Master Thesis

Comparative Study of Forecasting Algorithms for Energy Data

by

Nishat Fariha Rimi

April 8, 2019

Albert-Ludwigs-Universität Freiburg
Faculty of Engineering

Chair of Algorithms and Data Structures
Professor Dr. Hannah Bast

Supervisor
Prof. Dr. Hannah Bast

First Reviewer
Prof. Dr. Hannah Bast

Second Reviewer
Prof. Dr. Eng. Christof Wittwer

Advisors
Dr. David Fischer, Fraunhofer ISE, Freiburg, Germany.
MSc. Eng. Arne Groß, Fraunhofer ISE, Freiburg, Germany.

Table of Contents

Abstract

Accurate forecasting data is an important requirement of energy management systems and subsequent decision-making processes to ensure proper balance between supply and demand. The energy industry commonly carries out planning and forecasting of energy consumption for three types of prediction intervals: short-term, medium-term and long-term. The accuracy of a forecast depends on different parameters. Influence of the forecast horizon and the dataset type is significant among them. Often PV generation and electrical load profiles are subject to forecasting. Furthermore, different forecasting methods yield different accuracy. This study aims to identify the most appropriate method for accurately forecasting energy data based on the desired prediction horizon and the type of predicted dataset.

The analysed methods include three categories of approaches– statistical, machine learning and deep learning. Statistical approaches include: Autoregressive moving average (ARMA), Autoregressive integrated moving average (ARIMA), Exponential smoothing (ES) and Holt-Winters method (HW). Machine learning approaches are K-nearest neighbor (KNN) and Random forest (RF). From the deep learning approaches, Artificial neural network (ANN) and Recurrent neural network (RNN) have been focused. The forecasting quality for each method was measured using mean absolute error, mean absolute percentage error, root mean squared error, and the correlation coefficient R value. The results suggest that RF and KNN were the most appropriate forecasting algorithms for short-term (daily), medium-term (monthly) and long-term (3 monthly) forecasting horizon for both energy datasets. During the comparisons, these two methods provided higher accuracy and required minimum computational time to forecast among the compared methods. Additionally, ARIMA and ARMA methods performed better for very short-term (hourly) forecasting.

Zusammenfassung

Genaue Prognosedaten sind eine wichtige Voraussetzung für Energiemanagementsysteme und nachfolgende Entscheidungsprozesse, um das Gleichgewicht zwischen Angebot und Nachfrage zu gewährleisten. Die Energiewirtschaft plant und prognostiziert den Energieverbrauch für drei Arten von Vorhersageintervallen: kurz-, mittel- und langfristig. Die Genauigkeit einer Prognose hängt von verschiedenen Parametern ab. Dabei sind die Wahl des Prognosehorizonts und die Art des Datensatzes von großer Bedeutung. Häufig werden PV-Erzeugung und elektrische Lastprofile prognostiziert. Weiterhin ergeben verschiedene Prognosemethoden unterschiedliche Genauigkeit. Ziel dieser Studie ist es, die an der besten geeigneten Methode zur genauen Vorhersage von Energiedaten basierend auf dem gewünschten Vorhersagehorizont und der Art des vorhergesagten Datensatzes zu identifizieren.

Die analysierten Methoden umfassen drei Kategorien von Ansätzen - statistisches, maschinelles und tiefgehendes Lernen (deep learning). Statistische Ansätze beinhalten: Autoregressive moving average (ARMA), Autoregressive integrated moving average (ARIMA), Exponential smoothing (ES) und Holt-Winters method (HW). Machine Learning Ansätze sind die K-nearest neighbour (KNN) und Random forest (RF). Von den Deep-Learning-Ansätzen wurden Artificial neural network (ANN) und Recurrent neural network (RNN) fokussiert. Zusätzlich zum Faktor "Benutzerfreundlichkeit" wurde die Vorhersagequalität für jede Methode mit dem mittleren absoluten Fehler, dem mittleren absoluten Prozentfehler, dem mittleren quadratischen Fehler und dem Korrelationskoeffizienten R-Wert gemessen. Die Ergebnisse deuten darauf hin, dass RF und KNN die am besten geeigneten Prognoseverfahren für den kurzfristigen (täglich), mittelfristigen (monatlich) und langfristigen (3 Monate) Prognosehorizont für beide Energiedatensätze waren. Während der Vergleiche lieferten diese beiden Methoden eine höhere Genauigkeit und eine minimale Rechenzeit für die Vorhersage im Vergleich zu anderen Methoden. Bei sehr kurzfristigen (stündlichen) Prognosen schnitten die ARIMA und ARMA Methoden besser ab.

Declaration

I hereby declare that this Master Thesis has been composed by me based on my own work, that I have not used any sources other than those specified, and that all passages those have been taken literally or meaningfully from published writings have been specified as such. Furthermore, I declare that this thesis has not been fully or partially presented for any other test.

_____ _____

Place, Date **Nishat Fariha Rimi**

Acknowledgement

I express my sincere gratitude to the almighty God.

I am grateful to Prof. Dr. Hannah Bast for agreeing to be my academic supervisor and first reviewer of this thesis. Her supervision has been constructive. I also thank Professor Dr. Christof Wittwer for being my second supervisor and for sharing his unique knowledge and experience during the thesis.

I express my gratitude to Dr. David Fischer, senior scientist at the Group of Power Grids and Energy Management of Fraunhofer ISE for offering me this master thesis opportunity. His practical guidance, insights and valuable advices were very instrumental in carrying out this work.

My special thanks go to Arne Groß for being a patient mentor and helping me with discussions and clarifications. His audit and feedback on the drafts have been very useful in enhancing the nature of the theory. I appreciate and thank my colleagues for their assistance and extension of cooperation throughout my work at the Fraunhofer ISE.

Finally, my heartfelt appreciation and gratefulness go to my family and friends who continuously encouraged and supported me in completing this thesis.

List of Figures

List of Abbreviations

ACF	Autocorrelation function
AIC	Akaike information criterion
ANNs	Artificial neural networks
AR	Auto-regressive
ARIMA	Autoregressive integrated moving average
ARMA	Autoregressive moving average
CART	Classification and regression tree
ES	Exponential smoothing
HW	Holt Winters method
ISE	Institute for Solar Energy Systems
KNN	K-nearest neighbor
LTF	Long-term forecasting
MA	Moving average
MAE	Mean absolute error
MSE	Mean squared error
MTF	Medium-term forecasting
PACF	Partial autocorrelation function
PV	Photovoltaic
RF	Random forest
RNN	Recurrent neural network
STF	Short-term forecasting
TSA	Time series analysis
VSTF	Very short-term forecasting

1. Introduction

In recent years, greenhouse gas emissions have become an alarming issue. To mitigate the negative impacts and to cope with the increasing power demand, the diffusion of renewable-based domestic power generation plants has led to the placement of several smart grids throughout the world [1]. Renewable energy production in the EU has increased by two thirds between 2006-2017 [2] and reached almost 40% by the end of 2018 in Germany [3]. Renewable energy sources like wind and solar cannot be pre-scheduled, have their capacity constraints and unable to absorb, store or dispatch energy in the classical sense [4]. Moreover, the generated energy needs to be coordinated with the classical energy producers such as coal plants to avoid an energy surplus [5]. Thus, with the growing industry of renewable energy, there is a particular need for better forecasting of potential energy demands.

Energy load forecasting provides information about future demand and enables balancing between demand and production which eventually ensures that the energy produced will not be wasted (due to over-production) or insufficient (due to under-production). Forecasting techniques involve the study of past usages within a specific period of certain areas and then predict potential future consumptions as accurately as possible. In this research, the forecasting of energy data has been studied as the prediction of time-dependent data in the form of electrical load and photovoltaic load.

1.1. Motivation

It is vital to maintain a constant flawless supply of utilities. That is where forecasting can provide the information about high or low peak occurrences in future. The shortage of storage options and system losses of electricity could cause unnecessary expenses, while even a little enhancement in energy forecasting could decrease production expenses and increase exchanging benefits, specifically during the periods of peak electricity consumption [6]. According to Haida and Muto [7], the operation cost also raises due to the effect of forecasting errors (positive or negative). Specifically, new technologies like intelligent Energy Management Systems (EMS) [8], and model predictive control [9] have been explored to handle these issues which require a precise forecasting of load profiles to guarantee the optimal usage and safety. Load forecasting influences the operation and management of utility companies to ensure proper balancing between supply and demand. Forecasting minimizes the commercial operation, as the planned energy generation closely resemble future demand [5].

Appropriate forecasting also aids in planning for power systems to determine the required assets. A practical example can be planning the amount of fuel required for the coming week to operate power-generators. In the case of renewable sources, energy production should match the required demand as it is difficult to process or store unwanted large scales of electricity in

the grid. The forecasting data feeds into the energy management system (smart or traditional EMS) and helps the management to make appropriate decisions. Therefore, an optimal forecasting result with reasonable accuracy is vital for operational, commercial as well as for environmental reasons.

1.2. Research gap

The study on forecasting energy consumption can be dated back to 1966 [10]. A variety of forecasting approaches have been developed to improve the quality of prediction for energy data forecasting. The accuracy highly depends on the load forecasting methods as well as on the forecasted scenarios [26]. Various studies have classified the forecasting approaches in three main groups. For example, in their research Lahouar and Ben (2015) have grouped them in statistical methods, artificial intelligence methods and hybrid methods [11]. Statistical techniques such as Regression, Autoregressive moving average (ARMA), Autoregressive integrated moving average (ARIMA), Exponential smoothing (ES), and Holt-Winters (HW) have been used for short-term time series forecasting. Some other forecasting techniques are devoted to the use of "similar day" approach [12]. Approaches based on machine learning algorithms and deep neural network have received a lot of attention in recent years. This group includes different methods like K-nearest neighbor (KNN), Random forest (RF), Artificial neural networks (ANN), and Recurrent neural network (RNN).

A considerable amount of literature has focused on hybrid methods which combine the formerly mentioned approaches to improve accuracy by increasing their complexity [13]. However, these methods may not always produce optimal results and are often prone to 'overfitting' [14], [15]. Moreover, many studies have investigated the forecasting accuracy from different angles. It could be based on a specific prediction horizon (short-time/ long-time) or based on a specific historical data type or application. The selection of a forecasting algorithm depends on various aspects like the prediction time scale, input data type, and training sample size. The accuracy of forecasting algorithms changes according to these different forecasting aspects. Short-term forecasting of household electricity consumption has used ARIMA, NNs, and ES where results specify that depending on the choice of forecasting methods and parameter configuration, the accuracy of forecasting varies significantly [16]. Also, depending on the application area a different length of forecasting may be chosen; for example, the prediction of weather data requires short-term forecasting, whereas the production planning in a company needs a larger range. Another aspect of forecasting is that an appropriate amount of historical data should be analysed for making a good prediction as the size of the training data largely influences the accuracy of forecasting algorithms. As the prediction horizon increases, the accuracy of load forecasting decreases and vice versa.

There is a lack of research on determining the appropriate algorithm to apply for forecasting energy data of different time scales like short-term, medium-term or long-term. Theoretical studies mainly investigated forecasting algorithms with some specific or simple assumptions and provide a performance evaluation but do not serve practical and robust solutions quite as frequently; they are limited to specific interest. For example, some studies have investigated forecasting accuracy only for short-term forecasting of electricity consumption [17], or for medium-term load forecasting [18], and some others are focused on long-term load consumption [19]. In research on evaluating forecasting methods, J. S. Armstrong (2001) stated that the principles for testing forecasting methods are mostly based on commonly accepted methodological procedures, including pre-specified criteria. However, a comparison of accuracy between several forecasting algorithms considering all these aspects is so far lacking. In the vast area of forecasting, finding a proper solution to this question is a challenging task. Analysing the effect of dataset types and the size of training data on different forecasting approaches are essential, while the learning time and predicting time comparison is also necessary. Therefore, the combination of all the above conditions together for a detailed investigation will lead to a comparison of methods recognising the limitations of the individual method.

1.3. Goal

At the Fraunhofer Institute for Solar Energy Systems (also known as Fraunhofer ISE), different predictive methods have been used for the regulation of their energy system models. However, previous research has been limited to a specific aspect of forecasting. Therefore, it is not yet known which method is efficient for what kind of energy data (such as electricity load or PV generation) depending on the prediction horizons and training sample sizes. Hence, this thesis aims to examine various forecasting methods and evaluate the prediction quality depending on these forecasting aspects.

This work aims to analyse eight forecasting methods based on three different types of approaches- statistical, machine learning, and deep learning approaches. Photovoltaic generation and usage of electrical household load are chosen as the application areas for forecasting. This study assesses each method for various prediction horizons using different training sample size. Finally, this study attempts to comprehensively examine the predicted data using various key performances indicators. It will provide guidance to select the best suitable forecasting method for prediction in these application areas for a chosen forecasting horizon for both industrial and scientific usage.

1.4. Outline of the thesis

This thesis report is divided into six chapters. *Chapter 1* presents the introduction and background of the thesis.

Chapter 2 contains the literature review of the relevant research publications of load forecasting methods and distinguishes considerable arguments. A classification of the selected forecasting methods is presented at the end of this chapter.

Chapter 3 begins with the foundation of forecasting energy usage and time series analysis. Later, the theoretical background of selected methods is briefly presented, denoting the working procedure of each method and their limitations.

Chapter 4 covers the methodology used. Here, the data collection and processes are explained and justified respectively. Different forecasting aspects like an exogenous or endogenous variable and the training process are also discussed. The overview of forecasting accuracy measurements to evaluate the forecasting methods has been presented. At the end of this chapter, the implementation procedure including the coding platform and significant forecasting functions are also discussed.

Chapter 5 discusses the results that includes an analysis on the hyperparameter optimization procedure and the performance of all methods according to the key performance indicators (KPIs). Comparison of findings based on surveys and implementation is also justified in this section.

Chapter 6 draws conclusions based on the goal and objectives of this research and suggests future research topics and limitation of this research.

2. Related work

In general, the forecasting of energy data can be classified as a regression problem. A regression problem provides the output based on input data supplied to the system. The task here is to find the relationship between this input and output also known as regression analysis. While forecasting different energy data sources, an additional difficulty is the nonlinearity of this data [20]. Also, the seasonal component is included here which affects the load throughout the day. A variety of mathematical ideas and methods have been developed for forecasting which has led to more accurate energy data prediction.

The forecasting approaches can be classified into statistical, machine learning and deep learning approaches. Statistical approaches are considered as the white box models [11], where the internal processing is known along with the input and output relationships that are explicitly linked through mathematical equations. These approaches describe the forecasting that relay on past historic data. The simplest statistical method is based on a similar day approach [21]. The study of Zhang et al. (2015) presents a 'similar day' model which is used by data of three different locations throughout the world [12]. Other statistical approaches can be classified into autoregressive models and smoothing techniques. As per research work of Zheng in 2017 [22], statistical autoregressive methods (ARMA, ARIMA) forecast works in assuming that the two time series (observed and future) are linearly related. Autoregressive categories contain various autoregressive models, such as autoregressive moving average (ARMA) [23], Autoregressive integrated moving average (ARIMA) [24], [25] the well-known Box–Jenkins models and smoothing techniques covered by the Exponential smoothing (ES) [26] and Holt-Winters (HW) [27]. In time series analysis and forecasting, these smoothing methods are broadly used. Among them, the most common Holt-Winters exponential smoothing method uses a formulation to treat trend, seasonal patterns, and level separately. Another benefit of using statistical models is that if the time series has single variable of interest (dependent variable) available for different periods then unlike regression problem, statistical methods can forecast this univariate time series. Although the implementation of these statistical approaches is simple and adapted well for the short-term scale usage, autoregressive models are mainly used where previous load values is a linear aggregate of current data [28]. Description of this category has been presented in *section 3.3*.

Deep learning methods can overcome these limitations by discovering nonlinear relationships and can increase the forecast performance by a learning process using input-output relationships in time series. Because of having a complex structure, these are considered more like black-box models where the input and output (along with the relationship between them) is known but the internal mechanism is not well understood or known. These approaches have displayed advantages compared with statistical approaches in analysing a very short time frame

for large amounts of data [17]. Here, the input and output relationship is defined through using the learning process of artificial neural networks (ANN) where the current data is aggregated with the previous data. Various deep learning techniques such as Artificial neural networks (ANN) [29], [30] and Recurrent Neural Network [22] are largely used in the literature reviewed. The non-linear characteristics of electrical loads, which result according to occupants' behaviours and seasonal effects, have led many researches to focus on using artificial neural networks [31] to forecast. Nevertheless, the selection and configuration of input variables are difficult [32] and training phase has a high computational cost [33]. This forces a limitation to the input data and can result in less accurate forecasting. Zheng in 2017 pointed out that the non-stationary and non-seasonal nature in the time series of short-term load creates these forecasting challenges. This can be overcome using recurrent neural networks (RNN). A detailed description of this category has given in *section 3.5*.

Approaches based on machine learning algorithms have received much attention in recent years. This family includes some non-linear machine learning techniques, like k nearest neighbor [34], support vector machine (SVM) [35], [36] etc. Other than these methods, little attention was paid to the ensemble machine learning methods, such as Random Forest (RF) [37], [11]. Lahouar and Ben (2015) demonstrated that the RF method has the flexibility due to having low sensitivity to parameter values and can handle a load profile with complex customer behaviour. Dudek (2011) studied the effectiveness of RF in short-term forecasting for load data in Poland, and the performance has been compared with CART, ARIMA, ES, NN where the result highlighted that RF could be used for better forecasting accuracy. A brief description of this category has been presented in *section 3.4*.

Although these methods are effective, they still have limitations regarding optimal architecture or parameter tuning [11] which are exceeded by hybridisation. Lahouar (2015) reveals that the majority of researchers have developed hybrid ANN for both medium-term and long-term energy forecasting [11]. Therefore, research leads to hybrid methods [13] which combine the other forecasting approaches for increasing accuracy. A review on photovoltaic power forecasting has stated that varied machine learning approaches are studied broadly so far [38]. Therefore, based on these reviews, we have chosen some convincing methods to compare different energy forecasting. The classification of the chosen methods can be graphically represented according to the working mechanism in five groups as shown in *Figure 2.1*.

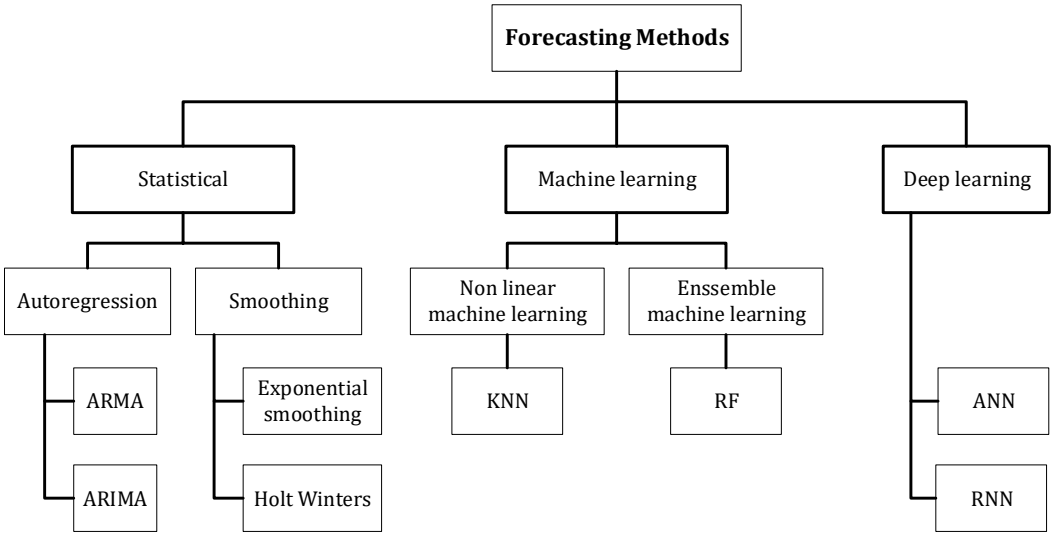

Figure 2.1: Different approaches for energy demand forecasting.

3. Theoretical background

This chapter provides the theoretical background of forecasting with various aspects. The basic definitions of time series analysis are discussed. This chapter also provides the background of different mathematical formulation of the selected methods, based on the literature overview. A comparative study of the reviews and remarks of these statistical approaches, machine learning approaches, and deep learning approaches are presented briefly.

3.1. Forecasting energy load

As discussed before, the requirement of energy differs from season to season, every day and every minute. Forecasting of energy data is the prediction of the rquired demand in short-term, medium-term and long-term prediction horizons. It is a guess of how much energy will be needed by the residences, companies, or other institutions in the future.

Energy data

Energy data refers to the enormous amounts of data related to energy consumption. This includes direct measures of energy usage (such as electricity load, PV generation, wind, gas, steam, heating load etc.). Different sources of energy are generated at distinct time intervals and units. This energy information is useful and used as historical data to train the forecasting algorithms and to get the forecast that defines the time series for a future time. The key point here is that a lot of information to make accurate results is required. To differentiate and examine the forecasting efficiency over different energy data applications, the PV generation and electrical load datasets of residential usage in this research work have been considered.

Forecasting horizons (Time scale)

Various types of approaches and models are covered in the literature for forecasting of energy data both for industrial and residential usage. Although, there is no official categorization defined for the type of forecasting horizons within the energy industries, it can be divided into four categories based on the time period or forecasting interval: very short-term forecasting (VSTF), short-term forecasting (STF), medium-term forecasting (MTF), and long-term forecasting (LTF) [39]. VSTF is a comparatively new type of forecasting, which mainly refers to forecasting that has time frame minute wise till 1 hour ahead [27]. The management system of demand sight often uses STF that considers a time range from hours to a day length ahead [17]. VSTF and STF are primarily applied to daily operation and scheduling of the power and spot price calculation where the required accuracy is much higher than a long term prediction [39]. This kind of forecasting is vital to ensure better efficiency of limited electricity in developing countries [40]. MTF refers to a period up to a month ahead, and has been used for scheduling maintenance and the development of the system of the grid for years [18]. Whereas,

LTF looks at a period exceeding from months to years ahead which apply to the scheduling of power supplies and resource planning. Such type of forecasting is utilized to determine system planning when power usage prediction over a longer period is of interest [39]. MTF and LTF often suffer from the forecast errors over time [41], so they usually are used only to forecast peak loads.

Table 3.1: Forecasting horizons

Forecasting horizons	Time scale
Very short-term	5min-1h
Short-term	1h- 24h
Medium-term	24h- weeks
Long-term	months-years

In research, Xia and Wang showed that the accuracy of STF is influenced by factors like temperature, humidity and wind speed [42]. Therefore, the influencing factors for forecasting can be the historical load data, weather variables (humidity, temperature), season/ time of the year, the day of the week/ hour of the day, or even special holidays or festivals.

