Data mining/ Machine Learning for Smart House infrastructure

Tsalikidis Nikolaos

SID: 3308210044

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Data Science

JANUARY 2023

THESSALONIKI – GREECE

Data mining/ML for Smart House infrastructure

Tsalikidis Nikolaos

SID: 3308210044

Supervisor:	Prof. Christos Tjortjis
Supervising Committee Members:	Dr. Dimosthenis. Ioannidis (ITI/CERTH)
	Dr. Paraskeuas Koukaras

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Data Science

JANUARY 2023

THESSALONIKI – GREECE

To my father Peter,

my inspiration, my role model, my rock,

for his lifelong support and love.

Acknowledgments

Firstly, I would like to express my gratitude to my supervisor, Professor Christos Tjortjis for entrusting me to take on this master thesis, his continuous guidance and valuable advice to tackle any obstacles along the way.

I would like to also extend my gratitude to Dr. Dimosthenis Ioannidis for his involvement in the conceptualization of the research goals and his authorization to use the ITI Smart Home data to experiment with.

Also, I would like to sincerely thank PhD candidate Aris Mystakidis for his continuous support and for providing valuable guidance as well as key technical insights into my research. Aris inspired me with his passion for data modelling and Machine Learning models and without his active involvement, this master's dissertation could not have been successfully conducted.

Finally, I would like to dedicate this work to my family and especially my beloved father, who supported me and gave me strength throughout my academic journey.

Tsalikidis Nikolaos
7 January 2023

Abstract

This dissertation was written as a part of the MSc in Data Science at the International Hellenic University.

The recent exponential growth of available data in today's fast-paced technology-driven world has made systematic data collection and the concurrent extraction of meaningful information through them a necessity. In light of the adverse effects of climate change, data analysis and AI techniques can be directly applied to provide forecasts that could be used for the efficient scheduling and operation of energy usage. Especially in the built environment, Energy Load Forecasting (ELF) enables Distribution System Operators (DSO) or Aggregators to accurately predict the energy demand and generation tradeoffs. Today's near-zero Energy buildings (nZEBs), which are buildings with very high energy performance, have become the standard in Europe in terms of building performance from 2021 onwards and benefiting from the recent Internet of Things (IoT) technological advances, they are equipped with on-site smart monitoring systems and services, advanced communication technologies. The growing amount of data produced by such IoT applications has empowered decision-making via data analytics techniques.

This dissertation focuses on the development and comparison of predictive algorithms for a state-of-the-art nZEB smart building and metered data produced by its sensors and advanced monitoring systems. In particular, this involves energy load consumption, as well as weather data which are used to develop, train, and evaluate Machine Learning (ML) models. Various ML algorithms and methodologies, as well as combinations thereof, are explored and put to the test, each with its unique characteristics, in order to produce a robust approach for One-Step-Ahead Energy Load Forecasting (OSA-ELF). The effect of exogenous parameters is assessed, and comparisons are made between the different methods tested on both variating portions of the training dataset but also on new unseen data with similar properties and characteristics.

Contents

List of Figures

List of Tables

1 Introduction

One of the most dangerous global threats, climate change, is causing widespread disruption in the natural environment and affecting the lives of billions of people around the world, which calls for immediate actions to reduce the risks involved. To that end, the European Commission's action plan for climate change the so-called ‘‘Fit for 55” aims for Europe to achieve climate neutrality and 55% reduction of Greenhouse Gas (GHG) emissions by 2030 [1]. As part of the efforts to mitigate climate change and the looming energy crisis, the push for more energy-efficient buildings and cities is largely predicated on energy systems' capacity to cope with greater penetration of intermittent renewable energy resources under tight energy efficiency, adaptability, and resilience requirements.

At the same time, the daily interchange of data and information between systems and people has become the major driving force behind the technological development of the so-called ‘Smart’ devices. Ranging from smart phones and tablets to smart gadgets all throughout today’s household, it is apparent that consumers prefer intelligent or smart technologies and suites that may improve their daily routines. A prime example of this new trend is the concept of smart cities and smart buildings.

Indeed, the urban building infrastructure is at the center of the efforts to create smart grids, and to that end, benefiting from recent Internet of Things (IoT) technological advances, on-site smart monitoring systems and services, advanced communication technologies have been developed and installed today to facilitate this ‘smart’ transition[2]. Such technological applications enable interconnection over a network through which data and services related to smart meters and sensors can be exchanged and have modernized energy-management systems in the built environment[3]. At a grid level, this provides the capacity to dynamically adjust energy supply, which helps to reduce inefficient, which contributes to reducing energy wastage [4].

The profiling and forecasting of building energy usage, has attracted more attention from researchers in order to improve energy efficiency and reduce environmental impacts. For example, Demand Response (DR) aggregators are able to accurately predict the potential demand reduction capabilities, and grid operators to effectively plan ahead for their short and long-term supply requirements[5].

The abrupt increase in the amount of data in this new era of digitalization and IoT applications has empowered such decision-making via data analytics techniques. Both Data Mining (DM) and Machine Learning (ML) methods are implemented to analyze, process historical data, and use them to accurately forecast future energy load. The primary research goal of this thesis revolves around building energy load forecasting. This work aims to analyze the energy load time-series patterns of a smart building infrastructure and explore various predictive models and approaches to provide the most accurate predictions possible. To achieve these goals, the research steps for this study are the following:

- The first part highlights the concepts of Smart and 'near-zero Energy buildings' (nZEB) buildings, their characteristics and their interaction with monitoring and controlling systems, which aim to reduce energy consumption.
- Then, an extensive literature review provides key insights into several relevant studies, in which data-driven predictive models have been used for time series forecasting, with a focus on buildings' energy load. The state-of-the-art methods and their inner workings are highlighted. Also, a comparative basis is synthesized based on the evaluation metrics of relevant studies, in order to benchmark our developed models.
- The second part is focused on the data analysis approach, applied for the selection of input variables that should be introduced to the predictive models. This preselection process is useful for the development of predictive models since it clarifies parameters that mostly affect the total energy load. Overall, the utilized dataset includes both weather and consumption data collected for a Smart House infrastructure for almost 2 consecutive years.
- The third part is mainly focused on the development and validation of an innovative approach for One-Step-Ahead Energy Load Forecasting (OSA-ELF).

The analysis of the data and all the experimental modelling on this dissertation were executed in Python 3.9[1]. Specifically, the packages Pandas[2], Matplotlib[3], are mainly used for data preprocessing, analysis, and creation of visualizations. Also, all the implemented algorithms come from the Scikit-Learn [6] package and for Neural Networks the Keras[4]

[1] https://www.python.org/
[2] https://pandas.pydata.org/
[3] https://matplotlib.org/
[4] https://github.com/keras-team/keras

deep learning API package has been used. For the deployment of the models, the Visual Studio Code[5] and Jupyter Notebook[6] suites were used.

The structure of the thesis is the following. After a short introduction in Chapter 1, Chapter 2 includes an introduction to Smart/near-Zero Energy Building concept, Chapter 3 presents a literature review on time series problems and forecasting methods/algorithms Chapter 4 discusses the proposed methodology (along with a brief description of the datasets used, pre-processing steps and some exploratory data analysis results) while in Chapter 5 the modelling approach and results are presented. In Chapter 6 the results are discussed and evaluated and in Chapter 7 conclusions and future directions are highlighted. Finally, elements concerning the validity of our approach and future research directions are touched upon in the last two chapters.

[5] https://code.visualstudio.com/
[6] https://jupyter.org/

2 Smart/nZEB buildings

2.1 Smart building concept

Over recent decades a rapid urbanization rate has taken place, as more and more people have been moving to urban areas. In fact, forecasts indicate that by 2050 more than 67% of the total global population will be in urban habitats [7]. Buildings are a focal point of our everyday routines, especially in urban areas, as we spend a significant portion of our time in them, whether at home, at work, or in our leisure time. At the same time, the flow of data between systems, sensors, and users, the so-called internet-of-things (IoT) has deemed the term ‘smart’ relevant to almost everything, especially for urban residents.

Often there is some confusion or a lack of common understanding of the definition of a smart building. Smart buildings are designed to utilize Information and Communication Technologies (ICT) to automate and better control their core functions. Unlike traditional buildings, which feature systems that operate separately, smart buildings employ interconnected ICT (i.e., sensors, meters, wireless devices) in order to improve the daily lives of tenants but at the same time improve overall building energy performance. Smart buildings also enable building operators and tenants to interact with the structure, offering visibility into its operations and day-to-day functions data [8]. The infrastructure of such systems may offer for example a precise prediction of the following day's energy use, allowing them to plan their activities sensibly, such as transferring the load of their demands to lower demand hours, in order to save money and operate in a more ecologically friendly manner. A smart building has become the new standard not only for residents but also for businesses that integrate intelligence into their goods/services and aim to reduce their ecological footprint. In the context of the present thesis, the term smart buildings should be construed broadly, since possible applications to various types of buildings are discussed, including commercial (e.g., offices), residential (smart houses), and public buildings (universities, hospitals, etc.)

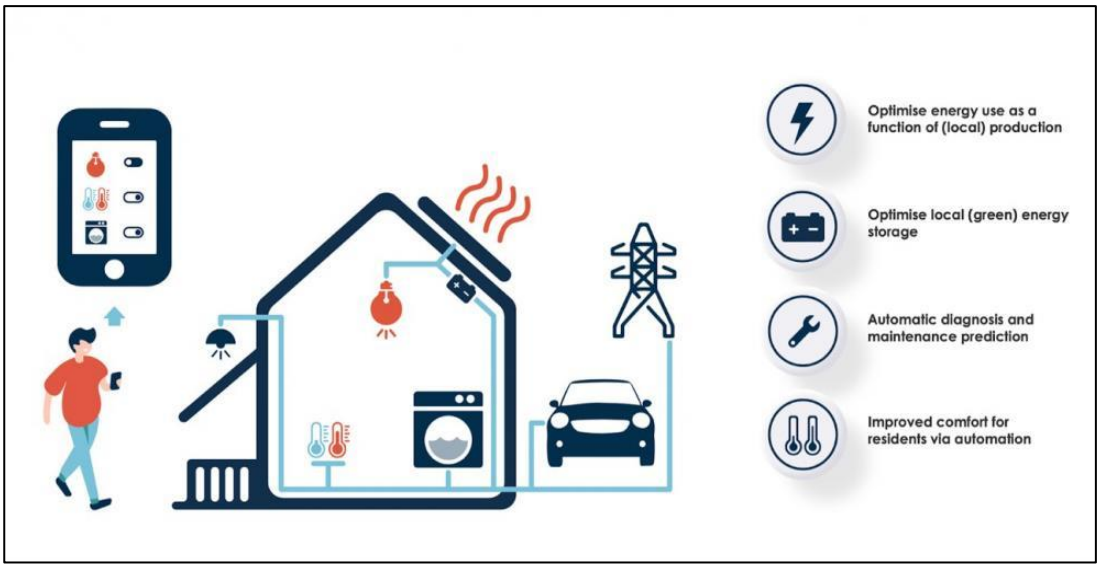


Figure 1: Advantages of utilizing smart technologies in buildings [9].

2.2 nZEB buildings

The built environment has been identified as a major contributor to fossil fuel energy use and global carbon dioxide emissions. Increased heating and cooling demands in the building sector are one of the primary causes of rising energy consumption and emissions globally. More precisely, overall, the built environment accounts for the largest share of the total EU energy consumed (40%) and produces approximately 35% of all energy-related GHG emissions[10].

A near-zero Energy building (nZEB), constitutes a building that has high energy efficiency, requiring nearly zero or very low amount of energy for its daily needs, to be at least partially recovered from renewable sources. In order to mitigate carbon emissions from the building sector the newly revised Energy Performance of Building Directive 2021/12/14EU (EPBD) obligates EU Member States to achieve a minimum 55% reduction in greenhouse gas (GHG) emissions by 2030, compared with 1990 levels [11]. The abovementioned directive with its latest updates makes nZEB the standard for new buildings until the application of the zero Energy building (ZEB) standard in 2030, which will replace the nZEB. As a result, many building energy-efficient technologies, such as passive design and active system measures (e.g. photovoltaic panels), have been proposed and developed to meet these new European regulatory targets of lowering building energy load and GHG emissions, some of which are illustrated in Figure 2.

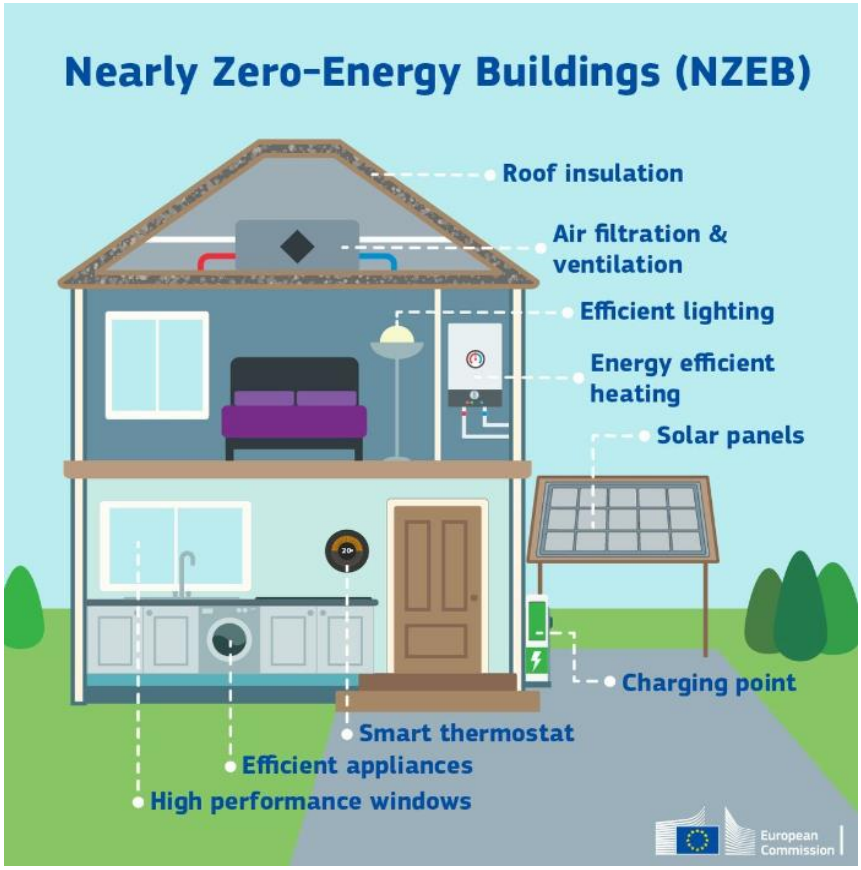


Figure 2: nZEB concept [12]

Today, there seems to be a high interaction of nZEB and ICT, as an integrated design approach, in order to become "smarter" and more eco-friendly in their daily operations [13]. For example, solar energy varies depending on weather conditions, hence the use of photovoltaics cannot always result in a steady stream of energy supply in this case, an IoT-enabled demand-driven smart grid with nZEBs can help addresses issues related to energy generation periodicity [14].

2.3 Case study: CERTH/ITI Smart House

In the context of the present thesis, the research activity involves the use of data mining/ML algorithms to improve the lives of urban residents of a modern-day smart house. This research focuses on utilizing metering or sensor data on specific time intervals to apply predictive and parametrized models. The data derives from a selected publicly available dataset which consists of measurements and data collected from and for the Centre for Research and Technology-Hellas (CERTH) Smart House infrastructure in Thessaloniki, Greece. This building is a prototype infrastructure functioning as a living

lab which is deployed and operated by the Information Technology Institute (ITI)[7]. It makes use of various state-of-the-art technologies in an effort to emulate to a great extent a real domestic building where residents can explore cutting-edge smart IoT-based technology while experiencing real-world living situations, with services for energy, health, big data, robotics, and artificial intelligence (AI) being offered [15].

Figure 3: CERTH/ITI Smart House and its roof-based PV Installation [16]

The Smart house is equipped with a variety of sensors, and devices such as smart meters, environmental sensors, occupancy sensors, smart appliances, weather stations, etc. These instruments enable continuous monitoring of the energy consumption(load), energy production from photovoltaic(PV) arrays and living conditions, whereas automated algorithms perform automation and efficiency scenarios while taking occupant preferences into account.

It is important to highlight, that the ITI Smart House, as part of CERTH, is used as an active workspace for researchers, hence as it would be analyzed in the next sections, the energy load patterns are closer to an office-type of building, rather than a typical residential home.

[7] www.iti.gr

3 Time series forecasting

3.1 Introduction

Smart buildings ought to be capable to adapt their operations to fluctuating conditions and constraints such as occupants' needs, climate change effects, electricity pricing, regulatory requirements etc. Core functions that are associated with a building operating environment, such as energy load, are prone to unpredictability, necessitating the system-level capability of learning and adapting [17].

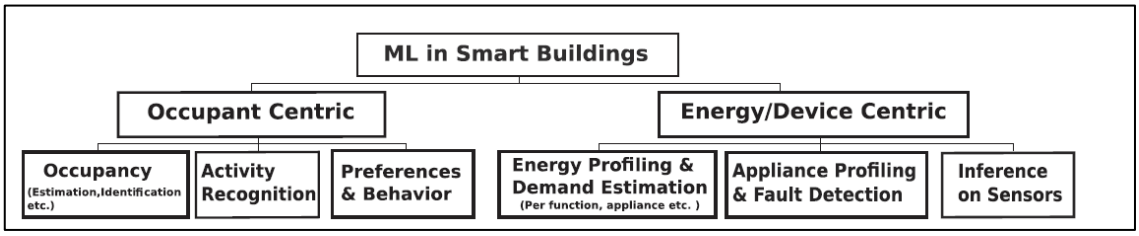


Figure 4: The range of applications of ML in modern smart buildings [4]

At the same time, Machine Learning (ML) has become a key enabler of data analytics and enhanced control of energy systems. As already mentioned, the technological advancements regarding IoT devices have now made it possible to acquire a building's energy load data via smart meters [18], which form a so-called time series. In general, time series is an ordered sequence of values of a variable that includes information about the time at which they were recorded (i.e. timestamp). Time series analysis entails comprehending the internal structure that these temporal points may have, while time series forecasting is the prediction of a future value over a period of time. The complexity of time series forecasting spans from predicting a single future value using a handful of past values to predicting multiple future values for many variables at each time step.

3.2 Energy load forecasting (ELF)

3.2.1 Forecasting horizon and time steps

Energy load forecasting (ELF) to precisely estimate energy load is one of its most sought-after applications in recent times. Precise ELF can lead to the proactive optimization of control choices in order to achieve improved energy efficiency, and lower operating costs in various domains and applications [19]. Overall, ELF can be categorized into four general groups, classified based on the time horizon of their forecasts, as follows [20], [21]:

1) Very short-term load forecasting (VSTLF), aims at very-short time intervals, with their prediction ranging from a few minutes to up to one hour (1h)
2) Short-term load forecasting (STLF), which typically deals with a prediction range of 1h up to 2 weeks and is a very commonly used forecasting horizon
3) Medium-term load forecasting (MTLF), extends the prediction range from a couple of weeks to up to 1 year
4) Long-term load forecasting (LTLF), which targets a range of more than a year

Short-term load forecasting data that are accurate can be utilized to create a more appropriate building energy scheduling plan [22]. Medium-term load forecasting is often used for scheduling maintenance and power supply [23]. Since practical limitations exist on the storage of electricity, electricity generation, transmission, and distribution must occur concurrently with demand. As a result, for the uninterrupted operation of national rids, the electric energy supply and demand must constantly maintain a dynamic equilibrium [5]. Hence, long-term load forecasting empowers energy producers and grid distributors to dynamically adjust their generation and transaction plans in the market [24].

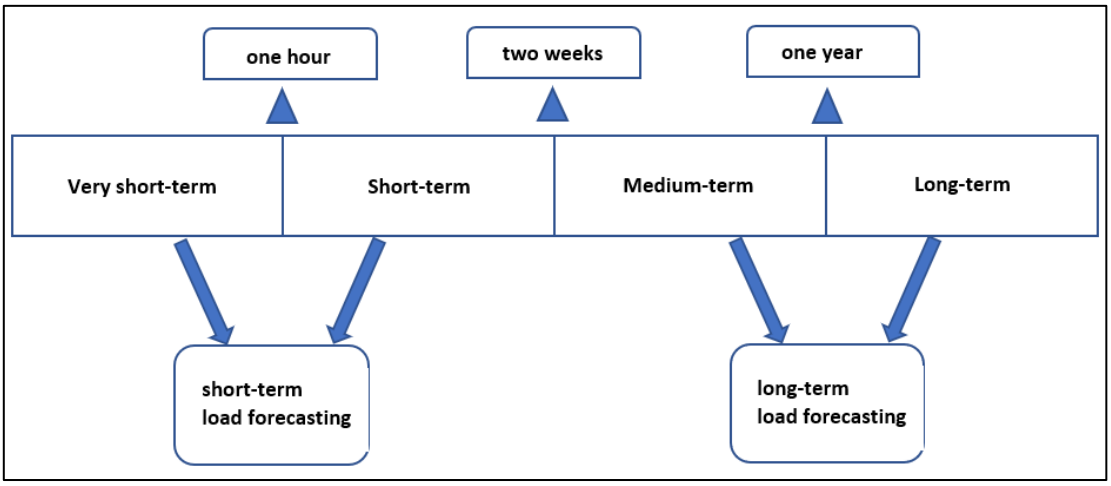


Figure 5: Time-based classification of ELF horizons

Energy load prediction can be generally divided into two subgroups, in terms of forecasting steps. One-step forecasting, which estimates future demand or supplies one step ahead in time (e.g. an hour ahead), and multi-step forecasting, which predicts variating time intervals into the future [25].

3.2.2 Characteristics of Building Energy Loads

Building load forecasting, which produces predictions not only on the lower brackets of the electricity networks, namely residential homes but also for large-scale consumers such as commercial businesses and industry, accounts for a large portion of the overall energy load forecasting problem for the power industry. Monitoring the energy load of a building not only enables occupants to have insights into their consumption patterns but also acts as a vital tool to provide accurate information to power/utility suppliers to improve their distribution strategies and scheduling of electricity supply to the grid [26], [27]. What is more, load forecasting has now become increasingly important with the inclusion of smart grids in cities, and smart energy management systems including electric vehicle charging infrastructure, since they require precise forecasting to maintain optimal grid functioning. Certain factors that affect ELF include but are not limited to [28]:

- **Socioeconomic factors**: population, GDP, income, type of industry
- **Temporal elements**: seasonal effects, day of the week, hour of the day, national holidays
- **Weather and climate effects**: ambient temperature, relative humidity, wind speed, cloud coverage, etc;
- **Pricing aspects**: real-time electricity pricing, fuel pricing

Since such factors affect load forecasting, ML methods are often enriched with the inclusion of additional features i.e., weather information, temporal features, and demographic data to improve their forecasting accuracy. If there are such multiple variables introduced, then such a forecasting problem becomes from univariate, a multivariate problem. In such techniques, historical climatic data, as well as historical time series of energy load, are translated to a plethora of input variables. These features are utilized with energy load time-series data to generate multidimensional input parameters, and while more sophisticated than typical forecasting models, they yield more accurate forecasts [22].

However, at the same time due to the constantly changing environment, energy load patterns of residential as well as commercial buildings often come with complex non-stationary characteristics and display seasonal trends [29]. Despite STLF being a preferred method for energy load management, it can often be more challenging than mid- and long-term forecasting because of the greater variance in the respective energy load patterns [30]. The high variance in the respective energy load patterns is because the individual household energy load patterns can vary depending on the daily routines of the customers, and the short monitoring period produces a high-frequency energy load graph. Another reason why individual building energy load forecasting is challenging is because of the unique nature of the energy load pattern of different buildings, which adds additional obstacles in developing a single generalized model that can adjust to the consumption patterns of each building[19].

3.3 Forecasting methods

Based on our literature review, three main methods can be found for performing ELF [18]:

1. Physics principles and thermodynamic rules to estimate and analyze energy load
2. Statistical based methods
3. Machine learning-based models

With the emergence of big data in energy-related domains, there is an inclination by researchers to implement more data-driven methods instead of physics-based methods in load forecasting[31], and in the context of the present Thesis, we will focus on the second and third group.

3.3.1 Statistical based methods

For the second group, the so-called traditional statistical methods are implemented for time series predictions regarding energy load, since it is primarily a univariate time series. The basic concept behind such methods is to generate a stationary time series (e.g., a series with constant mean/variance), by filter-ing out the relevant pattern from the series (trend, seasonality, etc.). Some of the most widely used methods for time series analysis and forecasting are the Autoregressive Integrated Moving Average (ARIMA) and the Autoregressive Moving Average (ARMA) a special type of ARIMA where differencing is taken into account and other variants, for example, SARIMAX which deals with

seasonality effects in the time series and incorporates exogenous variables (e.g. weather effects) [32], [33]. The ARIMA family of models predicts future values based on its own past values of a time series.

However, despite the simplicity and high interpretability of such statistical methods, they do not yield convincing forecasting accuracies for complicated data patterns, compared to the more advanced ML models of today, especially deep learning. This is mainly due to the fact that they fail to address potential nonlinear dependencies within the data [34], [35]. As ML techniques are increasingly gaining prominence in recent times there have been numerous studies and research papers [5], [16], [20], [36]–[40], which highlight the weaknesses of these traditional statistical methods compared to the state-of-the-art ML algorithms, especially for short-term predictions, in terms of predictive performance (Figure 6).

