Project

Master in Data Science

Day-Ahead Energy Consumption Forecasting in Academic Campi with Deep Learning Models

Simão Matos Conceição Horta

Leiria, September 2025

Project

Master in Data Science

Day-Ahead Energy consumption Forecasting in Academic Campi with Deep Learning Models

Simão Matos Conceição Horta

Master project developed under the supervision of Professor Carlos Fernando de Almeida Grilo, Professor João Miguel Charrua de Sousa, Professor Luís Miguel de Oliveira Pegado de Noronha e Távora and Professor Pedro José Franco Marques.

Leiria, September de 2025

Statement of Originality and Copyright

This project is original, made only for this purpose, and all authors whose studies and publications were used to complete it are duly acknowledged.

Acknowledgements

I would like to express my gratitude to Professor Carlos Fernando de Almeida Grilo, Professor João Miguel Charrua de Sousa, Professor Luís Miguel de Oliveira Pegado de Noronha Távora and Professor Pedro José Franco Marques. Their provision of unwavering support and critical feedback was instrumental in the directing the project towards excellence. Furthermore, I want to recognise the contribution of all the professors from the master's programme. Their teachings have proven instrumental in the present work, having had a transformative impact on the author's life and career.

I would also like to acknowledge the support from IT – Instituto de Telecomunicações, a research unit funded by national funds through FCT – Fundação para a Ciência e a Tecnologia, I.P. and, when eligible, co funded by EU funds under project/support UID/50008/2025 – Instituto de Telecomunicações, with DOI identifier 10.54499/UID/50008/2025 and LA/P/0109/2020 (DOI: 10.54499/LA/P/0109/2020).

Abstract

Electricity is essential in today's society, with global demand projected to increase 50% by 2050. However, there are significant inefficiencies in power systems throughout production, transmission and distribution. Therefore, to ensure an efficient energy supply it is important to accurately forecast the energy consumption since it allows to deliver only the necessary resources. Advances in data science have provided essential tools to address these challenges by improving the reliability of energy consumption forecasts. Based on these principles, the current project has developed a daily predictive model with a one-day horizon to support energy management decisions, with the goal to optimize the energy efficiency at Campus 2 of the Polytechnic Institute of Leiria (IPL). For this purpose, the historical energy consumption on the campus was used to optimize several models through the *NeuralForecast* framework. Models were developed using only the endogenous consumption variable, as well as models incorporating the exogenous variables Day Type (which distinguishes between weekdays, Saturdays, and Sundays/holidays) and Academic Calendar (which distinguishes between classes, evaluations, school breaks, and vacation periods). The performance of each model was then evaluated using the Mean Absolute Percentage Error (MAPE) and the computational performance of the model was measured by the Total Parameter Count (TPC). The results show that the best-performing model was N-BEATSx trained on both exogenous variables, achieving a MAPE of 4.92% and 806k total number of parameters. However, the model that best balances complexity and performance is the MultiLayer Perceptron (MLP) with both exogenous variables, with only around 51k parameters and a MAPE of 5.57%. In summary, this study developed robust neural networks for energy consumption forecasting, providing both theoretical and practical advances to support decision-making aimed at optimizing energy efficiency.

Key words: Energy forecasting, Neural networks, Time series analysis, Exogenous variables, Predictive modelling, Energy efficiency.

Resumo

A eletricidade é essencial na sociedade atual, com uma procura mundial que se estima crescer 50% até 2050. No entanto, os sistemas de produção, transporte e distribuição de energia apresentam elevadas ineficiências. Para tornar estes sistemas mais eficientes, é necessário prever com precisão o consumo energético para alocar apenas os recursos necessários. Os avanços na ciência de dados têm fornecido ferramentas importantes para enfrentar estes desafios, ao melhorar a fiabilidade das previsões do consumo de eletricidade. Baseado nestes princípios, o presente projeto tem como objetivo desenvolver um modelo preditivo diário com um horizonte de previsão de um dia, que apoie as decisões de gestão de energia com vista a uma maior eficiência energética ao Campus 2 do Instituto Politécnico de Leiria (IPL). Para tal, o histórico do consumo energético foi utilizado para otimizar vários modelos usando a plataforma *NeuralForecast*. Foram desenvolvidos modelos que utilizam apenas a variável endógena de consumo, mas também modelos que incorporam a variável exógena Day Type (que distingue dias úteis, sábados e domingos/feriados) e a variável exógena Academic Calendar (que distingue períodos de aulas, avaliação, interrupção e férias). O desempenho dos modelos foi depois avaliado utilizando o erro percentual absoluto médio (MAPE) e o desempenho computacional do modelo, medido pelo número total de parâmetros (TPC). Os resultados mostram que o modelo com melhor desempenho foi o N-BEATSx treinado com ambas as variáveis exógenas, atingindo um MAPE de 4,92% e 806k parâmetros. No entanto, o modelo que melhor equilibra complexidade e desempenho é o MultiLayer Perceptron (MLP) com ambas as variáveis exógenas, com apenas 51k parâmetros e um MAPE de 5,57%. Em síntese, o presente estudo permitiu desenvolver redes neuronais robustas para a previsão do consumo de energia, fornecendo avanços teóricos e práticos para a tomada de decisões com vista à otimização da eficiência energética.

Palavras-chave: Previsão de energia, Redes neuronais, Análise de séries temporais, Variáveis exógenas, Modelação preditiva, Eficiência energética.

Table of Contents

Figure List

Table List

Abbreviation List

AE	Absolute Error
AI	Artificial Intelligence
ANN	Artificial Neural Network
ARIMA	Autoregressive Integrated Moving Average
CNN	Convolutional Neural Network
CRISP-DM	Cross-Industry Standard Process for Data Mining
CSV	Comma-Separated Values
CUDA	Compute Unified Device Architecture
DST	Daylight-Saving Time
ESTG	Escola Superior de Tecnologia e Gestão
FNN	Feedforward Neural Network
GPU	Graphics Processing Unit
GRN	Gated Residual Networks
GRU	Gated Recurrent Unit
IPL	Polytechnic Institute of Leiria
KANs	Kolmogorov–Arnold Networks
kWh	Kilowatt-hour
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
ML	Machine Learning
MLP	Multilayer Perceptron
MLR	Multiple Linear Regression
MMB	Estimated Model Size in Megabytes
MSE	Mean Squared Error
NBEATS	Neural Basis Expansion Analysis
NHITS	Neural Hierarchical Interpolation for Time Series
PatchTST	Channel-Independent Patch Time Series Transformer
PE	Percentage Absolute Error
RF	Random Forest

RMSE	Root Mean Squared Error
RNN	Recurrent Neural Networks
SARIMA	Seasonal Autoregressive Integrated Moving Average
SD	Standard Deviation
SMAPE	Symmetric Mean Absolute Percentage Error
TCN	Temporal Convolutional Network
TFT	Temporal Fusion Transformer
TiDE	Time-series Dense Encoder
TPC	Total Parameters Count
XGBoost	Extreme Gradient Boost

1. Introduction

Electricity is a crucial element of modern society, powering a wide range of technologies that have greatly enhanced human capabilities in areas such as construction, communication, transportation, health, education and safety. This technology has enabled citizens to be efficient, freeing them from manual labour and giving them more leisure time, thereby improving their standards of living and wellbeing. Consequently, countries that consume more energy tend to be more developed since this is directly linked to industrial activity, infrastructure, and access to essential services, which are key indicators of economic progress and quality of life. Consequently, energy demand has been constantly increasing, and it is predicted that energy consumption will increase by 50% in 2050 [1]. This trend can be explained mainly due to the rapid economic progress, population growth and urbanisation, particularly in developing countries [1].

In order to meet the needs of their growing populations, countries must increase their energy supply and adapt to rising demand. The generation of electricity was predominantly accomplished through the utilisation of fossil fuels. Nowadays alternative energy sources such as hydropower, wind power, solar power and nuclear power, are also greatly employed. From the power plants, the electricity is transmitted through high-voltage power lines across regions and then distributed through lower-voltage lines to homes and factories [2]. However, energy is inevitably lost at every stage of this process. Around 33–40% of energy is lost in thermal power stations, and around 5–10% is lost during transmission and distribution [3]. Another major disadvantage of electricity is that it cannot easily be put in storage for later use. This means that electricity must be consumed the moment it is generated. Consequently, blackouts can occur when insufficient energy is generated at a given moment. On the other hand, if too much energy is generated, the excess electricity must be discarded to prevent equipment damage, or this generation excess must be injected into the grid and sold adopting some negotiation schemes [4].

To minimize all these losses, it is important to promote a more efficient electricity usage. One important way to achieve this is to have access to reliable energy consumption forecasts. Knowing the expected energy demand at a specific period in the future makes it possible to align production and optimise resource allocation [5], enabling energy

systems to become more reliable, efficient and affordable [6]. Data science has become crucial in this respect, providing tools to extract valuable insights from large energy consumption time-series. These insights include the ability to accurately forecast energy consumption which has a direct impact on optimizing the balance between energy demand and supply [7, 8].

1.1 Objectives

The objective of this dissertation is to develop and access a day-ahead predictive models that can accurately forecast the energy consumption of the *campus* 2 of IPL. The current project is built on the work developed by Paulo Oliveira [9], in which a satisfactory outcome was achieved for forecasting the energy consumption in *campus* 2 of IPL. The current project aims to expand and improve the model's accuracy previously achieved by the Paulo' project, keeping the models not depending on exogenous variables, except calendar variables. Ultimately, the objective of both project is to enhance the management decisions at IPL, with a particular focus on the planning of electricity expenses and the formulation of strategies for cost reduction. One of those strategies that are being considered are the installation of solar panels on the roofs of the *Campi* buildings. By knowing these trends, it would be possible to adjust consumption patterns to better match the energy generated by the solar panels. Furthermore, demand flexibility is also strategic when dealing with spot market prices.

1.2 Document Organization

The document is organised in the following structure. The purpose of Chapter 2 is to provide the reader with the necessary background information to facilitate the comprehension of the process of time series forecasting and the characteristics of the models. In Chapter 3, a review of the current literature on the forecasting of time series over both the long and short term is presented. In Chapter 4, the methodology employed, and the metrics used to assess the performance of the models are defined. The fifth chapter provides a comprehensive overview of the procedures employed to the dataset, including the methods of data analysis, treatment, and transformation. The sixth chapter provides a detailed explanation on the characteristics of the modelling process. In Chapter 7, the results obtained from the study are presented and a comparison is drawn between the performance of the models when subjected to different variables. In Chapter 8, the

conclusion of the work is presented, along with a discussion of eventual future improvements.

2. **Background**

This chapter provides a description of key concepts that served as foundation for the present dissertation. This information also guides the reader better understand the overall project and the chosen methodology.

2.1 Time Series

A time series is a set of observations ordered in time and collected in regular intervals [10]. In the Observed plot of Figure 1 it is possible to visualize an example of a time series, which shows the data points that were collected over the years. Time series can be univariate or multivariate. Univariate time series are characterised by the presence of a single time-dependent variable, such as the annual recording of temperature. In contrast, multivariate time series exhibit multiple time-dependent variables, for example the annual recording of weather (e.g. temperatures, humidity, wind speed). The time series data can be decomposed into four key components: the trend, the recurring seasonal effects, the cyclical fluctuations, and random irregular variations. These components can be visualized in Figure 1, in which there is a time series which was not considered for the study. The trend component describes the direction on which the series is evolving. The seasonality component refers to the oscillatory patterns occurring in repetitive intervals, across the series. The cyclic component is related to long duration repetitive patterns that occur in irregular intervals. The residual component represents unpredictable and random fluctuations that do not follow any defined pattern or periodicity [10]. In the Figure 1 it is possible to notice the that the time series shows a general increasing trend with a fluctuation around 2009, and a strong twelve-month seasonal pattern.

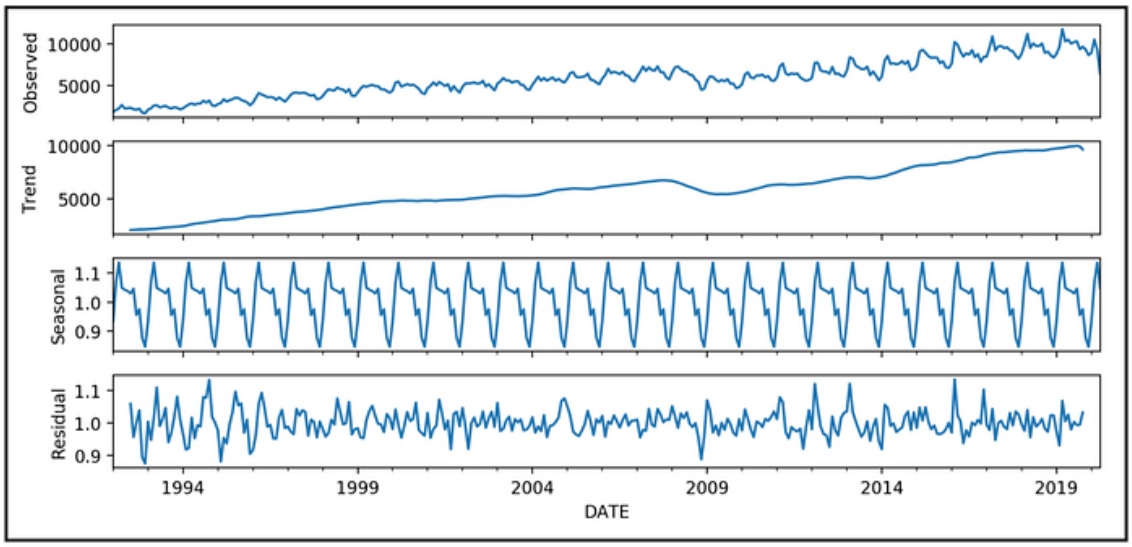


Figure 1: Decomposition of a time series into trend, seasonal, and residual components, along with the original observed series [11].

Analyses like this are often performed to understand how the time series evolved previously. This analysis is often useful because by recognizing patterns from the past, it is also possible to make predictions about the future. The theory of forecasting is based on this assumption that past patterns contain sufficient information to model and predict future values [12]. The goal is not to yield an exact prediction, but rather to improve decision-making in the face of uncertainty. For instance, forecasting is extensively used by the industry to inform decisions related to inventory control, logistics, production and demand management [13].

In the early stages, the development of statistical models, such as ARIMA (Autoregressive Integrated Moving Average) and SARIMA (Seasonal Autoregressive Integrated Moving Average), has enabled the analysis and forecasting of time series with a level of precision that was previously unattainable [14]. ARIMA-based models offer a high interpretability, in addition to the fact that they do not require large datasets to produce reliable forecasts [10]. These models are established on the principles of linear regression and are engineered to detect patterns such as temporal dependencies and noise within the data [15]. The autoregressive component is responsible for the correlations observed with past values, whilst the moving average component is responsible for the modelling of the influence of past errors. In order to address trends or seasonality, the series is differenced, thus rendering it more stable and suitable for analysis.

Previously, simple statistical models have shown habitually superior performance in forecasting tasks when compared to more sophisticated machine learning methodologies.

However, in recent years, these methods have undergone a process of evolution, due to the arrival of innovative algorithms and the accessibility to big datasets that have accelerated the development of these machine learning models [16]. The most significant enhancement that machine models offer is the minimal assumptions they need and their ability to modulate intricate, noisy, and non-linear patterns of more complex time-series. In the next section, a more detailed discussion of the most relevant machine learning models is presented [16].

2.2 Machine Learning

Recent technological and scientific advances have led to the development of artificial intelligence (AI), defined as the capacity of computer systems to execute tasks typically requiring human intelligence [17]. Machine learning (ML) is a subfield of artificial intelligence that focuses on the development of models capable of identifying patterns and deriving meaningful insights from data. In other words, the objective is to develop a model that can learn from historical information and accurately predict unseen data [18]. Based on the learning process employed, ML can be categorised into three distinct groups. Unsupervised learning is a process that exclusively deals with input data that is not categorised or classified. The algorithm leverages the data to reveal hidden patterns, correlations, or structures, frequently resulting in the organisation of the data into clusters based on shared characteristics, with no prior guidance or predefined outcomes. Reinforcement learning is a machine learning process in which an agent learns to make a sequence of decisions by interacting with an environment [19]. After each action, the algorithm is provided with feedback in the form of rewards or penalties, which it uses to adjust its strategy [20]. The objective is to optimise the cumulative reward over time. Reinforcement learning has been demonstrated to be particularly effective in scenarios where the optimal solution is not known in advance and must be achieved through trial and error [19]. Supervised learning is a methodology in which the model is trained to predict specific outcomes by learning from historical data containing known input and output pairs [21]. Several key stages must be completed to achieve this. First, the model needs to be trained using a subset of the labelled dataset, defined as the training dataset. During this phase, it learns to identify patterns and model the relationships between input variables and their associated target values. Once training is complete, the model is evaluated using the test set, which consists of data not known during training. In this

phase, only the input is provided, and the model is tasked to deliver the output [21]. These predictions are then compared to the real output to assess the performance. The model's ability to generalize well to unseen data is a critical indicator of its robustness and applicability in real-world scenarios. Furthermore, hyperparameter tuning is frequently employed as a process to optimise the external configuration settings of a machine learning model. Achieving correct calibration is essential for optimal performance and effective generalisation [21]. Within the supervised learning there are various ML models that use different approaches to achieve high accuracy and strong generalization capabilities. Traditional machine learning models, such as linear regression, support vector machines, and decision trees, rely on manually engineered features and are often effective for structured data and well-defined problems. However, their performance often declines when facing highly nonlinear or high-dimensional datasets. To address these limitations, artificial neural networks have emerged as a powerful alternative, capable of automatically learning hierarchical representations and capturing complex dependencies within the data [22].

2.3 Artificial Neural Networks for Time Series Forecast

To understand the nature of an artificial neural network (ANN), it is necessary to first understand how artificial neurons work [23]. These units are the building blocks of every neural network and can be represented as shown in Figure 2. Overall, these elements operate by accepting a vector x of features as input. This input can be data from the training dataset or the output from a preceding neuron. Next, each input value x_i is multiplied by its corresponding weight w_i and all the weighted inputs are added to yield a weighted sum. The bias b is also added to the weighted sum to adjust the output value, ensuring the model is a best fit for the input data. In the end, an activation function is applied to the weighted sum to determine the output y of the neuron. This output can be either the next neuron input or could be the final output of the neural network [24].

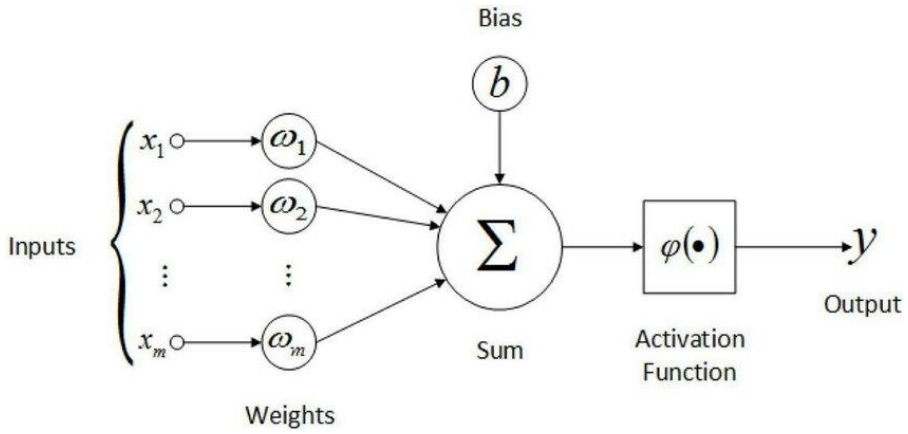


Figure 2: Perceptron overview [25].

Artificial neural networks consist in a set of interconnected artificial neurons organized into a sequence of layers. When these neurons are grouped in one or two layers, between the input and output, the model is referred to as a shallow neural network. In contrast, when ANNs comprise more than two hidden layers of interconnected neurons, some authors classify them as deep neural networks [26]. The increased depth of the network allows it to learn hierarchical representations of data, where each successive layer captures progressively more abstract features. The hierarchical learning structure enhances the model's capacity to detect and model complex, non-linear relationships, being particularly effective for more complex tasks such as image classification, speech recognition, and time series forecasting. However, it is imperative to carefully optimize the number of layers. The addition of excessive layers within a neural network can result in a phenomenon known as overfitting. Overfitting occurs when the network becomes overly reliant on the specific training data, rather than developing the capacity to generalise and recognise patterns that are more widely applicable. This has a detrimental effect on the performance of the system when dealing with unseen data. Conversely, an insufficient number of layers may result in underfitting, where the model is unable to capture the essential structures present in the data. Hyperparameter tuning helps to avoid overfitting and underfitting by adjusting the architectural parameters in order to achieve a balance between model complexity and generalisability.

