Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Espen Hansen Høijord

Explainable AI (XAI) for grid loss forecasting

Master's thesis in Computer Science
Supervisor: Helge Langseth
June 2022

NTNU
Norwegian University of
Science and Technology

Espen Hansen Højjord

Explainable AI (XAI) for grid loss forecasting

Master's thesis in Computer Science
Supervisor: Helge Langseth
June 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Espen Hansen Høijord

Explainable AI (XAI) for grid loss forecasting

Master's thesis, June 2022
Supervisor: Helge Langseth

Artificial Intelligence Group
Department of Computer and Information Science
Faculty of Information Technology, Mathematics and Electrical Engineering

Abstract

In many businesses and industries, forecasting future values of time series data is an important part of the operations of the business. When the conditions are stable, focusing only on optimizing the prediction accuracy in order to operate most efficiently or making the most amount of money is reasonable. However, in times of changing conditions, the prediction accuracy can suddenly get worse, possibly with major financial and operational consequences. In those situations, having transparency or explanations of the predictions can be helpful to understand the new conditions or how to adapt.

This research is based on the problem of grid loss prediction for the Norwegian power producer TrønderEnergi Kraft, that each day need to report to the market operator how much energy that it is expecting to lose in the power grid each hour the next day. The goal of this research has been to investigate how to best explain machine learning based forecasting methods in order to help the domain experts to know if the predictions are to be trusted or not, and possibly identify what contributed to the bad forecasts.

The research began with a review of the state of the art within the field of explainable AI (XAI) in which promising methods for the problem were identified. Then, a selection of the methods were made based on a number of selection criteria, such as what types of explanations that are needed and the forecasting accuracy. Then, experiments were performed using the selected methods, and domain experts from TrønderEnergi evaluated how well the explanations produces by those methods fitted their need for explanations. In particular, the explanations of the predictions from the days with worst prediction accuracy were examined.

One hypothesis made during the work was that complex forecasting models that output interpretable information about its inner workings would be able to provide better explanations than separate explanation methods. The results show that using separate forecasting and explanation methods is the best approach for this forecasting problem. Specifically, the SHAP method provided explanations that domain experts at TrønderEnergi found useful.

Sammendrag

Å predikere fremtidige verdier av tidsseriedata er en viktig del av driften til mange bedrifter og virksomheter innen mange ulike sektorer. Når forholdene er stabile, kan man fokusere på å optimalisere prediksjonsmodellene og dermed oppnå mest mulig effektiv drift eller oppnå best mulig finansielle resultater. Dersom forholdene endres, kan prediksjonene plutselig bli mye verre, og potensielt få store finansielle eller operasjonelle konsekvenser. I slike situasjoner kan det være avgjørende å ha innsikt i hvordan prediksjonsmodellene fungerer og hvordan de kan forbedres. For maskinlæringsmodeller som ikke er transparente, kan dette oppnås ved hjelp av forklaringsmodeller.

Dette prosjektet er basert på kraftprodusenten TrønderEnergis jobb med prediksjon av nettap, som er tap av elektrisk energi i distribusjonsnettverket. Hver dag kl 12 må TrønderEnergi melde inn hvor mye energi som de forventer vil gå tapt i distribusjonsnettet for hver time den neste dagen. Dette prosjektet har tatt utgangspunkt i dette prediksjonsproblemet og tilhørende datasett, og målet har vært å utforske hvordan best forklare prediksjoner til maskinlæringsmodeller, som kan hjelpe domeneeksperter til å få innsikt i hvordan prediksjonsmodellene har brukt dataene og eventuelt hva som har bidratt til dårlige prediksjoner. Dette kan bidra til å fortelle når man kan ha tillit til prediksjonene og når man må finne alternativer.

Arbeidet startet med en gjennomgang av literaturen innen feltet forklarbar kunstig intelligens, forkortet XAI, som resulterte i en beskrivelse av state-of-the-art og identifisering av aktuelle metoder som kunne passe til problemet. Deretter ble noen av disse metodene valgt ut basert på noen kriterier for videre uttesting, som hva slags forklaringer de kunne produsere. Eksperimenter ble utført med utvalgte metoder på nettap-datasettet til TrønderEnergi, og resultatene ble gjennomgått og evaluert sammen med domeneeksperter hos TrønderEnergi. Spesielt forklaringene for dagene med dårligst prediksjoner ble studert.

En hypotese underveis i arbeidet var at komplekse prediksjonsmodeller som gir ut tolkbar informasjon om indre parametere kunne gi bedre forklaringer enn separate forklaringsmodeller som ikke har noe informasjon om det indre i en prediksjonsmodell. Et eksempel på slik indre informasjon kan være vekter i et nevralt nettverk. Resultatene i dette arbeidet har vist det motsatte, at den beste fremgangsmåten for et slikt prediksjonsproblem er å ha separate prediksjons- og forklaringsmodeller. Da kan man optimalisere prediksjonsmodellen til å gi best mulig prediksjoner, og forklaringsmodellen kan gi forklaringer. Spesifikt ga metoden SHAP gode forklaringer som virket nyttige for domeneekspertene.

Preface

This thesis is the final work of the master's program in Computer Science at the Department of Computer and Information Science at NTNU.

I would like to thank my supervisor from NTNU, Professor Helge Langseth, for his valuable guidance throughout the whole project. I would also like to thank TrønderEnergi for the opportunity to work with this topic and dataset, and for their helpful feedback and guidance. In particular, Odd Erik Gundersen for his guidance throughout the project, and Nisha Dalal and Nina Økstad for their feedback and domain expertise towards the end of the project, that helped set the final direction for the project and evaluate the results.

Espen Hansen Høijord
Trondheim, June 13, 2022

Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence.

CNN Convolutional Neural Network.

DeepLIFT Deep Learning Important FeaTures.

Grad-CAM Gradient-weighted Class Activation Mapping.

ICE Individual Conditional Expectation.

LIME Local Interpretable Model-agnostic Explanations.

MAPE Mean Absolute Percentage Error.

PDP Partial Dependency Plots.

RMSE Root Mean Squared Error.

RNN Recurrent Neural Network.

SHAP Shapley Additive Explanations.

TFT Temporal Fusion Transformer.

XAI Explainable Artificial Intelligence.

Chapter 1

Introduction

This chapter introduces the background for the field of Explainable Artificial Intelligence (XAI), and the motivation and problem description of this particular project. It also introduces the goals for the project and its division into several research question, and how research methods are used to provide answers to the research questions. Finally, the contributions of the thesis is described and the structure of the rest of the thesis is outlined.

1.1 Background

The use of Artificial Intelligence (AI) and machine learning models are increasing at a rapid speed every year, and are being applied in ever more sectors of society, and forecasts indicate that the growth will continue [Bus, 2019]. Along with the increasing spread of applications of machine learning systems, the machine learning models used are also increasing in complexity. Where models of the previous decades included statistical and classical machine learning methods, now deep learning and ensemble methods based on big data sets dominate. This is made available by faster and better hardware and cloud computing, and is able to deliver more accurate predictions and insight from larger amounts of data than traditional methods. As the application of these machine learning models enter more regulated and critical sectors, there has been a growing demand for more transparency and justifications to how and why the models reach the predictions they do, in order to increase trust, while retaining the benefits of improved accuracy. An example of a project that address that demand is the XAI program by DARPA, the research and development agency of the United States Department of Defence, launched in May 2017 [Gunning and Aha, 2019]. The goal of this program is to create AI systems that can be understood and appropriately trusted

by end users, through the use of effective explanations. To avoid loosing prediction accuracy, methods for creating explanations are developed to not interfere with predictions, either by being applied on top of the prediction models, or as integral parts of the prediction models in which internal information is output together with predictions.

The work on creating methods that are able to create explanations of predictions by machine learning models has been developed in the field that is now known under the name of Explainable Artificial Intelligence (XAI). Common examples of applications within the field of XAI include highlighting which parts of an image that are most important for classifying which object is in the image, or which values of certain variables are needed to get a loan application accepted, for instance income or number of kids. While there has been published much work on XAI on data such as image, speech, text and tabular data, there is comparatively little work published regarding time series data, and time series forecasting tasks in particular. Nevertheless, there has lately been developed and published multiple methods that has the potential to provide meaningful insight into forecasting predictions.

1.2 Problem and Motivation

Forecasting time series data for future time steps is an important task in multiple fields, including weather forecasting and forecasting of sales for a business. Improving the forecasting accuracy by a small amount can have a large impact on a business, both financially and operational, meaning that production of products can be optimized to meet the expected demand, or that prices can be tailored to match what the market is willing to spend. Although accuracy is important for such forecasting tasks, the ability for a forecasting model to give insight into how the model has reached its forecasts can also be of high importance, especially if wrong predictions can have large consequences for the business or entity, financially or otherwise.

This thesis uses grid loss data from TrønderEnergi Kraft, a Norwegian power producer, as a case study to gain insight into the possibilities for explaining forecasts built on machine learning and neural networks. The accuracy of forecasts has an economic consequence for TrønderEnergi Kraft, which has led them to focus their attention on developing the most accurate predictions possible. When conditions are stable, optimizing the prediction accuracy is rational to minimize economic consequences of prediction errors. However, when conditions change, the predictions might suddenly get worse. In such situations, it would be helpful to get insight into how the predictions are made, in order to identify if the predictions can be trusted, and possibly how the predictions could be improved.

The work in this thesis is not intended to result in a method or a system that

can be directly applied by TrønderEnergi in their operations, but rather as an investigation of the possibilities of generating explanations of time series predictions, that can serve as a starting point for further exploration by TrønderEnergi. This thesis can be used as input to decide if this is a field that they see potential in and that they want to study further in order to improve their operations. The work can also help to identify other approaches or problem descriptions that could be helpful to study further.

1.3 Goals and Research Questions

This section describes the main goal of this project, and three research questions that each need to be answered to reach the goal. The research questions defines the direction of the project, and makes a step-wise plan for how to answer the goal. The main goal of this research is stated below.

Goal *Find the best approach to generate explanations of time series predictions given by black box machine learning models to a domain expert, to enable the domain expert to understand what influenced the predictions and if the predictions can be trusted or not.*

This formulation of the goal includes some assumptions that should be addressed. First, there are multiple approaches to generate explanations, and the goal is to find out which approach best fits the need of domain experts at TrønderEnergi. Second, the predictions are assumed to be made by black box machine learning models, meaning that the inner workings of the models are hidden such that it is unclear what influence the predictions.

In order for a prediction to be trusted by a domain expert, the domain expert needs enough information about the prediction to see that it is reasonable given the characteristics of the input data. This goal can be broken down into multiple research questions that each need to be addressed in order to achieve the goal.

Research question 1 *What is the state-of-the-art within Explainable Artificial Intelligence (XAI) in general, and for time series forecasting tasks specifically?*

This research question involves conducting a literature review of the field of XAI and specifically within time series forecasting. The end product of this literature review is an overview of the field and a description of important methods within the state-of-the-art. In order for the review to provide a basis for further investigation, it must describe methods with different approaches, such that it is possible to decide which method or methods are appropriate for the specific use case. Chapter 3 is an answer to this research question with a description of the directions within XAI, along with detailed descriptions of important methods.

Research question 2 *What constitutes a good explanation, and how can explanations be evaluated?*

In order to evaluate the explanations of black box machine learning predictions, evaluation criteria or metrics must be established to provide measures of the quality of explanations. It could be either qualitative evaluation criteria or quantitative metrics. The basis for this project is that explanations should give domain experts information such that they can decide if the predictions are to be trusted or not, which in itself is a subjective decision. However, there can be evaluation criteria that can help in this regard. The theory behind explanations and evaluation is in Chapter 2, whereas the specific evaluation methods used in this work is described in Chapter 4.

Research question 3 *Which explainability approach can best answer the need of domain experts to explain forecasting predictions?*

Finding which approach that works best for providing explanations for predictions answers the main goal of this project. Important factors are what type of information is provided in the explanations and whether the methods for giving explanations can be applied while at the same time producing accurate predictions. The main focus is on whether methods that provide explanations together with forecasts or separate forecasting and explanation methods are the best approach to this problem. Candidate methods and approaches are identified in the state-of-the-art in Chapter 3 and selected for testing in Chapter 4, and Chapter 5 and Chapter 6 provides experiments and evaluations of the selected methods in order to decide which approach works best for the domain experts of this particular problem.

1.4 Research Methods

The research in this thesis can be divided into two parts. The first part is a literature review of the state-of-the-art within XAI. The second part is experiments conducted on selected methods to decide which approach best fit the needs of the domain experts at TrønderEnergi.

The first part of this project was based on performing a literature review to examine the state-of-the-art within XAI, both generally and within time series forecasting. This was performed using the snowballing technique, where the reference list of selected papers are examined to find further important papers within the field. Also the citations of the methods were examined to check if the methods had been improved or to see results of applying the methods. Papers were initially found using search in Google Scholar, with multiple combinations of

keywords such as "xai", "time series forecasting", "interpretable", "explainable", and "explainable ai", and through survey articles.

The reason for this choice of strategy for literature review is that with this method, one gets most of the relevant papers within the field, either through the queries, the survey articles, or through references and citations. In addition, conducting a full Structured Literature Review is time consuming, and would limit the amount of time available for the second part of the project.

The second part of this project was conducting experiments on methods selected on the basis of the state-of-the-art to provide experimental evidence towards answering the research questions. The evaluations of methods in this project are based on the evaluation from domain experts from TrønderEnergi. The goal of this research is to find the best approach to creating explanations for this particular use case, and in this part promising approaches are selected and experimentally tested to get evidence towards this goal.

1.5 Contributions

The contributions of this thesis is two-fold. First, it provides a review of the field of XAI and its applications within time series forecasting. Second, it provides experimental evidence and evaluation by domain experts of selected methods of explaining forecasts, to find which approach is the best to provide explanations of forecasting methods. The best approach is to use separate forecasting and explanation methods, and SHAP provides useful explanations. Lastly, some potential future research directions are discussed in Section 7.3.

1.6 Thesis Structure

The remainder of this thesis is structures as follows. Chapter 2 introduces background theory important for understanding the rest of the thesis, including definitions of terminology, how explanations work for humans, how to evaluate explanations, and defines the problem of multi-step time series forecasting. Chapter 3 outlines the state-of-the-art within XAI in general and for time series forecasting, including descriptions of some of the important methods within the field. Then, Chapter 4 describes the methods that is selected based on the state-of-the-art with reasons for why they were chosen based on selection criteria. Chapter 5 describes the forecasting problem and associated dataset that is the use case in this thesis, and the experimental plan for how the methods will be tested and evaluated. Chapter 6 describes the results from the experiments along with evaluation of each of the experiments separately. Finally, in Chapter 7, the results are

evaluated and discussed, a conclusion is drawn, and possible further directions of research is presented.

Chapter 2

Background Theory

The work in this project is in the area of Explainable Artificial Intelligence (XAI). This chapter begins with definitions of important terms within XAI. Then an overview of findings from cognitive science and psychology about explanations for humans is presented, which can help decide which properties that are important when creating explanations from machine learning models. Then, the problem of multi-step time series forcasting is formally defined. Finally, an overview of how to evaluate explanations from XAI methods is presented.

2.1 Terminology

In their comprehensive XAI literature review, Barredo Arrieta et al. [2020] argues that there has not been consistency within the field of XAI in the use of terminology. In particular, interchangable use of the terms interpretability and explainability are used.