3.2. Time series analysis

Time series is a collection of observations recorded over a discrete or continuous period. This dataset can be a list of numbers together with time information where numbers were collected at regular time intervals [43]. Time series analyses explore and extract the set of information from the dataset which is collected over a period. The period between observations is measured at any standard interval like hourly, daily and weekly. A way to analyze these structures is the decomposition of the time series. The time series can be decomposed into four components. These components are named as the trend, cyclical, seasonal and irregular components.

- **Trend:** It is a tendency of increases, decreases variations of the time series over a long time. When the trend is in a decreasing trend (downward trend) from an increasing trend (upward trend), it can be referred to as “changing direction”.
- **Seasonal:** Time series data can be influenced by seasonal factors which are the variations in pattern during the seasons through the year. The influence of these seasonal factors generates a seasonal pattern or periodic fluctuations in the time series [43]. As an example, weather condition, traditional habits can be the vital factors causing this seasonal variation.
- **Cyclical:** It refers that some fluctuations or patterns can be repeated throughout the time series. It means the presence of non-periodic variations which are repeated in cycles. These cycles generally exceed two years [43]. In practice, the above two components, trend and cyclic are combined and referred together as the trend.

- **Irregular components:** The involvement of the random component in the time series is considered as the residuals. It is also called the noise component that represents the random fluctuations at each instant. Due to the presence of these irregular components in time series, a perfect forecast is not possible.

Time series decomposition can be done using two models, namely additive and multiplicative decomposition. Time series can be further categorized into stationary or non-stationary according to the presence or absence of a trend in the dataset. In a stationary time series, the mean and covariance among the observations remain constant over time [44]. If an upward or downward trend is presented in the dataset, the time series is considered non-stationary. The absence of a trend in the dataset makes the time series stationary. Before modelling with non-stationary time series, it needs to be processed. Moreover, the inconsistencies in raw data need to be removed by cleaning and be re-scaled by the transformation process. A way to obtain this processing of time series data can be done by using differencing where it computes the differences between successive observations and create a new time series.

The detection of non-stationarity in time series can be done using autocorrelation function. In simple word, correlation measures the linear relationship between two variables. Two statistical tools known as the autocorrelation function (ACF) and partial autocorrelation function (PACF) can identify this pattern in time series. ACF and PACF can be graphically represented using various statistical software packages. These correlograms are also used to identify the order of parameter range of an autoregressive (AR) model by using the PACF plot, and the ACF plot mostly identifies the parameter range of a moving average (MA) model. These models are described briefly in *section 3.3.1* and *3.3.2. Figure 3.1* and *Figure 3.2* shows the ACF and PACF plots for autoregressive models.

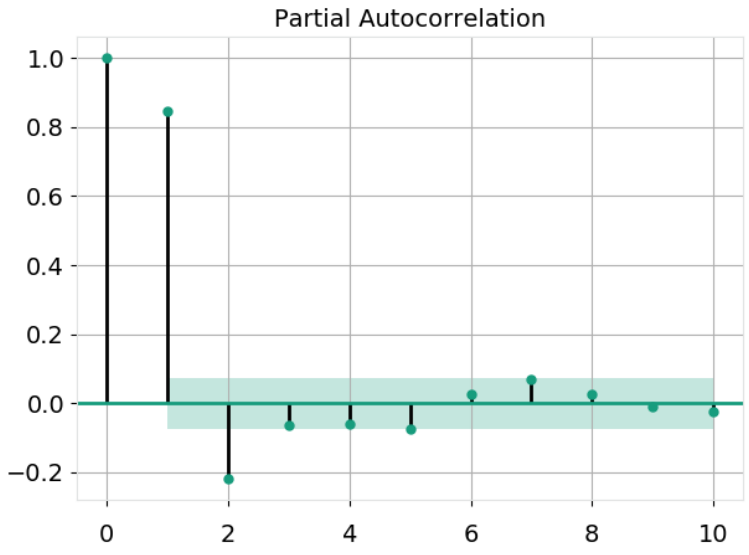

Figure 3.1: The PACF plot of AR (1) model for PV generation.

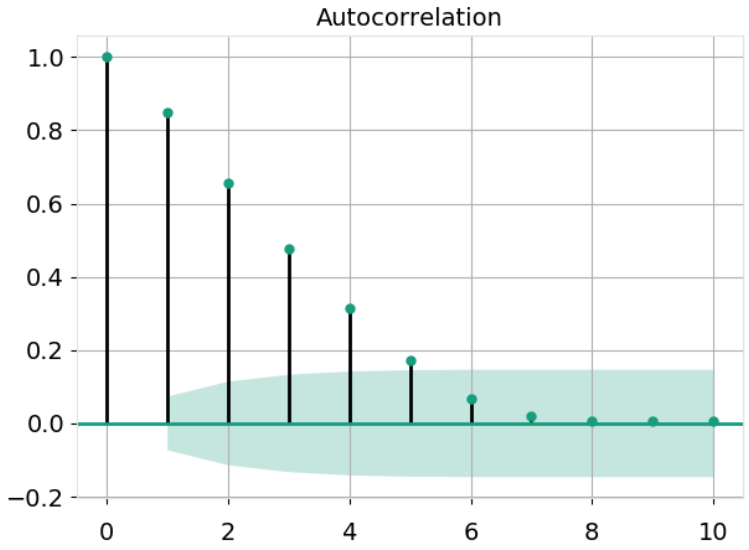

Figure 3.2: The ACF plot of MA (4) model for PV generation.

3.3. Statistical approaches

The forecasting field has been influenced, for a long time, by linear statistical methods such as the autoregressive (AR) model, the moving average (MA) model and some models that derive from them such as ARMA (autoregressive moving average), ARIMA (autoregressive integrated moving average). The use of these models has earned much popularity among the forecasting researchers. These models are sometimes denoted as Box-Jenkins models. Another statistical category is the Exponential smoothing methods, including simple exponential smoothing, Holt-Winters method. Altogether, these models are referred to as ETS model, which is the explicit modelling of error, trend and seasonality. In this section, we will describe these forecasting methods from the book Forecasting: Principle and practice [43] by Hyndman and Athanasopoulos.

3.3.1. Autoregression (AR) model

Autoregression model is used to predict a value from a specific time series which is (auto) regressed on previous values from that same time series. This method is mainly appropriate for univariate time series (where a single observations measurement recorded over equal time intervals) without having trend and seasonality in it. This model uses a linear combination between the current observations and the predicted variable against the past p observations of that variable, with certain error [45].

In the autoregressive process, a predicted output y_t variable linearly depends on its previous values such as ($y_{t-1}, y_{t-2}, \dots y_{t-p}$) as well as on the white noise $\mathcal{E}_t$. This white noise states as a set of identically distributed, uncorrelated random variables through a finite variance σ^2 and zero mean. It is symbolized by $WN(0, \sigma^2)$ [46].

The $AR(p)$ model can be expressed as [43] follows:

$$y_t = \mathrm{c} + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \mathcal{E}_t \quad (1)$$

where $\mathcal{E}_t$ is white noise $WN(0, \sigma^2)$, C and $\{\phi = 1, \cdots, \mathrm{p}\}$ are the parameters of the model which are constants. Depending on the order p, the model decides the number of previous observations to be taken to predict the present value. The simplest example of an AR process is $AR(1)$ is a first order autoregression model, can be denoted by

$$y_t = \phi_1 y_{t-1} + \mathcal{E}_t \quad (2)$$

AR process shows a smooth decay in autocorrelations coefficients. Therefore, identifying the order of the model becomes difficult using ACF. PACF plot can be considered as a solution here because after the lag p, it highlights a cut off [47]. Autoregressive models are normally restricted to stationary data which means some constraints on parameters values are required [43].

3.3.2. Moving average (MA) model

Moving average models are generally used to express single-variable time series [48]. A stationary time series is said to be a moving average process which uses past forecast errors in a regression-like model rather than using past values of the forecast variable. In other words, In the $MA(q)$ process, the component represents an error series of the model as a linear combination in terms of current observation against previous (unobserved) q innovations [45]. The $MA(q)$ model, a moving average model of order q can be written [43] as:

$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q} \quad (3)$$

where $\mathcal{E}_t$ is white noise error term, c and $\{\theta = 1, \cdots, q\}$ are the constant parameters of the model. According to [43], each y_t value here represents a weighted moving average of the previous few forecast errors. A first-order moving average $MA(1)$ can be expressed as:

$$y_t = \varepsilon_t + \theta_1 \varepsilon_{t-1}. \quad (4)$$

By using the ACF plot, the order of the MA model is usually spotted. The ACF plot shows a sharp cut off usually after lag q meaning that for the lags beyond q the autocorrelation coefficients are close to zero. On the contarary, PACF plot shows a gentle decay to zero for moving average process [47].

3.3.3. Autoregressive moving average (ARMA) model

The ARMA is one of the most widely used models which combines the advantages of auto-regressive AR(p) and moving average MA(q) models. That means the autoregressive moving average model $ARMA(p,q)$ is a combination of $AR(p)$ and $MA(q)$ models for stationary time series. The model was originally proposed by Peter Whittle (in 1951) in his paper on "Hypothesis testing in time series analysis" and later, was adapted by George E. [49]. An ARMA (p, q) model of order (p, q) can be written by:

$$y_t = c + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} \quad (5)$$

where y_t is the original series and $\mathcal{E}_t$ is a series of random errors which follow the normal probability distribution, $\{\phi = 1, \cdots, p\}$ and $\{\theta = 1, \cdots, q\}$ are respectively the coefficients of the AR and MA terms.

From the ACF and PCF graphical plots, the order of the model *(p,q)* can be determined. It is sometimes difficult to estimate the appropriate values for parameters p and q value of ARMA model simply using the ACF and PACF plots. In contrast to this, we can rely on the Akaike information criterion (AIC) which select a better fit model for the provided time series. AIC provides the goodness of fit of the model and penalise the increasing of the number of parameters in models. Hence this penalty discourages the overfitting of the model. Therefore, the minimization of the AIC gives the best ARMA model [50]. AIC can be written as [43].

$$AIC = -2log(\mathcal{L}) + 2(p + q + k + 1) \quad (6)$$

Where $\mathcal{L}$ is the likelihood of the data, $k = 1$ if $c \neq 0$ and $k = 0$ if $c = 0$. Here the last term in this parenthesis refers to the parameter number of the model.

3.3.4. Autoregressive integrated moving average (ARIMA) model

Autoregressive integrated moving average (ARIMA) model forecasts variable based on linear dependency to the past values to it. The models we have discussed in previous sections, defined as AR, MA, and ARMA are preferred for stationary time series analysis [51]. However, in real life, time series are mostly non-stationary. To fit stationary models, it is essential to get rid of the variation of non-stationary sources in time series [52]. One solution to this, Box and Jenkins [53] introduced the ARIMA model which can effectively transform the non-stationary data into stationary by introducing a differencing process and overcome the limitation [52] [44].

In ARIMA models, the initial step is to eliminate this non-stationarity using differencing. It is done by subtracting a current observation from observation at the previous time step. As an example, a first-order differencing can be done by replacing y_t via $y_t = y_t - y_{t-1}$. By this procedure, the stationary model that is fitted to the differenced data must be summed or integrated so that it can provide a model for original non-stationary data. Therefore, the ARIMA model is called "Integrated" ARMA. The general form of the ARIMA(*p, d, q*) process is described as [43]:

$$y'_t = \mathrm{c} + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} \tag{7}$$

where y'_t is the differenced new series (after the subtractions) The right-hand side "predictors" include both lagged errors and lagged values of y_t. $\mathcal{E}_t$ is $WN(0, \sigma^2)$, $\{\phi = 1, \cdots, \mathrm{p}\}$ and $\{\theta = 1, \cdots, q\}$ are respectively the coefficients of the AR and MA.

Because of combining these components, we get many complicated models, but at the same time, it becomes much easier to work with the backshift notation. For example, Equation (11) can be expressed in backshift notation as [43] follows:

$$\left(1 - \phi_1 B - \cdots - \phi_p B^p\right)(1 - B)^d y_t = c + \left(1 + \theta_1 B + \cdots + \theta_q B^q\right)\mathcal{E}_t \tag{8}$$

When the number differencing is recognized, then the procedure to estimate the value of model parameters (*p, q*) is similar to Akaike's Information Criterion (AIC) process that has described in ARMA model [43].

In short ARIMA (*p, d, q*) model can be summed up as:

- AR: Autoregression model uses the dependent relationship between observation and lagged observations. *p*: The number of lag observations.
- I: Integrated to make the time series stationary by differencing of raw observations. *d*: The number of times that observations are differenced.
- MA: This model uses dependency between an observation and a residual error from a lagged observation applied moving average. *q*: The size of the moving average window or moving average order.

3.3.5. Exponential smoothing model

Exponential smoothing (ES) method can be considered as weighted moving averages. It was introduced by Brown's (1959,1962) and Holt's (1960) research work that was aimed at stock control system forecasting. The weighted average from past observations are taken here. In this method, the past observations are weighted with exponentially decreasing ratio, and in contrast, the most recent observations are associated with the largest weights. Because of its suitable adaptation in terms of short-time forecasting, this performs well in daily basis predictions.

Single exponential smoothing

This is the simplest exponential smoothing, also known as SES, is a forecasting method for data with no clear trend or seasonality. SES exponentially decreasing weights are assigning to past observations over time. In SES the estimation of future values is done using the forecast built for the observation of the previous period and combining it with the forecast error. The formation [43] of SES can be written as follows:

$$\begin{aligned} \hat{y}_{T+1|T} &= \alpha y_T + \alpha(1-\alpha)y_{T-1} + \alpha(1-\alpha)^2 y_{T-2+\cdots} \\ F_{t+p} &= L_t = \alpha y_t + (1-\alpha)F_{t-1} \end{aligned} \tag{9}$$

Where $y_{T+1|T}$ is the new forecast, $0 \leq \alpha \leq 1$ is the smoothing parameter, y_T is the last actual value, y_{T-1} is the last forecasted value and so on. The parameter α, also known as the smoothing factor, controls the rate at which the weights decrease. For one step ahead time $T+1$, the forecast is the weighted average of all observation in series $y_1, \cdots, y_T$. This series only contains the information of level values. In time period t, y_t is the new observation (or actual value) of the series, F_{t+p} is the new forecast value (or smoothed value) for next period p step ahead and F_t is the old forecast (or smoothed value) at t.

This level equation updates the level by integrating information from the most recent data point (y_t) and updating the previous level at time $t-1$ (can be expressed as L_{t-1}). In SES, the value for α needs to optimize so that it minimizes the error of forecasting. Large values for α (i.e., close to 1) refers more weight is specified to most recent past observations which are paying main attention to the model. On the other hand, smaller values for α mean for making the new forecast close to the old forecast where the main observation has a small impact on it.

3.3.6. Holt-Winters model

In contrast to most of ES method, the Holt-Winters smoothing method considers seasonality or trend. This method was extended by Holt (1957) and Winters (1960) to capture seasonality and trend by considering the smoothing factors, the trend (a slope) and seasonality (cyclical, repeating pattern) into account. These three aspects of the time series (value, trend, and seasonality) are referred to as triple exponential smoothing.

The Holt-Winters' adaptive model allows the level, trend and seasonality patterns to change corresponding on time. Two variations of this method are used in the time series data depending on the seasonality components. One of them is the additive method which is used when the series seasonal variations are roughly stable. The other one is the multiplicative method which is chosen when seasonal variations are changing corresponding to the level in the series.

The equations below are for multiplicative methods, consisting smoothing constant for the data α ($0 \leq \alpha \leq 1$) for new smoothed value ℓ_t, the trend smoothing constant β ($0 \leq \beta \leq 1$) for trend estimate b_t, and seasonal soothing constant γ($0 \leq \gamma \leq 1$) for the seasonality estimate s_t [43].

$$\begin{aligned} \ell_t &= \alpha \frac{y_t}{s_{t-\mathrm{s}}} + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\ b_t &= \beta(\ell_t - \ell_{t-1}) + (1-\beta) b_{t-1} \\ s_t &= \gamma \frac{\mathrm{y_t}}{(\ell_t + \mathrm{b}_{t-1})} + (1-\gamma) s_{t-\mathrm{s}} \\ F_{t+p} &= \ell_t + b_t + \mathrm{S}_{\mathrm{t+p-s}} \end{aligned} \tag{10}$$

where at time t, ℓ_t is the estimated level of the series, b_t is the estimated trend (slope) of the series, s_t is the estimated seasonal component of the series, and y_t is the new observation or actual value at period t in the series. F_{t+p} is the forecast for p periods into future by updating level, seasonality, the trend with previously observed ℓ_t, b_t and s_t. The frequency of the seasonality (the number of seasons) is denoted by s, as an example it may be s =4, (for quarterly data) or s =12 (for monthly data). For time t, the level equation presents a weighted average between the non-seasonal forecast $(\ell_{t-1} + b_{t-1})$ and the seasonally adjusted observation $(y_t/(s_{t-\mathrm{s}}))$. The trend equation that used here is the same as to Holt's linear method.

The additive method is quite similar to the previous one; the only difference is in the seasonality which is subtracted [54].

$$\begin{aligned} \ell_t &= \alpha \frac{y_t}{s_{t-m}} + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\ b_t &= \beta(\ell_t - \ell_{t-1}) + (1-\beta) b_{t-1} \\ s_t &= \gamma(y_t - \ell_{t-1} - b_{t-1}) + (1-\gamma) s_{t-m} \end{aligned} \tag{11}$$

This also has the same initialisation values calculation as a multiplicative method for L_s and b_s, but for S_s the initialization value is estimated as follows:

$$S_1 = Y_1 - L_1, S_2 = Y_2 - L_2, \cdots, S_s = Y_s - L_s \tag{12}$$

3.4. Machine learning approaches

While considering the machine learning approaches, we have considered the ensemble learning method Random forest (RF) and the nonlinear approach K-nearest neighbor (KNN) into account. This section describes the working procedure of these machine learning algorithms for time series forecasting.

3.4.1. Random forest model

Random forest is an ensemble learning method which creates many decision trees and then combines their predictions [11], whereas ensemble methods combine the results from a group of weak models and improve the results to form a powerful model. The principle of this method is to combine the set of binary decision trees referred to as CART (Breiman's classification and regression tree [55]). Therefore, an individual tree in random forest model is constructed using a bootstrap sample randomly chosen at each node, combined a subset of learning points and a feature (predictors or input variables) subsets at each node [37].

Bootstrap is the method that performs a random sampling with replacement (resample). It is a process of getting data from a big dataset with replacement. As every tree constructed by bootstrap is randomly chosen different dataset, so it eliminates the bias and it can test the stability of a solution [11].

In terms of binary decision tree, each internal node has exactly two outgoing edges (left child and right child), and for applying to the incoming data, each split node contains a test function. The final test result is stored in the final nodes of the tree, referred as leaves. Such a decision tree CART is used in the RF method to resolve both the classification and regression problems

[55]. From randomise bootstrapped data the subset of data sets is built along with the decision tree for each dataset. Then the final decision is obtained (bagging) by combining the ensemble, which is done by averaging the output in regression case or classification case by voting.

In research, Breiman pointed out that by adding more trees RF model does not overfit and gives a limiting value in RF generalisation errors which is calculated by an out-of-bag (OOB) error. OOB score is the custom validation method of random forest which is basically out of bag prediction. The out of bag samples are the training points that are not contained (about one-third of points are left out) in each bootstrap training set. This one-third unexploited training data (out of bag samples) can be used to testing. Thus, the accuracy of the random forest model is estimated by the proportion of out of bag samples that are correctly classified by the model. On the other hand, the proportion of out of bag samples which are classified incorrectly by the random forest model is OOB error. The estimation of this error is as similar to acquire by N-fold cross-validation [37]. When the OOB error is stabilised, the training can be ended. Two important features describe the RF model that is the out-of-bag error and the measure of variable importance. RF algorithm for regression [56] is shown below:

1. For $k = 1$ to K:
    1.1. Draw a bootstrap L of size N from the training data.
    1.2. Grow a random-forest tree T_k to the bootstrapped data, by recursively repeating the following steps for each node of the tree, until the minimum node size m is reached.
        1.2.1. Select F variables at random from the n variables.
        1.2.2. Pick the best variable/split-point among the F.
        1.2.3. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_k\}_{k=1,2,\dots,K}$.

To make a prediction at a new point **x:**

$$f(x) = \frac{1}{K}\sum_{k=1}^{K} T_k(x)$$

Two main parameters are needed to implement a random forest model [57]: the number of trees (ntree) in the forest K and the number of input variables (features) randomly chosen (mtry) at each F. The successive trees are added during the training procedure until the OOB error is stabilized [37]. Various studies have specified that satisfactory results could be gained with the default parameter value [58] [59]. Mainly, the low sensitivity to parameter values, ability to generalization, and built-in cross-validation are important advantages of the model [37]. This method is not affected by outliers for the decision tree and bootstrapping construction.

3.4.2. The K-nearest neighbor model

The k-nearest neighbor (KNN) was invented by Fix and Hodges Jr [60], afterwards formalises for classification tasks by Cover and Hart [61]. In general, the KNN is a developed version of instance-based learning algorithm based on the difference between features in the labelled dataset.

It searches for a group of k samples which are nearest to unknown samples based on distance functions. A labelled dataset bunch are used here for finding the k most similar instances which are nearest to the new data point. Thus, the algorithm gives the prediction based on how similar the new incoming observations are to the training observation. During the learning process, this algorithm retains the whole training set. From these *k* samples, the label (class) of unknown samples (the new input data) are compared to each instance in training set and determine the prediction of that unknown samples by calculating the average of the response variables [62]. Therefore, for a regression case, this can be the mean output variable, and for the classification task, it can be the most common class value.

For a given instance X, to compute $\hat{Y}$, KNN model uses the k closest instances in the training set and the prediction is the average of the corresponding targets. The model can be written as the simplest way:

$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i \tag{13}$$

Where $N_k(x)$ is the set of the closest points in x_i training sample. Determining the *k* value is difficult as the parameter *k* goes to 1, at the same time the error on the training set goes to 0 but the error on the test set starts increasing. This is due to the KNN model has low bias and high variance. To identify the closeness and measure the distance *d* between two data points, similarity metric (some distance functions) is used. A most common function here is Euclidean distance that measures the distance between x and y points as follows:

$$\mathrm{d}(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{14}$$

Some pros and cons of KNN are listed in the research work [63], in case of load profile forecasting KNN is good for a day or longer time. It has highly adaptive behaviour because of using local information. It also has some limitation due to large historical data storage required to create a database for k-nearest neighbour search.

3.5. Deep learning approaches

A biological neural network has inspired to the Artificial neural networks (ANN) where the outputs are found for the given data input using an approximation function. The idea behind ANN is to reduce the involvement of human brain functionality for resolving different problems. Just like the human neural network, the input information is linked to output information via a network of neurons. In this section, Artificial neural network and Recurrent neural network are discussed.