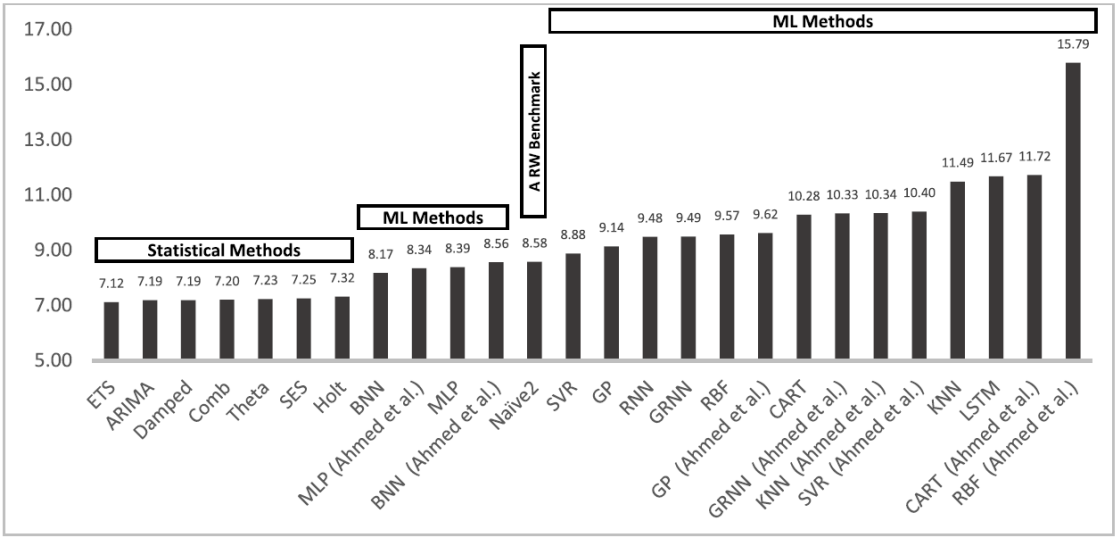


Figure 6: Forecasting performance (sMAPE) of ML and statistical methods, for one-step-ahead-forecasts [41]

3.3.2 ML algorithms and approaches

ML can be broadly classified as supervised, unsupervised, or reinforcement learning. In supervised learning, a considerable collection of historical data is utilized to train a mapping from independent variables to a projected dependent variable. Regarding building operations, smart meter measurements are frequently fed into the models to produce time-series forecasts or classifications. To tackle the complexities and fluctuations in energy load data, the newest research activity focuses mainly on ML techniques. Despite

extensive research efforts over the last few years, the achievement of highly accurate energy load forecasts still remains an open challenge.

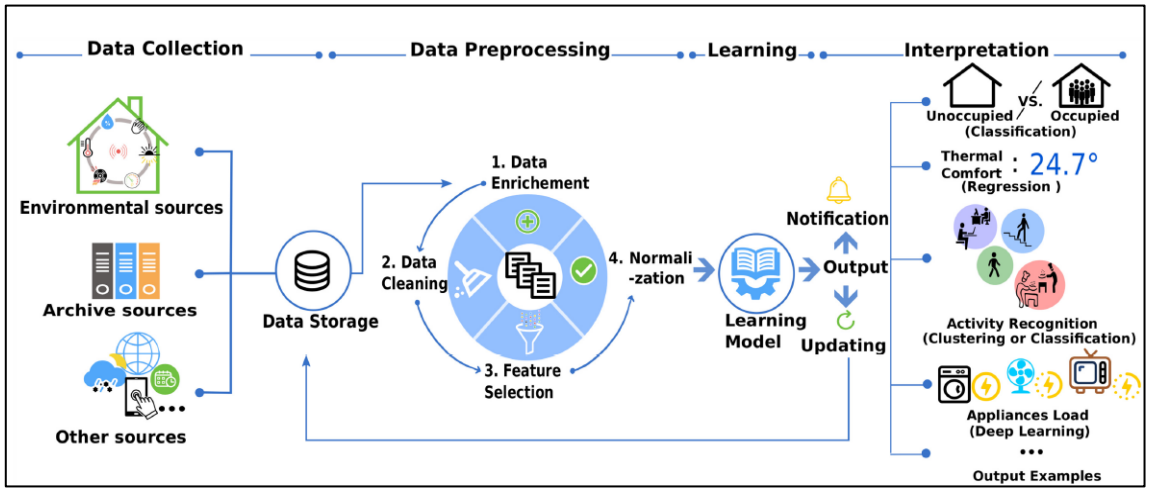


Figure 7: Outline of ML framework for Smart building operations [4]

Tree-based models and ensemble methods

A regression tree, i.e. a decision tree-based method for regression, is characterized as a base learner in the field of ML. The key benefit of decision-trees algorithms over linear regression models is that in cases when the connection between the characteristics and the response is significantly non-linear and complicated, decision trees may outperform traditional ML algorithms, such as linear regression. Ensemble methods are ML techniques that combine several simpler base models/learners in order to produce one optimal prediction model [42]. By combining predictions, ensemble techniques can deliver more reliable and robust forecasts than a single baseline prediction model[43]. Ensemble prediction approaches can be categorized into two general types, based on the base model generation process[44]:

1. *Homogeneous ensemble* model creates its base models by resampling from the original data different training data to the same learning algorithm with the same parameter values.
2. *Heterogeneous ensemble* model creates its basic models by running the same training data through multiple learning algorithms or running the same algorithms through alternative parameter settings.

In other words, heterogeneous ensemble aims to improve the forecast the prediction by exploiting the advantages of the interdependency between different ML algorithms, whereas the homogeneous ensemble model is similar to an optimization process that

improves the performance of a specific learning algorithm by training it multiple times with different datasets and combining their predictions.

Homogeneous ensembling: Bagging

The collection of 'M' base learners to ensemble is formed by bootstrap sampling on the training data in the case of bagging (bootstrap aggregating). M new training datasets are generated from the original training data set using random sampling and replacement, with each observation having the same chance of appearing in the new dataset. A new observation is predicted via bagging by averaging the responses of the M learners to the incoming input (or majority vote in case of classification problems). When we use bagging on regression trees, each individual tree has a high variance but generally a low bias. Averaging the resulting forecast of these N trees reduces variation and increases accuracy significantly [42].

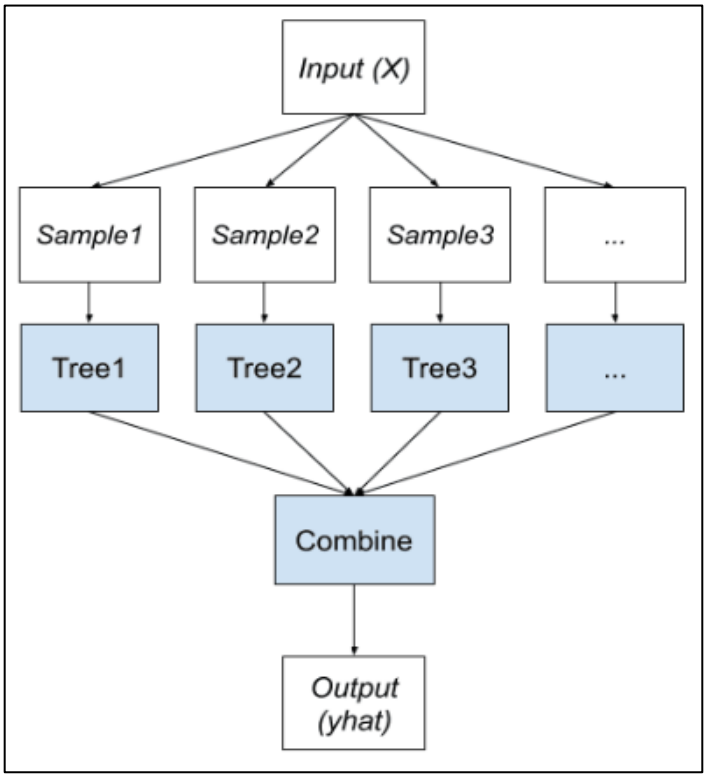


Figure 8: Bagging Ensemble method [45]

One popular example of a bagging is the **Random Forest** algorithm. The Random Forest algorithm is actually an extension of bagging, creating bootstrap samples of the data and then using them to grow a regression tree, a random subsample of the features is used in each fitting process. This process occurs a predefined number of times and the prediction

is then the aggregation of the individual trees' predictions, which for regression problems involves some type of averaging [46].

Heterogeneous ensembling*: *Boosting

Boosting is an ML strategy that learns from prior predictor errors in order to produce better predictions in the future. In essence, it creates a strong learner by merging a subset of lesser learners with low complexity, shorter training duration, and greater resistance to overfitting [47].

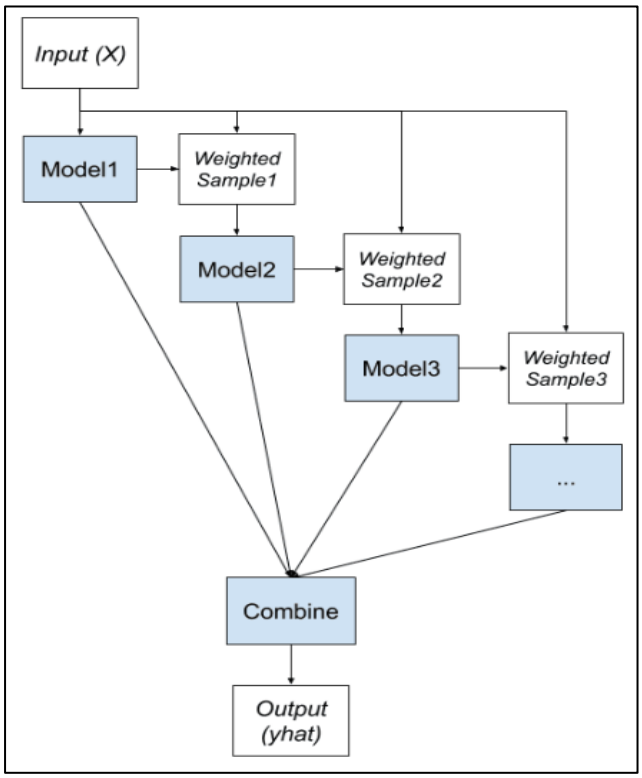


Figure 9: Boosting Ensemble method [45]

Boosting takes many forms, including gradient boosting. Gradient descent is a popular boosting strategy in classification and regression applications. It creates models in the shape of a group of poor learners and most implementations use decision trees as base predictors. The primary concept is to use a gradient descent learner to optimize the loss function [48]. Notable examples of Gradient Boosted Decision Trees (GBDT) used due to their low computational complexity and effectiveness are highlighted below.

1. **Extreme Gradient Boosting (XGBoost)**

   The XGBoost algorithm implements the weak learner by optimizing the structured loss function, and the XGBoost algorithm does not use the linear search method,

it directly uses the first derivative and the second derivative of the loss function. XGBoost has some key advantages such as better regularization in order to avoid overfitting, automatized handling of missing values, and cross-validation at each iteration step [49]. XGBoost is considered today as state-of-the-art and one of the most advanced supervised ML algorithms. It produces faster results and better performance scores in comparison with other ensemble methods [50].

2. **Light Gradient-Boosting Machine (LGBM)**

   Compared to other decision tree algorithms, LightGBM employs a novel method called gradient-based one-side sampling in order to pinpoint the optimal split for input data. LightGBM expands on a leaf-basis in contrast with other boosting algorithms that have a level-wise or depth-wise decision tree growth[51]. It is considered that the LGBM is lightweight and requires fewer resources than other GBDTs, resulting in faster training times and less computational needs, while maintaining high accuracy [52].

3. **CatBoost**

   CatBoost (for "Categorical boosting") builds symmetric (balanced) trees, unlike XGBoost and LightGBM. The leaves from the previous tree are divided using the same criteria in each phase. The feature-split pair with the lowest loss is chosen and utilized for all nodes in the level. Such a balanced tree structure contributes towards the efficient use of computational resources, decreases training/testing times, and reduces overfitting as the structure serves as a regularization method. Furthermore, CatBoost is designed for both numerical and categorical features and takes advantage of dealing with them during training as opposed to preprocessing time [53], [54]. Overall, CatBoost is highly effective in forecasting cases involving categorical, heterogeneous data.

Artificial Neural networks (ANNs)

An ANN is an advanced mathematical model that tries to emulate the function of the human brain, consisting of layers of connected units called neurons that are interlinked together with a weight value assigned for each connection. Today, as deep learning has become one of the most active technologies in many research areas, ANNs are heavily

implemented for image pattern recognition, algorithm optimization, and time-series prediction [55], [56]. In the context of the present thesis there has been much attention in using deep learning algorithms for time series prediction, and hence we provide a short review of the basic principles, strengths and weaknesses of notable ANNs.

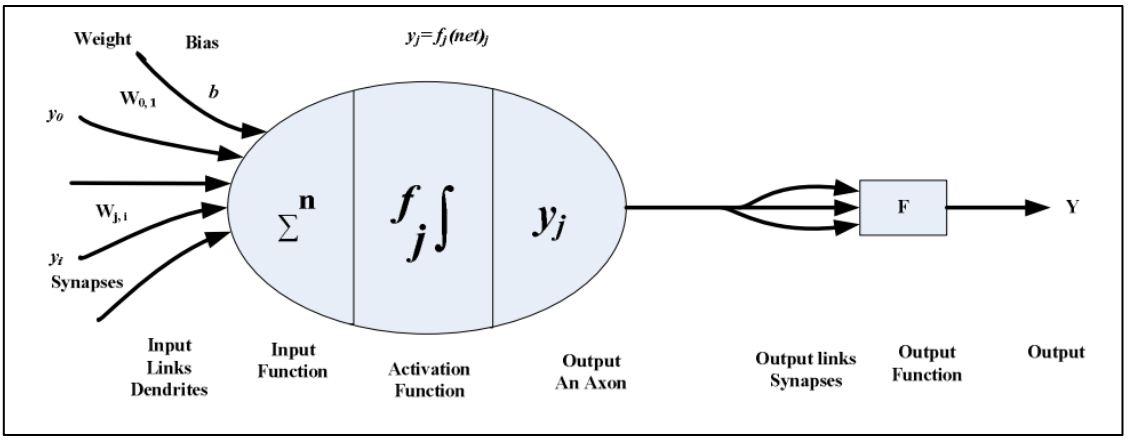


Figure 10: Basic Artificial Neural Network (ANN) structure [57].

In contrast to ARIMA-based linear forecasting approaches, ANNs are a collection of non-linear self-adaptive methods that are data-driven, which implies that no previous knowledge of the connection between the models and the data variables is required. It is commonly known that ANNs can approximate any nonlinear function [58]. Moreover, compared to decision-tree based models, deep learning models automatically extract features along with making predictions, which results in much more accurate load predictions. However, these deep networks often need to be trained with large amounts of data in order to achieve optimum accuracies, and this tends to make them more time-consuming and computationally intensive [59].

Multilayer Perceptron

A very common ANN architecture is known as a Multilayer Perceptron (MLP), consisting of a network of individual nodes, called perceptrons, organized in a series of layers, one input layer, a single hidden layer, and an output layer. The basic structure of MLP is shown in Figure 11. The overall learning process is based on the backpropagation error utilizing the using the gradient descent search method. Despite its simple concept and design, an MLP can learn any function, particularly by approximating non-linear correlations between input and output variables [60]. An MLP has a very good capability of accurately learning from the series of past observations of a time series and making predictions of the next value or sequence of steps. However, the training process may require a high computational burden to be processed with a slow convergence of weights.

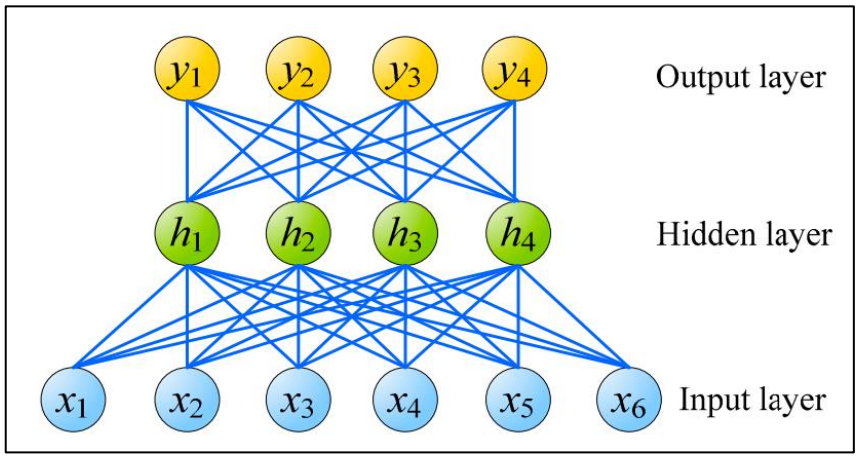


Figure 11: The Multilayer Perceptron (MLP) structure

Recurrent Neural Networks: Long Short-Term Memory (LSTM)

In earlier stages, neural network topologies, such as MLP, often treated each data point input as a separate parameter. However, data points are frequently interdependent on one another, as in natural language modeling or time series forecasting. Recurrent Neural Networks(RNNs) are designed to remember parts of past data through a methodology, called feedback, in which the training takes place not only from input to output (as feed-forward) but also utilizes a loop in the network to preserve some information and thus functions like a memory [61].

An ANN utilizing Long Short-Term Memory (LSTM), is a unique RNN architecture that can be utilized to learn temporal sequences and deal with long-term dependencies, which is in fact the basis for time series forecasting [62]. LSTM was designed to overcome the vanishing gradients problem of the standard RNN when dealing with long-term dependencies. In contrast to the standard RNN which has a series of repeating modules with a relatively simple structure, the hidden layers of LSTM have a more complicated format. Specifically, instead of neurons, LSTM networks have memory blocks that are connected through layers and introduce the concepts of gate and memory cells in each hidden layer. The basic structure for the LSTM model consists of four parts: an input gate layer, a forget gate layer, an output gate layer, and a memory cell layer, as depicted in Figure 12 [27].

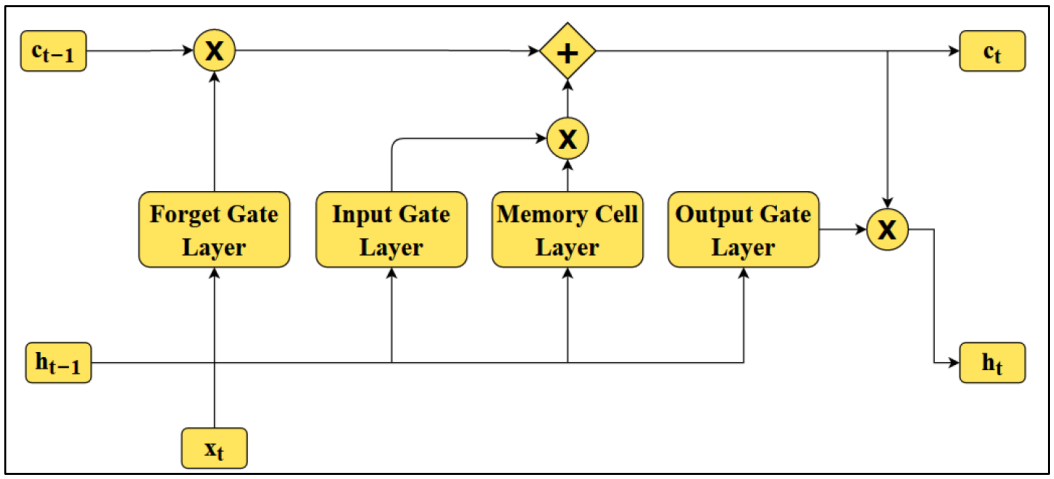


Figure 12: The basic structure of LSTM model [27]

The above structure enables LSTM to make the decision of preserving or ignoring the memory information. This results in the LSTM model capturing important features from input data and preserving this information over long time intervals. The decision of deleting or preserving the information is made based on the weight values assigned to the information during the training process. Hence, an LSTM model learns what information is worth preserving or removing.

Over recent years, several research/scientific papers [18], [27], [31], [58], [61]–[66] have emerged, which thoroughly explore the performance of LSTM particularly in time series forecasting (for variating forecasting horizons) and particularly ELF. In fact, some studies support that LSTM is highly suitable to forecast complex energy load time series that is characterized by non-stationarity effects [31], [58]. Especially for individual residential load forecasting, which is typically more inconsistent mainly due to the behavioral habits and lifestyle of residents, LSTM has proven to be able to better capture the associated long-term temporal dependencies [30], [63]. Another notable study [65] which implemented pooling data in order to combine data from interconnected users, showed very good results compared to other 'traditional' ML algorithms. However, LSTM models often come with certain notable drawbacks:

- They have a large range of parameters that need to be finetuned to reach an optimal structure to maximize their performance
- They tend to require a lot of computational resources and time for their training process

3.4 Cross-validation techniques

The performance of any ML model is primarily assessed by its performance when tested on new (unseen) data. Cross-validation (CV) is one of the primary techniques used to test the effectiveness of a model, by re-sampling and splitting data into two parts (referred to as splits) [67], one used to learn or train a model and another used to validate the model. Adopting appropriate CV techniques when dealing with complicated forecasting problems is critical for objectively replicating post-sample accuracy, avoiding over-fitting, and managing uncertainty.

A typical approach in CV is the k-fold validation method, in which data are randomly shuffled and split in K equally sized folds or blocks. Each fold is a subset of the data comprising N/K randomly assigned observations, where N is the number of observations. After this partitioning, k iterations of training and validation are performed such that within each iteration a different fold of the data is picked for testing for validation while the remaining $K - 1$ folds are used for training. However, as we already discussed, time series data due to their temporal structure are interdependent, such dependencies mean that it is not possible to randomly mix data in a fold, thus the use of standard cross-validation (i.e. k-fold) ineffective.

As a result, since we are working with a time series case study, a distinct portioning strategy is chosen. Time-series split is a sort of train-test split that tries to evaluate model predictions regardless of how train-test data sets are split. The model is trained on a subset of the time series from the beginning through time T, and predictions for the future T+i steps must be obtained, as well as errors. The training sample is then extended to T+i values, predictions from T+i to T+2i steps are produced, and the process of extending the time series' test segment is repeated until the final available information is reached.. In other words, the time series split ensures the train sets precede the test datasets, as depicted in Figure 13, which captures the temporal interdependencies of the time series.

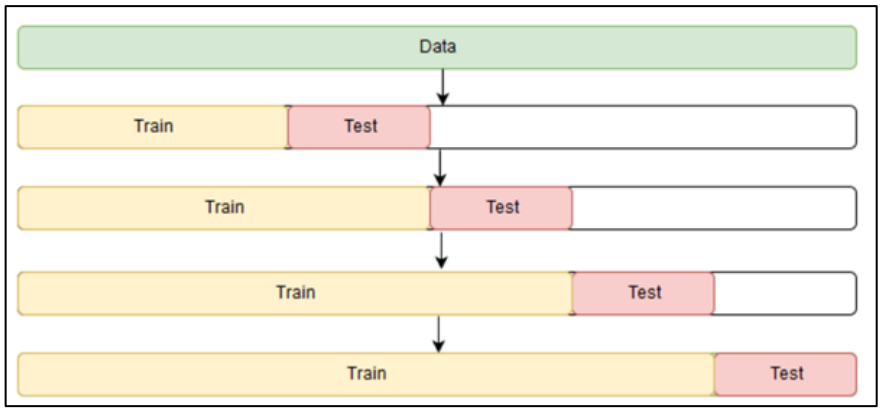


Figure 13: Time series cross-validation split strategy [68]

3.5 Relevant studies and metrics

For the present thesis, a systemic approach to the review of literature has been employed. A comprehensive literature search was conducted and was followed by a meta-analysis of the findings. To that end, key research queries and keywords were developed, like "*energy load/consumption timeseries forecasting*", "*electric/energy load forecasting models*", "*electric/energy load prediction models*" or "*electric/energy load demand models".* After this first search, texts including the keywords "*office*", "*building*", "*household*", "*smart building(s)*", "*building energy load/consumption*" were further targeted.

This initial screening process was followed by the analysis, which involved examining the scope and scenario objectives of each scientific work. The attention then shifts to an examination of the forecasting models used, as well as the prediction horizon, variables, and techniques applied. Finally, relevant patterns and repeating methods in the application of the chosen forecasting models are examined. In total, more than 70 scientific papers and research outputs were finally gathered, as a result of the above procedure, which were categorized on a structured basis, based on their content, source, year of publication, relevance etc.

Overall, over recent years, researchers have been developing and experimenting with various ML algorithms such as Linear Regression models, Neural Networks (NN), Decision-tree based models, and Support Vector Regression (SVR) for building ELF with promising results [69]. As a summary, emphasizing performance and relevance, this section includes a structured presentation of the gathered key literature in terms of ELF and notable performance metrics, mainly MAPE, since it is the most used metric for direct comparisons illustrated in Table 1.

