it is the other way around. We should note that the base model does perform better than the advanced model for the warmer months. Finally, we see that the advanced model almost always outperforms its single-granularity equivalents. The results for the residential model can be seen in Figure 6.4a. In terms of MAE, the different models all perform very similarly. Interestingly, the regular advanced model performs the worst here. This is also reflected in the bias, which is much higher for the advanced model.

In this error analysis, we also had a look at the RMSE. The RMSE aligned for the most part with the findings in the MAE. The main difference is that the advanced models relatively show a higher RMSE than the base model. This is expected as these models are optimized for the MAE instead, while the base model is optimized for the RMSE.

Meter Type	Year-Month	# Meters	0.00 Quantile	0.33 Quantile	0.66 Quantile	1.00 Quantile
Commercial	2021-01	164	17.27	335.52	712.00	39169.20
	2021-02	536	0.00	73.60	301.92	39169.20
	2021-03	121	0.00	85.37	227.36	2578.80
	2021-04	175	17.27	297.12	703.20	39169.20
	2021-05	482	0.00	106.80	338.88	39169.20
	2021-06	690	0.00	66.66	286.56	39169.20
Residential	2021-06	5589	0.00	5.96	8.27	52.00

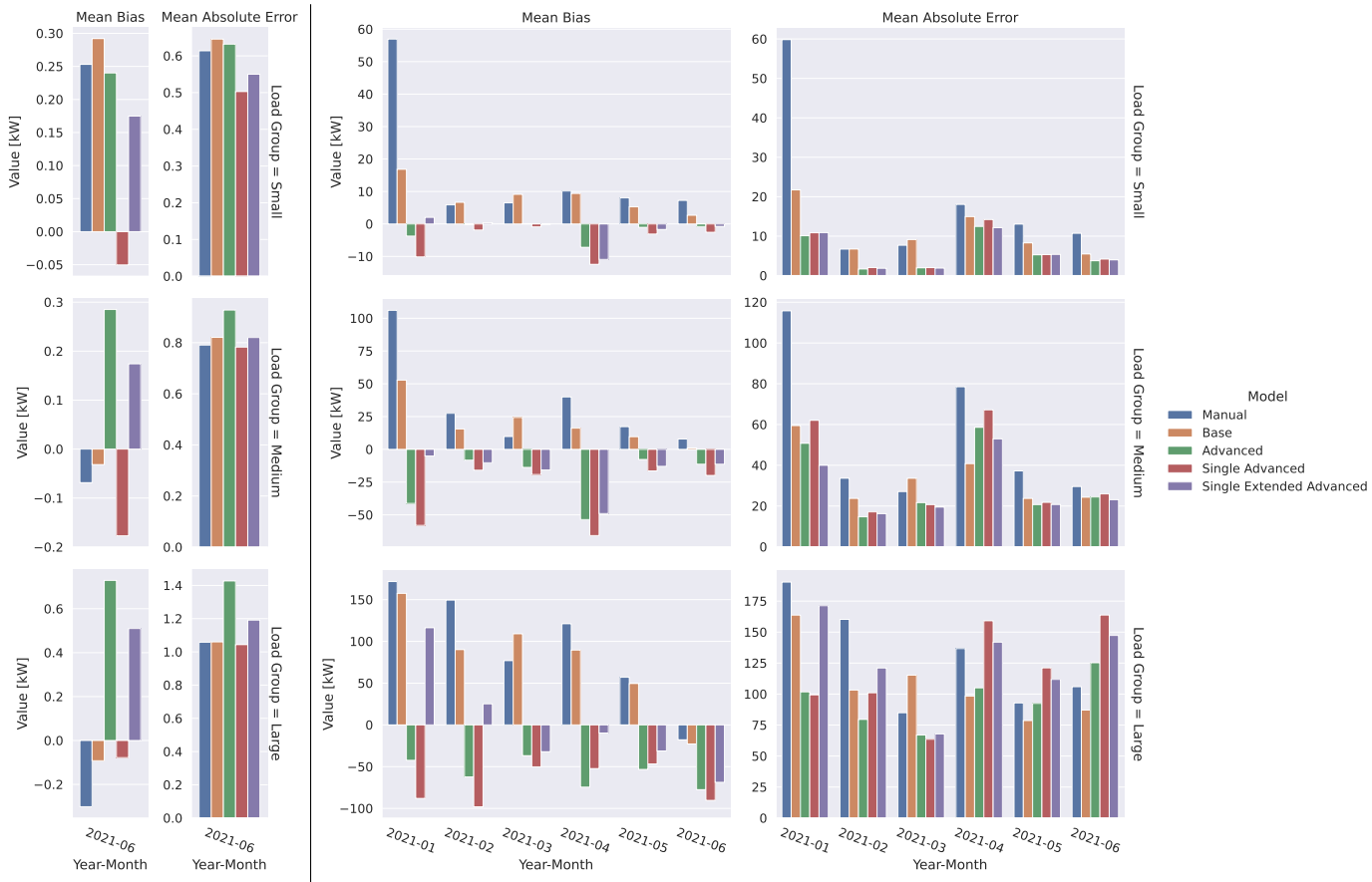
TABLE 6.2: For each month, we show the number of meters that got dispatched and some information regarding the size of the meters as the average historical load. This is used to separate the meters into three equally sized load groups.