Deep learning models have been shown to be highly effective in a range of applications. However, it should be noted that these models require substantial quantities of data, considerable computational power, and are often challenging to interpret [27].

2.3.1 Multi-Layer Perceptron

The Multilayer Perceptron (MLP) is one of the most fundamental artificial neural networks in machine learning. In Figure 3, an MLP architecture is represented which contained an input layer, that receives two features, a hidden layer with three neurons, and an output layer that provides a final prediction or classification [28]. The information in this network is transmitted in a single direction from the input layer, through hidden layers, to the output layer. Also, there are no connections between neurons of the same layer. This type of network is defined as a feedforward neural network (FNN) [29]. In the initial stage of the training phase, the input data is fed forward through the network to generate predictions. However, at this stage, the predictions often deviate significantly from the actual target values, as the model's weights are still randomly initialized and do not yet capture the underlying patterns in the data. To quantify the forecast error between predicted and actual values, a *loss function* is calculated. This value is fundamental to the learning process since it guides the *backpropagation algorithm* to calculate the contribution of each neuron to the overall error [30]. The output error is propagated backwards through the network with the aim of computing the gradients for all neurons of the network. Then, the *gradient descent algorithm* is responsible for actually define the new weight values in a direction that reduces the loss function [31]. A single cycle of computing the output of the network (forward propagation), backpropagation, and weight adjustment by gradient descent, is defined as a *batch*. Each batch can process several samples, and the number of processed samples is called the *batch size*. In other words, for each batch, forward propagation and loss calculation occur batch size times. Once completed this phase, the backpropagation is performed once with the cumulative loss function. A complete passage of the batches through the whole training dataset is called an *epoch*. With each batch, the model gradually improves its performance by refining the weights. It is the ability of the network to update weights during the training phase that allows the model to reduce the loss and improve its predictions achieving the task proposed [32].

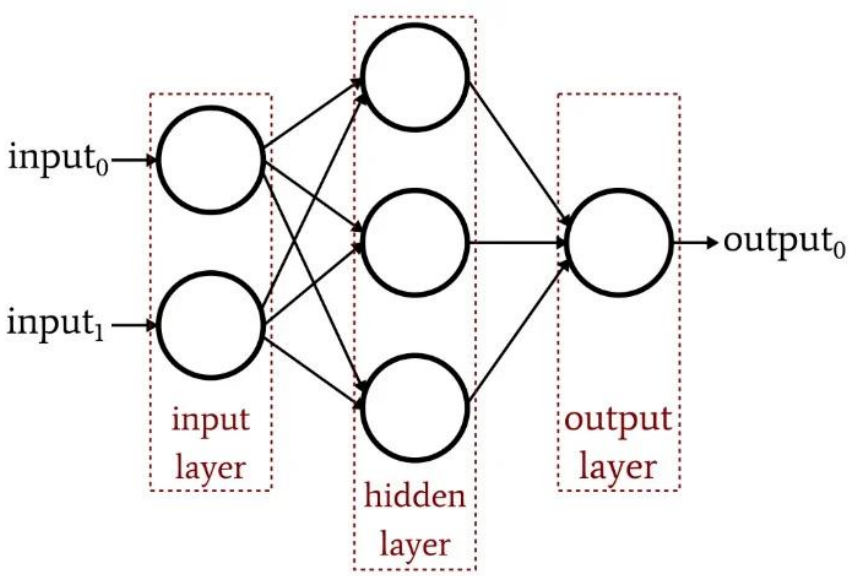


Figure 3: A multilayer feed-forward neural network with one input layer, one hidden layer, and an output layer [33]

2.3.2 Neural Basis Expansion Analysis

Multilayer Perceptron is the foundation of several other artificial neural networks, including the Neural Basis Expansion Analysis (NBEATS). This algorithm is a machine learning model based on MLPs that was originally developed for univariate time series data [34]. Later, Olivares *et al.* developed NBEATSx to address multivariate time series [35]. In summary, the NBEATS models are organised by stacks, and within each stack there are multiple blocks (Figure 4). The blocks are composed by four dense fully connected layers. The output is then directed in two directions. The first MLP returns the backcast and the second MLP returns the forecast. Upon completion of the initial block's inputs processing, the subsequent block is assigned the original input, along with the backcast from the preceding block. The backcast is defined as the model's estimate of the proportion of the input signal that it is capable of explaining; it can thus be subtracted from the model in order to allow the subsequent block to focus exclusively on the residuals that remain. In this manner, each block has the capacity to build upon the corrections of the preceding block, thereby enhancing the overall forecast accuracy [34]. In a similar manner, each stack uses the residuals from the preceding stack, with the objective of enhancing the forecast. The final global forecast is an aggregate of predictions from all stacks that integrate both short-term and long-term patterns from the time series [34].

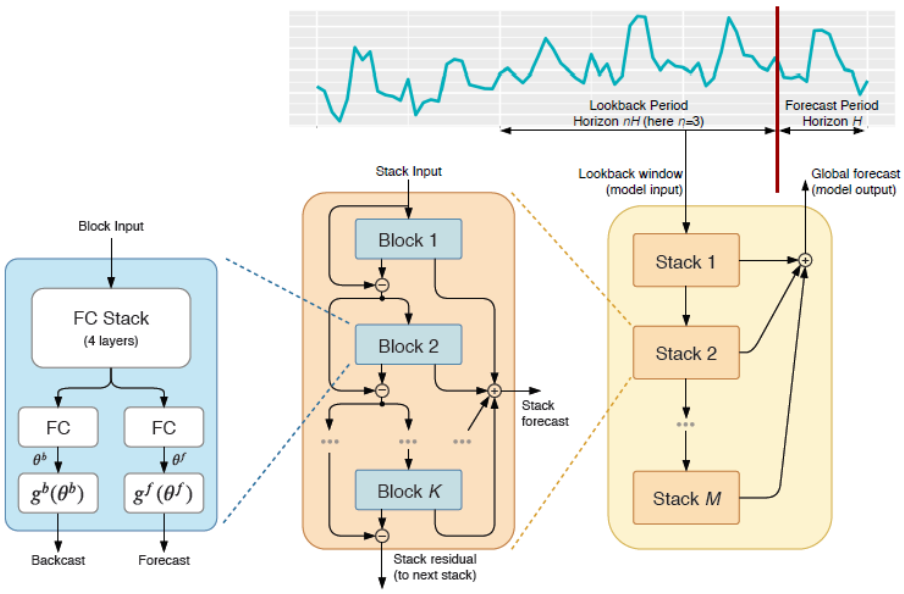


Figure 4: NBEATS architecture overview [35].

2.3.3 Neural Hierarchical Interpolation for Time Series

The Neural Hierarchical Interpolation for Time Series (NHITS) is another MLP-based neural network which is built upon the NBEATS architecture (Figure 5) [34]. The NHITS model incorporates new characteristics such as multi-rate input sampling and hierarchical interpolation [36]. Using these new features, the model can forecast longer horizons from more complex time series and reduce computational costs. The multi-rate input sampling technique was developed to enhance the system's capacity to discern patterns across diverse temporal resolutions. At the block level, rather than feeding the raw time series, the NHITS blocks start with a maximum polling operation that downsamples the input using a kernel [36]. For blocks closer to the raw input, larger kernels are employed to retrieve long-term, low-frequency dependencies. Blocks distant from the raw input use smaller kernels to concentrate more on short-term, high-frequency patterns [36]. As with NBEATS, the backcast structure enables NHITS to progressively refine the forecast by learning complex temporal patterns across different hierarchical levels. Each block is provided with the residuals of the preceding block and attempts to modulate the signal. In the NHITS model, the block uses its input to calculate coefficients that synthesize its backcast output. Next, the new residue for the subsequent block is determined by subtracting this output from the input of the current block. This structure facilitates downstream blocks to focus on signal components that have not yet been modelled [36]. NHITS has been shown to be computationally more efficient than NBEATS because it uses hierarchical interpolation, which allows for a reduction in the number of learnable

parameters. In hierarchical interpolation, NHITS predicts only key values, defined as interpolation coefficients, that are used as input in interpolation functions to generate the full forecast. This approach enables the network to learn with minimal parameters, enhancing generalisation, reducing overfitting, and minimising memory requirements [36].

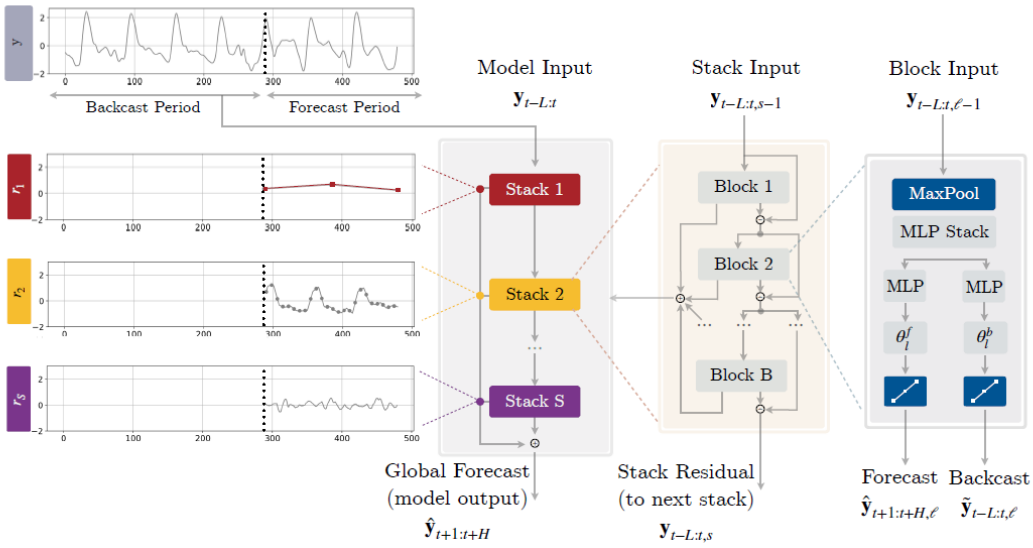


Figure 5: NHITS architecture overview [37].

2.3.4 Time-series Dense Encoder

The Time-series Dense Encoder (TiDE) is based on the MLPs architecture, which has been developed for the purpose of long-term time-series forecasting (Figure 6) [38]. Similar to the previous models, TiDE also employs residual blocks throughout the network to improve accuracy in the predictions. The architecture of the network starts by reducing the dimensionality of the data to improve the computational processing. Then the encoder maps the historical target values and the exogenous variables of a time-series into a dense hidden representation that aims to capture the latent structure and dependencies of the time series [38]. This encoded representation is then passed to the dense decoder, which generates a latent vector for each future time step. Subsequently, the temporal decoder refines each prediction with the future exogenous variables, thereby enabling adaptation to future events [38].

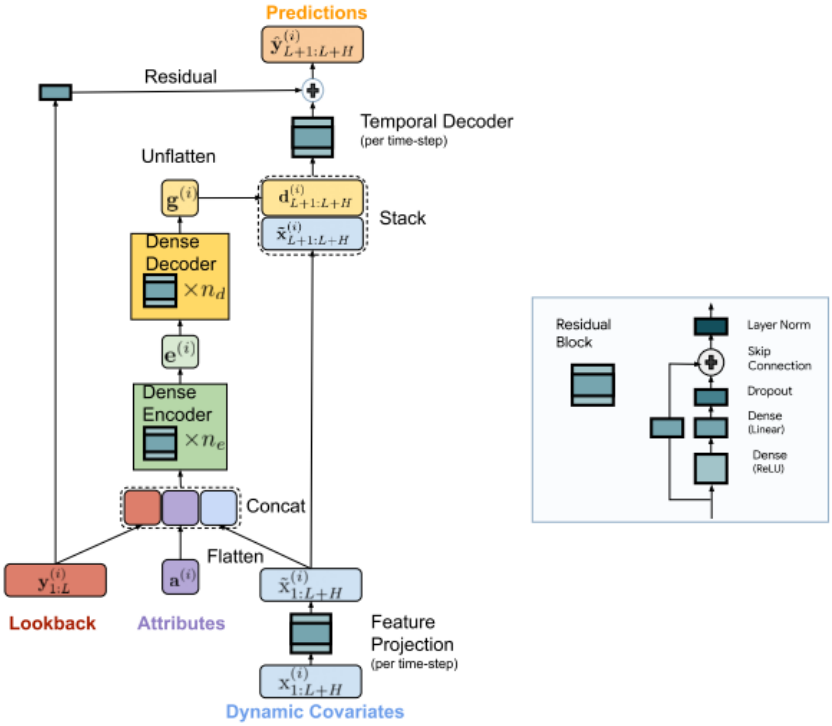


Figure 6: TiDE architecture overview [38].

2.3.6 Temporal Convolutional Network

One of the main limitations of MLP-based architectures is their inefficiency in handling sequential data, as they treat each data sample independently. To address this, Convolutional Neural Network (CNN) can be used. CNNs effectively capture hierarchies and relationships through convolutional operations, making them well-suited for data with spatial structure. CNNs are particularly effective for image processing. In this case, the convolutional algorithms scan small regions of the image at a time using a two-dimensional small array of weights (kernels) across the image's height and width (Figure 7) [39]. These filters compute the dot product between the kernel and the input and results in a feature map that highlights the presence and location of specific features. After the convolutional layers the pooling layers are used to perform dowsampling of a given input and reduce dimensionality. In the end, fully connected layers (MLPs) are used to produce the final output [39].

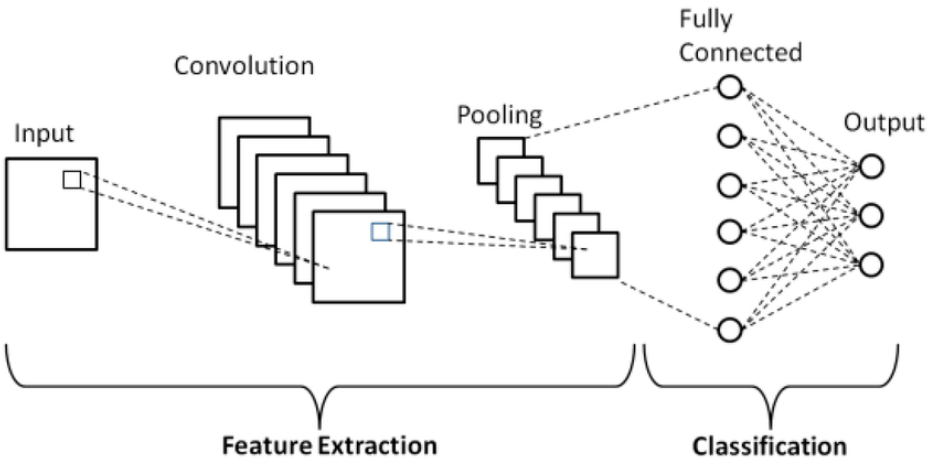


Figure 7: CNN architecture overview [40].

Convolutional Neural Networks have been adapted for time series data, with the Temporal Convolutional Network (TCN) being a notable example (Figure 8). TCNs use convolutional operations to model sequential dependencies while preserving the temporal order [41]. This CNN for time-series functions by extracting a segment of the time-series and employing a one-dimensional kernel, given that time series data are one-dimensional. The filter is confined to movement across the time axis, performing a dot product at each step. The resulting feature map extracts the presence and location of specific patterns (e.g. spikes, trends, seasonality) [41]. The objective is to capture multiple patterns over time, and to this end, the model employs multiple kernels in parallel to detect different temporal characteristics [42]. The TCN characteristics guarantees the prevention of data leakage from the future through the implementation of zero-padding and constrained kernel application. Furthermore, the model incorporates dilated convolutions and residual blocks with the objective of enhancing the model's efficacy [41].

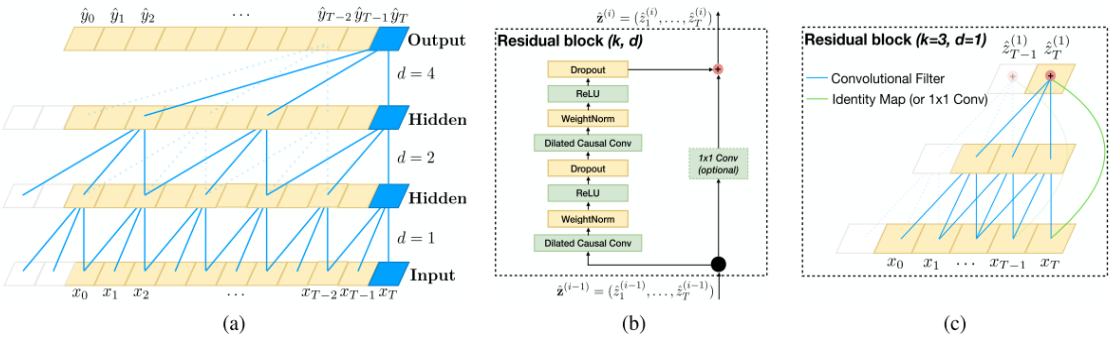


Figure 8: TCN architecture overview [41].

2.3.7 Recurrent Neural Networks based models

Recurrent Neural Networks (RNNs) constitute another type of artificial neural network that has been developed for the purpose of dealing with sequential data and capture temporal dependencies (Figure 9) [43]. To achieve this, the RNN processes data sequentially, retaining and using information from previous steps. The recurrent architecture of these systems maintains a hidden state, which serves as an internal memory, continuously updated with new inputs while preserving information from earlier time steps [44]. As a result, RNNs can learn patterns and dependencies over time, improving the accuracy of their outputs. However, RNNs face the vanishing/exploding gradient problem. This challenge arises because the same weights are reused across multiple time steps along the network. During backpropagation, gradients are repeatedly multiplied, which can cause them to either produce very small or very large updates. As a result, gradient descent struggles to converge to optimal parameters [45, 46].

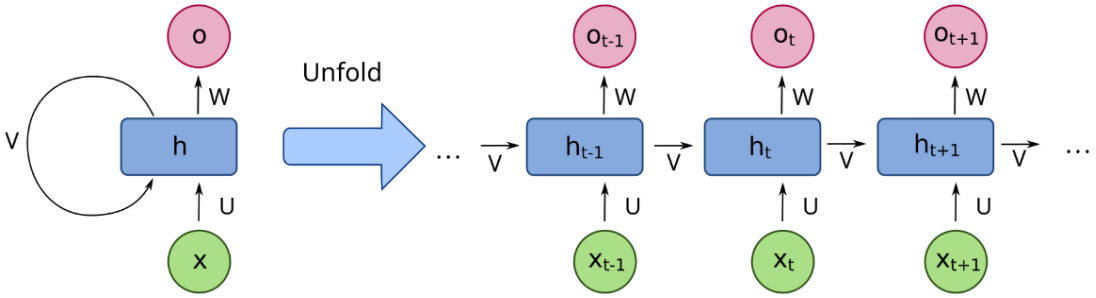


Figure 9: Basic RNN in compact form (left) and expanded form (right) [47].

2.3.8 Long-Short Term Memory (LSTM)

Long Short-Term Memory (LSTM) networks were developed to address the vanishing and exploding gradient issues encountered in conventional RNNs. The LSTMs are an improved version of RNNs that are more efficient in capturing long-term dependencies from sequential data [43]. To achieve this, LSTMs employ a gated mechanism that regulates the process of information retention, modification, or elimination. The LSTM architecture consists of recurrent units, each containing three gates (the forget gate, the input gate, and the output gate) that regulate the flow of information into, within, and out of the cell state (Figure 10) [43]. An LSTM unit takes the input vector, along with its previous hidden state (h_{t-1}) which stores the short-term dependencies information, and cell state (C_{t-1}) which stores the long-term dependencies information. Then, the LSTM unit performs the gate calculations. The forget gate calculates how much of the previous

cell state to keep or discard, based on the previous hidden state and the current input [48]. The input gate determines how much new information to add to the cell state, combining the current input and previous hidden state to update it [49]. Finally, the output gate uses the updated cell state to produce the new hidden state, which is passed to the next time step [48]. The LSTM network processes each element of the input sequence in a sequential manner across a series of units. The configuration of these units into layers enables the network to enhance its capacity to effectively capture complex short-term and long-term dependencies in data [49]. However, despite their advantages, LSTMs can be computationally intensive and slower to train compared to simpler models.