Table 1: Taxonomy of related literature findings in building-related ELF

#	Model	Input resolution	Exogenous Inputs	Forecast horizon	Indicative Metrics	Reference
1	BPN[8]	15 min	No	2h	MAPE: 6.21%	[20]
2	SVR	15 min	No	2h	MAPE: 5.2%	
3	MLP	1h	Yes	24h	MAPE: 1.7, R2: 0.98	[24]
4	MLP	15 min	No	1h	MAPE: 12.96%, RMSE (kWh): 0.651	[21]
5	LSTM	15 min	No	1h	MAPE: 14.09%, RMSE (kWh): 0.668	
6	XGBoost	15 min	No	1h	MAPE: 15.37%, RMSE (kWh): 0.699	
7	LGBM	30 min	Yes	1h	MAPE: 21.5%	[34]
8	LSTM	30 min	No	1h	MAPE: 9.1%	[22]
9	XGBoost	1h	Yes	1h	MAPE: 12%	[70]
10	XGBoost	1h	No	1h	MAPE: 3.50%	[42]
12	EBT[9]	15 min	Yes	1h	MAPE: 4.62%	[69]
13	XGBoost	1h	Yes	1h	R^2 : 0.977	[5]
14	RF	1h	Yes	1h	R^2 : 0.972	
15	MLP	1h	Yes	1h	R^2 : 0.98	
16	LSTM	1h	Yes	1h	R^2 : 0.977	
17	FF-DNN[10]	1h	Yes	1h	MAPE: 1.42%	[28]
18	GBDT	1h	No	24h	MAPE: 1.32%	[71]
19	RF	1h	No	24h	MAPE: 1.97%	
20	MLP	15 min	No	24h	MAPE: 14.53%	[26]
21	ELM[11]	15 min	No	24h	MAPE: 14.54%	
22	Model Ensemble	15 min	Yes	24h	MAPE:2.32%	[72]
23	ANN	1h	Yes	1h	MAPE: 5%	[2]

This summarized table acts only as a reference basis, to compare and contrast the results of the developed models of the present thesis. That said, each of these forecasting models

[8] Backpropagation Neural Network
[9] Ensemble Bagging Tree
[10] Feed-forward Deep Neural Network
[11] Extreme learning machine

has advantages and disadvantages and none is considered 100% efficient and applicable to all case studies. It should be emphasized that this listing comprises a detailed assessment of the results published in the models' respective publications, rather than an experimental evaluation by replication of the reported results. The scope, objectives, forecasting horizon, input data resolution and magnitude, model variables, and pre-processing techniques of the reference models differ from the present thesis's models. Also, some of the mentioned studies implemented some type of data smoothing, aggregation and/or clustering method. Hence the referenced metrics should be treated with a critical approach, more as a cumbersome point of comparison which has some scalability. As a notable example, Papadopoulos & Karakatsanis, 2015, show a very good performance of their forecasting model (MAPE around 1.5%), so we tried to understand the potential reasons behind this:

- **Area of application**: It is far bigger than a single building as they deal with an entire region in the US, the New England Electricity Market (with the Boston area as its epicenter)
- **Input resolution**: The load dataset spans over a 4-year period, from 2009 to 2012(Figure 14).

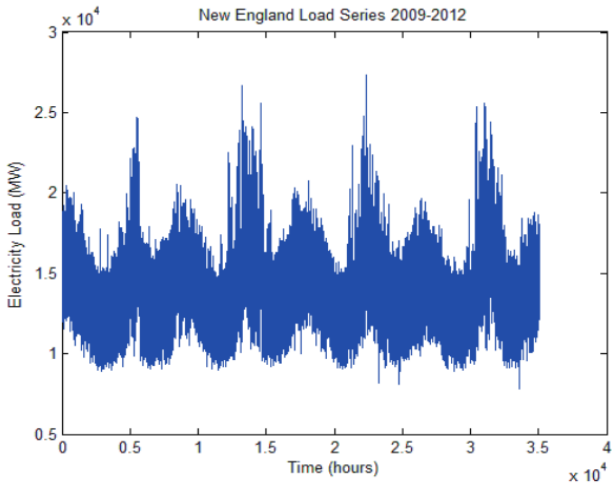


Figure 14: Energy load time series used in [71]

The dataset has a distinct pattern and repeatability, while its max/min values seem in the same range. This behavior is possibly due to it being a whole area as an energy consumer, hence there are no extreme highs and lows observed as small differentiations cannot gravely affect the mean values of energy needs, of a whole area.

- **More training data**: To train their models, the whole 4-year period from 2009 to 2012 except the last day of the year 2012 which has been used for testing. In our case study, the training data consisted of 18 months and the test dataset 6 months.

As a consequence of the above, the min/max range of the produced evaluation metrics from related scientific work varies significantly. From the related studies that were analyzed we see the MAPE value ranging from around 1.5% to about 20%, although most studies are in the 5-15% range. Hence, a MAPE value in this range was our general objective in the experimental modeling process.

4 Methodology

The goal of this thesis is to forecast the energy load for a prototype modern smart building. The strategy for the above goal was to utilize and evaluate well-known, state-of-the-art, conceptually simple, yet diverse data mining/ML models and techniques.

For this overall project three main steps were followed:

- At first, the raw dataset is explored, merged, visualized, and preprocessed. This was implemented via the use of python 3.9 programming language via Visual Studio Code interactive development environment.
- Implementation and evaluation of machine learning algorithms, primarily regression and ANN models, for ELF utilizing python's 3.9 Scikit learn and Tensor flow libraries
- Hyperparameter tuning was utilized to improve the performance of the forecasting models. Moreover, cross-validation techniques and model ensembling methods were utilized in order to test the capability/performance of the developed forecasting models to predict new (unseen) data. Finally, the developed ensemble approach was validated with an alternative dataset, with similar properties.

•

4.1 Data preparation

4.1.1 Dataset description

The open-access dataset [73] includes measurements and raw data collected from and for the CERTH Smart House nanogrid infrastructure discussed in Section 2.3. Mainly energy-related aspects are included in the publicly available datasets utilized, covering Energy load, Generation, and Storage, as well as Weather data information from external online APIs, between October 2018 and September 2020, with 15-minute resolution. More precisely, the dataset categories utilized for our analysis are described below:

Table 2: Dataset fields used for our analysis

Fields	Description	Unit of Measurement
Timestamp	Time of the recording in UTC time zone	2018-10-03T21:00:00Z
Energy Consumption	Total energy consumed from the entire building in a quarter	kWh
Photovoltaic Energy Generation	Energy Produced from an installed photovoltaic system in a quarter	kWh
Temperature_v1	Temperature from Online Weather API	°C
Relative_Humidity_v1	Relative Humidity from Online Weather API	%
Wind_Speed_v1	Wind Speed from Online Weather API	km/h
Clouds_v1	Cloud Coverage from Online Weather API	%

Other available data are excluded primarily due to a very high number of null values and/or irrelevance of the research goal.

4.1.2 Pre-processing

Single dataset construction

The dataset was provided in the form of .csv files and the very useful Pandas library in Python 3.9 was utilized in order to conduct the pre-processing. This is because the data are in their raw format as extracted from the physical installation including missing values, outliers, etc.

The first step was to combine the available energy load, and energy generation data from PV of the ITI Smart House with the weather/ambient data described in Section 4.1.1, as

these were contained in separate .csv files. Since the temporal resolution is the same for all data sets (i.e. 15 mins), the merging process was implanted with the use of the timestamps (e.g. 2018-10-10 15:00:00), to construct a single data frame containing all values (Figure 12)

new_Timestamp_UTC	Energy_Consumption	Energy_Generation	Temperature_v1	Relative_Humidity_v1	Wind_Speed_v1	Clouds_v1
2018-10-03 21:00:00	0.8	0.0	20.6	74.0	3.60	25.0
2018-10-03 21:15:00	0.7	0.0	20.6	74.0	3.60	0.0
2018-10-03 21:30:00	0.7	0.0	20.6	74.0	3.60	25.0
2018-10-03 21:45:00	0.7	0.0	20.6	70.0	3.60	25.0
2018-10-03 22:00:00	0.6	0.0	22.2	32.0	5.40	25.0
...	...	...	...	...	...	...
2020-09-30 19:45:00	0.5	0.0	18.9	48.0	1.34	100.0
2020-09-30 20:00:00	0.5	0.0	18.9	48.0	1.34	100.0
2020-09-30 20:15:00	0.5	0.0	18.3	50.0	0.45	0.0
2020-09-30 20:30:00	0.5	0.0	18.3	50.0	0.45	0.0
2020-09-30 20:45:00	0.5	0.0	18.3	50.0	0.45	0.0

63613 rows × 7 columns

Figure 15: Instance of the single merged data frame (15-min resolution)

Missing values and resampling

Our dataset contained a great number of missing values, in many cases for prolonged periods. This sort of irregularity in the data can have a significant impact on the accuracy of a forecasting model, hence data pretreatment procedures should be used prior to the building of forecasting models. Hence, the next step was to deal with missing values in the time series by assessing the best strategy that should be followed. The first exploratory attempt was to create and then fill the missing 15min intervals with the forward linear interpolation. As a result of this filling process, admittedly it is very challenging to have a genuine-looking pattern, especially in large periods of absent data/metering, as indicated in Figure 16.

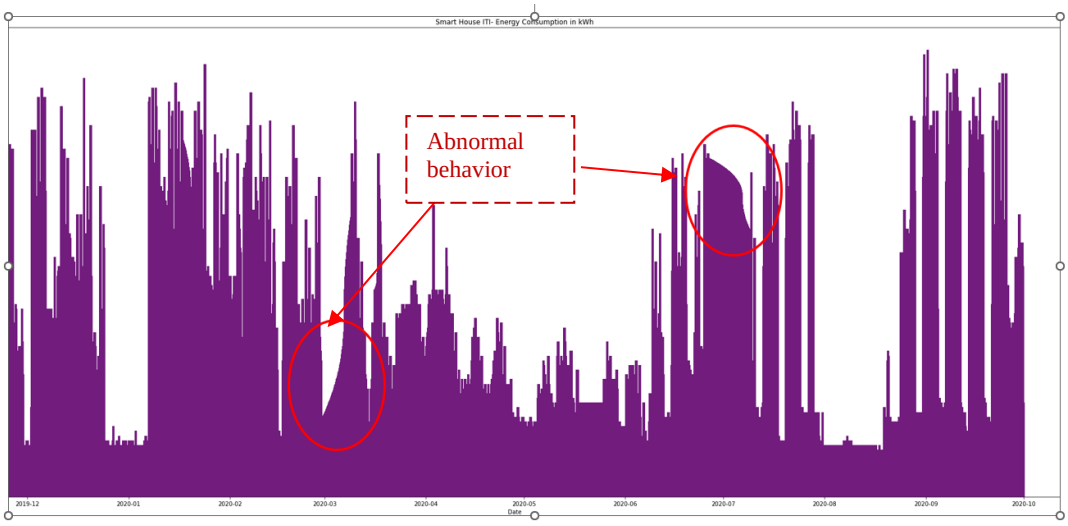


Figure 16: Instance of 1st attempt of interpolating missing values (Energy load)

In order to tackle the above problem, first we proceeded to resample the 15min time series to a 1h resolution. Using the sum for energy-related values (Energy load and PV Generation) and the average for weather recordings (i.e. Temperature), as this provides a better forecasting baseline for modelling. This is especially true when we considered the extensive literature review, as many studies had an input resolution of 1h for STLF. However, very long periods of missing values still existed in the resampled dataset (e.g. 15 consecutive days) which is not ideal. Hence, due to the many missing values, a hybrid approach was chosen using a forward and backward time-based interpolation method, for time intervals of up to 4h and to omit missing data extending to larger time periods than 4h. This procedure resulted in producing a full dataset for both target (Energy load) and exogenous features (Temperature, PV generation) and resulted in a far more 'normal' behavior and daily pattern, capturing both periodicity and trend, as it will be shown in Section 4.1.3.

Outlier detection

Identifying and removing outliers is a vital component in creating robust forecasting models, which avoid large errors. In particular, the analysis of outliers in time series data examines anomalous behaviors across time. For example, in our dataset there or illogical outliers such as temperatures well above 45°C (Figure 17).

```
[15]: df_cons_we.loc[df_cons_we["Temperature_v1"] >45  ]
```

[15]:

	new_Timestamp_UTC	Energy_Consumption	Temperature_v1	Relative_Humidity_v1	Wind_Speed_v1	Clouds_v1
19760	2019-05-05 01:45:00	0.3	70.0	93.0	7.200000	45.0
20049	2019-05-09 04:00:00	0.3	64.0	96.0	7.200000	100.0
20060	2019-05-09 06:45:00	0.8	73.4	78.0	16.919998	75.0
20138	2019-05-10 02:15:00	0.4	50.0	78.0	25.199999	100.0
20269	2019-05-11 10:45:00	0.5	67.0	94.0	12.240000	0.0

Figure 17: Identification of outliers in temperature

Hence, an outlier detection strategy was followed, which identifies values that were above the 99.99th and the 99.8th percentiles for all the temperature and energy load data points and replaces them with their equivalent max values that are equal to the percentiles (Figure 18).

```python
UpperOutlierPerc=99.99
upper_cons = np.percentile(df_cons_we['Temperature_v1'], UpperOutlierPerc)
df_cons_we['Temperature_v1'] = df_cons_we['Temperature_v1'].apply(lambda x : upper_cons if x > upper_cons else x)
```

```python
UpperOutlierPerc=99.8
upper_cons = np.percentile(df_cons_we['Energy_Consumption'], UpperOutlierPerc); upper_cons
df_cons_we['Energy_Consumption'] = df_cons_we['Energy_Consumption'].apply(lambda x : upper_cons if x > upper_cons else x)

```

Figure 18: Code snippets of the outlier detection and removal strategy

4.1.3 Exploratory Data Analysis

Before we conduct our modelling, it is important to understand the kind of data that we are dealing with and their characteristics and key trends. To that end in this section provides an exploratory data analysis (EDA), by presenting certain key visualizations along with supplementary comments, to provide a holistic view of our time series datasets. Below, Figure 19, shows the Smart House hourly Energy load in kWh, after extreme outliers were removed.

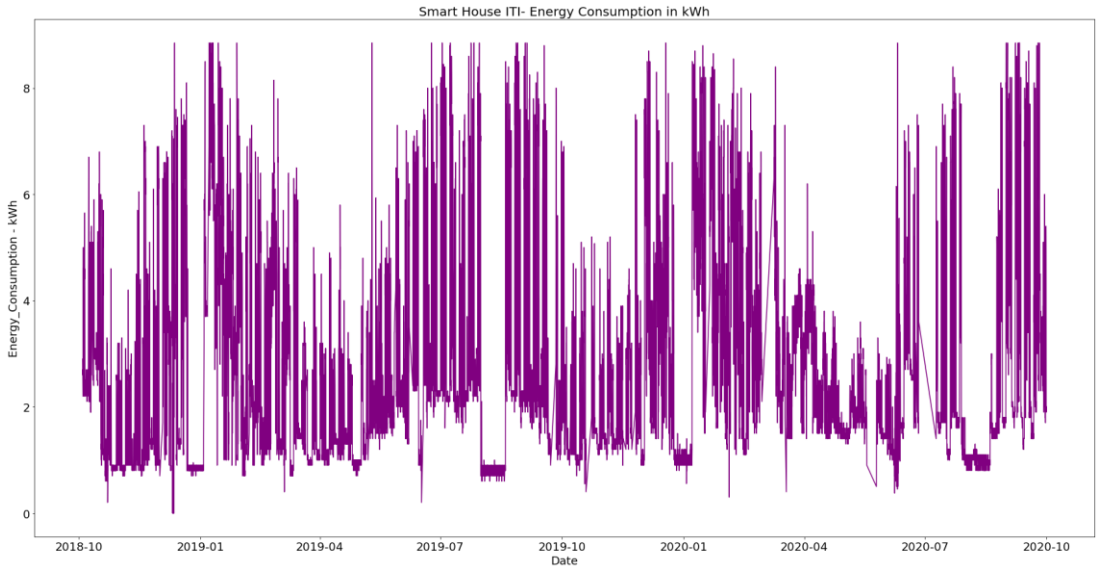


Figure 19: Smart House Energy load per hour in kWh (Oct. 2018-Sep. 2020)

Some notable patterns and a clear periodicity can be clearly observed. At first, we notice that during the winter and summer months the average hourly energy load is higher, which is to be expected due to the increased cooling and heating demands. Furthermore, since the Smart house is part of the ITI infrastructure, it acts as office space for researchers, thus it evident that there are periods in December and August that the energy load drops due to the leave of the bulk of ITI employees for Christmas and summer vacation accordingly.

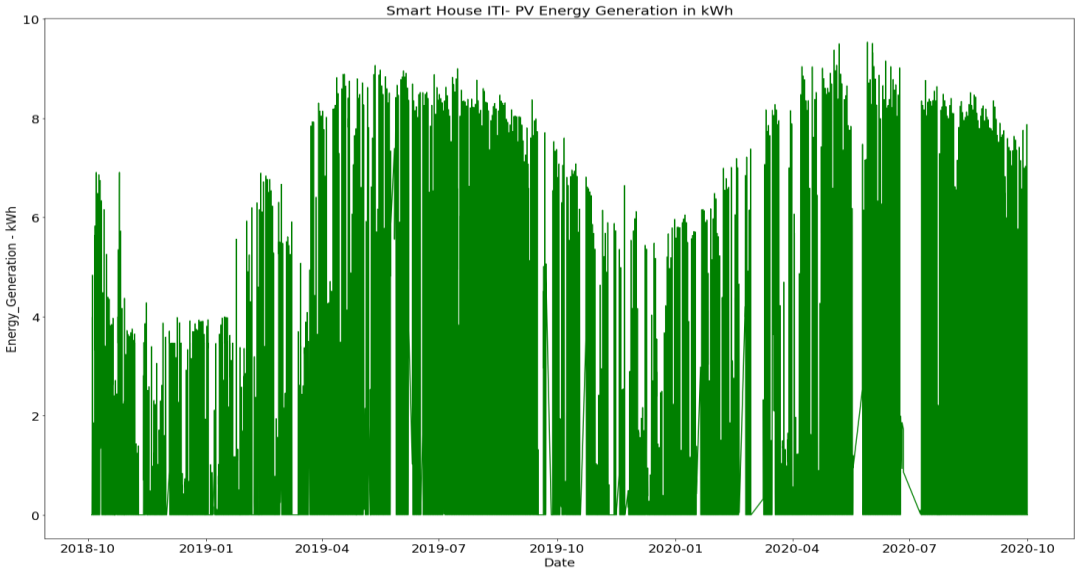


Figure 20: Smart House's PV Energy generation per hour in kWh (Oct. 2018-Sep. 2020)

Similarly, for electrical energy generated by the installed PV on the roof (Figure 20), it is evident that during the spring and summer months (April to September) there is increased energy generation due to the prolonged hours of sunshine. All energy load, generation, and exogenous weather parameters can be visualized in more detailed in Appendix A: Dataset Samples. Figure 21, illustrates the hourly resolution of both energy load and generation on a weekly basis. We notice here that the first day of the week (Monday) shows higher energy load, while naturally the energy load is minimized during the weekend, as there is no employee activity.

Figure 21: Energy load and PV generation in a week (Mon.-Sun.)

Regarding a characteristic daily profile (Figure 22), as expected both energy load and PV generation spike during working hours, peaking around noon.

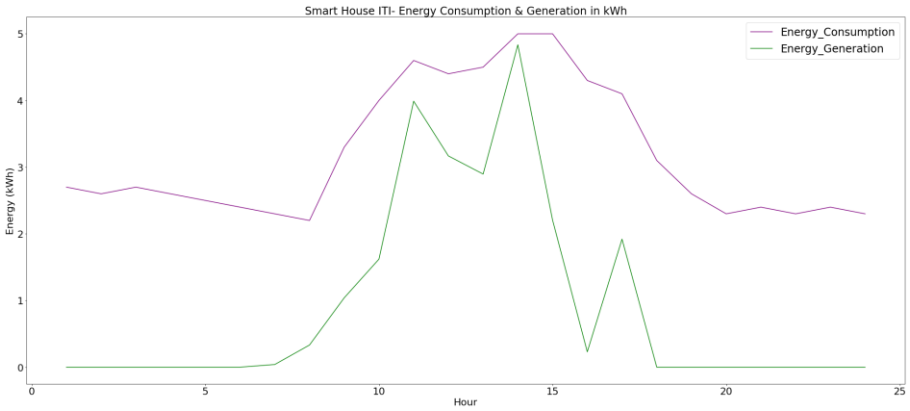


Figure 22: Energy load and PV generation in a typical day (24h)

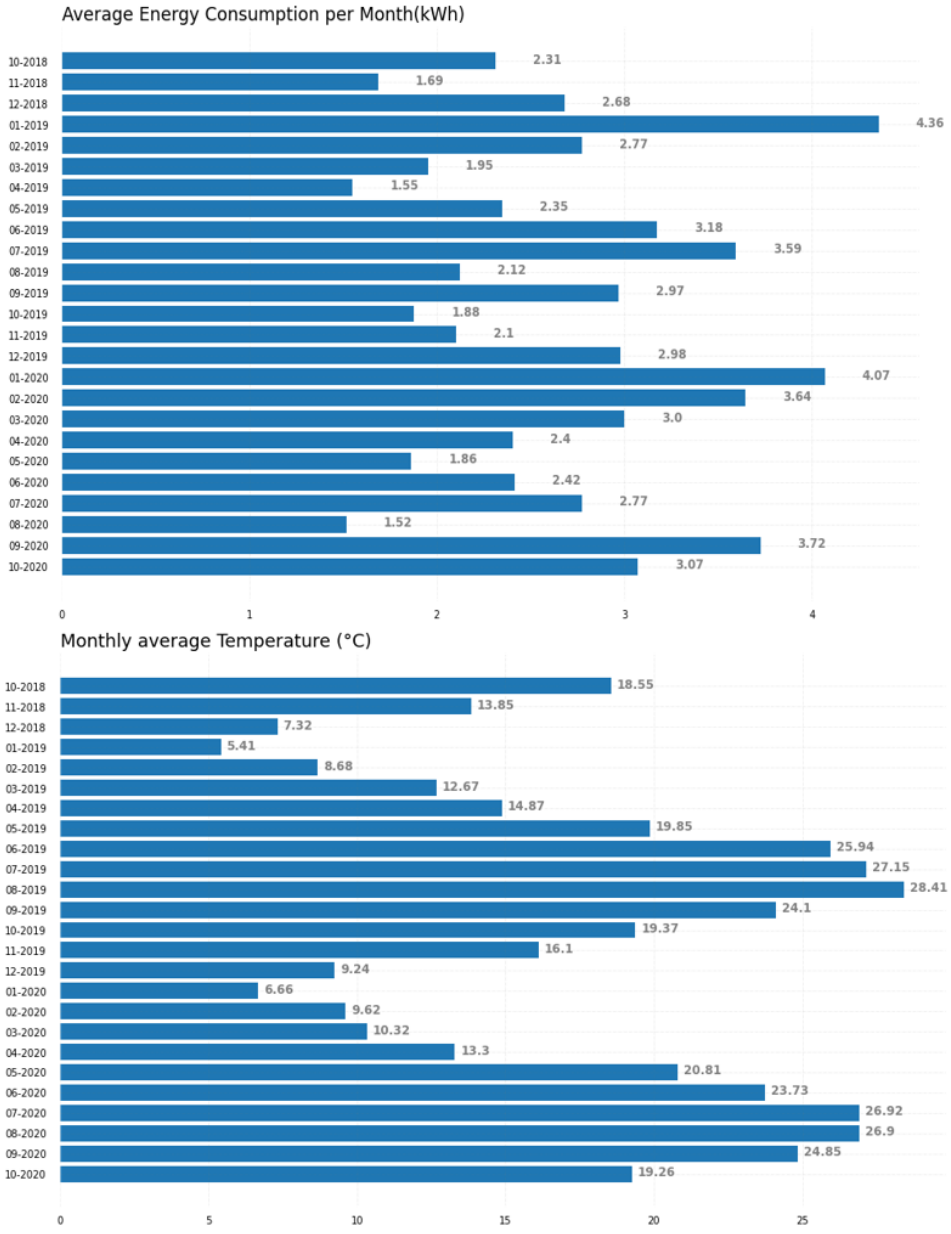
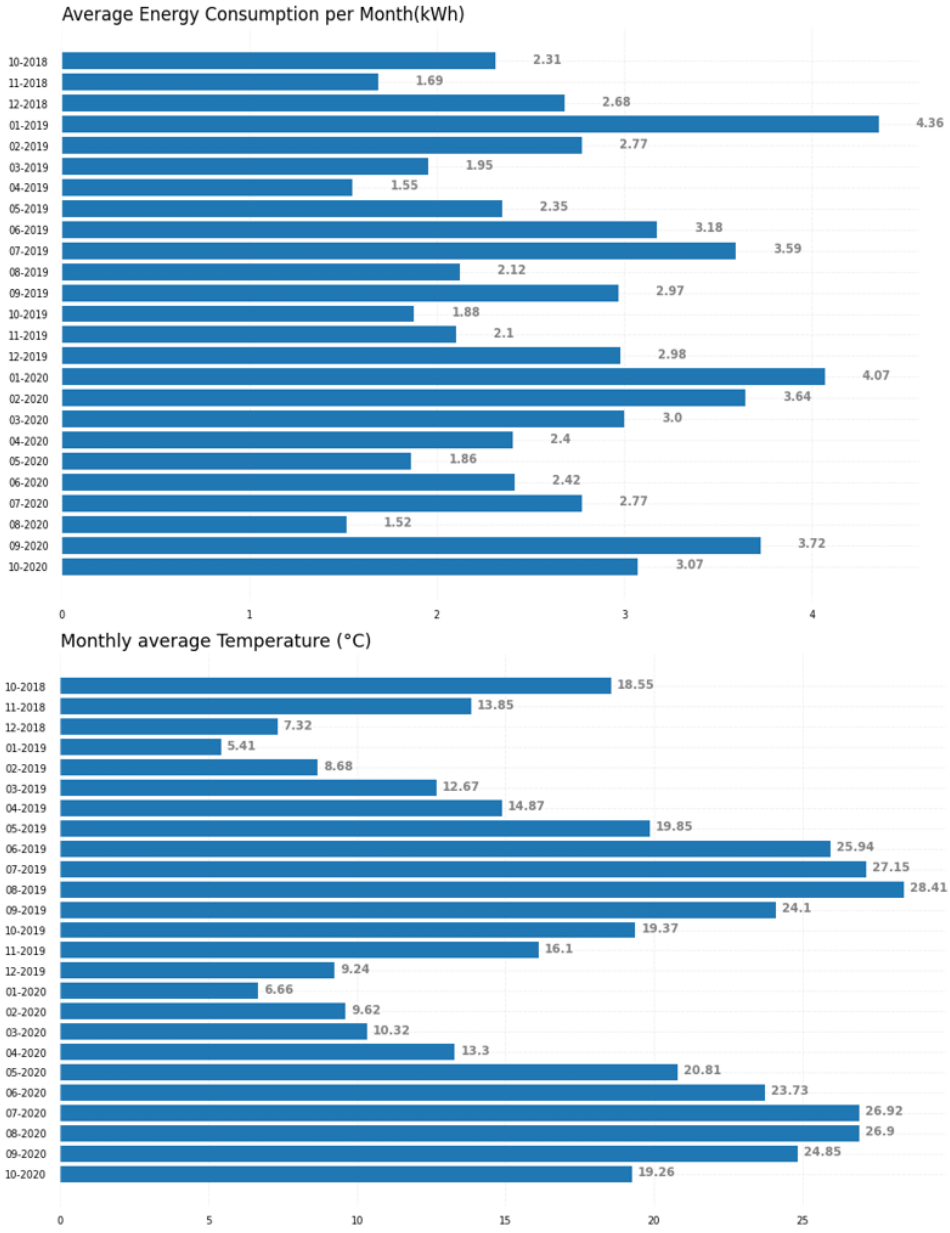


Figure 23: Monthly average Energy load and Temperature



From Figure 23, a clear pattern stands out which shows that the periods of high energy load coincide with the periods where the lower and higher temperatures are recorded. The month of August should be disregarded, as the ITI Smart House remains closed for a large period of time. However, no clear patterns or trends from the other weather/ambient parameters (i.e. wind speed, relative humidity, cloud coverage) in relation to the Smart house's energy load can be singled out, these parameters do not seem to have a high correlation with our target variable, the energy load, and do not improve the forecasting performance (Figure 24).

