Building Electricity Demand Forecasting with Temporal Hierarchies

by

Lorenzo Donadio

Master Thesis

Department of the Built Environment
Eindhoven University of Technology TUe

Supervisors:

Julien Leprince, TUe
Prof. J.L Scartezzini, EPFL
Prof. Wim Zeiler, TUe
Dr. Roberto Castello, EPFL

Eindhoven, June 2021

Acknowledgments

During the entire process of this master thesis I have received a great support and assistance.

I would first like to thank my supervisor, Julien Leprince, whose expertise and continuous oversight was of paramount importance in the completion of this thesis. I would also like to acknowledge my supervising professors, Wim Zeiler and J.L. Scartezzini for their support, enthusiasm and pertinent remarks for this thesis. Special thanks Roberto Castello for his relevant feedback and advise.

There must be a particular acknowledgement for the partner company, BAM Energy Systems. For providing all the data used in this research. BAM also facilitated the completion of this thesis through supervision and advise, specially given by Dennis Van Goch.

Abstract

Accurate building electricity load forecasts play a major role in the energy transition, as they facilitate flexibility deployment, grid stability and overall reduce costs and CO_2 emissions. This research leaverages forecasting and reconciliation of temporal hierarchies to achieve coherent and accurate forecasts for different time scales. Two different hierarchical structures and their variants were studied, a daily and hourly hierarchy. Four different forecasting algorithms were implemented and compared, as well as a variety of reconciliation methods, their improvements of base forecasts were quantified and assessed, with up to ten percent accuracy gains for certain levels of the hierarchical structure. This thesis constitutes a comprehensive comparison of and the developed tools for temporal hierarchical forecasting and reconciliation.

Contents

1 Introduction

The worlds final consumption of electricity by the building sector, both residential and commercial, has been consistently increasing for decades [1]. In a time where the energy transition is a crucial challenge, the field of building energetics continues to gain importance in the worldwide energy market. According to the International Energy Agency [IEA, 2016], The global building sector energy demand amounts up to 30% (120 EJ) of total final energy consumption in 2013. And the building electricity consumption has more than doubled since 1990. In contrast, the energy use per square meter has been at a steady but slow decrease since the 90's, this trend encourages the building sector to increase efficiency.

In Europe, according to a directive of the European Parliament,[EU, 2010], the building sector accounts for 40% of total energy consumption and 36% of total CO_2 emissions. This directive also states that the member states must ensure that *by 31 December 2020, all new buildings are nearly zero-energy buildings.* The guidelines on the future of building energetics are thus clear, there must be a reduction of energy consumption and an increased use of energy from renewable sources in the building sector. In particular there are important European targets to be reached by 2030 [2], such as a renewable share of 32% and a similar increase in energy efficiency, the built environment must play a crucial role in reaching those goals.

In the Netherlands [IEA, 2020], the current electricity generation is dominated by natural gas, accounting for around 50% of total production. The second source of electricity are coal fired power plants, despite the sharp decrease in coal since 2015, the sector still provides 27% of the electricity. In 2018, only 16.5% of electricity came from renewable sources; mainly wind and solar photo-voltaic (PV) [3], which have seen a substantial increase in recent years, and this trend is expected to continue.

The Netherlands signed a Climate Act in 2019 [4] which sets targets to reduce Green House Gas (GHG) emissions by 49% by 2030 and by 95% by 2050. The achievement of these goals would require at least 70% of renewable electricity generation. The Climate Act also calls for a sharp increase in vehicles powered by electricity or hydrogen, and shift towards electrified heating and industrial processes. This entails major changes in the current energy system.

To assist in these needed transformations, the Dutch government is supports the development of a digitalized energy system which enables a very high penetration of variable renewable energy, and also enhances coordination between electricity networks. A noteworthy example of this support is the mandate that all Dutch households be equipped with a smart meter, this shows the importance

[1] https://www.iea.org/reports/key-world-energy-statistics-2020/final-consumption

[2] https://ec.europa.eu/clima/policies/strategies/2030$_en$

[3] https://www.iea.org/countries/the-netherlands

[4] https://www.government.nl/topics/climate-change/climate-policy

that energy flexibility and data driven services will have in the coming years.

Flexibility in power systems is capacity to balance supply and demand, operate continuously in unexpected situations, and handle uncertainty on both supply and demand [Impram et al., 2020]. The supply uncertainty is bound to increase, due to the trend of power systems to have increasing renewable energy penetration, the study of generation uncertainty and availability of renewables is a very extensive field, this thesis focuses on its complementary the energy demand and more specific; electricity demand forecasting.

The more a building is able to respond to the grid's need for flexibility, the better [Junker et al., 2018]. Thus, smart buildings rely on accurate forecasts to make smart operational decisions, and by making buildings smarter, it becomes possible to consume locally generated energy when it is available. This provides benefits for the grid as well as the building clusters, resulting in overall in energy savings, increasing peak shaving capabilities, and a more efficient use of locally generated energy. Smart buildings are equipped with data gathering devices and digital infrastructures and thus the amount of data is constantly increasing, this increases the need for intelligent building management systems, energy efficiency measures and adaptive energy systems. Building data could become an event greater asset when utilizing the appropriate tools and frameworks. Building load forecasting is a key building block for most or these tools; flexibility deployment, building operations, management of energy storage systems, electricity purchase and schedule, among many others.

BAM is part of group of ten leading Dutch companies and institutions, which understood the importance of the aforementioned concepts. The consortium - called TROEF [5] - is starting a collaboration that aims to accelerate the energy transition in the Netherlands by leveraging the *Internet of Energy*. TROEF is currently developing a new layered energy ecosystem, that will serve as a platform where sustainable energy can be optimally and transparently exchanged between buildings and areas. The goal is that implementation of this platform will minimize CO2 emissions.

Companies, which aim to be better energy service provider (such as BAM) and improve their data-driven services, would greatly benefit from better forecasts. Because an improved forecasting accuracy leads to reduced imbalance and allows for optimization of dispatch planning, it also allows buildings to provide grid services. These factors are important particularly when regarding a framework such as TROEF. The main offices of BAM, located in Bunnik, are a so called *living lab*. They are equipped with a variety of sensors, batteries, solar PV, electric vehicle charging. This space is optimal for performing a proof of concept of new developments on forecasting and other building related technologies.

There are several applications of energy demand forecasting ranging from country level to building and appliances level, at different granularities and with

[5] https://www.troef-energy.nl

different forecasting horizons Among the applications of building demand forecasting are: medium to long term planning, with seasonal changes of varying demand, day-ahead planning with variable demand forecasts, intraday rescheduling demand with updated forecasts, and high frequency balancing of forecasts errors for grid stability. The two uses of electricity demand forecast that have immediate repercussions, on the Bunnik offices for example are: day-ahead electricity purchase and stabilizing the intraday or imbalance market. And that will be the focus of this thesis, short term forecasting, ranging from a hourly to daily forecasts at different granularities.

1.1 Building load forecasting

The world of time series forecasting has to be specially thankful to Rob J Hyndman and George Athanasopoulos, authors of *Forecasting: Principles and Practice* [Hyndman, 2018] and a staggering amount of other articles and blogs on forecasting. This book is one of the main pillars for time series forecasting, it is a comprehensive introduction to a variety of methods ranging from time series decomposition to advanced forecasting algorithms. This thesis mainly incorporated the contents from the chapters regarding Exponential Smoothing and ARIMA. Although this book uses the forecasting package of R, many existing Python forecasting libraries, which was used, cite it as the main reference and were built on top of Hyndmans work.

The number of studies on prediction model analysis is extensive, with numerous scientific publications as well as online forecasting competitions with thousands of participants, take for example the Kaggle M competition. [Hyndman, 2020] reviews the history of forecasting competitions, discusses the lessons learned about their implementation, and what they can teach us about forecasting. For a more in-depth review of the M4 competition, [Makridakis et al., 2020]

There is a collection of methods for forecasting and modeling of building energy demand, but they all fall into three main categories: physics driven models or white-box, hybrid models or gray-box and data-driven models or so called black-box models. This thesis will focus on a data-driven approach to forecasting. [Bourdeau et al., 2019] provides an extensive review of most of the existing black-box methods with detail explanation on each algorithm, a description of the different types of input data in the reviewed studies and a clear review of the metrics used in those studies. [Ghalehkhondabi et al., 2017] also provides a consistent overview of forecasting models published between 2005 and 2015 by categorizing the reviewed papers according to the type of the prediction model used in the research. This review also shows that the building demand forecasting is a topic that has gained interest in recent years, with a sharp increase in the number publications containing 'electricity', 'load' and 'power' in the title.

Classical time series forecasting methods only take into account one particular time series, but is reality there are time series that present a hierarchical structure,

or different patterns depending on the series granularity. In order to leverage this aspects of building load data, this thesis will focus on the implementation of temporal hierarchies for time series forecasting. A hierarchical time series (HTS) is a collection of time series that follows a hierarchical aggregation structure. The most classical example is the spatial hierarchy, but in this research, a temporal aggregation was appropriate; by hour, day, week, etc.

Electricity demand exhibits considerably different characteristics on different time scales. Therefore, the use of HTS for forecasting is interesting since it considers the characteristics of different time scales in the forecasting procedure

When temporal aggregation is applied to a time series it can strengthen or attenuate different elements. Non-overlapping tem- poral aggregation is a filter of high frequency components, there- fore at an aggregate view low frequency components, such as trend/cycle, will dominate. The opposite is true for disaggregate data, where potential seasonality may be visible. Therefore, tem- poral aggregation can be seen as a tool to better understand and model the data in hand. When forecasting a hierarchical time series many independent forecasts are produced, those forecasts must be reconciled to ensure their coherency and in doing so hopefully improving their accuracy.

The reconciliation of base forecasts, has been a field under active development. The two classical approaches are bottom-Up and top-Down, but these two are not optimal. The introduction of an optimal reconciliation method was first published by [Athanasopoulos et al., 2009]. The authors used a hierarchy based of spatial an categorical aggregation. The time series of the number of tourists in Australia was aggregated by four different travel purposes, seven different Australian regions and whether their destination was a capital city or not, and thus they constructed a hierarchical tree. This case study of Australian tourism data will be recurrent in a large portion of studies regarding hierarchical forecasting. The authors also demonstrate the performance of the proposed method for optimal reconciliation of base forecasts. This novel method proved superior to the conventional bottom-up and top-down methods, but by a slight margin. This prompted a series of publications aimed at improving this optimal reconciliation method. For example [van Erven and Cugliari, 2015] introduced a Game-Theoretically optimal reconciliation method, that is proven to only improve a given set of forecasts. This method was tested on simulated and real energy demand data.

The very concept of temporal hierarchies was introduced in this paper [Athanasopoulos et al., 2017], which lays out the ground for forecasting with temporal hierarchies. The authors constructed a temporal a hierarchy by aggregating in a temporal manner. A temporal hierarchy can be constructed for any time series by means of non-overlapping temporal aggregation. Previous to this study, forecasting and reconciliation of only spatial or categorical hierarchies had been explored. But the principle remains the same; the predictions for all aggregation levels are combined to result in temporally reconciled forecasts for the different selected granularities. The aforementioned paper is thus the biggest point of reference for

the following studies on temporal hierarchies.

Some heuristic alternatives to the reconciliation procedure were introduced by Fonzo and Girolimetto [2020]. The authors provide an iterative reconciliation framework which extends the optimal reconciliation procedure, and overcomes some of its weaknesses. In contrast [Jeon et al., 2019] proposed a probabilistic approach to reconciliation, which includes the use data-driven weights which are calculated via cross-validation in order to maximise statistical properties of the reconciled forecast distributions.

Yet another example of another hierarchical forecast strategy is presented in [Pennings and van Dalen, 2017]. The authors provide a, so called, integrated hierarchical forecasting framework, where different than traditional methods, the authors explore the use of all the series time series at once instead of selecting series from parts of the hierarchy for forecasting. The authors also use the structure of the hierarchy itself on the the data-generating process and instantaneously generate forecasts for all levels of the hierarchy, through state space models and Kalman filtering. The case study was the aggregation of food products, and the extension of this approach to temporal hierarchies is not straightforward, if feasible at all.

This masters thesis covers two hierarchical structures and their variations, the first and most classical structure: an hourly hierachy. Which is present in the majority of studies covering temporal hierarchies. The second structure, dedicated to a longer forecasting horizon is a daily hierarchy. A similar daily hierarchical tree structure, was used to reconcile day-ahead forecasts of PV production, from more than three hundred sites in California, as explained in [Yang et al., 2017]. In this paper the authors cover only the traditional optimal reconciliation reconciliation methods. The authors also consider the reconciliation step as a viable post-processing technique.

One of the most recent developments in the reconciliation of hierarchical forecasts was done by [Nystrup et al., 2020]. Where the authors introduce four different estimators for the covariance matrix that take into account the autocorrelation structure when reconciling forecasts in a temporal hierarchy. An important addition to traditional methods, that will play an important role in the content of this thesis. The authors also argue about the importance of time series modeling approaches that rely only historical data, and not on external data sources, at least when it regards short-term electricity load forecasting. In a later issue, [Nystrup et al., 2021] give a remarkably detailed and a comprehensive explanation of the forecasting and reconciliation process. In addition, the authors use the eigen-value decomposition of the in-sample error structure to reduce dimensionality and extract as most information as possible to use in the reconciliation step.

The use of Machine Learning (ML) in the reconciliation process was introduced by two closely related papers, first by [Abolghasemi et al., 2019]. Where the authors used different ML techniques such as Artificial Neural Networks, XGboost

and SVM to perform the reconciliation task. This study used a hierarchy made of sales time series, aggregated by retailer and distribution center, and focused on supply chain optimization. So it was not a temporal hierarchy. The second paper, [Spiliotis et al., 2020], explains more in detail the methodology of reconciling hierarchical forecasts based on machine learning. The proposed method allows for a non-linear combination of the base forecasts, thus being more general than the traditionally linear approaches. This method achieves in parallel an improved out-of-sample forecasting accuracy and forecast coherence. The authors test this method on two data sets: the Australian tourism data set, which involves a four-level hierarchy with the domestic visitor nights of Australia across 76 regions, grouped into 27 zones and finally aggregated into 7 states and territories. And the sales data set which is the same as in [Abolghasemi et al., 2019]. This shows that there has been active development of ML reconciliation methods in the past couple of years, but there is still no study available that uses any ML reconciliation method for temporal hierarchies.

1.2 Motivation

The improvement and validation of a variety of building electricity load forecasts methods is the main objective of this thesis. In practice, this research aims at providing a comprehensive overview and comparison of several forecasting and reconciliation methods. A performance assessment of different hierarchical structures is presented, to determine which one is best for each scenario. This thesis also explores which characteristics the different forecasting and reconciliation models have, for which granularity do they perform best, and which would be the practical applications of forecasts for those time scales. This thesis also aims at identifying the main advantages and disadvantages of certain reconciliation methods. And at uncovering the possible relationship between forecasting performance and reconciliation accuracy gains (or losses), and between building metadata and model performance. To accomplish this research a Python module dedicated to temporal hierarchies was written and can be found in a public repository[6]. This thesis tests and validates the capabilities of this library.

[6] https://github.com/lorenzodonadio/TimeHierarchy

2 Methods

This section covers the steps of a temporal hierarchical forecast. This process starts with pre-processed data that is aggregated into a temporal hierarchy. Then a forecasting algorithm is applied to the hierarchical tree, which yields a set of independent forecasts. These independent forecasts or base forecasts are not coherent and thus need to be reconciled. The following three sub-sections explain each of these steps in detail, and at last, the validation procedure and the metrics used in this study are discussed.

2.1 Construction of a Time Hierarchy

The first step to use forecast a HTS is to first create the hierachy, i.e. aggregate data in a tree structure; which is the best representation of a hierarchy. We start this process with a time series; A data frame that has the variable of interest and the time index for each observation.