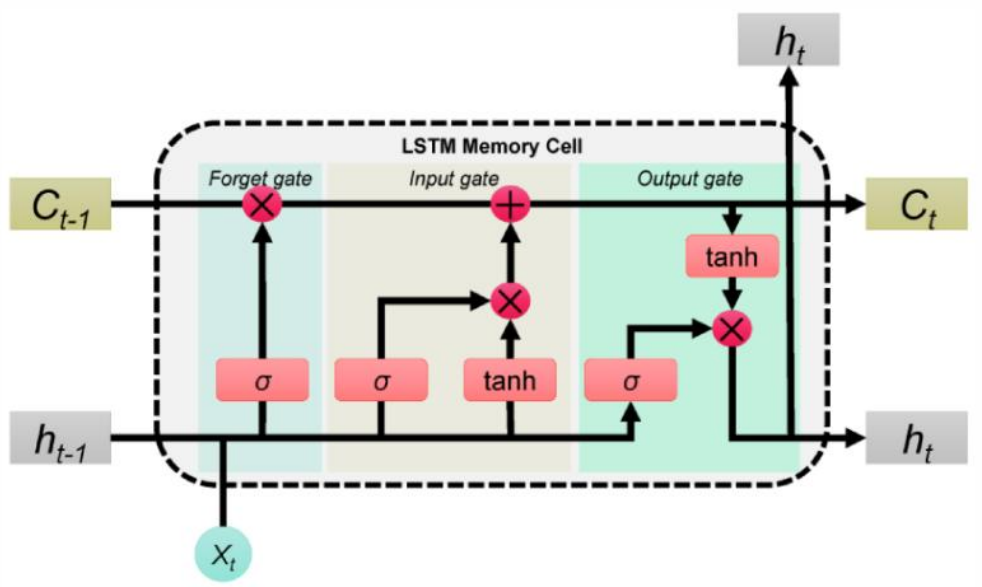


Figure 10: Standard LSTM block [50]

2.3.9 Gated Recurrent Unit (GRU)

The Gated Recurrent Unit (GRU) is a type of recurrent neural network that uses gating mechanisms to better capture long-term dependencies and mitigate the vanishing gradient problem. Additionally, the GRU requires less computational power than LSTM networks, due to its simpler design [49]. GRU combines both short and long-term dependencies information in only one hidden state that is passed between units, making it less memory intensive than LSTMs [49]. In contrast to the three gates used in LSTMs, each GRU unit has only two gates that regulate how the information is updated and remembered (Figure 11). The update gate uses the current input and the previous hidden state to decide how much of the past information to keep versus how much new information to incorporate, acting as a filter balancing old and new data [51]. The reset gate takes the current input and the previous hidden state to determine how much of the past hidden state should be

taken in consideration when computing the new candidate hidden state [51]. The new candidate hidden state is calculated using the current input and a reset-modified version of the previous hidden state [52]. This candidate is then combined with the previous hidden state, where the update gate decides how much of each should be passed on to the next GRU unit. [53]. This mechanism enables the GRU to selectively retain or update information, ensuring that complex temporal dependencies are captured.

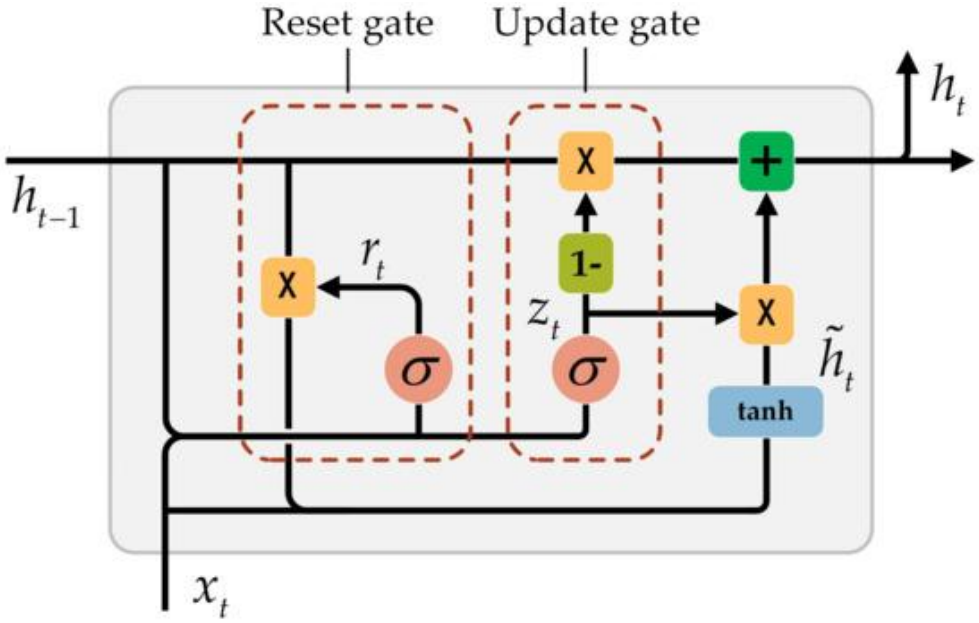


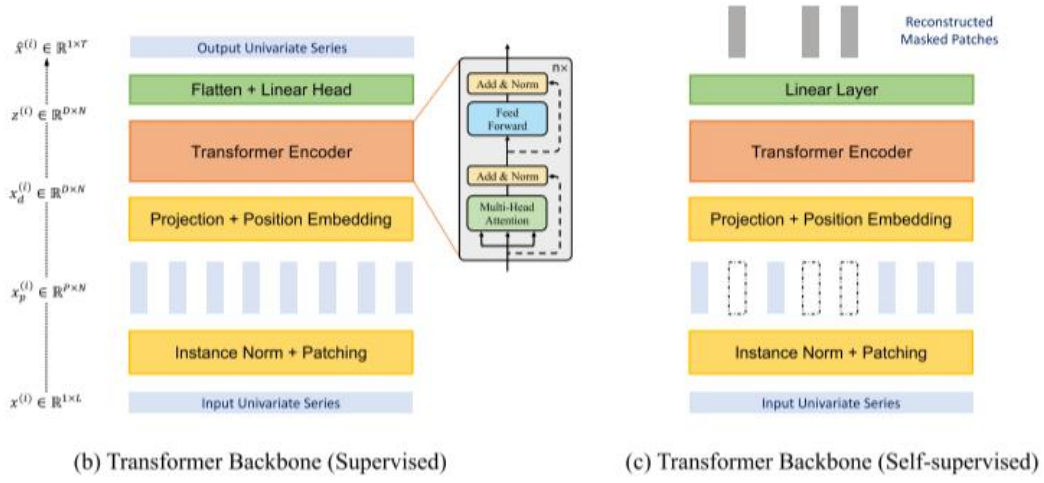
Figure 11: Gated Recurrent Neural Networks architecture overview [54].

2.3.10 Channel-Independent Patch Time Series Transformer

Transformers are another type of artificial neural network that employs a self-attention mechanism to process an entire input sequence at once [55]. This algorithm allows higher parallelization of processes and an improved capability of capturing long-range dependencies. The presence of these features allows this architecture to be more efficient during the train phase and facilitate the capture of long-range dependencies which is a primary challenge in tasks involving the transformation of one sequence into another, such as in translation [55].

The Channel-Independent Patch Time Series Transformer (PatchTST) is an example of a transformer model designed specifically for univariate or multivariate time series forecasting (Figure 12) [56]. PatchTST introduces an innovative approach by operating on patches which are short segments of time series. In summary, each patch is encoded into a high-dimensional embedding that preserves the local temporal structure. These embeddings are then passed through multiple layers of self-attention to model the

relationships between different patches of the time series. This allows the model to focus on relevant historical patterns and dependencies [56]. This patching mechanism enables the model to preserve the local semantic information, which is essential in time series where a single point rarely possesses significance in isolation. It also significantly reduces the computational complexity and memory usage of attention maps, and it enables the model to consider a substantially longer historical context [56].



(b) Transformer Backbone (Supervised) (c) Transformer Backbone (Self-supervised)

Figure 12: PatchTST model overview architecture [56].

2.3.11 Temporal Fusion Transformer

The Temporal Fusion Transformer (TFT) is another transformer-based model specifically tailored for multivariate time series forecasting [57]. In summary, TFT encodes each time step individually into high-dimensional embeddings. These embeddings are then processed using a multi-head attention mechanism, enabling the model to focus selectively on relevant time steps and variables, rather than treating all inputs equally [57]. One of TFT's key innovations is the integration of Gated Residual Networks (GRNs) and gating mechanisms, which help the model control information. It also incorporates recurrent layers for modelling temporal dependencies more effectively [57]. After the attention and gating operations, TFT uses an MLP to further extract patterns and refine the final output [57]. The diagram of the full architecture can be visualised on Figure 13. By combining all these mechanisms on the network, the TFT achieves a balance between capturing long-term dependencies and maintaining flexibility for complex forecasting tasks.

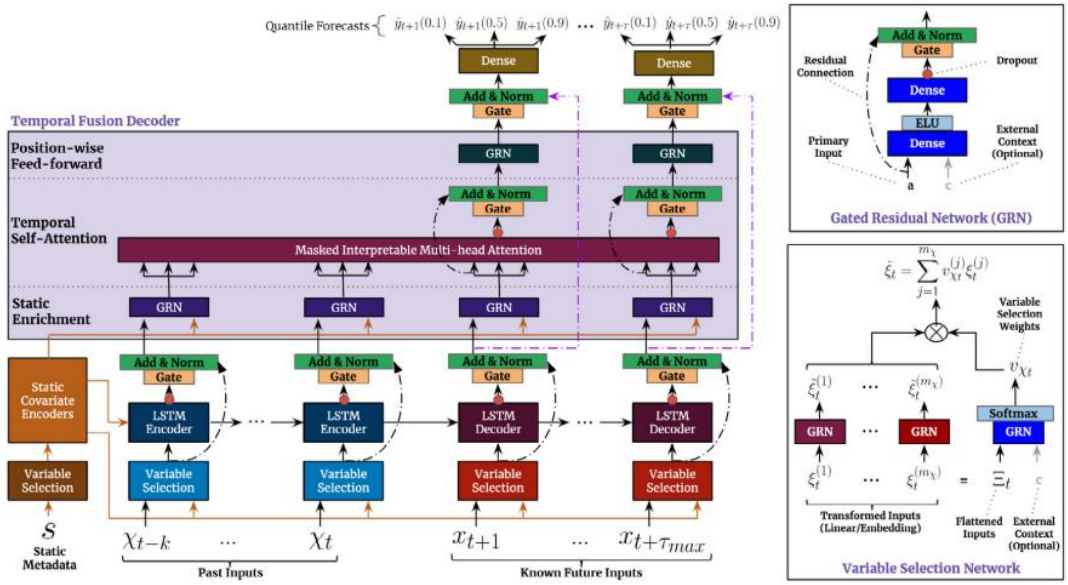


Figure 13: TFT model overview architecture [57].

2.3.12 Kolmogorov–Arnold Networks based Models

Kolmogorov–Arnold Networks (KANs) are a neural network architecture inspired by the Kolmogorov–Arnold Representation Theorem, which states that any continuous multivariate function can be expressed as a finite composition of continuous univariate functions and addition operations [58] .The KAN architecture is fundamentally different from all the previous artificial neural networks since, instead on using fixed activation functions, KAN proposes the learnable activation functions. A learnable univariate function is a single-variable function whose shape is not fixed but is adjusted during training so the model can better fit the data. Each linear weight parameter is replaced by a learnable univariate function, typically parameterized as a B-spline function with its own trainable coefficients, as shown on Figure 14 [59]. In each layer, every node computes the sum of these spline functions applied to its inputs, enabling nonlinear representations. The nodes within a KAN simply sum the input signals arriving through these spline functions, without applying additional nonlinearities at the nodes themselves. This approach transforms traditional weight matrices into collections of learnable functions. These features enable KANs to robustly learn complex nonlinear representations [58, 60].

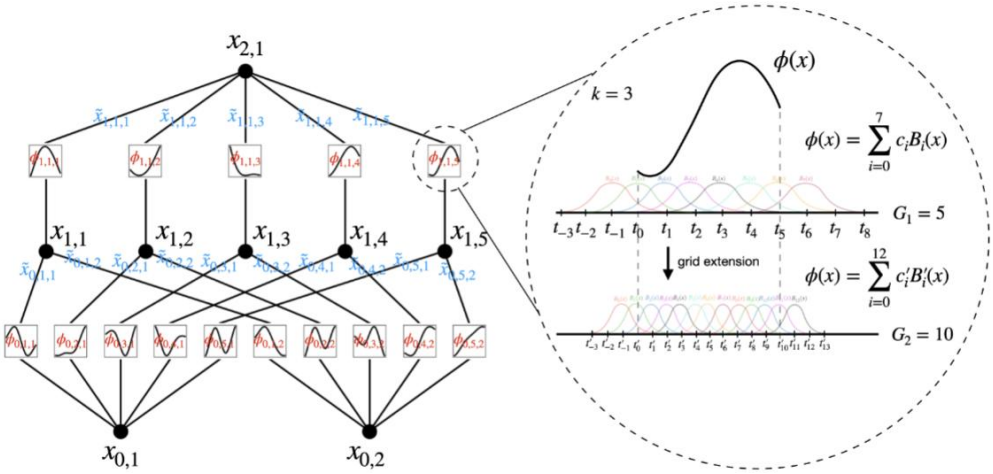


Figure 14: KAN overview network with the activation function parameterized as a B-spline on the right [58].

3. State-of-the-Art

This chapter comprises a review of the most recent literature regarding the forecasting of energy consumption using machine learning models. The studies reviewed in this chapter span over temporal and geographical coverage with the focus on identifying the most common techniques and models employed for medium and short-term forecasting across several spatial scales. In terms of terminology, it will be used the definitions proposed by Soliman [61] .The studies that attempt to predict energy consumption over a time span of more than a year are regarded as long-term forecasts. Models with a forecasting horizon of one week up to a year are classified as medium-term forecasts, and are crucial to support strategic and operational planning, which includes resource allocation, infrastructure maintenance, and policy development. Models with a forecasting horizon of up to one week are designated short-term forecasts and are mostly employed to fine-tune consumption patterns and minimize daily operational cost. The predictive models can also be categorised according to the spatial scale that they attempt to model. The scale of prediction may range from the level of individual buildings to cities, regions and countries [61].

3.1 Medium-Term Forecasts

The literature review on the studies that have employed machine learning models for medium-term energy forecasting showed that the majority focused on predicting the monthly demand [62-66], with some also developing models to forecast the week ahead. These models are generally developed for larger scales, specially applied to countries [62-64]. [66] districts [65]. Other studies attempt to predict the energy consumption only from the residential sector [62] and others focus more on the general electric load of a specific area [63- 66].

With regard to the data and features, the majority of the studies employs a minimum of 15 years of historical energy consumption. This information is frequently the only data source used from which certain studies derive their models [63-64], [66]. However, it is also common practice to add exogenous variables like weather features (e.g. temperature) [62], [65] and calendar data (e.g. holidays and weekends) [65]. A number of other exogenous features have been also documented, although with lower frequency. These

include the purchasing power (e.g. energy prices) [62], and variables associated with population density [65].

A comprehensive review of the existing literature reveals a wide range of models, methodologies and outcomes that have been employed for medium-term energy forecasting. For instance, Dudek *et al.* proposed a new LSTM with residual dilated and exponential smoothing [64]. The time-series used in this study was the monthly electricity consumption of thirty-five European countries. The findings demonstrate the overperformance of the proposed LSTM, with a MAPE of 4.09%, in comparison to the benchmark LSTM and MLP models that yielded MAPE of 6.11% and 5.27%, respectively [64]. These findings align with the results reported previously by Dudek *et al.* (2020) [66]. Using a more contemporary approach, an enhanced N-BEATS model was published by Kasprzyk *et al.* (2024) [63]. The findings of this study demonstrate that N-BEATS attained a good level of performance, demonstrated by its MAPE of 3.78%. In contrast, the MLP and LSTM benchmarks fell short in achieving such precision, with MAPEs of 5.27% and 6.11%, respectively [63]. In another study undertaken by Jung *et al.* (2020), the objective was to develop a deep neural network with transfer learning that could predict the monthly energy consumption for twenty-five districts in Seoul [65]. The time series used was the historical energy demand for the region which was recorded monthly. It is shown that the multiple linear regression (MAPE of 12.07%), random forest (MAPE of 8.02%), the extreme gradient boosting (MAPE of 6.73%) and MLP models (MAPE of 4.32%) were outperformed by MLP with transfer learning, which reached a MAPE of 3.59% [65].

To summarise, the existing literature suggests that, for medium-term forecasts, the development of LSTM hybrid models is the most common approach, since it has been demonstrated to deliver satisfactory results. However, recent developments in transfer learning and the N-BEATS models have emerged as leading performers, demonstrating significant potential to surpass the capabilities of traditional architectures.

3.2 Short-Term Forecasts

The following literature review focuses predominantly on studies that seek to model the following day's forecast. The existing body of research is both extensive and varied, encompassing a broad spectrum of spatial scales.

In this literature review, the studies are usually divided into two categories. The first category comprises large-scale energy demand models, with a particular emphasis on households [67, 68] and microgrids [69]. The second category comprises small-scale models, which are more oriented towards buildings [70-73]. In both categories, the endogenous energy consumption variable is commonly associated with exogenous variables, including weather-related variables such as temperature [69-71], humidity [69, 71], and precipitation [71]. Furthermore, temporal and calendar features are also often used like the hour [69], day of the week [69], and weekends and public holidays [69-71].

The literature review on short-term energy forecasting for households and microgrids shows a range of approaches and outcomes. Alonso *et al.* (2020) used a dataset comprising the energy consumption of 5567 London households measured at half-hour intervals, to develop a LSTM forecasting model [67]. The architecture was designed to generate 24-hour ahead energy consumption prediction. The findings of the study demonstrate that the LSTM model exhibits a 20% enhancement in mean absolute error (MAE) performance in comparison to the conventional ARIMA benchmark models [67]. In another study, Nevil & Rajendra (2021) arrive at similar conclusions using the "SmartMeter Energy Consumption Data in London Households" dataset. A comparison was made of the performance of the SARIMAX (Seasonal ARIMA with exogenous features), the LSTM model and a hybrid model created by a combination of both [68]. The hybrid model attains a good performance, evidenced by the MAPE of 3.058%. In comparison, the SARIMAX and LSTM benchmarks achieve a MAPE of 4.52% and 3.48%, respectively [68]. In a paper published in 2021, Yuan-Jia Ma and Ming-Yue Zhai propose a new hybrid model to forecast the day ahead electrical demand in an energy grid located in Beijing, China [69]. The time series used comprises the electrical consumption from that area from May 2014 to April 2015. The model under consideration was a MLP that was fine-tuned and optimized. Exogenous features like weather data (temperature and humidity), temporal indicators (hour of the day and day of the week), and calendar features (weekend and holiday) were also taken in consideration. The model achieves a high forecasting accuracy with a MAPE of 2.95% [69].

Directly aligned with the objective of the present work, the literature review on short-term energy forecasting for buildings highlights a diverse set of modelling approaches and performance results. In the paper published by Cao *et al.* (2020), the daily energy consumption of Shanghai Tenth People's Hospital is modelled [70]. Besides electricity

load, exogenous features like weather, occupancy and day type (weekend/holiday) are also considered as input. The Random Forest, XGBoost, and Support Vector Regression emerge as the most effective models in this study, achieving a MAPE of 9.64%, 9.81%, and 10.67%, respectively [70].

Deyslen Mariano-Hernández *et al.* (2023) also attempted to develop a model to forecast the energy consumption of two buildings from the University of Valladolid in Spain [71]. The time series used for this study spans from 2016 to 2019, with data recorded at a 15-minute resolution. Additionally, exogenous variables like temperature, humidity, precipitation and calendar data were incorporated into the model. Several models were used to predict the day ahead consumption, namely multiple linear regression (MLR), artificial neural networks (ANN), Random Forest (RF), extreme gradient boost (XGBoost), LSTMs, convolutional neural networks (CNN), temporal convolutional network (TCN) and Temporal Fusion Transformer (TFT). It was observed that, in building 1, the models that returned the lowest MAPE were XGBoost (8.83%), and TCN (9.02%), while in building 2, the lowest MAPE was recorded for CNN (9.59%) and TCN (9.01%) [71].