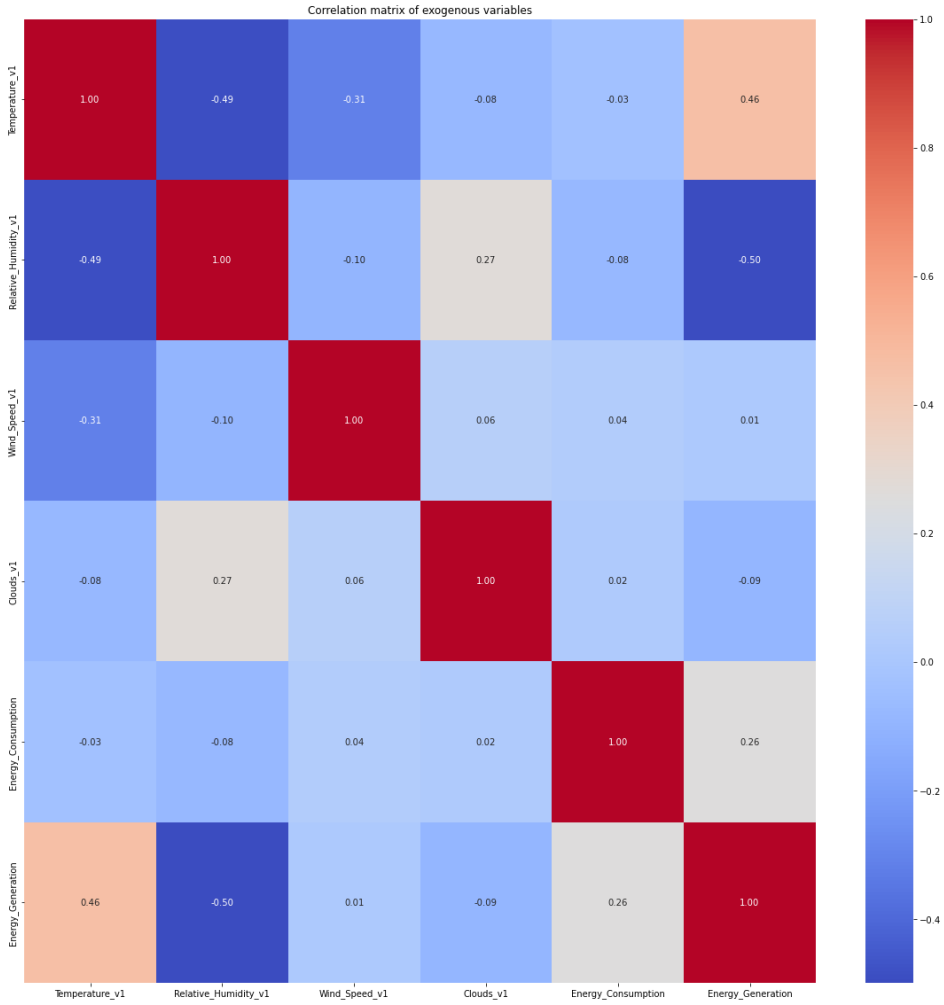


Figure 24: Correlation matrix of the exogenous parameters and the target variable

Time series stationarity

The basic assumption of a traditional regression model (e.g. linear regression) is that the observations are independent does not hold in this case. Due to the temporal dependencies in time series data, forecasting cannot rely on simple forecasting techniques. In fact, the autocorrelation plot(lags=500) of the ITI Smart House's energy consumption shows that the time-series is non-stationary (Figure 25)

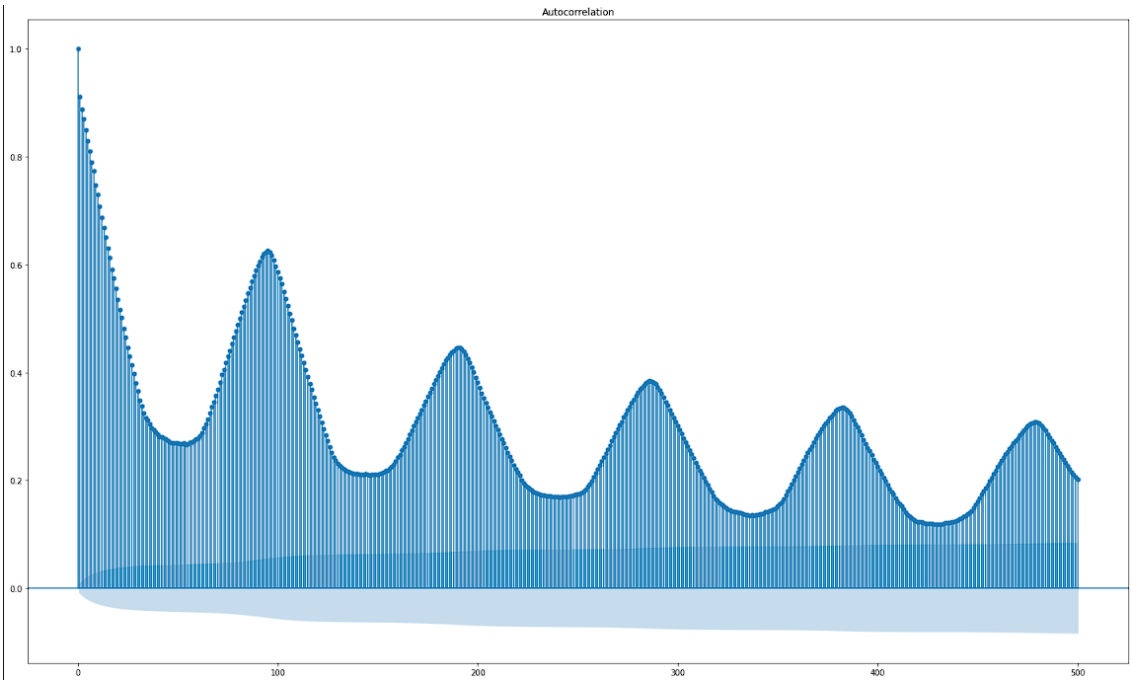


Figure 25: Autocorrelation plot for Energy consumption (entire dataset)

This shows irregular dynamics which traditional forecasting techniques are less suited for and motivates us to use supervised ML models, as such models are suitable to handle non-stationary time-series. Additionally, for a multivariate time series analysis as in our case, adds more complexity, which can be better dealt by supervised ML algorithms. However, necessary transformations are required in order to restructure the data to be used in a supervised learning problem, as presented in Section 4.2.1.

4.2 Data transformation

4.2.1 Feature engineering and selection

From empirical observations, energy load is influenced by several exogenous factors, such as the day of the week, holidays, or the month of the year, holidays, special occurrences, and weather effects, and therefore these factors must be considered in the training of the prediction model, in addition to historical load data[74]. In particular, ambient temperature is a factor that directly affects the energy load (cooling and heating of the smart home). Furthermore, in our case study, PV generation is also a helpful feature as high PV power generation indicates sunny weather, thus a lower need for heating in the winter period or prolonged sunlight periods in the summer period, which might translate to increased cooling needs. Both these exogenous features are used in addition to the historical data of the load to make a more accurate forecast.

Time series transformation to supervised

As it will be shown in Section 5, the forecasting models used, come within the category of supervised ML algorithms, which implies that they must be trained given a collection of feature vectors and their corresponding target values. To do this, we created a collection of hybrid feature vectors that include both automatically picked values from the energy load time series and manually selected temporal characteristics.

The sliding window technique was used for feature extraction, which is a standard approach for time series forecasting modelling. This entails transforming the data into lagged observations. That is, n prior values (t-n, t-n+1,...,t) were utilized to forecast the future value (t+1). This is accomplished by iteratively traversing the data, accepting n items as input and the next entry as output. In our case, a 3D sliding window (variables, rows, and time steps equal to 1) was utilized for the LSTM, while a 2D sliding window (time steps with variables and rows) was used for the remaining algorithms (tree-based).

As time-series data often exhibit trends or seasonal patterns, the relation between the model's input variables (independent variable) and the predicted value (output variable) fluctuates with time. In order to discover the datapoints where the values of independent variables are substantially related, the time-lag value must be accurately chosen. The correct estimation of the number of previous observations is a critical factor in time-series forecasting (lags).

After reviewing relevant literature for critical time-lagged features created (e.g. [71]), extensive experimental modelling and rating feature importance, the final selection of features, to forecast 1-step ahead (i.e. 1 hour ahead) included:

1. 24 lagged features (T-1,T-2,T-3,…T-24) for Energy_Consumption (Target). In essence, this is the energy load value of the day preceding the hour for which the prediction is made,
2. Similarly, 12 lagged features (T-1,T-2,T-3,…T-12) from 2 additional exogenous features the hourly PV energy generation (Energy_Generation) and the hourly ambient temperature (Temperature_v1),
3. The energy load value 48h before (T-48Cons)
4. The energy load value 72h before (T-72Cons)
5. The absolute standard deviation (std) between the previous hour energy load and the previous 24h energy load. The calculation formula is presented below (Equation 1)

$$\left|\frac{\mathrm{Cons}_{t-1}+\mathrm{Cons}_{t-24}}{2}-\mathrm{Cons}_{t-1}\right| \quad (1)$$

Naturally, as our time series dataset also implements the notion of time as a feature, the timestamp was split into categorical values, to create additional temporal features, as shown in (Table 3). This is a key difference between regular regression problems and time series problems, as it captures temporal characteristics such as periodicity and trends of the time series. Apart from the typically used temporal features such as hour of day, the day of the week, the month of each load data point, some specific features were synthesized based on the characteristics of our dataset, acknowledging that the ITI Smart House is used as a workspace, such as a distinction feature of office hours (i.e. 9:00-18:00) and non-peak hours (i.e. after 18:00). Other customized temporal features were also considered, such as importing Greek National holidays and time periods where ITI is closed for vacation (i.e. Christmas, 1-15 August), however, they were not used, as it would hamper the ability to create a more generic model and its applicability.

Table 3: Temporal features created

Features	**Description**
quarter	Dummy variable corresponding to the 3-month quarter of the year
month	Dummy variable corresponding to the month of the year
hour24	Dummy variable corresponding to the hour of the day
Week_day	Dummy variable corresponding to the day of the week
is_weekend	Dummy variable corresponding to Saturday & Sunday
off_hours	Dummy variable distinguishing off-peak hours (i.e. after 18:00)
working_hours	Dummy variable distinguishing peak hours (i.e. 9:00-18:00)

Since this is a time series forecasting problem, we preserved the order of time while splitting the data (utilizing a temporal train test split). Hence the first 80% of historical data are used for training and the remaining 20% of data points are used for testing the predictions of the models. For the entire dataset, this translates to roughly 18 months for the training of the ML algorithms and the remaining 6 months for testing. However, then smaller portions of the dataset were chosen for training and testing the algorithms.

4.3 Evaluation metrics

The forecasts that were produced by the regression-based algorithms' implementation were contrasted to the data set's testing segment and validated using key metrics. The predicted values in regression-based problems are continuous numbers and the primary principle behind evaluating regressor performance is to calculate the difference between the true and forecasted values. To that end, the performance was calculated and compared using five evaluation metrics: Mean Absolute Error (MAE), Root Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE), R-squared, and the elapsed training time was also recorded, which are described below:

1. Mean Absolute Error (MAE)

MAE calculates the mean of the non-negative differences between observed and predicted values. All differences have equal weight, as a result MAE fails to punish large errors in prediction. MAE calculates the error is on the same scale as the target, hence it is more interpretable. MAE is defined by the formula (Equation 2):

$$MAE = \sqrt{\frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n}} \qquad (2)$$

2. Root Mean Square Error (RMSE)

RMSE calculates the square root of the average squared difference of the observed and predicted values. This metric is able of identifying and heavily penalizing large errors and evaluate the fluctuation of model response in terms of variance. The mathematical formula for RMSE is (Equation 3):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}} \qquad (3)$$

3. Mean Absolute Percentage Error (MAPE)

The MAPE calculates the absolute value of the difference between observed and predicted. Typically, it quantifies accuracy in percentage terms and is effective for evaluating the performance of the prediction model by applying more weight on positive errors in relation with the negative ones. The MAPE is defined by the formula (Equation 4):

$$MAPE = \frac{1}{n} \times \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \times 100\% \qquad (4)$$

4. Coefficient of determination or R-squared (R^2):

R-squared is often referred as the goodness of fit (of a regression model), as it determines the proportion of the variance in the dependent variable that is predictable from the independent variables. A high R-squared value indicates the predicted values fit the observed values quite well. However, a high r-squared is not necessarily always a good indicator for the regression model. Many factors influence the statistical measure's quality, including the type of the variables included in the model, the units of measurement of the variables, and the data transformation used. The R-squared is defined as follows (Equation 5):

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \qquad (5)$$

5. Elapsed Time (ET)

Although ET is not an evaluator of a model's precision, it is selected as it is a useful metric that records the models' training times, in an effort to highlight the less computationally intensive model.

5 Experimental modelling

5.1 Entire dataset forecasting

The first step in training and testing our forecasting models, was to utilize the entire dataset in a temporal manner (first 18 months for training, 6 remaining months for testing), as this would enable the training process to use the highest possible number of data points. This is important, in particular for the Neural Networks implemented, especially the LSTM, which literature indicates that they maximize their performance when fed with quite larger training datasets.

Also, after some excessive modelling trials and experimenting with the effect of exogenous features on improving the performance of the models, it turns out that the rest of the exogenous weather/ambient data (Rela-tive_Humidity_v1,Wind_Speed_v1, Clouds_v1) have a poor correlation with our target (Energy_Consumption). This was evident in the relevant feature importance graphs that were produced. In other words, they had no significant effect in improving our forecasting accuracy and thus were not selected in the final feature matrix. Hence, after the data preprocessing, we finalize the training dataset used to feed the models. The final feature space includes 12721 rows × 59 columns, including both temporal and lagged features of the three separate time series (Energy load, PV generation, Temperature). An indicative snippet is depicted in Figure 26.

T-16Cons	T-15Cons	...	quarter	month	day_of_month	hour24	is_weekend	is_holiday	working_hours	off_hours	ITI_closed	Week_day
2.2	3.3	...	4	10	4	21	0	0	0	1	0	3
3.3	4.0	...	4	10	4	22	0	0	0	1	0	3
4.0	4.6	...	4	10	4	23	0	0	0	1	0	3
4.6	4.4	...	4	10	5	0	0	0	0	1	0	4
4.4	4.5	...	4	10	5	1	0	0	0	1	0	4
...	...	...	...	...	...	...	...	...	...	...	...	...
1.4	1.5	...	2	4	29	22	0	0	0	1	0	2
1.5	1.6	...	2	4	29	23	0	0	0	1	0	2
1.6	1.7	...	2	4	30	0	0	0	0	1	0	3
1.7	1.4	...	2	4	30	1	0	0	0	1	0	3
1.4	1.5	...	2	4	30	2	0	0	0	1	0	3

Figure 26: Snippet of the training dataset and features

Initially, the models were trained with their default parameters, without any finetuning, then a hyperparameter tuning process was used in a second round of training, in order to optimize the performance of models. In predictive analytics, hyperparameter optimization is an integral part of the modelling process. In general, models can perform well with the default parameters however, tuning can produce more accurate results. To that end, we attempted to identify the proper parameter grids for our selected modeling algorithms, by reviewing the relevant documentation for each algorithm as well as the bibliography. For this process a popular python library was utilized, the Grid Search Cross-Validation (GSCV) method. It finds the best combination of hyper-parameters that give optimal results for the model performance. For example, during the model training process, GSCV creates multiple models, each with a unique combination of hyper-parameters. The goal of GSCV is to train each of these models and evaluate their performance using cross-validation. Ultimately, the model that gives the best results is selected.

Furthermore, in the development of a supervised ML model, it is important to understand which features are most associated with the prediction outcome. In essence, they rank the input characteristics for a particular model, based on their contribution to predicting the target variable. Feature importance is frequently used for dimensionality reduction, which means we can use it as a filter method to remove extraneous features from our model and only keep those that are most strongly related to our desired outcome. The goal in ML is to achieve the maximum possible performance of a model, with the minimum number of features, as more features translate to increased training times and computational power. To that end feature importance scores were produced, which helped us with deciding which features (especially lagged features) to select for our final training dataset. This was implemented via Sklearn's embedded library (feature_importance_), which for regression problems, calculates the variance reduction based on the Mean Square Error. The steps of our overall modelling approach are summarized in Figure 27.

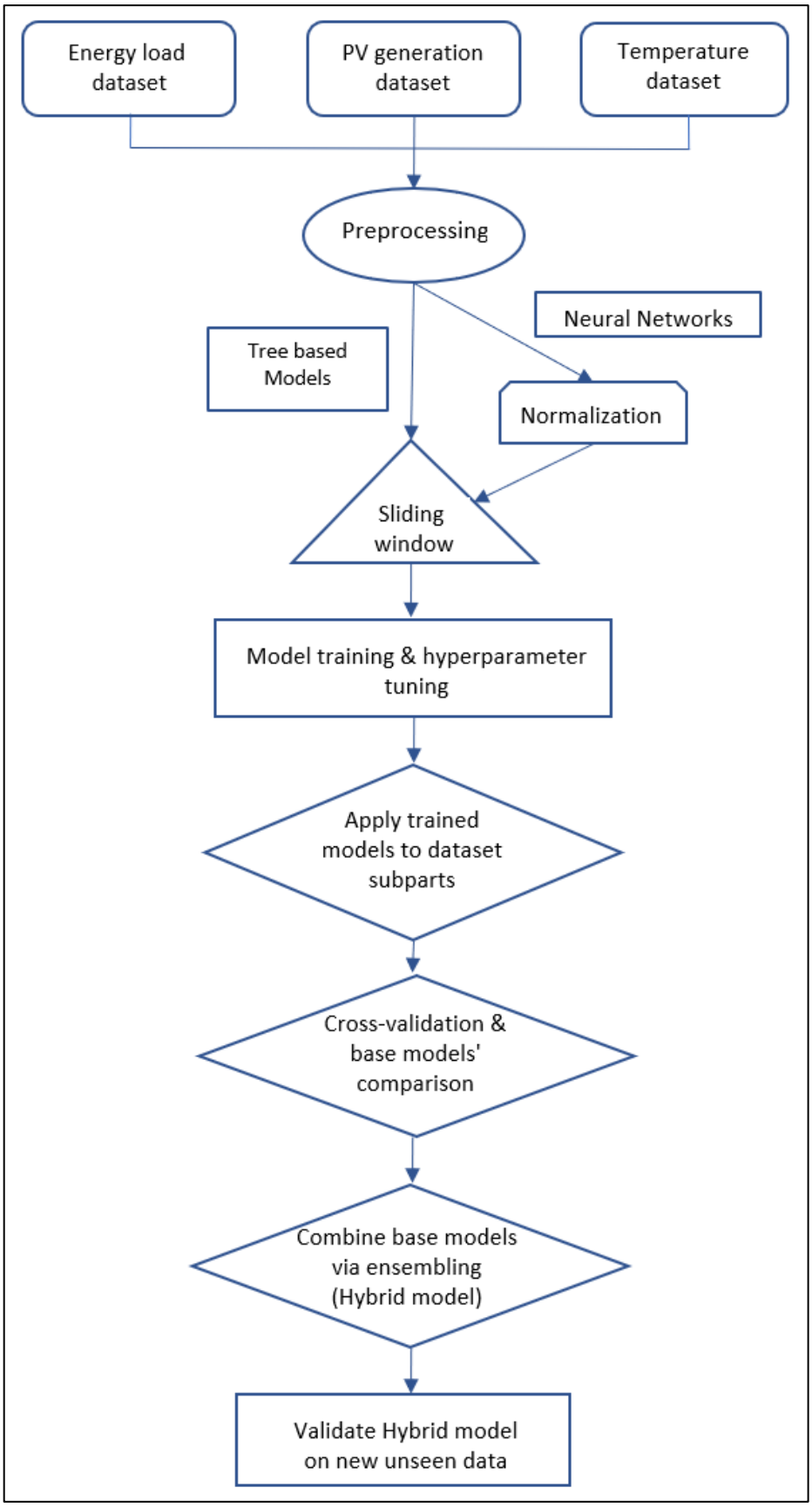


Figure 27: Flow chart of the overall modelling approach

5.1.1 Single algorithm selection & tuning

Extreme Gradient Boosting (XGBoost)

The first model utilized in our case study was the Extreme Gradient Boosting (XGB) Regressor. In our first attempt, the baseline model was implemented without any parameter tuning did not occur. For the XGBoost model, Figure 28, illustrates the feature importance scoring.

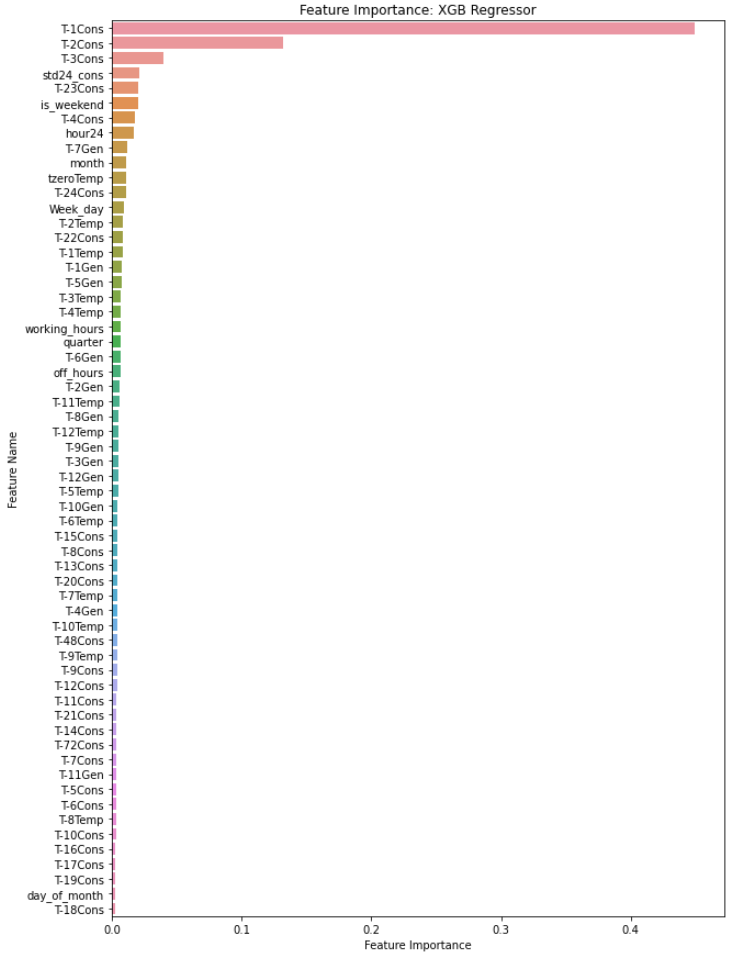


Figure 28: Feature importance (XGBoost)

Evidently, the most important feature is by far the T-1 values of our target, the energy load, followed by the T-2, T-3 values. This is quite anticipated in a time series dataset such as the energy load as in an hourly resolution a building does not frequently have 'jumps' in energy load for 2 or 3 hours backward. Such abrupt changes in energy load would mean that a sudden event of energy need occurs, such as an electric vehicle or a storage battery charge. Interestingly the standard deviation (std) between the previous hour and the previous 24h energy load value and the dummy variable that determines if it is Saturday or Sunday, are also in the top ranks in terms of feature importance. This can be probably explained due to the fact as we mentioned earlier that the ITI smart house is

used as a workspace, thus the energy load pattern during the weekend shows a clear reduction, whatever the time of year and weather circumstances.