3.5.1. Artificial neural network (ANN) model

In simple words, ANN can be pictured as interconnected neurons which pass the information between layers. ANN is constructed with three layers: the input layer, hidden layer, and output layer. Input layer includes all the input variable known as input neurons. Similarly, the output layer contains all output variables or neurons. The layers staying in between the input and output layer are referred to as hidden layers containing hidden neurons.

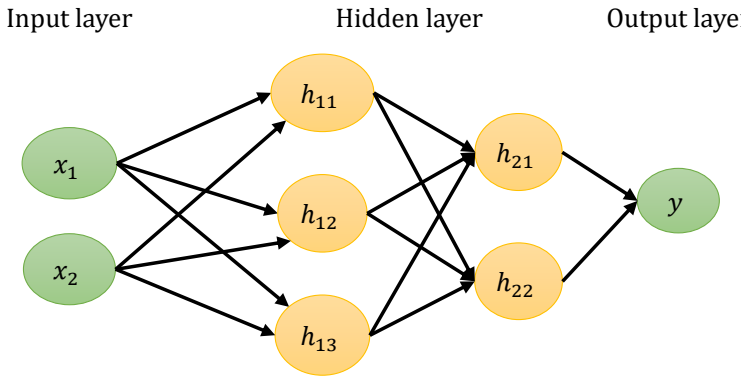

Figure 3.3 Example of ANN with two hidden layers.

Depending on the complexity of the forecasting problem, the hidden layer can have one or more layers. The more hidden layers and neurons, the network realise more complicated relationship. In the ANN model, each neuron is connected to the neurons of the previous layer and contains a nonlinear activation function. Every neuron processes the activation of previous layer neurons with a certain weight to its activation. In the ANN model, neurons in hidden layers process and adjust each input multiplying by the received signals with certain weights. Thus, these weighted signals are summed up and passed the cumulative signal an argument to an activation function [64]. Similarly, the activation process is done for all the layers successively to process the output activation. Some numerical weights are set to the connections by the learning from previous experience along with the current condition [65]. Thus, by adjusting the weights of the neurons (input variable), the desired output for given inputs set can be obtained.

ANN output can be computed through the mathematical expression given below [66]:

$$y_t = \alpha_0 + \sum_{j=1}^{q} \alpha_j \, g\left(\beta_{0j} + \sum_{i=1}^{p} \beta_{ij} \, y_{t-i}\right) + \varepsilon_t \tag{15}$$

Where y_t is the output and $y_{t-i}(i = 1,2,\cdots,p)$ are the p inputs to the network, p and q are the number of input nodes and hidden nodes respectively, $\alpha_j(j = 0,1,2,\cdots,q)$ and $\beta_{ij}(i = 0,1,2,\cdots,p;\ j = 0,1,2,\cdots,q)$ are weights to the connection and ε_t is the random shock. α_0 and β_{0j} are the bias terms.

Varies activation functions are used in the activation procedure. The most common functions are the linear, the threshold, and the sigmoid functions [67]. Generally, the logistic sigmoid function is used which can be [68] written as:

$$g(x) = \frac{1}{1 + e^{-x}} \tag{16}$$

The parameter set for each neuron is adjusted through the learning algorithm. Training of ANN contains two phases. First, the evaluation of the derivative for the error function concerning the weights is done, and then the modification of the weights is done. However, different architectures and training algorithms have been presented in previous researches. The fundamental algorithm is called Error Backpropagation by Rumelhart et al. that uses gradient descent for the second phase [73]. Backward propagation of errors or Back-propagation is the algorithm used with an optimization method as gradient descent. Back-propagation has two phases: in forward pass, through the network's layers a set of input (a pattern) is processed and for each time step computes the outputs. The final output vectors along with the partial derivatives of it for weights are returned to the network. Moreover, in case of a backward pass, concerning each weight, the gradient of the error function (an example: sum squared error) is computed and to minimize the error weights are modified with the gradient in downhill direction (negative gradient).

In a feed-forward neural network, the signal flow has one direction only from input to output, one layer at a time, where from the input the transmitted signals are passed via the hidden layers and produce the output. This network can be used for forecasting taking the past observations data of the time series as inputs to the network and the predictions of the future value as output to the network.

The feedforward ANN model (*Figure 3.3)* can perform non-linear functional mapping to the future value from the past observations of the time series which can be expressed as [68]:

$$y_t = f(y_{t-1}, y_{t-2}, \cdots, y_{t-p}, \boldsymbol{w}) + \varepsilon_t \quad (17)$$

Where y_t is the output series being forecasted, f is a function formed by network structure and connection weights, $(y_{t-1}, y_{t-2}, \cdots, y_{t-p})$ are the past values to series, value $\boldsymbol{w}$ is a vector of all of weights and ε_t is the residual. The training of ANNs begins having random weights and the goal here is to adjust them in a way so that for each time step it can minimize the forecasting error. Learning Rules are the optimization techniques used for minimizing the error function. In this context, Backpropagation or Generalized Delta Rule is the best-known learning rule found in the literature [69] [68]. Shah and Trivedi (in 2012), points out the advantages and challenges of ANN in their research work [29].

3.5.2. Recurrent neural network (RNN) model

Recurrent Neural Networks (RNNs) are designed to work with sequential problems. In RNNs, the output layer is added to the next input and feedback into the same layer [70]. This ability to operate with sequences leads this neural network model to a variety of applications. This network is suited for time series forecasting where the output can be either only the next value in a sequence or the next several values. Therefore, recurrent networks are preferred in several studies to investigated short-term power forecasting, and such an example is demonstrated by Kariniotakis [71].

A recurrent neural network has additional arrows forming loops backwards in the architecture so that these connections become the inputs either to the same neuron or to other neurons. This additional loop acquires the information of the previous period and joins to the current input. By this technique, using this simple loop RNN networks remembers the information from previous inputs which has been calculated so far.

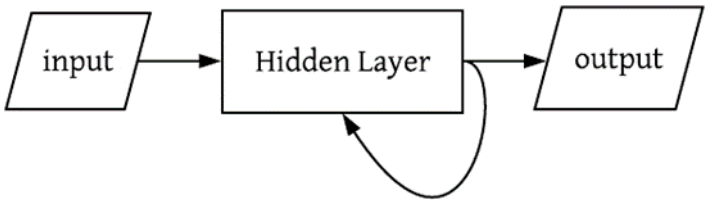

Figure 3.4: A Simple procedure of recurrent neural network.

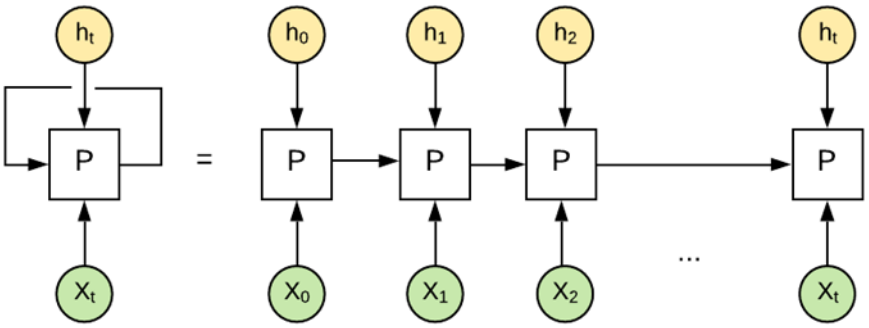

Figure 3.5: An unrolled version of basic RNN

As shown in *Figure 3.4*, this feedback connectivity permits neurons to transfer their signals backward and forward during the learning process until a minimum error is achieved. The output of the network can be used feedback as an input to the network with the next input vector and so on. In this way, RNN networks can think of having a "memory" to the network and allow it to realise broader concepts from input sequences.

Figure 3.5 presents a basic RNN structure. Where x_t is the input and h_t is the output to the network at time step *t*. At first, this network takes the input x_0 from the sequence and provide the outputs h_0 for time step *t=0*. This output along with the x_1 is considered as the input for next time step and so on. RNN keeps memorising the context with recursive function while training the network. The equation of RNN can be expressed as follows [72],

$$h_t = \phi(\boldsymbol{W}x_t + Uh_{t-1}) \tag{18}$$

Where h_t is the hidden state at t timestamp, $\boldsymbol{W}$ is the weight matrix for input to hidden layer at time stamp t, ϕ is the activation function (either Sigmoid or Tanh), U is the hidden layer weight matrix at time $t-1$ to hidden layer at t, h_{t-1} is the hidden state for time $t-1$.

The weights decide the importance of current input as well the importance of previous timestamp hidden state. With the modification of values from the previously hidden state and current state, it decides how much value should be taken for creating current output. This type of network learns these weights (U, **W**) while training by using the backpropagation. Training an RNN is very difficult. It has a major drawback, known as vanishing gradient problem, which cannot ensure high accuracy sometimes.

4. Methodology

This chapter starts with a short description of a common framework which can be treated as a forecasting toolbox to run and evaluate all the methods that have described in *chapter 3*. Afterwards, we have presented the forecasting accuracy measurement criteria that are used in this thesis work to evaluate the performance of individual methods. The implementation procedure with a short description of the coding language, working platform and the important library functions are described in the later section. The working process of the forecasting toolbox is presented briefly step by step at the end of this chapter.

4.1. Forecasting toolbox

The primary purpose of the toolbox is to create all the method instances in the same pattern and get the prediction through the same testing structure. This works as a method comparison toolkit that provides forecasting of each method for a given dataset. The workflow of the forecasting toolbox can be presented as follows:

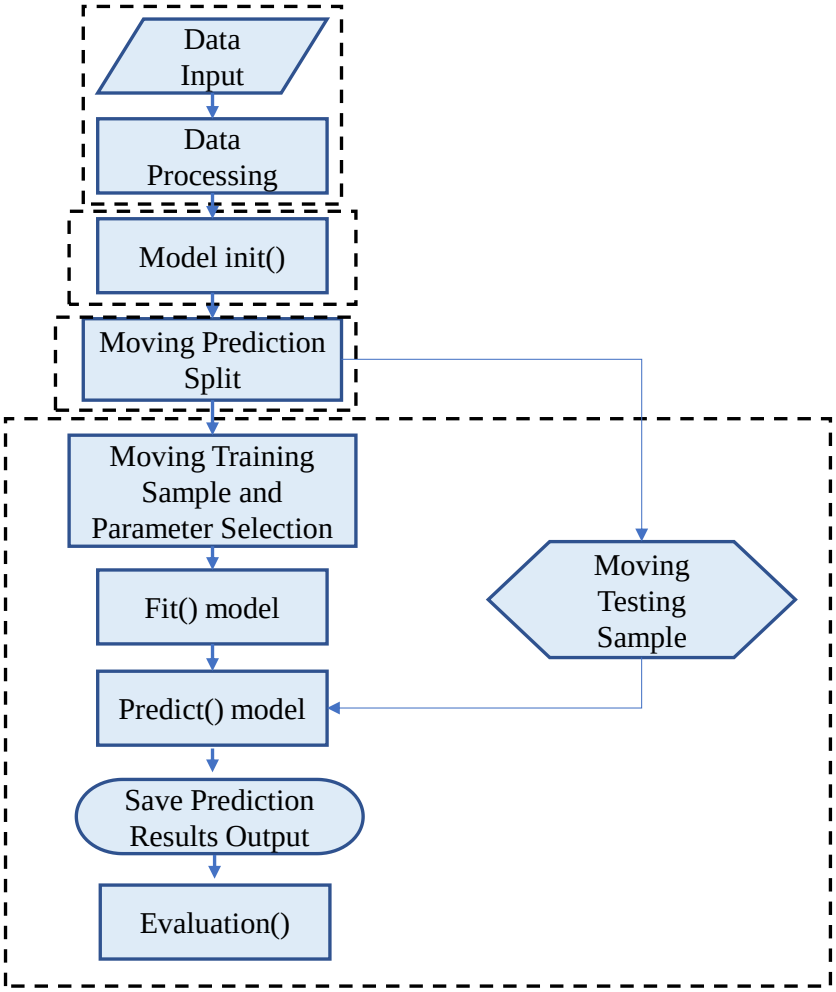

Figure 4.1: The workflow of the forecasting toolbox.

The input data files are read, and the processing of the data sets is carried out. Then, based on exogenous and endogenous (*section 4.2*) input, each method will be initialised. After that, through the moving prediction split mechanism, the dataset will be splitted in to training sample and testing sample for several times. These training samples will be passed to the *Fit()* function of each method along with the optimised hyperparameter value(s) to train the model. The test sample will be provided to the *Predict()* function of the respective method to produce the forecast. Finally, the prediction results are saved in the output file (for example, as CSV format) and the prediction quality is evaluated using the *Evaluation()* function. This procedure of the toolbox has described briefly in *section 4.6.*

4.2. Exogenous and endogenous variables

In a system, the exogenous variable is not affected by other variables. The term "Exogenous" refers to the external variables of a system. On the contrary, an endogenous variable is directly influenced by the system. For example, if we consider the PV generation, corresponding solar radiation have influence and the PV production is dependent on absorbed radiation. So, the solar radiation is the exogenous variable of PV generation forecasting and taken to be input data as well. In this example the endogenous variable would be PV generation time series data.

In case of electrical load data forecasting, no exogenous variable is needed in principle. The statistical approaches work only with an endogenous variable for time series forecasting. Here, the relationship between the endogenous variable and exogenous variable is not needed for a good prediction. However, different regression and machine learning algorithms count on this relationship. Hence, we require both exogenous and endogenous variables for their prediction algorithm structure. So, a dummy hour of the day column is added in the load profile dataset. This dummy data was used as an exogenous variable together with the electrical load time series as endogenous input for machine learning and deep learning approaches.

4.3. Data-sets

For the evaluation of different forecasting methods, each method uses the same time series data sets. The datasets are used in this research can be grouped into two categories based on application areas; the electrical load profiles and the PV generation for households. In the end, we compare the performances of different methods according to the type of datasets. So, all data sets are resampled to have equidistant (1h interval) time steps to compare correctly. Therefore, if some data set is provided with less or more time interval, then it was downsampled or unsampled to get the time interval of 1h to achieve the consistency of the evaluation result. From the dataset, we have taken the input of related exogenous and endogenous variable, and through the toolbox, for different methods get the prediction for the endogenous ones.

4.3.1. Household data sets

The dataset we have used for estimating household PV load has a measured photovoltaic generation (PV) profiles for Rieselfield Freiburg which is scaled to 10kWp and annual sum 2219 kWh for the year 2012. This dataset contains the hourly sum of diffuse solar radiation for the same year measured in j/cm^2which is influenced the generation of PV. The other dataset includes a very economical 4 persons (two workers and two kids) household load from 2012 Load Profile (kW) with annual sum 2219 kWh. This dataset has a measured electric load profile for a four-person household having annual sum 4647 kWh. From the annual sum's comparison of different households, we can observe that the number of inhabitants and their behaviour highly influence the load.

Table 4.1: Overview of household energy load dataset.

Energy data	Minimal load (KW)	Mean load (KW)	Maximal load (KW)
PV generation	0.00	1.50	10.33
Household load	0.034	0.25	2.65

The mean curves of all day for 365 days are presented below. We have shown the plotting of the average hourly load of individual time series for different days of the year 2012.

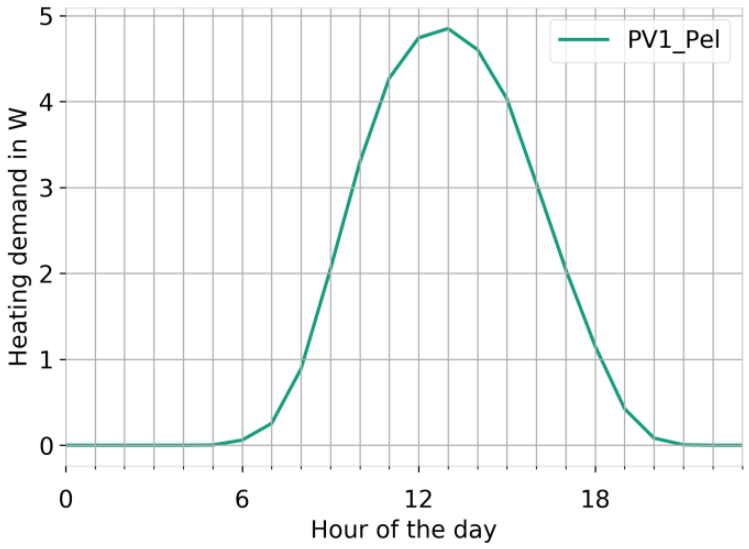

Figure 4.2: Mean day plot for the whole year for PV generation data.

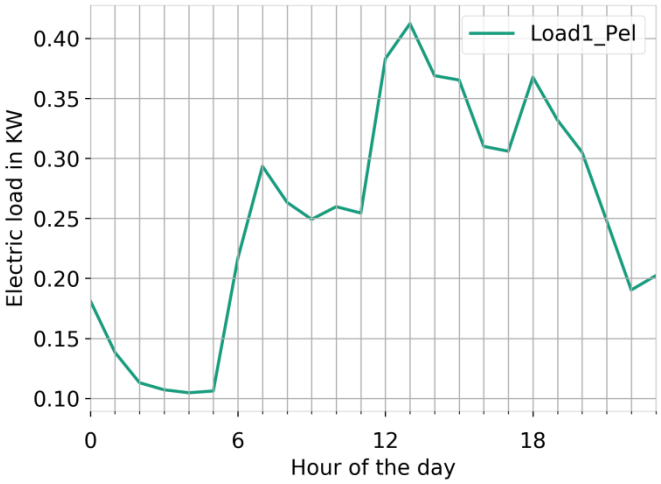

Figure 4.3: Mean day plot for the whole year for electrical load data.

4.3.2. Handling outliers

Outliers are the observation points that are faulty in the data. Usually, these outliers can be detected most of the time easily as they do not make sense in the data. There are several ways to label the outliers in the data by using box plot, scatter plot, z score and IQR score. In this study, we have checked for an interquartile range (IQR score), an outliers detection methods improved by John Tukey (the pioneer of data analysis). By keeping only valid values, outliers are left out, and these missing values can be treated by interpolation of data. Also, the NAN or undefined value from the dataset can be treated by replacing with zero. We have cross-checked with the provided datasets for outliers, NAN or missing values and observed that the data sets were clean. So, we trained our model with these data sets to learn the behaviour of measured data and provide the prediction. Also, from the duration curves of these datasets, we can see no significant effect of outliers for endogenous input.

4.3.3. Training and testing data for each method

According to the statistical literature, the input dataset is usually divided into two sets (training and testing sets). The training set is used for estimating and building the model, also known as “in-sample data”. The testing set is used for the estimation of forecasting using the parameters which is contained the unseen data by the model while it trained. Therefore, the test sets also known as "out-of-sample data”, that is a way to measure its performance of prediction for other data by comparing the predicted outcome with the real outcome of this test data. Typically, the size of the test set is 20% of the last part from the total sample dataset [43]. Sometimes, a validation set is needed to tune the model. As the model perceives this data set so it cannot be a test set. Having a validation set is better for a larger number of datasets.

We have used a moving training technique for “train set” split and moving prediction technique for “test set” split. The whole time series is divided over the different sample sizes several times (discussed in *section 4.6*).

4.4. Forecasting accuracy measurement

Accuracy is the measuring criteria which determine the quality evaluation for forecasting method. The basic behind a good forecast is to minimize the forecasting error as it maximises the efficiency of the forecast. This error is the difference measurement between an actual value and forecasted value. The commonly used error calculating metrices are the root mean squared error (RMSE), the mean absolute error (MAE), the mean absolute percentage error (MAPE) and the Pearson correlation coefficient (*R* value). In this thesis work, these accuracy metrices are used to evaluate the forecasting depending on different forecasting aspects so that we can have a clear view of the efficient method.

4.4.1. The root mean square error (RMSE)

The mean square error (MSE) uses the square sum of the difference between observed and forecasted values and is divided by the number of data periods. The root mean squared error (RMSE) is the square root of this MSE. RMSE is the popular evaluation criterion for regression analysis [73] and can be calculated by:

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(y' - y)^2}{n}} \tag{19}$$

where y is the observed or actual value, y' is the forecasted value and the total length of instances is n. RMSE is measured on the same data scale. This kind of scale dependent measure is not suitable while comparing different data sets. Which implies, the calculated RMSE can be compared only between models that measured from the same data set or in the same units. [74]. Another limitation of RMSE is that it is also prone to forecasting outliers [75].

4.4.2. The mean absolute error (MAE)

The mean absolute error (MAE) is used in the early forecasting literature to evaluate the performance of forecasting models as a primary measurement [76]. The MAE can be calculated by:

$$\mathrm{MA}E = \frac{1}{n}\sum_{i=1}^{n}|(y - y')| \tag{20}$$

MAE is also a scaled-dependent measure and has the same data unit as the original. Hence, calculated MAE can be compared between models whose measured errors are in the same units. However, MAE is also prone to extremly large outliers in data sets but has a slightly smaller magnitude compared to RMSE.

4.4.3. The mean absolute percentage error (MAPE)

Mean absolute percentage error is the scaled independent measure where the accuracy measure scale does not depend on the data scale. MAPE is the average or means of all percentage errors. It is preferable while comparing different data sets [75]. This error (MAPE) can be calculated as follows:

$$\text{MAP}E = \frac{1}{n}\sum_{i=1}^{n}\frac{|(y - y')|}{y} \times 100 \tag{21}$$

However, the limitation of percentage errors is that when data has values equal to zero or close to zero in the target time series, then MAPE can be extremely large or undefined. Moreover, from the research [76] it is noted that MAPE has a bias favouring estimation for the values below the actual values.

4.4.4. The Pearson correlation coefficient

Pearson's correlation coefficient is a statistical measurement which measures the linear relationship strength between paired data of interest. It is presented by Karl Pearson and denoted by r that can take a value range from +1 to -1. Here, a positive value greater than zero means a positive correlation coefficient. Meaning that increasing the value of one variable will increases the value of other variable or vice versa. On the other hand, a negative value of *r* reflects a negative correlation coefficient between those two variables. In this case, increasing the value of one variable will decrease the value for other variable and vice-versa. This Pearson's correlation coefficient *r* between two variables, in our case, the actual value *(x)* and the forecasted value *(y)* can be designed as follows,

$$r = \frac{\sum_{i=1}^{n}\big((x - \bar{x})(y - \bar{y})\big)}{\sqrt{\sum_{i=1}^{n}(x - \bar{x})^2 \ \sum_{i=1}^{n}(x - \bar{x})^2}} \tag{22}$$

where, $\bar{x}$, $\bar{y}$ are the mean of *x* and *y* variable, the value of r, closer to +1(a perfect positive relationship) or -1(a perfect negative relationship), indeed determine how close two variables are correlated. If *r* is equal or close to zero, then it reflects there is no association (relationship) between the variables *x* and *y*.