In another study performed by Oliveira, the forecast accuracy of a proposed multivariate transformer-based model was compared with LSTM and GRU models [72]. The data set under consideration was derived from the historical records of the "Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência" building, encompassing a two-year period and recorded at hourly intervals. The findings demonstrate that the transformer model yielded the best performance, with a MAPE of 7.09%. In comparison, the GRU and LSTM models could not attain MAPEs lower than 11.66% and 10.02%, respectively [72].

The project published by Paulo Oliveira, which was the foundation for the current study, achieved satisfactory results for the prediction of energy consumption in campus 2 of IPL [9]. It was tested several models like the SARIMA, KNN, XGBoost and neural networks like MLP, LSTM and GRU models, but the best performing model was the MLP with 6.86% of MAPE [9]. In this study, exogenous variables related with weather and the day type were incorporated into the model. The findings indicated that temperature did not enhance the accuracy of the models while, exogenous variables associated with weekday information demonstrated a positive impact on model accuracy [9].

To summarise, the studies reviewed here demonstrate that an enhanced accuracy in the forecasting of day-ahead energy consumption in buildings is consistently achieved when exogenous variables such as temperature, humidity, precipitation, occupancy, and calendar data are incorporated. It has been demonstrated that both Random Forest and TCN have exhibited consistent performance across a broad spectrum of contexts. Nevertheless, contemporary architectures, such as transformer-based models, are becoming increasingly accurate and should be taken into consideration, as they may offer enhanced performance. It is also important to recognize that model performance is highly dependent on the profile of consumption and the level of aggregation (e.g., household or district) associated with each particular study. For instance, in the study published by Pallonetto *et al.* (2022), the performances of the LSTM and SVM models were compared in order to predict the following day's energy consumption on a UCD campus in Dublin [73]. It was determined that LSTM demonstrates superior performance when dealing with large volumes of complex and noisy data. On the contrary, the SVM demonstrates superior performance in scenarios where data is limited, and resources are constrained [73]. This reinforces the need for model selection to be guided by data context and constraints rather than a single standard approach. These conclusions also align with the extensive review published by Vivas titled “A Systematic Review of Statistical and Machine Learning Methods for Electrical Power Forecasting with Reported MAPE Score”. Overall, the findings of the literature review demonstrated that the most accurate models are those which [74]:

1. Are based on hybrid configurations, meaning that they integrate new features and/or are the combination of simpler models.
2. Take exogenous variables in consideration. This study highlighted the importance of the occupancy level based on day type (eg: weekday, holiday) for the electricity consumption forecast.
3. Use machine learning methodology which demonstrate the most accurate approach for energy prediction.

4. **Methodology**

In this chapter, it is described how the Cross-Industry Standard Process for Data Mining (CRISP-DM) [75] methodology was used to derive the most precise model for forecasting the day-ahead energy consumption of the campus. The CRISP-DM is a structured procedure that guides data-driven projects ensuring the correct workflow and reproducibility in the development of predictive models. Based on this methodology, the current project developed a version that better suits its needs. The methodology version followed is shown in Figure 15. If in the evaluation phase the models could be further optimized, this version allows additional refinements to be added in the Data Preparation or in the Modelling phase. In the data phase, it could be introduced more information such as exogenous variables that could improve the model's performance. In the modelling phase, approaches such as adjusting of existing hyperparameter ranges or introducing new hyperparameter should be explored.

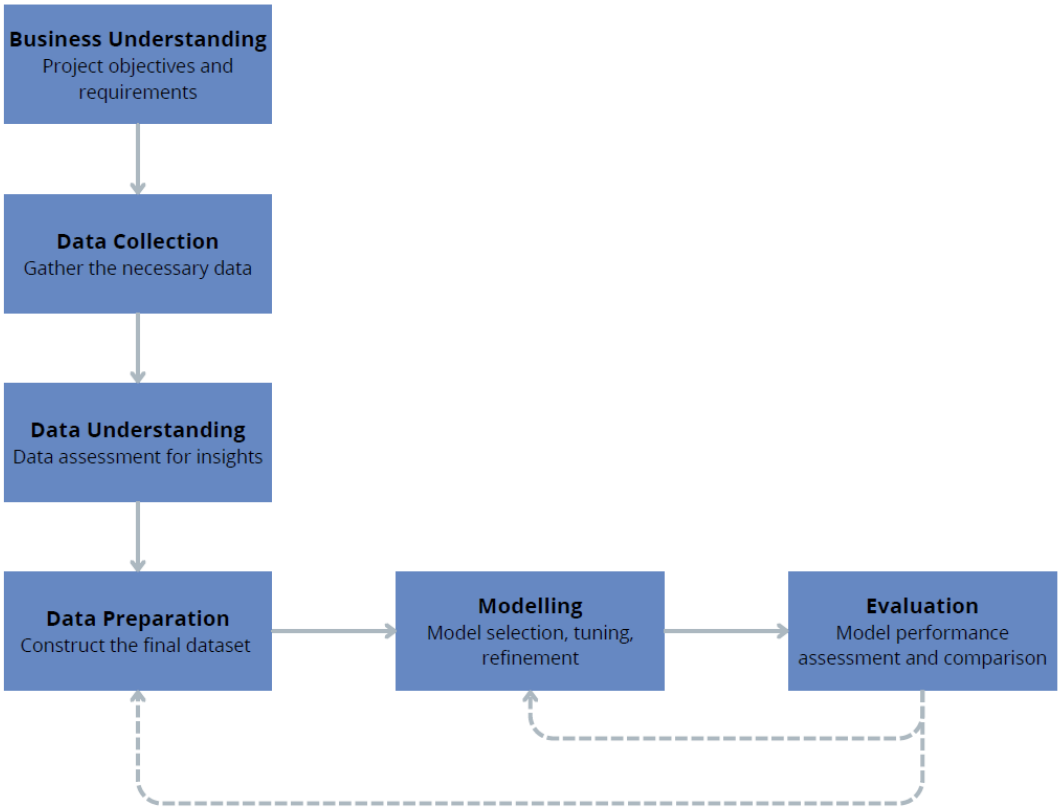


Figure 15: Methodological approach.

4.1 Business Understanding

This initial phase involved identifying the challenges faced by IPL managers in efficiently managing the buildings of the campus and translating these business needs into a machine learning problem. The managers identify the need to forecast the next-day energy consumption to support the daily allocation of resources, thereby improving IPL's energy efficiency. To accomplish this, the development of a forecasting model was proposed, which should be established on the most common practices described in the literature.

4.2 Data Collection, Understanding and Preparation

The data used in this project was provided by the electric energy distributor (E-Redes), removing the necessity for any data collection. In the data understanding phase, the dataset's structure, components, and distribution were examined to gain familiarity with the data and identify potential quality issues, including anomalies, incompleteness, and inconsistencies. Following this assessment, the data preparation phase involved addressing outliers, duplicates, and missing values. To enrich the dataset and support the modelling process, the Day Type exogenous variable was created with the purpose of distinguishing between working days, Saturdays and Sundays, and holidays. The Academic Calendar exogenous variable was also created to distinguish between classes, evaluations, interruption and holiday periods.

4.3 Modelling

In this phase, a range of modelling techniques was selected and applied, with their parameters calibrated to optimal levels. In order to facilitate this process, it was essential to select a unified environment for the assessment of the models' performance that permits the automatic hyperparameter tuning of multiple state-of-the-art forecasting models. To handle the modelling process, the *NeuralForecast*[1] library from Nixtla was the framework chosen [76]. Nixtla is an organization that focus on building state-of-the-art time series forecasting libraries that are easy to use, deliver high performance, and interoperability. On its turn, *NeuralForecast* is one of the most important Nixtla libraries that allows the implementation of deep learning forecasting models for both univariate and multivariate time series [76]. *NeuralForecast* supports a wide range of methods required to develop

[1] https://nixtlaverse.nixtla.io/neuralforecast/docs/getting-started/introduction.html

predictive models, allowing them to be trained on similar conditions. The models are built on *PyTorch* Lightning framework and can be optimized to operate on GPUs, facilitating accelerated training [76]. The *NeuralForecast* package offers a range of models, from which a subset was selected based on insights from the literature review, with the aim of maximising categories' diversity.

Using this platform, the modelling phase starts by the fine-tuning stage which involves optimizing each model's hyperparameters to improve its performance, generalization, and stability. Hyperparameters are external configuration settings of a neural network that can be individually tuned. To tune each model to the time series, the *Optuna* framework was chosen since it is open source, and it is integrated in the *NeuralForecast* environment. This tool optimizes models' hyperparameters by automatically searching for the optimal values within a defined hyperparameter space [77]. The search space was defined individually, focusing on the most important hyperparameters of each model. Another setting defined for all the models was the error metric to be optimized by *Optuna*. The MAE was chosen since it is the default loss applied by *NeuralForecast*. Additionality, it was also necessary to define the horizon as 1 for all the models, since we wanted to forecast the day-ahead. In the end, the optimal hyperparameters for each model are used to train the optimised model.

4.4 Evaluation

The performance of each optimised model is then subjected to evaluation to assess its accuracy and efficiency. This is done using evaluation metrics that measure the discrepancy between predictions and actual values, also defined as errors, allowing for an objective comparison of model accuracy. A range of measures were employed in this study to identify the most suitable model. For instance, it was used the mean absolute error (MAE) that calculates the average error values. The root mean squared error (RMSE), that measures the square root of the average of squared errors, penalizing larger errors, was also used [78]. The mean squared error (MSE), provides a direct measure of the average squared difference between predicted and actual values. While these metrics are useful for assessing model accuracy, they do not allow for an objective comparison across time series with different scales, as they are sensitive to the magnitude of the data. In order to resolve this issue, we also considered the mean absolute percentage error (MAPE) which measures the average percentage error between predicted and actual

values [79, 80]. However, MAPE has the potential to become unstable or undefined when actual values approach zero. To mitigate this, we also computed the Symmetric Mean Absolute Percentage Error (SMAPE) metric, which is bounded between 0% and 200% [80]. Additionally, the coefficient of determination (R^2) is employed to evaluate the proportion of variance in the dependent variable that is predictable from the independent variable [62]. These metrics are not scaled dependent which mean that they are not dependent of the scale of the data. This allows a more objective comparison between models that were applied to time series with different units or magnitudes. In addition to the error-based metrics, the efficiency of the models was also measured using other measures. To calculate the computational efficiency of the models, both the total parameters count (TPC) and the estimated model size in megabytes (MMB) were used. These metrics reflect the model complexity and dimension, thus enabling objective comparison across architectures [80].

5. Exploratory Analysis and Data Preparation

This chapter delineates the processing that the original dataset underwent to ensure its integration as an input to the models. Subsequently, the dataset was subjected to a series of analyses with the aim of familiarising oneself with the time series under study and understanding their patterns and behaviours. This analysis resulted in the creation of new variables, which were designed to enhance the accuracy of the models.

5.1 Data Processing

The dataset was subjected to a series of transformations to reach a suitable configuration for feeding into the models. The dataset was converted from in kilowatts (kW) into kilowatt-hour (kWh) and also from a 15-minute resolution to a daily frequency. Duplicates, outliers and missing values were also addressed.

5.1.1 Data Preparation

The original dataset pertaining to the energy consumption of IPL campus 2 was supplied by E-Redes. It comprised two columns, the 'Activa_kW', which represents the rate at which energy is used (or the active power), measured in kilowatts (kW), and the 'Data_hora', representing the date and time of the recording. The raw dataset provides the active power (kW) recorded at 15-minute intervals from 27/10/2015 at 11:30 to 14/12/2023 at 01:15. In order to predict the following day's energy consumption, it was necessary to convert the active power (kW) into kilowatt-hour (kWh), since this is the measurement unit of the total amount of energy used over a given period of time. The following formula was used: $kWh = kW \times hour\_fraction$. In practice, the "Activa_kW" was multiplied by 0.25, given that the samples were recorded at 15-minute intervals. Consequently, the column entitled "kWh" was created (Table 1). Next, the transformed Comma-Separated Values (CSV) file was read from the local repository and the column entitled "kWh" was assigned to float data type (float64), and the column entitled "Data_hora" was converted into datetime format.

Table 1: Converted dataset with the date and the respective 15-minute resolution consumption recorded in kilowatt-hour.

Data_hora	***kWh***
27/10/2015 11:30	13
27/10/2015 11:45	128.5
27/10/2015 12:00	133.75
...	…
14/12/2023 00:45	45.75
14/12/2023 01:00	43.25
14/12/2023 01:15	42

5.1.2 Data Cleaning

Considering the potential misassigned values on the data collected, it was necessary to undertake an analysis to determine the presence of any missing or duplicate values. The *diff()* function from *Pandas* [2] [81] and *NumPy* [3] [82] was used for this purpose, with the goal to calculate the difference between consecutive values in the "Data_hora" column. If every 15-minute recording is present, the time difference between consecutive entries should be 15 minutes. Nevertheless, this was not the case for the entirety of the dataset. Gaps larger than 15 minutes revealed missing values, while gaps of zero indicated duplicate records.

5.1.3 Duplicates values

To deal with the duplicates, and since the number of values were manageable, it was decided to perform this transformation directly in spreadsheet. All duplicates occurred in October, specifically at 1:00, 1:15, 1:30, and 1:45, which indicated they were caused by daylight-saving time (DST). To correct them, pairs of duplicate values were averaged and reassigned to their respective time slots. For instance, the average of the first two values (1:00 hour and 1:15 hour) was assigned to the 1:00 hour. Next, the average for the 1:30 hour and 1:45 hour was calculated and assigned to the 1:15 hour, and so on. This was done to all the set of duplicates created by DST. The Table 2 shows a sample of the data before and after duplicate values treatment.

[2] https://pandas.pydata.org/
[3] https://numpy.org/

Table 2: Example of duplicate samples treatment. On the left the original dataset, and on the right the treated dataset.

Data_hora	*kWh*
30/10/2016 01:00	49.25
30/10/2016 01:15	56
30/10/2016 01:30	51.25
30/10/2016 01:45	51.75
30/10/2016 01:00	86.75
30/10/2016 01:15	76
30/10/2016 01:30	57
30/10/2016 01:45	50

Data_hora	*kWh*
30/10/2016 01:00	52.625
30/10/2016 01:15	51.5
30/10/2016 01:30	81.375
30/10/2016 01:45	53.5

5.1.4 Outliers

The outliers were visually inspected using a scatter plot (Figure 16). From this observation it was determined that the values below 24 kWh were outliers, since they were noticeably distant from the main cluster of data points. In total, 14 values were identified below this threshold. To handle these outliers, these values were set to NaN. The next section describes how NaN values were handled.

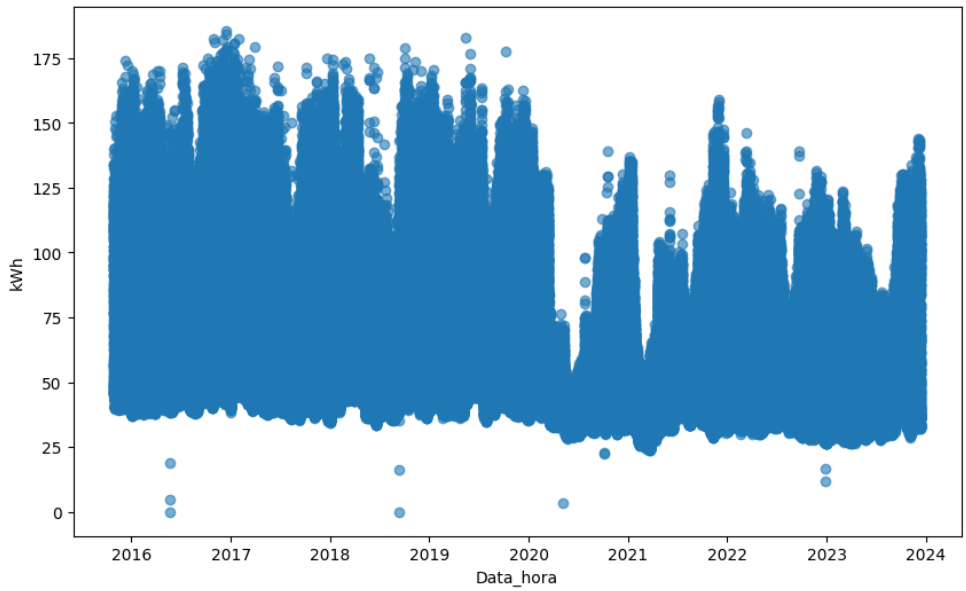


Figure 16: Scatter plot with the original time series. The x-axis represents the dates in years and on the y-axis is represented the energy consumption in kWh.

5.1.5 Missing values

To address missing entries, the Pandas function *date_range* was used to generate a complete sequence of timestamps covering the entire study period, from *2015-10-27*

11:30:00 to *2023-12-14 01:15:00*, with a fixed interval of 15 minutes. This ensured that every possible 15-minute entry within the time span was included, regardless of whether a measurement existed in the original dataset. Using this sequence, a new dataset called "*df_full_range"* was created, containing only the "Data_hora" column with the full set of expected timestamps. Next, the original dataset ("*df"*) was merged with "*df_full_range"* using a left join on the "Data_hora" column. This ensured that all timestamps from "*df_full_range"* were included, while the corresponding kWh values were filled in from "*df"* when available. As a result, a new dataframe was created with a complete set of timestamps for the "Data_hora" variable, with missing energy consumption values in the kWh column now represented as NaN.

Following an analysis of these missing values data, it was determined that these were attributable to either DST or due to the outlier's treatment. These outliers were often next to unusually low kWh readings. To correct this, those suspicious values were also treated as outliers and replaced with NaN. The missing values were estimated using linear regression interpolation between the last and the first known value for that gap. Once both surrounding values were identified, their respective timestamps and kWh values were used to train a linear regression model using *sklearn.linear_model.LinearRegression.* This model then predicted and filled the missing values of the gap. Each missing gap was treated independently, with a new model being trained on its adjacent valid points. This process was repeated until all missing kWh entries were filled. However, the algorithm proved to be incapable of handling the first record of the dataset, since the implementation of linear regression was impeded by the absence of a preceding entry. To deal with this missing value, the first record was set equal to the closest known value, which was the second record.

5.1.6 Fifteen-minute frequency to daily consumption

Since the objective of the present study is to forecast the day-ahead consumption, it was necessary to transform the 15-minute frequency kWh into daily kWh consumption. In order to accomplish this, it was necessary to aggregate the 15-minute resolution consumption values for each day. This was accomplished by using the function *resample*, that effectively reorganizes the data by day and calculate the total kWh consumed for each day. The original data begins at 2015-10-27 11:30:00 and ends at 2023-12-14 01:15:00, indicating that both dates are characterised by incomplete records.

Consequently, these days do not provide complete enough information to calculate the total daily energy consumption. In order to address this issue, it was determined that the 2015-10-27 and 2023-12-14 recordings should be deleted. Ultimately, the final dataframe (Table 3) was stored for subsequent analyses.

Table 3: Processed dataset with the date and the respective daily consumption in kilowatt-hour.

Date	***kWh***
28/10/2015	7724.25
29/10/2015	8536.75
30/10/2015	7997.25
...	…
11/12/2023	7638.25
12/12/2023	7996
13/12/2023	8134

5.2 Endogenous Variable

This section is dedicated to the analysis of the time series in order to understand its behaviour and characteristics. An overall view of the time series is shown in Figure 17.