The next step was to finetune the XGBoost model and focused on tuning the number of trees ('n_estimators'), the maximum of a tree ('max_depth'), the learning rate i.e. the step size shrinkage used in update to prevent overfitting ('learning_rate'), the subsample ('subsample') which denotes the fraction of observations to be randomly sampled for each tree. Furthermore, we tuned the subsample ratio of columns when constructing each tree ('colsample_bytree') is the subsample ratio of columns when constructing each tree. Finally, the 'objective' defines how the loss function will be minimized

Table 4: Hyperparameters for XGBoost

Hyperparameters	**Initial grid values**	**Final selection**
n_estimators	500, 550, 600…..1450, 1500	850
max_depth	2,4,6,8,10	6
learning_rate	0.01,0.05,0.1	0.01
colsample_bytree	0.1, 0.3,0.7, 1	0.7
subsample	0.5, 1, 2	1
objective	reg:squarederror, reg:absoluteerror	reg:squarederror

Random Forest

The next model we implemented is the Random Forest. Again, initially, the baseline model was implemented without any hyperparameter tuning. For the Random Forest model, Figure 28, illustrates the feature importance scoring.

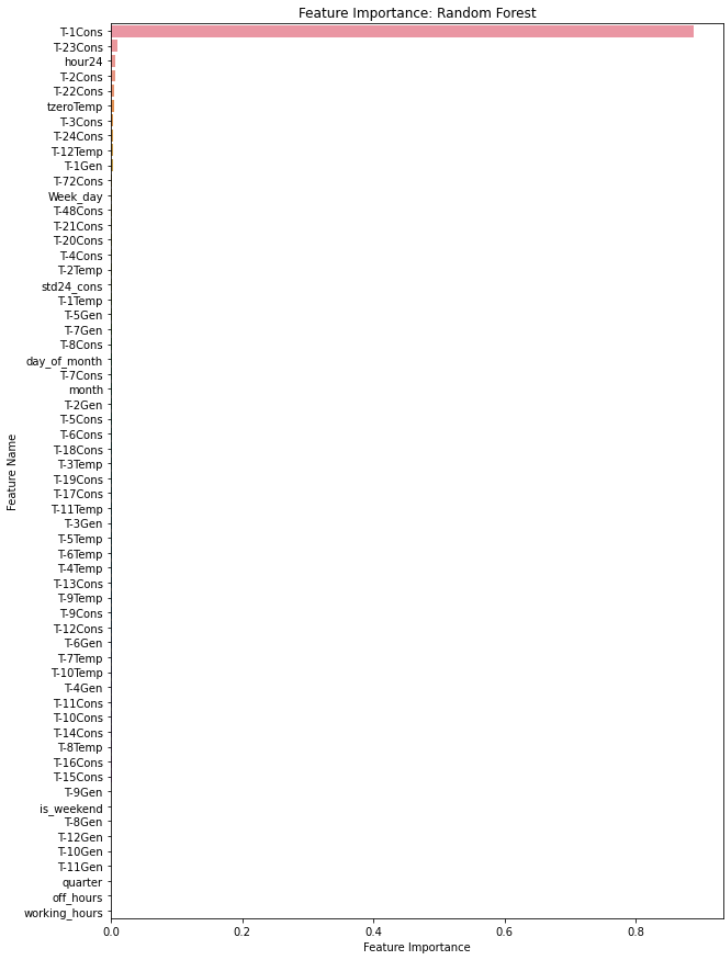


Figure 29: Feature importance (Random Forest)

Regarding feature importance we see a different picture in this case. The Random Forest algorithms relies heavily on the T-1 lagged energy load value.

Table 5: Hyperparameters for Random Forest

Hyperparame-ters	**Initial grid values**	**Final selection**
n_estimators	200, 300, 500, 800, 1000	500
max_depth	20,40,60,80	40
min_samples_leaf	2,4,6,8,10	6
min_sam-ples_split	2,4,6,8,10	6

For Random Forest, apart from the number of estimators and the maximum depth, we also focused on tuning the ‘min_samples_leaf’, which determine the minimum number of samples required to split an internal node and ‘min_samples_split’, which indicate the minimum number of samples required to be at a leaf node.

Light Gradient-Boosting Machine (LGBM)

We tried another notable ensemble tree-based regression algorithm the light gradient boosting machine (LGBM). Literature indicated that even a simple shallow learning model, like the LGBM has proved to provide accurate forecasts regarding residential energy load. Figure 30, represents the feature importance scoring for the LGBM model.

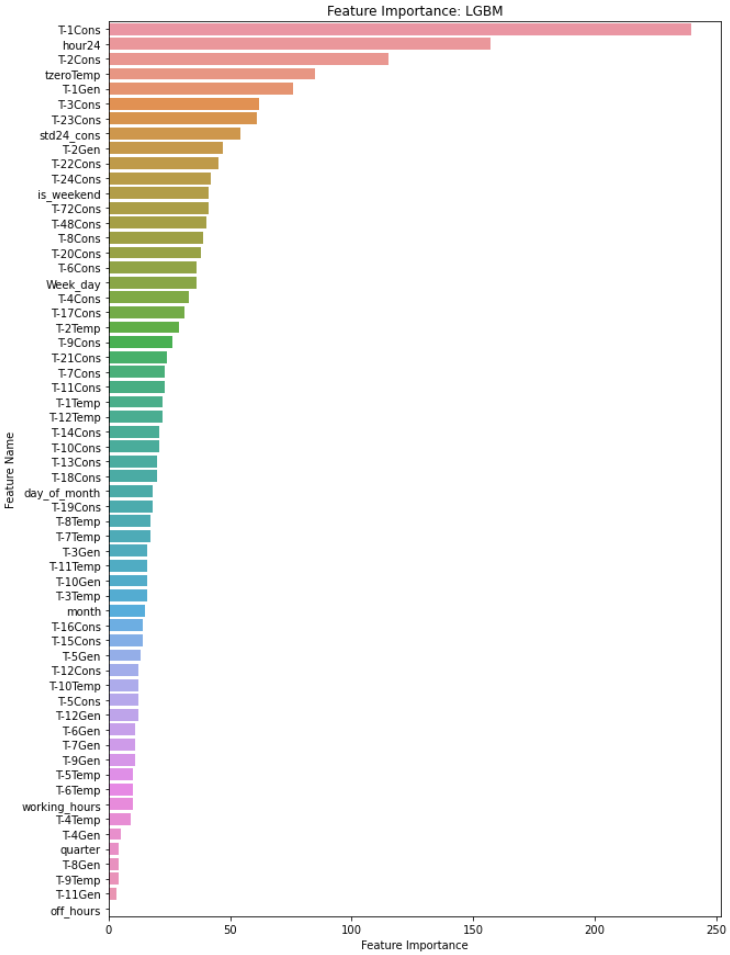


Figure 30: Feature importance (LGBM)

We notice that compared to XGBoost, LGBM has more features in the top ranks, in terms of contribution to its forecasting. Especially, the temperature and the hour of the day, and the previous hour energy generation from PV are more important for this model, compared to the other models. For the LGBM regressor, the number of leaves, the minimum data samples contained in a leaf and max number of bins ('max_bin') that feature values will be inserted in. Typically, smaller number of bins may reduce training accuracy but may increase general power (i.e. dealing with overfitting).

Table 6: Hyperparameters for LGBM

Hyperparame-ters	Initial grid values	Final selection
max_depth	4,8,12,16	10
num_leaves	6,10,14, 18, 22, 26, 30	20
min_data_in_leaf	40,60,80,100	70
max_bin	30,60,90,120	90

CatBoost

The next model implemented, CatBoost Regressor, is another member of the family of GBDT ensemble algorithms. Similarly, to the other GBDTs, the T-1 lagged energy load is by a large margin the most valuable feature, but also the day of the week is high up in the importance ranking.

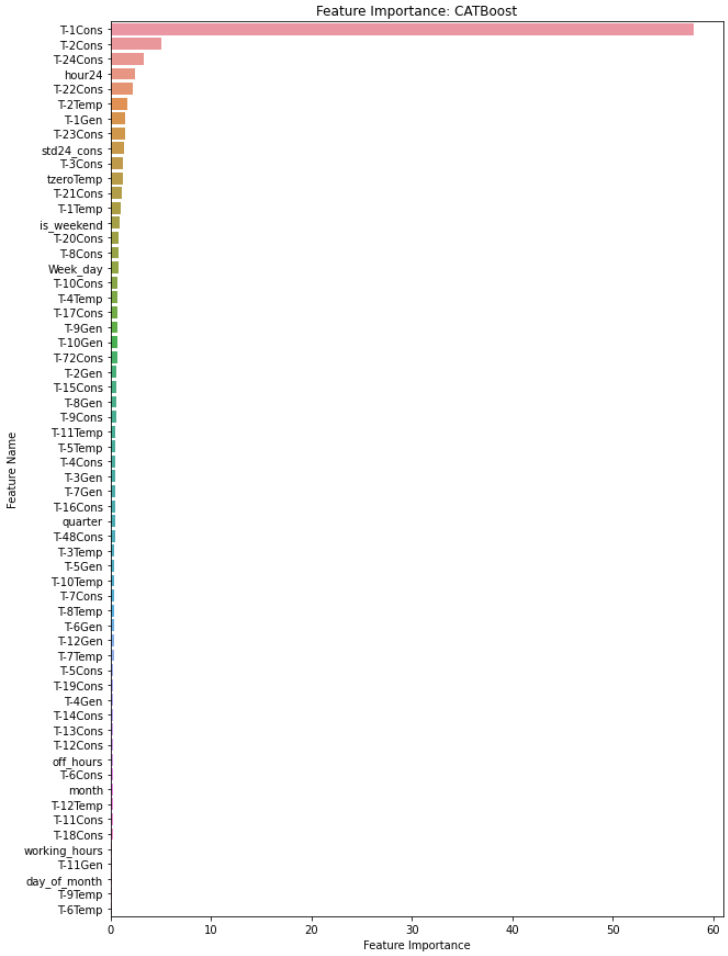


Figure 31: Feature importance (CatBoost)

For CatBoost regressor the parameters that were tuned were the number iterations, the learning rate which determines the reduction of the gradient descent step, the tree depth (‘depth’) and the L2 regularization parameter (‘l2_leaf_reg’), which deals overfitting by forcing weights to be small, but not making them exactly zero.

Table 7: Hyperparameters for CatBoost

Hyperparameters	Initial grid values	Final selection
iterations	200, 500, 850, 1000,1500,2000	850
learning_rate	0.01, 0.02, 0.05,0.1	0.02
depth	4, 6, 8, 10, 12	6
l2_leaf_reg	0.2, 0.4, 0.6, 1	0.6

MLP

The first ANN we implemented was the MLP regressor, which is trained trains via back-propagation with no activation function in the output layer, which can also be seen as using the identity function as activation function. Therefore, it uses the square error as the loss function, and the output is a set of continuous values. Initially we trained the algorithm with the default parameters and then we finetuned certain key hyperparameters (Table 8).

Table 8: Hyperparameters for MLP

Hyperparameters	Initial grid values	Final selection
hidden_layer_sizes	(50,100,50), (20, 60, 20), (100, 200, 100)	(50,100,50)
max_iter	500, 1000, 2000,3000	2000
learning_rate_ini	0.001, 0.01, 0.1	0.001
alpha	0.001, 0.005, 0.1	0.005
learning_rate	'constant', 'adaptive'	'adaptive'
optimizer	'adam', ‘lbfgs’	'lbfgs'
activation_function	'identity', 'relu'	'relu'

The hidden layer sizes define the number of neurons in each of hidden layers used. The initial learning rate controls the step-size in updating the weights, The activation function dictates the standard neuron activation function per layer. The optimizer specifies the gradient descent optimizer. The ‘alpha’ parameter represents the strength of the L2

regularization term, which decreases the likelihood of overfitting. The L2 regularization parameter is divided by the sample size when added to the loss.

LSTM

Finally, we implemented a state-of-the-art RNN the LSTM, which based on related literature findings, is heavily utilized for energy load-related forecasting, being able to better capture the volatility in energy load profiles of residential tenants. After careful examination and understanding of the characteristics, parameters, overall logic of this RNN, the first step was to normalize the input data. For this we used, the MinMaxScaler normalization method [75].We continued with trying out different layouts, regarding the structure of the LSTM model:

- Tried multiple hidden layers combinations (max number:5) which one's output is the other's input. After experimenting I found out only two hidden layers are optimal for my dataset and adding more had an adverse effect.
- Literature often mentioned that Bidirectional LSTM(B-LSTM) may behave better. The use of only 1 hidden layer of B-LSTM had a close performance with the 2 vanilla LSTM hidden layers but was slightly worse. Other combinations of B-LSTM and vanilla LSTM layers turned out to not be promising as well.
- One output layer was selected in order to have the output in the required format, using the linear method as the activation function.

After this trial and error exercise, we ended up with the final structure of the LSTM, consisting of two-hidden layers, with the first layer being the input layer and an output layer. A snippet of the model structure is illustrated in Figure 32.

```
def get_model(params, input_shape):
    model = Sequential()
    model.add(LSTM(units=(params["lstm_units"]),activation=params["activ_function"], return_sequences=True, input_shape=(X_train_LSTM.shape[1], X_train_LSTM.shape[2])
    model.add(Dropout(rate=params["dropout"]))

    model.add(LSTM(units=(params["lstm_units"]),activation=params["activ_function"], return_sequences=False))
    model.add(Dropout(rate=params["dropout"]))

    model.add(Dense(y_train_LSTM.shape[1],activation='linear'))

    model.compile(loss=params["loss"],
                  optimizer=params["optimizer"],
                  metrics=[MeanAbsoluteError(),RootMeanSquaredError()])

    return model
```

Figure 32: Structure of the LSTM model

The optimum hyperparameters for the LSTM model were then discovered using a rather time-consuming random search (Table 9). The results have shown that the ideal number of units is 128, epochs are set to 40, the batch size is equal to 128 and a 'Root Mean Squared

Propagation' (RMSProp) optimizer. RMSprop is a gradient-based optimization technique, which deals with the vanishing or exploding gradient of very complex functions like ANNs, by using a moving average of squared gradients to normalize the gradient. A very important parameter is the dropout, which acts as a regularizer for LSTM to avoid potential overfitting.

Table 9: Hyperparameters for LSTM

Hyperparameters	**Initial values**	**Final selection**
activation_function	tanh, relu	relu
loss	mse, mape,mae	mae
optimizer	adam, rmsprop, sgd	RMSprop
dropout	0.1,0.2,0.3	0.2
units	64,128,256.512	128
epochs	10,20,40,50,80	40
batch_size	64,128,256,512	128

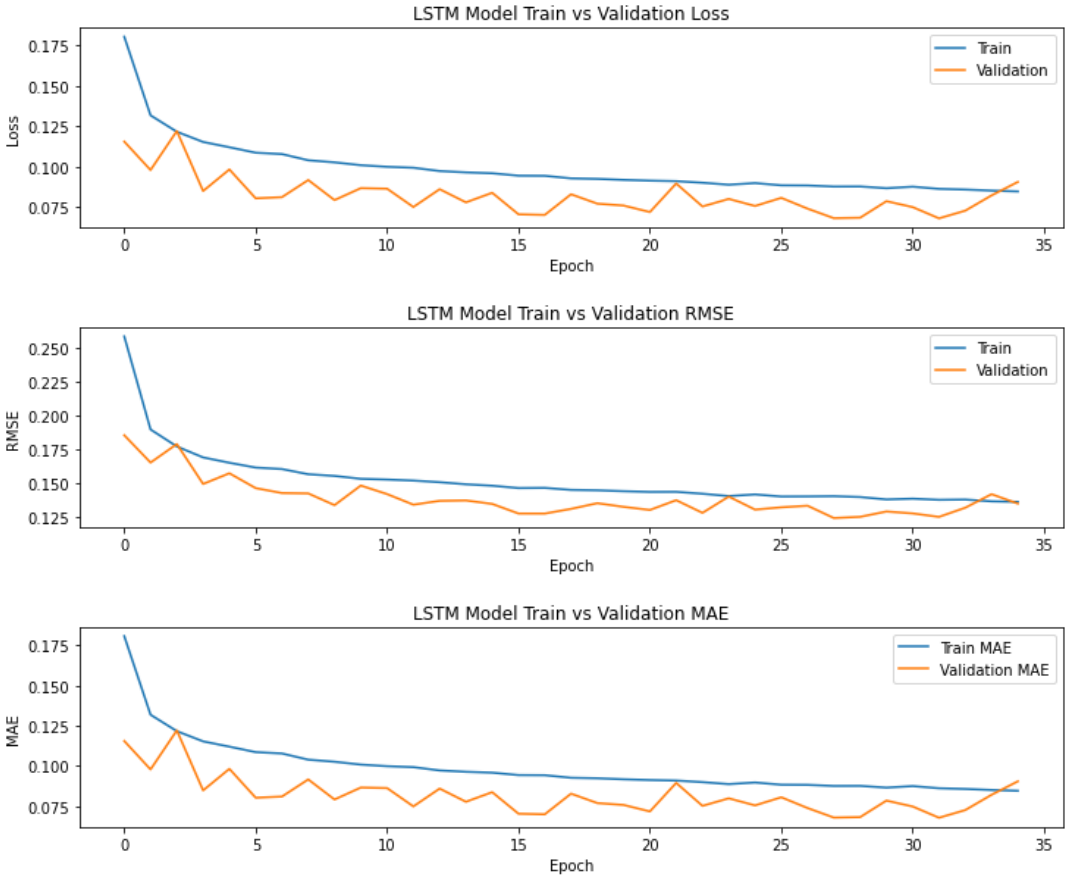


Figure 33: LSTM Training vs Validation Error, RMSE, MAE

5.1.2 Results

By observing the final selections for all algorithms, we can easily detect certain similarities among them. More precisely, all GBDT algorithms rely primarily in the T-1 lagged feature of energy load to make their predictions. Furthermore, all models, including the ANNs required the fine tuning of a regularization parameter to deal with excessive overfitting. The performance was again evaluated by the following evaluation metrics, MAPE, MAE, RMSE, R-squared and ET, as shown in Table 10.

Table 10: Evaluation metrics for the models applied to the entire dataset

Evaluation metrics	MAPE (%)	MAE (kWh)	RMSE (kWh)	R-squared (%)	ET(seconds)
Entire Dataset	**(Oct.2018 - Sep. 2020)**				
Single models w/o hypertuning					
XGBoost	12.957	0.298	0.551	90.89	0.747
Random Forest	13.36	0.302	0.547	91.02	24.862
LGBM	11.516	0.268	0.508	92.236	0.179
CatBoost	11.937	0.272	0.503	92.41	4.634
MLP	17.816	0.376	0.584	89.754	7.503
LSTM	24.927	0.5	0.657	87.005	14.032
Single models w/ hypertuning					
XGBoost	11.512	0.275	0.521	91.845	4.639
Random Forest	11.568	0.274	0.53	91.56	11.206
LGBM	11.383	0.266	0.502	92.427	0.12
CatBoost	10.548	0.257	0.528	91.624	4.166
MLP	12.374	0.293	0.545	91.071	93.922
LSTM	16.073	0.39	0.587	89.643	112.843

In both cases, with and without hyperparameter tuning, overall, the GBDTs seem to have the best performance, compared to the ANNs. More precisely, regarding the entire dataset training and validation, all the produced evaluation metrics show CatBoost to have a slight edge, compared to the other models, followed closely by the LGBM which offers similar results but with a much smaller training runtime (ET).

For the LGBM model, the improvements in the metrics due to the hyperparameter tuning are somewhat limited. Particularly, the improvement of MAPE was around -1% for LGBM and -11% for CatBoost. As for MAE the improvement was around -0.8% for LGBM and -5.5% for CatBoost, indicating marginal improvements for LGBM but significant for CatBoost. That said Random Forest, MLP and the LSTM models had the most

gains in accuracy for hyperparameter tuning and optimization, in particular MLP's MAPE was reduced by around -32%, Random forest's MAPE by around -13% and LSTM's MAPE around -42%. We consider MAPE as the most representative value as it does not relate with the range of values and the target

Variable (Energy load) and can be used for direct comparisons with other relevant scientific work, taking into account the used parameters and preprocessing approach. Figure 34, provides a visualization of the one-step forecasts for a typical 48h time interval, for each of the implemented models. It is evident that the LSTM performance is worse than the GBDTs, with all hyperparameters tuned. It captures the overall trend but with a small offset it the absolute values.

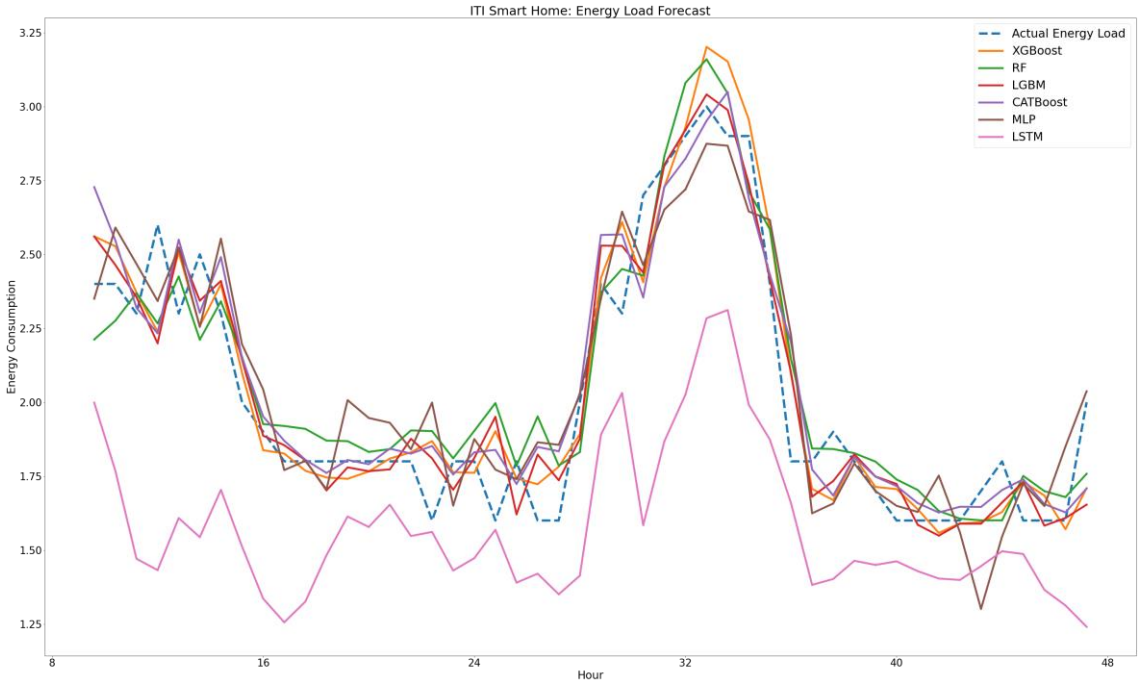


Figure 34: One step ahead prediction (48h) compared with the actual energy load

5.2 Application on dataset subsets

After developing and fine-tuning the selected forecasting models, our next approach was to test their behavior against smaller partitions of our dataset. This may possibly allow each algorithm to be able to better recognize a less complex pattern and make more accurate predictions within a subset of the data set rather than relying on a single model to fit into a larger portion of the data. In this regard, we partitioned the dataset in a random manner, aiming to create different time period each with different characteristics, and avoiding selecting large periods of inactivity. The developed and trained models as

discussed in Section 5.1.1, are tested independently using partitioned data for the following time periods:

- Mar.2019 - May. 2019
- Sep. 2019 – Nov. 2019
- Nov. 2019 – Jan. 2020
- Jun. 2020 – Aug. 2020

For the model performance we produced MAPE, MAE, RMSE and R-squared, as shown in the following tables.