2.1.1 Explainability and interpretability

According to Barredo Arrieta et al. [2020], interpretability refers to a passive characteristic of a model, the level at which a given model makes sense for a human. Explainability is an active characteristic of a model, denoting any action or procedure taken by a model with the intent of clarifying or detailing its internal functionality. There is a distinction between interpretable models that can be understood by themselves, and black-box models that need explanation models to explain the functioning of the black-box model. In this thesis, prediction models which output information about its inner workings will be refered to as interpretable or intrinsically interpretable, following the terminology that stated

above. Methods that are applied to a previously trained prediction model for creating explanations without having access to internal information about the inner workings of the prediction model, will be referred to as explainability or explanation methods.

Barredo Arrieta et al. [2020] divides the methods within XAI into transparent models or post-hoc explainability models, which is described in detail in Section 3.1. Post-hoc means that the method for generating explanations is applied on a previously trained prediction model, without access to internal information. Post-hoc explainability models is thus the same as explainability methods defined above, only that the term post-hoc is underlining that the model is applied on a previously trained model.

2.1.2 Explainable Artificial Intelligence (XAI)

Barredo Arrieta et al. [2020] define XAI in the following way: "Given an audience, an Explainable Artificial Intelligence is one that produces details or reasons to make its functioning clear or easy to understand." As the name suggests, an XAI method should output information that can be used to understand predictions. This definition also emphasize the audience, suggesting that explanations should answer the need for the specific audience receiving the explanations. That indicates that explanations should be tailored to the receiver of the explanations. Whether an explanation is clear or easy to understand depends on the knowledge of the receiver, described further in Section 2.2.1.

2.2 Explanations in social sciences

In order to get a sense of what explanations can contribute to and what should be taken into consideration when creating a system for generating explanations, it might be helpful to study how humans use explanations. The purpose is not to copy how humans create explanations, but to keep the principles in mind when creating and evaluating explanations from prediction models.

Miller [2019] provides a review of how humans define, generate, select, evaluate, and present explanations. The article reviews research within the fields of philosophy, psychology, and cognitive sciences on explanations between humans. To summarize, the author highlights four findings that he argues should be a part of XAI. These are:

1. Explanations are contrastive. People do not ask why event P happened, but rather why event P happened instead of some other event Q. An explanation is a response to a counterfactual question.

2. Explanations are selected in a biased manner. People select one or two causes from all possible causes as the explanation. The selection is influenced by cognitive biases.

3. Referring to probabilities or statistical relationships in explanations is not as effective as referring to causes.

4. Explanations are social, a transfer of knowledge, presented relative to the explainer's beliefs about the explainee's beliefs.

Points 1 and 3 say that causal and contrastive explanations are most effective, point 2 indicates that explanations should be simple, and point 4 says that explanations should take the knowledge of the recipient into consideration to make it understandable. Understanding how humans create and present explanations and what constitutes effective explanations can be helpful when considering explanations of predictions made by computer programs. As the points above suggests, explanations should not be evaluated solely on the basis of how mathematically accurate or complete they are, but to which extent they resemble explanations that humans are familiar with and make it understandable for the recipient of the explanation.

2.2.1 Goals of explanations

Explanations can take different forms and contain different information depending on the need of the user requesting explanations. Table 2.1 shows examples of possible target audiences for explanations, and what each group might want the explanations to help them with. The target group that explanations are intended for could be machine learning engineers or data scientists, domain experts within a domain that machine learning models are applied, end users such as customers using a product, or regulators. The machine learning engineers or data scientists might want information to debug a model, validate if it works as intended or to reduce bias. Domain experts might be interested in information to trust that predictions are reasonable in situations where errors could be critical. End users could be interested in why an advertisement is shown or why a certain movie is recommended on a streaming platform. Regulators might be interested in ensuring that a system is not discriminating users or violating their privacy.

Mohseni et al. [2021] divides approaches to explanations into XAI design goals based on three target groups: AI novices, data experts, and AI experts. AI novices are end-users that use AI products in daily life without any expertise on machine learning systems. Data experts includes data scientists and domain experts that use machine learning for analysis, decision making, and research. AI experts design machine learning algorithms and interpretability techniques for

Target group	Use case	Example
ML engineer/data scientist	Debugging, validation, bias reduction	What data features where responsible for the erroneous output?
Domain expert	Trust	Can I trust this prediction to work as intended?
End user	Reason for recommendation/decision	How can I change to get a bank loan?
Regulator	Discrimination, privacy	Does the model treat each person equally?

Table 2.1: Examples of different target groups for explanations and examples of what each group might want explanation to help them with.

XAI systems. This division is similar to the one in Table 2.1, where AI novices is end users, data experts is a combination of domain experts and data scientists, and AI experts are similar to ML engineers. However, Table 2.1 separates domain experts and data scientists because domain experts might be operators receiving predictions and trying to minimize financial consequences, whereas data scientists might be AI experts that train machine learning models and visualize data. Nevertheless, Mohseni et al. [2021] provides descriptions of design goals targeted at each of the target groups that are interesting to consider.

According to Mohseni et al. [2021], the four main design goals for AI novices are algorithmic transparency, user trust and reliance, bias mitigation, and privacy awareness. For data experts, the main goals are model visualizations and inspection, and model tuning and selection. For AI experts, the main design goals are model interpretability and model debugging.

Regardless of how different target groups are defined, it is important that the target audience for explanations are identified and defined before deciding how to create those explanations. The value an explanation provides depends on the goal of the explanation and the need of the target audience.

2.3 Evaluation of explanations

In order to evaluate if an explanation method is sufficient for an explanation goal or to compare different explanation methods, there is a need for specific evaluation criteria or metrics. Before deciding on the approach for evaluating explanations, one needs to define the target group and the explantion need, as described in Section 2.2.1. Common for many of the evaluation methods proposed in the

literature is that they are based on the field of human-computer interaction and either involves measuring how a human subject performs in specific tasks with the help of explanations or asking human participants about their experience with the explanations through interviews or questionnaires.

Doshi-Velez and Kim [2017] introduced a taxonomy of evaluation with three categories: application-grounded evaluation, human-grounded evaluation, and functionally-grounded evaluation. Application-grounded evaluation involves human experiments with a domain expert performing the task that explanations are meant to assist with. Human-grounded evaluation is also performed with human participants, but with a simplified task. Finally, functionally-grounded evaluation uses a formal definition of interpretability and involves no humans, but is purely computational. Mohseni et al. [2021] categorizes XAI evaluation measures in five categories: mental model, explanation usefulness and satisfaction, user trust and reliance, human-AI task performance, and computational measures. Both classifications are dominated by evaluations where human subjects from the target group are exposed to explanations and either use them to complete a task and measure an improvement in the performance on the task or are subsequently interviewed to give their subjective opinion on one or more aspects of the explanations.

2.4 Time series forecasting

A multivariate time series $\mathbf{X} = \{x_t\}_{t \in T}$ is an ordered set of k-dimensional vectors $x \in \mathbb{R}^k$, each recorded at a specific time $t \in T$, where T is the set of datetime values. One of the k values in each vector represents the target variable that are the variable to forecast, whereas the other values represent covariates. Below, the target variable is separated from the other variables by defining $\mathbf{y}$ as the vector of the target variable and $\mathbf{X}$ for covariates, with subscripts as below to indicate which sequence of time steps is included.

Time series forecasting is the process of predicting the values of a time series at one or more future time points based on the previous, historical values, and possibly the historical values of covariates and known future values of covariates. The problem of multi-step time series forecasting can be defined as predicting a sequence of h future values of target variable y at time t, written as $\mathbf{y}_{t+1:t+h} = [y_{t+1}, y_{t+2}, ..., y_{t+h}]$, given the sequence of p previous values of the target variable $\mathbf{y}_{t-p+1:t}$ and covariates $\mathbf{X}_{t-p+1:t}$. The subscript indicates the start and the end time point for the sequences. In some cases, the historical data might be delayed by m time steps such that the available sequence of previous values at time t is given by $\mathbf{X}_{t-m-p+1:t-m}$ for covariates and the same subscript for the target variable. Similarly, the desired horizon of future time points to predict might start n time steps into the future, such that the desired forecast horizon to be

forecasted at time t is given by $\mathbf{y}_{t+n+1:t+n+h}$.

2.4.1 Forecasting methods

Time series forecasting methods can be divided into classical statistical methods and methods based on machine learning. In the latter category there is also hybrid methods which combines statistical and machine learning methods. The focus of this thesis is forecasting methods based on machine learning and neural networks, in which the methods can be viewed as black boxes. The M_x forecasting competitions are open competitions on forecasting where the results often reflect the best forecasting methods available. The results from the M5 accuracy competition show that pure machine learning forecasting methods now outperform hybrid methods of statistical and machine learning methods that was best performer in the previous M4 competition [Makridakis et al., 2022]. As noted in [Makridakis et al., 2022], most of the best performing methods utilized the LightGBM method, in which multiple instances of LightGBM were used in an ensemble. LightGBM is a gradient boosting tree ensemble method developed by Microsoft [Ke et al., 2017]. Such ensembles of machine learning methods are not the easiest methods to explain, but can be explained individually and then combine the explanations in the same way as the ensembles are combined.

The statistical forecasting methods have been among the best performing methods until machine learning methods began outperforming them. ARIMA and exponential smoothing are widely used statistical forecasting methods.

2.4.2 Autoregressive model

In linear regression, the value of the target variable is predicted using a linear combination of one or more covariates. In an autoregressive model, the target variable is instead predicted using a linear combination of the previous values of the target variable itself. The number of previous time steps used in the linear combination is given by p, and the autoregressive model is then denoted by AR(p). The basic formula for an autoregressive model is given in Equation 2.1.

$$y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} + \epsilon_t, \tag{2.1}$$

where ϕ_i, $i \in [1, ..., p]$, are the linear weights, and ϵ_t is the error term.

Since a linear autoregressive model only consists of a linear combination of the previous values of the target variable, it is easily interpretable. By plotting the linear weights, one can see how much each of the previous time steps contributed towards a prediction.

2.4.3 Accuracy metrics

To quantify the accuracy of forecasting, one needs to decide on a metric. The metrics are based on the error, which is the difference between the forecasted value and the true value for each time step. For instance, the forecast error for the time step h into the future forecasted at time t is given by $e_{t+h} = y_{t+h} - \hat{y}_{t+h|t}$, where the first term is the true value and the second is the forecasted value. Hyndman and Athanasopoulos [2018] categorize point forecast accuracy metrics as scale-dependent, precentage errors, or scaled errors. Scale-dependent metrics, such as Root Mean Squared Error (RMSE), defined in Equation 2.2, are given in the same unit as the original values, and therefore the value of the error is dependent on the scale of the original data. Percentage errors are unit-free and could therefore make it easier to compare accuracy across different data sets. Percentage errors do not work well when the true value of the target variable at some time step is zero, or if the unit of measurement does not have a meaningful zero. The most common metric using percentage error is Mean Absolute Percentage Error (MAPE), given in Equation 2.3. Scaled errors are metrics for comparing forecasts across series of different units, and is not relevant in this thesis.

$$\text{RMSE} = \sqrt{\text{mean}(e_t^2)}. \tag{2.2}$$

$$\text{MAPE} = \text{mean}\left(\left|\frac{100 \cdot e_t}{y_t}\right|\right). \tag{2.3}$$

Chapter 3

State of the art

This chapter describes important work within the field of Explainable Artificial Intelligence (XAI), both general methods and for time series forecasting specifically, and is an answer to research question 1 in Section 1.3. The goal of this chapter is to describe the state-of-the-art within XAI, and to describe in detail some of the explanation methods with different approaches for explaining predictions. The latter is to highlight different approaches for achieving explainability, and to discuss advantages and disadvantages for each of them, in order to select approaches for later work in this project. The descriptions will be focused on the properties of the explanations the methods provides, and not too much on the algorithms used to calculate them. This chapter does not include all categories of XAI methods, but a selection of methods thought to be relevant. The chapter begins with an overview of the field in Section 3.1 with a classification of the methods within the field, describing what type of explainability each of the classes can provide. Then follows a description of interesting methods within XAI, divided into model-agnostic methods in Section 3.2, model-specific methods in Section 3.3, explanation methods for time series data in Section 3.4, and interpretable forecasting models in Section 3.5.

3.1 XAI overview

The explanation methods within XAI can be classified along multiple dimensions, depending on which properties of the explanation models are of interest. Barredo Arrieta et al. [2020] first divides XAI methods into transparent models and post-hoc explainability methods.

Transparent methods include linear regression and decision trees. These simple methods will not be a part of this thesis, because they are already interpretable

and are therefore outside the scope of this thesis. This chapter will rather include another class of methods which will be referred to as interpretable models. These include methods built on neural networks or other models that by themselves are black boxes, but which has interpretability techniques built in. This can for instance be that internal weights are outputted together with predictions.

Post-hoc explainability models are methods for generating explanations for predictions from already trained prediction models. This means that one has a prediction model for making predictions, and a separate explanation method for explaining those predictions. Post-hoc explainability methods can be either model-agnostic or model-specific. Model-agnostic means that they can be used to explain any prediction model. Model-specific methods are created to explain a specific model, for instance specific types of neural networks such as convolutional or recurrent neural networks. They are not able to generalize to explain all models. Post-hoc explainability methods can have either local or global scope. Local models give individual explanations to predictions of individual instances, whereas global models give explanations about the logic of the model [Adadi and Berrada, 2018].

3.2 Model-agnostic explanation methods

Barredo Arrieta et al. [2020] divide the model-agnostic methods into three types of techniques: Explanation by simplification, feature relevance explanations, and visual explanations. Model-agnostic methods are the most flexible of the explanation methods, and can be applied to explain many different prediction methods. However, there are drawbacks to model-agnostic methods, as there is a trade-off between the flexibility to explain many different methods and the ability to exploit the characteristics of the prediction model to be explained.

3.2.1 Explanation by simplification

Explanation methods by simplification constructs a simpler interpretable model to explain the already trained original model. The most widely used explanation method by simplification is Local Interpretable Model-agnostic Explanations (LIME), described below. When explaining a prediction model with a simplified model, the simplified explanation model will not be able to exactly reproduce the prediction model due to the simplification. One approach, followed by LIME, is to focus on a local subspace such that the simplified model gives a good approximation of the prediction model in that area. Another approach is to accept the loss in accuracy and have the simplified model explain the entire space the prediction model operates in. An example of that approach is to construct simple

global decision rules. The choice of approach depends on the goal of the explanations, either have a good local approximation to explain individual predictions or to get a rough simplification of an entire model.

LIME

Ribeiro et al. [2016] presents the algorithm Local Interpretable Model-agnostic Explanations (LIME), which provides local explanations to individual predictions for any classifier or regressor. The paper defines explaining a prediction as presenting information via text or visuals that provide understanding of the relationship between the instance's components and the model's prediction.

Explanations need to use a representation that is understandable to humans, which will be called interpretable representation. An example of an interpretable representation is the presence or absence of each word in a text. For the original representation of an instance $\mathbf{x} \in \mathbb{R}^d$, a binary vector for its interpretable representation is given by $\mathbf{x}' \in \{0,1\}^{d'}$. The original prediction model f to be explained operates on the original features represented by $\mathbf{x}$, and simplified explanation models $g \in G$ operates on the interpretable feature representation $\mathbf{x}'$, where G is the set of potential interpretable explanation models. Examples of sets G could be linear models and decision trees.