Scale

In the error analysis of the base model, we discovered patterns in the accuracy metrics by separating the meters by the size of the meters. Additionally, larger meters have a larger impact on the accuracy metrics. Therefore, we separate the meter into three equally sized groups by average meter load. Table 6.2 provides some details regarding the boundaries of these groups in terms of the average historical load of the meters in that group. Figure 6.6 displays this analysis for the previously mentioned models. What stands out is that the advanced models seem to perform better the smaller the commercial meters are. The advanced model has a MAE of approximately 50% of the manual process for the smallest meters. This makes it even more surprising that the advanced models do not perform as good on residential meters in general, even though these are all small meters. Still, also for the residential meters, it seems to be the case that the advanced models perform better when the meters become smaller.

Cold-start

One of the challenges in our data is that we want to forecast performance for meters that have no historical performance yet. Those meters that do not have historical performance are called cold-start meters, while those that do have historical performance are called hot-start meters. A comparison of the models on the accuracy metrics by first separating the meters by cold- or hot-start is displayed in Figure 6.5. The commercial meters saw about 75 cold-starts in February, May and June. The residential meters saw about 1500 cold-starts in June. For the residential meters, the



(A) The residential meters. About 5500 got dispatched in June.

(B) The commercial meters got dispatched in each month of 2021, with varying numbers of dispatched meters per month. In January, March and April, about 150 meters got dispatched. In the remaining months, about 600 meters got dispatched.

FIGURE 6.5: A comparison of several metrics for the first six months of 2021 between the manual process, the base model and the advanced models. We calculate these metrics by first separating the meters into three equally sized groups per month by the average historical load.

difference in the accuracy metrics between hot-start and cold-start meters is insignificant. On the other hand, the accuracy metrics for cold-start commercial meters are very different from those for hot-start commercial meters. [4] We observe a positive bias for the advanced models that are much lower than the bias of the manual process and the base model show. Additionally, we also see that the advanced models were much more accurate in terms of MAE.

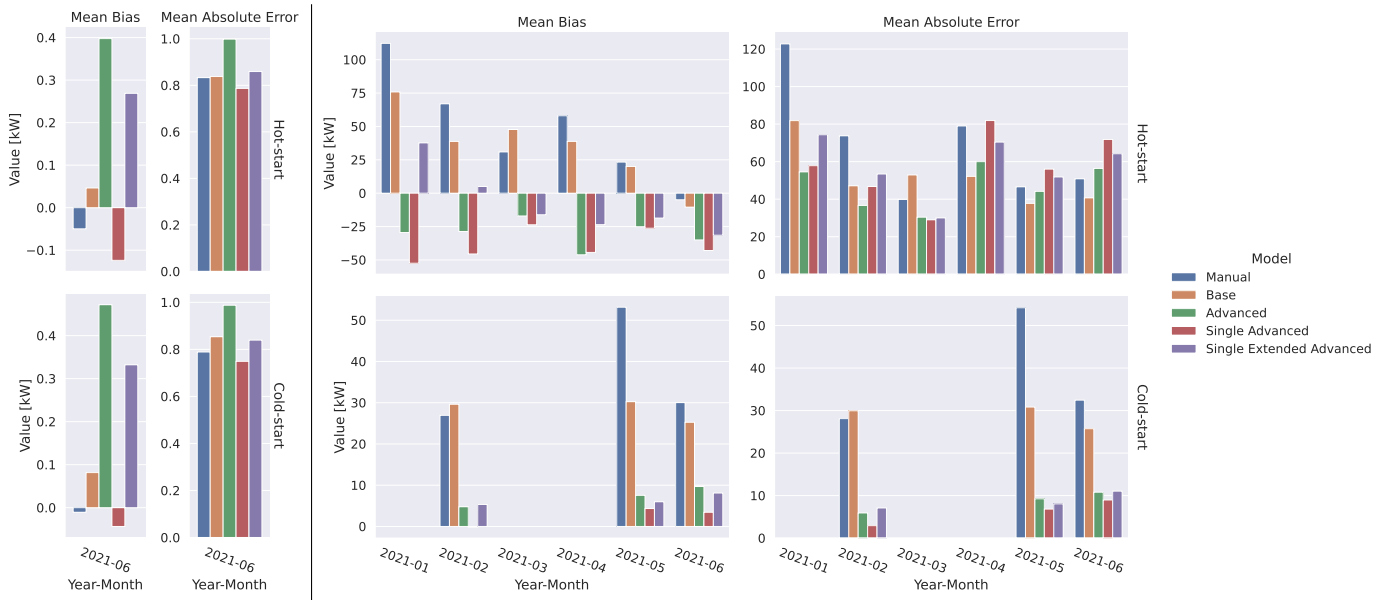
Quantile	0.02	0.10	0.25	0.50	0.75	0.90	0.98
Base Model	0.02	0.09	0.43	0.77	0.92	0.97	0.99
Advanced Model	0.02	0.08	0.04	0.48	0.59	0.48	0.93

TABLE 6.3: The overprediction rate by quantile.