Table 11: Model performance over dataset subpart 1 (Mar.2019 - May. 2019)

Evaluation metrics	MAPE (%)	MAE (kWh)	RMSE (kWh)	R-squared (%)
Dataset subpart 1	**(Mar.2019 - May. 2019)**			
Single model performance (w/ hypertuning)				
XGBoost	6.475	0.176	0.268	94.955
Random Forest	4.74	0.135	0.221	96.586
LGBM	7.156	0.194	0.294	93.939
CatBoost	6.933	0.197	0.306	93.435
MLP	9.137	0.252	0.377	90.013
LSTM	18.115	0.48	0.738	61.796

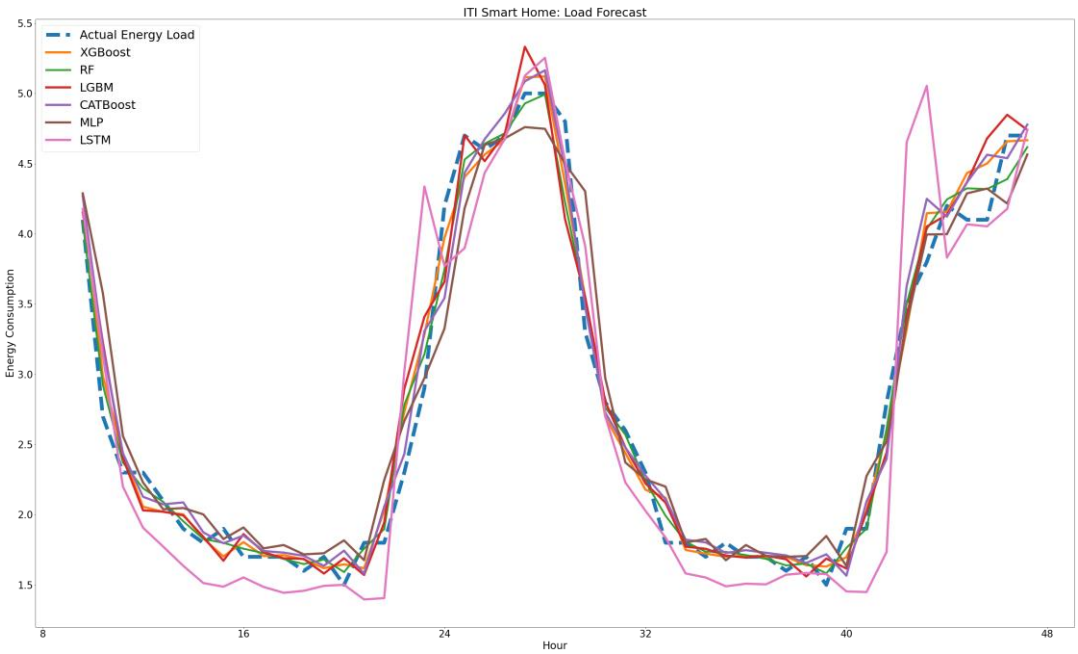


Figure 35: One step ahead prediction (48h) for Dataset subpart 1

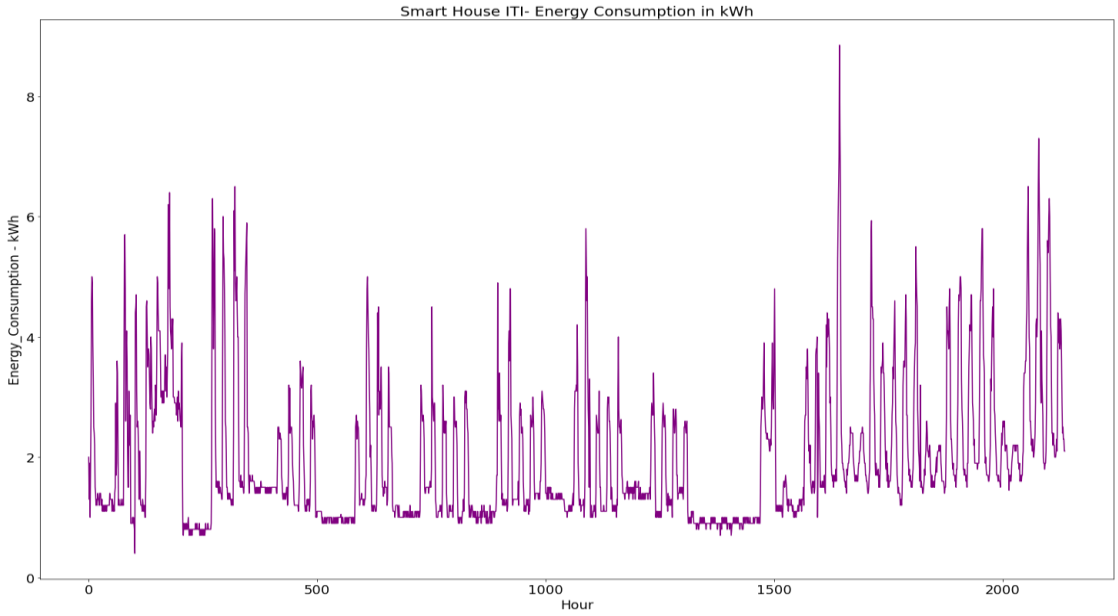


Figure 36: Energy load profile for dataset subpart 1 (Mar.2019 - May. 2019)

This is a similar subset of the ITI Smart House energy load data used in [21] and the performance metrics are greatly improved than the performance over the test subpart of the entire dataset, with the exception of LSTM. A possible explanation for this is that the selected subpart seems to have a repeatable pattern, without any extensive periods of low consumption, while the min-max values are around the same level for most datapoints. Hence, the models can make more accurate forecasts for this type of a more 'normalized' pattern.

Table 12: Model performance over dataset subpart 2 (Sep.2019 - Nov. 2019)

Evaluation metrics	**MAPE (%)**	**MAE (kWh)**	**RMSE (kWh)**	**R-squared (%)**
Dataset subpart 2: (Sep.2019 - Nov. 2019)				
Single model perfomance (w/ hypertuning)				
XGBoost	10.198	0.223	0.354	91.85
Random Forest	7.415	0.167	0.302	94.083
LGBM	11.1	0.244	0.388	90.239
CatBoost	10.82	0.252	0.452	86.738
MLP	14.357	0.31	0.487	84.558
LSTM	17.321	0.394	0.708	67.442

Figure 37: One step ahead prediction (48h) for Dataset subpart 2

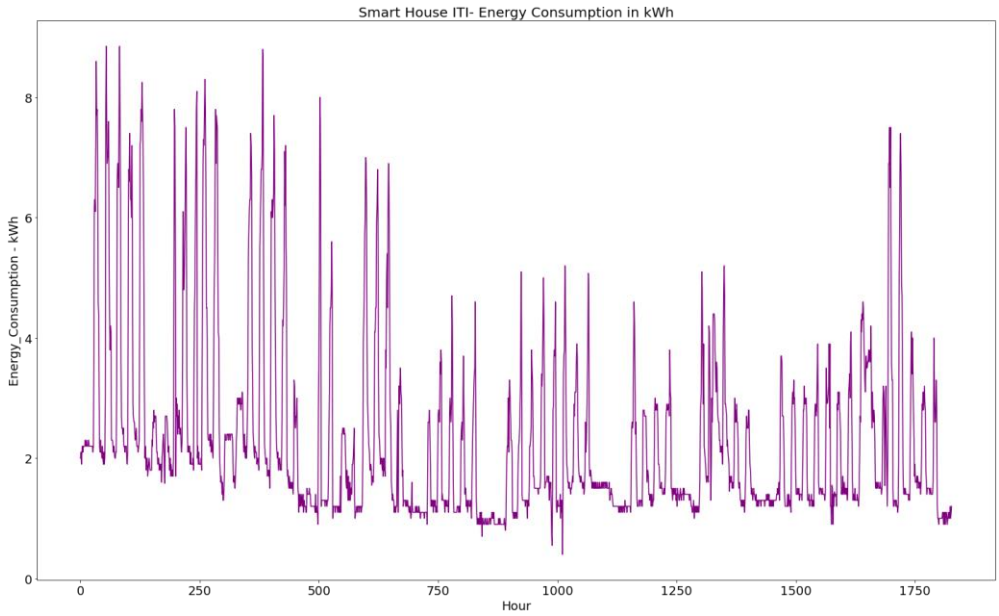


Figure 38: Energy load profile for dataset subpart 2 (Sep.2019 - Nov. 2019)

The metrics for the subpart 2 are worse than the metrics of subpart 1, however for some models they are still the same or better than the original metrics for forecasting over the entire dataset. Again, we see that the energy load profile for this period, is somewhat normal without any notable extremities, thus the models can predict with higher accuracies. Interestingly for both cases of subpart 1 and 2 the Random Forest algorithms has the best performance from the rest, by a considerable margin. More precisely, MAPE is

around -26% lower than the next best algorithm (XGBoost), indicating for such repetitive and less complex patterns Random Forest performs very well.

The evaluation metrics of the performance of the trained models on dataset subpart 3 are presented in Table 13. Overall, the metrics are worse for all models compared to the metrics for subparts 1 and 2.

Table 13: Model performance over dataset subpart 3 (Sep.2019 - Nov. 2019)

Evaluation metrics	MAPE (%)	MAE (kWh)	RMSE (kWh)	R-squared (%)
Dataset subpart 3 (Nov.2019 - Jan.2020)				
Single model perfomance (w/ hypertuning)				
XGBoost	10.854	0.444	0.605	86.495
Random Forest	10.676	0.425	0.625	85.596
LGBM	**13.094**	**0.521**	**0.741**	**79.748**
CatBoost	15.775	0.639	0.91	69.451
MLP	15.465	0.627	0.834	74.367
LSTM	17.387	0.762	1.04	60.068

Figure 39: One step ahead prediction (48h) for Dataset subpart 3

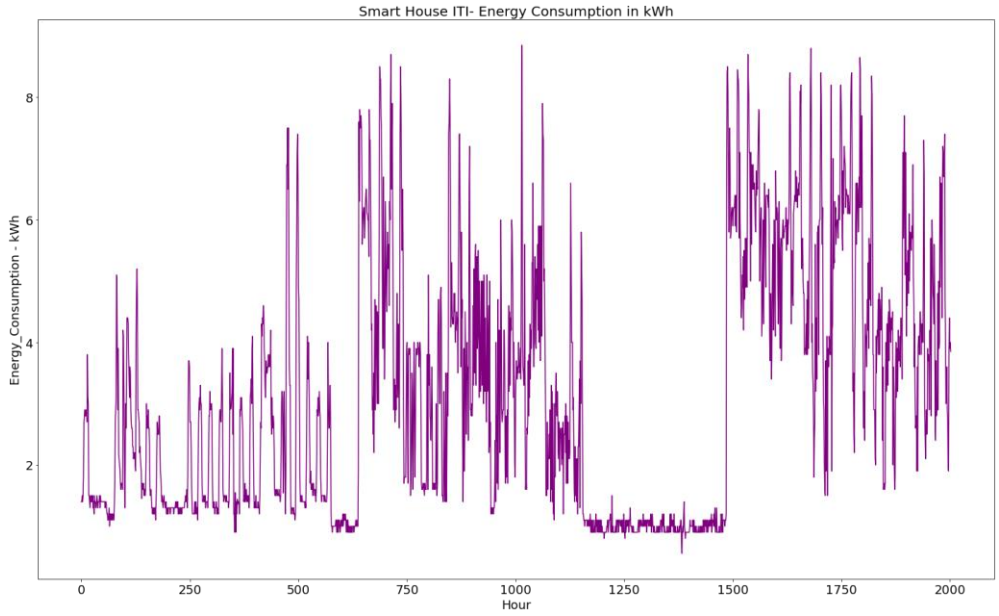


Figure 40: Energy load profile for dataset subpart 3 (Nov. 2019- Jan. 2020)

However, there is a similar pattern as subpart 1 and 2, the Random Forest algorithm is again the best performing (based on all produced metrics) and the ANNs are less accurate than the GBDTs. One possible scenario that may explain why the performance of the models deteriorates in this case, is the presence of a large period of near-zero energy load (due to Christmas holidays), as depicted in Figure 40, which disrupts the repetitive and similar pattern of the rest of the dataset. Also, the average energy load records an important increase towards the end of the dataset subpart 3 (i.e. January), compared with the early datapoints (i.e. November), indicating the change in thermal needs due to lower temperatures in January.

The evaluation metrics of the performance of the trained models on dataset subpart 4, which includes the time period between Jun.2020 - Aug. 2020, are presented in Table 14. Overall, the metrics are noticeably worse for all models compared to the metrics for subpart 3. Remarkably the Random Forest is once more the best-performing algorithm, in this subpart as well, while both ANNs continue their poor performance compared to the rest of the GBDTs.

Table 14: Model performance over dataset subpart 3 (Jun.2020 - Aug. 2020)

Evaluation metrics	MAPE (%)	MAE (kWh)	RMSE (kWh)	R-squared (%)
Dataset subpart 4 (Jun.2020 - Aug. 2020)				
Single model performance (w/ hypertuning)				
XGBoost	8.85	0.219	0.394	94.671
Random Forest	8.69	0.218	0.415	94.084
LGBM	**9.14**	**0.215**	**0.384**	**94.935**
CatBoost	8.61	0.218	0.434	93.532
MLP	10.79	0.238	0.422	93.897
LSTM	18.54	0.417	0.673	84.476

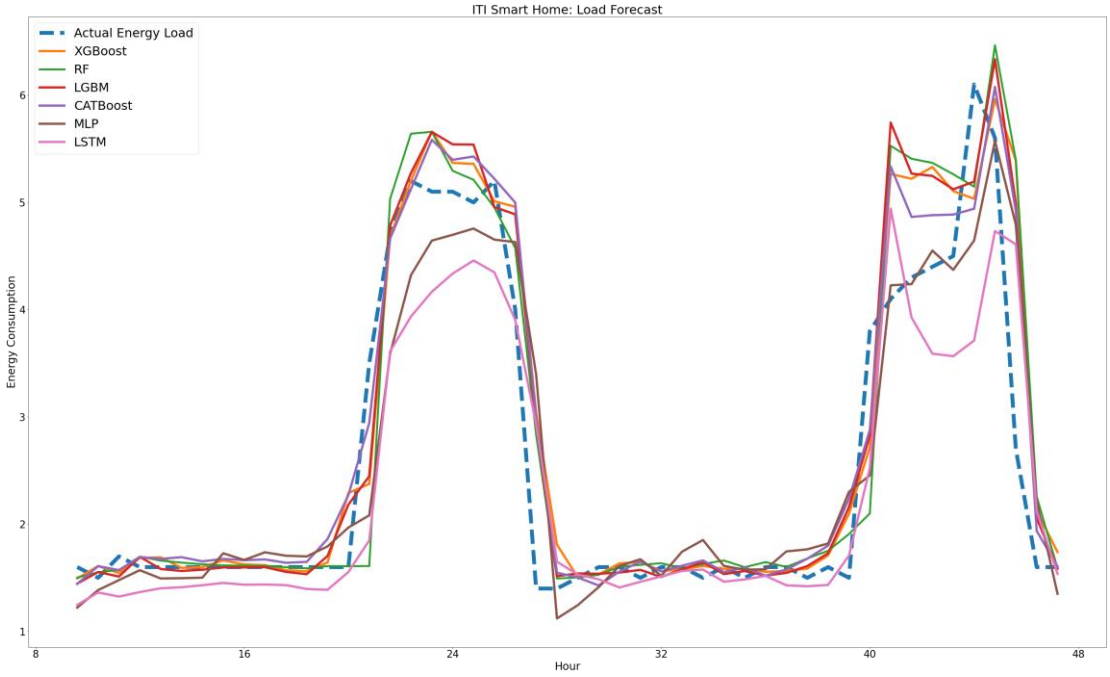


Figure 41: One step ahead prediction (48h) for Dataset subpart 4

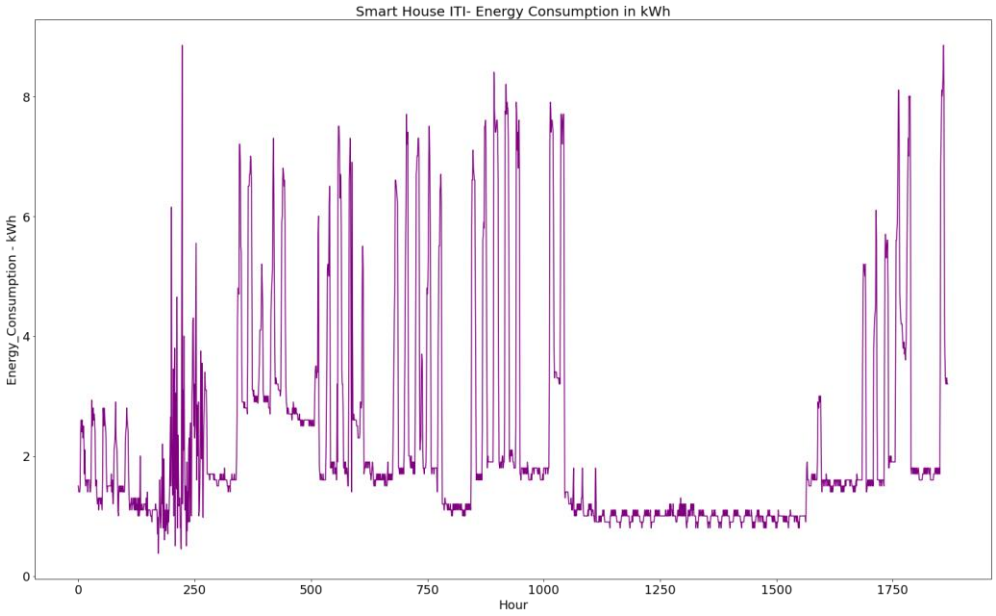


Figure 42: Energy load profile for dataset subpart 4 (Jun.2020 - Aug. 2020)

As shown in Figure 42, the energy load profile for this time period, records an even larger period of near-zero energy load compared to the one in subpart 3, which is due to the summer vacation in August, a period in which most employees take their annual summer leave. This possibly explains the further deterioration of the performance of all models, as it distorts the normally periodical pattern of energy load.

5.3 Cross-validation results

As highlighted in Section 3.4, cross-validation (CV) is an important step when developing a forecasting model, in order to assess its prediction performance over new/future test data. It is especially critical in our case as our dataset is not large or diverse enough to create representative train and test datasets. Since we are dealing with a time series problem, the typical k-fold validation technique used in most cases is not suitable, as it assumes that the data points are independent of each other. Instead, the most appropriate TimeSeriesSplit library of Sklearn was chosen, using a 5-fold split, which progressively divided the dataset into training and validation fold, maintaining the chronological order of the data points (Figure 43). The cross-validation averaged metrics included MAPE, MAE, RMSE and R-squared.

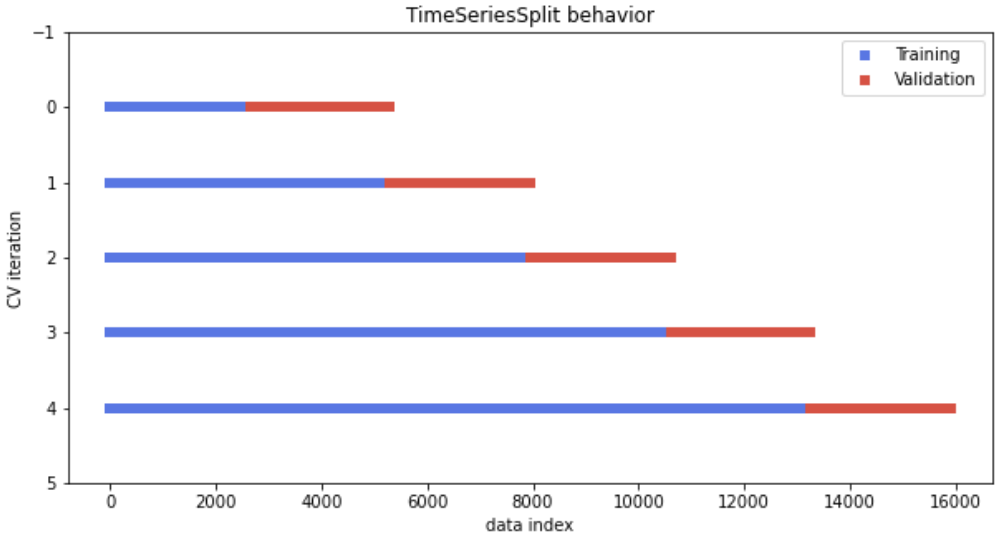


Figure 43: Time series 5-fold training/validation splits over the entire dataset

The results for the cross-validation averaged metrics concerning the entire test dataset are presented in Table 15. The averaged metrics from the cross-validation, are slightly worse than the single model runs, after hypermarket tuning (Section 5.1.2). This possibly indicates some extent of overfitting and may also be related to the hyperparameter optimization based on the entire dataset training/test split. Notably, the best-performing model based on the cross-validation results is the same with the single separate model case, the CatBoost model, followed closely by the second best the LGBM. Another key takeaway is the performance of the LSTM, which remains about the same level as in the single model results, in fact having the best RMSE and R-squared scores than the other cross-validated model results. This indicates that LSTM better deals with potential overfitting and is more suitable to deal with new (unseen) data, which may have conflicting patterns or frequent extremities.

Table 15: Cross validation results for the entire dataset

Evaluation metrics	**MAPE (%)**	**MAE (kWh)**	**RMSE (kWh)**	**R-squared (%)**
Entire Dataset	**(Oct.2018 - Sep. 2020)**			
Cross-validation averaged results				
XGBoost	14.128	0.369	0.627	86.56
Random Forest	14.271	0.369	0.632	86.07
LGBM	14.140	0.362	0.607	87.07
CatBoost	12.984	0.353	0.627	86.52
MLP	20.037	0.517	0.818	71.84
LSTM	15.093	0.366	0.592	87.60

The results for the cross-validation averaged metrics concerning the entire dataset are presented in Table 16.

Table 16: Cross-validation results for the dataset subparts

Evaluation metrics	MAPE (%)	MAE (kWh)	RMSE (kWh)	MAPE (%)	MAE (kWh)	RMSE (kWh)
Cross-validation averaged results						
Subpart 1 (Mar.2019 - May. 2019)				**Subpart 3 (Nov.2019 - Jan.2020)**		
XGBoost	13.962	0.267	0.447	35.055	0.9	1.279
Random Forest	12.571	0.243	0.404	22.074	0.696	1.032
LGBM	13.058	0.261	0.433	**26.929**	**0.781**	**1.105**
CatBoost	12.441	0.262	0.454	24.793	0.792	1.149
MLP	41.845	0.615	0.799	75.788	1.899	2.704
LSTM	20.265	0.336	0.498	24.783	0.588	0.791
Subpart 2 (Sep.2019 - Nov. 2019)				**Subpart 4 (Jun.2020 - Aug. 2020)**		
XGBoost	20.286	0.371	0.557	18.462	0.396	0.645
Random Forest	19.321	0.353	0.524	12.789	0.32	0.559
LGBM	19.589	0.356	0.529	**21.408**	**0.476**	**0.703**
CatBoost	21.384	0.391	0.587	16.154	0.388	0.638
MLP	34.353	0.672	1.053	41.585	0.891	1.031
LSTM	19.725	0.363	0.572	19.54	0.339	0.547

We first observe that the performance metrics of all models are generally worse than the single model performance presented in Section 5.2. This is to be expected as cross-validation simulates the models' performance on new unseen data. Furthermore, it is evident that apart from dataset subpart 1, for the rest of the subparts the best-performing model is the LSTM. This is potentially linked with the fact that apart from subpart 1 which records a more normalized and periodical pattern regarding the energy load profile, the rest subparts, and especially subparts 3 and 4, had a more complex pattern. More precisely the LSTM on subparts 3 and 4, has a similar performance as the CatBoost and LGBM in the cross-validation over the entire dataset (as shown in Table 15). Hence, this is a strong indicator that the LSTM model better deals with overfitting and performs quite well on new/unseen data, especially with more abnormal patterns. More precisely for subparts 3 and 4 the LSTM has lower RMSE by -2% and -23% respectively, from the best-performing algorithm (Random Forest). This behavior to a great extent coincides with the findings of the literature review (e.g. [30], [31], [58]), in which the improved accuracy of LSTM on more complex energy load datasets is highlighted

5.4 Hybrid Model

In ML, ensemble learning techniques combine several base learning algorithms to produce one predictive model with optimal performance results. Two of the most commonly used ensembling methods, used in regression problems are Stacking and Voting, which as summarized below.

Stacking

Stacking ensemble is a ML technique technology that combines conventional predictive algorithms to produce a more accurate prediction by using the predictions of the trained base algorithms as a new input [76]. A stacking model's architecture is made up of two or more base predictive algorithms and a meta-model that integrates the predictions of the base algorithms. For regression and classification applications, the stacking ensemble technique has been shown to out-perform traditional machine learning algorithms. The stacking ensemble strategy works well because it uses the capabilities of many models to provide a more accurate forecast [77], [78].

Voting

To forecast the final prediction based on the voting strategy, a voting ensemble approach aggregates the predictions of many base algorithms. The voting ensemble ap-proach, which in summary gathers the knowledge of each base learner, can create more accurate predictions than any single predictive algorithm. Furthermore, combining the predictions of multiple base algorithms contributes to better dealing with potential overfitting. In regression-based models built with voting, the predictions of each regression base model are averaged to produce a final prediction. Hence, a voting regressor model is an ensemble meta-estimator that fits many base algorithm regressors, one at a time, on the whole dataset, and then averages the individual fore-casts to generate a final prediction.

In order to improve the accuracy and overall robustness of our forecasts, taking into consideration the results in the previous Sections, we conceptualized and developed an ensemble approach. Furthermore, we looked into relevant literature, to explore similar approaches of ensembling, implemented in the field of energy/load forecasting, some notable studies are presented below:

- Phyo et al., 2022: forecasted energy load, with an average MAPE of 4.28%

- Zhao et al., 2021: focused on day-ahead STLF in Australia, achieving an average MAPE of 1.13%
- Moon et al., 2018: developed a hybrid STLF model for an educational building complex, achieving a MAPE ranging from 3-4.7%
- Khan and Byun, 2021: implemented an ensembling approach for energy prediction, with promising results
- Massaoudi et al., 2021: implemented a stacked generalization approach for STLF regarding power supply industry data

As a result, we developed a one-step ahead energy load forecasting Hybrid Model (OSA-ELF HM) which combines the benefits of both stacking and voting methods. We use a two-level stacking-based approach that combines the outputs of the first-level models on the second level with another model. This second-level model, however, is a voting regressor model. Based on the metrics produced in the previous Sections, we choose the three best overall performing models to synthesize our voting regressor, the CatBoost, the LGBM, and the Random Forest. The overall architecture of our approach is summarized in Figure 44. In more detail, the level-1 learners consist of all six trained base models while the level-2 meta-learner averages the predictions of CatBoost, Random Forest, and LGBM. Our strategy is to benefit from the forecasting potential of each heterogeneous base model and enhance the overall stacking ensemble with the high performance of the meta-learner's models.