Given the original model f, a set of simplified explanation models G, loss function $\mathcal{L}$, measure of complexity Ω, and proximity measure π_x, the explanation for a single instance $\mathbf{x}$ by LIME is given in Equation 3.1.

$$\xi(x) = \underset{g \in G}{\arg\min}\, \mathcal{L}(f, g, \pi_x) + \Omega(g). \tag{3.1}$$

The first part of the equation is a loss function $\mathcal{L}$ that measures how well the explanation function g approximates the prediction model f in the neighborhood around instance x, weighted by the proximity measure π_x. Weighing samples by how close they are to the instance ensures that the simplified model is a good approximation close to the instance. The second part is the complexity Ω of the explanation model g. The complexity is important to keep at a minimum because even a linear model or decision tree is hard to interpret if there are too many weights or nodes. Minimizing the sum of the approximation and the complexity ensures that the simplified local explanation model g gives a good approximation of the prediction model around the instance while being simple enough to provide explanations understandable by humans.

The loss function is approximated by drawing samples uniformly at random around the instance x, weighted by the proximity measure π_x, in other words by how close the sample is to the instance. LIME is model-agnostic, meaning that there are no assumptions made about the function f being explained. The class of explanation methods G must be simple, interpretable models, for instance linear

models with few weights or decision trees with few nodes. The original article uses only linear models as explanation methods.

LIME provides a simplified explanation model to explain the prediction of an instance from any prediction model. The advantages is that it works for any prediction model, since it only perturbs the input and studies the changes in the output, and that the resulting model is simple. Disadvantages is that some accuracy is lost when approximating with a simple model, and that more complex interactions between features are lost. Also, since the approximation is only valid around a single instance, a separate LIME explanation model must be calculated for each instance that one wants to explain.

Anchors

Anchors [Ribeiro et al., 2018] is a model-agnostic explanation method that computes decision rules, referred to as anchors, that is a simplification of classifications. The anchors explain a classification prediction such that changes in the features that are not included in the anchor do not affect the prediction. The prediction should be the same for all the instances that an anchor is valid for. Ribeiro et al. [2018] claim that anchors are intuitive, easy to comprehend, and have large coverage, meaning that a human should be able to correctly make predictions of unseen instances based on the anchors, and know when the rule applies.

Given a prediction model $f : X \to Y$ and an instance $x \in X$. Let $\mathcal{D}$ be a perturbation distribution around instance x using an interpretable representation, in the same way as LIME. Let A be a set of predicates using the interpretable representation, such that $A(x)$ returns 1 if all its predicates are true for instance x. An example of such predicate can be that a feature has a certain value, or a specific word is included in a string of text. Further, let $\mathcal{D}(\cdot|A)$ denote the conditional distribution when the rule A applies. Then A is an anchor if $A(x) = 1$ and A is a sufficient condition for $f(x)$ with high probability. Written mathematically,

$$\mathbb{E}_{\mathcal{D}(z|A)}[\mathbb{1}_{f(x)=f(z)}] \geq \tau, A(x) = 1, \tag{3.2}$$

where τ is the desired level of precision. The precision value need to be decided for how precise one wants the anchors to be. There is a trade-off between precision and coverage, with larger precision leading to lower coverage. For a precision of 1, all feature values are fixed, and the coverage is only for the instance being explained.

The anchors method operates on classification tasks, but can be used for regression if the regression task is discretized into a number of classes of outcomes. The downside to this is that one either needs to make a large number of classes

and have too few samples per class, or make only a few classes with a wide range of values within each class, making the predictions very limited in accuracy. This method is thus mostly relevant for classification tasks.

For a user, it can be easier to know when an anchor is valid that a linear explanation from LIME, because the predicates in an anchor is clearly defined, whereas the linear model in LIME is an approximation in a neighborhood around the instance.

3.2.2 Feature relevance explanations

Feature relevance explanations aims to describe how much influence each feature has on the prediction by a prediction model. This can be helpful for understanding how the model uses the data, and for investigating which features that contribute the most when a prediction is wrong. It can also be used for feature selection to make a predicition model simpler and easier to understand, by removing features with little importance. SHAP is a widely used and successful explanation model for feature relevance.

SHAP

Shapley Additive Explanations (SHAP) [Lundberg and Lee, 2017] is an explanation method that assign an importance value to each feature for an individual prediction. It can also give feature importance for a set of instances, and show both the mean importance and the distribution. SHAP unifies six previous explanation methods. Two of those are described in this chapter, LIME in Section 3.2.1, and DeepLIFT in Section 3.3. The last four is a method called layer-wise relevance propagation, which explains deep neural networks and is a special case of DeepLIFT, and three methods from cooperative game theory to compute Shapley values, which is defined below. All six methods belong to the class of additive feature attribution methods, defined in Equation 3.3.

Additive feature attribution methods have an explanation models that is a linear function of binary variables:

$$g(x') = \phi_0 + \sum_{i=1}^{M} \phi_i x'_i, \tag{3.3}$$

where g is a simple explanation model approximating the original prediction model f, using simplified binary input features x', and $\phi_i \in \mathbb{R}$ are the feature attributions. The simlified binary input features are the same as the interpretable representation in LIME and anchors in Section 3.2.1. A mapping function exists that can convert the binary input features to the original input features, such that original input $x = h(x')$, where h is the mapping function. The goal is then

that for all variables z' that is close to x', the explanation model approximates the origin prediction model, $g(z') \approx f(h(z'))$.

The SHAP method builds on theoretical results from a field called cooperative game theory. It can be shown that there is only one possible explanation model g that follows the definition of additive feature attribution methods in Equation 3.3, and at the same time satisfies three properties: local accuracy, missingness, and consistency. Local accuracy means that the explanation model equals the original model at the exact point of the instance. Missingness says that missing features have no attributed impact on the explanation. Consistency says that if a model changes such that the contribution from a feature increases or stays the same regardless of all the other features, then that feature's attribution should not decrease. These three properties are desirable. Local accuracy is necessary for it to be a good explanation of the instance, missingness is important because all attribution should be given to the features that are present, and consistency is important because when a feature contributes more, it should get more importance.

The Shapley values that SHAP is based on, ϕ_i, is the only possible explanation model g that follows the definition of additive feature attribution methods in Equation 3.3 and satisfies the three properties described above. They are given by the following equation:

$$\phi_i(f, x) = \sum_{z' \subseteq x'} \frac{|z'|!(M - |z'| - 1)!}{M!}[f(z') - f(z' \setminus i)], \tag{3.4}$$

where $|z'|$ is the number of non-zero entries in the simplified input features z', M is the total number of simplified input features. $z' \setminus i$ means setting $z'_i = 0$. $z' \subseteq x'$ are the vectors z' where all non-zero entries also appear in x'. The idea here is that when calculating the importance of feature i, one takes the difference between the output from the prediction model with and without the value of feature i, while holding all the other features the same. This is done for all available combinations of the other features. The Shapley value for feature i then says how much that feature contributed to the prediction compared to the average prediction of the dataset.

The values ϕ_i in Equation 3.4 are known as Shapley values, and this equation is a special case of Shapley values called SHAP values. Normally, Shapley values do not satisfy the local property, but this version with SHAP values does. The SHAP values are defined as Shapley values of a conditional expectation function of the original model, conditioned on simplified inputs z'. This means that the Shapley values are calculated conditioned on the presence or absence of input features, to calculate their contribition. The SHAP value for a feature is the change in the expected model prediction when conditioning on that feature. The

original model using simplified input is approximated using the expectation of the original input, given the simplified input:

$$f(z') = E[f(z)|z_S], \tag{3.5}$$

where S is the set of non-zero indexes in z'. The SHAP values attribute to each feature the change in the expected prediction when conditioning on that feature. When the model is linear and the features are independent, then the first SHAP value ϕ_0 is the expected prediction without knowing any feature values, $E[f(z)]$, then ϕ_1 is the change when conditioning on knowing the first feature, $E[f(z)|z_1 = x_1$. It continues with the second, when conditioning on knowing both the first and second feature, and so on until all feature attributions are calculated. When the model is not linear or the features not independent, then the SHAP values is the average of all orderings.

Exact SHAP values are computationally challenging to compute, especially due to the fact that for non-linear models or dependent features one need to calculate the values with all possible orderings of the features. Approximation techniques are used in practice to calculate the SHAP values. Two important approximation methods are Kernel SHAP and Deep SHAP.

Kernel SHAP is a model-agnostic approximation of the SHAP values using LIME, described in Section 3.2.1, with the set of explanation models G limited to linear models, and with loss function $\mathcal{L}$, proximity measure π_x and complexity measure Ω set to the following values:

$$\mathcal{L}(f, g, \pi_{x'}) = \sum_{z' \in Z} [f(h_x(z')) - g(z')]^2 \pi_{x'}(z'), \tag{3.6}$$

$$\pi_{x'}(z') = \frac{(M-1)}{\binom{M}{|z'|}|z'|(M-|z'|)}, \tag{3.7}$$

$$\Omega(g) = 0. \tag{3.8}$$

These values for LIME turns out to be the unique values that make LIME consistent with the three properties local accuracy, missingness and consistency.

Deep SHAP is another method for calculating SHAP values, and is a deep neural network specific approximation of the SHAP values using DeepLIFT, described in Section 3.3. Assuming that the input features are independent and the deep model is linear, Deep SHAP approximates the SHAP values using DeepLIFT with the expected model output $E[f(x)]$ as reference value. The feature attributions are then found by the difference to the reference value.

To conclude, the SHAP method can produce feature importance scores for a single instance, where the feature importance is the difference that each feature makes on the output, compared with the average output from the training set.

SHAP has a theoretical basis with desirable properties, which the methods it builds on do not have, including LIME. Kernel SHAP uses LIME to comput the SHAP values, with a configuration such that the properties are satisfied.

SHAP for tree ensembles

In [Lundberg et al., 2018], the original creators of the SHAP method describes a new approach for calculating SHAP values for feature attribution specifically for trees and tree ensembles, with an algorithm calles Tree SHAP. This method improves the speed of calculating the SHAP values for tree ensembles, and have the same desirable properties as the other methods for calculating SHAP values. The method is implemented in the official releases for the gradient boosting methods XGBoost and LightGBM. The details of the algorithm is not included here, only mentioned and referenced here because this is the actual method for calculating SHAP values when using XGBoost, LightGBM or other tree ensemble prediction models and use SHAP to explain them.

3.2.3 Visual explanations

Visual explanation methods provides visualizations that work together with feature relevance explanation methods. A notable method is Individual Conditional Expectation (ICE), which visualizes how a prediction of the target variable changes when changing one specific covariate while holding all other constant. Visualizations are also used in the other categories of explanation methods as well, but the focus in this category is to gain insight into feature interactions. Barredo Arrieta et al. [2020] argues that creating such visualization methods as model-agnostic post-hoc methods is a complex task as it needs to work well for any machine learning model without access to its inner structure, only to input and output. Therefore, most visualization methods are model-specific, and almost all model-agnostic visualization methods work together with feature relevance techniques.

Individual Conditional Expectation (ICE)

Individual Conditional Expectation (ICE) [Goldstein et al., 2015] is a visualization method that plots for a given covariate and target variable, how the prediction of the target variable changes for different values of the covariate. It is named Individual Conditional Expectation because each plot displays a separate line for each individual instance, such that one can observe the effect a covariate has on each specific instance. This is an improvement of an earlier method called Partial Dependency Plots (PDP) [Friedman, 2001], which plot an average over

all instances. By taking the average over all instances, individual effects that are specific to some instances can be hidden in the average.

ICE is limited to display the effect of changes of one covariate at a time, and is therefore not able to show interactions between multiple covariates. It is also possible that when changing one covariate while holding all other constant, one is including combinations of values that are impossible in the real world situation the model approximates. Then, the behaviour of the prediction model in the plots might not be representative of how it works for valid value combinations. An advantage of PDP over ICE is that it is easier to see the overall effect of how one covariate affects the predictions in PDP, as it is an average over all the instances, and that in ICE it can be hard to get the overall picture when there is a large number of different instances. However, ICE is able to show in more detail how each covariate influences a prediction.

3.3 Model-specific explanation methods

Several explanation methods are developed for specific types of prediction models, most commonly for deep neural networks; multi-layer feed forward networks, Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs). According to Linardatos et al. [2021], most explanation methods for explaining deep learning models refer to image classification and produce saliency maps, that visually highlights the impact of each image region on the classification. This can be shown by marking more important areas with stronger colors than less important areas. Grad-CAM is one such explanation model that highlights important parts of an image for a classification. Another notable method within model-specific explanation methods is DeepLIFT, which calculates importance scores for each input to a deep neural network by comparing the output to a reference value.

Two approaches for model-specific explanation methods for deep neural networks are perturbation-based methods and backpropagation-methods. Perturbation-based methods makes perturbations, or changes, to the input, and observes the changes in later layers or the output. Backpropagation propagate a signal from the output to the input through the layers. The backpropagation approach is the most efficient. Both Grad-CAM and DeepLIFT are examples of backpropagation methods.

DeepLIFT

Deep Learning Important FeaTures (DeepLIFT) [Shrikumar et al., 2017] is an explanation method for deep neural networks which attributes an importance

score to each input calculated as the change in output that occurs when the input value is changed from its original value to a reference value.

Let t represent a target neuron, and t^0 the reference activation of t, which is the activation for the target neuron for a particular input that is chosen as the reference input. Define the quantity Δt as the difference from the reference, $\Delta t = t - t^0$. Let $x_1, x_2, ..., x_n$ represent neurons in one or more intermediate layers that are necessary and sufficient to compute t. Then DeepLIFT assigns contribution scores $C_{\Delta x_i \Delta t}$ to Δx_i such that the sum of the contribution scores for all x_i equals the difference from the reference, Δt, shown in Equation 3.9.

$$\sum_{i=1}^{n} C_{\Delta x_i \Delta t} = \Delta t. \tag{3.9}$$

The reference output of a neuron is its activation on the reference input. The reference input needs to be specified by the user, and normally requires domain knowledge in order to choose a good reference which yields meaningful results.

By treating positive and negative contributions separately, issues arising from positive and negative terms canceling each other out are avoided. For a neuron y, one can define Δy^+ and Δy^- as the positive and negative components of Δy. In the same way, one can separate the contribution score in the contribution from the positive components and the contributions from the negative components. Then, the activation and contribution are defined as the sum of the positive and negative components of each, respectively.

DeepLIFT decomposes the prediction of a deep neural network for a specific instance by backpropagating the contributions of all neurons to the features of the input. Sixt et al. [2020] performed a theoretical analysis of many backpropagation methods, and found that DeepLIFT was the only method that did not suffer from limitation of ignoring information in the later layers of neural networks.

Grad-CAM

Gradient-weighted Class Activation Mapping (Grad-CAM) [Selvaraju et al., 2017] is a model-specific method that produce visual explanations for any CNN classification. Visualizations by Grad-CAM as class-discriminative, meaning that the visualizations clearly indicate which class an image belongs to, unlike some previous methods that highlight important pixels but produces visualizations that look nearly identical for different classes. Without needing too much details, Grad-CAM weighs each pixel of each feature map of the last convolutional layer in a CNN with the gradient, and then average over the feature maps. Then all negative values are removed with a ReLu. Multiple improvements have been published that builds on Grad-CAM, but the idea that is presented here is that explanations can take the form of a heatmap of an image in which the highlighted

pixels and areas are discriminative such that one can see the difference between the heatmaps of different classes.