Uncertainty

Part of the motivation for developing the base model was to serve as a comparison for the uncertainty of the advanced model predictions. In Section 2.4.3 we define a version of the ρ-risk as the

[4] We do not refer to the scale of the metrics as there could be many explanations for that. For example, the hot-start meters could have been larger in terms of historical load. Therefore, we instead focus on the differences in accuracy between the models for each category.

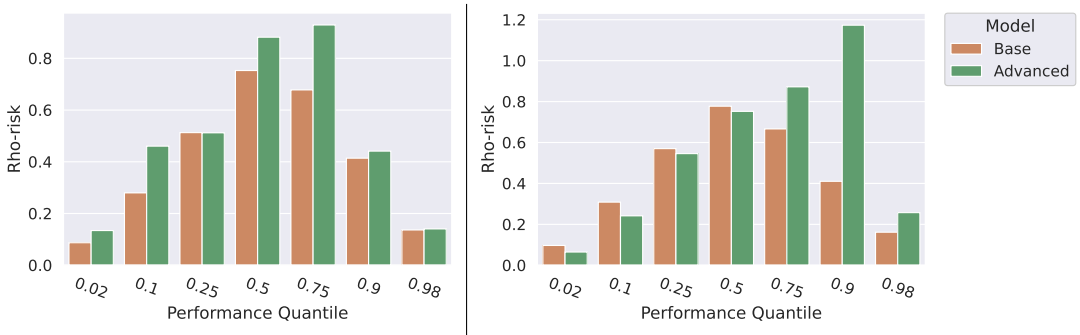


(A) The residential meters. About 5500 got dispatched in June.

(B) The commercial meters got dispatched in each month of 2021, with varying numbers of dispatched meters per month. In January, March and April, about 150 meters got dispatched. In the remaining months, about 600 meters got dispatched.

FIGURE 6.6: A comparison of several metrics for the first six months of 2021 between the manual process, the base model and the advanced models. We calculate these metrics by first separating the meters by hot-start and cold-start.

probabilistic accuracy metric. The base model can only forecast parameterized normal distributions and is, therefore, more restricted than the advanced model. In Figure 6.7, we compare the ρ-risk metrics for each predicted quantile in the advanced model across all predictions. In the residential predictions, the base model is consistently better or equal to the advanced model's uncertainty predictions. For the commercial predictions, the advanced model is better for the lower quantiles, while the predictions for the higher quantiles are much worse than the base model. To get some more insights into these results, we show the overprediction rate of the models per quantile in Table 6.3. We observe that the advanced model's results are not monotonic increasing, which is one of the downsides of quantile predictions. It is also clear that the base model tends to overpredict more than it should, and the advanced model tends to underpredict more than it should.



(A) The ρ-risk of the quantile predictions for the models trained on the residential meters.

(B) The ρ-risk of the quantile predictions for the models trained on the commercial meters.

FIGURE 6.7: A comparison of the ρ-risk for several prediction quantiles between the base and advanced models.

6.4 Discussion

The base model and the advanced model are very different in terms of their inner workings and optimization. This is also reflected in the accuracy metrics and in the types of meters for which they excel. The base model works better for large meters, whereas the advanced model works better for small meters. Using a different scaling variable, it might be possible to **strike a good balance** resulting in a model that does not favour meters of a specific size during optimization. Alternatively, we could expand the hierarchical model such that different sized meters are forecasted using one or the other model. The base model usually shows a positive bias, and the advanced model usually shows a negative bias. These findings can be used in multiple ways to create a better model. By ***ensembling*** the base and advanced model, we can potentially cancel out their biases and create more accurate forecasts. These same improvements are not observed for the residential meters. There are many potential explanations for this; extended analysis could help us figure out the correct one.

The largest limitations in optimising the models and the wider research are in the shape and size of the data. We could further improve the models using ***data augmentation***, for example, by scaling up or down all of the time series related to a single meter with a certain percentage. This would reduce overfitting and improve the generalization capabilities of the model. Another limitation of the data is that we are not able to conduct a **complete evaluation** of the MGTSD. The MGTSD seems to be an improvement over the single-granularity approach in specific cases; however, it comes at the cost of a much increased computational complexity. In this research, we had minimal time to optimise the advanced model and have minimal data for its evaluation. Further research into this subject could set up an experiment in which the MGTSD approach is compared with the single-granularity equivalent using the datasets described by Lim et al. (2020).

One of the major challenges and goals of this research was providing predictions for cold-start scenarios. For cold-start commercial meters, the advanced models make huge improvements in accuracy over the manual process and base model. As the feature set of the advanced models does not include detailed historical meter performance, it is **not surprising that these models still perform** well if this data is absent. We are not sure why the manual process accuracy degrades this much when the historical performance is missing. We could augment the dataset used to train the base model by randomly removing the historical performance from the training data. This also exposes a potential improvement to the advanced model. In the current multi-granularity setup, it is not trivial to use two granularities for just the encoder. It is computationally expensive to feed in more historical data to provide more information about the historical performance of a meter in the multi-granularity setting. However, if we can achieve this, then it could make a big improvement to the advanced model.