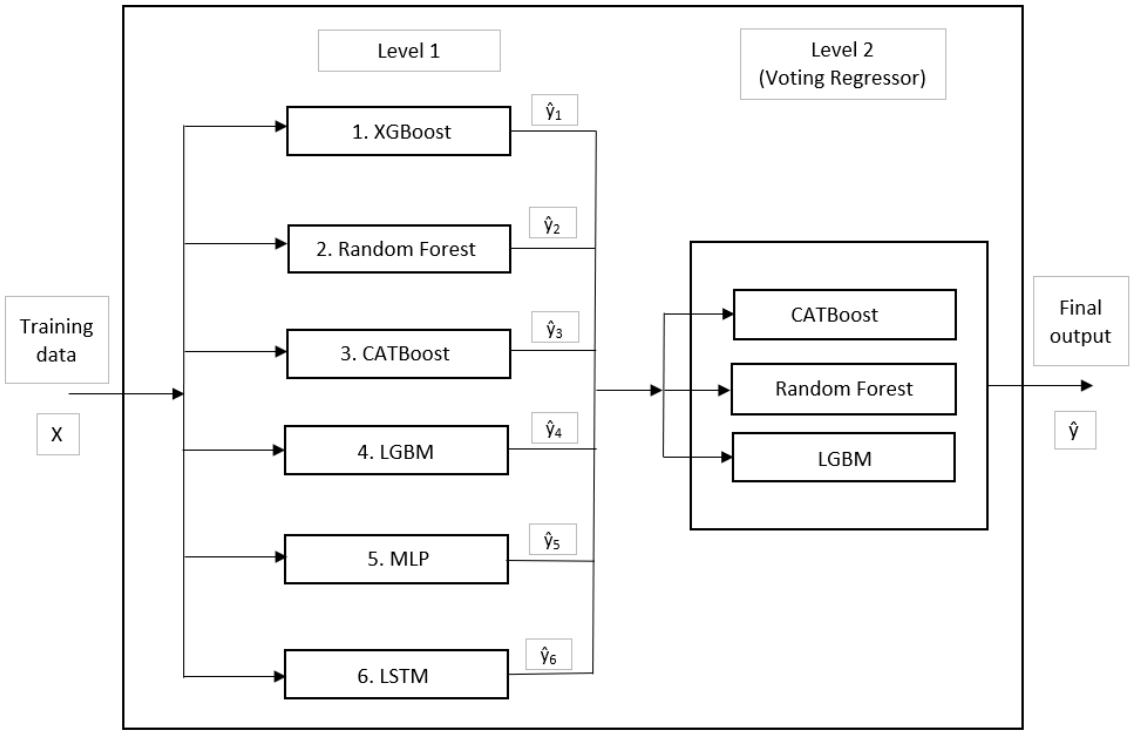


Figure 44: The architecture of the OSA-ELF HM

At the first level, the test feature dataset named X is fed to the six already trained and tuned base learners. The outputs of the 1st-level models form a new feature vector ($\hat{y}_1$, $\hat{y}_2$,… $\hat{y}_6$). The new vector is then fed as training parameters into the second-level voting regressor model which outputs our target $\hat{y}$, which is one step ahead Energy load forecast. This hybrid meta-ensembling approach aims to further improve the accuracy of the predictions of the already trained base models. The resulting evaluation metrics from this process are presented in Table 17.

Table 17: Evaluation metric for the OSA-ELF HM

Evaluation metrics	**MAPE (%)**	**MAE (kWh)**	**RMSE (kWh)**	**R-squared (%)**
Entire Dataset	**(Oct.2018 - Sep. 2020)**			
OSA-ELF HM	5.818	0.139	0.286	97.545

The OSA-ELF HM improves the forecasting accuracy of our target (Energy load). All evaluation metrics are improved across the board compared with each individual base model's performance on the entire test dataset (Table 10). The energy load forecasting curves of the proposed method vs. the actual historical load for a typical 72-hour window are shown in

Figure 45.

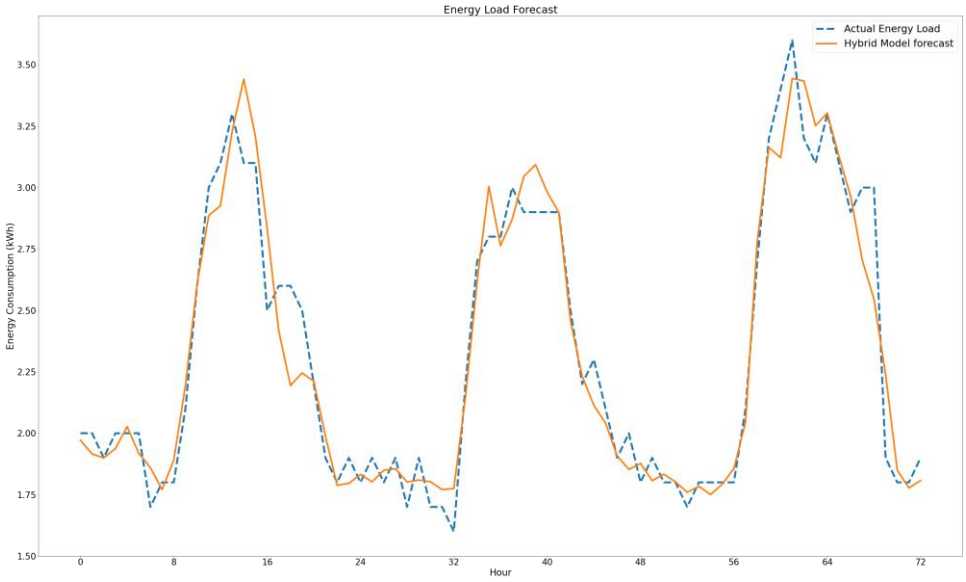


Figure 45: 72h prediction of the OSA-ELF HM for our original dataset

5.5 Validation of Hybrid Model

Instead of utilizing the cross-validation approach to validate the OSA-ELF HM approach on our test dataset, we decided to benchmark our approach based on new and unseen data, that have similar properties as the ITI Smart House dataset, i.e. an office-use type of building, which we used for training our base models. In recent years, a lot of efforts focusing on building energy analytics have been made in order to create publicly available energy-related data measurements for providing test datasets or benchmarking modeling algorithms. One such prominent effort is linked with the Building Data Genome Project 2 (BDGP2) dataset, led by an international collaboration of building energy-related academics and experts [84]. BDGP2[12] is an open data repository has data from 1,636 non-residential buildings. It includes hourly whole-building data for two years, from different kinds of (smart) meters: electricity load, heating, cooling water, steam, energy from solar (PV) and other meter data; in addition, this data set integrates outdoor temperature, humidity, cloud coverage, and other climatic factors that can affect energy load. Hence, it matches the characteristics of the ITI Smart House dataset in order to benchmark our OSA-ELF HM approach. From the BDGP2, we selected the

[12] https://doi.org/10.5281/zenodo.3887305

‘*Bobcat_education_Coleman*’ building, which is an education/academic building located in the US. Two important reasons for choosing this particular building are that it has solar power production (i.e. from PV) and matches the usage-profile of the ITI Smart house as an office/academic/research type of building. The time interval extracted data range from 01/2016 to 31/2016 while the sampling interval is 1h as our original train/test dataset. The detailed definitions of the data are described in Table 18.

Table 18: The parameters used from the BDGP2 dataset

Variables	**Units**	**Definition**
TimeStamp	-	Date and time in the local time zone
Load	kWh	Total electric energy used over 1h
AirTemperature	°C	The temperature of the air in degrees Celsius
Solar	kWh	Total energy produced by PV over 1h

We applied the same outlier detection strategy as our original dataset (see Section 4.1.2) in the new dataset and removed extreme anomalies. The final dataset to validate our approach is depicted Figure 46.

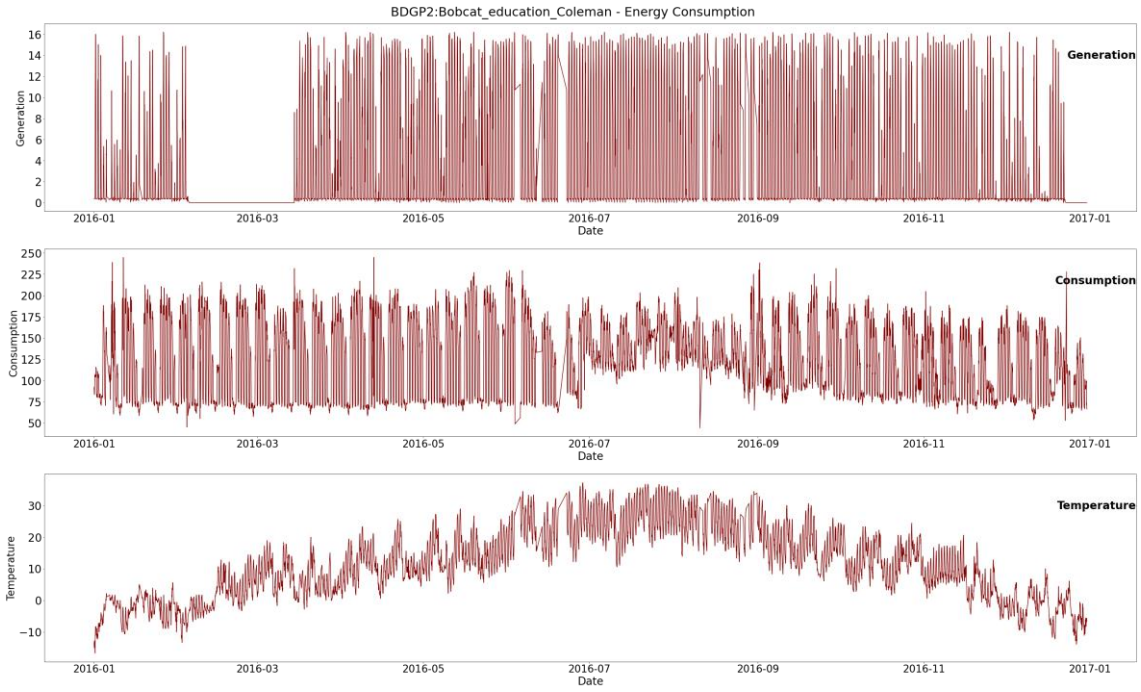


Figure 46: BDGP2 (Coleman Building) all hourly time series data

It is necessary to highlight that this is a significantly larger building than the ITI Smart House, covering a total area of 10552 square meters, as a result, the range of the Energy load values is significantly higher than our original dataset, as the energy needs are higher (heating, electricity,

cooling). More precisely the mean value for Energy load is around 125 kWh (per hour) while for the ITI Smart House dataset it is around 2.7 kWh, the rest of parameters' statistics from the two datasets are summarized in Table 19.

Table 19: Statistics comparison between the original and the validation datasets (after outlier detection)

Parameters	**mean**	**std**	**min**	**max**
ITI Smart House Dataset				
Energy_Consumption (kWh)	2.66	1.85	0.00	8.85
Energy_Generation (kWh)	1.62	2.52	0.00	9.54
Temperature_v1 (°C)	17.21	8.42	-3.90	38.05
BDGP2 Dataset				
Consumption (kWh)	125.45	42.78	53.00	222.25
Generation (kWh)	3.00	4.75	0.00	16.20
Temperature (°C)	11.23	11.07	-16.70	37.20

Table 20: Evaluation metric for the OSA-ELF HM on the BDGP2 data

Evaluation metrics	**MAPE (%)**	**MAE (kWh)**	**RMSE (kWh)**	**R-squared (%)**
Entire Dataset	**(Jan.2016 - Dec. 2016)**			
OSA-ELF HM	5.544	6.652	10.354	94.17

We note that our OSA-ELF HM has a similar performance on the new unseen data, based on the evaluation metrics depicted in Table 20. The focus is more on MAPE and R-squared, since they are directly comparable with the metrics produced by applying our OSA-ELF HM on our original dataset. This is due to the fact that as already discussed the BDGP2 dataset derives from a larger building, hence the mean value of the target variable (Energy load) is much higher, as a result, this raises the MAE and the RMSE accordingly, making them unsuitable for direct comparisons between the two validation datasets. The MAPE is improved by -5% compared with the original dataset. The energy load forecasting curves of the proposed method vs. the actual historical load for a typical 72-hour window are shown in Figure 47.

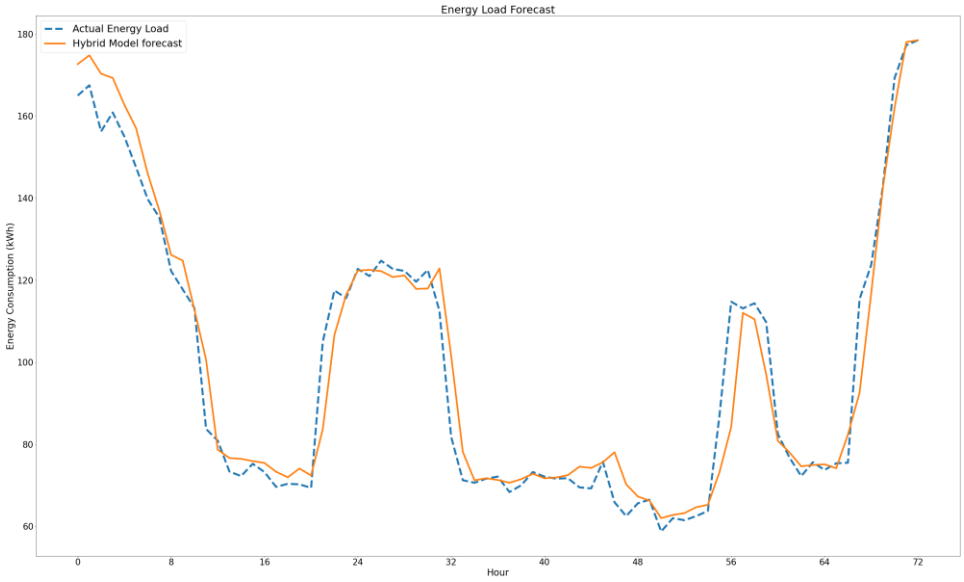


Figure 47: 72h prediction of the OSA-ELF HM for unseen data (BDGP2 dataset)

6 Discussion

6.1 Results evaluation

This work aimed to review relevant literature and analyze the factors affecting building energy load patterns, the state-of-the-art preprocessing techniques and ML forecasting algorithms. It also aimed to develop a customized framework of forecasting models, taking into account the unique characteristics of the utilized dataset and validating it on new unseen data.

Overall, 6 different base ML models were developed and tuned for energy load prediction, each with its own strengths and weaknesses. Moreover, extensive feature selection, synthesis, and transformation took place in order to maximize the performance of the models. Overall, tree-based ensemble algorithms had the best forecasting performance, even without choosing the best hyperparameters. More specifically, CATBoost and LGBM produced particularly good results, with minimal execution times.

The most important takeaways from this work are briefly listed below:

- All deployed models benefited in terms of performance, from hyper parameter tuning. That said, there was a larger benefit for the ANNs rather than the tree-based models, which saw smaller improvements compared to their unoptimized forecasting results.
- The average results from the cross-validation performed on the entire test dataset, are slightly worse than the single model runs, but still, Catboost remains the best-performing model. This may indicate some extent of overfitting to the training dataset, possibly also related to the hyperparameter optimization process.
- The developed and trained base models were benchmarked with different subparts of the dataset. The accuracy of the models increases when the energy load follows more standardized patterns. That said, the Random Forest algorithm performed well on all subparts, indicating that it better deals with overfitting. This is most likely related to the element of randomness i.e. each tree draws a random sample of data from the training dataset when generating its splits

- Further, in the cross-validation exercise of the dataset subparts, the LSTM increased its performance significantly in terms of evaluation metrics. This indicates that the LSTM is less susceptible to overfitting and is better equipped to provide predictions on unseen data or more complex energy load patterns.
- The most important feature/predictor unanimously for the decision tree-based models, was the previous hour energy load value (T-1) to make their forecasts. Overall, proper feature selection seems to be more important than the selection of base algorithms or the evaluation techniques (i.e. cross-validation)
- The ANNs developed (MLP, LSTM) require longer training/execution times since they incorporate more computationally expensive computations.
- The developed Hybrid Model (OSA-ELF HM) significantly increased the forecasting accuracy on the ITI Smart House test dataset, achieving a MAPE of 5.818% and an R-squared of 97.545%. This translates to a 45% reduction in MAPE from the best-performing single base forecasting algorithm.
- This was further validated by applying the OSA-ELF HM to a dataset with similar properties, when it achieved similar accuracy (MAPE: 5.54%). Moreover, as a direct comparative basis, a very relevant scientific paper [21] utilizing the same dataset produced a forecasting model that achieved a MAPE of around 13%, without however the presence of exogenous parameters. Also, most of the relevant scientific papers identified in the literature review demonstrated a MAPE in the range of 5-15%. Hence, the developed model falls closer to the higher end of this performance margin.

6.2 Threats to validity

Often scientific works are constrained by time limitations and computational resources availability to reach conclusions that are subjected to further analysis. Therefore, the work presented here may be subject to certain threats that might affect the validity of the results. The cause of such potential threats could be attributed to the nature of the data and its quality/granularity, as well as the selection of specific preprocessing methods or predictive algorithms.

A. The first potential threat is related to the training dataset itself. As mentioned in the preprocessing steps, the initial raw dataset contained many missing or non-valid values. These values were partly filled out via interpolation and aggregation to 1h resolution, which however meant that it limited the size of the training data. This is especially relevant to the performance of deep learning models (such as the LSTM), which, as the literature suggests, typically require large amounts of data to achieve higher levels of accuracy. Perhaps a larger training dataset could further improve their performance.
B. Another potential threat is related to the exogenous parameters required by our developed model. More specifically the model requires ambient temperature measurements and energy generation from PV. Particularly the second attribute may be more rarely available when one wants to test our model, especially for a non-smart or nZEB building. However, we consider that the accuracy loss of removing this feature would be minimal, based on its score on the feature important graphs.
C. Based on their high usability in the research community MAPE, MAE, RMSE, and R-squared are often used metrics for measuring overall forecasting performance. However, other relevant metrics exist, such as notably SMAPE and CVRMSE that are often utilized for regression-based forecasting, as well that may be integrated, potentially mitigating any unexplained discrepancies with the final evaluation of data.
D. This thesis recognizes the existence of various ensemble techniques, prediction algorithms, and time series forecasting methodologies. However, the selection and development of the models and combination thereof, presented are considered up-to-date popular methods with several applications to time series forecasting. In the case of prediction model possibilities, for example, state-of-the-art ANNs, MLP, and LSTM are given, as well as traditional ML models (e.g. XGBoost, Random Forest and LGBM).

7 Conclusions and future work

In this section, we attempt to critically review and provide a synopsis of the results and ideas for future work.

7.1 Conclusions

ELF, particularly in the build environment, has attracted increased research interest in recent years and remains a complex problem. Accurate and robust building ELF requires a meticulous development and a deep understanding of the algorithmic approaches implemented in this domain. The goal of this thesis was to provide the best possible forecast on a smart building's energy load, while accounting exogenous factors such as ambient temperature. To achieve this, we focused on examining some of the best known regression models and the eventual deployment of non-complex, interpretable, and state-of-the-art base models, as well as the evaluation of their performance in energy load prediction. Experimenting with selected single base algorithms, including hyper-parameter tweaking and cross-validation exercises, demonstrated that decision tree-based algorithms have the potential for improved performance. That said, to improve the forecasting ability, tackle over-fitting effects and augment overall robustness, a new one-step ahead hybrid model (OSA-ELF HM) was developed, using meta-ensembling approaches.

The publicly available utilized dataset contains real-world entries (metered data from a smart house infrastructure) associated with energy load values. A benchmarking public data set was utilized with the aim to validate the performance of the developed hybrid model. The model creates a heterogeneous feature matrix using automatically picked lagged values from the load time series and manually derived temporal parameters. Furthermore, it was explored how targeted hyperparameter tuning affects model accuracy and how certain building aspects used as features may increase accuracy. Three key findings of this study should be highlighted:

- The synthesis of a customized, heterogeneous yet reproducible in smart building case studies feature set significantly increased the forecasting accuracy of the developed model, even more so than the hyper parameter tuning exercise.
- The OSA-ELF HM model is superior to single ML models, as it improves forecasting accuracy and better tackles potential overfitting.
- The OSA-ELF HM model was tested on unseen data, and performed quite well, indicating that it has the potential to address in a uniform manner any given smart building energy load prediction task, regardless the ELF horizon (i.e. VSTLF, STLF, MTLF and LTLF)

7.2 Future research directions

This work focuses on a challenging problem that necessitated substantial re-search and testing with several regression-based forecasting techniques. Given the limited time and resources, this dissertation aimed to cover as many topics as possible in the context of energy load timeseries forecasting. Most research studies, however, have space for improvement and new components to be examined. We highlight here possible improvement paths for the proposed approach and future work:

A. It is obvious that bigger datasets with real-world energy load data entries always tend to lead to more accurate and reliable results. In our work, the buildings examined are used as office spaces, hence data from other buildings (e.g. residential or industrial) with PV installations and ideally weather information would provide a more comprehensive view on how to increase the forecasting accuracy and limitations on its applicability/scalability.
B. Regarding ANNs implementation, bigger training datasets and a more thorough selection of hyperparameters are needed to better assess the potential in energy load forecasting. Future work should also expand towards testing additional deep learning architectures, especially in the meta-learning stage, and supplementing the data representation possibly creating other alternative features to further enhance the model accuracy.
C. Another research direction is to expand the proposed single-step approach to a multi-step forecasting approach, focusing more on the STLF horizon. In a multi-step approach, forecasts are utilized as input values in future predictions; mean-

ing that it is a repeated one-step prediction process, with the actual prediction used to predict the following value.

Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
B-LSTM	Bidirectional LSTM
CERTH	Centre for Research & Technology
CV	Cross-validation
ET	Elapsed Time
GBDT	Gradient Boosted Decision Trees
GDP	Gross domestic product
GHG	Greenhouse Gas Emissions
GSCV	Grid search cross-validation
ICT	Information and communications technologies
ITI	Informatics and Telematics Institute
LTLF	Long-term load forecasting
ML	Machine Learning
MTLF	Medium-term load forecasting
nZEB	near Zero Energy Building
OSA-ELF	One-step ahead Energy Load Forecasting
OSA-ELF HM	One-step ahead Energy Load Forecasting Hybrid Model
PV	Photovoltaics
RNN	Recurrent Neural Network
STLF	Short-term load forecasting
VSTLF	Very short-term load forecasting
ZEB	Zero Energy Building

References

[1] European Commission, “Stepping up Europe’s 2030 climate ambition Investing in a climate-neutral future for the benefit of our people,” *COM(2020) 562 final*, 2020.

[2] Z. Jing *et al.*, “Commercial Building Load Forecasts with Artificial Neural Network,” in *2019 IEEE Power and Energy Society Innovative Smart Grid Technologies Conference, ISGT 2019*, Feb. 2019. doi: 10.1109/ISGT.2019.8791654.

[3] K. M. Al-Obaidi, M. Hossain, N. A. M. Alduais, H. S. Al-Duais, H. Omrany, and A. Ghaffarianhoseini, “A Review of Using IoT for Energy Efficient Buildings and Cities: A Built Environment Perspective,” *Energies*, vol. 15, no. 16. MDPI, Aug. 01, 2022. doi: 10.3390/en15165991.

[4] D. Djenouri, R. Laidi, Y. Djenouri, and I. Balasingham, “Machine learning for smart building applications: Review and taxonomy,” *ACM Comput Surv*, vol. 52, no. 2, May 2019, doi: 10.1145/3311950.

[5] A. Mystakidis *et al.*, “One Step Ahead Energy Load Forecasting: A Multi-model approach utilizing Machine and Deep Learning; One Step Ahead Energy Load Forecasting: A Multi-model approach utilizing Machine and Deep Learning,” 2022, doi: 10.1109/UPEC55022.2022.9917790.

[6] F. Pedregosa *et al.*, “Scikit-learn: Machine Learning in Python,” Jan. 2012, [Online]. Available: http://arxiv.org/abs/1201.0490

[7] Aristeidis Mystakidis and Christos Tjortjis, “Big Data Mining for Smart Cities: Predicting Traffic Congestion using Classification,” in *11th International Conference on Information, Intelligence, Systems and Applications (IISA)*, 2019.

[8] J. King and C. Perry, “Smart Buildings: Using Smart Technology to Save Energy in Existing Buildings,” 2017.

[9] European Commission, “Smart technologies in buildings,” 2022. https://energy.ec.europa.eu/topics/energy-efficiency/energy-efficient-buildings/smart-readiness-indicator/smart-technologies-buildings_en (accessed Sep. 21, 2022).

[10] European Commission, “In focus: Energy efficiency in buildings,” Feb. 2020.

[11] S. Attia *et al.*, “Overview and future challenges of nearly zero-energy building (nZEB) design in Eastern Europe,” *Energy Build*, vol. 267, Jul. 2022, doi: 10.1016/j.enbuild.2022.112165.

[12] European Commission, “Energy4Europe on Twitter: "A nearly zero-emission building #NZEB has a very high energy performance.,” 2022. https://twitter.com/Energy4Europe/status/1481321197547794434 (accessed Sep. 21, 2022).

[13] T. Karlessi *et al.*, “The Concept of Smart and NZEB Buildings and the Integrated Design Approach,” in *Procedia Engineering*, 2017, vol. 180, pp. 1316–1325. doi: 10.1016/j.proeng.2017.04.294.

[14] J. Glasco, “Smart Buildings and Carbon Neutrality: A Race Against Time,” 2021. [Online]. Available: www.smartcitiesworld.net

[15] “SmartHome / ITI,” 2022. https://smarthome.iti.gr/ (accessed Sep. 13, 2022).