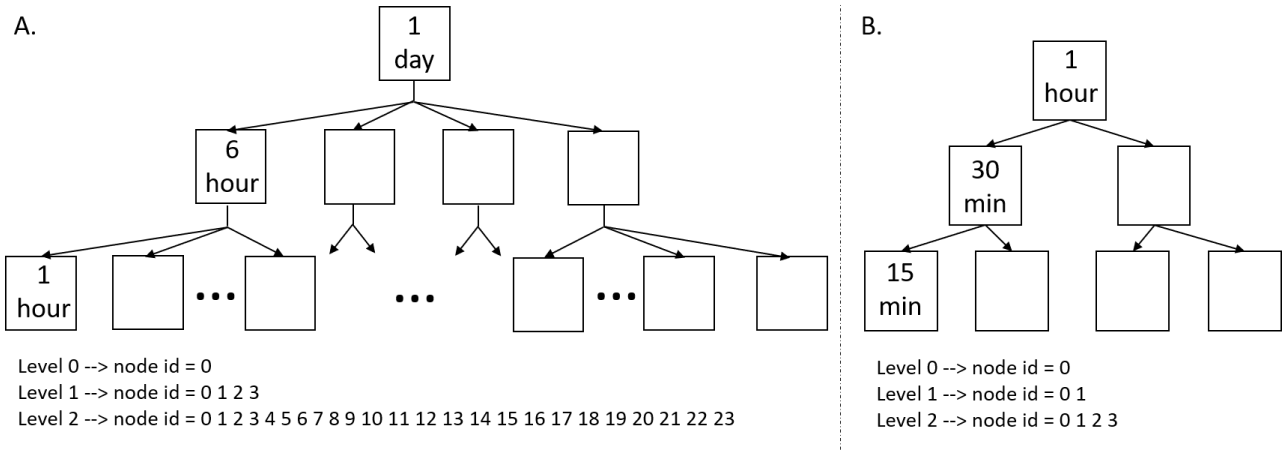

Figure 1: Two tree examples of hierarchical trees, the number of levels and nodes per level are presented. Tree A. illustrates a daily hierarchy, with a top aggregation level of 1 day and a bottom one of 1 hour. Tree B. represents an hourly hierarchy with a top and bottom levels of one hour and fifteen minutes, respectivly.

A temporal hierarchy is fully characterized by three elements: the granularity of the bottom (most disaggregated) time series, the number of aggregation levels, and the number of time series per aggregation level, which corresponds to the number of tree nodes in that particular level. The last two elements can be represented as a sum matrix, which entirely describes the hierarchical tree structure. Let us go through the notation with an example; the tree structure presented in Figure 1B. The first step is to build the tree structure from the bottom nodes up, by aggregating the data, k observations at a time, suppose the number of bottom nodes of a tree is noted by m, then the factor k must be a factor of m. So for the tree in question we have $m = 4$, so the possible aggregations are $k \in \{4, 2, 1\}$, and all of them where chosen, thus $k_1 = 4$, $k_2 = 2$ and $k_3 = 1$, this is the most

simplistic example of a hierarchy. In general a hierarchical tree has n nodes in total, m in the bottom level, K aggregation levels, and the aggregation factor $k \in \{k_1 ... k_K\}$ with $k_1 = m$ and $k_K = 1$. Note that the number of nodes in a certain aggregation level is given by m/k.

With that general formulation in mind, let us take a look at a slightly more complicated tree structure, the one depicted in Figure 1A, we start with hourly data, so we have $m = 24$, and so the possible aggregation levels are $k \in \{24, 12, 8, 6, 4, 3, 2, 1\}$, there are many more possibilities than in the previous example. But note that the case $k_1 = 24$ and $k_K = 1$ must be present since they represent the top and bottom levels of the tree, respectively. There remains the choice of which intermediate levels to include in the structure, and some combinations of aggregation factors are not mathematically possible. In Figure 1A we represent a tree where k_1=24, k_2=6, k_3=1 , note that when hourly data is aggregated by a factor of 6 then only 4 nodes are obtained at the intermediate level. Let us give an example of an erroneous combination of aggregation levels; if one attempts to aggregate according to: k_1=24, k_2=8,k_3=4,k_4=1, this clearly does not work even if 8 and 4 are factors of 24. First an aggregation by a factor of 4 would take place, yielding 6 nodes, and there is no possibility to aggregate 6 nodes by a groups of 8. Therefore not only the aggregation factors need to be a factor of m but the product of the $k_2 ... k_K$ also must be a factor of m and must be less than m.

$$S = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (1) \qquad S = \begin{bmatrix} I_{m/k_1} \otimes 1k_1 \\ \vdots \\ I_{m/k_K} \otimes 1k_k \end{bmatrix} \qquad (2)$$

Equation 1 shows the sum matrix associated with the tree structure illustrated in Figure 1B. The general formulation for the sum matrix is given in Equation 2, where $\otimes$ represents the Kronecker product, I_{m/k_1} is the identity matrix of order m/k_1 and $1k_1$ represents a vector of ones of size k_1.

In practice, it is important to programmatically create and store the hierarchical tree, for this task a tree is represented as a structure of two nested dictionaries (key-value pairs). The keys of the first dictionary represent the levels of the tree and inside each level there is another dictionary. Which keys represent the nodes of that particular level, the values of the second dictionary contain the actual time series that compose the hierarchical tree, each value (or tree node) contains a unique time series.

2.2 Forecasting Algorithms

The procedure to forecast a temporal hierarchy, consists on the generation of n independent forecasts; one for each node, it differs from how a traditional hierarchy (spatial, categorical etc.) is forecasted in that the time series of a temporal hierarchy have different granularities, depending on which aggregation level they belong to. Figure 2 shows a schematic of what a one step ahead forecast entails; the creation of a new forecasted tree, each level of the hierarchy must forecast a number of observations equal to the number of nodes in that hierarchy, for this example the bottom level forecasts four observations, the middle level two and the top level only one. This process can be repeated several times if the forecasting horizon is greater than one step (one hour for this example).

To perform a temporal hierarchical forecast there are two options. Forecasting models are fitted to either each node, or each aggregation level of the tree. The developed Python package allows for both choices, if one chooses to fit one model per node then: each time series has the same time-step, equal to the one of the top level, each time series has the same length (same amount of data) and the forecasting horizon is the same for each time series. On the other hand if one chooses to fit one model per tree level : The number of time series in the tree would be K instead of n. Each time series has a different length. The forecasting horizon is different for each time series, and proportional to the number of nodes of each level. K different forecasting models. This alternative was implemented but the results are not presented in this paper, because in terms of accuracy there was no significant difference and the first alternative is more similar to traditional hierarchical forecasting. Therefore, in total n different forecasting models will be trained, one model for each node.

Each model needs some hyperparameters, and it is a possibility to give different hyperparameters to the models of each of the levels of the hierarchy. Because there is no reason why the optimal hyperparameters for one level of the hierarchy are the same as for all the other levels. Therefore if deemed appropriate, the user can specify this option train models with different hyperparameters per aggregation level, this option gives the hierarchical forecasting more flexibility but also makes the model selection and parameter tuning somewhat more complex, since it adds a degree of freedom.

A forecasted tree, such as the one in yellow in Figure 2 has only one observation per node. Therefore this hierarchical tree can be represented as a vector, by stacking the observations from top to bottom. The resulting n-sized vector is called the base forecast vector and is denoted by $\hat{y} \in \mathbb{R}^n$. This notation will come useful when forecast reconciliation is discussed. Note that this $\hat{y}$ is a column vector that is valid for one step ahead forecast. If the horizon is greater than one step the notation could be extended. A collection of $\hat{y}$ vectors can be converted - by horizontal stacking of the vectors - into $\hat{y_h}$ which will then be a $n \times h$ matrix, where h is the forecasting horizon.

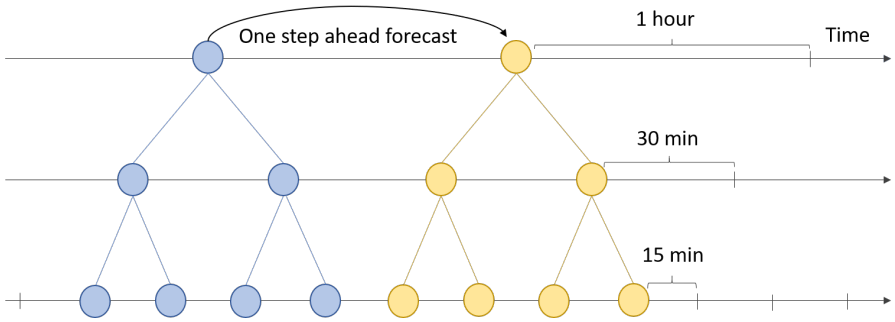

Figure 2: Illustration of one step ahead forecast for an hourly temporal hierarchy, the blue tree represents the most recent available observations and the yellow tree represents the forecasted tree.

Four forecasting algorithms were chosen for this thesis: ARIMA, ETS, TBATS and LGBM. Exponential Triple Smoothing (ETS) and ARIMA models are the two most widely used and traditional approaches to time series forecasting. These two models were implemented thanks to the statsmodels Python library [Seabold and Perktold, 2010].

2.2.1 ETS

In general, exponential smoothing methods do just that; smooth time series data by applying an exponential window function, which assigns exponentially decreasing weights over time. Exponential smoothing models are based on a description of the trend and seasonality in the data. In this study we use the most advanced version of exponential smoothing for forecasting which explicitly adds modeling for seasonality to the univariate time series: the seasonal Holt–Winters exponential triple smoothing. The most comprehensive review of the state of the art of exponential smoothing is provided by [Gardner, 2006], where the author gives an overview of the history of this method, reviews the formulation, and clearly explains different methods: additive or multiplicative trend and seasonality. The following are some examples of the use of ETS in forecasting utilities and traffic flows, water and gas and PV production, respectively [Gould et al., 2008],[Fildes et al., 1997],[Dev et al., 2018].

There are many parameters involved in ETS. Most of them do not need to be known a priori since they will be optimized automatically when fitting the model. However, the user must ensure that the seasonality is modeled correctly, the number of time steps in a seasonal period must be specified, a grid search and trial and error were used to select the appropriate seasonal periods. Since the focus of this study is forecasting a hierarchical tree, there is the possibility to use different hyperparameters - or seasonalities - for the ETS models at each level of the tree.

For this thesis only use additive seasonality and additive trend were used. Because the additive methods are more adapted to modeling time series where when the seasonal variations remain roughly constant, which is the case for the

building demand, as shown in Section 3. Whereas the multiplicative method is better suited when the seasonal variations are changing proportional to the level of the series. When using additive or multiplicative seasonality the time series is seasonally adjusted by subtracting or dividing by the seasonal component, respectively. In other words, the additive method expresses the seasonal component in absolute terms, for example at night a given building consumes 10kWh more than during the day, and then that value would be subtracted to the series. The multiplicative method on the other hand expresses the seasonal component in relative terms, for example at night the building consumes 50% less than in the day.

2.2.2 ARIMA

There should be no need to introduce ARIMA models, because it is a popular and evolving method since the 1970's. ARIMA models are still widely used in statistics, econometrics, and in particular in time series analysis and forecasting. The way ARIMA attempts to model a time series is complementary to exponential smoothing. ARIMA models aim to describe the autocorrelations in the data to make accurate forecasts.

ARIMA stands for autoregressive integrated moving average, and a simple model has three parameters to estimate: p,d and q. Each parameter is responsibly for a specific behaviour in the data modeling:

- p: the order of the autoregressive part; number of lag observations in the model.
- d: the number of times that the raw observations are differenced; also known as the degree of differencing.
- q: the size of the moving average window; also known as the order of the moving average

A problem with ARIMA is that it does not support seasonal data. That is a time series with a repeating cycle. And our data exhibits clear seasonality; daily cycles for example, but there may be more patterns depending on the time-scale and granularity of the data. So for implementation purposes a Seasonal ARIMA or SARIMA was preferred over ARIMA. With the addition of seasonality comes also the addition of four parameters:

- P: Seasonal autoregressive order.
- D: Seasonal difference order.
- Q: Seasonal moving average order.
- m: The number of time steps for a single seasonal period.

The notation of a SARIMA model is SARIMA(p,d,q)(P,D,Q,m). It is important to mention that the m parameter influences the P, D, and Q parameters and is the only parameter that must be known a priori, because it represents the intrinsic seasonality of the data. For example, an m of 24 for hourly data suggests a daily seasonal cycle. To select the correct seasonality (m), one could look at partial autocorrelation plots, base the choice on intuition or do a grid search of potential seasonalities to select the best one.

The drawback of Seasonal ARIMA, is that it can potentially have a large number of parameters and combinations of terms. And when using the wrong set of parameters the performance of the models drops drastically. Therefore these parameters must be optimized and in order to do so, various models must be tested and fitted to the data. Then the best performing model can be selected using an appropriate criterion. [Hyndman and Khandakar, 2008] introduced a method for automating the optimization of the ARIMA parameters, this method was first implemented in R under the name *auto_arima*, and has since then been translated to Python where it is called *pmdarima*[7], which was originally pyramid-arima for the anagram 'py' + 'arima'.

2.2.3 TBATS

TBATS is useful for forecasting time series with complex seasonal characteristics, for example non-integer seasonality, or a large-period seasonality; this makes it possible to create a detailed long-term forecast. This framework is fairly recent, being introduced by [Livera et al., 2011]. Despite it being so recent there is already a Python library specialized in TBATS[8], which was used in this project.

TBATS is an acronym for key features of the model:

- T: Trigonometric seasonality
- B: Box-Cox transformation
- A: ARIMA errors
- T: Trend
- S: Seasonal components

The price to pay for such a complex model is that when fitting a model, a multitude of different models are tested and only the best one selected. So when fitting TBATS to a large time series it becomes remarkably slow. But if the improvements in accuracy are significant, it might be worth the extra training time.

[7] https://pypi.org/project/pmdarima/

[8] https://pypi.org/project/tbats/

2.2.4 LGBM

The term Gradient Boosting Models (GBM) denotes a group of ML algorithms based on decision trees, GBM models can be used to solve a wide range of problems including regression, classification, and in this case forecasting. As the name implies, GBM use boosting as an ensembling approach among different decision trees. There are some effective implementations of GBM methods available such as XGBoost or CatBoost, but for this thesis LightGBM[9] was the chosen implementation. Since it is by far the fastest and most praised package, an explanation of the algorithm and what makes it faster is given by Ke et al. [2017].

LGBM is a very popular machine learning framework: it has been widely-used in many winning solutions of machine learning competitions, for example the M5 competition. LGBM was designed to be distributed and efficient. It has been proven to have the following advantages: faster training speed and higher efficiency, lower memory usage, better accuracy, support of parallel, distributed, and GPU learning and it is capable of handling large-scale data.

Differently than the previous three algorithms used for forecasting, LGBM is not precisely a forecasting model. It can primarily be used as a classifier, regressor or ranker. But isn't forecasting in the end just a regression problem?. The way to apply regression algorithms to a forecasting task is the following: first, the data must be transforms into the required tabular format, where the predictor variables would be some amount of lags of the target variable as well as some exogenous features; which are an optional addition to the model. Second, with data in a tabular format it is straightforward to fit a regressor and finally generate forecasts. This data manipulation must be repeated when new data is availible, to create out of sample forecasts; for example in a deployment environment. This entire process was automated in a wrapper to the LGBM regressor to make it suitable for forecasting.

There is a wide set of hyperparameters to tune in this case, the ones that are intrinsic to the LGBM model and the ones that come with the adaptation as a forecasting model. The intrinsic hyperparameters, which comprise the number of leaves, the learning rate, the evaluation metric and the number of boosted trees to fit, these parameters were set to their default values (see the API for more details). The main reason is because adjusting those parameters did not yield visible accuracy gains. Only the boosting method was subject of inspection, there are different boosting methods implemented in LGBM: *gbdt*, traditional Gradient Boosting Decision Tree. *dart*, Dropouts meet Multiple Additive Regression Trees and *goss*, Gradient-based One-Side Sampling.

The parameters that appear when adapting LGBM to a forecasting type are mostly in related to the explanatory variables; the number of lags (of the target variable) to include, and the selection of exogenous variables that may improve the model. If the target variable at time t: y_t, is to be predicted, then the en-

[9] https://lightgbm.readthedocs.io/en/latest/Python-API.html

dogenous predictor variables are: $y_{t-1}, y_{t-2}...y_{t-nlags}$, and the exogenous variables are: $x_t^1, x_t^1...x_t^{nfeatures}$, where $nlags$ represents the number of lags and $nfeatures$ the number of exogenous features. The exogenous features used in this study are divided into two groups: weather variables such as temperature, global radiation and relative humidity, and time related variables like hour of the day, day of the year, day of the month and day of the week. The time related variables are generated automatically from the timestamp of the time series, and are mapped by a cos or sin functions, because these transformations have been empirically proven to provide better results.

2.2.5 Hyperparameter Tuning

Each of the previously mentioned forecasting models has a set of hyperparameters that must be optimized in order to increase performance and select the best model for comparison. Considering that for a time hierarchy there is one forecasting model per tree node. The hyperparameter tuning can become truly tedious if a different set of hyperparameters is selected for each individual model, therefore this option was not implemented in this research. The most straightforward choice is to select one set of hyperparameters for all the models in the tree, this is simple but not very flexible and may not be the optimal solution in therms of accuracy. A compromise between complexity and flexibility is to allow for a different set of hyperparameters per aggregation level, arguing that the time series from different nodes belonging to one aggregation share more properties among themselves than with time series from other aggregation levels. The two previous options were implemented, and thus the user is able to give one set of hyperparameters for all the models in the tree or if deemed necessary one set of hyperparameters per aggregation level.