3.4 Explanation methods for time series data

Some model-agnostic methods have been adapted to be better suited for the time series domain, including SHAP [Bento et al., 2021], Grad-CAM [Saadallah et al., 2021] and LIME [Schlegel et al., 2021]. The original model-agnostic feature attribution methods explain how much each feature contribute towards predictions of single instances, and can be used also in the case of time series forecasting. This section describes two methods that are made specifically for time series data. TimeSHAP is an extension of SHAP for sequential data and recurrent methods. The series saliency method described here makes temporal saliency maps that includes the most important inputs in both the feature and temporal dimensions. This method could also be defined as interpretable and be included in the next section, since it is a combined forecasting and interpretation method, but it is included here because any deep learning module can be inserted for forecasting.

TimeSHAP

TimeSHAP [Bento et al., 2021] is an extension of SHAP for the sequential domain, extending KernelSHAP to a model-agnostic explainer that attributes importance to past events and features in the sequence. The approach is perturbation-based, meaning that the feature attributions are calculated by changing the input data and observing the changes in the output data. These perturbations can be made in three different domains, either along the temporal dimension, between features of the same time step, or for a specific feature at a specific time step. TimeSHAP is made specifically for explaining recurrent methods.

KernelSHAP is not applicable to sequential decision-making tasks, as it only calculates attributions for features of a single input instance. This means that if one has a sequence of decisions to make, with regular SHAP one would need to calculate the feature importance for each one individually. With TimeSHAP, the entire sequence can be taken into consideration, such that besides getting feature importance, one can get temporal importance, and the importance of a pair of feature and time.

To explain a sequential input, $X \in \mathbb{R}^{d \times l}$, with l events and d features, a linear explainer g is fit that approximates the local behavior of f by minimizing loss function as in SHAP. The perturbation function $h : \{0,1\} \mapsto \mathbb{R}^{d \times l}$ maps a simplified binary representation $z \in \{0,1\}^m$ to the original input space $\mathbb{R}^{d \times l}$.

Perturbations can be done either in the feature dimension or in the event dimension, as well as cell-level perturbations. Perturbations in the feature di-

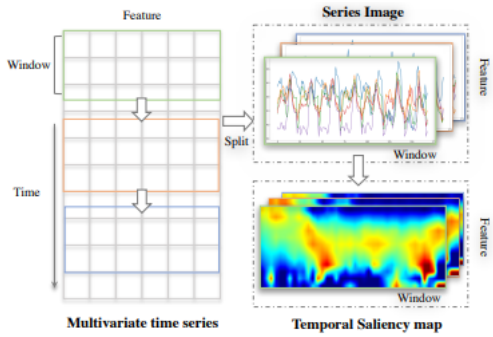


Figure 3.1: The series saliency method takes a multivariate time series as input, and produces series images with a sliding window of the time series. Along with a forecast, the interpretable output is a temporal saliency map, that for the window of the time series used as input to the forecast highlights how important each time step and feature was for the forecast. Figure from [Pan et al., 2021]

mension results in feature importance with SHAP values as in KernelSHAP. By perturbing in the event or time dimension, TimeSHAP can produce similar SHAP values for event imporance.

The difference between using SHAP as described in Section 3.2 and TimeShAP, is that SHAP explains a single instance with feature importance, whereas TimeSHAP can explain multiple instances in a sequence with both feature importance and temporal importance.

Series Saliency

Pan et al. [2021] presents a method for producing accurace time series forecasting and at the same time creating temporal saliency maps for explaining time series forecasting in both feature and time dimensions. It is building on work on saliency maps in computer vision, in which important parts of an image is highlighted. Figure 3.1 shows an example of a temporal saliency map that the method can create in the lower right of the figure, with the time steps in the horizontal direction and the features in the vertical direction. The series saliency method can be used with any deep learning method, meaning it is not specific to any model, but limited to deep learning methods. The predictions are split in two parts, one linear autoregressive module, and a deep learning module.

A multivariate time series is represented as a 2D series image, a matrix $X \in \mathbb{R}^{D \times T}$, with D features and window size T. A reference series image $\hat{X}$ is defined by adding noise or blur on each element of the original series image X. This is

done because blurring should help extracting trend, and noise should enhance local information. The series saliency method also has a learnable mask that selects between the original series image and the reference image, in order to have a balance between the characteristics of the original series image and exploring unexplored input space. The mask is used as a data augmentation method to help train the deep models. By changing the optimization, a similar mask can be used to find the temporal saliency map for interpretation. This is done by removing parts of the series image and end up with the most salient feature regions.

3.5 Interpretable forecasting methods

This section describes important time series forecasting methods based on neural networks or with neural network components, with built in interpretability that outputs information from internal parts of the models. These methods are usually not included in classifications of XAI methods, but since they provide information to interpret their predictions, they are included here as an important class of methods that can meet the objective of explaining predictions. They have various degrees of interpretability built into the architectures, that can be used for explanations of the predictions, either directly or as basis for explanations. Ideally, one should be able to explain the predictions of the most accurate forecasting methods, or compromise as little accuracy as possible in order to create meaningful explanations. However, the architectures and techniques used for learning patterns in data and produce predictions have various degrees of complexity, and therefore it might not be the best performing methods that can be best explained. The forecasting methods described in this section have different approaches to both how they produce forecasts and how interpretability is built into their architectures.

N-BEATS

N-BEATS is a neural network architecture consisting of a stack of fully connected neural network layers connected with forward and backward residual links, first introduced in Oreshkin et al. [2020] for univariate time series point forecasting, and in Oreshkin et al. [2021] for multi-horizon time series forecasting applied on mid-term electricity load forecasting. The architecture is a pure deep learning architecture with no time series specific feature engineering or input scaling. N-BEATS treat forecasting as a non-linear multivariate regression problem.

Figure 3.2 shows an overview of the N-BEATS architecture. The basic building block of N-BEATS is a block consisting of a sequence of fully-connected layers, followed by a split into a forecast and backcast. The forecast predicts future outputs. The backcast removes the contribution to the forecast from the input, and

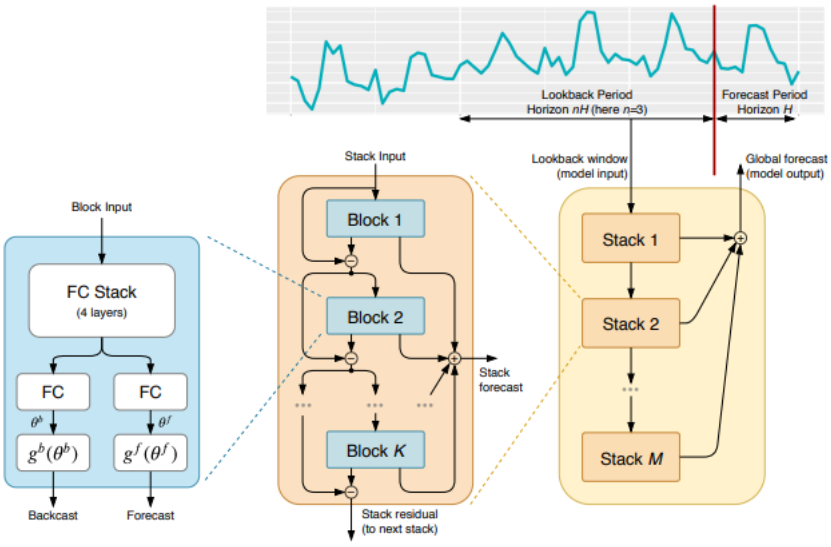


Figure 3.2: Overview of the modular N-BEATS architecture. Forecasts are made by summing the outputs from the M stacks, which consists of K blocks. Each block produces a forecast, and then removes the contribution to the forecast from the input, resulting in the backcast, which is then forwarded to the next block. This iteratively improves the forecast. Finally, each stack forecast is summed to create the final output. Figure from [Oreshkin et al., 2021].

forwards the remaining part of the input to the next layer. At the end of the stack of blocks, the stack forecast is constructed from the forecasts from all the individual blocks. Multiple stacks are then stacked together in the same way as the blocks in a stack, and the sum of the stack forecasts is the final forecast.

The architecture of a stack is as follows. Assume R blocks each having L hidden layers, and $\mathbf{x} \in \mathbf{R}^w$ as input. A fully connected layer with weights $\mathbf{W}^{r,l}$ and biases $\mathbf{b}^{r,l}$ is $\text{FC}_{r,l}(\mathbf{h}^{r,l-1}) \equiv \text{ReLu}(\mathbf{W}^{r,l}\mathbf{h}^{r,l-1} + \mathbf{b}^{r,l})$, where r is the block number, l is the layer number, $\mathbf{h}$ is a hidden layer, FC refers to fully connected layer, and ReLu is a rectified linear unit. This is a normal fully connected layer. Then, the stack is given by the following equations:

$$\mathbf{x}^r = \text{ReLu}[\mathbf{x}^{r-1} - \hat{\mathbf{x}}^{r-1}], \tag{3.10}$$

$$\mathbf{h}^{r,1} = \text{FC}_{r,1}(\mathbf{x}^r), ..., \mathbf{h}^{r,L} = \text{FC}_{r,L}(\mathbf{h}^{r,L-1}), \tag{3.11}$$

$$\hat{\mathbf{x}}^r = \mathbf{B}^r \mathbf{h}^{r,L}, \hat{\mathbf{y}}^r = \mathbf{F}^r \mathbf{h}^{r,L}, \tag{3.12}$$

where $\hat{\mathbf{y}}^r$ is the forecast from each of the blocks, and $\hat{\mathbf{x}}^r$ is the backcast. $\mathbf{B}^r$ and $\mathbf{F}^r$ are linear projection matrices for the backcast and forecast, respectively. The final forecast is the sum of forecasts of all blocks, $\hat{\mathbf{y}} = \sum_r \hat{\mathbf{y}}^r$.

Interpretability in N-BEATS is achieved using trend and seasonality decomposition on forecast outputs. The trend model is modeled using a polynomial of a small degree p, and with θ_r as polynomial coefficients predicted by the fully connected network of a given stack s with blocks r. The trend model is then given by:

$$\hat{\mathbf{y}}_{s,r}^{\text{trend}} = \sum_{i=0}^{p} \theta_{s,r,i} \mathbf{t}^i = \mathbf{T}\theta_{s,r} \tag{3.13}$$

where time vector $\mathbf{t} = [0, 1, 2, ..., H-2, H-1]^T/H$, H being the forecast horizon, and the alternative matrix representation with $\mathbf{T} = [\mathbf{1}, \mathbf{t}, ..., \mathbf{t}^p]$ is matrix of powers of $\mathbf{t}$.

The seasonality model uses Fourier series, as given below:

$$\hat{\mathbf{y}}_{s,r}^{\text{season}} = \mathbf{S}\theta_{s,r}, \tag{3.14}$$

where $\mathbf{S} = [\mathbf{1}, \cos(2\pi\mathbf{t}), ..., \cos(2\pi\lfloor H/2 - 1 \rfloor \mathbf{t}), \sin(2\pi\mathbf{t}), ..., \sin(2\pi\lfloor H/2 - 1 \rfloor \mathbf{t})]$ is the matrix of sine wave.

Although Oreshkin et al. [2020] claim that N-BEATS with the trend and seasonality model is interpretable, the interpretable information is limited compared to the other models in this chapter. The N-BEATS model works as a multivariate regression model, where the input is a window of previous values of the target variable, similar to the autoregressive model in Section 2.4.2. Even thought the input only consists of the previous values of the target variable, the model does not output information about how important each time step is to the predictions.

Recently, Olivares et al. [2022] published an extension to N-BEATS, named N-BEATSx, which includes exogenous variables in the model. The authors claim that this extended model improves the forecasting accuracy with nearly 20% compared to N-BEATS, and includes exogenous variables as a separate output for interpretability.

Transformer

Vaswani et al. [2017] introduced the Transformer, which is a sequence modelling architecture based on the attention mechanism and encoder-decoder structure. The Transformer is not an interpretable forecasting method, but it is a prerequisite for understanding the TFT model that comes after this. The Transformer outperforms recurrent and convolutional networks for detecting long term relationships in sequences, and make it possible to learn such relationships without being required to use recurrent or convolutional networks. Transformers were initially developed for natural language processing tasks such as language translation, but has later been applied to time series forecasting and other tasks involving sequencial data.

The Transformer has an encoder-decoder structure, in which an encoder takes the input and maps it to a continuous representation, and then a decoder generates an output sequence one element at a time. Both the encoder and the decoder consists of a stack of identical encoder or decoder layers, where the achitecture in [Vaswani et al., 2017] used 6 layers in each. In order to understand the architecture of the encoder and decoder layers, first the attention and multi-head attention need to be defined.

The attention function is a computation with three matrices, Q, K and V, where Q is called the query, K is called key, and V is called values. In essence, to calculate the attention between elements in a sequence, one takes each element, which is represented in the query, and multiply with all elements in the sequence, which is the keys. Then, one has a set of weights, and multiply again with the values to get the result. These are originally only vectors, but are converted to matrices in order to be trainable. The calculation is shown in Equation 3.15.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V. \tag{3.15}$$

Multi-head attention is the process of calculating attention for multiple heads simultaneously, where each head is a different representation for query, key, and value, shown in Equation 3.16. Each head uses its own representation, and then the heads are concatenated to calculate the combined attention. Using multiple heads can help learn multiple different relationships in the sequence, including short-term and long-term relationships.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O, \tag{3.16}$$

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V), \tag{3.17}$$

where the W-matrices are projection matrices with superscript indicating that it belongs to either query, key, value, or output. The heads are concatenated, and

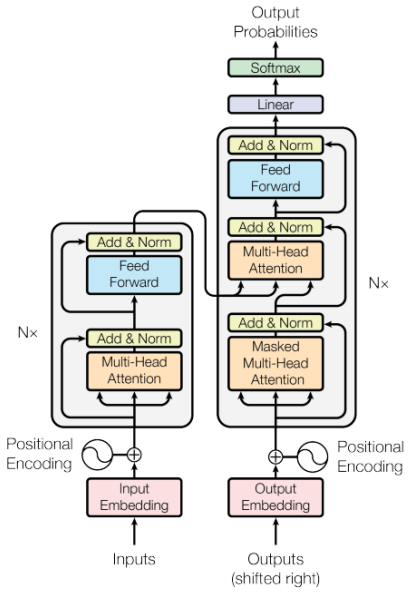


Figure 3.3: Overview of the Transformer architecture. The left part is the encoder, which converts the input sequence to a continuous representation. The right part is the decoder, which takes the output from the encoder and input for known covariates for the next steps of the sequence, and produces predictions for those next steps of the sequence. Figure from [Vaswani et al., 2017].

then multiplied by a final matrix to get the output attention.

The encoder layers consists of a multi-head self-attention layer followed by a position-wise fully connected feed forward layer, with residual connections and layer normalization. Self-attention means that attention is calculated on different positions of the same sequence, and attention is a way to quantify how important each time step is in the input for the predictions. The decoder layers is similar to the encoder, but have an additional multi-head attention over the output of the encoder stack. The decoder thus has one attention module for the target sequence, and one for the output from the encoder.

By relying only on self-attention to detect long-term dependencies instead of using recurrent or convolutional network, computation complexity is reduced, and is allows for more parallelization of the computations. Each head can be calculated in parallel, and calculating attention is less computationally complex than both recurrent and convolutional networks.

Temporal Fusion Tranformers

Temporal Fusion Transformer (TFT) [Lim et al., 2021] is an attention-based architecture that uses recurrent layers and interpretable self-attention layers to learn temporal relationships and to produce multi-horizon forecasts. It includes interpretable information to give insight into what influenced the forecasts, in particular variable importance from its variable selection modules, and temporal attention from its temporal self-attention layer. TFT has a structure similar to the Transformer method described above, an encoder-decoder structure, but has added multiple other modules to be able to process different types of input, including static, past and future known variables of different types.