[16] A. I. Salamanis *et al.*, “Benchmark comparison of analytical, data-based and hybrid models for multi-step short-term photovoltaic power generation forecasting,” *Energies (Basel)*, vol. 13, no. 22, Nov. 2020, doi: 10.3390/en13225978.

[17] X. Xie, Q. Lu, M. Herrera, Q. Yu, A. K. Parlikad, and J. M. Schooling, “Does historical data still count? Exploring the applicability of smart building applications in the post-pandemic period,” *Sustain Cities Soc*, vol. 69, Jun. 2021, doi: 10.1016/j.scs.2021.102804.

[18] D. L. Marino, K. Amarasinghe, and M. Manic, “Building energy load forecasting using Deep Neural Networks,” in *IECON Proceedings (Industrial Electronics Conference)*, Dec. 2016, pp. 7046–7051. doi: 10.1109/IECON.2016.7793413.

[19] A. Wahab, M. A. Tahir, N. Iqbal, A. Ul-Hasan, F. Shafait, and S. M. Raza Kazmi, “A Novel Technique for Short-Term Load Forecasting Using Sequential Models and Feature Engineering,” *IEEE Access*, vol. 9, pp. 96221–96232, 2021, doi: 10.1109/ACCESS.2021.3093481.

[20] Y. H. Hsiao, “Household electricity demand forecast based on context information and user daily schedule analysis from meter data,” *IEEE Trans Industr Inform*, vol. 11, no. 1, pp. 33–43, Feb. 2015, doi: 10.1109/TII.2014.2363584.

[21] P. Koukaras, N. Bezas, P. Gkaidatzis, D. Ioannidis, D. Tzovaras, and C. Tjortjis, “Introducing a novel approach in one-step ahead energy load forecasting,”

Sustainable Computing: Informatics and Systems, vol. 32, Dec. 2021, doi: 10.1016/j.suscom.2021.100616.

[22] T. Hou *et al.*, "A novel short-term residential electric load forecasting method based on adaptive load aggregation and deep learning algorithms," *Energies (Basel)*, vol. 14, no. 22, Nov. 2021, doi: 10.3390/en14227820.

[23] M. Alamaniotis, D. Bargiotas, and L. H. Tsoukalas, "Towards smart energy systems: application of kernel machine regression for medium term electricity load forecasting," *Springerplus*, vol. 5, no. 1, pp. 1–15, Dec. 2016, doi: 10.1186/s40064-016-1665-z.

[24] A. I. Arvanitidis, D. Bargiotas, A. Daskalopulu, D. Kontogiannis, I. P. Panapakidis, and L. H. Tsoukalas, "Clustering Informed MLP Models for Fast and Accurate Short-Term Load Forecasting," *Energies (Basel)*, vol. 15, no. 4, Feb. 2022, doi: 10.3390/en15041295.

[25] A. M. Pirbazari, E. Sharma, A. Chakravorty, W. Elmenreich, and C. Rong, "An Ensemble Approach for Multi-Step Ahead Energy Forecasting of Household Communities," *IEEE Access*, vol. 9, pp. 36218–36240, 2021, doi: 10.1109/ACCESS.2021.3063066.

[26] P. S. G. de Mattos Neto *et al.*, "Energy consumption forecasting for smart meters using extreme learning machine ensemble," *Sensors*, vol. 21, no. 23, Dec. 2021, doi: 10.3390/s21238096.

[27] M. J. A. Shohan, M. O. Faruque, and S. Y. Foo, "Forecasting of Electric Load Using a Hybrid LSTM-Neural Prophet Model," *Energies (Basel)*, vol. 15, no. 6, Mar. 2022, doi: 10.3390/en15062158.

[28] G. Mohi Ud Din and A. K. Marnerides, "Short Term Power Load Forecasting Using Deep Neural Networks," in *2017 International Conference on Computing, Networking and Communications (ICNC)*, 2017. doi: 10.1109/ICCNC.2017.7876196.

[29] C. Bennett, R. A. Stewart, and J. Lu, "Autoregressive with exogenous variables and neural network short-term load forecast models for residential low voltage distribution networks," *Energies (Basel)*, vol. 7, no. 5, pp. 2938–2960, 2014, doi: 10.3390/en7052938.

[30] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, "Short-Term Residential Load Forecasting Based on LSTM Recurrent Neural Network," *IEEE*

Trans Smart Grid, vol. 10, no. 1, pp. 841–851, Jan. 2019, doi: 10.1109/TSG.2017.2753802.

[31] S. Bouktif, A. Fiaz, A. Ouni, and M. A. Serhani, "Optimal deep learning LSTM model for electric load forecasting using feature selection and genetic algorithm: Comparison with machine learning approaches," *Energies (Basel)*, vol. 11, no. 7, 2018, doi: 10.3390/en11071636.

[32] M. A. Khairalla, X. Ning, N. T. AL-Jallad, and M. O. El-Faroug, "Short-Term Forecasting for Energy Consumption through Stacking Heterogeneous Ensemble Learning Model," *Energies (Basel)*, vol. 11, no. 6, Jun. 2018, doi: 10.3390/en11061605.

[33] C. Kuster, Y. Rezgui, and M. Mourshed, "Electrical load forecasting models: A critical systematic review," *Sustainable Cities and Society*, vol. 35. Elsevier Ltd, pp. 257–270, Nov. 01, 2017. doi: 10.1016/j.scs.2017.08.009.

[34] N. Bezas, C. Timplalexis, A. Salamanis, V. Karapatsias, D. Ioannidis, and D. Kehagias, "Novel feature extraction and model retraining techniques for short-term and day-ahead residential load forecasting," 2021.

[35] N. Kohzadi, M. S. Boyd, B. Kermanshahi, and I. Kaastra, "A comparison of artificial neural network and time series models for forecasting commodity prices," *Neurocomputing*, vol. 10, pp. 169–181, 1996.

[36] A. Kumar Dubey, A. Kumar, V. García-Díaz, A. Kumar Sharma, and K. Kanhaiya, "Study and analysis of SARIMA and LSTM in forecasting time series data," *Sustainable Energy Technologies and Assessments*, vol. 47, Oct. 2021, doi: 10.1016/j.seta.2021.101474.

[37] S. Masum, Y. Liu, and J. Chiverton, "Multi-step Time Series Forecasting of Electric Load using Machine Learning Models," in *Proc. ICAISC, 2018*, 2018, pp. 158–159.

[38] S. Papadopoulos and I. Karakatsanis, "Short-term electricity load forecasting using time series and ensemble learning methods," in *2015 IEEE Power and Energy Conference at Illinois, PECI 2015*, Mar. 2015. doi: 10.1109/PECI.2015.7064913.

[39] L. Menculini *et al.*, "Comparing Prophet and Deep Learning to ARIMA in Forecasting Wholesale Food Prices," *Forecasting*, vol. 3, no. 3, pp. 644–662, Sep. 2021, doi: 10.3390/forecast3030040.

[40] S. Siami-Namini, N. Tavakoli, and A. Siami Namin, “A Comparison of ARIMA and LSTM in Forecasting Time Series,” in *Proceedings - 17th IEEE International Conference on Machine Learning and Applications, ICMLA 2018*, Jan. 2019, pp. 1394–1401. doi: 10.1109/ICMLA.2018.00227.

[41] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, “Statistical and Machine Learning forecasting methods: Concerns and ways forward,” *PLoS One*, vol. 13, no. 3, Mar. 2018, doi: 10.1371/journal.pone.0194889.

[42] M. D. C. Ruiz-Abellón, A. Gabaldón, and A. Guillamón, “Load forecasting for a campus university using ensemble methods based on regression trees,” *Energies (Basel)*, vol. 11, no. 8, 2018, doi: 10.3390/en11082038.

[43] C. Fan, F. Xiao, and S. Wang, “Development of prediction models for next-day building energy consumption and peak power demand using data mining techniques,” *Appl Energy*, vol. 127, pp. 1–10, Aug. 2014, doi: 10.1016/j.apenergy.2014.04.016.

[44] J. Gastinger, S. Nicolas, D. Stepić, M. Schmidt, and A. Schülke, “A study on Ensemble Learning for Time Series Forecasting and the need for Meta-Learning,” Apr. 2021, [Online]. Available: http://arxiv.org/abs/2104.11475

[45] J. Brownlee, “A Gentle Introduction to Ensemble Learning Algorithms,” 2021. https://machinelearningmastery.com/tour-of-ensemble-learning-algorithms/ (accessed Sep. 28, 2022).

[46] L. Breiman, “Random Forests,” *Mach Learn*, vol. 45, pp. 5–32, 2001.

[47] J. H. Friedman, “Stochastic gradient boosting,” 2002. [Online]. Available: www.elsevier.com/locate/csda

[48] Y. Wang, “Short-term Power Load Forecasting Based on Machine Learning,” KTH ROYAL INSTITUTE OF TECHNOLOGY, 2019.

[49] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2016, vol. 13-17-August-2016, pp. 785–794. doi: 10.1145/2939672.2939785.

[50] M. Adnan, A. A. S. Alarood, M. I. Uddin, and I. ur Rehman, “Utilizing grid search cross-validation with adaptive boosting for augmenting performance of machine learning models,” *PeerJ Comput Sci*, vol. 8, 2022, doi: 10.7717/PEERJ-CS.803.

[51] G. Ke *et al.*, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 3156–3154. Accessed: Sep. 28, 2022. [Online]. Available: http://papers.nips.cc/paper/6907-lightgbma- highly-efficient-gradient-boosting-decisiontree. pdf.

[52] Y. Ju, G. Sun, Q. Chen, M. Zhang, H. Zhu, and M. U. Rehman, "A model combining convolutional neural network and lightgbm algorithm for ultra-short-term wind power forecasting," *IEEE Access*, vol. 7, pp. 28309–28318, 2019, doi: 10.1109/ACCESS.2019.2901920.

[53] A. V. Dorogush, V. Ershov, and A. Gulin, "CatBoost: gradient boosting with categorical features support," Oct. 2018, [Online]. Available: http://arxiv.org/abs/1810.11363

[54] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "CatBoost: unbiased boosting with categorical features," Jun. 2017, [Online]. Available: http://arxiv.org/abs/1706.09516

[55] T. L. Fine, "Fundamentals of Artificial Neural Networks [Book Reviews]," *IEEE Trans Inf Theory*, vol. 42, no. 4, p. 1322, Apr. 2005, doi: 10.1109/tit.1996.508868.

[56] O. Surakhi *et al.*, "Time-lag selection for time-series forecasting using neural network and heuristic algorithm," *Electronics (Switzerland)*, vol. 10, no. 20, Oct. 2021, doi: 10.3390/electronics10202518.

[57] A. Nyandwi and D. Kumar, "Neural Network Approach to Short and Long Term Load Forecasting Using Weather Conditioning," in *ICE3-2020 : International Conference on Electrical and Electronics Engineering : February 14-15, 2020*, 2020.

[58] J. Zheng, C. Xu, Z. Zhang, and X. Li, "Electric Load Forecasting in Smart Grid Using Long-Short-Term-Memory based Recurrent Neural Network," in *51st Annual Conference on Information Sciences and Systems (CISS)*, 2017. doi: 10.1109/CISS.2017.7926112.

[59] C. Chen *et al.*, "Deep Learning on Computational-Resource-Limited Platforms: A Survey," *Mobile Information Systems*, vol. 2020. Hindawi Limited, 2020. doi: 10.1155/2020/8454327.

[60] F. Murtagh, "Neurocomputing 2 (1990/1991) 183/197," *Neurocomputing 2*, vol. 5, no. 6, pp. 183–197, 1991, doi: https://doi.org/10.1016/0925-2312 (91)90023-5.

[61] S. Siami-Namini, N. Tavakoli, and A. S. Namin, *The Performance of LSTM and BiLSTM in Forecasting Time Series; The Performance of LSTM and BiLSTM in Forecasting Time Series*. 2019.

[62] S. J. Hochreiter and J. Schmidhuber, *"Long short-term memory", Neural computation,* No. 8., vol. 9. 1997.

[63] A. A. Peñaloza, R. C. Leborgne, and A. Balbinot, "Comparative Analysis of Residential Load Forecasting with Different Levels of Aggregation," in *The 8th International Conference on Time Series and Forecasting*, Jun. 2022, p. 29. doi: 10.3390/engproc2022018029.

[64] Y. Wang, N. Zhang, and X. Chen, "A short-term residential load forecasting model based on lstm recurrent neural network considering weather features," *Energies (Basel)*, vol. 14, no. 10, May 2020, doi: 10.3390/en14102737.

[65] H. Zang *et al.*, "Residential load forecasting based on LSTM fusing self-attention mechanism with pooling," *Energy*, vol. 229, Aug. 2021, doi: 10.1016/j.energy.2021.120682.

[66] L. Han, Y. Peng, Y. Li, B. Yong, Q. Zhou, and L. Shu, "Enhanced deep networks for short-term and medium-term load forecasting," *IEEE Access*, vol. 7, pp. 4045–4055, 2019, doi: 10.1109/ACCESS.2018.2888978.

[67] S. Bates, T. Hastie, and R. Tibshirani, "Cross-validation: what does it estimate and how well does it do it?," Apr. 2021, [Online]. Available: http://arxiv.org/abs/2104.00673

[68] S. V. Venishetty, H. Kusetogullari, and A. Boone, "Forecasting Sales of Truck Components: A Machine Learning Approach," in *IEEE 10th International Conference on Intelligent Systems*, 2020.

[69] Z. Wang, Y. Wang, and R. S. Srinivasan, "A novel ensemble learning approach to support building energy use prediction," *Energy Build*, vol. 159, pp. 109–122, Jan. 2018, doi: 10.1016/j.enbuild.2017.10.085.

[70] X. Liao, N. Cao, M. Li, and X. Kang, "Research on Short-Term Load Forecasting Using XGBoost Based on Similar Days," in *Proceedings - 2019 International Conference on Intelligent Transportation, Big Data and Smart City, ICITBS 2019*, Mar. 2019, pp. 675–678. doi: 10.1109/ICITBS.2019.00167.

[71] S. Papadopoulos and I. Karakatsanis, “Short-term electricity load forecasting using time series and ensemble learning methods,” in *2015 IEEE Power and Energy Conference at Illinois, PECI 2015*, Mar. 2015. doi: 10.1109/PECI.2015.7064913.

[72] C. Fan, F. Xiao, and S. Wang, “Development of prediction models for next-day building energy consumption and peak power demand using data mining techniques,” *Appl Energy*, vol. 127, pp. 1–10, Aug. 2014, doi: 10.1016/j.apenergy.2014.04.016.

[73] L. Zyglakis *et al.*, “Greek Smart House Nanogrid Dataset,” *Zenodo*, 2020. https://explore.openaire.eu/search/dataset?pid=10.5281%2Fzenodo.4246525 (accessed Sep. 26, 2022).

[74] E. Spiliotis, V. Assimakopoulos, S. Makridakis, and V. Assimakopoulos, “The M5 Accuracy competition: Results, findings and conclusions,” *Int J Forecast*, vol. 38, no. 4, Oct. 2020, doi: https://doi.org/10.1016/j.ijforecast.2021.11.013.

[75] L. Buitinck *et al.*, “API design for machine learning software: experiences from the scikit-learn project,” Sep. 2013, [Online]. Available: http://arxiv.org/abs/1309.0238

[76] Z. Ma, P. Wang, Z. Gao, R. Wang, and K. Khalighi, “Ensemble of machine learning algorithms using the stacked generalization approach to estimate the warfarin dose,” *PLoS One*, vol. 13, no. 10, Oct. 2018, doi: 10.1371/journal.pone.0205872.

[77] X. Luo *et al.*, “Short-Term Wind Speed Forecasting via Stacked Extreme Extreme Learning Machine With Generalized Correntropy,” *EEE Trans Ind Inf 2018*, vol. 14, no. 11, 2018, doi: https://doi.org/10.1109/TII.2018.2854549.

[78] Z. Ma and Q. Dai, “Selected an Stacking ELMs for Time Series Prediction,” *Neural Process Lett*, vol. 44, no. 3, pp. 831–856, Dec. 2016, doi: 10.1007/s11063-016-9499-9.

[79] P. P. Phyo, Y. C. Byun, and N. Park, “Short-Term Energy Forecasting Using Machine-Learning-Based Ensemble Voting Regression,” *Symmetry (Basel)*, vol. 14, no. 1, Jan. 2022, doi: 10.3390/sym14010160.

[80] Z. Zhao *et al.*, “Short-Term Load Forecasting Based on the Transformer Model,” *Information (Switzerland)*, vol. 12, no. 12, Dec. 2021, doi: 10.3390/INFO12120516.

[81] J. Moon, Y. Kim, M. Son, and E. Hwang, “Hybrid short-term load forecasting scheme using random forest and multilayer perceptron,” *Energies (Basel)*, vol. 11, no. 12, Dec. 2018, doi: 10.3390/en11123283.

[82] P. W. Khan and Y. C. Byun, “Adaptive Error Curve Learning Ensemble Model for Improving Energy Consumption Forecasting,” *Computers, Materials and Continua*, vol. 69, no. 2, pp. 1893–1913, 2021, doi: 10.32604/cmc.2021.018523.

[83] M. Massaoudi, S. S. Refaat, I. Chihi, M. Trabelsi, F. S. Oueslati, and H. Abu-Rub, “A novel stacked generalization ensemble-based hybrid LGBM-XGB-MLP model for Short-Term Load Forecasting,” *Energy*, vol. 214, Jan. 2021, doi: 10.1016/j.energy.2020.118874.

[84] C. Miller *et al.*, “The Building Data Genome Project 2, energy meter data from the ASHRAE Great Energy Predictor III competition,” *Sci Data*, vol. 7, no. 1, Dec. 2020, doi: 10.1038/s41597-020-00712-x.

Appendices

Appendix A: Dataset Samples

ITI Smart House: Instance of raw weather-related data set

	Timestamp_UTC	Temperature_v1	Relative_Humidity_v1	Wind_Speed_v1	Clouds_v1	Clouds_low	Clouds_mid	Clouds_high	Temperature_v2
0	2018-10-03T21:00:00Z	20.6	74.0	3.60	25.0	NaN	NaN	NaN	NaN
1	2018-10-03T21:15:00Z	20.6	74.0	3.60	0.0	NaN	NaN	NaN	NaN
2	2018-10-03T21:30:00Z	20.6	74.0	3.60	25.0	NaN	NaN	NaN	NaN
3	2018-10-03T21:45:00Z	20.6	70.0	3.60	25.0	NaN	NaN	NaN	NaN
4	2018-10-03T22:00:00Z	22.2	32.0	5.40	25.0	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...
64902	2020-09-30T19:45:00Z	18.9	48.0	1.34	100.0	NaN	NaN	NaN	18.59
64903	2020-09-30T20:00:00Z	18.9	48.0	1.34	100.0	NaN	NaN	NaN	18.44
64904	2020-09-30T20:15:00Z	18.3	50.0	0.45	0.0	NaN	NaN	NaN	18.30
64905	2020-09-30T20:30:00Z	18.3	50.0	0.45	0.0	NaN	NaN	NaN	18.17
64906	2020-09-30T20:45:00Z	18.3	50.0	0.45	0.0	NaN	NaN	NaN	18.10

64907 rows × 14 columns

ITI Smart House: Instance of Energy load and Energy generation from PV dataset

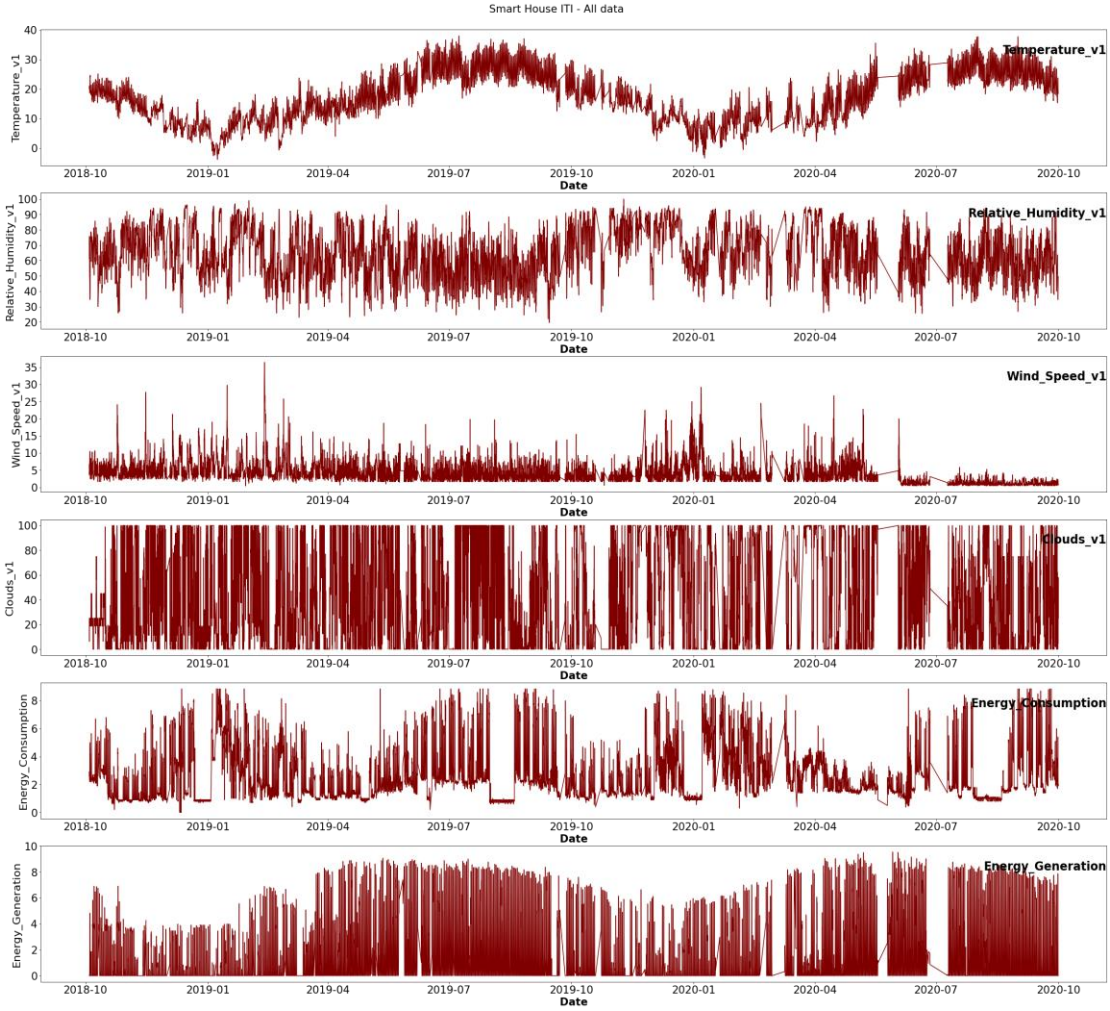
	Timestamp_UTC	Energy_Consumption
0	2018-10-03T21:00:00Z	0.8
1	2018-10-03T21:15:00Z	0.7
2	2018-10-03T21:30:00Z	0.7
3	2018-10-03T21:45:00Z	0.7
4	2018-10-03T22:00:00Z	0.6
...	...	...
63179	2020-09-30T19:45:00Z	0.5
63180	2020-09-30T20:00:00Z	0.5
63181	2020-09-30T20:15:00Z	0.5
63182	2020-09-30T20:30:00Z	0.5
63183	2020-09-30T20:45:00Z	0.5

63184 rows × 4 columns

	Timestamp_UTC	Energy_Generation
0	2018-10-03T21:00:00Z	0.0
1	2018-10-03T21:15:00Z	0.0
2	2018-10-03T21:30:00Z	0.0
3	2018-10-03T21:45:00Z	0.0
4	2018-10-03T22:00:00Z	0.0
...	...	...
66478	2020-09-30T19:45:00Z	0.0
66479	2020-09-30T20:00:00Z	0.0
66480	2020-09-30T20:15:00Z	0.0
66481	2020-09-30T20:30:00Z	0.0
66482	2020-09-30T20:45:00Z	0.0

66483 rows × 2 columns

ITI Smart House all hourly time series data (Oct. 2018-Sep. 2020)



BDGP2 (Coleman Building): Instance of Energy load, PV generation and Temperature dataset

	timestamp	Generation	Consumption	Temperature	Hour
timestamp					
2016-01-01 00:00:00	2016-01-01 00:00:00	0.37	92.1202	-14.4	0
2016-01-01 01:00:00	2016-01-01 01:00:00	0.39	87.0000	-13.3	1
2016-01-01 02:00:00	2016-01-01 02:00:00	0.39	85.2501	-15.0	2
2016-01-01 03:00:00	2016-01-01 03:00:00	0.38	83.7502	-12.8	3
2016-01-01 04:00:00	2016-01-01 04:00:00	0.39	83.5001	-14.4	4
...	...	...	...	...	...
2016-12-30 19:00:00	2016-12-30 19:00:00	0.00	97.5000	-6.1	8178
2016-12-30 20:00:00	2016-12-30 20:00:00	0.00	99.5000	-6.1	8179
2016-12-30 21:00:00	2016-12-30 21:00:00	0.00	71.5000	-5.6	8180
2016-12-30 22:00:00	2016-12-30 22:00:00	0.00	73.5000	-5.6	8181
2016-12-30 23:00:00	2016-12-30 23:00:00	0.00	66.5000	-6.7	8182

8183 rows × 5 columns

Appendix B: Scripts for modeling

The Python scripts developed for this thesis are archived in the author's GitHub page:

https://github.com/nikotsal/One-step-ahead-energy-load-forecasting-hybrid-approach-using-meta-ensembling.git