The actual parameter tuning used in this research is a simple grid-search on a predefined set of parameters. The starting parameters are not too far off from providing a well performing model, because they were selected after inspecting the data and the autocorrelation plots. In such manner, for each of the sets of parameters, an entire validation process was run, and then the best parameter set was selected based on the metrics from that validation process. This entails fitting the models thousands of times and thus is a very time consuming process.

The hyperparameter tuning of ARIMA is by far the most complex process of all forecasting algorithms, because only for this particular method the models have different parameters depending on which aggregation level they belong to. Thus for an hourly tree with three levels there will be three different optimal sets or ARIMA models. As explanined in Section 2.2.2, *pmdarima* was used to find the best set of parameters, and only the seasonality had to be specified.

For the other forecasting algorithms it was chosen to fit all the models in the tree with the same set of hyperparameters; the simpler option was chosen since providing a different set of parameters per aggregation level did not yield

differences in terms of performance. In addition there is not a package equivalent to *pmdarima* for the rest of the algorithms, which assists in an easier automation of the parameter tuning.

For ETS only one parameter had to be estimated: the seasonal periods; because only additive trend and additive seasonality where used. Regarding LGBM, the tuning process consisted mainly in feature selection; which weather and time features to include, how many observation lags (look back parameter) and which boosting method to use, since the parameters such as the number of leaves and the learning rate were kept as the default values.

2.3 Reconciliation Process

Let us assume the forecasting step is finished, so now independent forecasts were generated for all the aggregation levels. These forecasts will not add up to each other; they are not coherent. Thus they need to be combined with different methods to result in temporally reconciled, accurate and robust forecasts and yield a coherent hierarchical tree. This process supports aligned decisions at different horizons and granularities.

The most basic and straightforward way of reconciling forecasts is disregarding the upper levels and building a new tree by adding the bottom level, this is known as **Bottom Up** reconciliation. Using this method will yield un-biased and coherent forecasts but it will will entirely disregard the information on the upper levels of the tree, thus the accuracy will not be significant. Another basic reconciliation forecast is the **Top Down** approach, where only the top level of the hierarchy is forecasted and then it is disaggregated these down the hierarchy based many different measures, for example on the historical proportions of the data [Gross and Sohl, 1990]. The problem with Top Down methods is that they are inherently biased reconciliation methods and they disregard all the lower level forecasts.

It is clear by now that the optimal reconciliation process involves information transfer along all levels of the tree. In this thesis we explore three main methods of reconciliation, the so called generalized least squares (GLS), methods that account for auto-correlation and machine learning methods.

In an effort to generalize the notation of reconciliation methods, recall the base forecast notation $\hat{y}$, which represents the independent forecasts in a vector notation. Starting with the base forecasts the aim is to construct a similar vector of reconciled forecasts $\tilde{y}$, which satisfies the coherency constraint. This two vectors are related as follows:

$$\tilde{y} = SG\hat{y} \tag{3}$$

Where S is the n×m sum matrix and G is a m×n matrix that maps the base forecasts into the bottom-levels, this matrix will define the information exchanges

between nodes, and it is defined according to the desired reconciliation approach. The G matrix for the bottom up and top method would be the following:

$$BottomUp : G = [0_{m\times(n-m)}|I_m] \qquad TopDown : G = [p_{m\times 1}|0_{m\times(n-1)}]$$

In the above equations I_m is the identity matrix and $p_{m\times 1}$ is a vector of historical proportions or any other Top Down dissagregation factors, it is worth to note that the columns of the G matrix have are non-zero elements, are the ones that will be used to generate the reconciled forecasts, it is thus clear that there is an important information loss in the above described methods.

$$SGS = S \tag{4}$$

Equation 4 presents a constraint to the G matrix, that ensures that the reconciled forecasts are unbiased. The Bottom Up method satisfies this constraint but no Top Down method does.

2.3.1 Optimal Reconciliation Method

Given a tree of independent forecasts, an optimal reconciliation method should provide forecasts that are better, than any Top Down or Bottom Up method. The reconciliation process should also be free of bias, i.e. satisfy Equation 4. [Hyndman et al., 2011] Introduced an an optimal reconciliation method based on an estimate of the covariance matrix of the coherency errors. Let us assume the forecasting process as a linear regression as follows:

$$\hat{y}_h = S\beta(h) + \epsilon_h \tag{5}$$

Where $\beta(h) = \mathrm{E}[y_{K,t+h}|y_1...y_t]$ is the unknown mean of the bottom level K of the hierarchy at horizon h (t denotes the time index). And ϵ_h are the coherency errors (error between the base forecast and the reconciled forecast) for horizon h, the coherency errors are assumed to have zero mean and covariance matrix Σ_h. With this formulation we can use a Generalized Least Squares estimators (GLS) [Aitken, 1934] for $\beta(h)$, we obtain:

$$\hat{\beta}(h) = (S^{\mathsf{T}}\Sigma_h^{-1}S)^{-1}S^{\mathsf{T}}\Sigma_h^{-1}\hat{y}_h \tag{6}$$

This result allows us, using Equation 5, to calculate the reconciled forecasts. This method is called **GLS reconciliation**:

$$\tilde{y}_h = SG\hat{y}_h = S(S^{\mathsf{T}}\Sigma_h^{-1}S)^{-1}S^{\mathsf{T}}\Sigma_h^{-1}\hat{y}_h \tag{7}$$

From the above expression we can identify $\mathrm{G} = (S^{\mathsf{T}}\Sigma_h^{-1}S)^{-1}S^{\mathsf{T}}\Sigma_h^{-1}$, this matrix clearly satisfies the constraint in Equation 4; so if the base forecasts are unbiased so will the reconciled forecasts. The main problem with this method is the estimation of the covariance matrix of the coherency errors Σ_h, since it was

proven by [Wickramasuriya et al., 2019], that it can not be identified. [Wickramasuriya et al., 2019] also provides a different methodology to the GLS, to find and optimal reconciliation, is is done via trace minimization of the covariance matrix of the base forecast errors, this method is called Minimum Trace Estimator or **MinT**. In the aforementioned paper it is proven that covariance matrix of the h-step-ahead reconciled forecast errors (errors between the reconciled forecasts and the true values) can be written as follows: $\text{var}[y_{t+h}\tilde{y}_{t+h}|y_1...y_t] = SGW_hG^\top S^\top$, where W_h is the is the covariance matrix of the h-step-ahead base forecast errors, i.e the errors between the base forecasts and the true values. The MinT method minimizes finds the reconciled forecasts to be the following:

$$\tilde{y}_h = S(S^\top W_h^{-1} S)^{-1} S^\top W_h^{-1} \hat{y}_h \tag{8}$$

Note that it coincides with Equation 7, but with a different covariance matrix. This is no coincidence, since in a later publication [Wickramasuriya, 2021], the two formulations: GLS and MinT are formally proven to be equivalent. We will adopt the name and notation from MinT for simplicity.

After this brief theory background the three more traditional and simpler methods for the estimation of the covariance matrix are presented as follows:

- Ordinary least squares (**OLS**) estimator: this estimator is by far the simplest, the covariance matrix is set to be the identity matrix $W_h = I_n$. This estimator neglects all statistical properties of the time series as well as the structure itself of the hierarchy, and assigns the same weights to each node of the tree for the reconciliation process.
- Structural estimator (**STR**): this method is an improvement on the OLS estimator, it still neglects statistical properties of the data but it includes information about the structure of the hierarchical tree. It can be computed as follows: $W_h = S1_n$ where S is the sum matrix for the given hierarchy and 1_n is a vector of ones of size n. An example of the resulting matrix for an hourly hierarchy, such as the one presented in Figure 1 B is the following:

$$W_h = \begin{bmatrix} 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- Variance estimator (**VAR**): this method only estimated the variance, i.e only the diagonal elements of the covariance matrix are non zero, and thus $W_h = \Lambda$, where Λ is the diagonal variance matrix of the forecasting errors, and can be written as follows:

$$\Lambda = \begin{bmatrix} \sigma^2_{e_1} & 0 & \dots & 0 \\ 0 & \sigma^2_{e_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma^2_{e_n} \end{bmatrix}$$

Where the subscript e_1 means the forecasting error at node number one. The variance estimator, implicitly takes into account the structure of the hierarchy; it has different weights per level of aggregation. The order of magnitude of the error variance is different for each aggregation level, the higher the granularity the bigger the cumulative demand for that period of time and thus the errors tend to be higher.

2.3.2 Accounting for auto-correlation and cross-correlation

Based on the same Mint framework, this methods incorporates autocorrelation and cross correlation in the weighting scheme and should provide better results. The three estimators previously proposed (OLS,STR,VAR), are all diagonal matrices, and therefore neglect any correlation between levels and nodes of the hierarchy. Several alternative estimators for the covariance matrix were proposed by [Nystrup et al., 2020]. By including non zero entries for some of the non diagonal elements of the covariance matrix, autocorrelation is accounted for. The variance between, two nodes of same level of the hierarchy, is in essence the autocorrelation, to illustrate this fact we refer again to the hourly tree depicted in Figure 1B. Take for example the bottom nodes, they represent the four quarters of an hour (15min granularity), and so they can be denoted by: q1,q2,q3,q4. It is clear that var(q1,q2) is closely linked to the autocorrelation function with lag one (of a sub sample of the time series), var(q1,q3) to lag two and so on. The full empirical covariance matrix is also included in this category, since it is a full matrix and so it also accounts the autocorralation effects (it has non zero off-diagonal elements). All the reconciliation methods described in this section are based on the availability of one-step-ahead in-sample forecasting errors. For some models such as ARIMA and ETS those errors are automatically computed, but others such as LGBM needed an additional functionality for the in-sample error calculation.

- Full covariance estimator (COV): In this method we compute the full covariance matrix of the base forecast errors, two ways of calculating the matrix were implemented. The first uses the definition; the i,j element of a covariance matrix is given by[10]: $C(i,j) = \frac{1}{N-1}\sum_{n=1}^{n=N}(x_{i,n} - \bar{x_i})(x_{j,n} - \bar{x_j})$.The second, leverages the fact that errors are assumed to have a zero mean (which is a fair assumption), so we can simply write the covariance matrix of the base errors as: $W_h = \frac{1}{N-1}\epsilon_h\epsilon_h^{\mathsf{T}}$, where ϵ_h is a n $\times$ N matrix of base errors (n is the total number of nodes in the hierarchy and N is the number

[10] https://numpy.org/doc/stable/reference/generated/numpy.cov.html

of samples). The second formulation is simpler and faster to compute so it is the default method. The representation of the full covariance matrix in element notation is the following:

$$W_h = \frac{1}{N-1} \begin{bmatrix} e_1 e_1^\top & e_1 e_2^\top & \cdots & e_1 e_n^\top \\ e_2 e_1^\top & e_2 e_2^\top & \cdots & e_2 e_n^\top \\ \vdots & \vdots & \ddots & \vdots \\ e_n e_1^\top & e_n e_2^\top & \cdots & e_n e_n^\top \end{bmatrix}$$

- Block covariance estimator (blockCOV). This estimator imposes a zero inter level covariance, this means that only the covariances between nodes of the same granularity are accounted for, in practice it is equavalent to enforce the covariance matrix to have a block-like structure, for example the W_h matrix for an hourly tree would look as follows:)

$$W_h = \begin{bmatrix} e_1 e_1^\top & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & e_2 e_2^\top & e_2 e_3^\top & 0 & 0 & 0 & 0 \\ 0 & e_3 e_2^\top & e_3 e_3^\top & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & e_4 e_4^\top & e_4 e_5^\top & e_4 e_6^\top & e_4 e_7^\top \\ 0 & 0 & 0 & e_5 e_4^\top & e_5 e_5^\top & e_5 e_6^\top & e_5 e_7^\top \\ 0 & 0 & 0 & e_6 e_4^\top & e_6 e_5^\top & e_6 e_6^\top & e_6 e_7^\top \\ 0 & 0 & 0 & e_7 e_4^\top & e_7 e_5^\top & e_7 e_6^\top & e_7 e_7^\top \end{bmatrix}$$

- Glasso estimator [11], with this method the the precision matrix is estimated directly. The precision is the inverse of the variance: ($\rho = 1/\sigma^2$), which corresponds to W_h^{-1}. There is then no need to compute W_h, since in 8 there is no use for the matrix W_h, only of its inverse. The Glasso estimator is l1 penalized and the estimated matrix will be sparse, in accordance with the regularization parameter. Only the default value of the regularization parameter was used.
- Block Glasso estimator, this method is simply a way to impose to the Glasso estimator a block matrix structure. Therefore only inner level correlations are included. The imposed structure is analogous to the blockCOV structure, only that for the blockCOV we imposed a block structure to the W_h matrix and then computed its inverse, and for the block Glasso the block structure is directly enforced to the estimated precision matrix, i.e. W_h^{-1}.
- Cross-Correlation Shrinkage (CCS) estimator. This shrinkage estimator of the cross-correlation matrix was considered as a better was of dealing with heteroscedasticity problems that may arise with the cross-covariance matrix. The covariance matrix as follows: $W_h^{shrink} = \Lambda^{1/2} R_{shrink} \Lambda^{1/2}$. Where I_n is the identity matrix if size n and $R_{shrink} = (1-\lambda) * R + \lambda * I_n$. In the

[11] https://scikit-learn.org/stable/modules/generated/sklearn.covariance.GraphicalLasso.html

previous expression, R represents the Pearson product-moment correlation coefficients, or correlation matrix [12] where the element of $R_{i,j} = \frac{C_{ij}}{\sqrt{C_{ii} * C_{jj}}}$ (C is the covariance matrix). And where Λ is the diagonal matrix of variance. λ is a regularization parameter between 0 and 1, if λ is equal to one, then this method is equivalent to the variance scaling (VAR), and if it is equal to zero then it corresponds to a full covariance scaling (COV). For this research λ was set to 0.5 for simplicity.

- Markov estimator. This method integrates autocorrelation and variance information, and it only requires the computation of the first-order autocorrelation coefficient at each aggregation. level. It inherently has a block structure, and so no relationships between the aggregation levels are present. The estimated covariance matrix has the following form: $W_h^{markov} = \Lambda^{1/2} R_{markov} \Lambda^{1/2}$ Where again Λ is the diagonal variance matrix and R_{markov} contains the autocorrelation, an example for an hourly hierarchy is presented as follows:

$$R_{markov} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & \rho_{hh} & 0 & 0 & 0 & 0 \\ 0 & \rho_{hh} & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \rho_q & \rho_q^2 & \rho_q^3 \\ 0 & 0 & 0 & \rho_q & 1 & \rho_q & \rho_q^2 \\ 0 & 0 & 0 & \rho_q^2 & \rho_q & 1 & \rho_q \\ 0 & 0 & 0 & \rho_q^3 & \rho_q^2 & \rho_q & 1 \end{bmatrix}$$

  Where ρ_{hh} represents the first order autocorrelation of the half-hour (30min) level, and ρ_q the autocorrelation of the quarter (15min) level. Notice that the top level autocorrelation is not present, and that for the bottom levels the autocorrelation have an exponent which is proportional to their distance to the diagonal. Since the autocorrelation is always less than one, the exponents ensure that the values far from the diagonal will tend towards zero. This makes this method suitable for large hierarchies.

In Figure 3 the similarities between VAR and STR methods are clear but also the fact that the variance estimator brings more information to the reconciliation process. Note that for this method the in-sample forecasting errors are needed.