Figure 3.4 shows an overview of the TFT architecture. The TFT architecture is made up of multiple components, with separate components made to build feature representations of a distinct input type, including static inputs, known future inputs and observed inputs. The most relevant components from explanations are the variable selection networks that select relevant input variables at each time step, and temporal self-attention layer that learn long- and short-term temporal relationships. Other components are gating mechanisms to skip unused components of the architecture, static covariate encoders to integrate static features, and prediction intervals via quantile forecasts to determine the rantge of likely target values at each prediction horizon.

Gated Residual Networks (GRN) are part of the TFT architecture in multiple places to give the model the flexibility to apply non-linear processing only where needed, which is difficult to know in advance. Its structure is shown in the upper right corner of Figure 3.4.

The variable selection networks are the basis for the variable importance that the model outputs, with importances for static variables, encoder variables and decoder variables separately. The variable selection is made for each instance. Variable selection networks use embeddings for categorical variables as feature representations, and linear transformations for continuous variables. All static, past and future inputs has their own separate variable selection networks. The output from the static variable selection network is used to to produce context vectors, which is used for the variable selection for past input and future known input. The input together with a context vector is passed through a Gated Residual Network (GRN), and the structure of both the variable selection network and the GRN are shown in Figure 3.4.

TFT uses a multi-head self-attention mechanism, similar to the one described in Section 3.5 for the Transformer with minor adjustments, to learn long-term relationships across different time steps. The multi-head attention is modified to share values in each head, and using additive aggregation of all heads, in order to be more easily interpretable.

The multi-head self-attention mechanism is part of a module called temporal

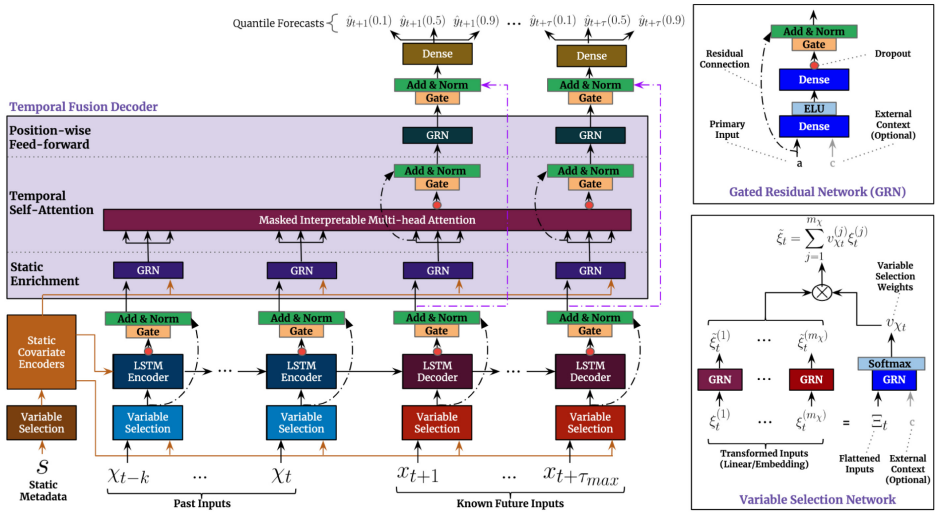


Figure 3.4: Overview of the Temporal Fusion Transformer architecture. Each input go through a variable selection network, then past input is encoded, future known input is decoded, and static data has its own encoder. Then, there is a multi-head self-attention layer to learn temporal relationships, before a feed-forward network leading to predictions. Figure from [Lim et al., 2021].

fusion decoder, which uses multiple layers to learn temporal relationships. In addition to the self-attention layer, there is also a sequence-to-sequence layer for time ordering of inputs as replacement for positional encoding in Transformer, static enrichment layer to add static metadata to the temporal features, and a position-wise feed forward layer.

The explanations that TFT provides are variables importances from its variable selection networks, and temporal attention from its temporal fusion decoder with the multi-head self-attention layer. The variable importance is split into static variables, encoder variables and decoder variables. The attention quantifies the importance of each time step in the input. One thing to note is that that there is a single attention for multi-horizon forecasting, meaning that when predicting multiple time step, one get one calculation of attention, not one for each of the time steps that are predicted. With these two types of explanations, TFT provides information that can be used to understand the predicions it makes.

NeuralProphet

NeuralProphet [Triebe et al., 2021] is an extension of Facebook's Prophet [Taylor and Letham, 2018], and is a forecasting method with interpretable components based on additive decomposition of time series. It uses autoregression and neural networks on lagged variables for improved forecasting accuracy compared to Prophet.

The NeuralProphet model is made up of multiple additive components that each produce outputs of h time steps at once, which is the number of time steps into the future that is being forecasted. For a forecast of length $h = 1$, the model can be written as in Equation 3.18.

$$\hat{y}_t = T(t) + S(t) + E(t) + F(t) + A(t) + L(t), \tag{3.18}$$

where $T(t)$ is the trend at time t, $S(t)$ is the seasonal effects at time t, $E(t)$ is the event and holiday effect at time t, $F(t)$ is the regression effect at time t for future-known exogenous variables, $A(t)$ is the auto-regression effects at time t based on past observations, and $L(t)$ is the regression effects at time t for lagged observations of exogenous variables.

The trend is a piece-wise linear function where the growth rate changes at a number of changepoints. The number of changepoints is given as a hyperparameter to the model. Before training, the points are assigned with equal distance from each other, and during training the most relevant points are selected. The user can also manually define specific time points for the changepoints, but this requires the user to know the dataset well.

Seasonality is modelled using Fourier terms, with k terms for seasonality with periodicity p, given by Equation 3.19. Default number of terms are $k = 6$ for

yearly seasonality with $p = 365.25$, $k = 3$ for weekly seasonality with $p = 7$, and $k = 6$ for daily seasonality with $p = 1$. Each of the seasonal periodicities is automatically used if the data frequency has a higher resolution than the periodicity and the data spans at least two full periods.

$$S_p(t) = \sum_{j=1}^{k} \left(a_j \cdot \cos\left(\frac{2\pi jt}{p}\right) + b_j \cdot \sin\left(\frac{2\pi jt}{p}\right) \right). \tag{3.19}$$

Auto-regression uses past values of a variable to predict future values of the same variable, and the order p refers to the number of past values to use in the model, denoted $AR(p)$. The classic definition of such auto-regression model is given in Equation 3.20.

$$y_t = c + \sum_{i=1}^{i=p} \theta_i \cdot y_{t-i} + e_t, \tag{3.20}$$

where c is the intercept, each θ-coefficient is fitted for a past value and e is a noise term. For multi-step horizon, $h > 1$, one model for each forecast step must be fitted for classical AR method. The auto-regression module in NeuralProphet is based on AR-Net [Triebe et al., 2019], which enables the model to produce all h forecast steps with one model. NeuralProphet provides three types of auto-regression, named linear AR, deep AR, and sparse AR. Linear AR consists of a single neural network layer with p inputs, h output, and no biases or activation functions, which is identical to the classic AR model.

Deep AR trains a fully connected neural network with a number of hidden layers and dimensions controlled by the user. The input to the first layer is the p last observations, and the final layer outputs h values. A rectified linear unit is used after each layer except the final layer. Formally, for l hidden layers with dimensions d, the network can be written as in Equation 3.21.

$$\begin{aligned} a_1 &= \text{ReLU}(W_1 x + b_1), \\ a_i &= \text{ReLU}(W_i a_{i-1} + b_i) \text{ for } i \in [2, ..., l], \\ y &= W_{l+1} a_l. \end{aligned} \tag{3.21}$$

Sparse AR can be used in which the order p can be set to a high value, and then using a regularization function with the result than only a selection of the past time steps get a non-zero weight. In that way, the model selects the most significant time steps itself.

The lagged regressors module is identical to the auto-regression module, with the only difference being that the input is the past values of a covariate instead of the target variable. A separate lagged regressor model is made for each covariate.

Future regressors are also modelled in the same way as auto-regression, but for future regressors, both the past and the future values of the regressor must be known. It must be known for the entire forecast horizon. Events and holidays are modelled with a binary variable with 1 when the event occurs and 0 otherwise. They are otherwise modelled in the same way as future regressors, the difference being that the values are either 0 if it is no event or holiday or 1 if it is. They must also be known for the entire forecast horizon.

Interpretation is achieved by considering each component individually. Trend, daily, weekly and yearly seasonality is straight forward to see the contribution. For the other components, one can see how much each component contributes to the predictions by looking at their values. Also, the model outputs parameters of the trained model.

Chapter 4

Method

This chapter describes the approach used for the practical part of this thesis. The original plan was to create a novel method based on attention and time series analysis, with a Transformer as a possible starting point, but it was later abandoned due to technical issues and limited time. Instead, the plan was changed to compare different approaches for explaining forecasts, and evaluate them together with domain experts. The chapter discusses methods selected based on the state-of-the-art in Chapter 3 certain selection criteria, and how the explanations generated by the methods will be evaluated. As described in the state-of-the-art, there are two main approaches to explain forecast predictions, either by interpretable forecasting models that outputs internal information that are used as explanations or as basis for explanations, or by applying post-hoc explanation models on top of non-interpretable forecasting models.

An advantage of an interpretable model is that it outputs information about its internal workings, such that it is possible to get insight into how those parts of the model use the data. On the other hand, a post-hoc explanation model can be applied to different forecasting methods, such that one is not limited to the specific forecasting method, and can later change forecasting method without sacrificing the explainability. However, post-hoc methods can never fully understand what is going on inside the forecasting model, as they are limited to exploring how changes in the input changes the output from the model.

4.1 Selection criteria

Chapter 3 describes notable methods within XAI using different approaches for explanation, and time series forecasting methods with various degrees of interpretability. In order to decide which approaches that are appropriate or relevant

for the task of this thesis, here some selection criterias are established that characterizes the problem at hand and what is expected from an explanation method in order to be suited for the task.

First, the target group that the explanations are made for need to be identified, as described in Section 2.2.1. In this case, the target group is domain experts within electricity power operations. Domain experts know how the electricity market works in the real world, but they might not be interested in information for tuning or debugging machine learning models.

Second, understanding the reasons why the target group want explanations are critical to understand what type of information the models should output, as is also described in Section 2.2.1. In this case, the domain experts are interested in explanations in order to decide whether to trust the predictions or not. Both the temporal and the feature dimensions are important in that regard, and graphs that show that the results are reasonable. Specifically, predictions that have the largest errors are interesting to get explanations for, in order to understand the reason why the prediction model make bad predictions in certain situations.

Third, the type of the explanations are important for how effectively they manage to fulfill the explanation need. Section 2.2 describes how humans use and understand explanations. Building on that knowledge, some characteristics of the ideal explanations can be described. Visualizations should be preferred over statistics or textual explanations, as they provide much information while being easy to understand. Explanations should also be contrastive, and be targeted to the knowledge and need of the recipient.

Fourth, the forecasting accuracy is important to consider. As the forecasting accuracy directly affects the financial outcome of the prediction problem, it is important with a lowest possible forecasting accuracy. However, since there is a need for explanations for the predictions at the same time as the error should be minimized, a compromise might be needed. The goal of this thesis is not to produce the best possible forecasting methods, such that the forecasting accuracy is not optimized. Nevertheless, it is important to have accuracy in mind when selecting approaches and when evaluating the result.

Lastly, the implementations of the methods must be taken into account, as the implementation affect how closely the methods achieve their theoretical performance, how optimized the code is, the amount of bugs the code has, and the quality of testing and documentation. In this thesis, this is measured by the amount of github stars the repositories have, as this measures how many people is involved in improving the code base by making contributions, reporting on issues, and reporting on how well the code works for real world applications. All implementations used in the experiments comes from code repositories with more than 1000 stars. This is not neccessarily a good measure of the quality, but it indicates that the implementations are tested by many and are more likely to

detect flaws.

4.2 Choice of methods

Given the selection criteria in Section 4.1, the methods or approaches selected should be able to provide meaningful explanations that are appropriate for the domain experts, that include information from both the temporal dynamics of the time series as well as the individual contributions from each of the features. The output should be presented using visualizations. Only methods that was described in detail in Chapter 3 is discussed here, as they were highlighted in the state-of-the-art because they represented different approaches to XAI. Other less known methods could be better within each approach, but these methods should nevertheless be representative of different approaches to explanations.

SHAP is a general XAI method that can be used for multiple types of models and data types, and has a strong mathematical foundation. It has been applied to explain multivariate time series forecasting problems in multiple papers, and is a method that could be suitable for the forecasting problem in this thesis given that it is used together with a forecasting method with good performance. As described in Section 2.4.1, LightGBM was the method used by most of the top performers in the latest forecasting accuracy competition in the M series, M5. For this gradient boosting tree ensemble method, SHAP values for feature attribution are calculated using the Tree SHAP algorithm mentioned in Section 3.2.2. This method is implemented specifically for LightGBM. Given the result of the forecasting competition and the established implementation by Microsoft, it is a natural choice to use LightGBM.

Lime is also a widely used model-agnostic method for providing explanations, but works in a different way than SHAP. Instead of calculating how important the value of each feature is for an instance, it trains a separate simpler model for each instance, with values in the neighborhood of the instance to explain. The explanation is then this simplified, often linear model. In this project, SHAP is chosen instead of LIME because SHAP has a better theoretical basis and provides feature importances that can be compared across instances. However, LIME could be a good candidate, and is mentioned in further work in Section 7.3.

TimeSHAP is a less known methods that can provide feature importance explanations as SHAP, but can also provide temporal explanations. It has thus the potential to be more useful than SHAP for explaining sequential tasks, as it also includes the temporal dimension in its explanations. However, it is an untested approach with source code released only after the models had been picked for experimentation in this thesis. It might be worth experimenting with in future work, but it seems to not be better for the forecasting task in this thesis, as that is not a sequential task in the way that is described for TimeSHAP.

The forecasting problem is not sequential in the way that forecast for one day influences the forecast for the next day, but is done independently.

The series saliency method is also an interesting method, due to the temporal saliency maps it produces which can give an understandable overview of how each feature and time step influence the predictions. However, like TimeSHAP, the method is not established in open source implementations and will therefore not be considered in this thesis.

Visualization methods such as ICE would be more suitable for machine learning experts and data scientists analyzing and debugging machine learning models or investigating patterns in the data, than for domain experts not involved in the development of forecasting methods from explaining a single prediction. This method is therefore not found to be targeted at the right target group and explanation need.

NBEATS is a promising forecasting method, but the only interpretable information it provides is the trend and seasonality, and it does not include exogenous variables. This information does not seem to be enough to provide acceptable explanations of predictions in this case. The recently published N-BEATSx method extends N-BEATS with exogenous variables, and would be a possible candidate method for testing in this thesis, but was published too late to be considered. With this extension, also exogenous variables would be included in interpretation, which would have given more insight.

The additive modular architecture of NeuralProphet makes it possible to visualize the contribution from each component individually. It is built up with components measuring trend and different seasonalities, as well as contribution from earlier time steps using autoregression and from covariates. This setup satisfies the criteria and should provide meaningful information to give insight into how each part contributes to the final predictions. Since each component can be visualized individually, and it includes trends and seasonalities as well as feature contribution for each time step, it provides much of the information that should be needed in order to trust the model.