Performance values below zero are rare, and outliers to high performances are more common. As the base model optimizes the RMSE, it is not surprising that it tends to **overpredict for most quantiles**. This is where quantile forecasting shines as we do not enforce any specific distribution. The downside of quantile forecasting is that we do not have any guarantees that, when we increase the quantile, the predictions do monotonically increase as well. The advanced model seems to suffer from this problem extensively, probably due to the limitations of the dataset. We could potentially relieve this by modifying the loss function so that it penalizes this behaviour. It would be useful to analyse and visualise the attention mechanism of the advanced model in further research to provide insight into the many questions raised about the performance of the complex advanced models.

The base and advanced models show **auspicious results**, each of which **excels in different areas**. The strong parts of both models can be leveraged and combined to provide better results. As shown in this discussion section, there are **plenty opportunities** to use and improve the models. The next chapter will dive into how Leap or the academic field can best move forward with the gained knowledge and raised questions.

Chapter 7

Discussion & Conclusion

In this chapter, we will reiterate the most important parts of the discussion sections in Chapter 5 and 6. Along with that, we provide a general perspective on the implications, limitations and recommendations. We can separate this research into three phases: the discovery and exploration of the data and literature, the base model development and the advanced model development. As a result, we provide the reader with an extended analysis of the considerations, possibilities, and limitations we encountered. Finally, we provide two fully elaborated approaches to solving automated meter performance forecasting. In that way, we answer the main research question:

> ***Research Question:*** *How can we forecast the performance of electricity consumers using data such as the historical load, historical performance and meter-specific characteristics to create accurate meter nominations?*

The problem and data at hand pose several challenges. Individually these challenges are encountered and tackled in related work, but they are not encountered as part of a single problem or project. We describe two methodologies towards tackling this problem, each of which has its own advantages and disadvantages. Contrary to approaches published in related work, we can:

1. Use sparse time series to train robust traditional machine learning models.
2. Use multi-granularity time series to train high potential deep-learning based models.

In developing the base model, we decided to approach this problem as a regression problem rather than a time series forecasting problem. We transform the time series data to a tabular format to work around the sparsity of the dataset and train traditional machine learning models. In developing the advanced model, we did approach the problem as a time series forecasting problem. However, we took a special interest in the multi-granular nature of the data. We modify and extend existing tools to preprocess multi-granular data and feed it to a state-of-the-art deep learning architecture.

In the general setting, these models provide competitive performance forecasts using a completely **automated process**. Additionally, the models **surpass the manual process** in terms of performance by different margins on several subsets of the dataset. The base model consistently makes large improvements for large meters; this improvement is about 10-20% in terms of MAE. The advanced model consistently makes large improvements for small meters; this improvement is about 30-40% in terms of MAE. Additionally, we find that the advanced model creates very accurate predictions in cold-start scenarios compared to the manual process and the base model; this improvement is about 70-80% in terms of MAE. Currently, the manual process is only concerned with point forecast predictions. The approaches developed in this research expand on this by providing estimates of the uncertainty of these predictions.

7.1 Insights

As a result of conducting this research, we have gained extended knowledge of the data associated with this problem. Some of this knowledge has immediately been applied to the research. We discovered the large differences in meter characteristics and data availability between residential and commercial meters. The creation of the hierarchical model leveraged this knowledge. We also learned about the stochastic nature and uncertainty in the behaviour of electricity consumers and

the performance they can show. We tried to immediately leverage this knowledge by developing the models so that they could provide uncertainty estimates.

Additionally, we have found that even though there is much uncertainty in the performance, we can extract valuable patterns in an automated data-driven manner using machine learning. We develop two different approaches. On each of these, we approximately spend an equal amount of time. As a result, we have the opportunity to compare these two approaches. We learn that traditional machine learning models result in stable models with competitive accuracy but require more effort for preprocessing and feature engineering. On the other hand, deep learning models do automated feature extraction and can construct more complex features; however, they give no insight into their inner workings, which makes the stability of these models uncertain.

7.1.1 Usage

This research suggests two approaches for performance forecasting and provides insights into the underlying data. If used by Leap, it could save tremendous amounts of work currently conducted in a manual process and improve on this process simultaneously. To do so, Leap has to decide how they can integrate the model into existing business processes. Leap has to decide which prototypes they will continue developing to turn them into a full-fledged machine learning system/performance forecast provider.

From my gained knowledge and experience, it is my recommendation that Leap integrates any solution for performance forecasting in an **advisory role** into their existing business processes. Certain types of information or edge cases are challenging to include into the data-driven models proposed in this research. For example, it happens from time to time that a partner reaches out to the PartnerSuccess team to notify them of a special scenario, like maintenance on one of their meters. It is not feasible at this moment to enable the model to handle these kinds of potential edge cases. The performance forecasting model should suggest a nomination. The PartnerSuccess team then decides how they use this suggestion.