2.3.3 Machine Learning Reconciliation

Machine learning reconciliation is a new method that bypasses the linear algebra of the previous methods by using ML models to directly provide the reconciled forecast, and thus aims to obtain better accuracy. ML methods combine the base forecasts in a non-linear way, thus being more general than the previously described linear approaches. This reconciliation is fairly recent, only a few studies

[12] https://numpy.org/doc/stable/reference/generated/numpy.corrcoef.html

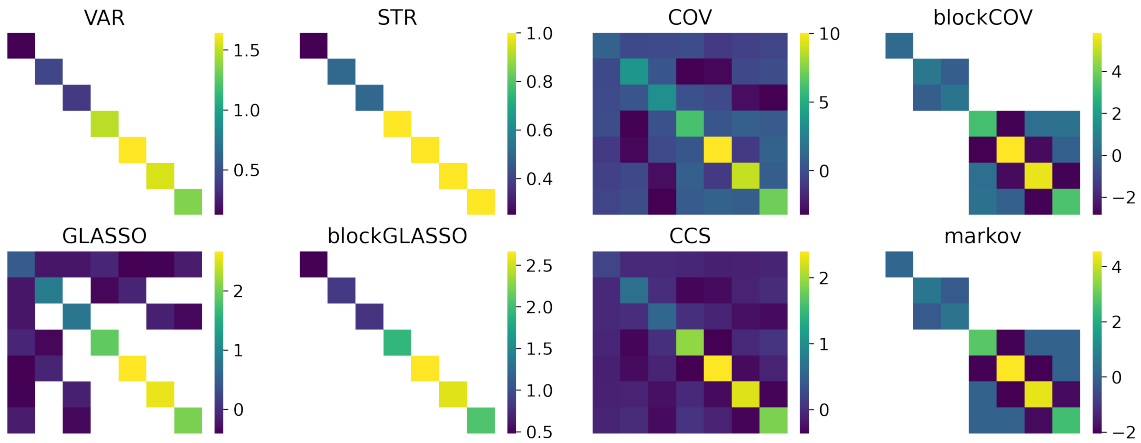

Figure 3: examples of the inverse covariance matrix W_h^{-1}, for an hourly tree. The first element, from left to right and top to bottom, corresponds to the top level node of the hierarchy (1 hour), the next two elements to the second level (30 minutes) and the last four elements to the third level (15 minutes).

have been published,[Abolghasemi et al., 2019] and [Spiliotis et al., 2020]. These two studies share similar methodologies, the first uses three ML methods: Xgboost, Artificial Neural Networks (ANN) and Support Vector Regression (SVR). The second Xgboost and Random Forests. For this thesis we implemented LGBM and SVM for reconciliaton, but only the results from LGBM reconciliation are presented since it greatly outperformed SVM.

An ML method is characterized by its explanatory variables (features or input) and by its target variable or variables. For the reconciliation task, ML attempts to predict the true values of the bottom level of the hierarchy based on the entire tree of base forecasts. In other words, the target variables are the true values of each of the bottom nodes, and the explanatory variables are the base forecasts for all the nodes in the hierarchy.

1. A forecasting model is fitted to each time series, and an in-sample forecast is produced.
2. Create the predictor matrix $X_{N\times n}$ and the target variable matrix $Y_{N\times m}$. Notice that it is a $\mathbb{R}^n \to \mathbb{R}^m$ transformation, and most ML models support only a $\mathbb{R}^n \to \mathbb{R}$ operation, with the exception of multi output methods such as ANN.
3. Fit a ML reconciliation model to each of the bottom nodes of the hierarchy, m in total.
4. Perform an out of sample forecast for the entire tree, using the models in step 1.
5. Using m models trained in step 3, predict the values of the bottom nodes of the forecasted hierarchy.
6. Use the Bottom Up method to recreate an entire tree of reconciled forecasts hierarchy; aggregate the nodes starting from the bottom level.

2.4 Validation and Metrics

The selected metrics for this research are Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and Median Absolute Error (MedAE). Their scaled versions are presented, so that they are comparable between buildings and tree levels. The inclusion of three different metrics aims to increase the robustness of the results, if all the metrics are in alignment. Or to point out interesting behaviours if they differ.

The RMSE is equivalent to the standard deviation of the forecasting errors. In other words RMSE measures how spread out these errors are. Because it involves squaring the errors, a large but few frequent error will have an important influence on the RMSE. In contrast each error contributes to MAE in proportion to the absolute value of the error, strictly speaking MAE is the average absolute difference between true and predicted data. The MedAE is a more robust measure of the error variability, since the median is more robust to outliers than the mean. But the MedAE is not the most adapted metric at quantifying the few frequent and large errors of the forecast.

In this research the intention is to quantify the accuracy of the two stages of predicting temporal hierarchies, the forecasting step and the reconciliation step. To assess the accuracy in forecasting step, the normalized version of the aforementioned metric were thus chosen, namely: the Mean Absolute Error (nMAE), the normalized Root Mean Squared Error (nRMSE) and the normalized Median Absolute Error (nMedAE). Let y represent the true values, $\bar{y}$ the average of the true values, $\hat{y}$ the predicted values and N the sample size. With that notation the metrics can be written as follows:

$$\text{nMAE} = \frac{100}{\bar{y}} \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i| \tag{9}$$

$$\text{nRMSE} = \frac{100}{\bar{y}} \sqrt{\frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2} \tag{10}$$

$$\text{nMedAE} = \frac{100}{\bar{y}} \text{Median}(\mid y_1 - \hat{y}_1 \mid, \ldots, \mid y_N - \hat{y}_N \mid) \tag{11}$$

After the forecast, comes the reconciliation step, once it is finished, the metrics are calculated for the reconciled forecasts. In order to assess if an improvement on the base forecast was made or if on the other had the reconciled forecasts present worse accuracy, the reconciliation metrics are transformed into the Percentage Relative Improvement in Average Loss (PRIAL), as recommended by [Hyndman and Koehler, 2006] and [Nystrup et al., 2021].

By definition, the PRIAL of base forecasts is zero. When comparing the accuracy of reconciled and base forecasts. A positive entry shows a percentage

decrease in the regarded metric relative to the base forecast, i.e. improved accuracy, and a negative entry a decrease in accuracy. Thus the PRIAL provides with a very straightforward way of quantifying the effect of reconciliation on the based forecasts.

$$PRIAL_{MAE} = 100 \times (1 - \frac{MAE}{MAE_{Base}}) \quad PRIAL_{RMSE} = 100 \times (1 - \frac{RMSE}{RMSE_{Base}}) \tag{12}$$

For the purpose of testing the performance of the different forecasting algorithms, i.e. calculating the metrics on the data in a fair way that reassembles real life forecasting, the validation scheme shown in Figure 4 was devised. The original time series was split in train set and test set, both sets together account only to a fraction for the entire data set, and the test set comes immediately after the train set. In addition, the size of the test set corresponds to the forecasting horizon, usually very small when compared to the train size. Once the data is split, the models are trained and a forecast for the test sample is generated and stored, the train and test sets are shifted, by the size of the test set, so that there are no overlapping points in the new test set. This process is repeated for the remaining data or until a maximum number or iterations is reached. This validation procedure was also used in several articles but [Yang et al., 2015] provides the reader with an adequate explanation.

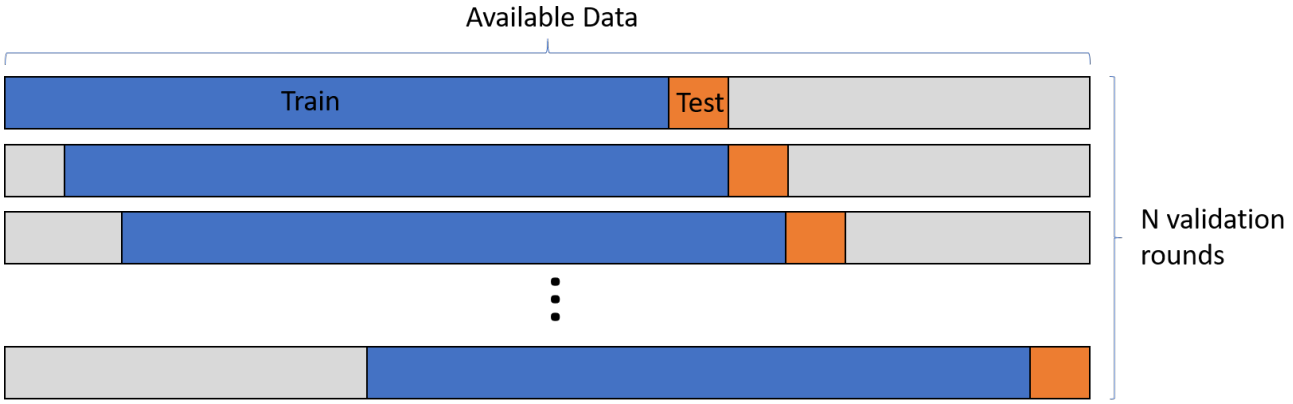

Figure 4: Illustration of the validation process. For each iteration the predicted variable and its true values are stored, at the end of the process they are concatenated, and only then are the metrics calculated.

In practice a test size of only five observations was used for this research, this means that the forecasts in question are at most five steps ahead, so the forecasting horizon would be five hours and five days for an hourly and daily tree, respectively. The train size for the hourly tree was 12.5 % of the entire dataset, and for the daily tree it was 50%. The reason for the small train size is that the models do not need more data to be performant, and it prevents the models from learning from trends that occurred far in the past, but are not present in recent

observations. In addition, the bigger the train set, the slower the training process and thus the slower the validation procedure, which is already time consuming, specially when it is used for hyperparameter tuning. Due to time constraints a final addition to the validation process was made: imposing one hundred as a limit on the validation rounds. After this adjustment and to make sure the results were fair, an increased shift was introduced after each validation round, so that the test set would run through the remaining portion of the data set.

3 Case Study

This study analyses building electrical demand for 16 BAM office buildings, located in 13 different Dutch cities. The electricity demand is the net demand, i.e. the local PV production has already been subtracted from the demand, so the net demand is the amount of energy that actually needs to be provided to the building by the grid. The time series are stored with a granularity of 15 minutes, for this research 27 months of data were used, from January 2019 to April 2021. The buildings in question were of different sizes, electricity consumption, energy label and year of construction. Also some building were equipped with local PV production and some with natural gas meters, as presented in Table 1. The buildings that have a gas meter, mainly use it for heating, but there is no knowledge about the heating of the buildings that do not have a gas meter installed, they might still use gas but as part of district heating or they might not use gas at all and rely on heat pumps for example. The fact that the internal building processes are a black box makes this a strong and realistic test case.

Building	Average Demand (kWh)	Gas	PV	Area (m2)	Year	Energy Label
r1	25.8	✓	✓	3178	-	-
n1	23.4	✓		10335	1970	C
p3	21.7			7339	1994	C
t1	20.7	✓		9073	2002	A
e1	19.2			4561	-	-
s4	18.3			5116	-	A
p31	15.1	✓	✓	3954	2004	A
d1	12.0			2200	2019	A
s1	8.0	✓		4709	1999	D
p8	7.1	✓		2933	-	-
l6	6.6	✓		2682	-	C
k2	6.4	✓		4880	-	C
k3	5.6	✓	✓	4118	-	G
j1	5.6	✓	✓	3159	2003	A
r3	2.7	✓	✓	2124	1990	A
t14	0.8			1941	2009	A

Table 1: Building metadata, the average electricity demand is per the average demand per 15 minute intervals. Gas and PV indicate if a building is equipped with a gas meter or PV production. The building names are abbreviated for privacy reasons. The Average Demand is per 15 minute intervals

The majority of buildings have an A as energy label, and the buildings with a C label are older, as expected. Nevertheless there is one building with D and another one with G energy labels, which according to Dutch law will not be allowed From January first, 2023 for office buildings [13].

[13] https://business.gov.nl/regulation/energy-labels/#article-energy-labels-for-buildings

The preprocessing of the data was not a very extensive process. There were some days with missing data, and they were removed accordingly. On the other hand there was no outlier detection in place, nor data scaling. Even if the developed Python package natively supports scaling it did not influence performance so it was decided against. The weekends were removed from the time series, because the demand on the weekends is mostly constant, so forecasting models are used for the week-days only. This is the way that the current forecasting is done in the BAM headquarter offices, and approach that works fairly well. In practice, different ML algorithms forecast week-data and weekend demand is modeled as the average profile of past weekends. Therefore the focus is shifted entirely to the weekdays. The weather data used for forecasting with LGBM, comes from KNMI, the Dutch weather service. For each building the weather data was interpolated from the closest weather station.

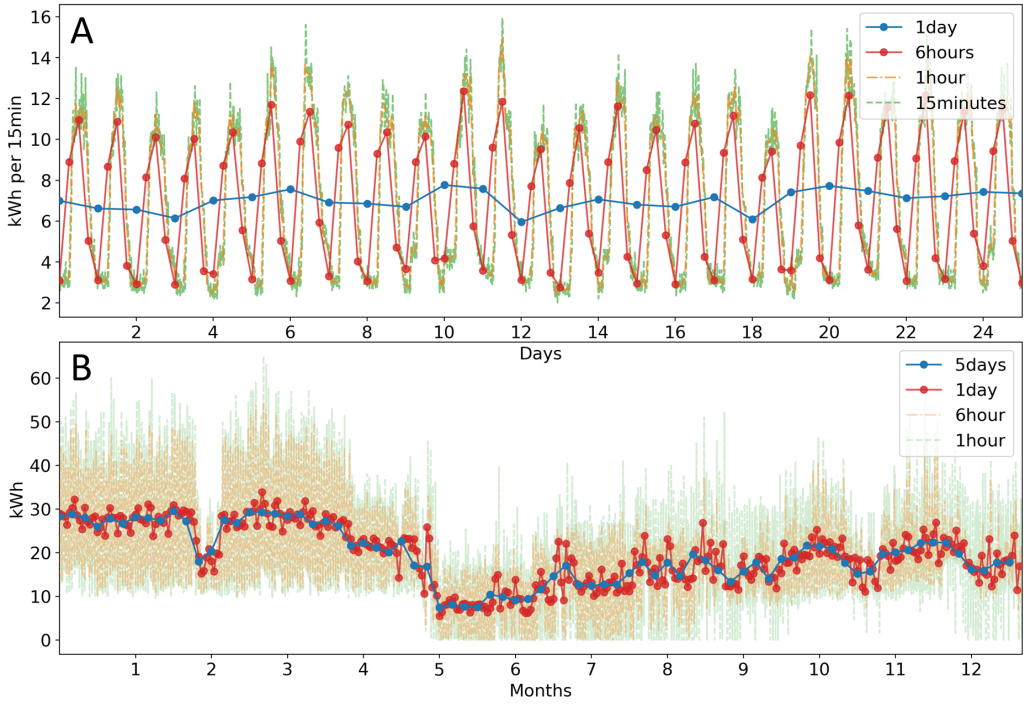

Figure 5: Illustration of the time series for one building, for different granularities and time spans. A shows a short term behaviour, and B shows long term behaviour. In B we can see Christmas effect and also the effect of Covid-19 lockdown. The series starts at the beginning November, so month number two corresponds to Christmas

Aggregating time series, has a smoothing effect, it is similar to filtering the high frequencies. Figure 5 A shows that 15 minute data is very volatile, there are big variations. When aggregated to hourly data this variation decreases but only slightly, it is only when aggregated to a full daily cycle that it becomes stable. Forecasting daily demand should thus be easier than hourly or 15 minute demand.

On panel B, the Christmas of 2019 effect is visible; a period of low demand because of closed buildings. But most importantly the immediate effect of Covid-19 lockdown in March 2020 is clearly noticeable. When remote work became the norm and thus electricity demand dropped drastically for office buildings, the

profile of the demand became more erratic because peoples habits were entirely different and so was energy consumption. This poses a big challenge to the forecasting models because they are confronted to forecasting data that looks nothing like there was before, so it is expected that the most resilient model will have the best performances. In Figure 5 B, one can observe that even at 1 day granularity there is still an important variability of the demand; some high frequency oscillations remain, even after the daily cycle has vanished due to aggregation. But when aggregated at 5 day scale only a smooth trend of the data remains, and if it were not for Covid-19 the constant behaviour from the first two months of the graph is expected to remain more or less unchanged throughout the year.

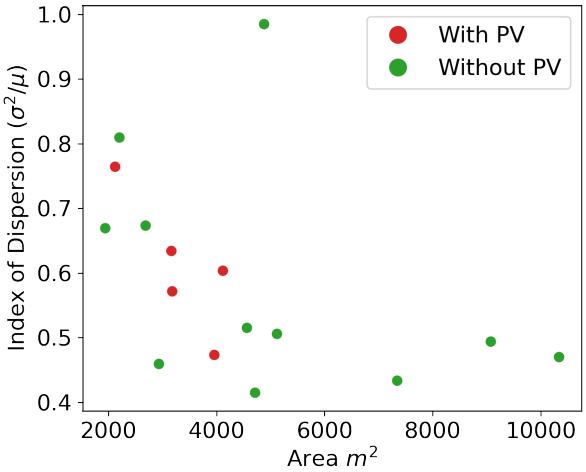

Figure 6: Dispersion index vs surface area of the buildings, this shows a tendency for larger buildings to have a smaller dispersion index.