4.5. Coding language and important libraries

For the implementation of this toolbox, we have used Python 3.6 as a coding language. Integrated development environment (IDE) Eclipse has been used as a development platform. Here, The PyDev extension is aggregated with the default Eclipse to run and execute Python code. Pydev is basically a plugin which enables this IDE to use as a Python IDE. Advanced type inference techniques are used here which provide flexibility such as code analysis and code completion, refactoring, interactive console. Along with other popular Python libraries, we have used Pandas and matplotlib for plotting. To implement all the chosen categories of forecasting algorithm, we have used three main library functions. They are Statsmodel, Scikit learn, and Tensorflow.

Statsmodel Library: Statsmodels is a Python library that built specifically to solve statistical problems. This python library is built on top of NumPy, SciPy, and matplotlib. Statsmodel provides functions and classes for different statistical models including statistical tests, modelling, and data exploration. For the implementation of the statistical algorithms ARMA, ARIMA, ES, and HW these built-in library functions have been used.

Scikit Learn library: Scikit-learn library has a vast range of built-in algorithms for machine learning algorithms through a Python consistent interface. This library is built over the Scientific Python (SciPy) and NumPy. It features various classification and regression algorithms. We have used this library to implement machine learning algorithms such as RF and KNN.

Tensor flow library: TensorFlow is a foundation library created by Google team for numerical computation and deep learning. This is used to create deep Learning models using wrapper libraries or directly. Therefore, it combines machine learning and deep learning (neural network) models. TensorFlow has APIs available in languages (such as Python API) for executing and constructing TensorFlow graph. For the implementation of neural networks ANN and RNN, we have used this library.

4.6. Implementation procedure of the forecasting toolbox

The working mechanism of the forecasting toolbox has several layers. We start with the selection of forecasting method by providing input dataset and finish once we get the output prediction values associated with errors. The working procedure of the forecasting toolbox can be illustrated as following figure:

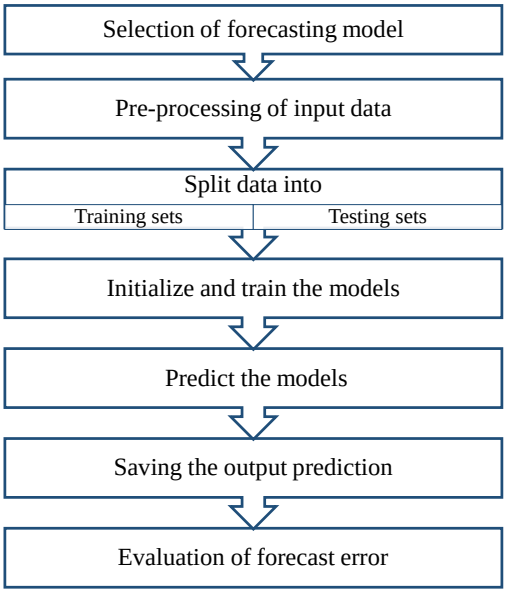

Figure 4.4: Working procedure of forecasting technique.

Read the input file

read_timeseries() function reads the time series data, and it is further stored in variable *ts* as shown below. After that, the prediction variable (endogenous) is set to *values to predict*, and the corresponding exogenous variable is set to *exogenous variable* from dataset. Later, these will be passed to the forecasting toolbox as an argument.

ts= read _timeseries(input_filename)

*values to predict= (*the endogenous variable*)*

*exogenous variable = (*the exogenous variable*)*

Creating the method instance

A list containing all the forecasting method names are created, from where we can select either one single method or all methods together and run the methods for given input dataset. A simple method implementation having endogenous and exogenous input data along with parameter setup to produce the prediction is shown in *Figure 4.5* below:

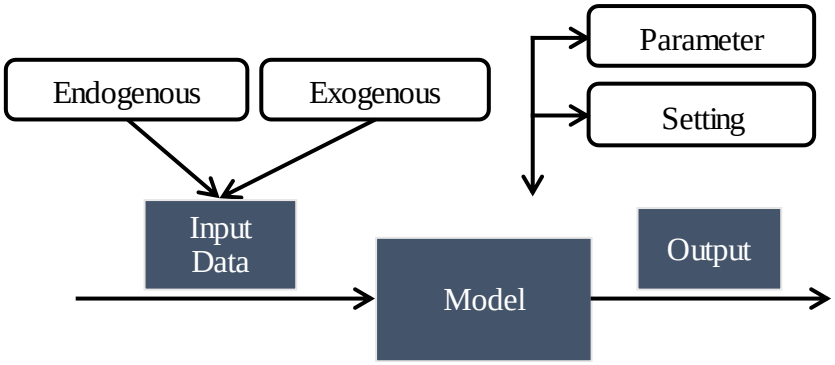

Figure 4.5: Illustration of a method implementation.

After the selection of the method, the instance of forecasting class object is created by the *init()* function from forecasting toolbox. This function takes all the specified values that is discussed before as argument to the forecast wrapper class, which is the master class for forecasting. It is shown below:

PF = Forecast(values to predict, exogen values, method name).

Time scale

Time scale analysis is one of the most important aspects for all kinds of energy forecasting. For our experiment, we have considered five different timescales. They have been categorised as very short-term hourly (1H), short-term daily (1D), weekly (1W), medium-term monthly (1M) and long-term (3M). We will observe that the result accuracy of prediction highly differs corresponding different time scales in the following chapter. We took a list of evaluation horizon which is looped over the method training and prediction.

evaluation_horizon= ['1H','1D','1W','1M','3M']

Sample size

Training set is split over the sample size by moving training technique. We have set the list of the sample sizes (given below) using a "for loop" over the whole dataset. As an example, the first time the training will be done with having 50 data points and each time the same amount of successive data points from the time series are passed through the moving technique to fit a method. The procedure has described briefly in the below section "**Moving prediction"**

samples_sizes= [50, 100, 200, 400, 800, 1600,2200]

Moving training and prediction

We have split the provided time series dataset over the sample sizes and prediction horizons. This is done using a loop over the sample sizes using a *training window* and a *prediction window*. Here, the prediction window is equal to the size of evaluation horizons that is chosen according to the prediction horizon list. For example, if we choose 50 data points for the first run, then from the whole time series the first 50 data points will be chosen as a training set. If we chose the prediction horizon as one day for the same iteration, then the index for this prediction interval will be taken from the time series. According to this prediction horizon, the forecast length *prediction window* is assigned. This will run over the whole time series till the end by splitting the dataset as *ts_train_moving* and *ts_test_moving* just like a moving window.

training window = sample size
prediction window= pred_sample_num

This moving training and moving prediction are illustrated in the below figure

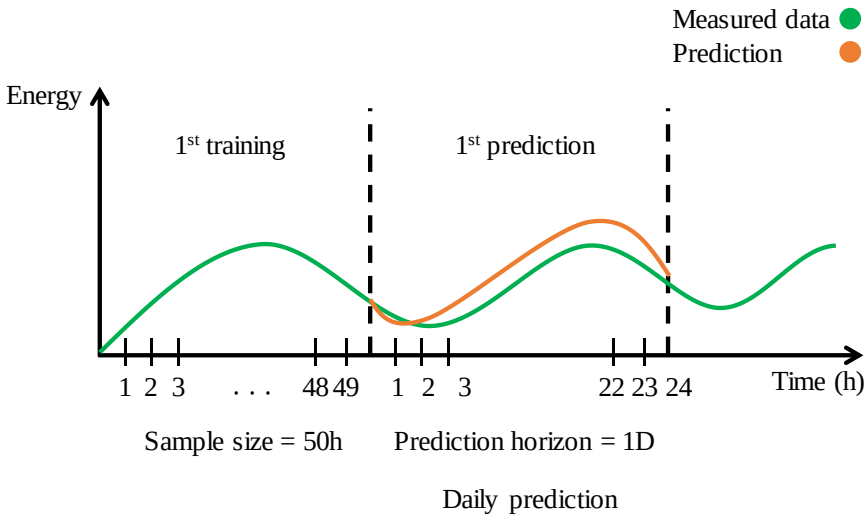

Figure 4.6: Step-1 of moving forecasting for training sample size 50 and daily prediction horizon.

Figure 4.6 shows a moving prediction technique for the measured data (coloured green) and the prediction data (coloured orange). Step-1 shows the first training set (ts_train_moving) with the chosen training sample size 50 and the first testing set (ts_test_moving) for one day ahead prediction horizon. The second step follows the same technique but moving the training set and testing set by 24 (prediction window length for 1D) data points ahead (*Figure 4.7).*

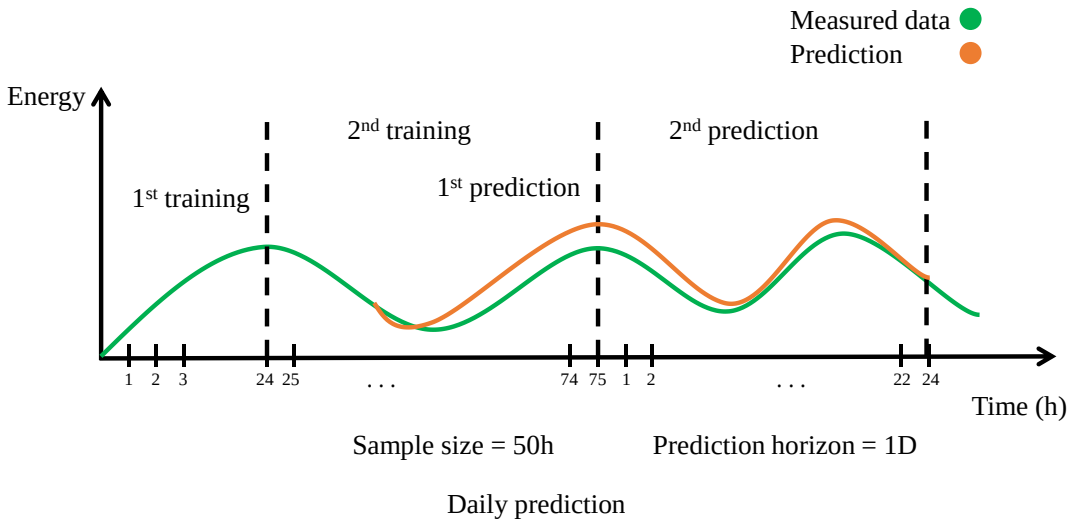

Figure 4.7: Step-2 of moving forecasting for training sample size 50 and daily prediction horizon.

Train the method

Each moving training set in sent to *fit ()* function of the forecasting toolbox. This function will generate the fit function of the specified method. As we have already discussed in the previous section, each method has some parameter of its own. A proper training of the method and a good prediction result is highly dependent on these methods parameter value. We have set this parameter value using some optimization procedures (in *section 5.1*). Along with these optimal parameter values, the endogenous and exogenous input from the toolbox train the method with its own *fit()* function (the training function).

PF.fit(ts_train_moving)

Here we have calculated the computational learning time needed for each method for training by using the date-time differences between the function call and training the model.

Predict the method output

The forecasting of each method is achieved by using the prediction function from the forecasting toolbox. This function will generate the prediction of the specified method with the provided moving test dataset from the toolbox. Here the computational prediction time is calculated using *date-time* differences between the function call and the result generation.

Predicted_result= PF.predicts(ts _test_ moving)

After getting the *Predicted_result*, we have stored this prediction data in the old time series using a new prediction column. Hence, the time series now have both the real value and the prediction value. Later, using the function *plot_all_time_series()*, the timeseries is plotted with real data and the predicted data together.

Evaluation of the forecasting method

The comparison between the true value and *Predicted_result* is done through an evaluation function. All the parameters like method name, dataset type, training sample size, learning time, prediction time, prediction horizon are passed to the evaluation function for each method. All these are passed as an argument to the function and calculate the error metrics (RMSE, MAE, MAPE and R value). The output of the function is saved in another data frame named "store" and save as .csv file.

Store= PF.forecasting_evaluation()

Result plots

Depending on the goodness of fit criteria of each method, result comparison plots are presented in *section 5.2*.

The working flowchart of the forecasting toolbox is illustrated in *Figure 4.8*. The workflow starts by processing the input files and normalizing the inputs. Next, online training is done by moving training and moving prediction. Each method is trained several times. We also applied the offline training by splitting the train set and test set once and trained each method for one time to get the prediction. To fit the methods, we have tuned the parameter of each method according to the provided dataset. Then from the list of the methods, each method is initialized with the optimal parameter values. Then via testing the methods we get the prediction result. These results are stored in the data frame and evaluated using accuracy metrices and plot for each method.

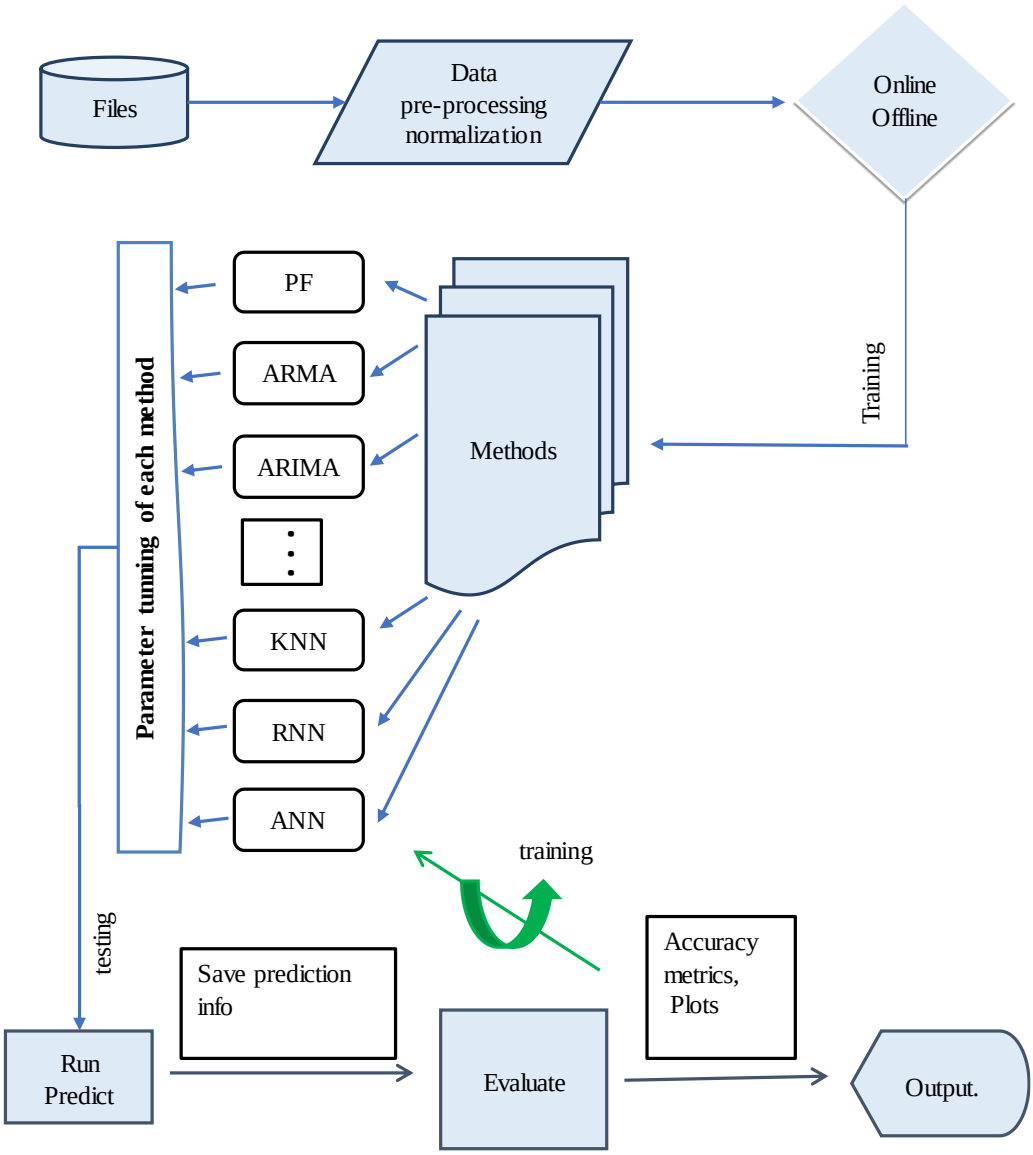

Figure 4.8: Diagram of the forecasting toolbox.

5. Performance analysis

It is necessary to compare all the methods to find an efficient method covering the reasonable forecasting aspects (*section 1.3*). This chapter presents a detailed analysis of the hyperparameter selection and optimization procedure for each method. Afterwards, a computational and error measurements comparison have been described for each method according to the size of the training sample and prediction horizon. Later, based on the finding, a comparison of the methods including the pros and cons is discussed.

5.1. Parameter optimization

In this section, different settings for the critical parameters for each method is highlighted and the optimization procedure of these parameters are discussed.

5.1.1. ARMA and ARIMA

The steps in Box Jekins approach is followed here to choose the optimal value of the parameter for the ARMA and ARIMA method. The identification of the parameter (*p, d,* and *q*) range is done from ACF and PACF plots. Next, the best fit parameter value is estimated by fitting the ARIMA model over the identification range and by using maximum likelihood information AIC criteria. From the set of these candidates of ARIMA models, the best fit model is the one having minimum AIC value. This provided combination with minimum AIC of hyper-parameter will not over-fit the model. The formula of AIC is described in *section 3.3.3*. We further checked the diagnostic of the model to see if the selected parameter values based on AIC criterion is fitted in properly or not. Then the optimal parameter value is used to predict the time series data using the model.

The range of the *p, d* and *q* for PV generation is found at (0,3). Then we checked for the best fit of the model over this range based on minimum AIC. Depending on the normality assumption, AIC will give several minimum values for (*p, q* and *d*) where the least number of AIC is the best fit of the parameter for the model. Alternatively, we can apply the stepwise algorithm auto.arima() library function that is implemented in the forecast library to identify the optimal model parameters. This automated tool stepwise auto.arima is used to identify the subset of predictors where the minimum of AIC estimates the best fit model. This function can identify optimal *p* and *q* parameters and seasonal differencing *d*, but it is computationally slower than the loop search. We compared the optimal value of the hyperparameters generated using auto.arima function and for-loop function over the same range. From this, the most fitted model ARIMA (2,1,2) is found based on minimum AIC=10213.75 for PV data. Some results are listed in *Table 5.1* below:

Table 5.1: Hyperparameter optimization of ARIMA model for PV generation.

ARIMA (2, 1, 1)	AIC=19561.83
ARIMA (2, 1, 2)	AIC=10213.75
ARIMA (2, 0, 2)	AIC=12231.33
ARIMA (1, 1, 2)	AIC=10258.19

Further, we have looked for the diagnostic plot of the ARIMA model to recheck our selection of *p,d* and *q* order. From the autocorrelation plot, we observe that the residuals of the time series have low correlation with itself lagged version. So, we have used ARIMA (2,1,2) and ARMA (2,0,2) model for the prediction of PV generation data.

In the same way, we have checked the optimal value of *p, d* and *q* within the same range for electrical load dataset and got ARIMA (2,1,2) with minimum AIC = 552.90 is the most fitted model parameter value. Finally, we have used ARIMA (2,1,2) and ARMA (2,0,2) model for electrical load forecasting.

5.1.2. Exponential smoothing

As discussed in *section 3.3.5*, the accuracy of the simple exponential smoothing algorithm depends on the smoothing factor α. Several algorithms exist to calculate the best α. However, by comparing a number of values between zero and one, a good value for α can be identified [54]. The traditional optimization method has used to figure out the optimal value for alpha which is based on the lowest mean squared error (MSE).

We separated the dataset into the train and test set to train the model with the train set and to check the model evaluation over the test set. The fractional range of α (0,1) is taken and selected the best α that fits the simple exponential smoothing model based on MSE between the prediction and the true value. The value which gives minimum mean squared error is selected. Training of this algorithm is done using the Statsmodel library function. Then the resulting comparison plot of MSE according to different α value is presented below. The minimum MSE of 5.28 is found for α = 0.038.

Some significant results are listed below:

Table 5.2: Optimal value of α found from loop search.

α	MSE in $(\mathrm{KW})^2$
0.0010	5.86
0.0027	6.08
0.0378	5.28
0.9970	7.07

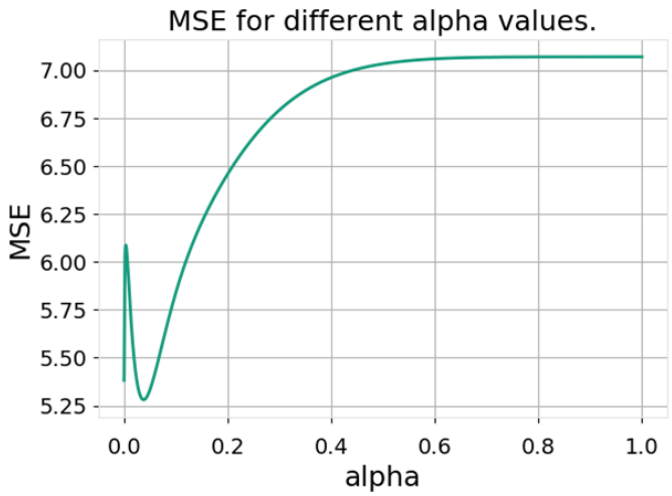

Figure 5.1: The optimal hyper parameter value of exponential smoothing for PV generation.

An alternative time series split procedure from *sklearn* library is used to make the train set and test set split to find the optimal α. In this splitting procedure, the successive training sets maintain sequence order in splitting so that they are the supersets of those order that come before them. We have used the time series cross validation split with 3 folds for the PV generation dataset and fit the model with forecast library function over this train set and test split through these folds. Similar to the first procedure, this train and test sets were looped over the α range.

Table 5.3: Hyperparameter optimization of ES for PV generation.

α	MSE in $(KW)^2$
0.0028	4.50
0.0663	2.10
0.9998	5.55

As the second result evaluation has less MSE error, so we have chosen the smoothing value as 0.0663 to train the model for PV generation and forecast finally.

Similarly, we have checked the optimal α value for electrical load dataset for the range of alpha and got the optimal value α =0.0001 with minimum MSE 0.077. Using the time series cross-validation split, we have got optimal value for α = 0.6367 with minimum MSE= 0.045. So, α = 0.6367 is used to train and forecast the electrical load.