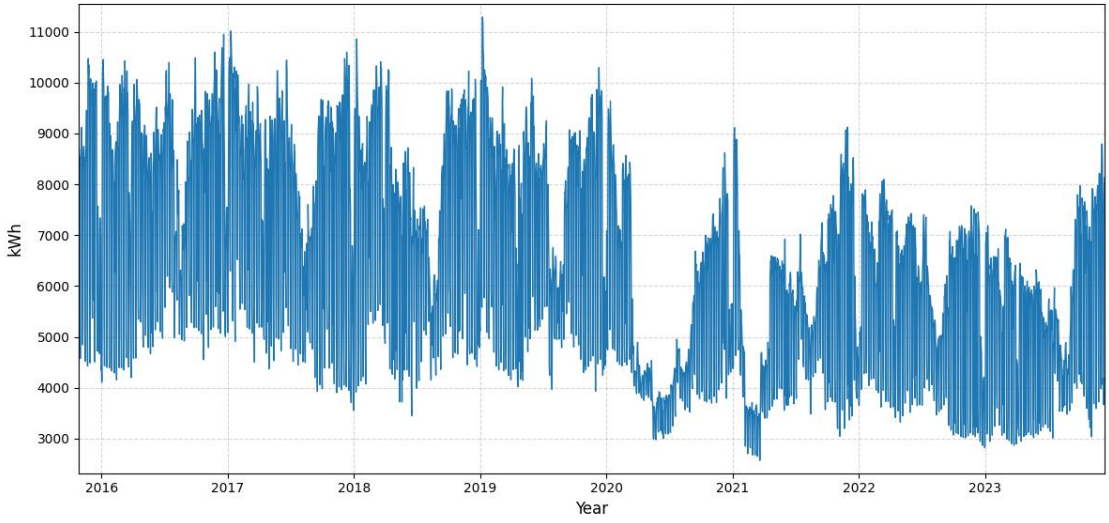


Figure 17: Evolution of the energy consumption time series across the years and recorded in kilowatt-hour.

The trend observed in the timeseries shows an overall linear decrease of electricity consumption over time (Figure 18). A more detailed examination of the trend reveals a gradual decline from 2016 to the beginning of 2020. It is at this stage that the initiation of a new cycle becomes observable, which starts at the early month of 2020 and finished

in the middle of that same year. Subsequently, an additional cycle was observed, spanning from the beginning to the end of 2021 due to the closure of public buildings caused by the pandemic. Consequently, there was a reduction in energy consumption during that period. Another component present in the time series is the seasonality, which is most evident when Figure 19 is visualized. The plot shows the average electric consumption by month where it can be noticed that the energy consumption decreases before the end of the year, followed by a peak at the beginning of the subsequent year. Afterwards, there is a decline in consumption from January, that stabilises towards the middle of the year. Next, an abrupt decline in energy consumption in August, followed by a period of continuous and steady increase that extends until November.

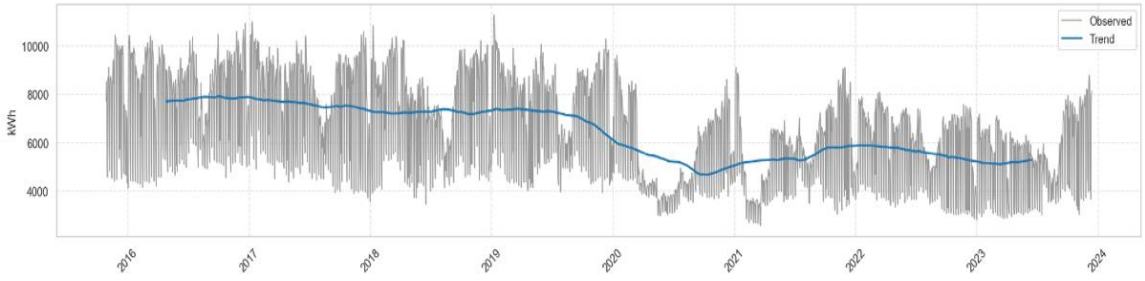


Figure 18: Trend decomposition of the consumption time series.

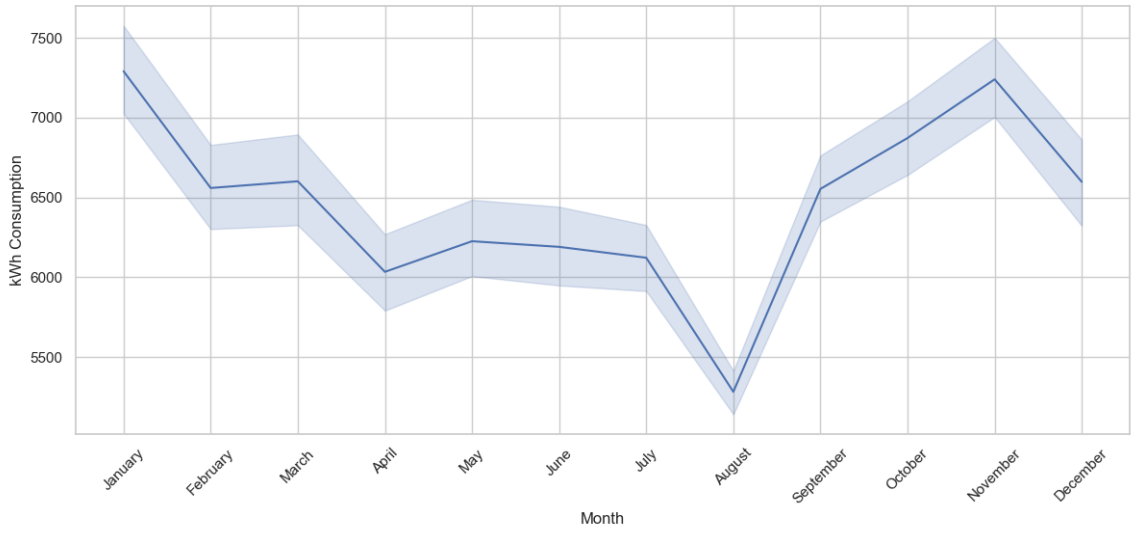


Figure 19: Average kWh consumption across a year.

This annual fluctuation in consumption is associated with the distinct academic periods experienced by the campus. The academic year is divided into two semesters, each comprising a series of classes and evaluation periods. These periods are subject to interruption by public holidays, such as Christmas, Carnival and Easter, as well as the

vacation period in August. This behaviour is evident in Figure 20 where there is a significant difference in the average energy consumption across these periods.

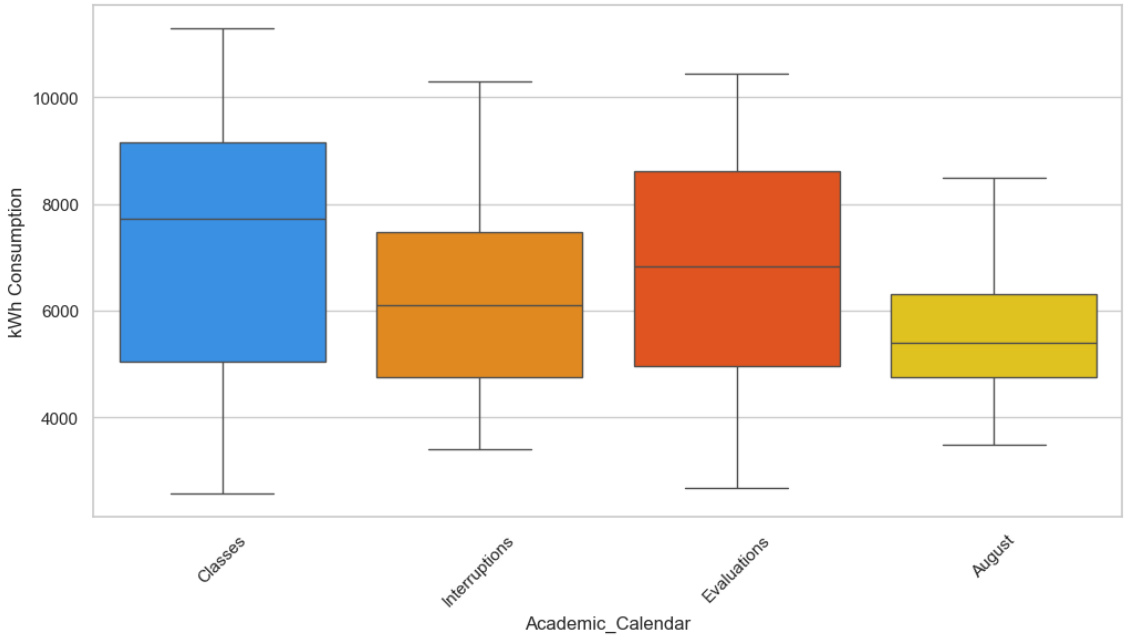


Figure 20: Average kWh consumption across the university periods.

In addition to the identification of annual patterns, weekly patterns were also identified. As demonstrated in Figure 21, the analysis of the data reveals that Saturday and Sunday exhibit a lower average energy consumption in comparison to the remaining days of the week. This phenomenon can be attributed to the reduced level of activity experienced during weekends, given that the majority of university activities are scheduled on working days.

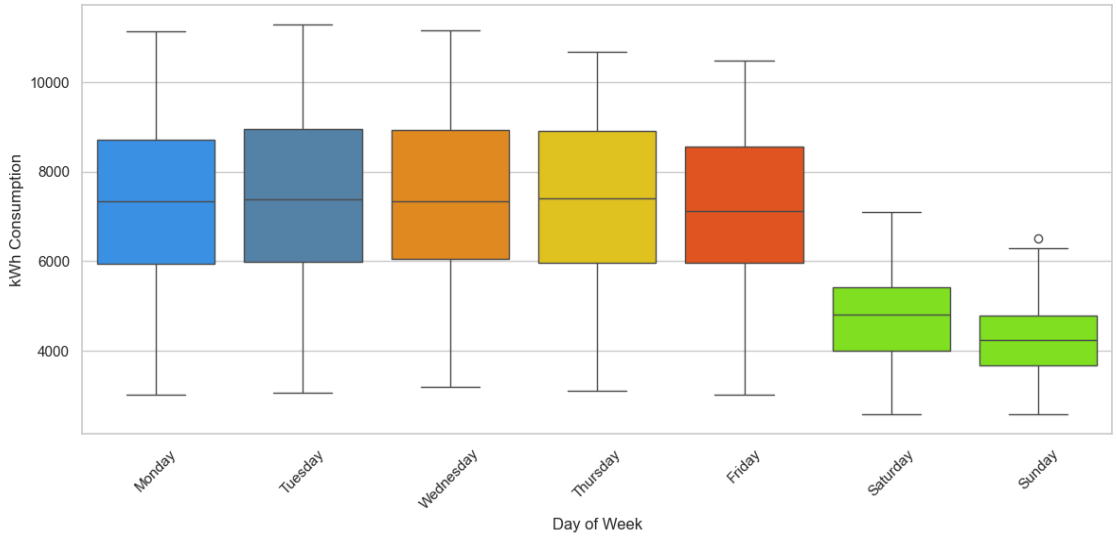


Figure 21: Average kWh consumption across a week.

It is therefore reasonable to assume that the same pattern would also be observed during the holidays. In order to perform this analysis, the average consumption was calculated for the holiday period and then compared with the consumption levels observed on weekends and working days. As demonstrated in Figure 22, the mean consumption during the holiday period is comparable to that on the weekends, but lower than the average of working days.

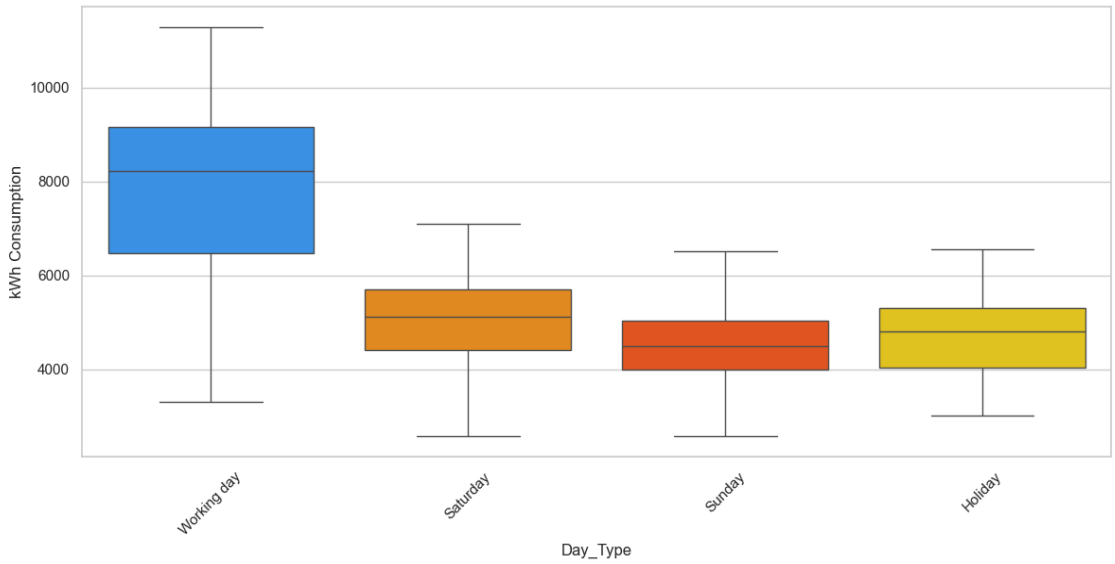


Figure 22: Average kWh consumption between day categories.

5.3 Exogenous variables

The above analysis shows a clear relationship between energy consumption and both specific weekdays and holidays, as well as the school calendar period. These patterns could provide a valuable opportunity to improve model accuracy through the creation of exogenous features.

5.3.1 Day Type

A new variable called Day Type, was created to classify the samples in either Saturday, Sunday, Holiday and Working Days. To do that, a custom formula was developed that assigns each date to the respective category. Additionally, to assist the model in comprehending the difference between these categories, a number was assigned to each category. These normalized values were created by dividing the average consumption of each day type by the average consumption on working days. Is also important to mention that the calculation of these values was based exclusively on the training data set. Using

these criteria, each date was assigned the following value based on its designated category:

- **Saturdays:** 0.654
    - Average Saturday consumption (5052.90 kWh) / Average working day consumption (7724.17 kWh);
- **Sundays**: 0.580
    - Average Sundays consumption (4483.40 kWh) / Average working day consumption (7724.17 kWh);
- **Holidays**: 0.615
    - Average Holidays consumption (4747.75 kWh) / Average working day consumption (7724.17 kWh);
- **Working Days**: 1.00
    - Average working day consumption (7724.17 kWh) / Average working day consumption (7724.17 kWh).

5.3.2 Academic calendar

The Academic Calendar variable was created in order to map the different academic periods that occurred previously. This dataset contained the “Data” column that holds the dates from 27/10/2015 to 13/12/2023, and the "codigo_CE" column that categorizes each date into one of the four academic periods with the corresponding code: 1: Classes; 2: Evaluations; 3: Interruptions; 4: August. As described in the previous section, these different categories were normalized by comparing the average consumption of each period to the average consumption of the classes period. It is important to note that the calculation of these values was done exclusively using the training dataset. Based on this procedure, each date was assigned the following value according to its category:

- **Evaluations period**: 0.940
    - Average Evaluations period consumption (6717.48 kWh) / Average Classes period consumption (7147.67 kWh).
- **Interruption period:** 0.868
    - Average Interruption period consumption (6203.16 kWh) / Average Classes period consumption (7147.67 kWh).
- **August period:** 0.774

  - Average Evaluations period consumption (5534.58 kWh) / Average Classes period consumption (7147.67 kWh).
- **Classes period**: 1.00
  - Average Classes period consumption (7147.67 kWh) / Average Classes period consumption (7147.67 kWh).

6. **Modelling**

The modelling phase is the stage where the models learn the underling patterns from data in order to generate predictions for the unseen data. From the several models that *NeuralForecast* offers, the following were used:

1. Feedforward-based neural network-based models: MLP, NBEATS, NHITS and TiDE;
2. Recurrent neural network-based models: LSTM and GRU;
3. Transformer-based neural network models: TFT and PatchTST;
4. Convolutional Neural Networks based Models: TCN;
5. Kolmogorov–Arnold Network-Based Models: KAN.

During the modelling phase, the dataset was divided into the train and test subsets. To evaluate the model performance over a full year, the last 365 days were reserved for the test set (12.29% of the original dataset), while the remaining data was used for training (representing 87.71% of the original time-series). With the train dataset, the models ran 200 times, each with a new configuration, to evaluate the model error across multiple combinations and ultimately identify the set of hyperparameters that produced the highest accuracy. However, it is often necessary to further adjust the space of search. For that, *Optuna* also provides validation plots after each run. The validation chart present in Figure 23 shows that the *Optuna* rapidly identifies the promising hyperparameter configurations for the LSTM model within the first trials, while subsequent iterations gradually refine the objective value. In instances where the current hyperparameters did not yield further enhancements in performance, additional hyperparameters associated with the model structure that had not previously been considered were introduced. In instances where the hyperparameter was determined to be relevant for the model structure but the designated search window was considered inappropriate, the search space could be expanded or reduced accordingly. The analysis of all the contour plots for each combination of hyperparameters, was a crucial step in this process. Hyperparameter tuning is a process which is computationally intensive and may require a significant amount of time. In order to enhance the efficacy of this step, the *CUDA* programming model was employed, facilitating GPU acceleration to reduce overall computation time

and enable a more exhaustive exploration of the hyperparameter space. After adjusting the hyperparameters and optimizing the search space, the best-performing set of hyperparameters for each model was identified and saved. The models best hyperparameters can be accessed in the Appendix C.

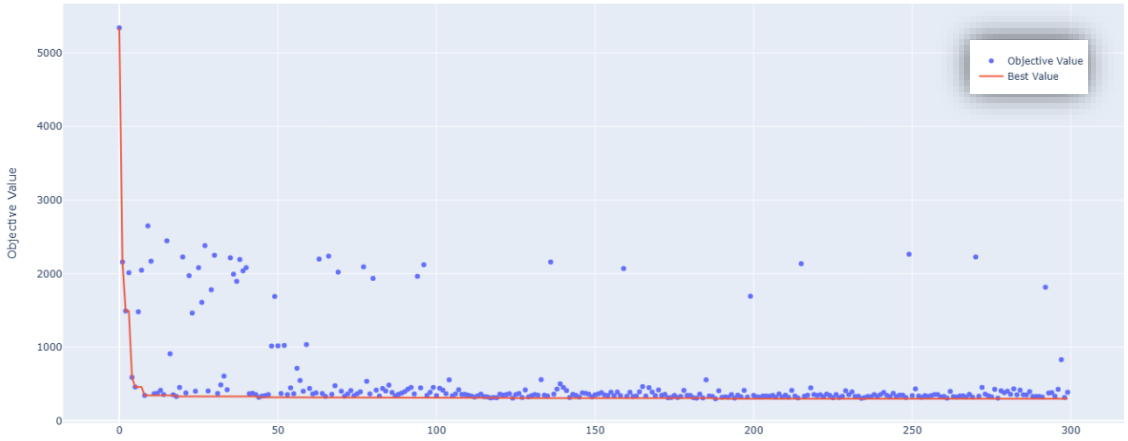


Figure 23: Optuna optimization plot showing the objective values across trials during LSTM hyperparameter fine-tuning.

During the training stage, each optimal model was obtained using their respective optimal hyperparameters derived from the fine-tuning stage. To ensure a more robust assessment of model performance, each model was run between 100 and 200 times, with a different sequential random seed assigned to each iteration. This ensures reproducibility since it defines how the weights initialize. In other words, for each model, it was saved several different versions of the same optimized model [83]. Also, during this phase, the train and validation losses were plotted to monitor if overfitting takes place. Computational efficiency metrics, such as the total number of parameters and model size, were also recorded and saved at this step.

During the test phase, each version of each model saved at the training stage is evaluated using a rolling window forecast to simulate a realistic deployment scenario. For each prediction, the model relied on the most recent actual values. The number of previous values utilised is dependent upon the specific model in question, as different models require different input sizes. This iterative forecasting process was repeated sequentially for all 365 days in the test dataset. For each run, the predicted and actual values were stored, and a comparison chart was generated to visually assess the forecast performance. Once all the predictions were completed, the error metrics were calculated for that run. This process was repeated across all versions of every model. Finally, the metrics from

the individual versions were aggregated to compute the overall average performance of each model, providing a more statistically robust evaluation.

7. Results and Discussion

In this chapter, the performance of the optimized models on each variable combination is compared to access the best performing model. The model accuracy is assessed on the test set using the Mean Absolute Percentage Error (MAPE), while model complexity is evaluated through the Total Parameters Count (TPC) measurement. The remaining metrics are not included in this discussion, as they convey the same information as the MAPE and would lead to identical conclusions. Nevertheless, in order to consult all the metrics yielded by each model, please refer to Appendix A. Additionally, since the PatchTST and the TiDE models did not yield satisfactory results, they were also excluded from the analyses. To add to the discussion of the results, an analysis of the errors of the best models was also conducted to identify the areas in which the models exhibited the most significant predictive inaccuracies.