As stability is key and the need for an improved nomination forecasting model is urgent, I recommend that Leap start with integrating the base model. Several tasks are required for this model to become trustworthy and usable. First, Leap should continue the analysis to **pin point those meters** for which the model works exceptionally well. Using this information, they can develop a strategy to gradually roll out and integrate the model into the current business processes. Second, **data quality checks** on the inputs and outputs of this model need to be improved. As the current models are prototypes, they are not always able to create a forecast, for example, due to partially missing data. As part of the productionisation of the model, this will have to be fixed. However, Leap should be made aware in the future when a prediction cannot be made or is unreasonable. Next, **supporting infrastructure for continuous evaluation** has to be set up to evaluate the model over time. The time span of our training and testing dataset is relatively small. To determine the true quality of any model, we should evaluate it over a longer time span. Finally, as new data comes in, the model should be **re-trained frequently**. Having access to more data will allow leap to deliver more robust and more accurate models continuously. When these components are in place, Leap can decide to iteratively improve and test new models, like the advanced model.

7.1.2 Limitations

As mentioned multiple times before, the dataset is limited in time span. As a result, we get a minimal and one-sided view of the accuracy of the developed models. The presented results are insightful, but anyone should be sceptical about drawing conclusions from them. The continued evaluation suggested in the previous section should help relieve this limitation. Leap could also try to limit the sparsity of the dataset. Currently, meters are only dispatched in such a way that usefulness and revenue are maximized. Leap could opt to balance the exploitation and exploration to gather more data.

Due to the computational and time constraints, we trained the final models once and then used them for evaluation. A better approach, which can be used in continued work by Leap, is to train several instances of each model. Each of these instances is then used to make forecasts. As a result, we will be able to compute the average performance metrics and calculate the standard deviations. This will give us more information on the stability of the training procedure.

In this research, we have access to a dataset partially created and influenced by the strategy and decisions made at Leap. For example, the MarketOps team make decisions on the frequency of dispatches for a certain meter in a given month. This means that **the dataset reflects the current processes at Leap**. The models we train in this research try to detect and extrapolate patterns present in the dataset. If the MarketOps team decides to change their strategy drastically, then the dataset no longer reflects the strategy that we can expect in the future. As a result, the accuracy of the models will likely decrease. Over time, as the dataset will reflect this new strategy, the models could recover and start to learn the patterns related to the new strategy.

The point forecasts of the advanced model are optimized for the MAE, which is equivalent to optimizing for the median. As the overpredictions and underpredictions of this model are well balanced, it seems that the advanced is good at predicting the median. However, we do observe a negative bias for these models in commercial meters. This is likely explained by the fact that **positive outliers are more likely than negative outliers**, as shown in Section 4.4. This observation suggests that there is a positive skew in the performance labels. In a positively skewed distribution, the median is smaller than the mean. This likely results in the negative bias of the advanced model for commercial meters. The base model is optimized for the RMSE, which is more sensitive to outliers. Combined with the higher prevalence of positive outliers, this likely results in the positive bias of the base model for commercial meters.

This same bias is present when we look at the overprediction rate of the base and advanced model's uncertainty predictions. Also, when analysing the uncertainty estimate, we observe that the models often consider a vast range of future performance feasible. Therefore, Leap has to consider whether they deem the uncertainty predictions to be useful. Uncertainty predictions might become more useful in high aggregated loads, such as resources. Potentially, the uncertainty estimates can be used in the future for forecasting resource performance.

7.1.3 Reflection

We discovered several learnings and takeaways during the research. Naturally, if this knowledge would have been at my disposal at the start of this research, then I would have made many different decisions. However, these processes are at the core of doing research. There are, however, a few decisions that I made during the research that I consider to be mistakes for the sake of this research.

Leap is a very young company, and therefore a lot of infrastructure one would find at a larger company is simply not existing yet. At times I spent more time working on preprocessing infrastructure or writing scripts to run calculations in the cloud than would have been best for this research. Additionally, there is a certain temporal component to this research and this report. As Leap moves fast, requirements and interpretations of objectives have changed on the regular. This is the cause of some inconsistencies between the data analysis, the base model and the advanced model.

Using a BGA for feature selection was not very useful, and the choice to use it was not sufficiently substantiated by related work. We could have probably found the same results with some straightforward feature selection alternatives. The model was chosen mostly out of personal curiosity, not due to proven effectiveness in related work.

Developing a deep learning model is a time-consuming task. This research tried to develop a deep learning model by using a new technique on a new dataset. The result is that the amount of time spent on training and optimizing the model was very limited. I recommend any other students to either use a new dataset or a new technique if they find themselves in a similar situation, but not both at the same time. This will allow for more structured research with fewer unanswered questions.

7.2 Future Work

There are virtually unlimited ways to work on and improve the models proposed in this research. Earlier in this chapter, we already made a few suggestions. In this section, we would like to expand on these suggestions. We touch upon the most fruitful ideas based on my current knowledge and experience with the data.

Both models could benefit from the following improvements. In the advanced model, we first scale the performances by the historical load, while we do not do this in the base model. From

this difference, we observe that how we scale the data impacts how the model is optimized. We have observed that some meters have a minimal historical load, which results in exploding values resulting from division by this scaling factor. Future work could use a different scaling variable that is a bit more robust. The historically maximum two-hour consecutive baseline would be a good candidate. It is effectively the upper limit for the historical performance even if the meter never got dispatched. Using extensive **data augmentation** might also be very beneficial for both models. The following two approaches have high potential. First, we could scale the load, baseline and performance timeseries up or down by a certain random percentage. If the percentage is reasonably small, then we would limit overfitting while keeping the meter scale informative. Second, we could augment training samples by randomly masking the partner using an 'unknown' flag. This would allow the model to learn to forecast performance on meters from an entirely new partner.