The larger buildings are, the more random processes that take place inside the building; There are more people that come and go, more appliances, computers, lights and so on. Thus for large buildings the variations of these random processes tend to cancel out and produce a demand profile with less variability. This behaviour is related to the law of large numbers in probability theory, which states that the averages of some random events have a long-term stability. In addition, larger buildings tend to have more operations that are scheduled and which account for a big part of the electricity consumption, such as ventilation or cooling. In Figure 6 this relationship between building size, expressed with the buildings surface are in this case, and the index of dispersion is presented. The index of dispersion is a variance to mean ratio, i.e. how big is the variance with respect to the mean. There is no clustering of building with and without PV, therefore statistical properties of those buildings are not immensely different and it is thus fair to compare the performance of the forecasts.

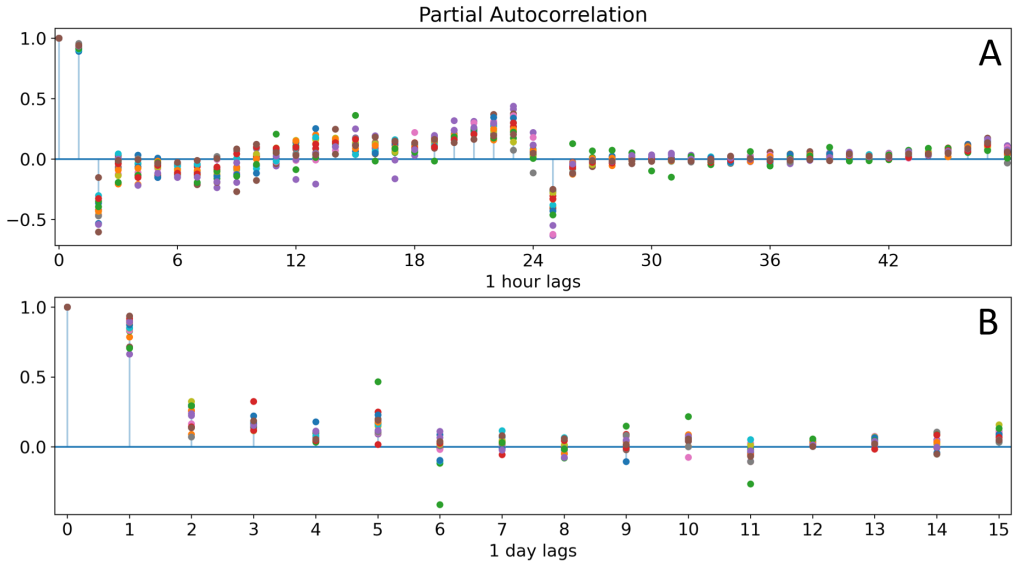

Figure 7: Partial autocorrelation functions for all 16 buildings. Panel A shows data aggregated at hourly granularity, and panel B at daily granularity.

The partial autocorrelation for all the buildings is shown in Figure 7. Panel A shows data aggregated at hourly granularity and panel B at daily granularity. These two granularities correspond to the top level of aggregation for a daily and hourly tree, as shown in Figure 1. Studying the autocorrelation at those particular levels can help identify the behaviour of the time series, and thus the parameters, specially the seasonality, of the forecasting models. Panel A clearly shows, as expected, a daily cycle and some weak intra-day autocorrelation. In contrast Panel B lacks the weekly cycle due to the removal of weekend data, but nevertheless there is a small peak in autocorrelation at day five; which indicates that, for example Mondays are more similar to Mondays than to other day of the week.

4 Results and Discussion

This section is split in two parts, the results for the forecasting step and the results for the reconciliation step. The forecast results allow for a selection of the best set of hyperparameters for each algorithm, and identify the most performant one for each hierarchy. The reconciliation results are built on the basis of the most accurate forecasting algorithm for each building. After that, a selection of reconciliation algorithms are applied only to the best forecast algorithm. With the aim of identifying the best reconciliation method. This chapter separately evaluates the two different application cases: hour ahead and day ahead forecasting.

4.1 Forecast

The hyperparameter tuning was performed with a subset of eight buildings; half of the total. This choice is based on simulation time constraints. For each forecasting algorithm, several sets of parameters where tested; this entails running the entire validation process many times, calculating the metrics for each set of hyperparameters and then selecting the best set.

Three forecasting algorithms were selected for this process: ARIMA, ETS and LGBM. In contrast TBATS was disregarded because of its prohibitively low training speed, the ARIMA, ETS and LGBM all train all the models in a tree structure in 5 to 10 seconds on average, but TBATS is two orders of magnitude slower, therefore the parameter tuning was impossible to perform with TBATS.

4.1.1 Hourly Tree Structure

This tree structure is focused on the short term forecasting; few hours ahead. The main pattern in hourly data is the daily cycle of electricity demand. This tree structure includes the smallest granularity and thus the time series are characterized by high frequency variability. All this factors will play a role on which forecasting algorithm is best suited for and hourly tree structure.

ARIMA

Based on the partial autocorrelation form Figure 7, a seasonality up to 24 hours seems reasonable, so three different seasonalities were tested: 6, 12 and 24 hours. The results for the best parameters are shown in Table 2.

	Order (p,d,q)	Seasonal (P,D,Q,m)
Level 1	(4, 1, 0)	(2, 0, 0, 12)
Level 2	(2, 1, 2)	(2, 0, 0, 12)
Level 3	(1, 1, 5)	(0, 0, 0, 12)

Table 2: Optimal ARIMA parameters per aggregation level for an hourly tree.

It is interesting to note that the selected seasonality does not amount to an entire day but instead to 12 hours, half a daily cycle, this may be linked to the day and night differences. However the difference in performance between a seasonality of 24 and 12 is small; the order of 1% of RMSE and 0.5% of MAE when contrasted to the 10-20% decrease of performance when a seasonality of 6 hours is chosen. Nevertheless The seasonal component plays a role in the first two levels of the hierarchy (non-zero PDQ) whereas for the last level all the seasonal parameters are zero, so it has no influence. It is worth noting that the optimal ARIMA parameters change indeed between aggregation levels, this is backed my the fact that aggregation changes the properties of a time series and thus different hyperparameters are needed for its modeling. Figure 8 presents the metrics from this optimal set of ARIMA models.

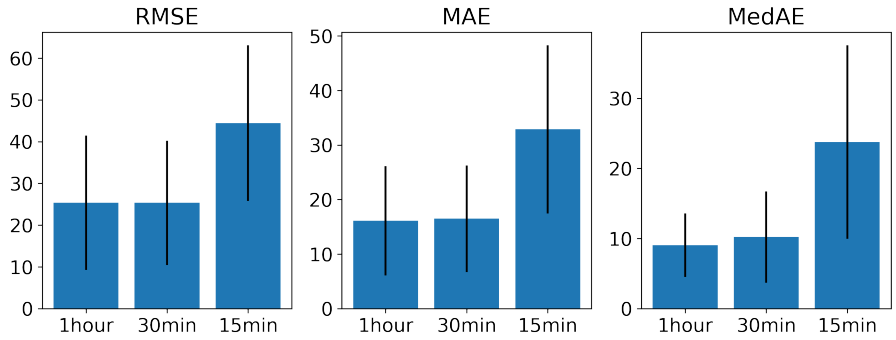

Figure 8: Average across buildings of the normalized RMSE, MAE and MedAE for the optimal ARIMA models fitted on an hourly hierarchy.

The three concerned metrics show a similar behaviour; they show an important drop in performance when regarding level 3 (15min granularity), the metrics almost double. The metrics for the first two levels are acceptable but not remarkably good: 25% RMSE and 16% MAE and 10% MedAE. The standard deviation is considerable: 16% for RMSE and 10% MAE and 4.5% MedAE, which indicates a high variability of performance across buildings which is not desirable. The fact that there is no seasonality included in the 15 minute granularity could explain why the performance is poor for that level, another possible explanation is the higher variability of the data at that particular level, this could also be a factor that affects performance.

ETS

Concerning ETS, the seasonal periods that were considered, ranges from 12 to 96, i.e. half a day up to four days. The optimal seasonal period is found to be 24, so an entire daily cycle. The metrics from this model are presented in Figure 9.

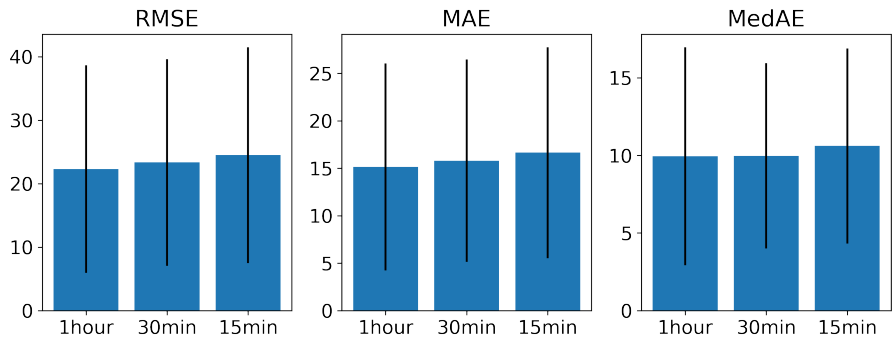

Figure 9: Average across buildings of the normalized RMSE, MAE and MedAE for the optimal ETS models fitted on an hourly hierarchy.

The three metrics are in agreement, showing a slight increase in performance as the granularity increases, the best forecast at the 1 hour level have 22% RMSE, 15% MAE and 10% MedAE, this is a slight improvement from ARIMA. But the greatest improvement comes at the 15 min aggregation level, this shows that the simplicity of ETS makes it robust enough to over-perform ARIMA. Nevertheless the standard deviation is the highest of all three algorithms: 16% RMSE, 11% MAE and 7% MedAE, this is a red flag when selecting the best forecasting method; ETS is not a robust enough method for this particular task.

LGBM

The best parameters for the LGBM model are the following. The most adequate boosting type is undeniably the traditional *gbdt* (gradient boosting decision tree). The look back parameter (the number of lags included as features) was found to be 72, out of a range from 12 to 96. Which means that the LGBM model will have as input 3 days of past electricity demand in order to make a forecast. Regarding the exogenous predictors, only weather data was selected: temperature, global radiation and relative humidity. There were no time features selected such as hour of the day or day of the year. The results obtained using the previously described LGBM forecasting model are presented in Figure 10.

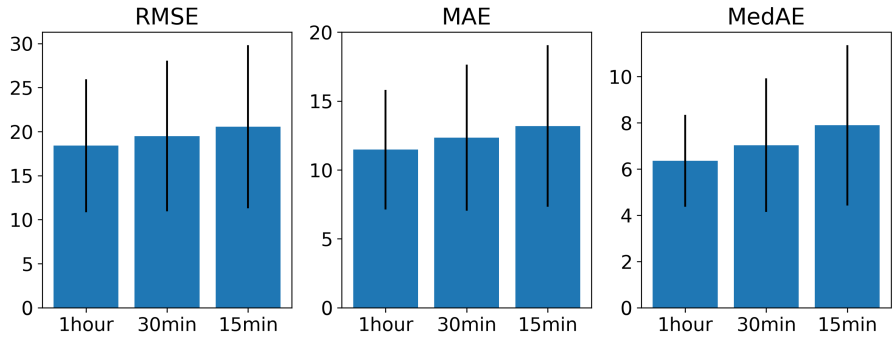

Figure 10: Average across buildings of the normalized RMSE, MAE and MedAE for the optimal LGBM models fitted on an hourly hierarchy.

The same agreement between metrics as in ARIMA and ETS is again present,

but this time there is a better overall performance and a sharper increase in accuracy when going up in the hierarchy levels. for the 1 hour level the metrics are: 18% RMSE, 11.5% MAE and 6% MedAE with the respective standard deviation being: 7.5% for RMSE, 4% for MAE and 2% for MedAE. These results show that LGBM is the most suited forecasting algorithm for this given hourly hierarchy, and that the top level of the hierarchy is better forecasted than the bottom level, there is a 3% difference of RMSE between the 1 hour and the 15 minute levels.

4.1.2 Daily Tree Structure

A daily tree structure entails that the intraday fluctuations have vanished due to the aggregation, thus any seasonality could come from weekly, monthly or even yearly cycles. To find the best seasonality time-scale for the models, the search began at small values: starting from no seasonality at all its value was then increased gradually to detect any possible improvement up to 28 day seasonality.

ARIMA

Based on the partial autocorrelation form Figure 7, a seasonality up to 24 hours seems reasonable, so three different seasonalities were tested: 6, 12 and 24 hours. The results for the best parameters are shown in Table 3.

	Order (p,d,q)	Seasonal (P,D,Q,m)
Level 1	(0, 1, 2)	(1, 0, 1, 3)
Level 2	(2, 0, 2)	(0, 1, 2, 3)
Level 3	(2, 0, 2)	(0, 0, 1, 3)

Table 3: Optimal ARIMA parameters per aggregation level for a daily tree.

The optimal seasonality was found to be of 3 days, and the non-zero P,D,Q parameters indicate that it does play an important role in the forecasting. Such seasonality is best to capture short term patterns, and shows the lack of monthly or weekly cycles. It is important to remember that the weekend data was removed. The metrics from this optimal model are presented in Figure 11.

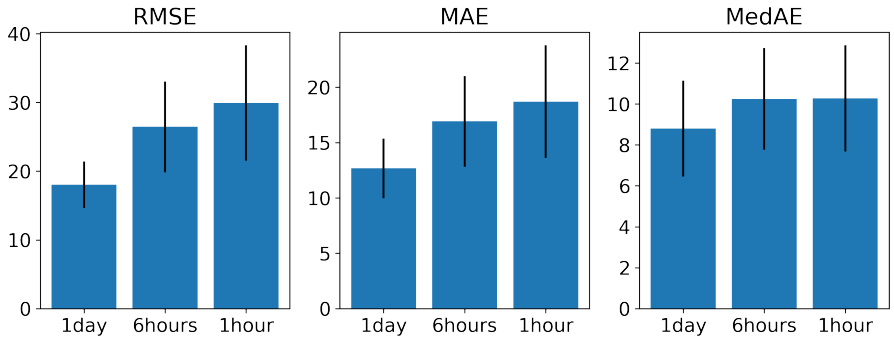

Figure 11: Average across buildings of the normalized RMSE, MAE and MedAE for the optimal ARIMA models fitted on a daily hierarchy.

The metrics coincide in the fact that the daily granularity is the best forecasted one, with a RMSE of 18%, MAE of 12.5% and MedAE of 8.5%, this constitutes the best performance of the three models. Nevertheless the accuracy of the 1 hour level still leaves room for improvement. In contrast with an hourly hierarchy, the standard deviation is drastically lower.

ETS

The optimal seasonal period was found to be of 3 days, same as for ARIMA. This further supports the claim that long-term cycles do not help the forecasting at a daily level. The results using the optimal ETS model are shown in Figure 12.

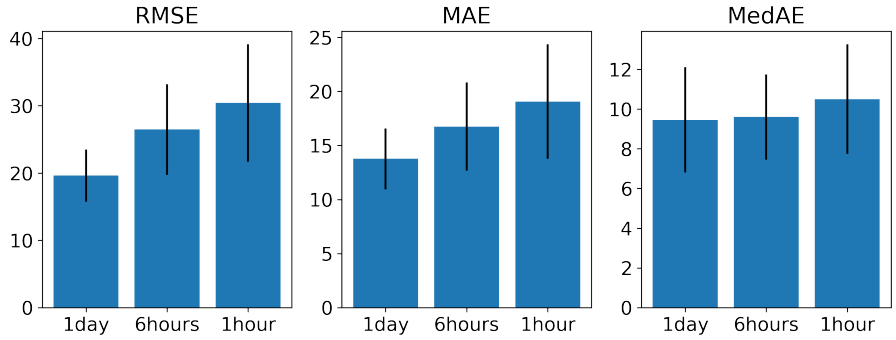

Figure 12: Average across buildings of the normalized RMSE,MAE and MedAE for the optimal ETS models fitted on a daily hierarchy.

The metrics corresponding the 1 day granularity are: 19.5% RMSE, 13.5% MAE and 9.5% MedAE. An acceptable accuracy for such a simple model, but around 1% worse than ARIMA.

LGBM

The selected parameters for LGBM are similar than those form the hourly hierarchy, again *gbdt* is the preferred boosting method and the weather features: temperature, global radiation and relative humidity were chosen, with no time features included. The only difference is the look back, or number of lags, the optimal value is 3, so to make a forecast only the past 3 days are included. The look back is in accordance to the seasonality from ARIMA and ETS, and shows that the short-term trends dominate when forecasting building load at a daily scale. The results from LGBM are presented in Figure 13.