The TFT outputs variable importance information from its variable selection networks, and attention which quantify the contribution from each of the time steps in the encoder. The information it outputs is not as extensive as that of NeuralProphet, but there is also a question of accuracy which must be taken into account.

Anchors and Grad-CAM are more suited for classification tasks than for the forecasting task in this thesis. DeepLIFT is included in SHAP, such that if a pure deep learning method would be used, then SHAP would be the method used to explain it, as it is based on DeepLIFT for explaining deep neural network models.

To conclude, for the experimentation in this thesis, TFT, NeuralProphet, and LightGBM with SHAP are selected for testing. These three methods represent

three different approaches to explanation of forecasting. TFT is a complex architecture with neural networks in different parts that provides interpretable output in the form of attention and feature importance. NeuralProphet is an additive forecasting model consisting of multiple components, each of which is fitted for a different aspect of the dataset. This can be trend, seasonality, or the contribution from previous time steps of a covariate. With the NeuralProphet model, one can see the contribution from each component separately. LightGBM with SHAP represent a more typical XAI approach, with separate forecasting and explanation methods, and with the explanation model having no direct insight into the workings of the prediction model. The goal of testing these methods is to answer research question three in Section 1.3, about which explanation approach is the best for explaining the forecasting problem of this thesis. In particular, finding out whether complex interpretable methods or dedicated forecasting methods with post-hoc explainable methods are the best approach is interesting to consider.

4.3 Evaluation

The accuracy of the forecasting methods will be compared to a baseline that uses simple autoregression. MAPE, defined in Section 2.4.3, is used as accuracy metric because it is an established metric used in the M5 forecasting competition, described in Section 2.4.1, and it is also the metric TrønderEnergi has used as metric for their forecasting methods. The forecasting accuracy is not the main metric to optimize in this thesis, but it gives an indication of whether the forecasting methods has learned from the data or not. That is to ensure that the models have learned enough from the data that the explanation methods is not trying to explain random predictions.

Evaluation of the explanations, which is the most important evaluation, will be a combination of qualitative assessment of the characteristics of the visualizations, and feedback from domain experts. For the user testing with domain experts, explanations will be generated with the chosen methods for predictions, and the domain experts give feedback on whether the visualizations of the explanations provide them with meaningful information, and then how they rate each explanation in comparison with the other explanations for the same time step.

The final evaluation of which method or methods are best suited for providing explanations of grid loss forecasting, all criteria will be taken into account, including MAPE forecasting accuracy, characteristics of the explanations, and the feedback from the domain experts from TrønderEnergi. The feedback from domain experts will be given the highest importance, since that is the target group for the explanations.

Chapter 5

Experiments

This chapter describes the experiments conducted during this research. It begins with a description of the dataset and data preprocessing of the dataset, then describes the time series forecasting problem that this work is attempting to explain. Then follows a description of the experimental setup and the experimental plan.

5.1 Dataset and data preprocessing

The dataset used in this thesis is from the Norwegian power produced TrønderEnergi Kraft, about grid loss, which describes the electric energy lost in the distribution network for each hour. This data is used to predict the amount of loss in the network that is expected the 24 hours of next day for each day at noon, which needs to be reported to the power market.

The dataset consists of hourly data, where each row in the dataset has a timestamp describing the date and hour. The target variable in the dataset is grid-loss, which describes how much energy is lost in the power grid in that hour, measured in MWh. The grid-load feature gives the total amount of power in the grid that hour, also measured in MWh. Grid-loss is directly correlated with grid-load. Additional features are forecasted temperature measured in Kelvin, and estimated demand for electricity in Trondheim, measured in MWh, which is the largest city in the region. At the time of forecasting, grid load and grid loss contains historical measurements, and is unknown in the future, whereas the temperature forecast and demand estimation is used as future known values for the forecast horizon. Additionally, the dataset contains a feature for holiday, which has value 1 if the data point is on a holiday, and 0 if not, and a feature stating if the row was marked incorrect by the experts at TrønderEnergi.

5.1.1 Data preprocessing

The dataset is already split in a training set and a test set, in which the training set contains hourly data from December 1st 2017 at 00:00, to November 30th 2019 at 23:00, in total 17520 hours of data. The test set contains hourly data from December 1st 2019 at 00:00 until Mai 31st 2020 at 00:00, in total 4369 hours of data. The dataset contains data from three different grids, and for simplification, only the data from the first grid is extracted and used in this thesis. The training data for the first grid contained two rows with missing values, which where removed before the data were used for experimentation. All other features, other than datetime, grid-loss, grid-load, temperature and demand, were removed from the dataset. No further preprocessing was done to the dataset except for individual adjustments for each method to make them work with the dataset.

5.2 Forecasting problem

The time series forecasting problem in this thesis is a grid loss forecasting problem from TrønderEnergi Kraft where for each day at 12.00, they need to forecast the grid loss for each hour the 24 hours the next day. Formally, at time t, the grid loss need to be forecasted for the interval $[t + 13, t + 36]$. Due to delays in the registration of data in the grid, the measured grid loss data are not reliable until six days after a measurement has been made, because the values are adjusted. In the real forecast scenario, at time t the last observable time point with reliable grid loss data is at $t - 144$, or six days before the prediction is to be made. However, to simplify the experimentation in this thesis, it is assumed that all data are available at the time of predictions, such that at time t at 12.00, all historical values up until time t is available. Then, the only values that are unknown are the future values from that point for grid loss and grid load. The forecasting problem then becomes a problem of forecasting the next 36 hours, and use the last 24 hours of those 36 as the forecast for the hours of the next day. This simplification does not change how the explanations are created or displayed, such that it should not affect the evaluation of the explanation methods.

5.3 Experimental setup

All code used for the experiments are written in the Python programming language. The code for the time series forecasting methods used in the experiments are from public repositories that are available on Github and as Python packages on the PyPi Python package index. Code was written and tested using Jupyter Notebooks. The versions of each of the packages are stated in Section 5.4 where each method is explained with hyperparameters.

The TFT method was the only method in the experiments that supported the use of GPUs, and the TFT models were trained on Google Colab using one GPU. The other methods were trained and tested locally on a laptop with an Intel Core i5-10210U CPU.

5.4 Experimental Plan

This section describes the experiments conducted during this research. It starts with a baseline that the rest of the experiments are compared against, and then describes experiments with the methods selected in Chapter 4. The purpose of the experiments is to give experimental evidence towards answering research question 3 in Section 1.3 regarding what is the best explanation approach for the forecasting problem.

5.4.1 Baseline model

A few simple baseline models for forecasting is used for comparison with the other models. A univariate autoregression model is used as baseline because it is interpretable as a linear combination of the p previous values of the target variable, and produces good results despite being a simple linear combination of previous values, without taking covariates or nonlinear relationships into account. If the other models do not produce more accurate predictions than the baseline, then one is better off using this simple interpretable model. Also, even if the other models do produce more accurate predictions, a comparison with the baseline is nevertheless useful to consider the tradeoff between accuracy and interpretability.

The autoregressive baseline model is tested using the ARIMA model in the Sktime Python package [Löning et al., 2019], sktime version 0.10.0. By specifying the p parameter as the window size of previous values in the autoregression, and setting parameters d and q to 0, for the integrated and moving average part of the ARIMA model, respectively, the ARIMA model becomes a simple autoregressive AR(p) model.

5.4.2 Forecasting method with post-hoc explanation

LightGBM is tested as forecasting method with SHAP as post-hos explanation method. Because SHAP is model-agnostic, LightGBM could be replaced by other forecasting methods, while giving the same explanations with SHAP. The LightGBM implementation used is the official implementation from Microsoft [Ke et al., 2017], lightgbm version 3.2.1. The SHAP implementation used is from the authors of the original SHAP paper, shap version 0.39.0. The SHAP implementation has built in support for explaining tree ensemble methods with an exact

algorithm, including for LightGBM, with the TreeSHAP method. This also supports other popular tree ensemble methods, including XGBoost and CatBoost, which makes it possible to replace LightGBM with any of the other methods while getting the same explanations for the predictions.

5.4.3 Interpretable models

Each of the selected methods will be tested using equal historical time window and equal prediction horizon. The selected forecasting methods with interpretable information are NeuralProphet and TFT.

NeuralProphet is tested using the original implementation from the authors of the model, neuralprophet version 0.3.2. The package is written in Python and built on top of the Pytorch open source machine learning framework [Paszke et al., 2019]. The NeuralProphet model is configured with trend, weekly seasonality, and daily seasonality enabled. Autoregression on the historical grid loss values is achieved by setting the n_lags parameter to the number of hours of previous grid loss data to use as input for each forecast. The forecasting horizon is set using n_forecasts parameter. The grid load feature is added as lagged regressor, and temperatur and demand are added as future regressors.

Temporal Fusion Transformer (TFT) is tested using the implementation from the Pytorch-forecasting framework [Falcon and The PyTorch Lightning team, 2019], pytorch-forecasting version 0.9.2, which is built on top of Pytorch lightning, which is a framework for high performance deep learning using Pytorch and is used as standard for submitting code to NeurIPS Reproducibility Challenge due to its favorable reproducibility abilities [NeurIPS, 2019].

5.4.4 Experiment 1

The first experiment tests the overall performance of the models compared to the baseline. Each model is trained on the entire training set, and then tested for each day in the test set by taking the window length of historical time steps until 12 at the day of prediction, and using the model to predict the 24 hours of the next day. In practice, that means predicting the next 36 hours, and using the last 24 hours of thos 36 hours as the predictions for the next day. This is repeated for all days in the test set. Then all predictions for all days in the test set are combined, and the MAPE of the entire test set is calculated, compared to the real data. This experiment provides the forecasting accuracy for each model, and is used to validate that the models do learn from the data and perform better than the simple baseline models.

The forecasting accuracy is also used to choose which configuration of each model to use in experiment 2, by training each model with different configurations and choosing the one with best forecasting accuracy. One important parameter is

the window size of historical data to use for each prediction. This is tested for one week, four weeks, and eight weeks of data. Also, some model specific parameters are tested with different values. The configurations used for each model is given in Chapter 6 for each result.

Finally, the forecasting accuracy can also be an important factor when evaluating which explanation method is the best. This is not necessarily based on the forecasting accuracy of this testing, but on the potential for improvements that can be made to the models in order to improve their prediction accuracy. How flexible the models are for improvements is important for that.

5.4.5 Experiment 2

The second experiment selects specific days from the test set based on the predictions, and the predictions and explanations are evaluated together with domain experts at TrønderEnergi. The first day in the test set is selected to test how well the models predict data that directly follows the end of the training set, and thus should not differ substantially from the patterns of the training data. Then, the last day of the test set is selected to test the data points that are furthest away from the training set.

In addition to the pre-selected days, the day with the worst MAPE for each of the methods is selected to test how the models explain the worst performing predictions. This is important because the need for explanations is greatest when the predictions deviates the most from the real values, as these are the predictions with the worst economic consequences. Also the day with the best MAPE is considered, and used to compare against the worst day.

This experiment is the main experiment that is used as experimental evidence towards answering research question 3. Evaluation of the explanations given by these models by domain experts at TrønderEnergi will make up a significant part of the final conclusion about which approach is best.

Chapter 6

Results

This chapter describes the results of the experiments outlined in Chapter 5. Experiment 1 tests each model on the entire test set and reports the forecasting accuracy. This experiment is also used to select which trained models to use for experiment 2. Experiment 2 examines explanations for forecasts of specific 24-hour periods.

6.1 Experiment 1

In this experiment, each model is trained on the training set, where each model is tested with the same window sizes, meaning the number of previous time steps to include in the input to the predictions. The tested window sizes are 168 hours, corresponding to one week, 672 hours, corresponding to four weeks, and 1344 hours, corresponding to eight weeks. Then, for each day in the test set, the given window size of previous data until 12 pm the day before is used as input, and the next 36 hours is predicted. The last 24 hours of that 36 hour period is used as prediction for the day that is predicted. When all days are predicted in the same way, the MAPE is calculated for the entire test set.

6.1.1 Baseline

The baseline for the experiments in this thesis is a simple autoregressive model, which predicts the next time step using a linear combination of the p previous values. This method is easy to interpret, by looking at the weights for each of the lagged values. In addition to the autoregressive baseline, also an even simpler method is included here, which is a one week persistance, in which the

Method	MAPE
1 week persistance	16.89
AR(24)	14.77
AR(48)	13.99
AR(72)	13.58

Table 6.1: Baseline methods and their respective MAPE error on the entire test set. AR means autoregressive, and the number in parenthesis means how many time steps of previous data is used for prediction.

Window length	MAPE
1 week	8.30
4 weeks	9.44
8 weeks	10.09

Table 6.2: The table shows the forecasting error on the entire test set of of the TFT method, with one, four, and eight weeks of historical data as input to the predictions.

prediction for a given time step is the value of the same hour one week earlier. The performance of the baseline methods is shown in Table 6.1.

The autoregressive baseline was tested with 24 hours, 48 hours, and 72 hours of data. The computational time increased significantly with 72 hours of data, and the model is not suited for very large amounts of data. This baseline provides a basis for comparison, both in terms of forecasting accuracy, and for explainability, as described in Section 6.2.1.

6.1.2 TFT

The TFT model was tested with one, four, and eight weeks of historical data as input. As shown in Table 6.2, the TFT model performed best with one week of historical data as input to each prediction, compared to four or eight weeks of historical data, with a MAPE of 8.30. The fact that the model performs worse when predicting with more data might indicate that the conditions change for grid loss over time, such that a longer time window introduces more deviations from the current conditions. It can also be due to the model not being able to learn long term relationships well enough. However, that seems unlikely as one of the strengths of the model reported in the original paper is that it is better at learning long term relationships than recurrent or convolutional networks.

The best MAPE result for TFT is 8.30 for one week of historical data. This is significantly lower than the baseline results in Section 6.1.1. This indicates that

Parameters	MAPE
1 week, linear	8.89
1 week, neural	8.81
2 weeks, linear	9.06
4 weeks, linear	9.36

Table 6.3: The table shows the forecasting accuracy of NeuralProphet on the test set with different configurations. The number of weeks indicate the window size of historical data used for each prediction, and linear or neural means if the model uses either linear model or neural network model for autoregression, and for lagged and future covariates.

the model has learned some relationships within the data, such that explanations of the model should be able to find importance in the data that is not random. Since the model trained with one week of historical data for each prediction performed best in this experiment, this model will be used as the TFT model in experiment 2, reported in Section 6.2.2.

6.1.3 NeuralProphet

NeuralProphet was also tested with one week, four weeks, and eight weeks of data with both linear and neural network regression. Also two weeks of data were tested. The results for some of the configurations is shown in Table 6.3. The table does not include all tested configurations, but those with longer window sizes performed worse than the ones reported in the table. The forecasting accuracy results for NeuralProphet were similar to those of TFT, in that the MAPE is close and that the results are worse for longer window size than for one week. As the table shows, the best performing model configuration was using one week of historical training data, and using neural networks for autoregression, lagged regressors and future regressors. The configuration for the best performing method had neural networks with two hidden layers and a dimension of 32 for each of the hidden layers.

6.1.4 LightGBM

LightGBM was tested with one week, four weeks, and eight weeks of previous data, in the same way as for the other models. The performance of the LightGBM models on the test data, given by the forecasting accuracy MAPE, is given in Table 6.4. As the table shows, the performance was best with one week of historical data, compared to four or eight weeks. The historical data for each feature was added by a separate column in the dataset for each lag. This led

Parameters	MAPE
1 week	8.75
4 weeks	9.52
8 weeks	10.05

Table 6.4: The table shows the forecasting accuracy on the test set measured by MAPE by LightGBM using different lenghts of previous data.

to a large number of values for a single row, since each prediction were treated as a multivariate regression problem, with the lagged features as input and the grid loss as target. It might be possible that the model was not able to learn the relationships between all the features, since there were a large number of values for each row. There might be other ways to configure the LightGBM method, but this was what was achieved during the work on this project.