Base model

Additionally, we can specifically improve the base model by expanding the hierarchical model. A suggestion would be to add new branches at the commercial meters leaf that has new leaves for groups of meters based on historical meter load. For each of these groups, we can train a separate model. As suggested before, it would likely be fruitful to do more analysis to determine for which meters the model performs best. The results of such an analysis can also be used to expand the hierarchical model.

Furthermore, it would be interesting to explore whether quantile forecasting using the base model is beneficial. There are some advantages of quantile regression as opposed to fitting a fixed distribution type. This would also result in the base model optimizing for the MAE instead of the RMSE.

Advanced model

Deep learning models take much time and computational resources to develop. It isn't easy to properly feed the historical meter performance into the model in the current implementation. Additionally, the optimization of the advanced model in this research was quite limited, especially in terms of feature selection. Optimizing the feature set, potentially by including more information on historical meter performance, could greatly improve the advanced model.

Academic research

The previous suggestions are interesting both for Leap and the academic research field. Apart from those, there is one suggestion that is especially important for academic research. For the advanced model, we develop a new technique for a new dataset. This makes it difficult to determine how the new technique would perform in more general scenarios. Further research could analyse the added value of the MGTSD by training the model on datasets described in the related work. These datasets provide a lot of data on a long time span, making them great candidates for setting up an experiment to evaluate the MGTSD.

7.3 Conclusion

The growing effects of climate change and the introduction of cheap renewable energy bring transformational changes to electricity grids worldwide. Long term storage of electricity is often not feasible, and therefore many grids resort to demand response programs to cover peak loads. Demand response providers are required to provide performance forecasts as part of resource adequacy programs. As Leap brings demand response to many small electricity consumers, they are tasked with difficult, labour-intensive performance forecasting.

Leap has access to a large amount of oddly shaped data which is also used to calculate historical performance. This research developed two approaches to forecast meter performance using this sparse multi-granularity time series data. The base model approach transformed the time series data to a tabular format and tackled the problem like a regression problem. This enabled us to develop a robust data-driven model based on traditional machine learning techniques with competitive accuracy. The advanced model approach leveraged the multi-granular nature of the

dataset to work around the sparsity of the time series data. This enabled us to develop a state-of-the-art model with the potential for highly accurate performance forecasting.

In the end, we deliver insights into the data and prototypes for performance forecasting to Leap. Leap will use these findings to reduce the labour intensity of the process and improve the accuracy of the performance forecasts. To do so, Leap will set up supporting infrastructure and continue developing the models to improve them further. Additionally, we present empirical evidence for the feasibility of multi-granular time series forecasting. There are many scenarios in which this technique has the potential to improve on commonly used alternatives. However, an extended evaluation has to be conducted to solidify the feasibility of multi-granularity time series forecasting.

References

Abeywardana, S. (2017, Oct). *Sequence to sequence tutorial.* Towards Data Science. Retrieved from https://towardsdatascience.com/sequence-to-sequence-tutorial-4fde3ee798d8

Babatunde, O., Armstrong, L., Leng, J., & Diepeveen, D. (2014, January). A Genetic Algorithm-Based Feature Selection. *ECU Publications Post 2013.* Retrieved from https://ro.ecu.edu.au/ecuworkspost2013/653

Blumsack, S. (2017). *Introduction to electricity markets.* Pennsylvania State University. Retrieved from https://www.e-education.psu.edu/ebf483/

Blumsack, S. (2021). *Demand response in electricity markets.* Pennsylvania State University. Retrieved from https://www.e-education.psu.edu/eme801/node/542

Bottou, L., & Lin, C.-J. (2007). *Support Vector Machine Solvers Support Vector Machine Solvers.*

Box, G., Jenkins, G., & Reinsel, G. (2008). Time Series Analysis. In *Time Series Analysis* (pp. i–xxiv). John Wiley & Sons, Ltd. Retrieved 2021-02-09, from http://onlinelibrary.wiley.com/doi/abs/10.1002/9781118619193.fmatter (_eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118619193.fmatter) doi: 10.1002/9781118619193.fmatter

Breiman, L. (2001, October). Random Forests. *Machine Language*, *45*(1), 5–32. Retrieved 2021-06-13, from https://doi.org/10.1023/A:1010933404324 doi: 10.1023/A:1010933404324

Che, Z., Purushotham, S., Cho, K., Sontag, D., & Liu, Y. (2018, April). Recurrent Neural Networks for Multivariate Time Series with Missing Values. *Scientific Reports*, *8*(1), 6085. Retrieved 2021-02-08, from https://www.nature.com/articles/s41598-018-24271-9 (Number: 1 Publisher: Nature Publishing Group) doi: 10.1038/s41598-018-24271-9

Chen, T., & Guestrin, C. (2016, August). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. Retrieved 2021-06-13, from http://arxiv.org/abs/1603.02754 (arXiv: 1603.02754) doi: 10.1145/2939672.2939785