7.1 Endogenous Variable

The models that were trained exclusively on the energy consumption variable achieved the levels of accuracy and complexity shown in Table 4. The table is arranged in ascending order of MAPE. The MAPE was calculated by averaging the values provided by all versions of the models, with the calculation of standard deviation used to provide a quantitative measure of the variability.

Table 4: Results of models trained exclusively on the endogenous variable. Average of the MAPE (Average ± Standard Deviation).

Models	***MAPE***	***TPC***
TFT	6.49 ± 0.14	203 406
NBEATSx	6.92 ± 0.04	805 393
NHITS	7.06 ± 0.07	2 402 861
MLP	7.36 ± 0.27	1 233
KAN	7.63 ± 0.05	8 235 260
LSTM	7.66 ± 0.52	199 793
GRU	7.76 ± 0.46	22 111
TCN	7.81 ± 0.24	364 027

The model that has been found to yield the highest level of accuracy is the TFT, which has a MAPE of 6.49%. The performance of the TFT model on the test dataset is illustrated in Figure 24. However, when the TPC is also taken into consideration, the MLP model demonstrates a good trade-off between accuracy and complexity. This is due to the fact that, despite exhibiting a higher MAPE in comparison to the TFT, the MLP is unquestionably the simpler model (1233 TPC) with the lowest MAPE (7.36%).

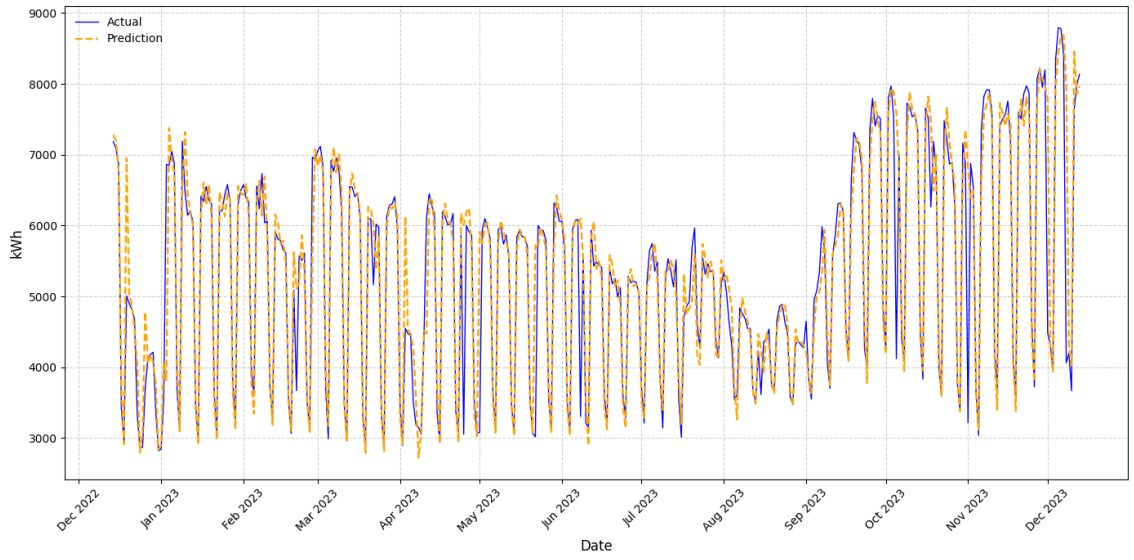


Figure 24: Comparison between the real and the predicted values for the TFT model trained exclusively on the endogenous variable.

In addition to these findings, the average error was computed for each date for the best performing model. This calculation considered all the predictions of the same record for the TFT model. Each average was then compared with the corresponding actual value to yield the Absolute Error (AE) and the Percentage Absolute Error (PE). The comparison of AE and PE between records enabled the identification of the dates and day categories associated with the highest inaccuracies. Table 5, shows the ten records that yielded the highest errors. By analysing this table, it is evident that the TFT model exhibited a persistent inability to accurately predict holidays. As demonstrated in Table 5, and also on Figure 24, the record with the most significant discrepancy between predicted and actual values is the 1st November of 2023, with a PE of 112.34%. This indicates that the prediction deviated from the actual value by more than double. Additionally, all the top ten records are associated with holidays, with the exception of the 3rd of January 2023. This date coincides with the beginning of a new evaluation period on campus following the vacation break, indicating that the models struggle to switch between these periods.

Table 5: Top 10 days with the highest error for all the models trained exclusively on the endogenous variable.

Date	*AE*	*PE*	*Day_Category*
01/11/2023	3614.87	112.34	Holiday
25/04/2023	2952.15	96.67	Holiday
08/12/2023	3922.04	96.57	Holiday
01/05/2023	2834.90	92.02	Holiday
22/05/2023	2703.13	89.59	Holiday
08/06/2023	2790.80	84.43	Holiday
05/10/2023	3466.85	84.09	Holiday
01/12/2023	3300.76	73.57	Holiday
03/01/2023	3049.07	44.43	Working day
21/02/2023	1443.54	39.34	Holiday

7.2 Exogenous Variables

This section is dedicated to the examination of models that have been trained using an endogenous variable in combination with the exogenous variables.

7.2.1 Academic Calendar Variable

Table 6 illustrates the outcomes attained by the models that were trained using the energy-consuming plus the academic calendar variable.

Table 6: Results of models trained on the endogenous and academic calendar variables. Average of the MAPE (Average ± Standard Deviation).

Models	*MAPE*	*TPC*
NHITS	6.88 ± 0.14	2 397 211
NBEATSx	6.95 ± 0.12	803 338
TFT	6.97 ± 0.12	202 382
MLP	7.79 ± 0.12	99 976
KAN	7.83 ± 0.12	2 051 090
LSTM	9.84 ± 0.12	26 153
TCN	9.97 ± 0.12	627 947
GRU	10.50 ± 0.12	35 849

NHITS demonstrates the highest accuracy with a MAPE of 6.88%. The visualization of this accuracy can be done on Figure 25 where it is shown how the NHITS models performed on the test set. Nevertheless, MLP is the model that best balances both metrics

in consideration, achieving a MAPE of 7.79% and a TPC of 99k. A comparison of the scores achieved by models trained with the academic calendar and those trained exclusively on the endogenous variable reveals a decline in performance. Nevertheless, the NHITS model is the only one to demonstrate enhanced performance upon the incorporation of this new variable. This finding suggests that the academic calendar in isolation is not a source of valuable information that would enhance the precision of the models. The NHITS model shows the top ten dates with the highest discrepancy between predictions and actual values are still the holiday records, as demonstrated in Table 7. It is evident that the NHITS exhibited an inability to accurately predict holidays like in the above trial.

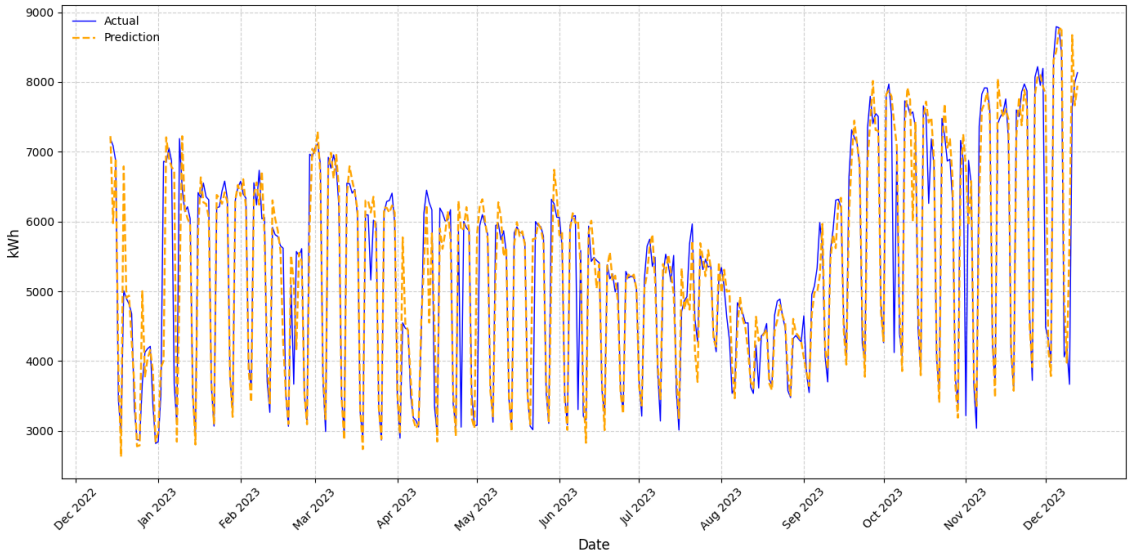


Figure 25: Comparison between the real and the predicted values for the NHITS model trained on the endogenous and academic calendar variables.

Table 7: Top 10 days with the highest error for all the models trained exclusively on the endogenous and academic calendar variables

Date	*AE*	*PE*	*Day_Category*
01/11/2023	3551.85	110.38	Holiday
25/04/2023	2817.92	92.28	Holiday
22/05/2023	2730.11	90.48	Holiday
01/05/2023	2736.13	88.81	Holiday
07/01/2023	3056.70	83.14	Saturday
08/06/2023	2708.16	81.93	Holiday
05/10/2023	3286.44	79.71	Holiday
01/12/2023	3347.74	74.62	Holiday
10/12/2023	2043.68	55.73	Sunday
03/01/2023	2858.33	41.65	Working day

7.2.2 Day type Variable

The models were also trained using the day type exogenous variable. Table 8 shows the results obtained by each model arranged in ascending order of MAPE.

Table 8: Results of models trained on the endogenous and day type variables. Average of the MAPE (Average ± Standard Deviation).

Models	*MAPE*	*TPC*
NBEATSx	5.41 ± 0.08	801 802
KAN	5.52 ± 0.15	3 792 800
NHITS	5.69 ± 0.10	2 397 211
TFT	5.73 ± 0.32	1 016 366
MLP	5.93 ± 0.22	43 153
LSTM	7.75 ± 1.32	57 417
GRU	10.4 ± 5.78	1 769
TCN	10.9 ± 2.41	772 789

It is evident that the NBEATSx model demonstrates the most favourable performance for this trial, exhibiting a MAPE of 5.41%. Regarding the complexity of the models, the GRU model obtained the lowest TPC. However, despite its simplicity, GRU's performance is considerably worst. In contrast, MLP once again demonstrated its capacity to balance both metrics in an effective manner.

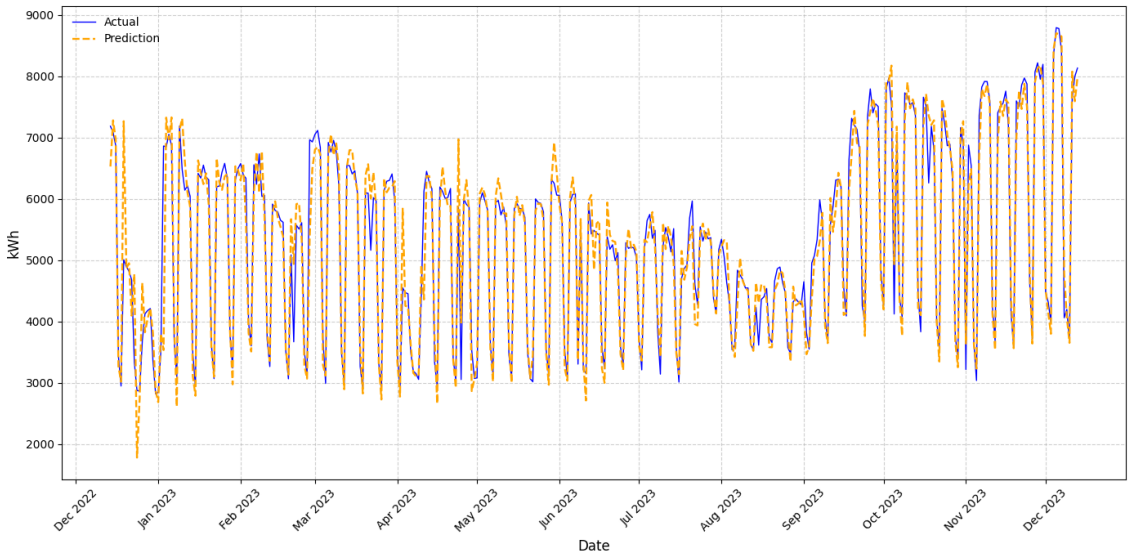


Figure 26: Comparison between the real and the predicted values for the NBEATSx model trained on the endogenous and day type variables.

By analysing Figure 26 and Table 9, we can see that this model has the capacity to predict holidays with a greater degree of efficacy. In this trial, both the number of holidays

associated with the highest errors and the corresponding percentage absolute errors decreased considerably when compared to previous trials. The ability of these models in distinguishing holidays and weekends provides a clear advantage over models trained only in the endogenous, which is reflected on the MAPE achieved. Nonetheless, these models demonstrate an inability to grasp the transitions between academic periods. The presence of a high error value at the beginning or end of these periods indicates this, as evidenced by the high errors associated to 23rd December 2023, 24th December 2023 and 3rd January 2023 records.

Table 9: Top 10 days with the highest error for all the models trained exclusively on the endogenous and day type variables

Date	*AE*	*PE*	*Day_Category*
03/01/2023	3328.44	48.50	Working day
19/12/2022	2260.12	45.13	Working day
23/12/2022	1436.86	43.10	Working day
24/12/2022	1093.19	38.04	Saturday
21/02/2023	1269.01	34.59	Holiday
25/04/2023	928.11	30.39	Holiday
11/04/2023	1762.95	28.80	Working day
03/04/2023	1292.73	28.42	Working day
26/12/2022	944.06	25.63	Working day
27/02/2023	1587.71	22.80	Working day

7.2.3 Academic Calendar plus Day-Type Variable

The models were also trained using both exogenous variables and the results from those models are shown in Table 10.

Table 10: Results of models trained on the endogenous, academic calendar and day type variables. Average of the MAPE (Average ± Standard Deviation).

Models	*MAPE*	*TPC*
NBEATSx	4.49 ± 0.08	806 410
KAN	4.92 ± 0.13	1 328 940
NHITS	5.13 ± 0.08	2 416 161
MLP	5.57 ± 0.15	51 337
TFT	5.59 ± 0.23	245 018
GRU	7.14 ± 1.21	48 599
TCN	9.61 ± 1.84	524 671
LSTM	10.30 ± 1.90	93 225

The models that demonstrated the highest degree of accuracy were NBEATSx and KAN, with a MAPE value of 4.92%. This is the lowest MAPE achieved in this study indicating that the combination of both exogenous variables provides valuable information to the modelling process. Additionally, GRU and MLP were the models that achieved the lower model complexity. However, MLP with a MAPE of 5.57 and a TPC of 51k has the best balance between accuracy and complexity.

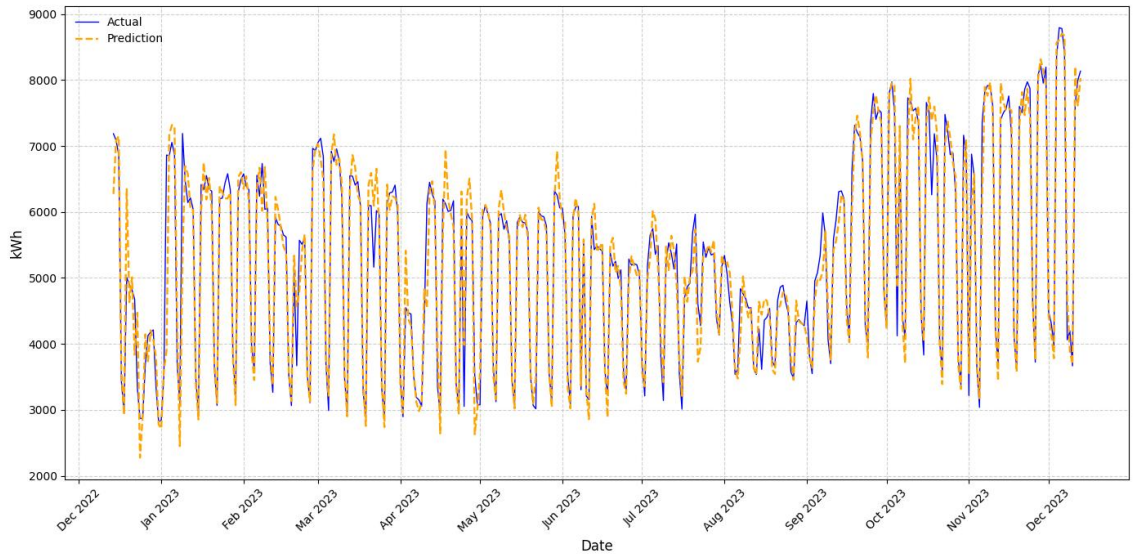


Figure 27:Comparison between the real and the predicted values for the NBEATSx model trained on the endogenous, academic calendar and day type variables.

The enhanced performance on holidays and the transition periods can be visualized in Figure 27. Table 11 further confirms this precision, as the results demonstrate a marked enhancement in the AE and PE of these records when compared to earlier trials. Nevertheless, as supported by Table 11, the transition periods and the holidays continue to demonstrate the lowest levels of accuracy, representing the domains in which the models encounter the most forecasting challenges.

Table 11: Top 10 days with the highest error for all the models trained exclusively on the endogenous, day type and academic calendar variables

Date	*AE*	*PE*	*Day_Category*
03/01/2023	2961.11	43.15	Working day
23/12/2022	1139.47	34.18	Working day
25/04/2023	930.67	30.48	Holiday
19/12/2022	1339.73	26.75	Working day
11/04/2023	1576.36	25.75	Working day
09/01/2023	1838.54	25.57	Working day
29/04/2023	897.68	25.44	Saturday
21/02/2023	897.99	24.47	Holiday
15/08/2023	829.04	22.93	Holiday
08/01/2023	644.37	20.83	Sunday

7.3 Overall Comparison

In order to obtain a comprehensive perspective on the full scope of the results, Figure 28 and the Table 12 provide a comparison of the models according to the forecast accuracy, given by MAPE, and the model complexity given by the count of the total parameters (TPC) across the different combinations of variables.

Table 12: Overall comparison table with all the results from each model across different variables. Average of the MAPE (Average ± Standard Deviation).