5.1.3. Holt-Winters

The Holt-Winters smoothing method mainly depends on three smoothing factors α, β, γ for the appropriate fit model (in *section 3.3.6*). The range of these parameters are $0< \alpha <1$, $0< \beta <1$, $0< \gamma <1$. We have used similar time series cross-validation split with 3 folds for PV generation dataset. The optimal parameter values were found and fitted the model with forecast library function over this train set and test split through these folds. A detailed procedure of this cross validation has presented in the random forest parameter optimization section. The Holt-Winters model is fitted with the train set and the evaluation of the model is done in the test set. The training of the model is looped over the range α, β, γ ((0,1), (0,1), (0,1)) using *scipy* minimization library function and select the best parameter for the Holt-Winters smoothing model. Training of this algorithm is carried out using the Statsmodel library function. The model evaluation is done using MSE between the prediction and the true value for different α, β and γ value. Based on the MSE of each model, the function returns the score for optimal α, β, γ and the value that results the minimum MSE is selected.

Some significant results in a range $0< \alpha <1$, $0< \beta <1$, $0< \gamma <1$ is listed below:

Table 5.4: Hyperparameter optimization of Holt-Winters for PV generation.

α	β	γ	MSE in $(KW)^2$
0.0027	0.0038	0.2274	1.74
0.0026	0.0037	0.2274	1.90
0.0026	0.0037	0.2275	2.25
0.0018	0.0045	0.2137	2.36
0.0042	0.0036	0.0034	5.08

We got the optimal value for alpha is 0.0027, beta is 0.0038 and gamma is 0.2274 with minimum MSE 1.74 for the model. Further, we used the optimal value for these parameters as smoothing level, smoothing trend and smoothing seasonality respectively to train the model and get the final forecast for PV generation dataset.

The same trend is followed to find the optimal value for electrical load data and we have got the optimal alpha is 0.0028, beta is 0.0021 and gamma is 0.0291 with minimum MSE 0.057. Further, we have used these optimal values to exponential smoothing function to train the model and get the final electrical load forecasting.

5.1.4. Artificial neural network

ANN requires a special pre-processing of the input data. The input vector of continuous variables is normalized using *MinMaxScaler()* in a range (0,1). This results the feature to scale down within the given range. Then, using *fit_transform()* function, each feature is scaled and translated individually in the given range (0,1) over the training set. For processing the output, an inverse transformation of the dependent variable is applied to scale back the data to the original demonstration. The tensor flow library function is used to train the neural network model and forecast the result.

Corresponding to the data shape, the input and output layer shape is set to 1. The optimal weights *W_1* and *W_2* variables are learnt through iterations. For batch optimization, the ANN is trained by gradient descent optimiser and the performance are calculated within the testing part by MSE. Therefore, it calculates the error between prediction and the true value. To activate the neurons in the artificial network sigmoid function is used. The size of the training dataset and the testing dataset is controlled by the forecasting framework. For hyperparameter optimization, we can look at some other important parameters: "epochs", is the maximal number of training round in the network and "hidden number of nodes", is the maximal number of nodes (or neurons) in the hidden layer.

We have compared MSE error in the test part using both one-and two-hidden layers. Interestingly, adding more hidden layers did not improve the forecast quality. After that, different number of nodes in the hidden layer was tested, in a range of (5, 100). We have also checked over the range of epochs (100, 600). The choice for epochs that provided the minimum MSE in the prediction was at 200 and 300. It seemed that it is enough to have 20 hidden neurons in the hidden layer. As adding more nodes to the layer also did not improve the forecast. Some significant result of epochs, hidden layers and hidden numbers of nodes based on minimum MSE is given below:

Table 5.5: Hyperparameter optimization of ANN for PV generation.

Hidden nodes	Number of epochs	Hidden layers	MSE $(\mathbf{KW})^2$
10	300	2	0.895
40	500	2	2.011
30	400	2	2.026
50	700	2	5.394
20	300	1	0.863
50	600	1	5.480
20	200	1	0.896

For the PV generation forecast, we finally used epochs = 300 with a hidden number of nodes 20 for one hidden layer in the ANN network. In a similar trend, for electrical load forecasting, we have got minimum of MSE of 0.140 for a hidden number of nodes 20 and epochs 1000. Here the forecasting window and size of the history data are chosen by prediction horizon and training sample size respectively. This parameter values are used to train ANN and get the forecast of PV generation and electrical load forecasting respectively.

5.1.5. Recurrent neural network

Other than ANN, RNN requires special processing of the input data in terms of the number of batch sizes. To make sure that the same number of observations are passed for each batch of input data, we have set the *N_smaple* size equal to the forecast window. The time series training data is divided into subsets according to this shape, which is split into *x_batches* and *y_ batches* chunk of training data where we have reshaped each batch accordingly. From example, if we pass *prediction horizon* = '1D', then *N_smaple* size= 24 (having 24 data points for hourly interval time series) so that the forecast window will be for next 1D prediction. Therefore, each batch of training data have 24 data points, and the number of batches depending on the provided size of the training sample, where the batch number will be equal to ((*training sample size-N_smaple* size) +1). So, we do not have to optimise the number of samples in training bathes as it will change according to our provided prediction horizon.

For simplification of the model, we have used input neuron as 1(that is the feature) and output as one so that the output to be the same format as input. This ensures easy comparisons of loss functions. Here mean square error (MSE) is used to calculate the loss function. For activation of the neuron, Rectified Linear Unit (Relu) is used. Adom optimiser, which is a general-purpose optimiser, is used to optimise network input and output for training pattern in the tensor flow. Similar to ANN training data set size, prediction window size and *N_smaple size* is controlled by the forecasting framework, so for hyperparameter optimization is applicable for "epochs", hidden number of nodes, hidden number layers and the learning rate.

We have chosen the number of hidden node range (10, 100) and check with one hidden layer and two hidden layers. Also, we have checked over the epochs range (100, 1000) with learning rate= (.01,.001, .0001). After the completion of learning iteration, for each network combination, the MSE error measure was checked. The choice for epochs that provided the least MSE in the train part was at 600 and with one hidden layer having 70 hidden number of nodes the network showed seemed sufficient as it provided less MSE in the testing part. Adding more hidden layers does not change the forecast quality in the network. Learning rate 0.001 was satisfactory to provide a good forecast for all except weekly prediction horizons. For a weekly prediction, RNN needed a different parameter setting. A good prediction for training sample size 200 and 800 was achieved having a higher learning rate of 0.02, 0.1 with epochs

150 and 200 respectively with hidden nodes=50. For the other prediction horizons, some evaluation over the range of epochs, hidden layers and hidden node numbers based on minimum MSE is presented below.

Table 5.6: Hyperparameter optimization of RNN for PV generation.

Hidden nodes	Number of epochs	Hidden layers	MSE $(KW)^2$
70	500	2	1.07
50	400	2	1.24
80	600	2	1.15
30	400	1	1.11
70	600	1	1.06
50	400	1	1.09

In case of electrical load dataset in a similar way, we have got the minimum MSE = 1.130 with learning rate= .001 and epochs= 400. Finally, we set the optimal parameter values to train the RNN and get the forecast of PV generation and electrical load forecasting.

5.1.6. Random forest model

An ensemble approach random forest algorithm was described briefly in *section 3.4.1*. The procedure of this model is not overly sensitive for the parameter variable. Breiman [57] stated it might be unnecessary to use more than the required number of trees, but this will not damage the model. In addition to it, Feng [77] stated that RF could achieve accurate results with ntree = 200. According to many research, $m = 5$ is recommended as smaller m value gives deeper tree. However, to find the optimal RF model for regression, a range of values were tested and evaluated. The algorithm was trained with *RandomForestRegressor()* function which is implemented in Scikit learn library. The main parameters for the good fit of this model are:

- ntree: the decision tree numbers in the forest, and *max_depth* is the maximum leaves number in each decision tree.
- *max_features*: the number of variables should be chosen from available variable in the dataset when looking for the best splitting at each node, the ‘auto’ selection of *'max_features'* means taking all the features for regression analysis and the selection of sqrt(n_features) for *'max_features'* is basically used for classification problems.
- *min_smaple_leaf:* the minimum sample number or data point are required in newly created leaves.

- *min_sample_split:* the number of minimum data points must be placed in a node before the split of that node. The split point of any depth will expand the nodes until all leaves have less sample than *min_samples_split.*

```
param_grid = {
        'bootstrap': [True],
        'max_depth': [80, 90, 100, 110],
        'max_features': ['auto', 'sqrt']
        'min_samples_leaf': [3, 4, 5],
        'min_samples_split': [8, 10, 12],
        'n_estimators': [100, 200, 300, 1000]
      }
```

The optimization function GridSearchCV is used to find out the best model using parameter grid dictionary defined above. Also, time series cross-validation function from Scikit learn is used to compute the cross-validation over the trainset. Cross-validation is an approach to estimate the performance automatically for a model. It works by splitting the dataset into subsets with the provided *k* number of folds, in our case k = 3. This 3-folds split the training set into n (here 3) parts. Then training is done for (*n-1)* folds and the testing is done on the left-out fold. For each fold, the accuracy of cross-validation is measured, and the parameters are estimated in-sample.

We passed the train set to fit the *Randomforestregressor()* and tuned the parameters using grid-search over the param grid and recorded the accuracy for each combination based on the determination of coefficient R^2. The hyper parameter having the highest average accuracy through the n-folds is selected which minimizes the risk of overfitting the model. The optimal parameter value is chosen based on the maximum accuracy of the model. We use grid search over random search for parameter optimization of a random forest as the run time of randomised search is drastically lower but both produce same space of parameters.

```
RandomForestRegressor(bootstrap=True,
                      criterion='mse',
                      max_depth=80,
                  max_features='sqrt',
                      max_leaf_nodes=5,
                  min_samples_split =10,
                  n_estimators=100)
With best_score 0.7770855240573122
```

The optimal parameter value found from the Grid search for PV generation is listed above. We used these values of the hyperparameter for the *RandomForestRegressor()* model and forecast the PV generation.

In case of electrical consumption data using the same param_grid and the above-mentioned procedure we have got the optimal parameter value as below:

```
RandomForestRegressor(bootstrap=True,
                      criterion='mse',
                      max_depth=80,
                  max_features='sqrt',
                      max_leaf_nodes=5,
                  min_samples_split =10,
                  n_estimators=100)
With best_score 0.379626843183269672
```

Finally, we have used these optimal hyperparameter values for the RandomForestRegressor() model and forecast the electrical load.

5.1.7. K-nearest neighbor

The working procedure and the basis of the k-nearest neighbor algorithm have presented briefly in *section 3.4.2*. This algorithm searches the k-closest training samples in the feature space based on the calculated distance. As a result, the parameter *k* plays an important role in the evaluation of the KNN which implies *k* as the key tuning parameter. So, the goal here is to find the optimal *k* value for the model based on the dataset. For training the KNN model we have used the *KNeighborRegressor()* implemented in Scikit learn library. We have used two procedures to find the optimal parameter, first we provided 3-fold cross-validation with the 80% train test split, and 20% test set to split from the whole year of PV generation data. Then we looped through the reasonable number of *k* in a range (0,50) and used the 3-fold cross-validation to estimate the optimal value of *k* and to record the output accuracy based on R^2 between the feature test set and the response test set. For each iteration of the loop, the KNR is instantiated with *n_neighbor= k* and for each fold, the accuracy is measured.

Within the range from 1 to 50, we have found the optimal k value = 40 with the highest score of 0.692 applying k-fold cross validation.

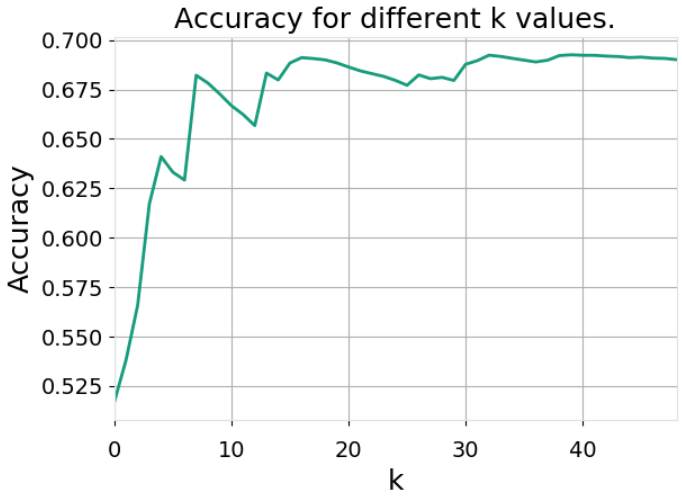

Figure 5.2: The optimal hyper parameter value of KNN for PV generation.

The other procedure is using the *GridSearchCV()* to find the optimal value of *k* for the model. We split the whole year dataset by *traintestsplit ()* as discussed in the random forest parameter optimization and passed the train set to *GridSearchCV()* with fold number = 3 and parameter grid of *k* = [1, 2, …, 50]. The KNN regressor is fitted and tuned using grid search over each value of *k* value in the grid. For each iteration, this will estimate the performance score and the optimal value for the *k* which has the best model accuracy. From the grid search, we get the optimal *k* = 41. The grid search result is presented below:

```
KNeighborsRegressor(algorithm='auto',
                    leaf_size=30,
                    n_jobs=1,
                    n_neighbors=41,
                    weights='uniform')
With best_score_ 0.707
```

The forecasting accuracy is also recorded for the *k* values by calculating the MSE between the true value and the forecasted value. Some significant result is presented below:

Table 5.7: Hyperparameter optimization of KNN for PV generation.

k	MSE in $(KW)^2$
28	0.932
31	0.361
35	0.358
40	0.355
41	0.353

From this table, we see that the grid search provides a similar result as the loop performance just slightly better. However, the computational time is slower than the loop search. In KNN, the higher *k* value produces lower complexity of the model and choosing the odd *k* is better to avoid the class problem. So, we choose *k* = 41 and set as the *n_neighbor* value to train the KNN regressor model and forecast the output for PV generation.

Similarly, we checked for the optimal *k* value for the electrical load data. We have got the *k*=46 from the cross-validation split, but the most optimised value we have got from grid search for the same range of *k* values as mentioned above.

```
KNeighborsRegressor(algorithm='auto',
                    leaf_size=30,
                    n_jobs=1,
                    n_neighbors= 49,
                    weights='uniform')
With best_score_ 0.292
```

So, we have chosen *k* = 49 and set as the *n_neighbor* value for training the KNN regressor model and forecast the output for the electrical load.

5.2. Evaluation

Several common accuracy measures such as RMSE, MAE, MAPE, and R-value have been used to compare the performance of the predictions of each forecasting method against the different forecasting aspects described in *section 1.3*.

5.2.1. Comparison of the computational performance

In this section, we present the comparison of model-dependent parameters, the learning time and the predicting time that is needed to train and forecast a new PV or electrical load profile from the model. These calculations have been carried out on a Windows 10 computer, with 4 Cores, 8GB of ram, and with 3.4 GHz clock speed.

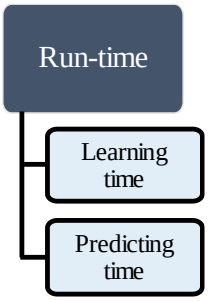

Figure 5.3: Computational comparison criteria.

For simplicity of discussions, we have presented the mean learning time used by each method to train with the optimized hyperparameter values and to predict daily PV generation with a different sample size over the year (in *Table 5.8)*

Table 5.8: Required learning time for each method with different training sizes for daily PV generation.

	Learning time (in seconds)						
	Sample size						
	50	**100**	**200**	**400**	**800**	**1600**	**2200**
ARMA	0.008	0.009	0.009	0.01	0.01	0.01	0.01
ARIMA	0.10	0.11	0.16	0.25	0.38	0.70	0.86
ES	0.006	0.006	0.01	0.02	0.04	0.06	0.07
HW	0.003	0.004	0.006	0.009	0.01	0.02	0.03
KNN	0.005	0.005	0.006	0.008	0.006	0.009	0.01
RF	0.31	0.29	0.30	0.35	0.36	0.47	0.49
ANN	8.79	8.33	8.65	9.08	10.09	12.39	12.24
RNN	30.29	32.98	36.46	41.37	57.34	80.23	101.12

Table 5.8 shows that for all methods the learning time increases gradually with the increase of training size. It can be noted that KNN method has minimal learning time (~0.005 s) over all sizes of training samples for daily forecast. Although, HW has slightly lower learning time over small training size (50,100,200) than KNN, for lager training size the computational time have increased more than KNN. So, the 2nd best choice could be the HW and ES respectively.

ARIMA model needs larger training time compared to the smoothing methods and ARMA. The training time of RF is relatively higher than the smoothing methods. But the longest learning time is required for neural networks, where the learning time is highly dependent on training size. Whereas, RNN requires the longest time for training. The same trend is observed for different prediction horizons such as hourly, weekly and monthly. So, we can conclude that KNN showed better performance according to the learning time comparison.

We compared the mean learning time required by each method for forecasting daily electrical load. The learning time is measured for different training sample sizes. The comparison is shown in *Table 5.9.* From the comparison of mean learning time for electrical load forecasting, we spotted a similar trend as PV dataset. KNN has minimal learning time compared to all other methods over all sample sizes. ES and ARMA could be the second-best choice in this case. However, the largest learning time is needed for deep learning approaches, whereas RNN has the significantly highest learning time compared to all other methods.

Table 5.9: Required learning time for each method with different training sizes for daily electrical load forecasting

	Learning time (in seconds)						
	Sample size						
	50	100	200	400	800	1600	2200
ARMA	0.029	0.034	0.034	0.034	0.036	0.038	0.042
ARIMA	0.09	0.11	0.15	0.27	0.47	0.75	0.95
ES	0.007	0.008	0.01	0.013	0.013	0.02	0.03
HW	0.14	0.25	0.48	0.93	1.50	1.89	2.25
KNN	0.008	0.008	0.009	0.007	0.007	0.008	0.01
RF	0.28	0.35	0.38	0.32	0.40	0.42	0.34
ANN	8.36	8.41	9.22	7.20	7.55	7.45	9.34
RNN	26.97	48.27	70.92	12.13	43.53	55.06	147.3

The forecast phase begins after the training, where depending on the size of the prediction horizon, the forecast has to be repeated until the end of the year. As an example, we have presented the mean predicting time for all prediction horizons for training sample 2200 covering all the methods in *Table 5.10*.

Table 5.10: Required predicting time for each method with different forecasting horizons for PV generation forecasting.

	Predicting time (in seconds)				
	Prediction horizons				
	1H	1D	1W	1M	3M
ARMA	0.003	0.003	0.004	0.004	0.006
ARIMA	0.008	0.01	0.011	0.018	0.036
ES	0.018	0.012	0.011	0.011	0.013
HW	0.105	0.027	0.027	0.027	0.03
KNN	0.008	0.011	0.021	0.038	0.04
RF	0.037	0.035	0.044	0.048	0.042
ANN	-	0.05	0.056	0.057	0.059
RNN	-	5.12	0.44	0.055	0.044

The *Table 5.10* shows that, with 2200 training size, predicting time has increased gradually according to the prediction horizons for all the methods. In this context, the monthly prediction has a large predicting time for all the methods. The predicting time is low for the statistical methods whereas ARMA has minimal predicting time (~0.006s) for all the prediction horizons. KNN has lower predicting time compared to the ARIMA and other methods.

In case of smoothing methods, ES has relatively lower predicting time than HW. RF has shown slightly high predicting rate than all methods, but longest predicting time is required for the neural networks, ANN and RNN respectively. Similarly, for other training sample sizes such as (50, 100, 200, ….), same observations have repeated. From the comparison of the predicting time, we conclude that ARMA has a better performance according to the predicting time

comparison. We further compared the mean predicting time for all prediction horizons for training sample size 2200. The result comparison is shown in *Table 5.11*.

Table 5.11: Required predicting time for each method with different forecasting horizons and training sample size 2200 for electrical load forecasting.

	Predicting time (in seconds)				
	Prediction horizons				
	1H	1D	1W	1M	3M
ARMA	0.01	0.012	0.011	0.013	0.047
ARIMA	0.011	0.012	0.012	0.02	0.04
ES	0.013	0.026	0.025	0.023	0.017
HW	0.03	0.031	0.031	0.031	0.031
KNN	0.009	0.01	0.008	0.021	0.008
RF	0.03	0.032	0.04	0.048	0.041
ANN	-	0.104	0.056	0.057	0.06
RNN	-	2.24	1.62	0.98	1.21

The same trend is followed in the electrical load dataset as well. Whereas, KNN and ARMA have the minimal predicting time for all the prediction horizons respectively. Same as the PV dataset we observed the highest predicting time for deep learning approaches, especially for the RNN.

5.2.2. Result and discussion

We have presented an overview of eight forecasting algorithms: ARMA, ARIMA, ES, HW, KNN, RF, ANN, and RNN. The methods are individually tested for several training sample sizes illustrated as “size of samples” in *Figure 5.4*. As discussed in *section 4.6*, a common framework is used to train and test all the methods using the same pattern. Then, the forecast is done over different prediction horizons using various training sample sizes for PV generation and electrical load usage. Also, the prediction quality of each method is evaluated using the RMSE, MAE, MAPE and R-value for each combination (*Figure 5.4)*.

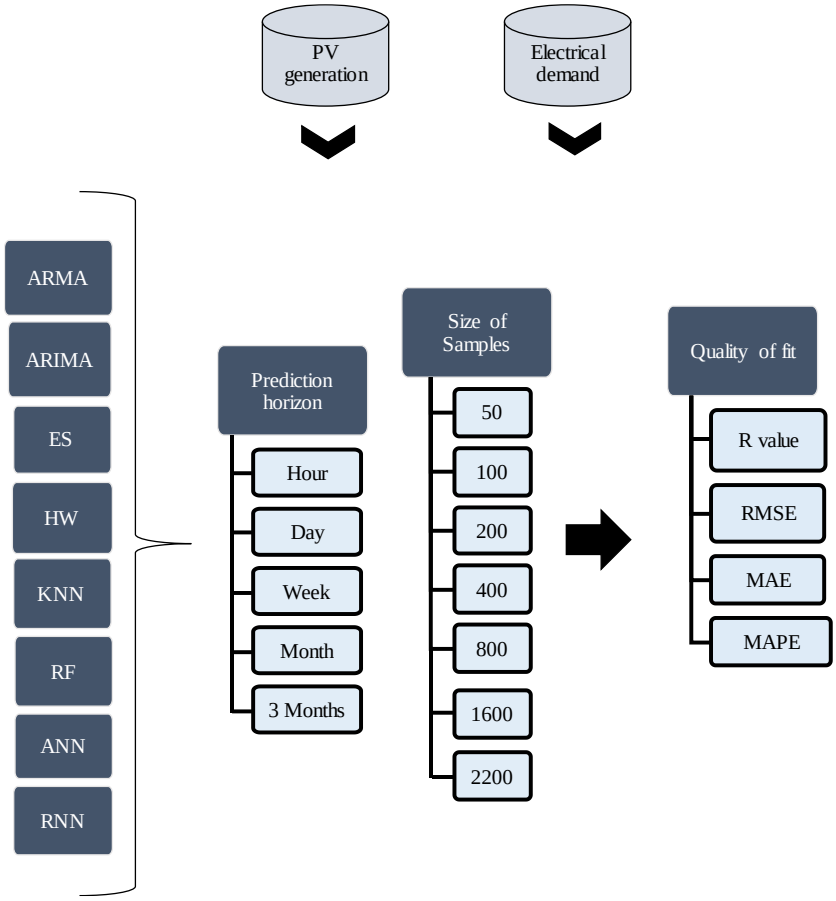

Figure 5.4: Key performance indicators (KPIs) for the evaluation of each method.