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014, October). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1724–1734). Doha, Qatar: Association for Computational Linguistics. Retrieved 2021-03-14, from https://www.aclweb.org/anthology/D14-1179 doi: 10.3115/v1/D14-1179

Cortes, C., & Vapnik, V. (1995, September). Support-Vector Networks. *Machine Learning*, *20*(3), 273–297. Retrieved 2021-02-11, from https://doi.org/10.1023/A:1022627411411 doi: 10.1023/A:1022627411411

Das, U. K., Tey, K. S., Seyedmahmoudian, M., Mekhilef, S., Idris, M. Y. I., Van Deventer, W., … Stojcevski, A. (2018, January). Forecasting of photovoltaic power generation and model optimization: A review. *Renewable and Sustainable Energy Reviews*, *81*, 912–928. Retrieved 2020-12-10, from http://www.sciencedirect.com/science/article/pii/S1364032117311620 doi: 10.1016/j.rser.2017.08.017

Ding, Y., & Simonoff, J. S. (2010). An Investigation of Missing Data Methods for Classification Trees Applied to Binary Response Data. , 40.

Duan, T., Avati, A., Ding, D. Y., Thai, K. K., Basu, S., Ng, A. Y., & Schuler, A. (2020, June). NGBoost: Natural Gradient Boosting for Probabilistic Prediction. *arXiv:1910.03225 [cs, stat]*. Retrieved 2021-03-23, from http://arxiv.org/abs/1910.03225 (arXiv: 1910.03225)

Eseye, A., & Lehtonen, M. (2020). Short-Term Forecasting of Heat Demand of Buildings for Efficient and Optimal Energy Management Based on Integrated Machine Learning Models. *IEEE Transactions on Industrial Informatics*, *16*(12), 7743–7755. doi: 10.1109/TII.2020.2970165

Fan, H., MacGill, I. F., & Sproul, A. B. (2015, October). Statistical analysis of driving factors of

residential energy demand in the greater Sydney region, Australia. *Energy and Buildings*, *105*, 9–25. Retrieved 2021-02-02, from http://www.sciencedirect.com/science/article/pii/S0378778815301419 doi: 10.1016/j.enbuild.2015.07.030

Garulli, A., Paoletti, S., & Vicino, A. (2015, May). Models and Techniques for Electric Load Forecasting in the Presence of Demand Response. *IEEE Transactions on Control Systems Technology*, *23*(3), 1087–1097. (Conference Name: IEEE Transactions on Control Systems Technology) doi: 10.1109/TCST.2014.2361807

Hernández, L., Baladrón, C., Aguiar, J. M., Calavia, L., Carro, B., Sánchez-Esguevillas, A., … Lloret, J. (2014, March). Artificial Neural Network for Short-Term Load Forecasting in Distribution Systems. *Energies*, *7*(3), 1576–1598. Retrieved 2021-02-02, from https://www.mdpi.com/1996-1073/7/3/1576 (Number: 3 Publisher: Multidisciplinary Digital Publishing Institute) doi: 10.3390/en7031576

Hernández, L., Baladrón, C., Aguiar, J. M., Carro, B., Sánchez-Esguevillas, A., & Lloret, J. (2014, October). Artificial neural networks for short-term load forecasting in microgrids environment. *Energy*, *75*, 252–264. Retrieved 2021-02-02, from http://www.sciencedirect.com/science/article/pii/S0360544214008871 doi: 10.1016/j.energy.2014.07.065

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Hsiao, Y. (2015, February). Household Electricity Demand Forecast Based on Context Information and User Daily Schedule Analysis From Meter Data. *IEEE Transactions on Industrial Informatics*, *11*(1), 33–43. (Conference Name: IEEE Transactions on Industrial Informatics) doi: 10.1109/TII.2014.2363584

Jha, A., Ray, S., Seaman, B., & Dhillon, I. S. (2015, April). Clustering to forecast sparse time-series data. In *2015 IEEE 31st International Conference on Data Engineering* (pp. 1388–1399). (ISSN: 2375-026X) doi: 10.1109/ICDE.2015.7113385

Kuster, C., Rezgui, Y., & Mourshed, M. (2017, November). Electrical load forecasting models: A critical systematic review. *Sustainable Cities and Society*, *35*, 257–270. Retrieved 2020-07-08, from http://www.sciencedirect.com/science/article/pii/S2210670717305899 doi: 10.1016/j.scs.2017.08.009

Lim, B., Arik, S. O., Loeff, N., & Pfister, T. (2020, September). Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting. *arXiv:1912.09363 [cs, stat]*. Retrieved 2021-06-07, from http://arxiv.org/abs/1912.09363 (arXiv: 1912.09363)

Lim, B., & Zohren, S. (2021, April). Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *379*(2194), 20200209. Retrieved 2021-03-14, from https://royalsocietypublishing-org.ezproxy2.utwente.nl/doi/10.1098/rsta.2020.0209 (Publisher: Royal Society) doi: 10.1098/rsta.2020.0209