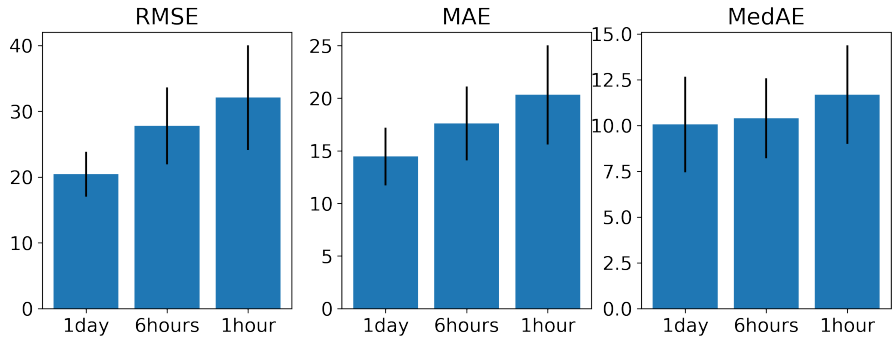

Figure 13: Average across buildings of the normalized RMSE,MAE and MedAE for the optimal LGBM models fitted on a daily hierarchy.

The metrics for the 1 day granularity are the following: 20.5% RMSE, 14.5% MAE and 10% MedAE. Again, it is an acceptable accuracy but around 1% worse than ETS and 2% worse than ARIMA.

4.1.3 Base Forecast Correlation

The performance of the base forecasts is related to some building characteristics. In Figure 14 the RMSE of the best forecasts (of all the levels of both daily and hourly hierarchies) is compared with some of the metadata from Table 1. The energy label, gas or PV meters and the year of construction of the building did not present any correlation and thus were excluded from the scatter plot. Only metadata about the building electricity demand and the area are presented.

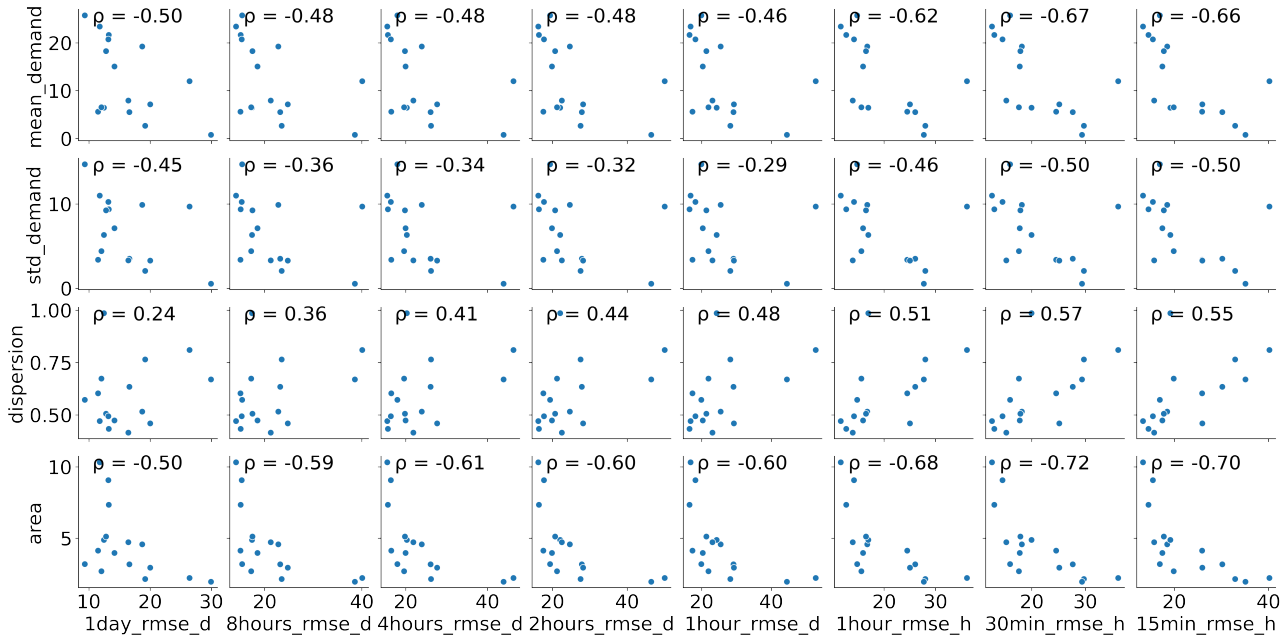

Figure 14: The RMSE of the best independent forecasts is plotted against building metadata,namely: mean electricity demand, standard deviation of the electricity demand (both in kWh per 15min), dispersion of the electricity demand and building area (units of 1000m^2). It shows a consistent correlation for both tree structures, a different hierarchical structure is indicated by the subscript _h for hourly or _d for daily tree structure. The Pearson correlation is presented for each scatter-plot

There is a clear negative correlation between the average demand and the RMSE, hence larger buildings have more accurate forecasts. This claim is also supported by the negative correlation between area and RMSE. It is interesting to note a negative correlation between the standard deviation of the electricity demand and the RMSE. This implies that the higher the variability the better the prediction, and this claim is misleading. In reality when a building increases in size, so does the electricity demand and also its the standard deviation, seen from this perspective the correlation of the standard deviation is due to the standard deviation becoming a proxy for building size more due to the variability of the time series. To detach the building size from the variability the dispersion should be the variable in question, recall that the dispersion is the standard deviation scaled by the mean. In this case there is a positive correlation between dispersion and RMSE, which in accordance to common knowledge, indicates that more noisy and variable a time series, the more difficult it is to forecast it.

4.2 Reconciliation

In this section the results from the reconciliation step are presented.Even if in the previous section about forecasting, the best algorithm (on average) per tree structure was identified, the reconciliation process is be based on all tree forecasting algorithms and then the best pair forecasting-reconciliation per building is selected.

The reason for this choice is simply that it is impossible to known a priori how much the reconciliation will improve the forecasts, and if it will give the same improvement for all the forecasting algorithms or if it will perform better on one of them, which may not necessarily be the best in terms of base forecast metrics. For example, let us assume that for one building, ARIMA was the best algorithm for forecasting, but LGBM was not far behind, and that after the reconciliation step, the overall accuracy is better for LGBM plus reconciliation than for ARIMA plus reconciliation.

It is worth mentioning that only one machine learning reconciliation method is presented: LGBM. Because the performance of SVM in reconciliation was extremely poor and was not deemed worthy of discussion.

4.2.1 Hourly Tree Structure

In this section, the following reconciliation methods are presented: VAR, STR, OLS, LGBM (machine learning reconciliation), COV, GLASSO, blockCOV, CCS, markov.

For each building, all the reconciliation methods were applied to the best base forecasts. The results are presented in Figure 15. It is clear that, overall, the reconciliation improves the base forecasts in terms of RMSE; the vast majority of the values in the heatmap are positive. Only LGBM reconciliation presents some

negative PRIAL values, that go as low as -1.7% for building *j1*, but remarkably enough LGBM also provides the best PRIAL for a different building.

Regarding the best reconciliation per building, highlighted in blue. The maximum improvement is of 7.3% of RMSE for building *k2* and comes from LGBM reconciliation. The minimum improvement is of 2.4% for building *t1* and comes from markov reconciliation.

The reconciliation method that performed the best for the most buildings is COV; 4 buildings. followed by VAR, STR and LGBM, each of them was selected best for 3 buildings. Then comes markov with 2 buildings and last blockCOV with 1 building. GLASSO and CCS were never selected as best reconciliation for any building. Nevertheless, on several occurrences, the differences in performance were limited, so that picking one reconciliation method over another for a given building would not change much in practice.

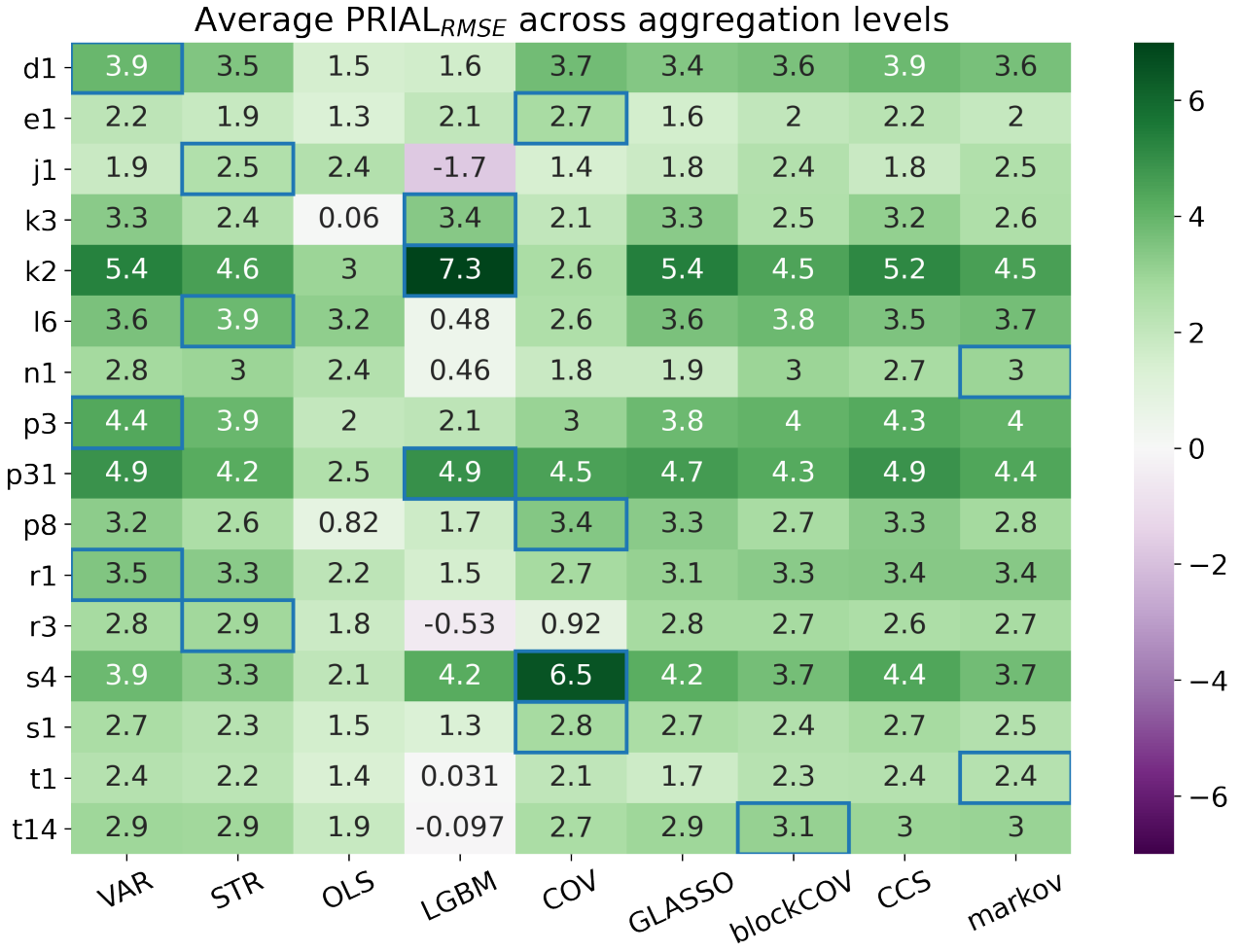

Figure 15: For each building, the average PRIAL RMSE across aggregation levels is presented. The highest value is highlighted in blue, which corresponds to the best reconciliation method.

It is also important to inspect how the reconciliation step acts on every level of the hierarchy, therefore the highlighted methods in Figure 15 will be studied in more detail. In Figure 16 the PRIAL for each aggregation level of the best reconciliation methods are shown. The most clear and unequivocal remark is that

the best forecasting method was LGBM for all the buildings, again in accordance with Section 4.1.1. There is a significant difference on how the reconciliation improved the forecasts for the different levels. On the 15 minute level there is a slight improvement for most of the buildings, an almost zero improvement for four of them and a worse forecast for only one; building *k3*. Regarding the two upper levels of the hierarchy, they show a relatively good overall improvement, of an average of 5.8% improvement for the 1 hour level and a 4.3% improvement for the 30 minute level. The noisy data from the 15 minute level could be the reason for the small reconciliation improvements for that level. Another possibility is that the reconciliation algorithms adjust less the values at the bottom of the tree, this possibility is supported by the weighting schemes in Section 2.3.

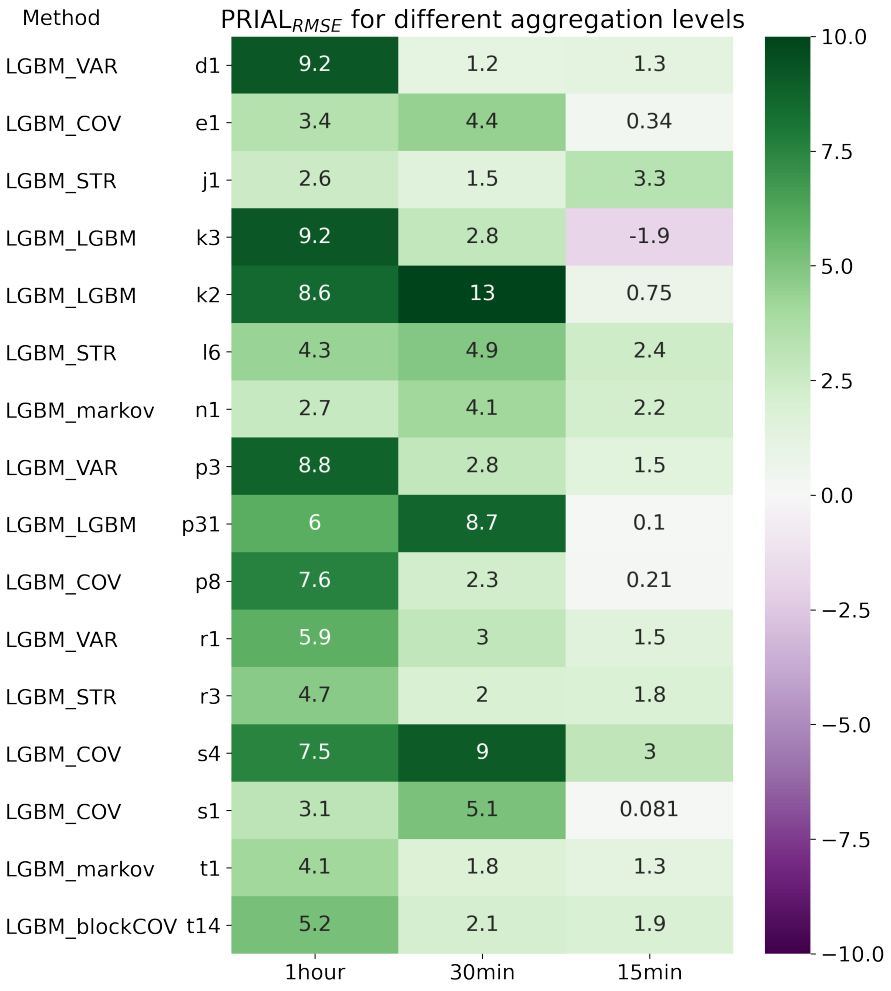

Figure 16: Best forecasting and reconciliation algorithm and the corresponding PRIAL RMSE for the different aggregation levels of each of the buildings in the case study.

4.2.2 Different Number of Aggregation Levels

Before diving into the results for a daily tree structure, let us have a word about the other possible tree structures for a daily hierarchy. The tree structure from Figure 1A is not the only possible tree that can be constructed with a top granularity of one day and a bottom granularity of one hour. Here the results of three

different structures are presented, each with a different number of aggregation levels: three, four, and five.

- Three aggregation levels. This structure is the one presented in Figure 1A. It has three different granularities: 1 day, 6 hours, 1 hour. The 6 hour level has a total of four nodes.
- Four aggregation levels. This structure has the following levels: 1 day, 6 hours, 3 hours, 1 hour, notice the only modification is the addition of the 3 hour aggregation level, which has a total of eight nodes.
- Five aggregation levels. This structure is quite different from the last two, since none of the intermediate levels share the same granularity. This five level tree has the following levels: 1 day, 8 hours, 4 hours, 2 hours, 1 hour. The 2 hours, 4 hours and 8 hours aggregation levels have twelve, six and three nodes, respectively.

It is worth noting that the only two granularities shared by all three structures are 1 day and 1 hour; which correspond to the top and bottom levels. The following tables compare the performances of these two levels in terms of normalized RMSE.