Comparing both datasets together and showing total error performances is difficult. In this section, we have presented the result evaluation based on each error measurement separately for each dataset to make it more understandable. So, we have presented the error metrices results separately including all training sample sizes and prediction horizons for each dataset. The relation between prediction horizon and training sample size for different methods according to different performance indicators (RMSE, MAE, MAPE and R value) are presented in this section. To simulate the results visually, we have used some error plots to compare the performances of selected methods.

We have checked two training procedures for the methods; online training where whole data set is splitted into train and test set over the moving training sample for several times and the offline training where the models are trained once with the whole training data by *train and test split()*. The result shows that with online training the prediction gets better. So finally, we compare the result for this online moving prediction.

RMSE comparison:

The first set of analyses examined the impact of RMSE between the forecasted and the actual data for all the methods depending on training sample sizes and four forecasting categories: very short-term: hourly, short-term: daily, medium-term: weekly, monthly and for long-term: three-month duration forecasting of both PV consumption and electrical load separately. Surprisingly, with the increase of the training sample size, no significant improvement is observed in the quality of prediction for some methods.

Closer inspection to RMSE performance metrics of PV generation forecasting showed that (*in Figure 5.5 (a))* for very short-term (hourly) forecasting, the ARIMA and ARMA methods exhibit lower error compared to other methods. For this specific case of hourly forecasting, the ARIMA method has the lowest RMSE 0.84, and it decreases gradually with the increase of the training sample size. The ES method displays the worst RMSE for this hourly forecasting scenario which is greater than 2.0 for all training sizes. The HW method shows the 2nd highest error in this case and the error decreases significantly with the increase of training sample sizes.

Though RF and KNN did not show a comparatively good performance for hourly forecasting, for longer forecasting horizons the error rate is lower than all methods. The RF method showed consistently better performance (low RMSE value around 1.5 to less than 2.0) for short-term, medium-term and long-term forecasting horizon (*Figure 5.5(b) – (e)).* For daily and weekly forecasting, (*Figure 5.5(b), (c)*) the HW method also presented a relatively good performance, just behind the RF and KNN methods. However, the performance of HW method became worse with the increasing of prediction horizons. As seen in *Figure 5.5*(e), for three-monthly prediction, the highest error rate is produced by HW whereas the error is relatively low for other methods.

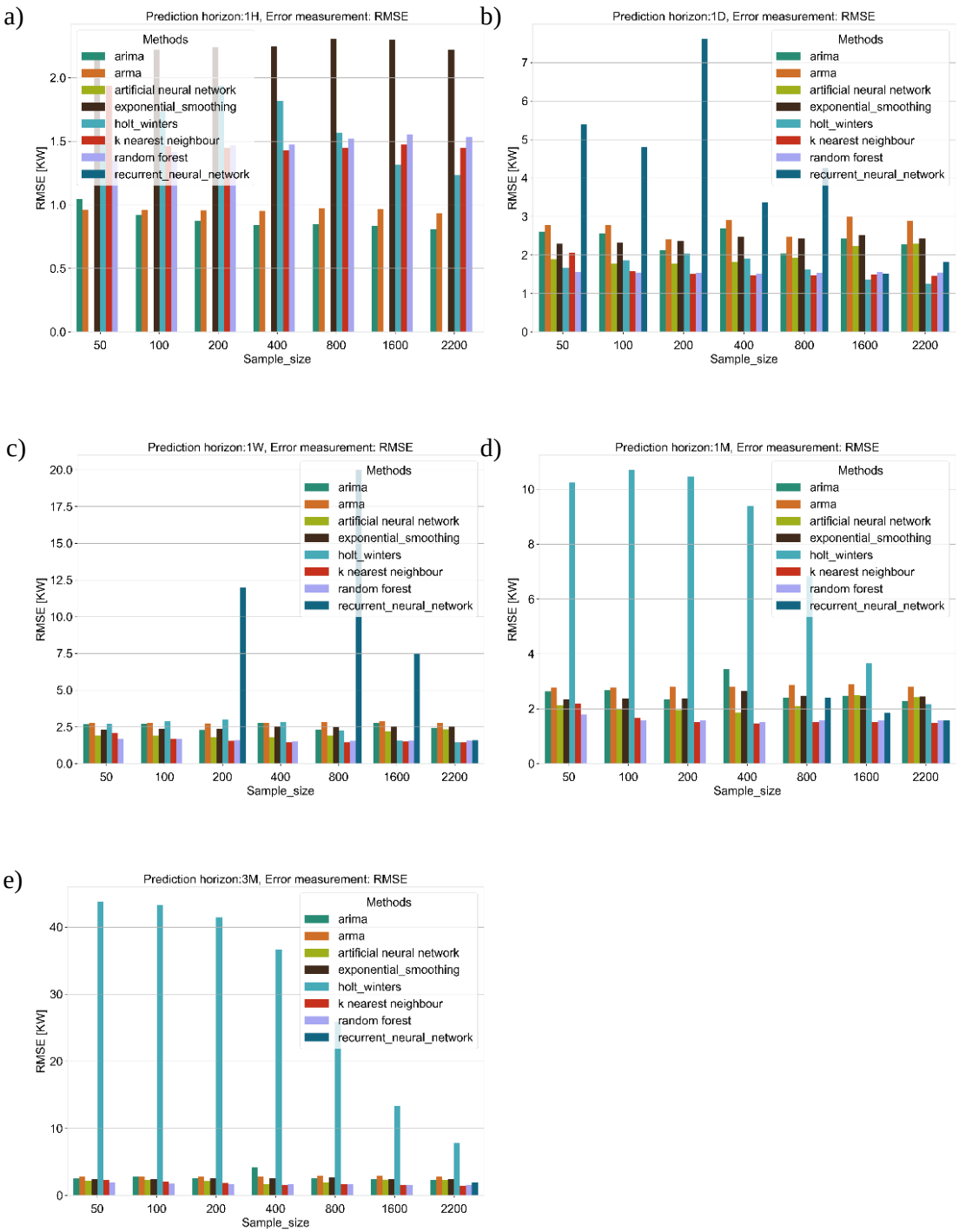

Figure 5.5: RMSE comparison of PV generation forecasting for all the methods depending on different horizons- (a) hourly, (b) daily, (c) weekly, (d) monthly and (e) 3 monthly prediction for all training sample sizes.

Additionally, the deep learning-based approaches performed better than the statistical approaches. In this case, the decrement of RMSE value is quite sharp with the increment of sample sizes. As a longer training time is required for deep learning approaches, so we only observed the performance for daily, weekly and monthly forecasting in this case. However, RNN has an implementation limitation regarding sample size for different prediction horizons.

As a result, we could only achieve the prediction using some specific training sizes for specific horizons. For example, in *Figure 5.5(c),* for weekly forecasting we could not achieve any forecast for RNN with sample size lower than 200. This is due to the fact that the sample size must be equal or greater than the forecast window size which is 168h for weekly horizon. Although RNN showed better performance compared to ANN with a higher training sample size for all prediction horizons, ANN achieves a reasonable performance with smaller training samples, and this is consistent for all prediction horizons.

Figure 5.6 shows the comparison of RMSE performance metrices for electrical load forecasting for five different prediction timelines. RF, KNN, ARIMA, ANN, and RNN presented persistently lower error value for all five different prediction horizons. The prediction of ARMA and ES methods show a higher error for daily and weekly forecasting range (*Figure 5.6 (b, c)*). For long-term prediction horizon, HW displays a very high error value when the training sample size is smaller than 200 (*Figure 5.6 (e)).*

The RNN and ANN exhibit a good prediction for electrical load forecasting compared to PV forecasting (*Figure 5.5*). For electrical load forecasting, the RMSE is consistently lower for all prediction horizons and similar to the machine learning approaches. It can be noted that the performance is slightly better for RNN and the increment of training sample sizes gradually increases the accuracy. Therefore, it is obvious that the forecasting quality highly depends on the historical data. The data sets of the PV generation have some flaws due to exogenous variable (radiation diffusion) outliers; this behavior can also be seen in error measurement and forecasting plots (*Figure 5.5*). This can be contributed to the fact that RMSE is prone to outliers. On the other hand, electrical load forecasting is influenced by the dummy exogenous variable as well which is the hours of the day. For both the datasets (PV and electrical load), again, the RF and KNN persistently achieve good prediction results. But for electrical load forecasting a good accuracy is also achieved by ARIMA, ANN, RNN and with HW for higher training sample size.

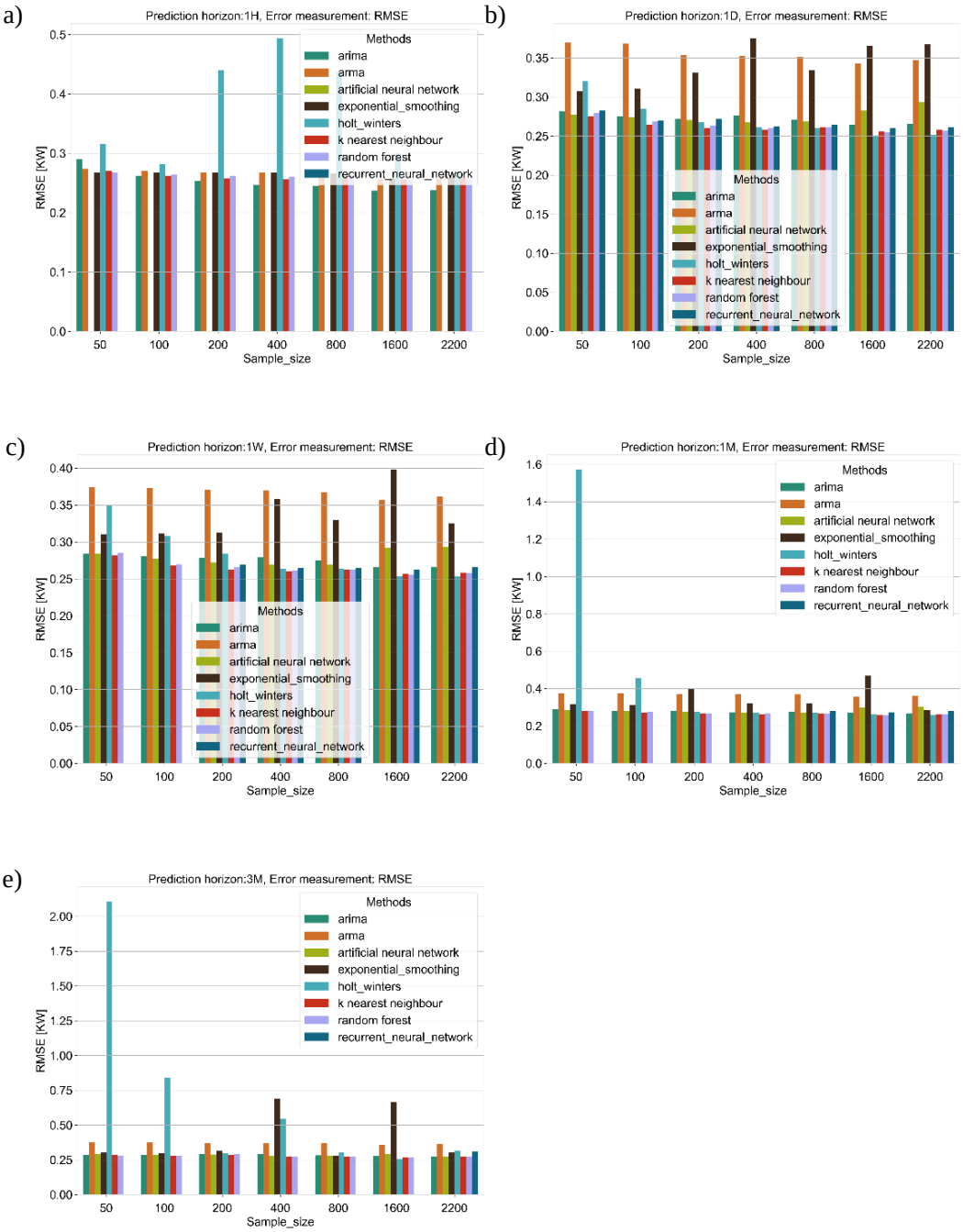

Figure 5.6: RMSE comparison of electrical load forecasting for all the methods depending on different horizons- (a) hourly, (b) daily, (c) weekly, (d) monthly and (e) 3 monthly prediction for all training sample sizes.

MAE comparison:

The second comparative analysis is presented based on the error measurement of MAE value between the forecast of the methods and the actual data. This comparison is also performed for all the methods depending on all training sample sizes for all prediction horizons. Firstly, in *Figure 5.7* the comparison for PV forecasting is presented.

MAE comparison in *Figure 5.7* shows that, for very short-term hourly prediction horizon, the ARIMA and ARMA methods showed low error value close to 0.5. Though, with the increase in prediction timeline, the error increases significantly for both methods (*Figure 5.7(b-d)*). For daily prediction, the error value is the highest for the ARMA method which rises even more with the larger sample size (*Figure 5.7(b*)). For this forecasting horizon, the HW presents relatively better performance than ES. The HW method operates better forecasting for short-term (daily), medium-term (weekly) but for a long-term forecasting (monthly, 3 monthly) the performance declines. Although a sharp decrease in the error trend is observed for HW with the increase of sample sizes, it produced the highest error for long-term prediction for all sample sizes (*Figure 5.7*(*e*). RF and KNN regularly performed well for daily, weekly and monthly prediction and were not largely affected by the sample sizes, which is an advantage. It is noticeable that RNN has a high-performance dependency on the training sample size. It is better evident in *Figure 5.7 (b)* and (c) that for a daily and weekly prediction for some smaller training sample sizes, the RNN method produced a high error value. However, with the higher sample size of more than 800, MAE is relatively low and close to the RF and KNN. The RNN method achieves more accuracy than ANN for daily, weekly and monthly prediction horizons.

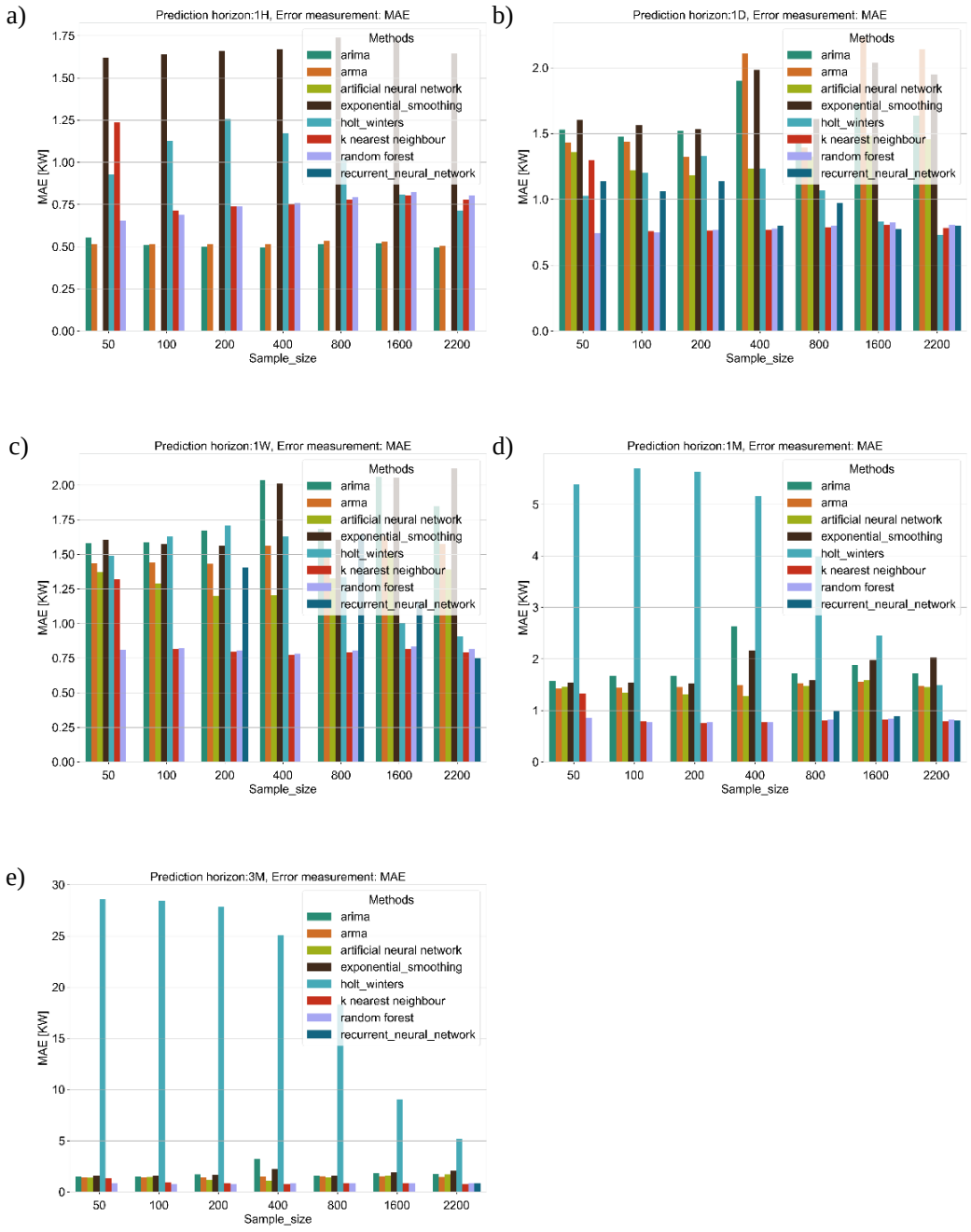

Figure 5.7: MAE comparison of PV generation forecasting for all the methods depending on different horizons- (a) hourly, (b) daily, (c) weekly, (d) monthly and (e) 3 monthly prediction for all training sample sizes.

Figure 5.8 presents the electrical load forecasting performance for various methods by comparing the MAE value. For very short-term (hourly) forecasting in (a), the ARMA method exhibits lower error (MAE 0.27) compared to all other methods. The error value decreases gradually with the increase of the training sample size. In case of smoothing methods, the ES achieves good performance, whereas the HW method attains the highest error for this hourly forecasting scenario with some specific training sample sizes.

Like, RMSE comparison, RF and KNN did not show a comparatively good performance for hourly prediction. From daily and weekly prediction (*in Figure 5.8 (b), (c)),* it is observed that ARMA and ES methods predicted with higher error compared to the other methods over all sample sizes. Here, we noticed a gradual decrease of error with the increase of sample sizes for the HW and the accuracy is higher with the larger training sample sizes for all horizons.

While other approaches struggle to keep the error rate low, the RF and KNN exhibits steady error for all forecasting horizons with all different sample sizes. RF and KNN methods show a steady low error performance for short-term, medium-term and long-term forecasting horizon (in *Figure 5.8 (b) – (e)*). On the other hand, the deep learning approaches perform better than the statistical approaches. RNN had performed slightly better than ANN for all prediction horizons and keep the error value close to machine learning approaches

In the overall performance comparison, it is apparent that the mean absolute error for RF and KNN has the lowest value for the PV generation and electrical load datasets for short-term, medium-term and long-term forecasting. For both datasets, the statistical approach (ARIMA, ARMA) delivers lower errors for very short-term (hourly) forecasting than the machine learning approaches. The deep learning approaches could not beat the machine learning approaches but have a close error comparison with some specific training sample sizes, where RNN showed a high error dependency according to training sample size.

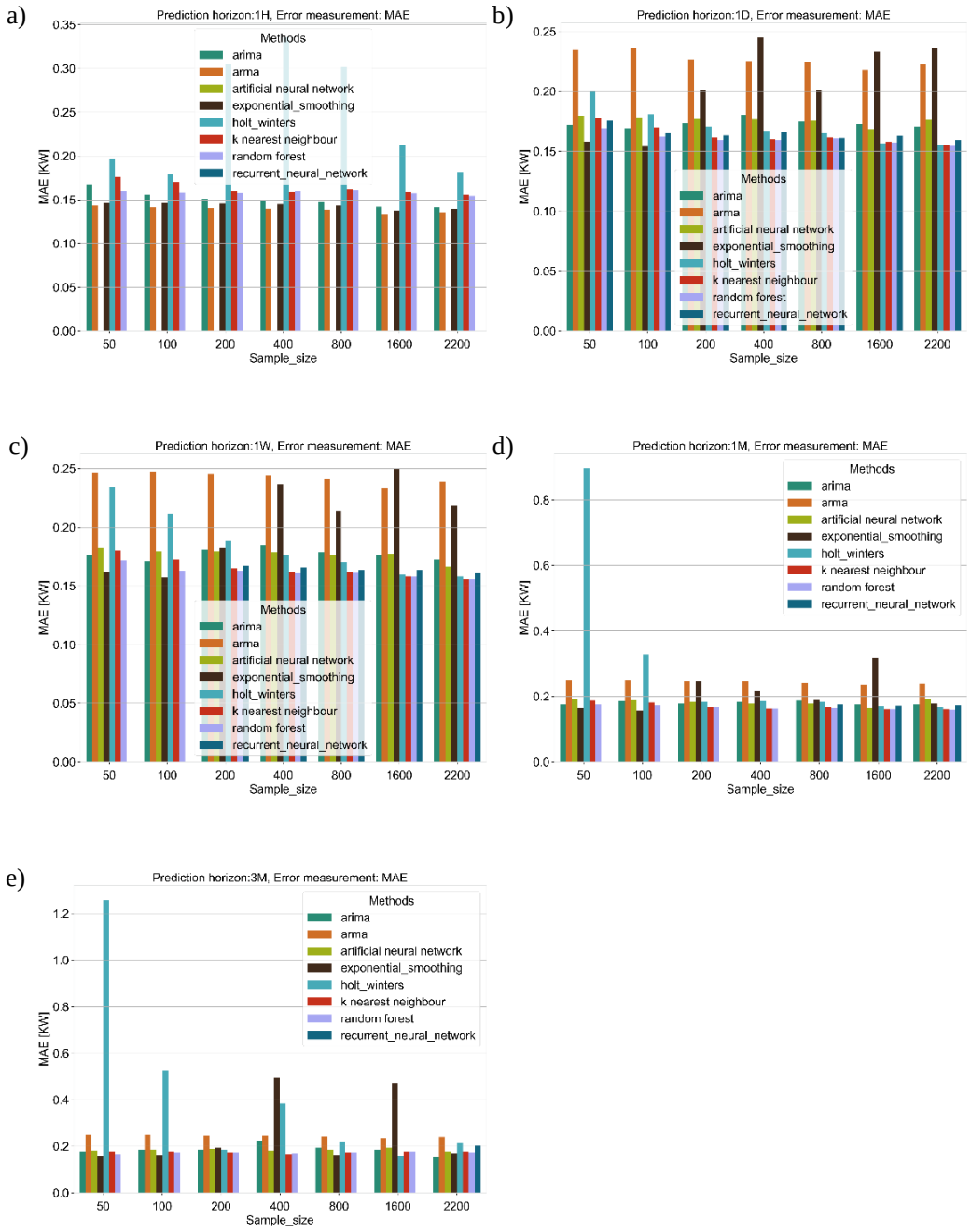

Figure 5.8 MAE comparison of electrical load forecasting for all the methods depending on different horizons- (a) hourly, (b) daily, (c) weekly, (d) monthly and (e) 3 monthly prediction for all training sample sizes.