Lu, J., Zhao, T., & Zhang, Y. (2008, December). Feature selection based-on genetic algorithm for image annotation. *Knowledge-Based Systems*, *21*(8), 887–891. Retrieved 2021-05-26, from https://www.sciencedirect.com/science/article/pii/S095070510800097X doi: 10.1016/j.knosys.2008.03.051

Lu, Z., Pu, H., Wang, F., Hu, Z., & Wang, L. (2017, November). The Expressive Power of Neural Networks: A View from the Width. *arXiv:1709.02540 [cs]*. Retrieved 2021-02-11, from http://arxiv.org/abs/1709.02540 (arXiv: 1709.02540)

Mamun, A. A., Sohel, M., Mohammad, N., Sunny, M. S. H., Dipta, D. R., & Hossain, E. (2020). A Comprehensive Review of the Load Forecasting Techniques Using Single and Hybrid Predictive Models. *IEEE Access*, *8*, 134911–134939. (Conference Name: IEEE Access) doi: 10.1109/ACCESS.2020.3010702

Mathieu, J. L., Price, P. N., Kiliccote, S., & Piette, M. A. (2011, September). Quantifying Changes in Building Electricity Use, With Application to Demand Response. *IEEE Transactions on Smart Grid*, *2*(3), 507–518. (Conference Name: IEEE Transactions on Smart Grid) doi: 10.1109/TSG.2011.2145010

Mena, R., Rodríguez, F., Castilla, M., & Arahal, M. R. (2014, October). A prediction model based on neural networks for the energy consumption of a bioclimatic building. *Energy and Buildings*, *82*, 142–155. Retrieved 2021-02-02, from http://www.sciencedirect.com/science/article/pii/S0378778814005349 doi: 10.1016/j.enbuild.2014.06.052

Olah, C. (2015, Aug). *Understanding lstm networks*. Retrieved from https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Salinas, D., Flunkert, V., & Gasthaus, J. (2019, February). DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks. *arXiv:1704.04110 [cs, stat]*. Retrieved 2021-02-26, from http://arxiv.org/abs/1704.04110 (arXiv: 1704.04110)

Sato, K., & Azuma, S. (2016, October). Controllability of Aggregate Demand Response Systems for Real-Time Pricing. In *2016 IEEE 4th International Conference on Cyber-Physical Systems, Networks, and Applications (CPSNA)* (pp. 33–34). doi: 10.1109/CPSNA.2016.15

Shen, Z., Zhang, X., Xia, B., Liu, Z., & Li, Y. (2019, December). Multi-Granularity Power Prediction for Data Center Operations via Long Short-Term Memory Network. In *2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)* (pp. 194–201). Xiamen, China: IEEE. Retrieved 2021-01-11, from https://ieeexplore.ieee.org/document/9047259/ doi: 10.1109/ISPA-BDCloud-SustainCom-SocialCom48970.2019.00037

Sutskever, I., Vinyals, O., & Le, Q. V. (2014, December). Sequence to Sequence Learning with Neural Networks. *arXiv:1409.3215 [cs]*. Retrieved 2021-03-13, from http://arxiv.org/abs/1409.3215 (arXiv: 1409.3215)

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … Polosukhin, I. (2017, December). Attention Is All You Need. *arXiv:1706.03762 [cs]*. Retrieved 2021-08-07, from http://arxiv.org/abs/1706.03762 (arXiv: 1706.03762)

Wang, H., Lei, Z., Zhang, X., Zhou, B., & Peng, J. (2019, October). A review of deep learning for renewable energy forecasting. *Energy Conversion and Management*, *198*, 111799. Retrieved 2020-12-10, from http://www.sciencedirect.com/science/article/pii/S0196890419307812 doi: 10.1016/j.enconman.2019.111799

Wen, R., Torkkola, K., Narayanaswamy, B., & Madeka, D. (2018, June). A Multi-Horizon Quantile Recurrent Forecaster. *arXiv:1711.11053 [stat]*. Retrieved 2021-02-06, from http://arxiv.org/abs/1711.11053 (arXiv: 1711.11053)

Wright, L. (2019). *Ranger - a synergistic optimizer*. https://github.com/lessw2020/Ranger-Deep-Learning-Optimizer. GitHub.

Xie, C., Tank, A., Greaves-Tunnell, A., & Fox, E. (2018, August). A Unified Framework for Long Range and Cold Start Forecasting of Seasonal Profiles in Time Series. *arXiv:1710.08473 [cs, stat]*. Retrieved 2021-03-12, from http://arxiv.org/abs/1710.08473 (arXiv: 1710.08473)

Yang, R. (2019, Nov). *Omphalos, uber's parallel and language-extensible time series backtesting tool*. Retrieved from https://eng.uber.com/omphalos/

Yang, S., Yu, X., & Zhou, Y. (2020, June). LSTM and GRU Neural Network Performance Comparison Study: Taking Yelp Review Dataset as an Example. In *2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI)* (pp. 98–101). doi: 10.1109/IWECAI50956.2020.00027

Zhai, Y. (2009). *Time series forecasting competition among three sophisticated paradigms*. Retrieved 2021-02-10, from /paper/Time-series-forecasting-competition-among-three-Zhai/a2fdddd8942ef5c7cf58f7df897dd61b9c2546e3