1 Day	3 levels		4 levels		5 levels	
building	$RMSE_{Min}$	Algorithm	$RMSE_{Min}$	Algorithm	$RMSE_{Min}$	Algorithm
d1	**22.00**	ETS_LGBM	25.93	LGBM_CCS	25.84	ARIMA_LGBM
e1	**17.88**	ARIMA_OLS	18.66	ETS_LGBM	18.64	LGBM_LGBM
j1	22.74	ARIMA_OLS	16.11	ETS_LGBM	**15.62**	ETS_LGBM
k3	16.09	ARIMA_markov	11.47	ARIMA_OLS	**11.45**	ARIMA_OLS
k2	17.57	ARIMA_VAR	**11.11**	LGBM_VAR	11.16	LGBM_CCS
l6	12.78	ARIMA_STR	12.05	ETS_LGBM	**11.80**	ETS_LGBM
n1	12.77	ARIMA_STR	**10.68**	ARIMA_CCS	10.71	ARIMA_VAR
p3	18.27	ARIMA_STR	**12.79**	ARIMA	**12.79**	ARIMA
p31	15.87	ETS_COV	**12.33**	LGBM_CCS	12.56	LGBM_VAR
p8	**17.49**	ARIMA_VAR	20.02	ARIMA	20.02	ARIMA
r1	12.73	ARIMA_LGBM	**9.33**	ARIMA	**9.33**	ARIMA
r3	22.32	ARIMA_OLS	19.19	LGBM	**19.11**	LGBM_markov
s4	15.35	ARIMA_markov	**11.66**	ARIMA_OLS	11.67	ARIMA_OLS
s1	**15.45**	ARIMA_OLS	16.45	ETS	16.45	ETS
t1	**10.79**	ARIMA_VAR	11.60	ARIMA_VAR	11.67	ARIMA_CCS
t14	**23.04**	ARIMA_OLS	29.33	ARIMA_VAR	29.52	ARIMA_VAR
Average	17.07	-	15.54	-	**15.52**	-

Table 4: Normalized RMSE for the 1 Day aggregation level - top level. This table compares different tree structures with three, four and five aggregation levels. For each building, the structure with the lower nRMSE is highlighted. It is worth noting that the *Min* subscript indicates that the RMSE corresponds to best algorithm for that particular building, forecast and reconciliation combined. The forecasting algorithm comes first and after the underscore is the corresponding reconciliation method

It is visible in Table 4 that the addition of one or two extra levels decreases on average 1.5% the RMSE. This is a small but significant improvement, but suggests that a 5 level tree is a more suitable structure. The results can also be analyzed from the perspective of performance of individual buildings, for some buildings the 3 level structure performs best, and for others the 4 or 5 level structure. Thus there is no such thing as a rule that more aggregation levels increase the forecasting plus reconciliation performance, but that modifying a tree structure is a tool to better model time series data that should be studied in an individual manner. The results from Table 5, for the 1 hour level, are similar but show a more defined preference towards a structure with 4 or 5 levels, this time the average difference in RMSE is of 3% and only three, out of the sixteen buildings, show their best performance with three aggregation levels.

Tables 4 and 5 also show the most performant foretasting and reconciliation methods per building and per tree structure. It is interesting to note that, in agreement with the results from Section 4.1.2, ARIMA is by a very big margin the most used algorithm for the 3 level tree, even if for a few buildings ETS is found to be best. But once the hierarchical structure starts to change, then ARIMA loses some relevance, for example with 5 aggregation levels, ARIMA is only chosen for seven buildings (in terms of RMSE), in contrast to the twelve buildings from the 3 level hierarchy.

It is also worth noting that for 3 levels of aggregation, the inclusion of a reconciliation algorithm always improved the forecasts; there is always a reconciliation method after the forecasting method (separated by an underscore). Whereas this is not always the case with a larger hierarchy. In Table 5 there are five buildings where only the forecasting algorithm is present (no underscore followed by the name of a reconciliation method), this means that for that particular level the forecasting alone provided a better RMSE. This hints to a weakness in the reconciliation methods; scalability to large hierarchies.

1 Hour	3 levels		4 levels		5 levels	
building	$RMSE_{Min}$	Algorithm	$RMSE_{Min}$	Algorithm	$RMSE_{Min}$	Algorithm
d1	**45.10**	ETS_LGBM	52.46	LGBM_CCS	52.67	LGBM_CCS
e1	25.81	ARIMA_CCS	**24.31**	ETS_LGBM	24.56	ETS_LGBM
j1	37.01	ARIMA_LGBM	28.49	ETS_LGBM	**27.89**	ETS_LGBM
k3	28.62	ARIMA_VAR	**17.12**	ARIMA_VAR	17.26	ARIMA_VAR
k2	30.41	ARIMA_VAR	24.16	LGBM_VAR	**24.10**	LGBM_VAR
l6	23.55	ETS_CCS	21.76	LGBM_OLS	**21.49**	ETS_LGBM
n1	20.04	ARIMA_LGBM	**16.21**	ETS	**16.21**	ETS
p3	24.75	ARIMA_STR	**16.42**	ARIMA	**16.42**	ARIMA
p31	25.91	ARIMA_VAR	20.09	ETS_LGBM	**19.96**	ETS_LGBM
p8	**27.53**	ARIMA_VAR	28.57	ARIMA_LGBM	28.44	ARIMA_LGBM
r1	22.50	ETS_LGBM	19.83	ARIMA_VAR	**19.75**	ARIMA_CCS
r3	38.73	ARIMA_VAR	**28.22**	LGBM	**28.22**	LGBM
s4	24.08	ARIMA_LGBM	21.23	ARIMA_LGBM	**21.01**	ARIMA_CCS
s1	22.95	ETS_CCS	**22.52**	ETS_LGBM	23.05	ETS_markov
t1	18.36	ARIMA_CCS	**17.85**	ARIMA	**17.85**	ARIMA
t14	**34.60**	ARIMA_CCS	44.43	ARIMA_LGBM	44.47	ARIMA
Average	28.12	-	25.23	-	**25.21**	-

Table 5: Normalized RMSE for the 1 hour aggregation level - bottom level. This table compares different tree structures with three, four and five aggregation levels. Please note that the *Min* subscript indicates that the RMSE corresponds to best algorithm for that particular building, forecast and reconciliation combined. The forecasting algorithm comes first and after the underscore is the corresponding reconciliation method

4.2.3 Daily Tree Structure

The results of a five level tree structure are presented, since it was previously shown that in terms of RMSE and MAE the best results overall are obtained with a five level tree. In this section we only present the following reconciliation methods: VAR, STR, OLS, LGBM (machine learning reconciliation), CCS, markov. Some methods were not suited for this tree structure: COV, GLASSO, blockCOV. Because of computational problems regarding the calculation of the inverse variance matrix.

For each building, all the reconciliation methods were applied to the best base forecasts. The results are presented in Figure 17.For this different tree structure, the improvement of reconciliation on the base forecasts. It is clear that, overall, the reconciliation improves the base forecasts in terms of RMSE; the vast majority of the values in the heatmap are positive. LGBM reconciliation presents abundant negative PRIAL values, that go as low as -7% for building *k2*, but on the other hand LGBM also provides the best PRIAL for two different buildings; for a daily tree structure LGBM reconciliation can be performant but it is unreliable. OLS reconciliation does not bring much to the table, with PRIAL values ranging from -1% to 1.9% even if for building *s1* it is regarded as the best reconciliation method.

Regarding the best reconciliation per building, highlighted in blue. The maximum improvement is of 5.3% of RMSE for building *p31* and comes from VAR

reconciliation. The minimum improvement is of 0.81% for building *s1* and comes from OLS reconciliation.

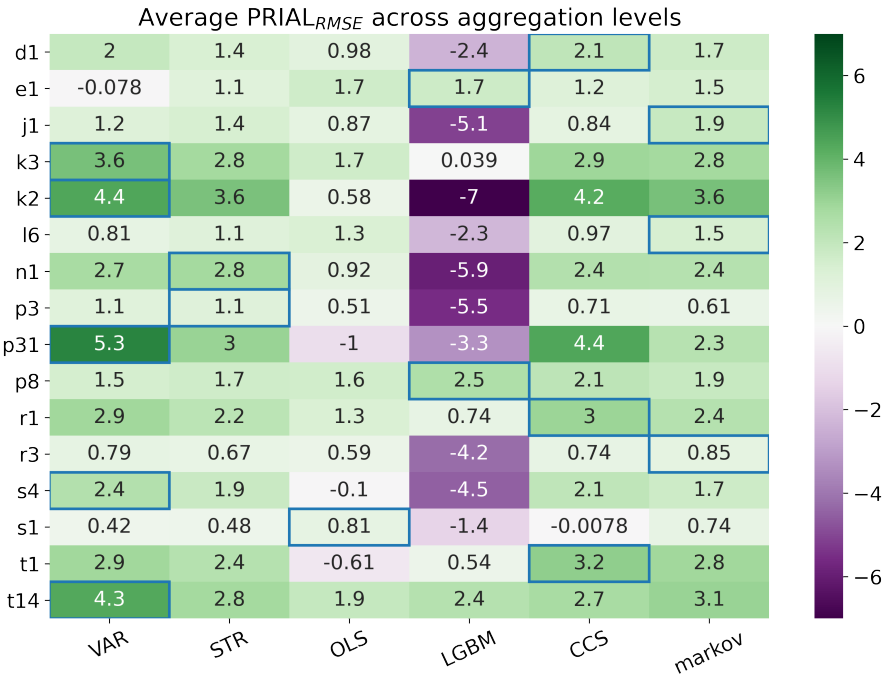

Figure 17: Average PRIAL RMSE across levels .

The reconciliation method that was the most performant for the highest amount of buildings is VAR; 5 buildings. followed by CCS and markov with 3 each. Then there is LGBM and STR each with 2 occurrences and last OLS which was only best for only 1 building.

For a more complex tree structure, the classic reconciliation methods, provide the better and more robust the results. VAR and STR, have merits to be important reconciliation methods despite their simplicity. Although the methods accounting for autocorrelation (CCS and markov) trail the classical methods but are not far behind, and deserve their recognition.

The intra-level performance of the best forecasting and reconciliation pairs of algorithms, highlighted in blue in Figure 17, is presented in Figure 18. The first important remark is that all three forecasting algorithms are present, different than the hourly hierarchy where LGBM was the most performant for every building. Nevertheless LGBM remains the forecast selected for the majority of buildings, with a total of nine, followed by ARIMA with four buildings and then ETS with three. Regarding reconciliation, the disparity in PRIAL in quite remarkable, both when comparing the results for different buildings or the different levels of the same building. The intermediate levels seem to have the most consistency in the results. In particular the 8 hours level, where the reconciliation always improved the forecasts by more than 1%, and has an average PRIAL of 4%. Similar to the behaviour of reconciliation on an hourly hierarchy, the bottom level is where the least improvements are made. In this case, the 1 hour level improved the forecast of only three buildings above 1%, and for two other buildings

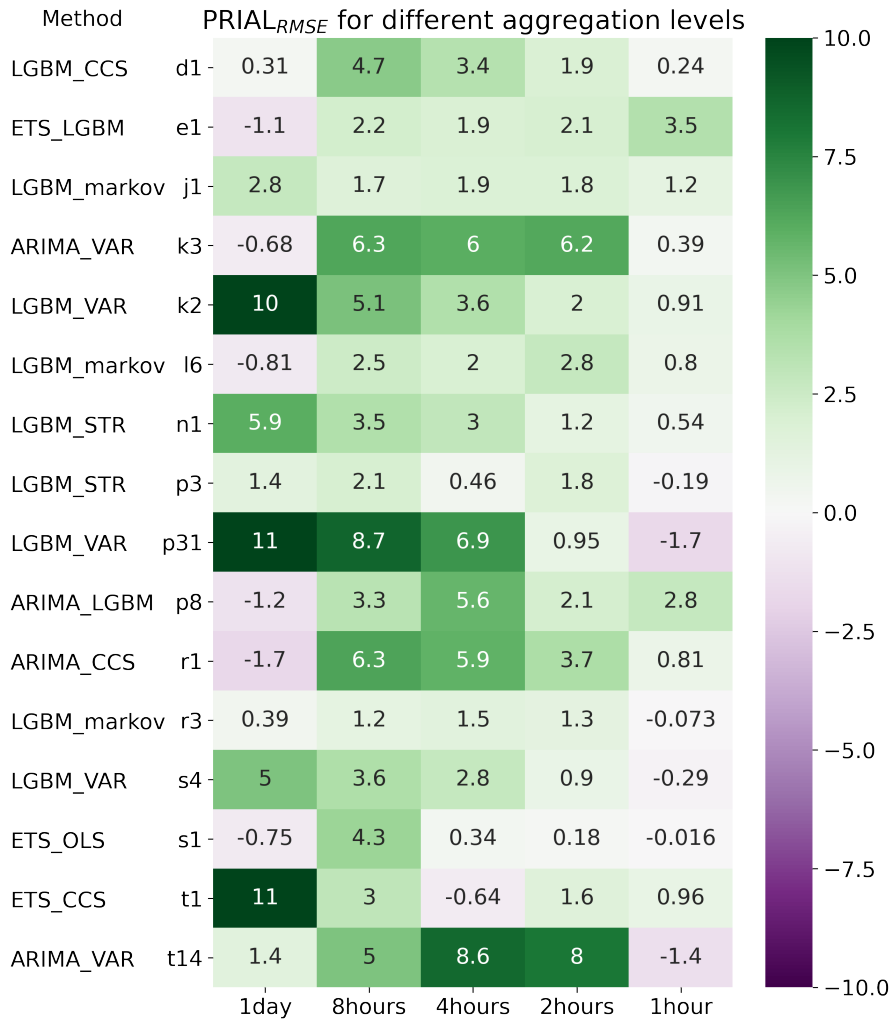

Figure 18: Best forecasting and reconciliation algorithm and the corresponding PRIAL RMSE for the different aggregation levels of each of the buildings in the case study.

the reconciliation worsened the forecasts by the same margin. In addition, the rest of the buildings have a PRIAL close to zero, which amounts to an average PRIAL of 0.5% for this 1 hour level, on average the forecasts are still improved but not in a substantial way.

The top level of the hierarchy; the 1 day granularity, is where the most surprising results happen, different than for an hourly tree where the top level showed consistent improvements, for a daily tree it is either an important improvement or a decrease in accuracy. There are three buildings with very substantial improvements from reconciliation; above 10%. Two other buildings also have a PRIAL of around 5%, so roughly a third of the buildings show significant gains in accuracy from reconciliation. On the other hand, six building exhibit negative PRIAL values, as low as -1.7%, this amounts to another third of buildings that lose in performance from reconciliation at the 1 day level. The rest of the buildings have a PRIAL near zero or one percent, so on average for the 1 day aggregation level the PRIAL is 2.7%, which is a high value but comes from striking differences between buildings.

4.2.4 Reconciled Forecast Correlations

Here the performance of the influence of building data and forecasting performance on the reconciliation step will be discussed. In Figure A1 the reconciliation PRIAL_{RMSE} is plotted against the same building data as in Figure 14, but no actual relationship is found. Which means that the performance of the reconciliation step is not influenced by the size of the building nor the dispersion of the original time series.

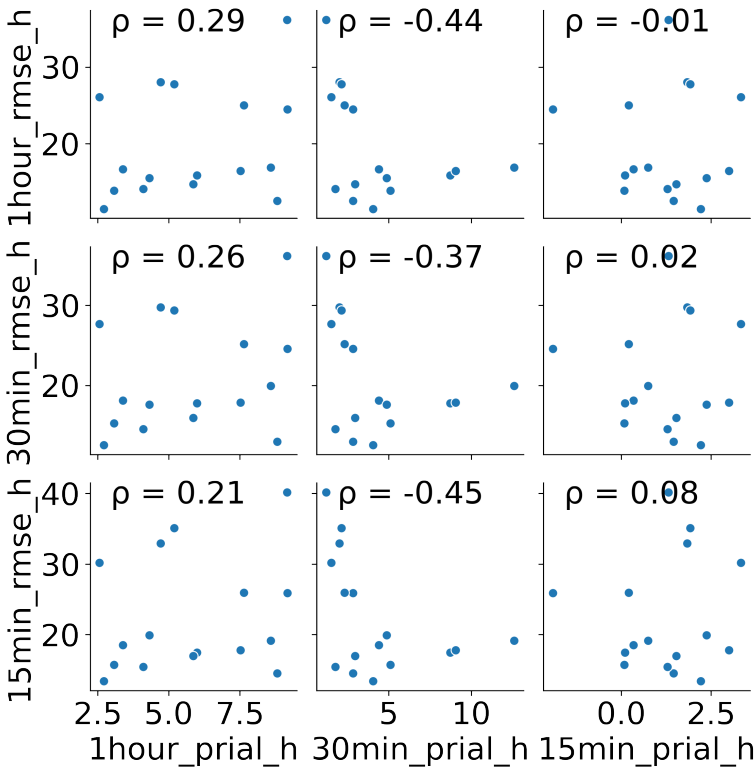

Figure 19: Reconciliation PRIAL plotted against forecasting RMSE for an hourly tree structure. The Pearson correlation is presented for each scatter-plot and it shows no link between forecasting and reconciliation performance.