Model	**Endogenous**		**Exogenous**					
			Academic Calendar (AC)		**Day Type (DT)**		**AC+DT**	
	MAPE	*TPC*	*MAPE*	*TPC*	*MAPE*	*TPC*	*MAPE*	*TPC*
MLP	7.36 ± 0.27	**1 233**	7.79 ± 0.27	99 976	5.93 ± 0.22	43 153	5.57 ± 0.15	51 337
NBEATSx	6.92 ± 0.04	805 393	6.95 ± 0.12	803 338	5.41 ± 0.08	801 802	**4.92 ± 0.08**	806 410
NHITS	7.06 ± 0.07	2402 861	6.88 ± 0.14	2397 211	5.69 ± 0.10	2397 211	5.13 ± 0.08	2416 161
KAN	7.63 ± 0.05	8235 260	7.83 ± 0.19	2051 090	5.52 ± 0.15	3792 800	4.92 ± 0.13	1328 940
TCN	7.81 ± 0.24	364 027	9.97 ± 1.67	627 947	10.9 ± 2.41	772 789	9.61 ± 1.84	524 671
GRU	7.76 ± 0.46	22 111	10.5 ± 2.33	35 849	10.4 ± 5.78	1 769	7.14 ± 1.21	48 599
LSTM	7.66 ± 0.52	199 793	9.84 ± 1.56	26 153	7.75 ± 1.32	57 417	10.3 ± 1.90	93 225
TFT	6.49 ± 0.14	203 406	6.97 ± 0.25	202 382	5.73 ± 0.32	1016 366	5.59 ± 0.23	245 018

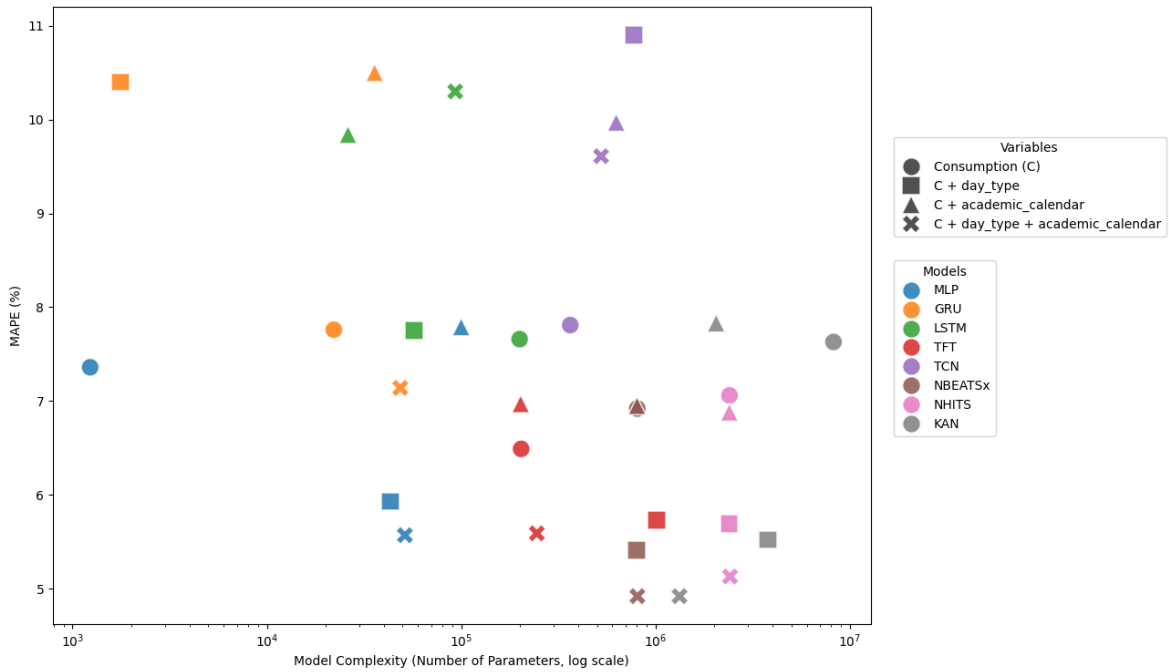


Figure 28: Comparison between model forecasting accuracy and complexity

The results show that the models trained on the academic calendar variable performed worse than the ones only trained with the endogenous variable. Additionally, the findings also indicate that the models which have been trained with the Day Type exogenous variable demonstrate higher levels of accuracy in comparison to those which have been exclusively trained with the endogenous variable. However, it is the combination of both exogenous variables that results in the most accurate models with the lowest MAPE. The same conclusions can be taken when considering Figure 29, that shows the overall errors deliver by best model on each variable. It is possible to notice that the models trained on the endogenous variables and on the academic calendar show the highest peaks, especially on holidays, while the other trials showed and significant increase in their accuracy.

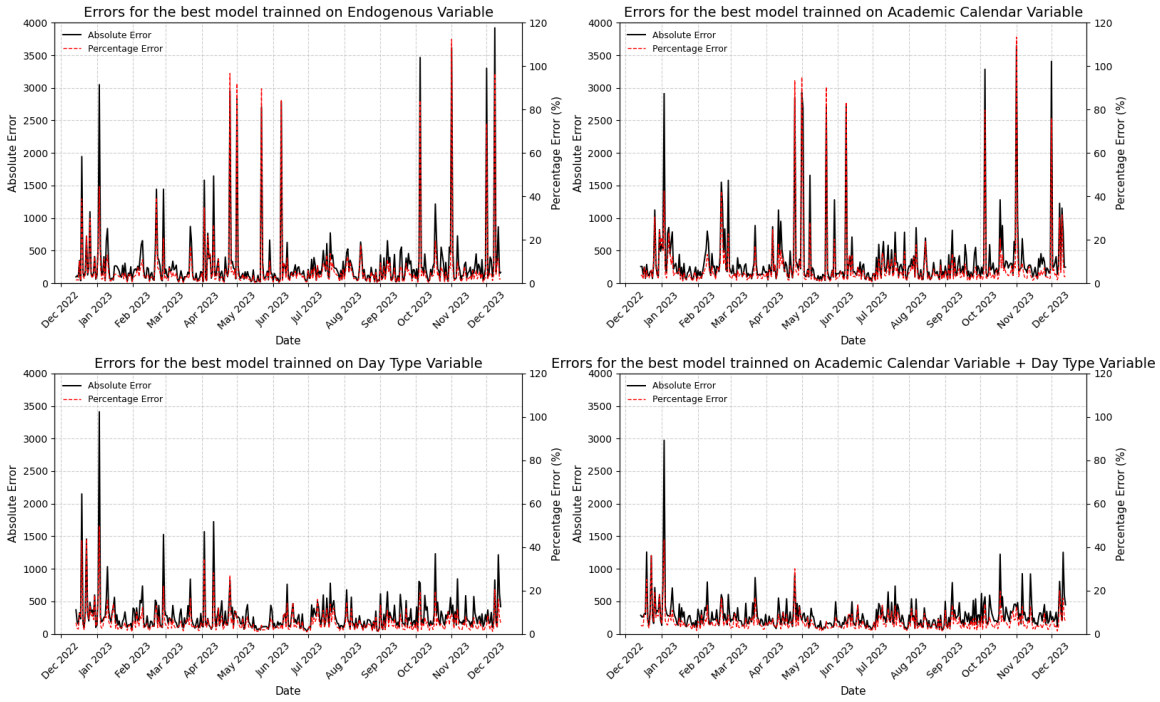


Figure 29: Absolute and percentage errors produced by the best model on the test set for each variable (TFT for endogenous variable, NHITS for academic calendar and NBEATSx for Day Type and Academic Calendar + Day Type)

These results indicate that the academic calendar variable alone does not improve the predictive power, while the Day Type is more informative for capturing consumption dynamics. This phenomenon could be attributed to the fact that the magnitude of consumption between the day types is more substantial than that observed between the academic periods. Nevertheless, it is the integration of complementary sources of information that enables the models to capture different dimensions of consumption behaviour simultaneously, thereby enhancing overall predictive accuracy.

Regarding the overall errors shown by best model, the NBEATSx trained on both exogenous variables, it is possible to notice that the months that demonstrate the highest average error rates are January, April and December (Figure 29 and Appendix B). This phenomenon can be attributed to the high number of public holidays observed in April and December. It is important to note that January also marks the beginning of an evaluation period. This is further compounded by the occurrence of holidays, which consequently impedes the capacity of the model to accurately predict these transition periods. Additionally, it is also evident in Figure 29 that all models encounter greater challenges in their predictions for the records from December and January. To sustain these findings, the results for the best model also demonstrate that the category of records

that demonstrate the highest levels of error is the holidays (Appendix B). This can be explained by the consumption patterns and behavioural shifts that occur during the holiday period that are atypical and deviate significantly from the regular temporal trends learned by the models. Despite these days being signalized by the Day Type variable, their capture remains challenging, particularly with regard to accurately measure their fluctuations. Regarding the errors by day of the week, it has been identified that the greatest error occurs on Mondays, Tuesdays and Saturdays (Appendix B). Overall, it can be concluded that the models exhibit the greatest difficulties in accurately predicting transition periods, either in the beginning and the end of the weeks, around holidays, and across different academic periods.

8. **Conclusion**

This work has demonstrated that the application of machine learning models can be used to accurately forecast energy consumption on a university campus. The development of these models was achieved by following a version of CRISP-DM methodology which allows a systematic evaluation of a range of machine learning models. The models under test were carefully chosen based on the literature review of similar studies. Additionally, the incorporation of long and short-term exogenous variables and the hyperparameter tuning allowed to identify the optimal conditions under which the performance of the models improves. The results show that the model that demonstrates the greatest accuracy is NBEATSx with a MAPE of 4.92%, trained on both exogenous variables. Additionally, the model that best balances accuracy and complexity is MLP, with a MAPE of 5.57% and a TPC of 51 337, also trained on both exogenous variables. Nevertheless, even in the best performing models, transition periods remain the most challenging records to forecast.

While the present work accomplishes its objectives, there is opportunity for the implementation of additional enhancements with a view to introduce more efficient approaches. The dataset used is significant in size however, increasing its size could result in the inclusion of additional information that may help the model to generalize better. The present study demonstrated the significance of exogenous variables, especially when exogenous variables that signal holidays and weekends were introduced. In future research, the improvement of the current exogenous variables or the introduction of additional exogenous variables could be considered with a view to enhancing the model's accuracy. Variables like the number of people on the campus and the number of students on the classroom are more precise indictors of the rate of occupancy that should be considered. Another important enhancement could be the incorporation of additional runs, alongside with the expansion of the number of hyperparameters to be optimized and the enlargement of the search space during the hyperparameter tuning. This would facilitate more extensive research and ensure that no potential variables are overlooked. The usage of enhanced computational capabilities could prove advantageous during this phase of the project. Also, the development of a model capable of predicting some hours

ahead would be a valuable endeavour. Given the higher resolution, such a model could improve the granularity of the results and enable a more precise management approach.

In summary, this work yielded both theoretical and practical insights into the developing of machine learning models for energy consumption prediction. Furthermore, it also provides additional evidence to support the establishment of neural networks as one of the most accurate methods for time series forecast, reinforcing their role in enabling more reliable data-driven decision-making.

Bibliography

[1] F. Rodrigues, C. Cardeira, J. M. F. Calado, and R. Melicio, "Short-Term Load Forecasting of Electricity Demand for the Residential Sector Based on Modelling Techniques: A Systematic Review," May 01, 2023, MDPI. doi: 10.3390/en16104098.

[2] J. Browell and M. Fasiolo, "Probabilistic Forecasting of Regional Net-Load With Conditional Extremes and Gridded NWP," IEEE Trans Smart Grid, vol. 12, no. 6, pp. 5011–5019, Nov. 2021, doi: 10.1109/TSG.2021.3107159.

[3] T. Kerr, "Combined Heat and Power: Evaluating the Benefits of Greater Global Investement," 2008. [Online]. Available: http://www.iea.org/Textbase/about/copyright.asp

[4] Brian Bothwell, "United States Government Accountability Office Utility-Scale Energy Storage Technologies and Challenges for an Evolving Grid," 2023.

[5] N. Shirzadi, A. Nizami, M. Khazen, and M. Nik-Bakht, "Medium-term regional electricity load forecasting through machine learning and deep learning," Designs (Basel), vol. 5, no. 2, Jun. 2021, doi: 10.3390/designs5020027.

[6] Turk Dave, "Together Secure Sustainable Digitalization & Energy," 2017. [Online]. Available: www.iea.org/t&c/

[7] H. Shareef, M. M. Islam, and A. Mohamed, "A review of the stage-of-the-art charging technologies, placement methodologies, and impacts of electric vehicles," Renewable and Sustainable Energy Reviews, vol. 64, pp. 403–420, Oct. 2016, doi: 10.1016/J.RSER.2016.06.033.

[8] Z. Wang, "Research on Deep Learning-Based Dynamic Load Forecasting and Optimal Dispatch in Smart Grids," Journal of Electronic Research and Application, vol. 9, no. 2, pp. 105–109, Apr. 2025, doi: 10.26689/jera.v9i2.10083.

[9] P. R. D. S. Oliveira, "Previsão de curto prazo para consumo de energia em Campi Universitário," Polytechnic Institute of Leiria, Leiria, 2024.

[10] C. , X. H. Chatfiedl, "The Analysis of Time Series, An Introduction with R, 7th Edition, ISBN-10: 1498795633," CRC Press, 2019.

[11] R. Dey, “Time Series Decomposition,” https://medium.com/@roshmitadey/time-series-decomposition-62cbf31ab65e.

[12] Box, G. E. P.; Jenkins, G. M.; Reinsel, G. C., “Time Series Analysis: Forecasting and Control (4th ed.),” Wiley, 2008.

[13] F. Petropoulos et al., “Forecasting: theory and practice,” Int J Forecast, vol. 38, no. 3, pp. 705–871, Jul. 2022, doi: 10.1016/j.ijforecast.2021.11.001.

[14] D. C Montgomery, Jennings, C. L.; Kulahci, M., Introduction to Time Series Analysis and Forecasting. 2015.

[15] A. Nielsenl, Practical time series analysis: Prediction with statistics and machine learning. O’Reilly Media, 2019.

[16] G. Bontempi, S. Ben Taieb, and Y. A. Le Borgne, “Machine learning strategies for time series forecasting,” in Lecture Notes in Business Information Processing, Springer Verlag, 2013, pp. 62–77. doi: 10.1007/978-3-642-36318-4_3.

[17] Turekian, V.; Jeong, T.; Gronvall, G. K.; Prescott, E.; Lee, G.; Lewis, R.; Woods, B., Building a smart partnership for the fourth industrial revolution. 2018.

[18] M. W. Cohen, M. Aga, and T. Weinberg, “Genetic Algorithm Software System for Analog Circuit Design,” Procedia CIRP, vol. 36, pp. 17–22, Jan. 2015, doi: 10.1016/J.PROCIR.2015.01.033.

[19] K. P. Murphy, Machine Learning: A Probabilistic Perspective. The MIT Press, 2012.

[20] S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd Edition. The MIT Press, 2018.

[21] P. Murphy Kevin, Machine Learning A Probabilistic Perspective . The MIT Press Cambridge, Massachusetts London, England, 2012.

[22] M. Schmitt, “Deep Learning in Business Analytics: A Clash of Expectations and Reality,” Jun. 2025, doi: 10.1016/j.jjimei.2022.100146.

[23] L. Hardesty, “Explained: neural networks,” MIT News 14, 20AD.

[24] H. Simon (1999). Neural Networks: A Comprehensive Foundation (2nd ed.). Prentice Hall.

[25] E. Cantez, H. Sahin, and O. F. Efe, “Regional Guidance System for Cleaning Robots as a Result of Pollution of Solar Panels,” The Eurasia Proceedings of Science Technology Engineering and Mathematics, vol. 22, pp. 274–279, Sep. 2023, doi: 10.55549/epstem.1350961.

[26] P. Opěla, I. Schindler, P. Kawulok, R. Kawulok, S. Rusz, and M. Sauer, “Shallow and deep learning of an artificial neural network model describing a hot flow stress Evolution: A comparative study,” Mater Des, vol. 220, Aug. 2022, doi: 10.1016/j.matdes.2022.110880.

[27] K. Benidis et al., “Deep Learning for Time Series Forecasting: Tutorial and Literature Survey,” Apr. 2020, doi: 10.1145/3533382.

[28] H. and J.-F. Mas. Taud, “Multilayer perceptron (MLP),” in Geomatic approaches for modeling land change scenarios, Cham: Springer International Publishing, 2017.

[29] J. Schmidhuber, “Deep Learning in Neural Networks: An Overview,” Apr. 2014, doi: 10.1016/j.neunet.2014.09.003.

[30] H. J. Kelley, “Gradient theory of optimal flight paths,” ARS Journal, 1960.

[31] S. V. L. Boyd, Convex Optimization. Cambridge University Press, 2004.

[32] Rumelhart, D. E.; Hinton, G. E.; Williams, R. J., “Learning representations by back-propagating errors,” Nature, 1986.

[33] Robert Keim, “An Introduction to Training Theory for Neural Networks,” https://www.allaboutcircuits.com/technical-articles/an-introduction-to-training-theory-for-neural-networks/.

[34] B. N. Oreshkin, Carpov D., Chapados N., and Bengio Y., “N-BEATS: Neural basis expansion analysis for interpretable time series forecasting,” May 2019, [Online]. Available: http://arxiv.org/abs/1905.10437

[35] K. G. Olivares, C. Challu, G. Marcjasz, R. Weron, and A. Dubrawski, “Neural basis expansion analysis with exogenous variables: Forecasting electricity prices with NBEATSx,” Apr. 01, 2023, Elsevier B.V. doi: 10.1016/j.ijforecast.2022.03.001.

[36] C. Challu, K. G. Olivares, B. N. Oreshkin, F. Garza, M. Mergenthaler-Canseco, and A. Dubrawski, "N-HiTS: Neural Hierarchical Interpolation for Time Series Forecasting," Jan. 2022, [Online]. Available: http://arxiv.org/abs/2201.12886

[37] C. Challu, K. G. Olivares, B. N. Oreshkin, F. Garza Ramirez, M. Mergenthaler-Canseco, and A. Dubrawski, "NHITS: Neural Hierarchical Interpolation for Time Series Forecasting," 2023. [Online]. Available: www.aaai.org

[38] A. Das, W. Kong, A. Leach, S. Mathur, R. Sen, and R. Yu, "Long-term Forecasting with TiDE: Time-series Dense Encoder," Apr. 2023, [Online]. Available: http://arxiv.org/abs/2304.08424

[39] K. O'Shea and R. Nash, "An Introduction to Convolutional Neural Networks," Dec. 2015, [Online]. Available: http://arxiv.org/abs/1511.08458

[40] V. H. Phung and E. J. Rhee, "A High-Accuracy Model Average Ensemble of Convolutional Neural Networks for Classification of Cloud Image Patches on Small Datasets," Applied Sciences, vol. 9, no. 21, p. 4500, Oct. 2019, doi: 10.3390/app9214500.

[41] S. Bai, J. Z. Kolter, and V. Koltun, "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling," Apr. 2018, [Online]. Available: http://arxiv.org/abs/1803.01271

[42] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu, "Convolutional neural networks for time series classification," Journal of Systems Engineering and Electronics, vol. 28, no. 1, pp. 162–169, Feb. 2017, doi: 10.21629/JSEE.2017.01.18.

[43] B. Ghojogh and A. Ghodsi, "Recurrent Neural Networks and Long Short-Term Memory Networks: Tutorial and Survey," Apr. 2023, [Online]. Available: http://arxiv.org/abs/2304.11461

[44] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A Critical Review of Recurrent Neural Networks for Sequence Learning," May 2015, [Online]. Available: http://arxiv.org/abs/1506.00019

[45] Y. Hu, A. Huber, J. Anumula, and S.-C. Liu, "Overcoming the vanishing gradient problem in plain recurrent networks," Jan. 2018, [Online]. Available: http://arxiv.org/abs/1801.06105

[46] R. Pascanu, T. Mikolov, and Y. Bengio, “On the difficulty of training Recurrent Neural Networks,” Nov. 2012, [Online]. Available: http://arxiv.org/abs/1211.5063

[47] Ixnay, “Recurrent neural network (RNN) and its unfold version,” https://commons.wikimedia.org/wiki/User:Ixnay.

[48] A. Graves, Long Short-Term Memory. In: Supervised Sequence Labelling with Recurrent Neural Networks. Studies in Computational Intelligence. Springer, Berlin, Heidelberg., 2012.

[49] R. C. Staudemeyer and E. R. Morris, “Understanding LSTM - a tutorial into Long Short-Term Memory Recurrent Neural Networks,” Sep. 2019, [Online]. Available: http://arxiv.org/abs/1909.09586

[50] Dida, “What is an LSTM Neural Network?” Sep. 2025, [Online]. Available: https://dida.do/what-is-an-lstm-neural-network

[51] R. Dey and F. M. Salem, “Gate-Variants of Gated Recurrent Unit (GRU) Neural Networks.”

[52] R. Rana, “Gated Recurrent Unit (GRU) for Emotion Classification from Noisy Speech,” Dec. 2016, [Online]. Available: http://arxiv.org/abs/1612.07778

[53] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling,” Dec. 2014, [Online]. Available: http://arxiv.org/abs/1412.3555

[54] P. Li et al., “Bidirectional Gated Recurrent Unit Neural Network for Chinese Address Element Segmentation,” ISPRS Int J Geoinf, vol. 9, no. 11, p. 635, Oct. 2020, doi: 10.3390/ijgi9110635.

[55] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention Is All You Need,” 2017, [Online]. Available: https://arxiv.org/abs/1706.03762

[56] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, “A Time Series is Worth 64 Words: Long-term Forecasting with Transformers,” Nov. 2022, [Online]. Available: http://arxiv.org/abs/2211.14730

[57] B. Lim, S. Arık, N. Loeff, and T. Pfister, "Temporal Fusion Transformers for interpretable multi-horizon time series forecasting," Int J Forecast, vol. 37, no. 4, pp. 1748–1764, Oct. 2021, doi: 10.1016/J.IJFORECAST.2021.03.012.