Correlation coefficient comparison:

In *Figure 5.9,* the correlation coefficient R value for PV generation forecasting is presented. The R-value is also calculated for five different forecasting horizons from hourly to three monthly (*Figure 5.9(a-e))* for various training sample sizes. A high value of R signifies a better correlation between the prediction and real data. For very short-term of hourly prediction, the ARIMA and ARMA methods show noticeably higher correlation value compared to others (*Figure 5.9 a*). The smallest correlation is produced by the ES method. Other methods like HW, KNN, RF generates R-value just below ARIMA and ARMA.

In case of daily prediction (*Figure 5.9 b*), the performance of the ES method is somewhat similar as in the hourly prediction with R-value around 0.3. In this scenario, the HW, KNN and RF methods repeat their better performance as before in hourly forecasting. However, the ARMA method showed very less correlation (low R) for training sample larger than 1600h.

For weekly prediction (*Figure 5.9 c*), better correlation (large R) from KNN and RF is again repeated. The ARMA exhibits lowest R-value in this case. The better forecasting from the KNN, RF, and RNN is persisting also for the longer-term prediction of monthly and three-monthly (long-term) duration. Interestingly, we can see negative value of R (*Figure 5.9 e*) for ARIMA in terms of long-term (3 months) prediction while the sample size is 100. This follows very less negative correlation for ARMA and ES with the increase of sample size (*Figure 5.9 e*). This negative R-value implies that the prediction result is negatively correlated with the actual value.

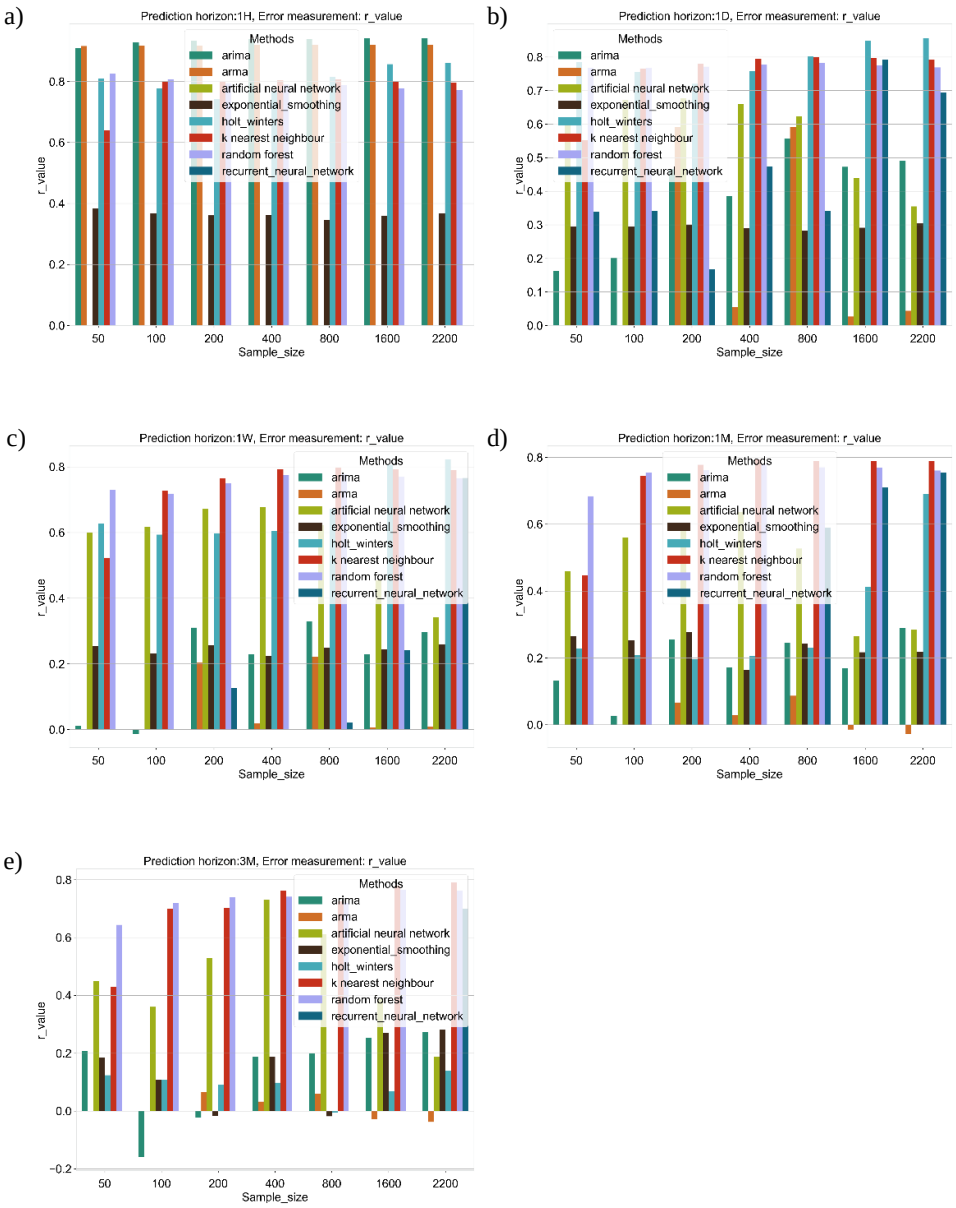

Figure 5.9: Correlation coefficient comparison of PV generation forecasting depending on different horizons- (a) hourly, (b) daily, (c) weekly, (d) monthly and (e) 3 monthly prediction for all training sample sizes.

The correlation coefficient comparison for electrical load forecasting and real value is presented in *Figure 5.10.* The results of the hourly forecast (*Figure 5.10(a))* reveal that ARMA and ES have the highest correlation in hourly forecasting, whereas RF and KNN have relatively low correlation than these methods. HW gives minimum correlation compared to other methods for some training sample sizes (*Figure 5.10(a*)).

For daily prediction (*Figure 5.10 (b)*), ES seems to have a low R-value, whereas ARMA exhibits a negative correlation for some training sample sizes. In this context HW, KNN, RF and RNN showed comparetively high correlation for daily prediction. The same trend is followed for weekly forecasting (in *Figure 5.10 (c)*), whereas ARMA has a relatively low and negative correlation and ES has the smallest R-value.

For monthly prediction, R-value for ES varies significantly with the sample size changes whereas HW seems to perform steadily for medium range sample sizes (*Figure 5.10(d)).* Here, RF and KNN showed a better correlation. For other prediction horizons (*Figure 5.10(b-e))*, we observe a sharp fluctuation especially in case of ARMA. Comparing to PV generation forecasting, we get significantly more negative R values for electrical load data which can be occurred due to the sparsity of data. Like the PV data, we get more of these negative R when we predict for long-term (*Figure 5.10(e)).* Following, in monthly prediction KNN provides slightly larger R-value compared to other methods. RF and KNN has performed consistently higher correlation for medium-term (monthly) and long-term (3 monthly) forecasting.

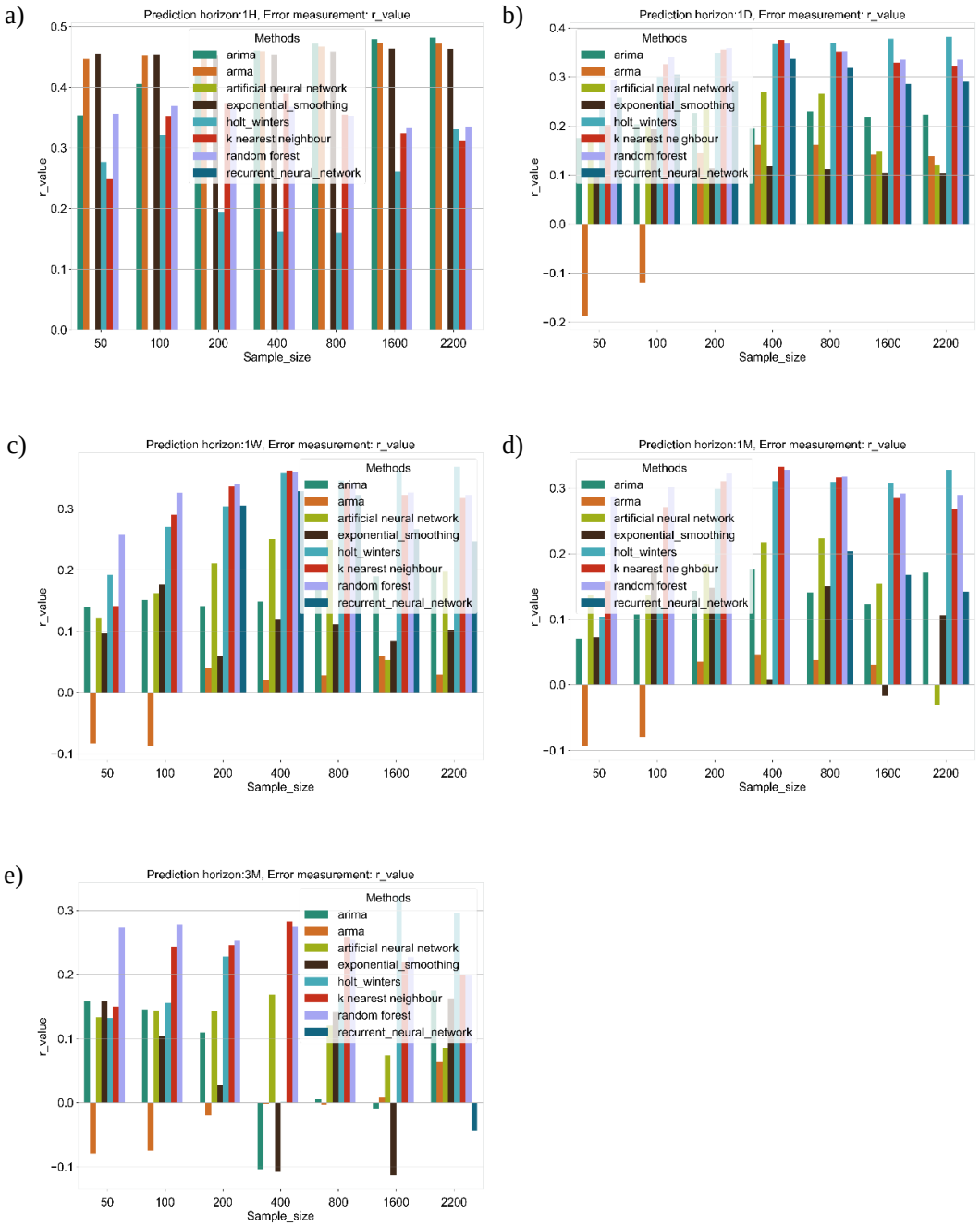

Figure 5.10: Correlation coefficient comparison of electrical load forecasting depending on different horizons- (a) hourly, (b) daily, (c) weekly, (d) monthly and (e) 3 monthly prediction for all training sample sizes.

MAPE Comparison

The MAPE is not a perfect way while comparing the forecast with actual value close to zero as the computation becomes unstable. In case of PV generation, as we have some actual value close to zero, so MAPE becomes infinity. However, the algorithm performs more accurately for the electrical load forecasting.

The performance comparison of MAPE for electrical load forecasting is presented in *Figure 5.11*. It shows that for hourly prediction horizon (*Figure 5.11 a*) ARIMA and HW has the highest MAPE value, whereas for daily forecasting horizon (*Figure 5.11 b*) ANN and ARIMA, HW has the higher value respectively. The lowest error is achieved by RF, KNN and ARMA respectively for all prediction horizon (a-e). ES has relatively low MAPE than HW for weekly (*Figure 5.11 c*) and monthly (*Figure 5.11 d*) prediction horizons. In the overall performance comparison, it becomes apparent that the mean absolute percentage error for RNN has a lower value than ANN for the electrical load datasets. Like, previous error measurements, machine learning approaches delivers lower errors than other approaches in this scenario.

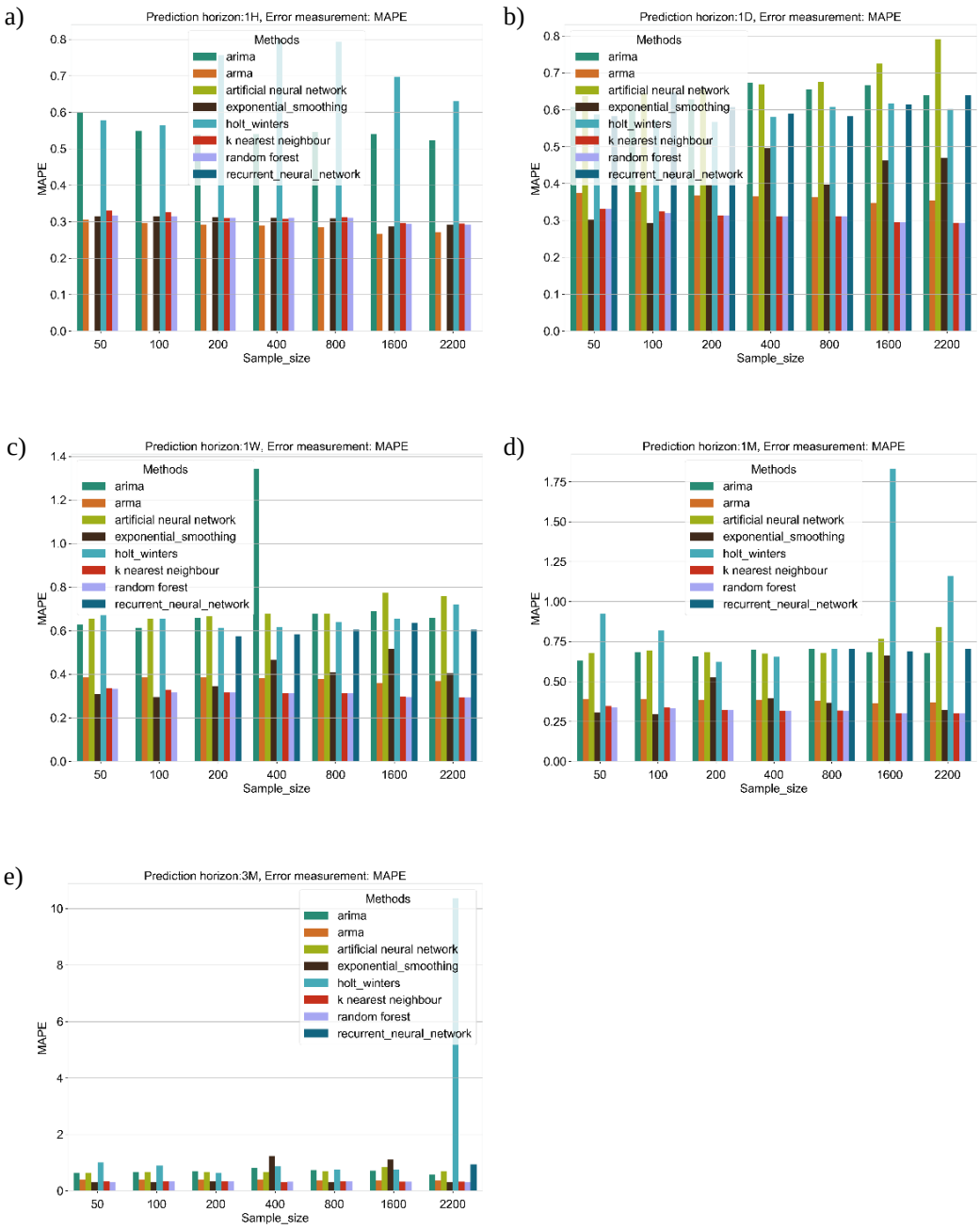

Figure 5.11: MAPE comparison of electrical load forecasting depending on different horizons- (a) hourly, (b) daily, (c) weekly, (d) monthly and (e) 3 monthly prediction for all training sample sizes

5.2.3. Performance summary

The diversity of these forecasting methods in the case of PV generation forecasting is observed in *Figure 5.12*. This shows a ten-day time series comparison of all the methods highlighting the variety of the hourly forecasts.

The RMSE comparison of PV generation discussed in *section 5.2.2* (RMSE comparison) shows that for very short-term (hourly) forecasting, ARIMA and ARMA methods perform best while for the other prediction horizons, RF and KNN present consistently low error value for other prediction horizons as expected. The performance for ARIMA is better than for ARMA due to the advantage of working with a non-stationary dataset (*Figure 5.12 (a)*). For hourly prediction, ARIMA method exhibits the lowest error as per our expectation. Contrary to the HW method, ES does not consider trend or seasonality in the time series. As a result, the prediction by the ES method has higher error values (*Figure 5.12 (b)*) than the HW. In case of daily forecasting, the results of ARIMA, HW and RNN comes after RF and KNN. Both the HW and RNN show a large dependency on the training sample size. The RMSE for electrical load also shows that RF and KNN provide better forecasting for a prediction range of daily or longer.

As we have discussed before (in *section 3.3*), statistical approaches relay only on the past value of endogenous variables for forecasting. But machine learning and deep learning approaches consider both the exogenous information and the endogenous variables together. Due to the availability of this additional information unlike statistical approaches, machine learning and deep learning approaches showed better performance for short-term, medium-term, long-term prediction. Moreover, the RF has built-in cross validation and bootstrap sampling methods to balance the errors in data sets. The RF method also has low parameter sensitivity and thus it achieves good performance with only a few training sample sizes (*Figure 5.12 (c)).* On the other hand, KNN uses local information yielding a largely adaptive behavior of the dataset. Also, from our experiment we have seen better consistency in the performance (in *section 5.2.2*). In the case of electrical load forecasting (presented as hourly prediction for seven days in *Figure 5.13)*, the performances of all methods are similar to the PV generation.

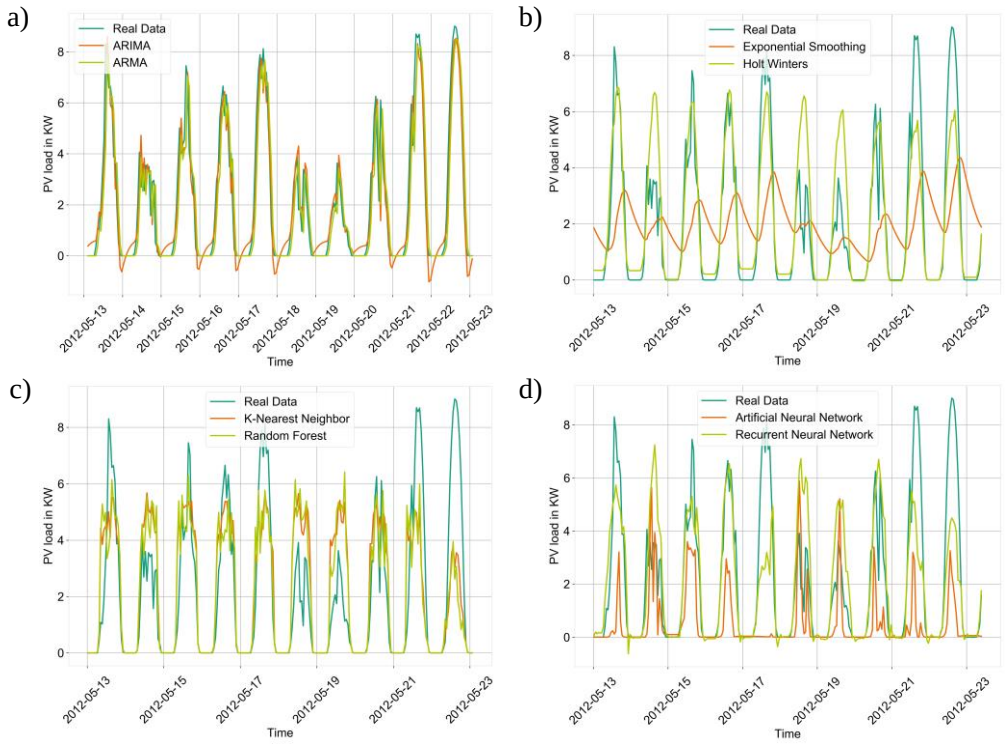

Figure 5.12: Hourly forecast and actual load comparison of PV generation for (a) ARIMA and ARMA, (b) ES and HW, (c) KNN and RF, and (d) ANN and RNN

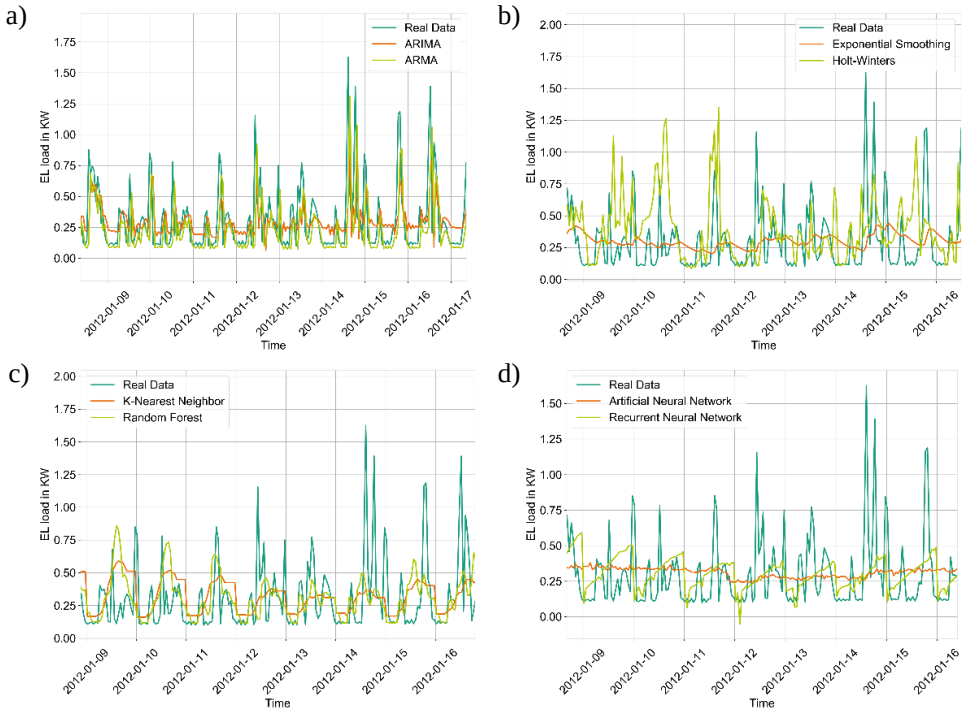

Figure 5.13: Hourly forecast and actual load comparison of electrical load data for (a) ARIMA and ARMA, (b) ES and HW, (c) KNN and RF, and (d) ANN and RNN.