If reconciliation performance is not related to building metadata, then it may be related for the forecast accuracy. To get some insight on this possible dependency, the best forecast RMSE is plotted against the corresponding reconciliation PRIAL_{RMSE}. In Figure 19 an example of an hourly hierarchy is shown. Remarkably the correlations are weak and there seem to be no consistent patterns. Therefore the reconciliation is not influenced by the forecast accuracy, based only on the base forecast there is no way to predict if the reconciliation step will perform well or not. This claim is also supported by Figure A2, where the same scatter plot of RMSE against PRIAL is shown, but this time for a daily tree structure.

5 Conclusion

Throughout this research different models were tested for building load forecast using a temporal hierarchical scheme. After a validation step the best models were selected based on different metrics. A variety of reconciliation methods were studied, to ensure forecasts coherency and increase accuracy. Two different hierarchical structures were examined: daily and hourly hierarchy, with a dataset of sixteen office buildings and more than two years of fifteen minute electricity demand data available.

The quality of electricity demand forecasting, is crucial to the operation and trading activities the energy market, the deployment of energy flexibility and ultimately the reduction of CO_2 emissions. One of the potential benefits of the studied forecasting methods is rapid deployment for a large portfolio where there is limited knowledge of the buildings.

For certain applications it might be more convenient to forecast building demand at a different granularities, for example regarding day ahead electricity purchase, the use of the one hour granularity is best suited. Because generally the bottom level of a hierarchy exhibits a lesser amount of improvements from reconciliation than the top and intermediate levels.

Among the four implemented forecasting models, the adaptation of LGBM to a forecasting task, proved superior to other methods overall, for both a daily and an hourly time hierarchy. The performance of ARIMA was a match for LGBM on a daily tree structure but it was overshadowed in an hourly hierarchy. On the other hand ETS, performed as expected and remained a robust and simple baseline forecasting algorithm. The prohibitively slow TBATS was not considered in the comparison, because it was impossible to run it as many times as the others, and on a reduced data set it did not prove to be strikingly better.

The structure itself of the temporal hierarchy was proved to influence the performance of both forecasting and reconciliation steps, as well as the choice of which forecasting algorithm is best suited for a particular building. Larger hierarchies could be further investigated, for example a one week hierarchical structure seems like the logical next step in this direction; to test the scalability of the algorithms.

This research considered only sixteen buildings, a further study could benefit from a larger number of buildings and more variety; including residential and commercial buildings as well. Nevertheless the results, indicate that an implementation of hierarchical forecasting and reconciliation needs a case by case approach.

An important limitation of this study is that, only one type of forecasting model was used for the entire tree, the reconciliation process allows for different models per aggregation level or even per node, since only the in-sample errors, and not the method from which they come is important in the reconciliation.

Another limitation is that the reconciliation approaches, with the exception of

machine learning reconciliation, either have constant weights or the weights are a function of in-sample errors. Therefore they are not determined in reference to the metrics used to assess forecasting performance. And with larger hierarchical structures the in-sample errors can produce near-singular weight matrices if full covariance methods are used.

On the other hand machine learning reconciliation, solves the dependency on in-sample errors and can be very performant, but it can also fail drastically and provide the worst results, all depending on which building is taken in to account, in other words its performance depends on the time series in question. The reliability of machine learning methods must be assessed case by case before deploying it in a real production environment.

The inclusion of autocorrelation in the reconciliation algorithm proved effective in improving the base forecasts, but the gains in accuracy are still limited and the traditional variance scaling remains a central asset in the reconciliation methods.

Overall a thorough investigation on the temporal hierarchies was completed, pointing out the strengths and weaknesses of this method. A Python package, specialized for the task, was developed and is now available for public use. This thesis contributes to increase the importance of temporal hierarchies in the field of time series forecasting.

References

Abolghasemi, M., Hyndman, R. J., Tarr, G., and Bergmeir, C. (2019). Machine learning applications in time series hierarchical forecasting.

Aitken, A. (1934). On least squares and linear combination of observations. 55:42–48.

Athanasopoulos, G., Ahmed, R. A., and Hyndman, R. J. (2009). Hierarchical forecasts for australian domestic tourism. *International Journal of Forecasting*, 25(1):146–166.

Athanasopoulos, G., Hyndman, R. J., Kourentzes, N., and Petropoulos, F. (2017). Forecasting with temporal hierarchies. *European Journal of Operational Research*, 262(1):60–74.

Bourdeau, M., qiang Zhai, X., Nefzaoui, E., Guo, X., and Chatellier, P. (2019). Modeling and forecasting building energy consumption: A review of data-driven techniques. *Sustainable Cities and Society*, 48:101533.

Dev, S., AlSkaif, T., Hossari, M., Godina, R., Louwen, A., and van Sark, W. (2018). Solar irradiance forecasting using triple exponential smoothing. pages 1–6.

EU (2010). Directive 2010/31/eu of the european parliament and of the council on the energy performance of buildings.

Fildes, R., Randall, A., and Stubbs, P. (1997). One day ahead demand forecasting in the utility industries: Two case studies. *Journal of the Operational Research Society*, 48(1):15–24.

Fonzo, T. D. and Girolimetto, D. (2020). Cross-temporal forecast reconciliation: Optimal combination method and heuristic alternatives.

Gardner, E. S. (2006). Exponential smoothing: The state of the art—part ii. *International Journal of Forecasting*, 22(4):637–666.

Ghalehkhondabi, I., Ardjmand, E., Weckman, G. R., and Young, W. A. (2017). An overview of energy demand forecasting methods published in 2005–2015. *Energy Systems*, 8.

Gould, P. G., Koehler, A. B., Ord, J. K., Snyder, R. D., Hyndman, R. J., and Vahid-Araghi, F. (2008). Forecasting time series with multiple seasonal patterns. *European Journal of Operational Research*, 191(1):207–222.

Gross, C. W. and Sohl, J. E. (1990). Disaggregation methods to expedite product line forecasting. *Journal of Forecasting*, 9(3):233–254.

Hyndman, R. . A. G. (2018). *Forecasting: Principles and Practice*. OTexts, Melbourne, Australia, 2 edition.

Hyndman, R. J. (2020). A brief history of forecasting competitions. *International Journal of Forecasting*, 36(1):7–14.

Hyndman, R. J., Ahmed, R. A., Athanasopoulos, G., and Shang, H. L. (2011). Optimal combination forecasts for hierarchical time series. *Computational Statistics Data Analysis*, 55(9):2579–2589.

Hyndman, R. J. and Khandakar, Y. (2008). Automatic time series forecasting: The forecast package for r. *Journal of Statistical Software, Articles*, 27(3):1–22.

Hyndman, R. J. and Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679–688.

IEA (2016). Tracking clean energy progress 2016.

IEA (2020). The netherlands 2020, iea.

Impram, S., Varbak Nese, S., and Oral, B. (2020). Challenges of renewable energy penetration on power system flexibility: A survey. *Energy Strategy Reviews*, 31:100539.

Jeon, J., Panagiotelis, A., and Petropoulos, F. (2019). Probabilistic forecast reconciliation with applications to wind power and electric load. *European Journal of Operational Research*, 279(2):364–379.

Junker, R. G., Azar, A. G., Lopes, R. A., Lindberg, K. B., Reynders, G., Relan, R., and Madsen, H. (2018). Characterizing the energy flexibility of buildings and districts. *Applied Energy*, 225:175–182.

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. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Livera, A. M. D., Hyndman, R. J., and Snyder, R. D. (2011). Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American Statistical Association*, 106(496):1513–1527.

Makridakis, S., Spiliotis, E., and Assimakopoulos, V. (2020). The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54–74. M4 Competition.

Nystrup, P., Lindström, E., Møller, J. K., and Madsen, H. (2021). Dimensionality reduction in forecasting with temporal hierarchies. *International Journal of Forecasting*.

Nystrup, P., Lindström, E., Pinson, P., and Madsen, H. (2020). Temporal hierarchies with autocorrelation for load forecasting. *European Journal of Operational Research*, 280(3):876 – 888.

Pennings, C. L. and van Dalen, J. (2017). Integrated hierarchical forecasting. *European Journal of Operational Research*, 263(2):412–418.

Seabold, S. and Perktold, J. (2010). statsmodels: Econometric and statistical modeling with python.

Spiliotis, E., Abolghasemi, M., Hyndman, R. J., Petropoulos, F., and Assimakopoulos, V. (2020). Hierarchical forecast reconciliation with machine learning.

van Erven, T. and Cugliari, J. (2015). Game-theoretically optimal reconciliation of contemporaneous hierarchical time series forecasts. In Antoniadis, A., Poggi, J.-M., and Brossat, X., editors, *Modeling and Stochastic Learning for Forecasting in High Dimensions*, pages 297–317, Cham. Springer International Publishing.

Wickramasuriya, S. L. (2021). Properties of point forecast reconciliation approaches.

Wickramasuriya, S. L., Athanasopoulos, G., and Hyndman, R. J. (2019). Optimal forecast reconciliation for hierarchical and grouped time series through trace minimization. *Journal of the American Statistical Association*, 114(526):804–819.

Yang, D., Goh, G. S. W., Jiang, S., Zhang, A. N., and Akcan, O. (2015). Forecast upc-level fmcg demand, part ii: Hierarchical reconciliation. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 2113–2121.

Yang, D., Quan, H., Disfani, V. R., and Rodríguez-Gallegos, C. D. (2017). Reconciling solar forecasts: Temporal hierarchy. *Solar Energy*, 158:332–346.

Appendices

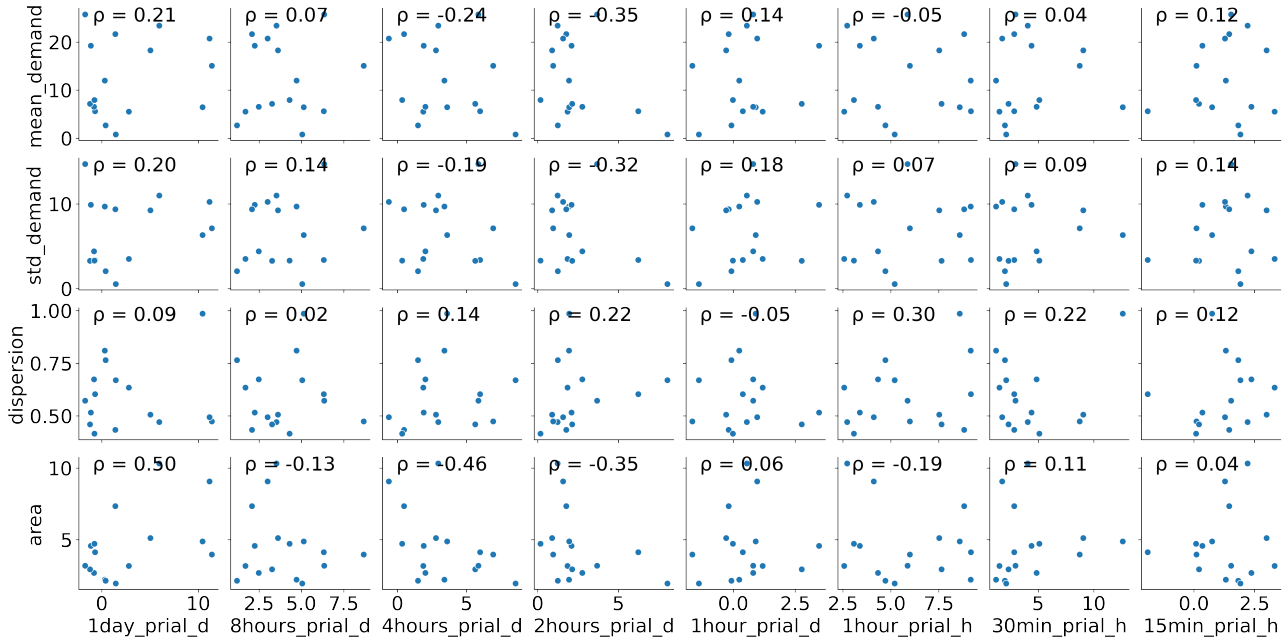

Figure A1: Reconciliation PRIAL plotted against building metadata ,namely: mean electricity demand, standard deviation of the electricity demand (both in kWh per 15min), dispersion of the electricity demand and building area (units of 1000m^2). A different hierarchical structure is indicated by the subscript _h for hourly or _d for daily tree structure. The Pearson correlation is presented for each scatter-plot and it shows a little to no correlation between reconciliation performance and building characteristics.

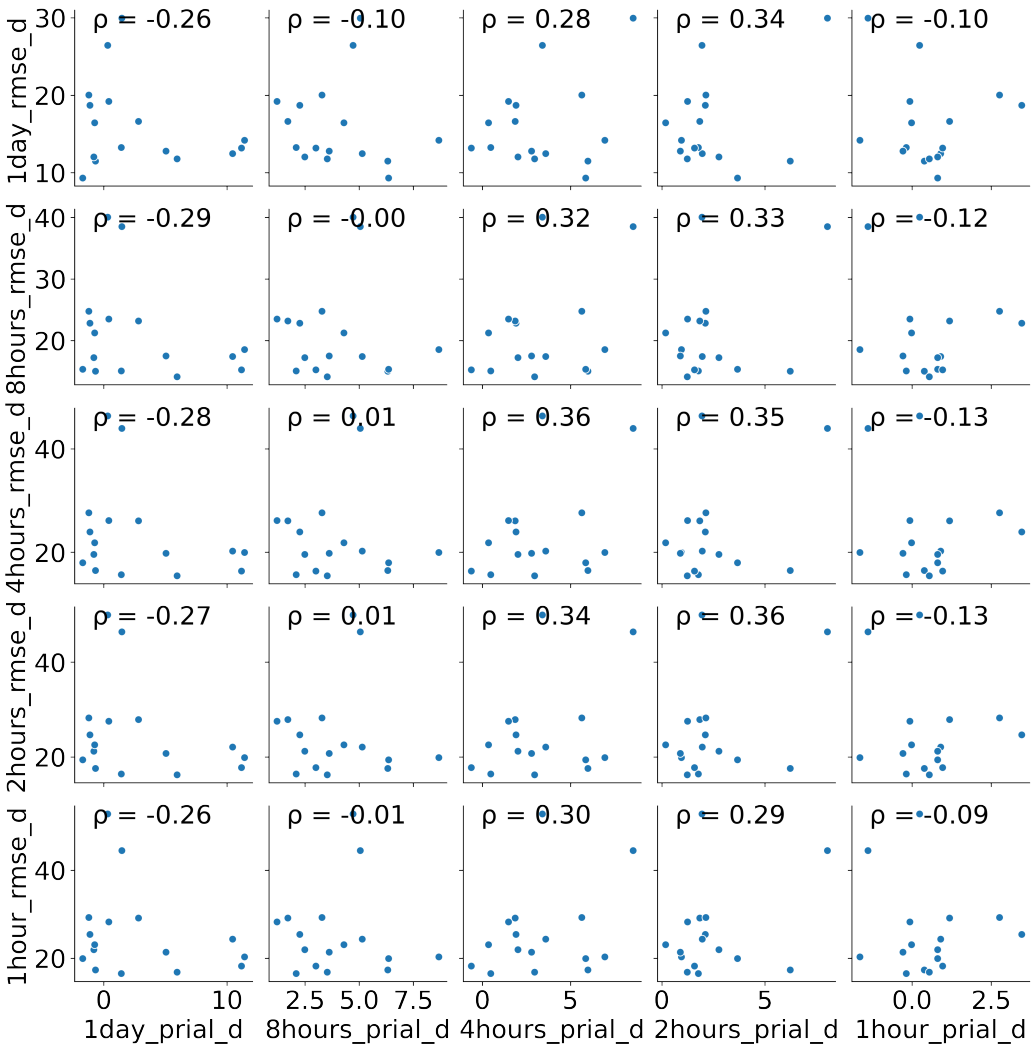

Figure A2: Reconciliation PRIAL plotted against forecasting RMSE for a daily tree structure. It shows no link between forecasting and reconciliation performance