[58] Z. Liu et al., "KAN: Kolmogorov-Arnold Networks," Apr. 2024, [Online]. Available: http://arxiv.org/abs/2404.19756

[59] V. Dhiman, "KAN: Kolmogorov-Arnold Networks: A review," 2024. [Online]. Available: https://personal.math.vt.edu/embree/math5466/lecture10.pdf

[60] T. Ji, Y. Hou, and D. Zhang, "A Comprehensive Survey on Kolmogorov Arnold Networks (KAN)," Jul. 2024, [Online]. Available: http://arxiv.org/abs/2407.11075

[61] Grob, A.; Lenders, A.; Schwenker, F.; Braun, D. A.; Fischer, D., "Comparison of short-term electrical load forecasting methods for different building types." Energy Informatics, vol. 4 (Suppl 3), article 13, 2021. doi: 10.1186/s42162-021-00172-6

[62] H. Son and C. Kim, "A deep learning approach to forecasting monthly demand for residential-sector electricity," Sustainability (Switzerland), vol. 12, no. 8, p. 3103, Apr. 2020, doi: 10.3390/SU12083103.

[63] M. Kasprzyk, P. Pełka, B. N. Oreshkin, and G. Dudek, "Enhanced N-BEATS for Mid-Term Electricity Demand Forecasting," Dec. 2024, [Online]. Available: http://arxiv.org/abs/2412.02722

[64] G. Dudek, "3ETS+RD-LSTM: A New Hybrid Model for Electrical Energy Consumption Forecasting."

[65] S. M. Jung, S. Park, S. W. Jung, and E. Hwang, "Monthly electric load forecasting using transfer learning for smart cities," Sustainability (Switzerland), vol. 12, no. 16, Aug. 2020, doi: 10.3390/SU12166364.

[66] G. Dudek, P. Pełka, and S. Smyl, "A Hybrid Residual Dilated LSTM end Exponential Smoothing Model for Mid-Term Electric Load Forecasting," Mar. 2020, [Online]. Available: http://arxiv.org/abs/2004.00508

[67] A. M. Alonso, F. J. Nogales, and C. Ruiz, "A Single Scalable LSTM Model for Short-Term Forecasting of Disaggregated Electricity Loads," Oct. 2019, [Online]. Available: http://arxiv.org/abs/1910.06640

[68] N. Pooniwala and R. Sutar, “Forecasting Short-Term Electric Load with a Hybrid of ARIMA Model and LSTM Network,” in 2021 International Conference on Computer Communication and Informatics (ICCCI), IEEE, Jan. 2021, pp. 1–6. doi: 10.1109/ICCCI50826.2021.9402461.

[69] Y. J. Ma and M. Y. Zhai, “Day-Ahead prediction of microgrid electricity demand using a hybrid artificial intelligence model,” Processes, vol. 7, no. 6, Jun. 2019, doi: 10.3390/pr7060320.

[70] L. Cao, Y. Li, J. Zhang, Y. Jiang, Y. Han, and J. Wei, “Electrical load prediction of healthcare buildings through single and ensemble learning,” Energy Reports, vol. 6, pp. 2751–2767, Nov. 2020, doi: 10.1016/j.egyr.2020.10.005.

[71] D. Mariano-Hernández et al., “A data-driven forecasting strategy to predict continuous hourly energy demand in smart buildings,” Applied Sciences (Switzerland), vol. 11, no. 17, Sep. 2021, doi: 10.3390/app11177886.

[72] H. S. Oliveira and H. P. Oliveira, “Transformers for Energy Forecast,” Sensors, vol. 23, no. 15, Aug. 2023, doi: 10.3390/s23156840.

[73] F. Pallonetto, C. Jin, and E. Mangina, “Forecast electricity demand in commercial building with machine learning models to enable demand response programs,” Energy and AI, vol. 7, Jan. 2022, doi: 10.1016/j.egyai.2021.100121.

[74] E. Vivas, H. Allende-Cid, and R. Salas, “A systematic review of statistical and machine learning methods for electrical power forecasting with reported mape score,” Dec. 01, 2020, MDPI AG. doi: 10.3390/e22121412.

[75] Ncr and J. Clinton, “Step-by-step data mining guide,” DaimlerChrysler, 2000.

[76] Kin G. Olivares, Cristian Challú, Max Mergenthaler Canseco, Federico Garza, and Artur Dubrawski, “NeuralForecast: User friendly state-of-the-art neural forecasting models.,” 2022, PyCon Salt Lake City, Utah, US.

[77] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna,” in Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, New York, NY, USA: ACM, Jul. 2019, pp. 2623–2631. doi: 10.1145/3292500.3330701.

[78] Joos Korstanje, Advanced Forecasting with Python: With State-of-the-Art-Models Including LSTMs, Facebook's Prophet, and Amazon's DeepAR. Springer, 2021.

[79] R. Fildes and A. Davydenko, "Measuring forecasting accuracy: The case of judgmental adjustments to SKU-level demand forecasts," Int J Forecast, 2013.

[80] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," International Journal of Forecasting, vol. 22, no. 4, pp. 679–688, 2006.

[81] The pandas development team, "pandas-dev/pandas: Pandas," 2020, Zenodo.

[82] C. R. Harris et al., "Array programming with NumPy," Nature, vol. 585, no. 7825, pp. 357–362, Sep. 2020, doi: 10.1038/s41586-020-2649-2.

[83] A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition. O'Reilly Media, Inc., 2019.

Appendix A

In this appendix the metrics yielded by each model at each variable are displayed.

Table A1: Error metrics and corresponding standard deviations obtained for each model trained exclusively on the endogenous variable. Average ± standard deviation.

	MAE	***MSE***	***RMSE***	***MAPE***	***SMAPE***	***R2***
GRU	357.47±19.5	425245.4±25119.58	651.83±19.08	7.76±0.46	6.97±0.39	0.81±0.01
KAN	354.33±2.59	450743.68±5496	671.36±4.09	7.63±0.05	6.88±0.06	0.8±0.00
LSTM	354.15±22.44	432171.06±28498.16	657.05±21.35	7.66±0.52	6.93±0.45	0.81±0.01
MLP	345.35±13.57	430590.19±39352.79	655.53±29.5	7.37±0.28	6.76±0.28	0.81±0.02
NBEATSx	320.77±1.91	400829.04±4912.29	633.1±3.87	6.92±0.04	6.22±0.04	0.83±0
NHITS	330.39±3.48	396417.43±4153.54	629.61±3.3	7.06±0.07	6.32±0.07	0.83±0
TCN	361.74±12.01	455811.08±19174	674.99±14.05	7.81±0.24	7.08±0.24	0.8±0.01
TFT	302.48±7.35	371144.21±7212.37	609.19±5.89	6.48±0.15	5.82±0.15	0.84±0

Table A2: Error metrics and corresponding standard deviations obtained for each model trained on the endogenous variable and the academic calendar. Average ± standard deviation.

	MAE	***MSE***	***RMSE***	***MAPE***	***SMAPE***	***R2***
GRU	382.68±41.9	451641.47±55449.23	670.91±39.01	8.38±1.02	7.67±1.02	0.8±0.02
KAN	374.8±9.47	418334.27±20614.78	646.59±15.95	7.83±0.19	7.23±0.18	0.82±0.01
LSTM	898.9±74.98	1708606.61±265436.53	1303.07±103.05	18.86±1.56	17.48±1.42	0.26±0.12
MLP	348.42±14.96	416475.86±42270.36	644.54±32.37	7.31±0.27	6.83±0.32	0.82±0.02
NBEATSx	340.19±8.93	436681.14±27677.32	660.49±20.94	7.31±0.17	6.63±0.18	0.81±0.01
NHITS	341.37±8.21	412872.23±23546.61	642.29±18.21	7.2±0.15	6.6±0.16	0.82±0.01
TCN	476.11±84.65	590415.41±97419.59	766.07±59.63	9.97±1.67	9.26±1.64	0.74±0.04
TFT	348.29±15.31	407262.19±30940.96	637.72±24.08	7.36±0.33	6.86±0.32	0.82±0.01

Table A3: Error metrics and corresponding standard deviations obtained for each model trained on the endogenous variable and the day type. Average ± standard deviation.

	MAE	***MSE***	***RMSE***	***MAPE***	***SMAPE***	***R^2***
GRU	542.05±317.12	634598.23±2403097.76	691.13±396.16	11.72±6.62	11.04±4.17	0.72±1.05
KAN	279.73±6.91	183043.07±8595.89	427.72±10.06	5.52±0.15	5.54±0.15	0.92±0
LSTM	376.5±50.48	284587.93±54165.81	531.32±47.8	7.98±1.24	7.74±1.09	0.88±0.02
MLP	297.86±10.32	215106.69±17565.09	463.42±18.72	5.93±0.22	5.9±0.23	0.91±0.01
NBEATSx	272.1±3.64	176072.98±4306.04	419.58±5.12	5.41±0.08	5.41±0.08	0.92±0
NHITS	286.6±5.29	193434.77±7638.08	439.73±8.69	5.69±0.1	5.67±0.1	0.92±0
TCN	530.76±126.38	674259.95±164960.51	816.19±89.95	10.97±2.41	10.36±2.58	0.71±0.07
TFT	280.7±10.13	186056.54±9079.63	431.21±10.53	5.51±0.21	5.5±0.21	0.92±0

Table A4: Error metrics and corresponding standard deviations obtained for each model trained on the endogenous variable, the academic calendar and the day type. Average ± standard deviation.

	MAE	***MSE***	***RMSE***	***MAPE***	***SMAPE***	***R^2***
GRU	337.31±49.53	220373.73±47783	467±47.76	7.14±1.21	6.93±1.04	0.9±0.02
KAN	242.67±5.97	114512.31±5927.48	338.28±8.76	4.92±0.13	4.9±0.13	0.95±0
LSTM	488.36±79.11	411434.94±105907.53	636.44±79.85	10.34±1.9	9.99±1.61	0.82±0.05
MLP	270.82±7.25	139671.88±7499.6	373.59±10.01	5.57±0.15	5.54±0.16	0.94±0
NBEATSx	258.78±4.7	154391.68±6897.28	392.83±8.76	4.49±0.08	5.08±0.09	0.93±0
NHITS	259.06±4.05	137472.18±5461.47	370.7±7.35	5.13±0.08	5.1±0.08	0.94±0
TCN	444.84±90.64	570032.36±122618.54	751.38±73.91	9.61±1.84	8.63±1.71	0.75±0.05
TFT	284.52±11.25	166138.52±14305.64	407.23±17.49	5.59±0.23	5.58±0.23	0.93±0.01

Appendix B

In this appendix, the errors for the best model were calculated by several categories.

Table B1: Monthly averages of the absolute error (AE) and percentual absolute error (PE) aggregated for all the NBEATSx runs on both exogenous variables.

Month	***AE***	***PE***
April	349.73	7.73
December	347.41	7.61
January	378.97	6.63
July	273.43	5.69
February	259.02	4.98
October	290.49	4.78
August	194.82	4.54
March	230.59	4.20
September	232.43	4.18
November	232.05	3.62
June	167.78	3.61
May	146.91	3.16

Table B2: Weekly averages of the absolute error (AE) and percentual error (PE) aggregated for all the NBEATSx runs on both exogenous variables.

Day of Week	***AE***	***PE***
Monday	380.23	6.78
Tuesday	376.28	6.77
Saturday	172.93	4.69
Wednesday	267.52	4.45
Thursday	253.41	4.43
Friday	227.08	4.32
Sunday	133.81	4.02

Table B3: Day Category averages of the absolute error (AE) and percentual error (PE) aggregated for all the NBEATSx runs on both exogenous variables.

Day of Week	***AE***	***PE***
Holiday	395.40	11.48
Working day	291.57	4.94
Saturday	174.85	4.73
Sunday	136.96	4.10

Appendix C

Best hyperparameters results for each model trained exclusively on the endogenous variable.

GRU: batch_size = 8; early_stop_patience_steps = 20; max_steps = 7500; val_check_steps = 50; input_size = 29; learning_rate = 0.000444585; scaler_type = standard; encoder_n_layers = 2; encoder_dropout = 0.014743542; encoder_hidden_size = 46; decoder_layers = 1; context_size = 8

KAN: batch_size = 24; early_stop_patience_steps = 40; max_steps = 15000; val_check_steps = 50; input_size = 10; learning_rate = 2.11E-05; scaler_type = minmax; hidden_size = 902; n_hidden_layers = 2

LSTM: batch_size = 16; early_stop_patience_steps = 40; max_steps = 15000; val_check_steps = 50; input_size = 35; learning_rate = 0.000116081; scaler_type = standard; encoder_n_layers = 1; encoder_dropout = 0.392630507; encoder_hidden_size = 196; decoder_layers = 3; context_size = 8

MLP: batch_size = 8; early_stop_patience_steps = 20; max_steps = 7500; val_check_steps = 50; input_size = 8; learning_rate = 0.000877348; scaler_type = minmax; hidden_size = 22; num_layers = 3

NBEATS: batch_size = 24; early_stop_patience_steps = 40; max_steps = 15000; val_check_steps = 50; input_size = 16; learning_rate = 6.92E-06; scaler_type = minmax; stack_types = ['identity']; dropout_prob_theta = 0.390188954; activation = ReLU

NHITS: batch_size = 24; early_stop_patience_steps = 20; max_steps = 7500; val_check_steps = 50; input_size = 14; learning_rate = 5.40E-06; scaler_type = minmax

TCN: batch_size = 24; early_stop_patience_steps = 40; max_steps = 15000; val_check_steps = 50; input_size = 40; learning_rate = 4.97E-05; scaler_type = identity; encoder_hidden_size = 212; decoder_hidden_size = 214; context_size = 6; decoder_layers = 1; encoder_activation = ReLU

TFT: batch_size = 128; early_stop_patience_steps = 40; max_steps = 15000; val_check_steps = 50; input_size = 10; learning_rate = 8.38E-06; scaler_type = standard; hidden_size = 64; n_head = 4

Best hyperparameters results for each model trained on the endogenous and academic calendar variable.

GRU: batch_size = 100; early_stop_patience_steps = 40; max_steps = 100; val_check_steps = 50; input_size = 20; learning_rate = 0.002397041; scaler_type = standard; encoder_n_layers = 1; encoder_dropout = 0.125382207; encoder_hidden_size = 168

KAN: batch_size = 101; early_stop_patience_steps = 40; max_steps = 100; val_check_steps = 50; input_size = 9; learning_rate = 0.003908712; scaler_type = standard; hidden_size = 443; n_hidden_layers = 2

LSTM: batch_size = 29; early_stop_patience_steps = 40; max_steps = 100; val_check_steps = 50; input_size = 12; learning_rate = 0.000109289; scaler_type = minmax; encoder_n_layers = 1; encoder_dropout = 0.024299456; encoder_hidden_size = 67

MLP: batch_size = 340; early_stop_patience_steps = 40; max_steps = 100; val_check_steps = 50; input_size = 8; learning_rate = 0.010682598; scaler_type = standard; hidden_size = 68; num_layers = 7

NBEATSX: batch_size = 111; early_stop_patience_steps = 40; max_steps = 100; val_check_steps = 50; input_size = 9; learning_rate = 0.004536644; scaler_type = minmax; stack_types = ['identity']; dropout_prob_theta = 0.157898425; activation = ReLU

NHITS: batch_size = 301; early_stop_patience_steps = 40; max_steps = 100; val_check_steps = 50; input_size = 10; learning_rate = 0.002670448; scaler_type = standard; pooling_mode = MaxPool1d; interpolation_mode = linear; dropout_prob_theta = 0.28238001; activation = ReLU

TCN: batch_size = 267; early_stop_patience_steps = 40; max_steps = 100; val_check_steps = 50; input_size = 40; learning_rate = 0.001058395; scaler_type = identity; encoder_hidden_size = 278; decoder_hidden_size = 196; context_size = 15; decoder_layers = 1; encoder_activation = ReLU

TFT: batch_size = 463; early_stop_patience_steps = 40; max_steps = 100; val_check_steps = 50; input_size = 7; learning_rate = 0.001615295; scaler_type = standard; hidden_size = 152; n_head = 8; attn_dropout = 0.366822972; dropout = 0.598265892; rnn_type = gru

Best hyperparameters results for each model trained on the endogenous and day type variable.

GRU: batch_size = 313; early_stop_patience_steps = 40; max_steps = 100; val_check_steps = 50; input_size = 14; learning_rate = 0.027693373; scaler_type = standard; encoder_n_layers = 1; encoder_dropout = 0.579097648; encoder_hidden_size = 9; decoder_layers = 2; context_size = 4

KAN: batch_size = 423; early_stop_patience_steps = 40; max_steps = 100; val_check_steps = 50; input_size = 8; learning_rate = 0.007881644; scaler_type = standard; hidden_size = 431; n_hidden_layers = 3

LSTM: batch_size = 232; early_stop_patience_steps = 40; max_steps = 100; val_check_steps = 50; input_size = 19; learning_rate = 0.001511232; scaler_type = standard; encoder_n_layers = 1; encoder_dropout = 0.272580602; encoder_hidden_size = 115; decoder_layers = 2; context_size = 8

MLP: batch_size = 189; early_stop_patience_steps = 40; max_steps = 100; val_check_steps = 50; input_size = 9; learning_rate = 0.015235472; scaler_type = standard; hidden_size = 116; num_layers = 4

NBEATSX: batch_size = 39; early_stop_patience_steps = 40; max_steps = 100; val_check_steps = 50; input_size = 8; learning_rate = 0.001447193; scaler_type = standard; stack_types = ['identity']; dropout_prob_theta = 0.002914144; activation = PReLU

NHITS: batch_size = 170; early_stop_patience_steps = 40; max_steps = 100; val_check_steps = 50; input_size = 8; learning_rate = 0.00055211; scaler_type = standard; pooling_mode = MaxPool1d; interpolation_mode = cubic; dropout_prob_theta = 0.026163935; activation = LeakyReLU

TCN: batch_size = 394; early_stop_patience_steps = 40; max_steps = 100; val_check_steps = 50; input_size = 35; learning_rate = 0.001144242; scaler_type = identity; encoder_hidden_size = 309; decoder_hidden_size = 203; context_size = 12; decoder_layers = 1; encoder_activation = ReLU

TFT: batch_size = 69; early_stop_patience_steps = 40; max_steps = 100; val_check_steps = 50; input_size = 11; learning_rate = 0.001573876; scaler_type = standard; hidden_size = 144; n_head = 4; attn_dropout = 0.246984903; dropout = 0.309739251; rnn_type = gru

Best hyperparameters results for each model trained on the endogenous, academic calendar and day type variable.

GRU: batch_size = 88; early_stop_patience_steps = 40; max_steps = 100; val_check_steps = 50; input_size = 9; learning_rate = 0.001863524; scaler_type = standard; encoder_n_layers = 1; encoder_dropout = 0.568780358; encoder_hidden_size = 123; decoder_layers = 1; context_size = 4

KAN: batch_size = 274; early_stop_patience_steps = 40; max_steps = 100; val_check_steps = 50; input_size = 30; learning_rate = 0.005073915; scaler_type = standard; hidden_size = 321; n_hidden_layers = 2

LSTM: batch_size = 433; early_stop_patience_steps = 40; max_steps = 100; val_check_steps = 50; input_size = 11; learning_rate = 0.001128533; scaler_type = standard; encoder_n_layers = 1; encoder_dropout = 0.142062145; encoder_hidden_size = 110; decoder_layers = 3; context_size = 8

MLP: batch_size = 72; early_stop_patience_steps = 40; max_steps = 100; val_check_steps = 50; input_size = 30; learning_rate = 0.00421921; scaler_type = standard; hidden_size = 138; num_layers = 3

NBEATSX: batch_size = 233; early_stop_patience_steps = 40; max_steps = 100; val_check_steps = 50; input_size = 8; learning_rate = 0.001131408; scaler_type = standard; stack_types = ['identity']; dropout_prob_theta = 0.001110827; activation = PReLU

NHITS: batch_size = 366; early_stop_patience_steps = 40; max_steps = 100; val_check_steps = 50; input_size = 10; learning_rate = 0.000661075; scaler_type = standard; pooling_mode = MaxPool1d; interpolation_mode = nearest; dropout_prob_theta = 0.047592926; activation = LeakyReLU

TCN: batch_size = 81; early_stop_patience_steps = 40; max_steps = 100; val_check_steps = 50; input_size = 21; learning_rate = 0.001659315; scaler_type = identity; encoder_hidden_size = 255; decoder_hidden_size = 155; context_size = 5; decoder_layers = 1; encoder_activation = ReLU

TFT: batch_size = 237; early_stop_patience_steps = 40; max_steps = 100; val_check_steps = 50; input_size = 11; learning_rate = 0.002801682; scaler_type = standard; hidden_size = 64; n_head = 8; attn_dropout = 0.391024033; dropout = 0.091187621; rnn_type = gru