The deep learning approaches can discover nonlinear relationships between the input variables and output variables, this self-learning ability produces a better performance. Although, the accuracy of the forecasts generated with artificial neural networks depends on the chosen hyperparameters. Thus, the optimization of those parameters was important to avoid both under- and overfitting. RNN has shown better performance compared to ANN in case of electrical load forecasting due to having a memory like property and the opportunity to realize broader concepts from input sequences (*Figure 5.12 (d)).* But for PV generation forecasting, RNN performed well only with larger training sample size.

The evaluation of MAE for both PV and electrical loads in *section 5.2.2* (MAE comparison) showed that the RF and KNN methods performed best for forecasting from short-term to long -term horizon ranges. For hourly prediction in case of PV generation, the ARIMA and ARMA achieved the best performance respectively, while for the electric load it was achieved by ARMA and ES method. RNN showed better performance with large training samples in case of PV generation. For electrical load the accuracy from RNN was consistently better than ANN for all training sizes and forecasting horizons as expected.

From the correlation coefficient R comparison for PV generation in *section 5.2.2* (correlation coefficient comparison), for prediction horizons larger than hourly, the KNN and RF performed better respectively. In case of weekly forecasting, RF comes first while for daily prediction the HW method achieved the best correlation. The ARMA and ARIMA exhibit the best correlation for hourly prediction. It was also observed that RNN showed relatively better correlation than ANN and with higher training sample size it was comparable to RF and KNN. The correlation for electrical load indicates somewhat a similar rank for prediction ranges from one week to three months, where the RF and KNN rank first and second respectively. For daily forecasting, HW comes first and in case of hourly forecasting, ARMA and ES achieved the best correlation.

The performance of these machine learning and deep learning approaches are expected to gradually degrade with the increase of forecasting horizon (for example: long-term forecasting). This can happen due to having uncertainty in the future exogenous information. But in this experiment, a consistent better performance was observed for longer forecasting horizons for these two types of approaches. Possible reasons might be that the forecast was generated using real energy data and the given exogenous values. This reduced the uncertainty of external forecast of exogenous variables.

6. Conclusion

Managing the best possible balance between the supply and the demand to ensure proper load management is always challenging for the traditional energy generation industry. Besides, the renewable energy industry is also growing because of the increased demand for fossil fuels and to minimize the negative impacts of climate change. Better forecasting of demand is essential for renewable power plants since the availability of energy from natural resources, like wind and solar are intermittent and often the system has limited capacity to store the energy.

Previous research studied specific forecasting methods separately to predict different datasets and analyzed their performances and limitations. This thesis was undertaken to evaluate different forecasting methods under the same system configurations and datasets, in order to determine the most accurate method for forecasting energy data. Three different categories of forecasting methods namely statistical approaches (ARMA, ARIMA, ES, HW), machine learning approaches (KNN, RF) and deep learning approaches (ANN, RNN) were tested. Therefore, the study establishes a generic research framework (the forecasting toolbox) for determining the comparison of these eight forecasting algorithms over same training and testing datasets to forecast for different timescales. Application areas such as electrical load and PV generation have been analyzed based on very short-term (hourly), short-term (daily), medium-term (weekly and monthly) and long-term (three monthly) basis prediction horizons using each of these methods. Seven different training sample sizes (50, 100, 200, 400, 800, 1600, and 2200) were employed to capture the effect of historical data size on each model. The hyperparameter optimization for each method is carried out according to each dataset. All these dependencies of forecasting aspects based on datasets are constructed, and the corresponding model was evaluated in the result discussion (*section 5.2.2).*

Statistical approaches: the ARMA or ARIMA are simple to implement and well adapted for very short-term prediction horizons. But, these two methods are unable to handle the non-linearity in the time series properly as they assume the time series as a stationary process. When fitting the ARMA model on both the data sets for short-, medium- and long-term forecasting, the results contained large errors as expected. Among the tested smoothing methods, the HW provided reasonably good forecasting for very short-time (hourly) and short-time (daily) prediction compared to the ES according to RMSE value. However, the forecasting accuracy for the HW method highly depends on the prediction horizon and the training sample sizes. All the KPIs from forecasted PV generation indicated that the HW performed better for medium-term and long-term forecasting horizons when the training sample size was larger than 800. In case of electrical load forecasting, the HW achieved good accuracy with sample sizes lager than 100. The ES method showed no dependencies like this.

As per results, both RF and KNN showed high accuracy over the short-term, medium-term and long-term forecasting horizons. The KNN performed better than the RF in most cases except for training sample smaller than 100. These two methods showed good accuracy with less computational time (as discussed *in section 5.2.2*). Therefore, while working with a large number of datasets these methods can be convenient. Both of these methods repeated good accuracy for small training samples also.

Deep learning-based approaches ANN and RNN can produce approximate complex nonlinear mappings via hidden layers from the input samples by identifying the trends in data. In this case, we observed that the forecasting result highly depends on the size of training datasets. However, the forecasts of the deep learning approaches showed good performance indicated by low errors obtained on both datasets. Whereas, RNN has shown a better performance than ANN with higher training sample sizes for PV generation forecasting, but for electrical load it provided good accuracy with smaller sample size than ANN (*section 5.2.2* RMSE comparison).

In summary, statistical approaches: ARMA and ARIMA was the best choice for hourly and daily forecasting according to the RMSE, MAE and R-value. As discussed in *section 5.2.2*, machine learning approaches (RF, KNN) repeatedly achieved higher accuracy for short-, medium- and long-term forecasting. It can be concluded that for daily, weekly, monthly or longer forecasting timescale RF and KNN is the best option for both PV generation and electrical load usage. For electrical load forecasting, ARIMA, ANN, RNN and HW (only for higher training sample sizes) achieved accuracy closer to that of RF and KNN for similar forecasting timescales. While KNN offers such accurate predictions with minimal computational time for all cases, the RF method needs slightly higher computation time compared to KNN. The most computationally expensive methods from the experiment are the RNN and ANN (in *section 5.2.1*).

6.1. Future Work

In this study, we have used eight promising methods from three different approaches to forecast energy data (PV generation and electrical load). There are other methods such as linear regression, support vector regression or gaussian process regression etc. which can additionally be included as alternate techniques in the investigation. If time and resources permit, more methods and approaches could be further investigated.

Forecasting for a very short-time horizon is another intriguing possibility. The minimum time interval in dataset was 60 minutes. Therfore, this study is limited to 1h interval for very short-term prediction. Datasets of shorter time interval like 15 or 30 minutes will present the opportunity to predict for 15 or 30 minutes ahead also. This seems to be interesting future work. Additionally, the possibility of forecasting the peak loads can be achieved by changing the current procedure.

Modifying the parameters by using some dynamic optimization function, such as using a genetic algorithm for determining the hyperparameter, could be interesting to get a better optimization of the hyperparameter. Also, it was noticed that a reasonably good forecasting accuracy has been achieved by the smallest training size (50). Furthermore, the impact with even smaller training sample sizes can be investigated as a future research.

For implementation purposes and time limitation, two different datasets - PV generation data and electrical load of household consumptions for a year have been used. It would have been better if more of these datasets could be used because it is often the case that more data will significantly improve the performance of any forecasting methods. Initially it was planned to investigate space heating dataset as well in this study. But as mentioned in the performance section, due to long computation time, we excluded space heating data out of the timeframe of this research. If multiple case study objects could have applied, then the external validity of the findings of this research work might have increased. Apart from that, the prediction of electric prices of different energy markets is interesting for both the producers and consumers around the world [21]. Therefore, making a comparative analysis of the mentioned forecasting methods for electrical energy pricing with the historical data could be another interesting forecasting application to investigate [78]. Other applications to investigate with time series forecasting approaches could be the forecasting of stock exchanges.

Bibliography

[1] R. Bonetto and M. Rossi, “Machine Learning Approaches to Energy Consumption Forecasting in Households,” *CoRR*, vol. abs/1706.0, pp. 6–9, 2017.

[2] Eurostat, *The EU in the world*. 2721 Luxembourg, 2018.

[3] F. ISE, “ENERGY CHARTS - Stromproduktion in Deutschland.” .

[4] A. Laouafi and M. Mordjaoui, “One-Hour Ahead Electric Load and Wind-Solar Power Generation Forecasting using Artificial Neural Network,” *IREC2015 The Sixth International Renewable Energy Congress*, pp. 1–6, 2015.

[5] D. W. Bunn, “Short-Term Forecasting: A Review of Procedures in the Electricity Supply Industry,” *Journal of the Operational Research Society*, vol. 33, no. 6, pp. 533–545, Jun. 1982.

[6] H. Cho, Y. Goude, X. Brossat, and Q. Yao, “Modeling and Forecasting Daily Electricity Load Curves: A Hybrid Approach,” *Journal of the American Statistical Association*, vol. 108, no. 501, pp. 7–21, 2013.

[7] T. Haida and S. Muto, “Regression based peak load forecasting using a transformation technique,” *IEEE Transactions on Power Systems*, vol. 9, no. 4, pp. 1788–1794, Nov. 1994.

[8] X. Sun, X. Wang, J. Wu, and Y. Liu, “Hierarchical sparse learning for load forecasting in cyber-physical energy systems,” in *2013 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, 2013, pp. 533–538.

[9] G. Bruni, S. Cordiner, V. Mulone, V. Rocco, and F. Spagnolo, “A study on the energy management in domestic micro-grids based on Model Predictive Control strategies,” *Energy Conversion and Management*, vol. 102, pp. 50–58, 2015.

[10] G. T. Heinemann, D. A. Nordmian, and E. C. Plant, “The Relationship Between Summer Weather and Summer Loads - A Regression Analysis,” *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-85, no. 11, pp. 1144–1154, Nov. 1966.

[11] A. Lahouar and J. Ben Hadj Slama, “Day-ahead load forecast using random forest and expert input selection,” *Energy Conversion and Management*, vol. 103, pp. 1040–1051, 2015.

[12] S. S. Photovoltaic *et al.*, “Day-Ahead Power Output Forecasting for Electricity Generators,” *IEEE Transactions on Smart Grid*, vol. 6, no. 5, pp. 2253–2262, 2015.

[13] C. S. Ioakimidis and S. Lopez, “Solar Production Forecasting Based on Irradiance Forecasting Using Artificial Neural Networks,” *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*, pp. 8121–8126, 2013.

[14] J. W. Taylor, L. M. De Menezes, and P. E. Mcsharry, “A comparison of univariate methods for forecasting electricity demand up to a day ahead,” vol. 22, pp. 1–16, 2006.

[15] J. S. Armstrong and F. Collopy, “Error measures for generalizing about forecasting methods: Empirical comparisons,” *International Journal of Forecasting*, vol. 8, no. 1, pp. 69–80, 1992.

[16] A. Veit, C. Goebel, R. Tidke, C. Doblander, and H.-A. Jacobsen, “Household Electricity Demand Forecasting - Benchmarking State-of-the-Art Methods,” *CoRR*, vol. abs/1404.0200, 2014.

[17] K. Metaxiotis, A. Kagiannas, D. Askounis, and J. Psarras, “Artificial intelligence in

short term electric load forecasting: A state-of-the-art survey for the researcher," *Energy Conversion and Management*, vol. 44, pp. 1525–1534, 2003.

[18] M. Ghiassi, D. K. Zimbra, and H. Saidane, "Medium term system load forecasting with a dynamic artificial neural network model," *Electric Power Systems Research*, vol. 76, no. 5, pp. 302–316, 2006.

[19] F. Kaytez, M. C. Taplamacioglu, E. Cam, and F. Hardalac, "Forecasting electricity consumption: A comparison of regression analysis, neural networks and least squares support vector machines," Elsevier Ltd, 2015.

[20] H. zhou Liu, "Load forecasting based on weighted kernel partial least squares algorithm in smart grid," *IET Conference Proceedings*, p. 2.47-2.47(1), Jan. 2012.

[21] R. Weron, *Modeling and Forecasting Electricity Loads and Prices: A Statistical Approach*. Hugo Steinhaus Center, Wroclaw University of Technology, 2006.

[22] J. Zheng, C. Xu, Z. Zhang, and X. Li, "Electric Load Forecasting in Smart Grid Using Long-Short-Term-Memory based Recurrent Neural Network Electric Load Forecasting in Smart Grid Using Long-Short-Term-Memory based Recurrent Neural Network," *2017 51st Annual Conference on Information Sciences and Systems (CISS)*, no. January, pp. 1–6, 2017.

[23] Y. Jiang, X. Chen, K. Yu, and Y. Liao, "Short-term wind power forecasting using hybrid method based on enhanced boosting algorithm," *Journal of Modern Power Systems and Clean Energy*, vol. 5, no. 1, pp. 126–133, 2017.

[24] L. Gelažanskas and K. Gamage, "Forecasting HotWater Consumption in Residential Houses," *Energies*, vol. 8, no. 11, pp. 12702–12717, 2015.

[25] C.-M. Lee and C.-N. Ko, "Short-term load forecasting using lifting scheme and ARIMA models," *Expert Systems with Applications*, vol. 38, pp. 5902–5911, 2011.

[26] V. Mayrink and H. S. Hippert, "A hybrid method using Exponential Smoothing and Gradient Boosting for electrical short-term load forecasting," in *2016 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, 2016, pp. 1–6.

[27] J. W. Taylor, "An evaluation of methods for very short-term load forecasting using minute-by-minute British data," *International Journal of Forecasting*, vol. 24, no. 4, pp. 645–658, 2008.

[28] G. A. N. Mbamalu and M. E. El-Hawary, "Load forecasting via suboptimal seasonal autoregressive models and iteratively reweighted least squares estimation," *IEEE Transactions on Power Systems*, vol. 8, no. 1, pp. 343–348, Feb. 1993.

[29] B. Shah and B. H. Trivedi, "Artificial Neural Network based Intrusion Detection System : A Survey," *International Journal of Computer Applications*, vol. 39, no. 6, pp. 13–18, 2012.

[30] C. Deb, L. S. Eang, J. Yang, and M. Santamouris, "Forecasting diurnal cooling energy load for institutional buildings using Artificial Neural Networks," *Energy and Buildings*, vol. 121, pp. 284–297, 2016.

[31] D. C. Park, R. J. Marks, L. E. Atlas, and M. J. Damborg, "Electric Load Forecasting Using An Artificial Neural Network," *IEEE Transactions on Power Systems*, vol. 6, no. 2, pp. 442–449, 1991.

[32] H. S. Hippert and J. W. Taylor, "An Evaluation of Bayesian Techniques for Controlling Model Complexity and Selecting Inputs in a Neural Network for Short-term Load Forecasting," *Neural Networks*, vol. 23, no. 3, pp. 386–395, Apr. 2010.

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

[34] F. H. Al-Qahtani and S. F. Crone, “Multivariate k-nearest neighbour regression for time series data - A novel algorithm for forecasting UK electricity demand,” *Proceedings of the International Joint Conference on Neural Networks*, 2013.

[35] R. C. Deo, X. Wen, and F. Qi, “A wavelet-coupled support vector machine model for forecasting global incident solar radiation using limited meteorological dataset,” *Applied Energy*, vol. 168, pp. 568–593, 2016.

[36] A. Setiawan, I. Koprinska, and V. G. Agelidis, “Very short-term electricity load demand forecasting using support vector regression,” *2009 International Joint Conference on Neural Networks*, pp. 2888–2894, 2009.

[37] G. Dudek, “Short-Term Load Forecasting using Random Forests,” 2011.

[38] J. Antonanzas, N. Osorio, R. Escobar, R. Urraca, F. J. Ascacibar, and F. Antonanzas, “Review of photovoltaic power forecasting,” *Solar Energy*, vol. 136, 2016.

[39] P. Kuo, “A High Precision Artificial Neural Networks Model for Short-Term Energy Load Forecasting,” *Energies*, vol. 11, no. January, pp. 1–13, 2018.

[40] T. Hong, M. Gui, M. E. Baran, S. Member, and H. L. Willis, “Modeling and Forecasting Hourly Electric Load by Multiple Linear Regression with Interactions,” *IEEE PES General Meeting*, pp. 1–8, 2010.

[41] B. Chen, M. Chang, and C. Lin, “Load Forecasting Using Support Vector Machines : A Study on EUNITE Competition 2001,” *IEEE Transactions on Power Systems*, vol. 19, no. 4, pp. 1821–1830, 2004.

[42] C. Xia, J. Wang, and K. Rafferty, “Short, medium and long term load forecasting model and virtual load forecaster based on radial basis function neural networks,” *International Journal of Electrical Power & Energy Systems*, vol. 32, pp. 743–750, 2010.

[43] R. J. Hyndman and G. Athanasopoulos, *Forecasting : Principles and Practice*. OTexts, 2018.

[44] I. Moghram and S. Rahman, “Analysis and evaluation of five short-term load forecasting techniques,” *IEEE Transactions on Power Systems*, vol. 4, no. 4, pp. 1484–1491, Nov. 1989.

[45] J. D. Hamilton, *Time series analysis*. Princeton, N.J. and Chichester: Princeton University Press, 1994.

[46] P. J. Brockwell and R. A. Davis, *Introduction to time series and forecasting*. New York: Springer, 2002.

[47] R. H. Shumway and D. S. Stoffer, *Time series analysis and its applications: With R examples*, 3rd ed. New York: Springer, 2011.

[48] NIST/SEMATECH, “e-Handbook of Statistical Methods.” 2013.

[49] G. E. P. Box and G. Jenkins, *Time Series Analysis, Forecasting and Control*. San Francisco, CA, USA: Holden-Day, Inc., 1990.

[50] S. Bisgaard and M. Kulahci, *Time series analysis and forecasting by example*. John Wiley & Sons, 2011.

[51] X. Zhang, Y. Liu, M. Yang, T. Zhang, A. A. Young, and X. Li, “Comparative Study of Four Time Series Methods in Forecasting Typhoid Fever Incidence in China,” vol. 8, no. 5, 2013.

[52] C. Chatfield, *The analysis of time series: an introduction*, 6th ed. Florida, US: CRC Press, 2004.

[53] G. E. P. Box and G. M. Jenkins, “Times series Analysis Forecasting and Control.

Holden-Day San Francisco," 1970.

[54] S. G. Makridakis, S. C. Wheelwright, and R. J. Hyndman, *Forecasting : methods and applications*. John wiley & sons, 2008.

[55] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks, 1984.

[56] T. Hastie, R. Tibshirani, and J. J. H. Friedman, "The Elements of Statistical Learning," 2009.

[57] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[58] M. Immitzer, C. Atzberger, and T. Koukal, "Tree Species Classification with Random Forest Using Very High Spatial Resolution 8-Band WorldView-2 Satellite Data," *Remote Sensing*, vol. 4, no. 9, pp. 2661–2693, 2012.

[59] A. Liaw and M. Wiener, "Classification and Regression by RandomForest," *Forest*, vol. 23, 2001.

[60] E. Fix and J. L. Hodges, "Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties," *International Statistical Review / Revue Internationale de Statistique*, vol. 57, no. 3, pp. 238–247, 1989.

[61] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, Jan. 1967.

[62] M. Thanh Noi, Phan and Kappas, "Comparison of Random Forest, k-Nearest Neighbor, and Support Vector Machine Classifiers for Land Cover Classification Using Sentinel-2 Imagery," *Sensors*, vol. 18, no. 1, 2018.

[63] V. K. Pathirana, "Nearest Neighbor Foreign Exchange Rate Forecasting with Mahalanobis Distance," 2015.

[64] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, Dec. 1943.

[65] A. Terminal and S. Cell, "CHAPTER – 3 Back Propagation Neural Network (BPNN)," pp. 18–23.

[66] G. P. Zhang, "A Neural Network Ensemble Method with Jittered Training Data for Time Series Forecasting," *Information Sciences*, vol. 177, no. 23, pp. 5329–5346, Dec. 2007.

[67] J. Kamruzzaman, R. K. Begg, and R. A. Sarker, *Artificial Neural Networks in Finance and Manufacturing*. Hershey, PA, USA: IGI Global, 2006.

[68] R. Adhikari and R. K. Agrawal, "An Introductory Study on Time Series Modeling and Forecasting," 2013.

[69] J. Kihoro and C. Wafula, "Seasonal time series forecasting: A comparative study of ARIMA and ANN models," *African Journal of Science and Technology*, vol. 5, 2006.

[70] J. Bi, T. Xiong, S. Yu, M. Dundar, and R. Rao, "Machine Learning and Knowledge Discovery in Databases," *Machine Learning and Knowledge Discovery in Databases*, vol. 5211, pp. 117–132, 2014.

[71] G. Kariniotakis, G. Stavrakakis, and E. F. Nogaret, "Wind power forecasting using advanced neural networks models," *Energy Conversion, IEEE Transactions on*, vol. 11, pp. 762–767, 1996.

[72] A. Pradhan, "Recurrent Neural Networks." [Online]. Available: https://medium.com/lingvo-masino/introduction-to-recurrent-neural-network-d77a3fe2c56c. [Accessed: 24-Feb-2019].

[73] C. E. Rasmussen, C. K. I. Williams, G. Processes, M. I. T. Press, and M. I. Jordan, *Gaussian Processes for Machine Learning*. 2006.

[74] J. S. S. Armstrong and F. Collopy, "Error measures for generalizing about forecasting methods: Empirical comparisons," *International Journal of Forecasting*, vol. 8, no. 1, pp. 69–80, 1992.

[75] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," vol. 22, pp. 679–688, 2006.

[76] C. Chen, J. Twycross, and J. M. Garibaldi, "A new accuracy measure based on bounded relative error for time series forecasting," *PLOS ONE*, vol. 12, no. 3, pp. 1–23, 2017.

[77] Q. Feng, J. Liu, and J. Gong, "UAV Remote Sensing for Urban Vegetation Mapping Using Random Forest and Texture Analysis," *Remote Sensing*, vol. 7, no. 1, pp. 1074–1094, 2015.

[78] A. Daraeepour and N. Amjady, "Mixed price and load forecasting of electricity markets by a new iterative prediction method," *Electric Power Systems Research*, vol. 79, pp. 1329–1336, 2009.