LightGBM was tested with two different approaches for multi-horizon forecasting, both recursive and direct. The recursive approach means that the prediction horizon is predicted one step at a time, and the prediction of one time step is used as lagged feature for the next. For the direct approach, 24 LightGBM models were trained, one for each of the 24 hours the next day that is to be predicted at noon each day. The direct approach produced the best forecasting accuracy, and are used for all the results in this chapter.

6.2 Experiment 2

Section 6.1 showed the forecasting accuracy of the selected models for different model configurations. This section describes explanations from the best configuration of each of the models, in particular for the days with the best and the worst forecasting accuracy. As stated in Section 5.4.5, the first and last days of the test set where tested, as well as the best and worst day in terms of forecast accuracy. Since the best day in most situations were one of the first days of the test set, and the worst was mostly one of the last days, the first and last day for each of the models is not included here. The best and worst day of prediction together with explanations were shown to domain experts at TrønderEnergi. The worst days are displayed in this section.

6.2.1 Baseline

For the autoregressive baseline models, the explanation is equal for each prediction, as it is a simple linear combination of the previous p values. To interpret a prediction of the autoregressive models, one can look at the weights of the lagged

values. For instance, Figure 6.1 shows the weights of the AR(24). It is clear from the plot that the last value, with lag 1, has the largest weight, and is thus most important for the prediction of the next value. Further, one can see that some time steps are more important for the prediction than others, indicated by a larger weight. When investigating a specific prediction, one can examine each component to see which lags contribute most to the prediction. The contribution from a previous time step is the value of that time step multiplied with the weight of that lag as seen in the figure.

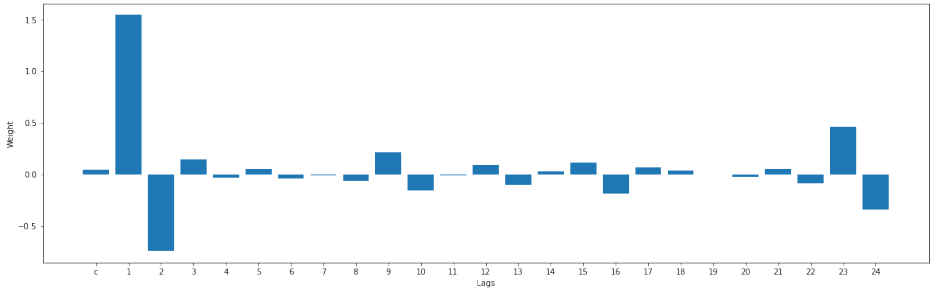


Figure 6.1: Plot of the weights from the AR(24) baseline model, which predicts the next values by taking the linear combination of the 24 previous time steps. c is the intercept, and the numbers on the x axis indicates the lags. For a prediction at time t, 1 means $t - 1$ and similar for the other lags.

6.2.2 TFT

The TFT model outputs two types of explanations that is used to interpret predictions. The first is attention, which highlights how important each of the time steps in the historical window is. The second is variable importance, which quantifies how important each of the input variables in the encoder is, and how important each of the known future values are in the decoder. The day with the worst predictions is especially interesting to study, since that is the day with the largest financial consequences. A plot of the day with the worst MAPE in the test set is shown in Figure 6.2. Plot (a) in Figure 6.2 shows the window with one week of historical grid loss values in blue. Then, the orange line shows the predictions for the 24 hours of the next day, and the green is the true values. One can see from the plot that the predicted values have the same shape as the true values, but all values are too low. This could be because the predicted day was a saturday in a weekend with multiple holidays, such that the values deviated from normal values. It could also be that the data is towards the end of the test set and thus far away from the training set, meaning that the conditions might have

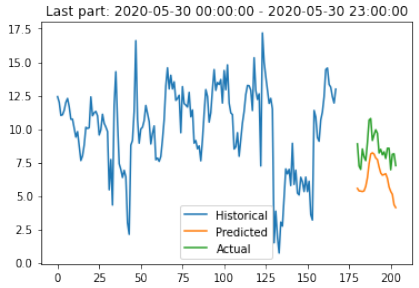
changed in the time between the training set and the predicted day. The attention plot in (b) shows that there are certain time steps that are marked as much more important than the other time steps. These time steps can be interesting to look closer at in an analysis of why the predictions deviates from the true values. Plots (c) and (d) show variable importance from historical data in the encoder and future known data for the prediction horizon in the decoder, respectively. The domain expert from TrønderEnergi evaluated that this distribution is consistent with their experience working with this dataset.

In general, the domain experts found that the model output useful information. Both the attention and the variable importance are interesting information to get an understanding of how the data was used. One could also see that the ranking of variable importance in this worst day in May and the best day, which was during the winter, that the load was more important than the temperature in May, but the other way around in the winter. This is consistent with previous work on this data.

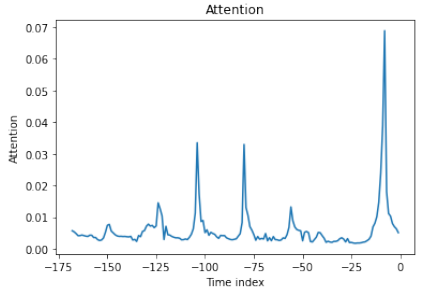
6.2.3 NeuralProphet

NeuralProphet is an additive model, such that the prediction for a time step is the sum of several components. The information that NeuralProphet provides for a prediction is therefore how much of the prediction each of the components contributed to. This NeuralProphet model consists of seven components, namely trend, weekly and daily seasonality, autoregression of previous values of the target variable grid loss, lagged values of grid load, and future known values of temperature and demand. This information can be visualized in different ways, either a separate plot for each component for the entire prediction horizon, or as a bar plot for a single time step with each of the components as separate bars. Figure 6.3 shows each component separately for the 24 hours of the day with the worst MAPE, and Figure 6.4 shows each component in a bar plot for the first hour of those 24 hours. Both plots are described below.

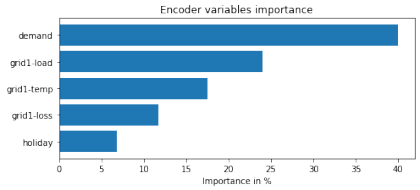
The additive components of NeuralProphet differs from the variable importance of TFT or SHAP in that the components do not quantify how important a component is, only how much it adds to the total sum. Also, for different days of prediction, the relative size of each component compared to the others could be completely different, which indicates that relying only on the size of the components to explain the predictions might be insufficient. This was the case when considering the plots from the worst and best day, where the relative contribution from autoregression of grid loss and lagged regression of grid load switched completely. For the worst day, as shown in Figure 6.3, autoregression of grid loss, named ar in plot (e), has much lower values than lagged regressor grid load in (f), whereas for the best day, it was complete opposite. This made it unclear to the



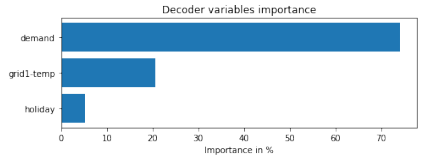
(a) Plot of the forecast. The blue line is the historical time steps for grid loss used, the red is the forecasted values, and the green is the true values for the forecast period.



(b) The attention plot show which time steps in the historical data is most important, where time index 0 is the latest observation. The figure shows clear spikes.



(c) Variable importance of the historical input data in the encoder.



(d) Variable importance of the future known variables in the decoder for the time steps in the prediction horizon.

Figure 6.2: Plots of the predictions of TFT model on the day in the test set with worst MAPE. (a) shows the predictions, (b) shows the attention, and plots (c) and (d) show variable importance for historical and future known time steps, respectively.

domain expert from TrønderEnergi to what extend the information gave insight into how the model actually works. Using the model parameters could help get a clearer picture of the contribution from specific time steps and get more insight into why the components have the size they have, but in general the interpretable information from NeuralProphet might not be as useful as expected.

Figure 6.3 shows the day that the NeuralProphet model performed worst, where plot (a) show the predicted values and the actual values for those 24 hours, and plots (b) through (h) shows the components that make up the prediction for each time step. The day is 28th of May 2020, and the MAPE for those 24 hours is 60.56, much larger than the total MAPE for the whole test set of 8.81. In the other end, the best performing day was 23th of December 2019, with a MAPE of

2.28. Figure 6.4 shows for the first time step in the same 24 hours how much each component contributes to the sum in a way that is easier to compare. Considering the scale of each component, it is clear that the trend, previous grid load, and future known temperature and demand contributed the most, where temperature contributed negatively and the other three positively towards the sum. The autoregressive component with grid loss and the seasonalities contributed comparatively little towards the predictions. One can see clear daily variation in multiple of the components that is consistent with domain knowledge, such as in plot (g) in Figure 6.4 where the electricity demand component increases steeply in the morning and then falls throughout the day, and in (h) for temperature with the opposite effect with larger temperatures leading to smaller grid loss.

6.2.4 LightGBM with SHAP

The explanation for a single predicted time step with LightGBM explained by SHAP shows how much the most important features move the prediction away from the expected value, which is the average value of from the training set. The domain expert from TrønderEnergi evaluated this method as informative, as it shows how much the value of each feature make the prediction deviate from the expected value. This explanation can be regarded as contrastive, that is one of the favorable properties of an explanation described in Section 2.2. The explanation says that the prediction would have been the expected value, if it had not been for the value of each feature, and quantifies the difference that each feature make.

The worst performing day for LightGBM, using one model for each of the 24 hours to predict and one week of lagged data, was 28th of May 2020, with a MAPE of 43.65. Figure 6.5 shows a plot of the feature contribution for the prediction of the first hour of that day in plot (a), and the mean feature contribution on prediction of the first hour of all days in the test set in (b). The plots show that the most important features are mostly the same for the specific prediction and for the mean, but the contributions are significantly different. By comparing the values for the contributions, one can get an explanation of why the prediction has its value. The expected value that is the average value of predictions over the training set, and the contributions say how far away from that expectation each feature make the prediction. As the plot shows, the demand and grid loss 24 hours before the prediction are the two most important features for this model, and they contribute much more for the prediction from the worst day than the mean.

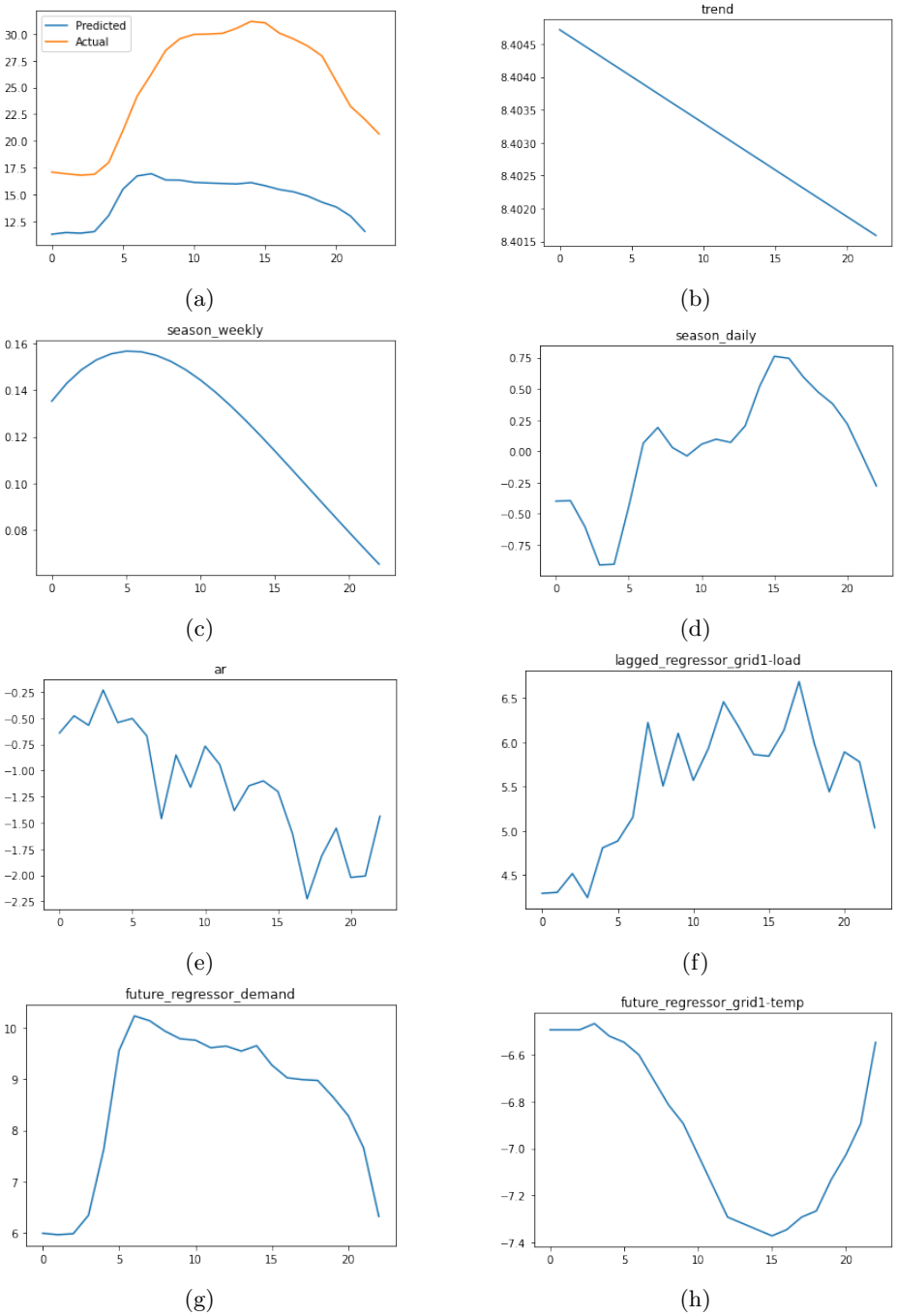


Figure 6.3: Prediction of the day with worst MAPE for NeuralProphet, with (a) showing predicted and actual values for those 24 hours, and (b) to (h) showing the components.

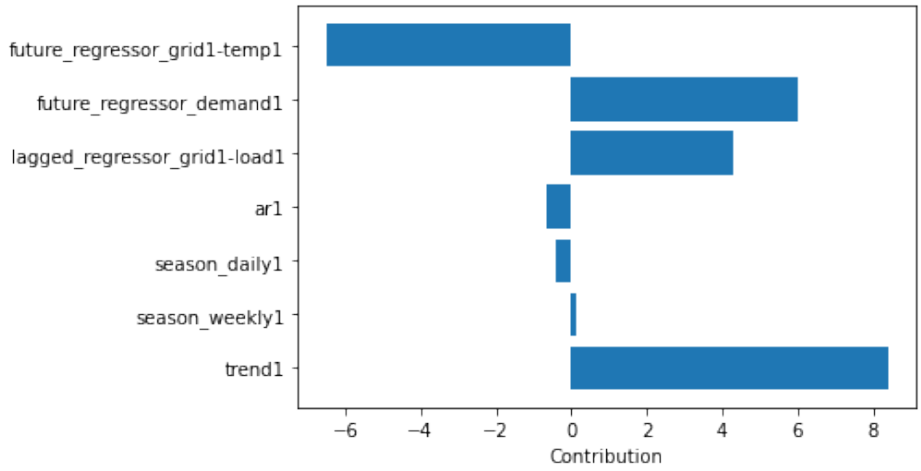
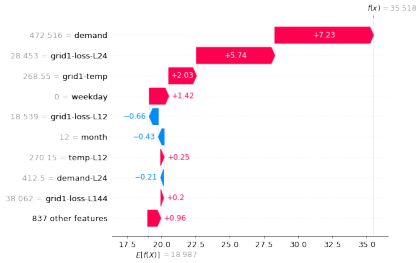
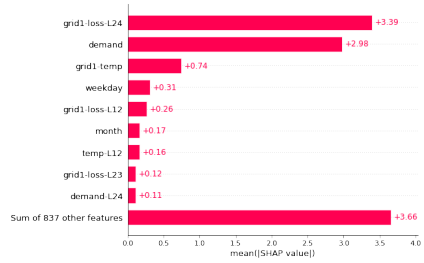


Figure 6.4: The bar chart shows how much each component in the NeuralProphet model contribute towards the prediction for the first hour of the day with the worst MAPE.



(a) Contribution to the prediction of the first hour of 28th of May 2020, the day with the worst MAPE.



(b) Mean contribution from the most important features to the prediction of the first hour of each day for the entire test set.

Figure 6.5: Plots of feature contribution made by SHAP for LightGBM model. (a) shows for one prediction, whereas (b) is mean over all predictions.

Chapter 7

Evaluation and Conclusion

This chapter presents a discussion of the results from the experimentation of this thesis, reported in Chapter 6. Then, it provides a summary of the answers to the research questions. A conclusion of the work is presented. Finally, suggestions for future work is described, including newly published methods that could not be included in this project, abandoned research directions, and an alternative problem description that was discovered towards the end of the project.

7.1 Evaluation and discussion

Three different approaches were tested in the experiments in this project. TFT is a complex neural network architecture with built in interpretability. NeuralProphet is an additive forecasting model where one can examine each component individually. LightGBM is a prediction model, which was used together with SHAP, that produces feature importance scores, that quantifies how far from the average over the training set the value of each feature take the prediction. Chapter 6 has provided results from the experimentation. Each of the three approaches produced forecasts better than the baseline methods as expected, and with similar accuracy. However, they produce different explanations.

The domain expert from TrønderEnergi rated the explantions from SHAP as most useful, followed by those from TFT, whereas the components from NeuralProphet were unclear how useful they are. The SHAP ranks the features by how important each was towards a prediction, and quantifies its contribution towards the prediction. When each time step is added as lagged features for a given prediction, one gets how important each time step for each feature was. The SHAP explanations can be regarded as contrastive, one of the desired properties of explanations reported in Section 2.2. TFT, on the other hand, has a separation

between attention, which points at important time steps, and variable importance, which quantifies how important each feature is. The variable importance only says how important a feature is in total, but not how it contributes to the prediction or how each time step is contributing. Thus, SHAP seems more useful to be able to investigate what contributed to a specific prediction.

Another aspect separating the models is in how flexible they are for improvements or replacement. For the TFT model, it is in practice hard to make adjustments, other than changing the hyperparameters, due to its complex structure. NeuralProphet is additive, which means that components can be removed or added as needed, and it is built on Pytorch which is a widely known framework for machine learning in Python. This means that it is possible to make changes as wanted. The approach with SHAP is the most flexible, as it is separated from the prediction model, and is model-agnostic, meaning that many different prediction models can be used with SHAP without needing to make any adjustments. It does also not require any knowledge of the structure or code of the prediction methods to change method, because SHAP can be applied on other forecasting models by a few lines of code.

Considering that SHAP produces the most informative explanations according to domain expert at TrønderEnergi, and the fact that the SHAP approach is more flexible because any forecasting method can be applied together with SHAP, the approach with SHAP is the best approach of the ones examined in this thesis. More generally, the results in this thesis indicates that intrinsically interpretable forecasting models, that combine forecasting with providing internal information for explanations, do not have an advantage over separate prediction and explanation models, that was one hypothesis during the work with this thesis. This can not be said to be valid for all methods based on the results of experiments here, but for the experiments made during the work with this thesis, and all methods considered during the work with the state of the art, it can be said that although interpretable models have information about internal workings available, this has not resulted in better explanations than what can be achieved with separate XAI methods.

A threat to the validity of the results in this thesis is the choice of hyperparameters. The models have been tested by several different hyperparameters, but there might be other hyperparameters and configurations that could change the results. It is therefore possible that the models would have been ranked in a different order if only focusing on forecasting accuracy. However, as the methods tested in the experimentation did provide different types of explanations, which was the focus of the evaluation by domain experts from TrønderEnergi, the ranking would still be the same.

In general, the most clear result that can be made from the work in this thesis is that for a forecasting problem the best approach to make predictions

and create explanations is to have separate prediction and explanation methods. This is especially the case for a forecasting problem in which the forecasting accuracy is of high importance.

7.2 Contributions

The contributions of this thesis is two parts. The first is a literature review of important methods within the field of XAI, both general methods and specifically for time series data. For time series data, a focus was on complex forecasting methods with interpretable information built into it, which is a category not always included in categorizations of XAI methods. The second contribution is experiments with three different approaches, with evaluation by a domain expert of how useful each of the approaches are.

The main conclusion from this work is that the best approach to make predictions and create explanations at the same time is to have separate prediction and explanation methods, and not a combined approach in which a complex forecasting method outputs information about its inner workings. Specifically, the explanations made by the SHAP model provide useful contrastive explanations that give insight into how much each feature impact each prediction.

7.3 Future Work

During the work of this thesis, new methods have been published that could have been interesting to test in the experimentation in this thesis, but were unavailable at the time of selection. Notably, the source code for the TimeSHAP method, described in Section 3.4, was published during the experimentation part of this work. Also, the extension of N-BEATS, called N-BEATSx, described in Section 3.5, was published in the final stage of this work. This method, which includes exogenous variables and improves the accuracy of N-BEATS, could be interesting to test. Also, the well known LIME method could be interesting to compare to the explanations from SHAP, to see which of the explanations are most useful for domain experts.

One idea that was initially explored during the project, but was abandoned, was to identify multiple parts of the sequence that had the same characteristics, and compare explanations for those similar situations to see if the explanations are also similar. The idea was that if similar conditions appear multiple times, they should have similar explanations.

This thesis has focused on forecasting of a target variable into an unknown future, and how to provide explanations of how those predictions. Although this

is the approach that is needed for a real time application, there is another type of problem that can be investigated when one has access to a complete dataset as the one used in this project. Instead of predicting the future, calculating the error and find an explanation of the prediction, one can first make a prediction, then give the real result to the model, and ask what would need to change to make the correct prediction. This is an interesting area of research for TrønderEnergi and could have been a focus of the research of this thesis, but the use case was not identified and described until the end of this project. Identifying that this is maybe even more interesting for TrønderEnergi to investigate than the problem of this thesis is one of the outputs from the work of this thesis. It might be a good starting point for a future master's project.

Bibliography

(2019). Worldwide spending on artificial intelligence systems will be nearly $98 billion in 2023, according to new idc spending guide. https://www.businesswire.com/news/home/20190904005570/en/Worldwide-Spending-on-Artificial-Intelligence-Systems-Will-Be-Nearly-98-Billion-i Accessed: 2022-06-13.

Adadi, A. and Berrada, M. (2018). Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE Access*, 6:52138–52160.

Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., and Herrera, F. (2020). Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115.

Bento, J., Saleiro, P., Cruz, A. F., Figueiredo, M. A., and Bizarro, P. (2021). Timeshap: Explaining recurrent models through sequence perturbations. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2565–2573.

Doshi-Velez, F. and Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Falcon, W. and The PyTorch Lightning team (2019). PyTorch Lightning.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.

Goldstein, A., Kapelner, A., Bleich, J., and Pitkin, E. (2015). Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *journal of Computational and Graphical Statistics*, 24(1):44–65.

Gunning, D. and Aha, D. (2019). Darpaâs explainable artificial intelligence (xai) program. *AI magazine*, 40(2):44–58.

Hyndman, R. J. and Athanasopoulos, G. (2018). *Forecasting: principles and practice*. OTexts Melbourne.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30.

Lim, B., Arık, S. Ö., Loeff, N., and Pfister, T. (2021). Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764.

Linardatos, P., Papastefanopoulos, V., and Kotsiantis, S. (2021). Explainable AI: A review of machine learning interpretability methods. *Entropy*, 23(1):18.

Löning, M., Bagnall, A., Ganesh, S., Kazakov, V., Lines, J., and Király, F. J. (2019). sktime: A Unified Interface for Machine Learning with Time Series. In *Workshop on Systems for ML at NeurIPS 2019*.

Lundberg, S. M., Erion, G. G., and Lee, S.-I. (2018). Consistent individualized feature attribution for tree ensembles. *arXiv preprint arXiv:1802.03888*.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 4768–4777. Curran Associates Inc.

Makridakis, S., Spiliotis, E., and Assimakopoulos, V. (2022). M5 accuracy competition: Results, findings, and conclusions. *International journal of forecasting*.

Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38.

Mohseni, S., Zarei, N., and Ragan, E. D. (2021). A multidisciplinary survey and framework for design and evaluation of explainable ai systems. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 11(3-4):1–45.

NeurIPS (2019). Reproducibility challenge. `https://reproducibility-challenge.github.io/neurips2019/resources/`. Accessed: 2022-06-12.

Olivares, K. G., Challu, C., Marcjasz, G., Weron, R., and Dubrawski, A. (2022). Neural basis expansion analysis with exogenous variables: Forecasting electricity prices with nbeatsx. *International Journal of Forecasting*.

Oreshkin, B., Carpov, D., Chapados, N., and Bengio, Y. (2020). N-beats: Neural basis expansion analysis for interpretable time series forecasting. *ICLR*.

Oreshkin, B. N., Dudek, G., PeÅka, P., and Turkina, E. (2021). N-beats neural network for mid-term electricity load forecasting. *Applied Energy*, 293:116918.

Pan, Q., Hu, W., and Chen, N. (2021). Two birds with one stone: Series saliency for accurate and interpretable multivariate time series forecasting. In Zhou, Z.-H., editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 2884–2891. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "Why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2018). Anchors: High-precision model-agnostic explanations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

Saadallah, A., Jakobs, M., and Morik, K. (2021). Explainable online deep neural network selection using adaptive saliency maps for time series forecasting. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 404–420. Springer.

Schlegel, U., Lam, D. V., Keim, D. A., and Seebacher, D. (2021). Ts-mule: Local interpretable model-agnostic explanations for time series forecast models. *arXiv preprint arXiv:2109.08438*.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626.

Shrikumar, A., Greenside, P., and Kundaje, A. (2017). Learning important features through propagating activation differences. In *International Conference on Machine Learning*, pages 3145–3153. PMLR.

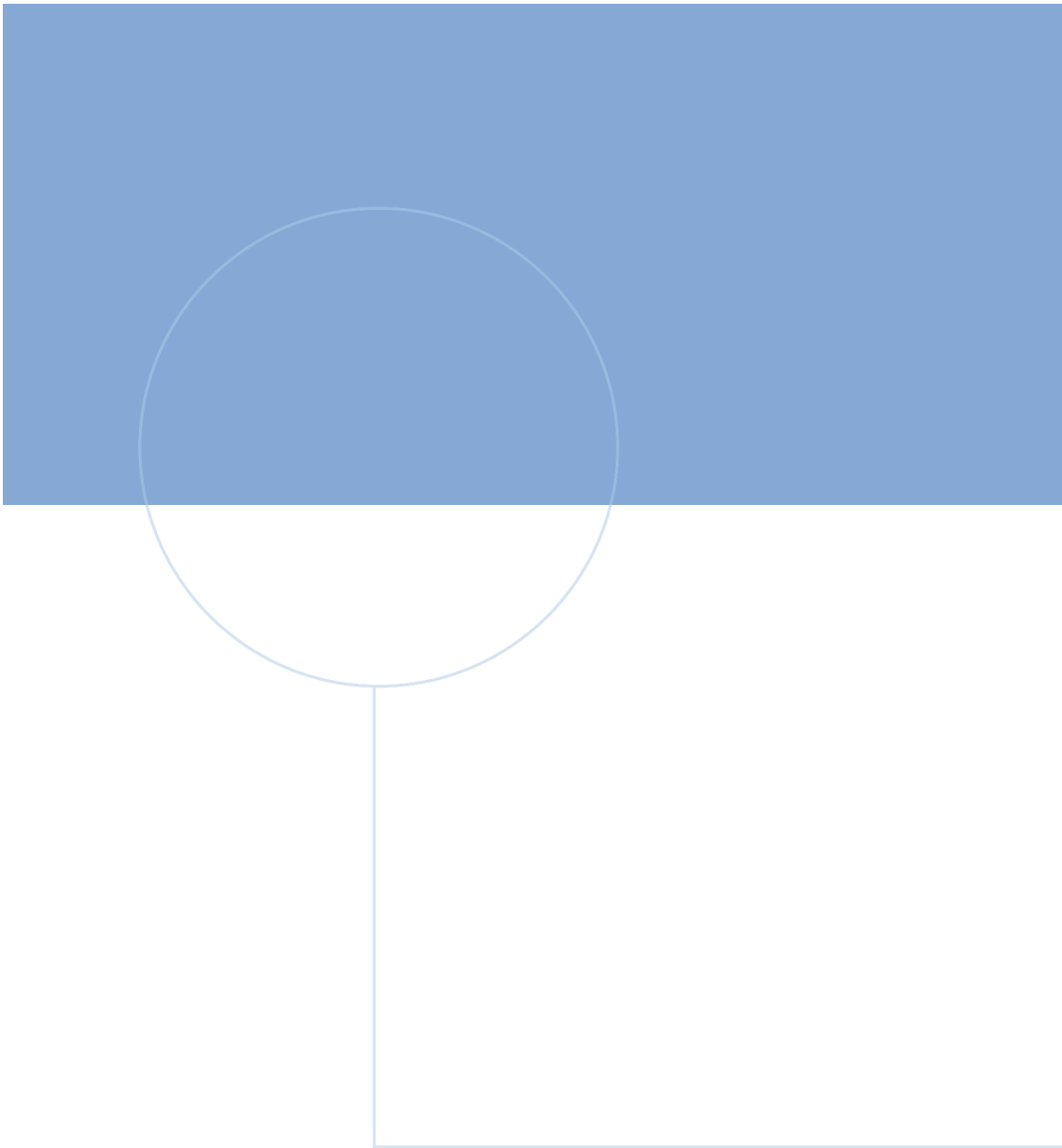
Sixt, L., Granz, M., and Landgraf, T. (2020). When explanations lie: Why many modified bp attributions fail. In *International Conference on Machine Learning*, pages 9046–9057. PMLR.

Taylor, S. J. and Letham, B. (2018). Forecasting at scale. *The American Statistician*, 72(1):37–45.

Triebe, O., Hewamalage, H., Pilyugina, P., Laptev, N., Bergmeir, C., and Rajagopal, R. (2021). Neuralprophet: Explainable forecasting at scale. *arXiv preprint arXiv:2111.15397*.

Triebe, O., Laptev, N., and Rajagopal, R. (2019). Ar-net: A simple autoregressive neural network for time-series. *arXiv preprint arXiv:1911.12436*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.



NTNU

Norwegian University of
Science and Technology